in silico Plants Vol. 4, No. 1, pp. 1–19
https://doi.org/10.1093/insilicoplants/diab039
Advance Access publication 10 December 2021
Original Research

in silico Plants
OXFORD

L-system models for image-based phenomics: case studies of maize and canola

Mikolaj Cieslak[1,*], Nazifa Khan[2], Pascal Ferraro[1], Raju Soolanayakanahally[3], Stephen J. Robinson[3], Isobel Parkin[3], Ian McQuillan[2] and Przemyslaw Prusinkiewicz[1,*]

[1]Department of Computer Science, University of Calgary, Calgary T2N 1N4, Canada
[2]Department of Computer Science, University of Saskatchewan, Saskatoon S7N 5C9, Canada
[3]Agriculture and Agri-Food Canada, Saskatoon S7N OX2, Canada
*Corresponding authors' e-mail addresses: msciesla@ucalgary.ca; pwp@ucalgary.ca

Handling Editor: Steve Long

Citation: Cieslak M, Khan N, Ferraro P, Soolanayakanahally R, Robinson SJ, Parkin I, McQuillan I, Prusinkiewicz P. 2021. L-system models for image-based phenomics: case studies of maize and canola. *In Silico Plants* **2021**: diab039; doi: 10.1093/insilicoplants/diab039

ABSTRACT

Artificial neural networks that recognize and quantify relevant aspects of crop plants show great promise in image-based phenomics, but their training requires many annotated images. The acquisition of these images is comparatively simple, but their manual annotation is time-consuming. Realistic plant models, which can be annotated automatically, thus present an attractive alternative to real plant images for training purposes. Here we show how such models can be constructed and calibrated quickly, using maize and canola as case studies.

KEYWORDS: *Brassica napus*; deep learning; L-system; model calibration; plant phenotyping; *Zea mays*.

1. INTRODUCTION

Forward phenomics relies on screening collections of plants to identify those with desirable traits (Furbank and Tester 2011). As traditional plant-measurement techniques are impractical for screening large numbers of plants, computer vision (image-based phenotyping) has become an alternative (Fahlgren *et al.* 2015; Tardieu *et al.* 2017; Artzet *et al.* 2019). Vision algorithms employing artificial neural networks and deep learning show particular promise (Singh *et al.* 2018; Jiang and Li 2020), but their training requires large sets of annotated images, which are difficult to obtain. Plant models that can be used to automatically generate large sets of realistic annotated images for training purposes alleviate this problem (Ubbens *et al.* 2018; Miao *et al.* 2019; Jiang and Li 2020). The question is how to create models that are faithful to reality quickly.

To some extent, this question is related to the generation of plant models given their images or point-cloud data. The literature on this topic is rich (reviewed, for example, by Guo *et al.* (2020)), but until now has been focused on trees. Attention has been given to the approximation of the overall tree silhouettes, rather than the recreation of the detailed features dominating the appearance of herbaceous plants, such as the progression of branch lengths or leaf shapes along the plant axes. Moreover, the methods devised so far consider plants as static structures and do not capture their development.

In this paper, we present an interactive method for creating and calibrating developmental plant models expressed using L-systems (Lindenmayer 1968a, b; 1971; Prusinkiewicz and Lindenmayer 1990; Prusinkiewicz 2004a; Prusinkiewicz *et al.* 2018). The models operate at the level of modular plant architecture, i.e. plant shoots are specified as growing assemblies of units (modules), produced sequentially by the shoot apices (Room *et al.* 1996). The basic units are phytomers: stem segments or internodes associated with one or more (in the case of multijugate phyllotaxis) leaves and lateral buds. The buds may develop into next-order shoots or flowers, remain dormant or abort. In principle, the form of each module in the developing plant can be characterized using functions of the module's position within the branching structure and the plant age. Such characterization may require several functions per individual module, which is not a practical solution, especially in the case of branched structures with many modules. Fortunately, repetitions, similarities and invariants present in the plant structure (Prusinkiewicz 2004b; Ferraro *et al.* 2005) can

be exploited to reduce the complexity of plant description while leaving enough flexibility to calibrate models to target phenotypes. Our method pursues this idea while taking advantage of three assumptions:

1) Mature phytomers of the same type (e.g. in the vegetative or reproductive part of the shoot) share common characteristics. Such phytomers can be represented by a common parametrized submodel, in which individual parameters (e.g. the length of the internode and the size of the associated leaf) depend on the module's position within the branching structure (Prusinkiewicz *et al.* 2001).
2) Modules of the same type follow similar developmental trajectories. This assumption is justified by experimental data depicting the size of consecutively produced plant organs as a family of sigmoidal curves offset in time by an approximately constant interval—the plastochron (Erickson and Michelini 1957; Prusinkiewicz *et al.* 1994; Mündermann 2003; Mündermann *et al.* 2005; Jullien et al. 2011, 2012). It makes it possible to characterize the growth of related modules using a common function of time.
3) Geometric attributes of organs, for example the length and width of leaves, may be linked allometrically (Huxley 1925; Huxley and Teissier 1936; Richards and Kavanagh 1945; Niklas 1994; Fournier and Andrieu 1998; Mündermann *et al.* 2005), which further reduces the number of parameters that need to be controlled separately.

We describe our method in the context of the Virtual Laboratory (vlab) plant modelling environment (Mercer *et al.* 1990; Federl and Prusinkiewicz 1999; Prusinkiewicz 2004a; http://www.algorithmicbotany.org/virtual_laboratory/). The modelling process consists of two phases: the qualitative (topological) specification of the developing plant architecture, and model calibration. The plant architecture is specified using the L+C modelling language (Karwowski and Prusinkiewicz 2003; Prusinkiewicz *et al.* 2007); for a tutorial introduction, see Prusinkiewicz *et al.* (2018). L+C combines concepts of L-systems (Lindenmayer 1968a, b; 1971) into C++, which allows for fast execution and interactive manipulation of developmental models required by the presented method. Plant development is described in terms of three types of rules that have the same syntax, but play different roles in the simulation. Ordinary parametric L-system productions (Prusinkiewicz and Hanan 1990; Prusinkiewicz and Lindenmayer 1990) advance the state of plant components (modules) over time. Decomposition rules (Prusinkiewicz et al. 2000, 2001; Karwowski and Prusinkiewicz 2003) define the structure of compound modules in terms of their constituent parts; for example, a phytomer may be decomposed into an internode, a leaf and a lateral bud. In programming practice it is also convenient to consider the production of a phytomer by an apex as a decomposition of an older apex into a younger apex and a phytomer. Interpretation rules (Kurth 1994) characterize geometry and visual attributes of the modules in terms of turtle geometry (Abelson and DiSessa 1986; Prusinkiewicz 1986; Prusinkiewicz and Lindenmayer 1990). The flow of simulation employing these rules is depicted in Fig. 1. The meaning of the code presented in this paper is explained through in-line comments and descriptions in the text without assuming familiarity with L+C.

A key component of the method is interactive model calibration. The standard method for calibrating plant models is to measure relevant features of plant architecture and fit approximating functions to the data (Prusinkiewicz 1998). Well-established plant-measurement methods make use of 3D digitizers, which return coordinates of plant locations selected with a handheld probe by a human operator (Moulia and Sinoquet 1993; Hanan and Room 1997). With the assistance of specialized software, the operator also organizes the data according to the topological structure of the plant and labels it (Godin et al. 1997, 1999; Hanan and Room 1997), thus effectively annotating it. Unfortunately, for many features—such as counting leaves or branches—plant measurement is significantly more time-consuming than the annotation alone. Consequently, we propose an alternative calibration method, based on aligning the model with a reference plant using a graphical interface. The idea of extensively using graphical interfaces to model plants was introduced by Lintermann and Deussen (1999), and incorporated into L-systems by Galbraith *et al.* (1999), Prusinkiewicz *et al.* (2001) and Mündermann (2003). An alternative approach adapted to image-based modelling was presented by Quan *et al.* (2006). Here we build upon these ideas to provide a practical guide to the fast construction and calibration of both static and developmental, visually realistic plant models. In addition to the L-system-based plant simulator, lpfg, the method relies on the following vlab tools (http://algorithmicbotany.org/virtual_laboratory/):

- user-configurable control panels, which facilitate the manipulation of numerical parameters during the calibration process;
- graphical editors of functions and curves, which can be incorporated for various purposes into the models, e.g. to control rates of development or the shape of leaves; and
- a timeline editor, which facilitates positioning of developmental events on the time axis.

During model calibration, quantitative model attributes are manipulated by the modeller to match images representing sample plants in a specific developmental stage or in a sequence of stages. The plant simulator, and function and shape editors, can display the reference plants

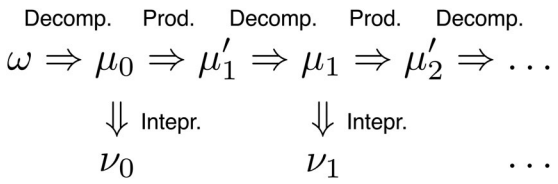

Figure 1. The flow of simulation using L-system models implemented in vlab. Starting with the initial state ω, the model states $\mu_0, \mu'_1, \mu_1, \mu'_2, \mu_2, \ldots$ result from the application of interleaved production and decomposition rules. The interpretation rules map the states following the decomposition into states $\nu_0, \nu_1, \ldots$ that specify details of the module appearance, needed for their visualization.

as background images. This facilitates interaction and comparisons of models in real time. We present the modelling process using the vegetative development of maize (*Zea mays*) and both the vegetative and flowering development of canola (*Brassica napus*) as examples. They represent a progression from the simple architecture of maize, with all leaves arranged along a single growth axis, to a more complex example of branching architecture represented by canola.

Randomization of parameter values makes it possible to generate large sets of related phenotypes as needed for training artificial neural networks. For uncorrelated values, such randomization is implemented by replacing fixed model parameters with calls to a random number generator. The variance values could be inferred rigorously by modelling a significant number of sample plants and quantifying the differences between the models, but in practice we estimate these values interactively, based on the visual plausibility of the generated phenotypes. The construction of more complex stochastic models with correlated variables is outside the scope of this paper.

2. THE MAIZE MODEL

Maize is an annual grass with an aerial vegetative structure consisting of a monopodial sequence of phytomers. Leaves are arranged into a distichous phyllotactic pattern, with consecutive leaves alternating between opposite sides of the stalk. Each leaf is composed of a sheath, gripping the stalk, and a long blade bending away from it. This simple architecture makes maize a convenient plant for developing algorithms related to phenotyping (Cabrera-Bosquet *et al.* 2016; Brichet *et al.* 2017; Das Choudhury *et al.* 2018; Khan *et al.* 2020). Several models of maize, including ADEL-maize (Fournier and Andrieu 1998, 1999), GRAAL (Drouet and Pagès 2003) and GREENLAB for Maize (Guo *et al.* 2006; Ma *et al.* 2008), already exist. These models combine architectural constraints, physiological processes and environmental conditions to simulate the mechanisms that control the development of the whole plant. They were parameterized and calibrated to detailed measurements from field experiments. It is not clear, however, how suitable they are to image-based calibration, where diverse phenotypes have to be reproduced quickly. We show how parametric L-systems can be used to create a simple, descriptive model of vegetative development in maize that is easy to calibrate.

2.1 Model construction

The L+C modules representing the main components of the plant are: an apex `A`, an internode `I` and a leaf `L`. Each module is characterized by two parameters: its chronological `age` and its phytomer number `n` counted from the base of the stem. The age of each module is measured from its time of creation, but the age of an apex is decremented each time it produces a phytomer. The phytomer number provides an input to graphically defined functions that specify the size of organs based on their position. We assume that, at the beginning of the simulation, the apex has both the `age` and phytomer count `n` set to zero.

To simulate development in continuous time, a global variable `t` representing plant age (from the time of seed germination) is incremented with a small timestep, `dt`. The age of each module is incremented accordingly. For example, the age of the apex is updated using production:

```
A(age,n) : produce A(age+dt,n);
```

Similar productions are applied to the internode and leaf modules.

The production of a new internode and leaf by the apex reaching the threshold age defined by the constant `PLASTOCHRON` is described by the following decomposition rule:

```
A(age,n) : {
     if (age >= PLASTOCHRON) {
          age = age-PLASTOCHRON;    // decrement apex age
          produce I(age,n)          // create new internode
                  SB                // start branch
                  // branching angle
                  Down(BrAngle*br_target_angle(n)*br_angle(age))
                  L(age,n)          // create leaf at position n
                  EB                // end branch
                  RollL(180)        // distichous phyllotaxis
                  A(age,n+1);       // recreate apex
     }
}
```

The `PLASTOCHRON` value depends on a number of factors including the genotype and temperature (Fournier and Andrieu 1998) and was estimated as 3 days for the conditions of the data set (see the section on model calibration for the tuning and randomization of parameter values). The new internode and leaf module are initialized with the age and phytomer number determined by the apex. Consecutive leaves and internodes thus have increasing numbers, which characterize their position in the shoot. The branching angle between the internode and leaf blade is controlled by graphically defined functions `br_angle(age)` and `br_target_angle(n)`. The first function specifies the increase of the branching angle over time, and the second function captures the maximum value of this angle, depending on the position of the leaf on the stem. The product of these functions is multiplied by a scaling factor, `BrAngle`. To simulate the distichous phyllotaxis characteristic of the vegetative shoot of maize, each successive leaf is rotated 180° around the stem, compared to its predecessor.

An internode is modelled as a cylinder using the following interpretation rule:

```
I(age,n): {
    // calculate internode length and width
    float len = IntLen*internode_target_len(n)*
                internode_length(age);
    float wid = powf(len,ExpIntRad)*internode_width(n);
    // draw a cylinder
    produce SetWidth(wid) F(len);
}
```

The target length and radius are determined by the graphically defined functions of phytomer position in the shoot: `internode_target_len(n)` and `internode_width(n)`. The target length is scaled by the factor `IntLen`. The increase in length over the course of development is characterized by the graphically defined function of time `internode_length(age)`. The increase in the internode radius over time is calculated by exploiting an allometric relation between the internode radius and length (Fournier and Andrieu 1998; Mündermann *et al.* 2005), and controlled by the exponent `ExpIntRad`.

A leaf is modelled by another interpretation rule:

```
L(age,n): {
     // calculate leaf length
     float len = LeafLen*leaf_target_len(n)*leaf_length(age);
     // set the initial leaf width
     nproduce SetWidth(powf(len,ExpLeafWid)*leaf_width(0));
     // start drawing the generalized cylinder
     nproduce StartGC;
     for (float x = 0; x < 1; x += DX) {
          // determine current leaf cross-section
          nproduce BlendedContour(0,1,x);
          // generate leaf segment
          nproduce F(len*DX);
          // set width and orientation of next segment
          nproduce SetWidth(powf(len,ExpLeafWid)*leaf_width(x))
                   Down(leaf_bend(x)*len*DX)
                   RollL(leaf_twist(x)*len*DX);
     }
     nproduce F(len*(1-x)); // generate last segment
     produce EndGC;         // end the generalized cylinder
}
```

Leaf length `len` is determined by the product of graphically defined functions `leaf_length(age)` and `leaf_target_len(n)`, multiplied by the scaling factor `LeafLen`, in the same manner as the length of the internodes. The shape of the leaves is, however, more complicated than the shape of internodes. To capture it, each leaf is modelled as a generalized cylinder (Lintermann and Deussen 1999; Prusinkiewicz *et al.* 2001) divided into `1/DX` segments of length `len*DX`. The `BlendedContour` module defines the leaf cross-section as a function of its (normalized) position x along the leaf axis, by interpolating between a closed cylinder representing the sheath (predefined contour 0), and an open shape representing the blade (predefined contour 1, Fig. 2). The size of each cross-section—i.e. the local width of the leaf—is determined by the graphically defined function `leaf_width(x)` and an allometric relation to length (which is controlled by exponent `ExpLeafWid`). The shape of the leaf is further controlled by two graphically defined functions, `leaf_bend(x)` and `leaf_twist(x)`, which bend and twist the leaf along the midrib, respectively. Moreover, leaf shape is affected by drooping due to gravity. The magnitude of this drooping depends on the length of the leaf: longer leaves curve more than shorter ones. To model this effect, we multiply the values of functions `leaf_bend(x)` and `leaf_twist(x)` by the current length of the segments into which the leaf is divided. To introduce slight differences between leaf shapes, these angles are additionally multiplied by a normally distributed random number with mean 1 and standard deviation 0.1 (not shown in the above code).

2.2 Model calibration

A crucial point of model construction is parameter fitting, which, in the case of plant models, tends to be significantly more time-consuming than specifying the L-system rules capturing qualitative aspects of the

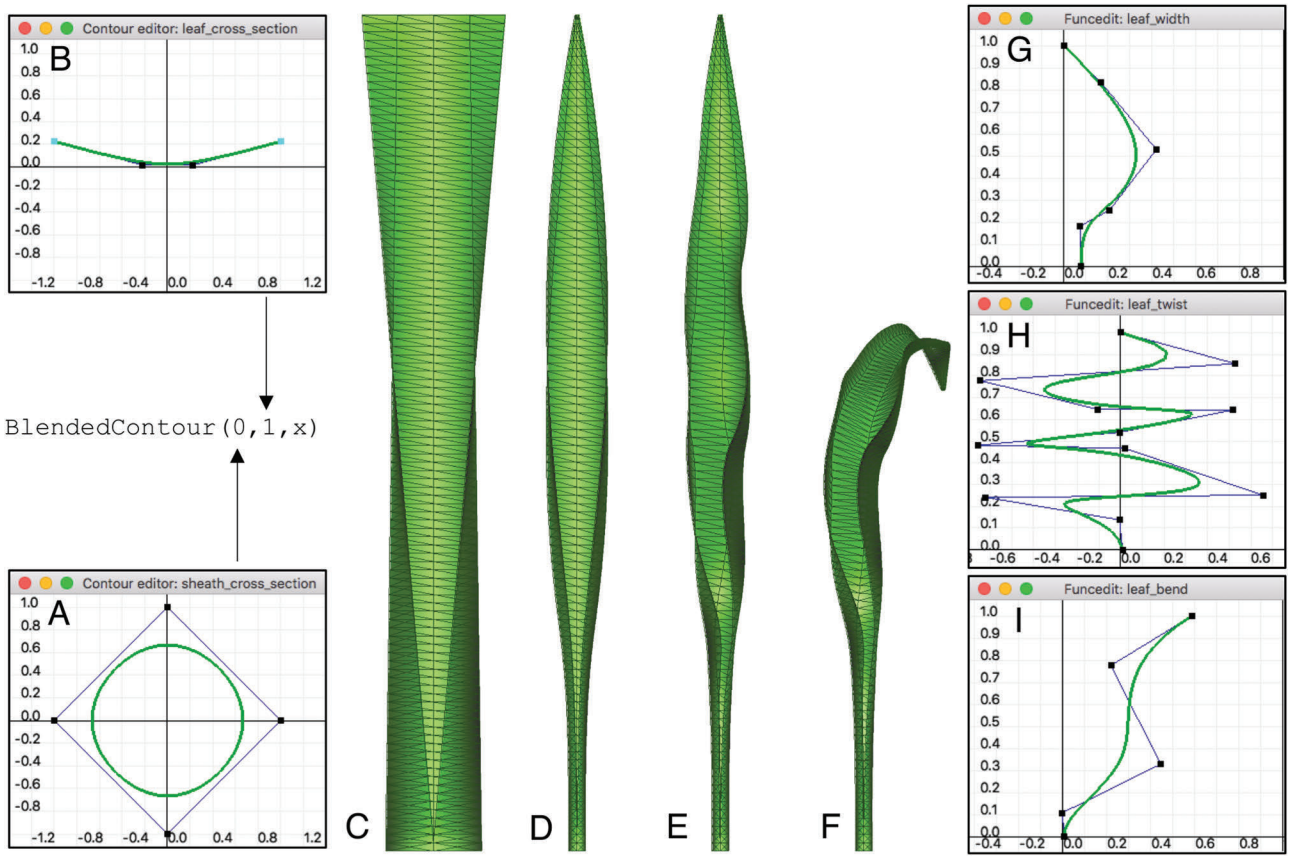

Figure 2. Modelling a single maize leaf as a generalized cylinder. The effect of each modelling operation is shown separately to illustrate the process. The BlendedContour(0,1,x) module linearly interpolates from the closed contour (index 0) (A) to the open contour (index 1) (B), producing a cylinder with a smoothly changing cross-section (C). The cylinder width is then scaled by the leaf_width(x) function (G), producing an intermediate shape (D). The leaf_twist(x) and leaf_bend(x) functions (H, I) change the orientation of the turtle as the cylinder is drawn, twisting the leaf in the *x*, *y* plane (E) and bending it out of the plane (F). Plots (A, B) are snapshots of the graphical contour editor, and plots (G–I) are snapshots of the function editor. The modeller defines the cross-sections and the functions by interactively manipulating control points of a uniform cubic B-spline (Bartels *et al.* 1995). The function plots are rotated such that the *x*-axis, representing position along the midrib, is vertical, and the *y*-axis, representing function values, is horizontal. The function argument is normalized such that the base of the leaf is at $x = 0$ and its tip is at $x = 1$. The displayed function values are scaled when used in the models, as described in the text.

developing plant architecture. To calibrate the maize model, we used the publicly available University of Nebraska–Lincoln Component Plant Phenotyping Dataset (UNL-CPPD) (https://plantvision.unl.edu/dataset). It consists of 13 maize genotypes, grown under the same environmental conditions, and photographed once per day using a Lemnatec Scanalyzer 3D high-throughput system at the UNL phenotyping facility. Twelve plants were photographed over 27 days, and one over 26 days. Each plant was photographed from the front and the side (orthogonal views), for a total of 700 images. All images were of the vegetative stage only. Das Choudhury *et al.* (2018) provide a more detailed description of the data set, including the experiment set-up. The data set also includes annotated images marking the leaves in order of emergence; however, only the visible leaves in each image were marked, so their number could differ between views of the same plant (Khan *et al.* 2020).

Calibration begins with a default branching structure superimposed on a reference photograph of the plant in the final stage of development depicted in the data (see Fig. 3 and **Supporting Information—Movie 1**). The number of phytomers is specified interactively using a virtual control panel (not shown, see Fig. 10E for an example). The modeller then interactively manipulates the graphically defined functions that determine the target length and width of internodes, the target length of leaves and the target leaf branching angles. The leaf shape functions are the same for all leaves but are scaled by length, resulting in different size and curvature of individual leaves.

Next, the model is calibrated over developmental time. To this end, the modeller is presented with an array of snapshots representing a sequence of developmental stages (Fig. 4, see also **Supporting Information—Movie 1**). The periods of internode elongation, leaf growth and progression of the branching angle, measured from the time of the phytomer production by the apex, are defined by adjusting the length of the corresponding lines in a timeline editor. Within each period, the rates of change are controlled by graphically defined functions associated with each parameter. Immediate visual feedback makes it possible to interactively align simulations to images in less than 10 min.

Once calibrated, the model can be used to generate multiple images of synthetic maize plants by randomizing the parameters using normally distributed random variables (Fig. 5). The models can be automatically annotated with any needed data, such as the lengths of internodes and positions of the leaf tips. Figure 6 provides an example comparing the annotation of synthetic and real images of plants. The annotation of real plants was included as part of the UNL-CPPD maize data set. The leaf labelled zero in the synthetic image was overlooked in the real plant (although it was labelled in earlier stages of development).

The maize example shows that visual calibration of L-system-based models is fast and effective for relatively simple branching structures. The canola model, presented next, extends this idea to structures with higher order branching.

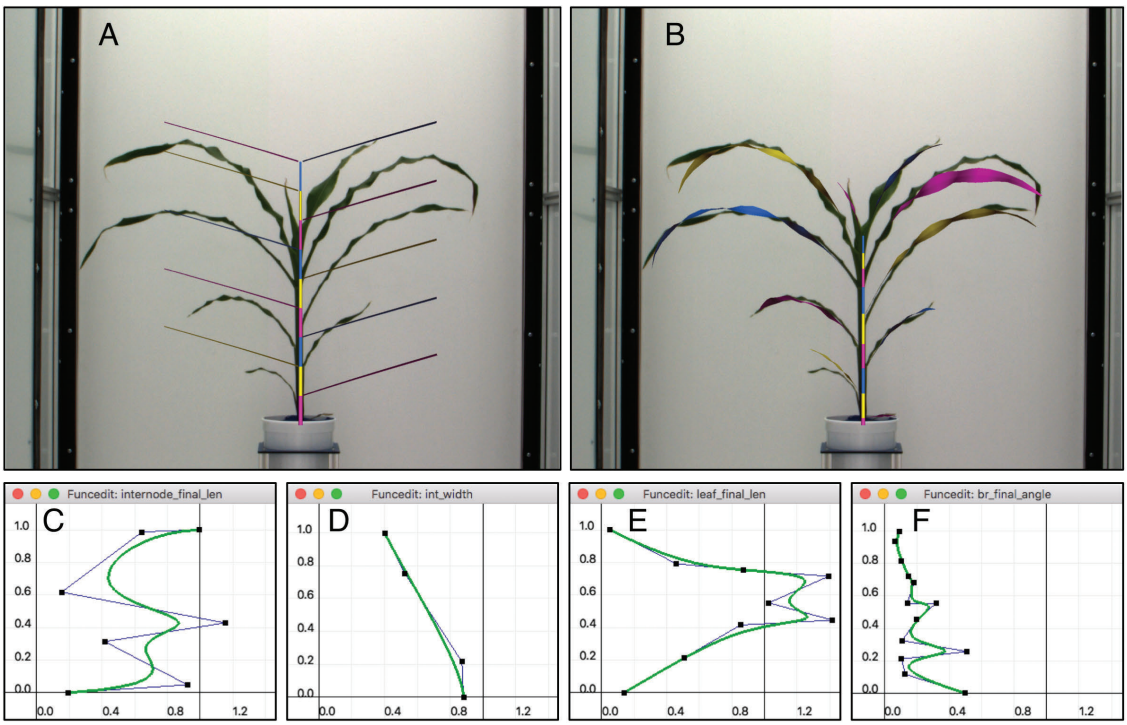

Figure 3. Screenshots showing the initial stage (A) and result (B) of visual calibration of the maize model. The model is overlaid on the reference image representing a 27-day-old plant from the UNNL-CPPD data set, CC BY-SA 4.0, Das Choudhury et al. (2018). Simulated leaves are distinguished from the photograph by artificial colours, which cycle between yellow, blue and magenta. Function plots control the length (C) and width (D) of internodes, the length of leaves (E) and the branching angles at which the leaves are inserted (F). The (vertical) *x*-axis represents phytomer numbers along the stem. The functions are manipulated interactively as in Fig. 2.

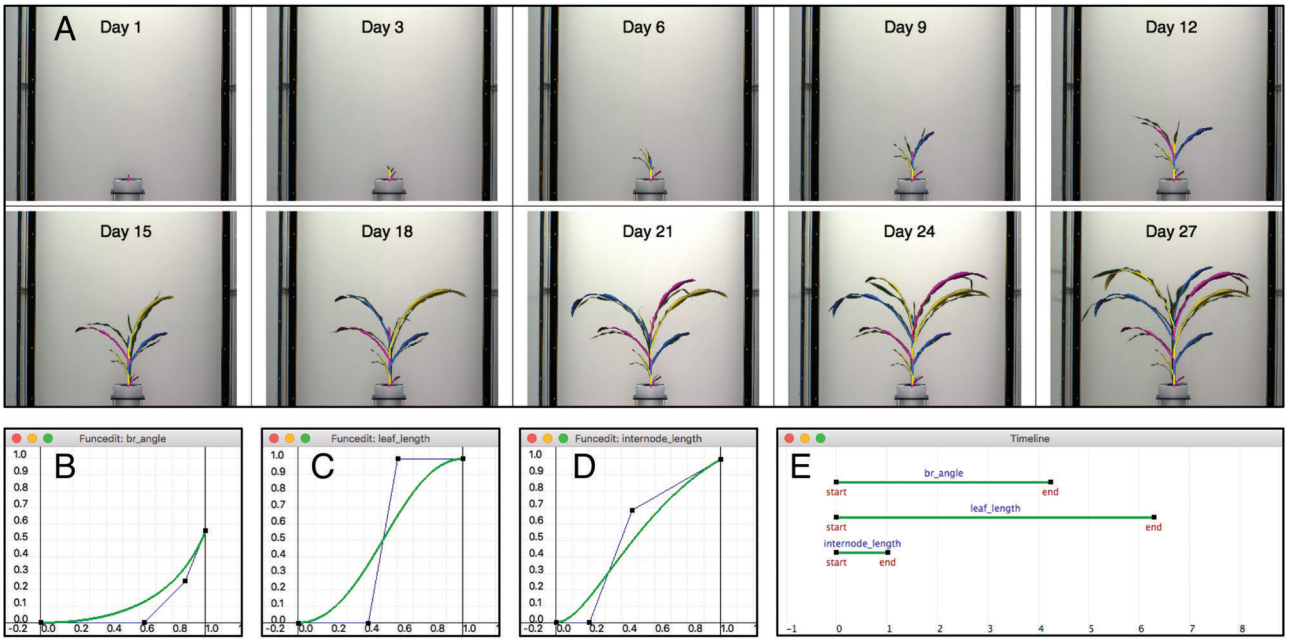

Figure 4. Screenshot showing the maize model calibrated to the developmental sequence of a maize plant. (A) The 10 background images, from the UNNL-CPPD data set, CC BY-SA 4.0, Das Choudhury et al. (2018), show the plant every 3 days. (B–D) The three graphically defined functions in the bottom row modify the branching angle, leaf size and internode length (cf. Fig. 3), given the phytomer age. (E) The time intervals (in days), over which these attributes change, are defined by the timeline editor on the bottom right.

3. THE CANOLA MODEL

Canola is an herbaceous annual plant with indeterminate inflorescences (Edwards 2011; Jullien *et al.* 2011; Canola Council of Canada 2020). The plant can grow between 70 and 175 cm high producing between 9 and 30 leaves with buds in the axil of the leaves on the main stem (Canola Council of Canada 2020). Lateral branches develop from the axillary buds subtended by the upper leaves; each branch then carries one to four leaves. The flowers on the main stem and secondary branches are arranged into racemes. Each flower has four yellow petals with six stamens. Flowers give rise to seed pods (siliques), which have the form of an elongated capsule.

Canola's growth can be divided into vegetative and reproductive phases. Their duration depends on environmental conditions, in particular temperature and day length. These phases can be further divided into seven principal stages (Canola Council of Canada 2020). The first stage, *germination*, begins with the growth of the hypocotyl and ends when both cotyledons are unfolded. A rosette of leaves is then formed in the *leaf production* stage. Near the end of this stage, *stem elongation*, followed by *flower bud development* begins. The *flowering* stage begins when the lowest flower bud opens on the main stem, and continues with further buds opening acropetally. Likewise, *pod development* begins at the first flower and progresses acropetally. The final stage is *seed development*.

3.1 Model construction

Our model simulates the entire vegetative phase and most of the reproductive phase up to seed development and ripening (Fig. 7). The basic structure of the L-system is similar to that of the maize model. Growth starts with both cotyledons and an apex as initial components. Production rules simulate the aging of organs, decomposition rules simulate the production of phytomers by apices and the activation of lateral buds and interpretation rules capture geometric aspects of the plant morphology.

To reduce the complexity of the model, we assume that canola's branching structure is topologically self-similar (Prusinkiewicz 2004b). We treat the main stem as a template for the lateral branches of the entire plant, mapping the functions and components of the main stem to the branches. Because of this repetition, the model uses a single set of functions for organ growth rates and target sizes. Some of the functions, however, are scaled depending on the branching order, e.g. the length of internodes and leaves is shortened on lateral branches compared to the main stem. Moreover, lateral apices are initialized with a phytomer count greater than zero, which reduces the total number of phytomers on a branch compared to that of the main stem.

The main components of the plant are represented by L+C modules. Table 1 outlines the modules and L-system rules used in the model.

In addition to the apices `A`, internodes `I` and leaves `L`, considered previously, the canola model incorporates axillary buds `B`, pedicels `P`, flower corollas `K` decomposed into petals `C` and pods `D`. An intermediate module, `M`, represents phytomers. To shorten the L-system specification, we associated all modules with the same data structure:

```
struct OrganData {
     float age; // chronological age (in days)
     int n;     // phytomer number along parent axis
     int order; // branching order, beginning at 0 for main stem
     int state; // the state of the apex: vegetative,
                // reproductive or inactive
};
```

Even though some parameters are needed for specific modules only, there are enough shared characteristics between different module types to make using a common data structure convenient.

Figure 5. Examples of the maize images generated using the same model with randomized parameter values. A background image of the Lemnatec photo chamber was added to facilitate unbiased comparisons of the synthetic images and photographs. All images represent the final day of growth (27th from seeding). The plastochron was chosen from a normal distribution with mean 3 days and standard deviation 0.5 days. The first leaf was equally likely to be on the left or right side. The whole plant was then rotated around the vertical axis by a normally distributed angle with mean 0° and standard deviation 10°. The divergence angles between consecutive leaves were randomized with mean 180° and standard deviation 5°. The normalized internode lengths returned by the `internode_target_len(n)` function were scaled by `IntLen` = 8.5 cm and multiplied by normally distributed random numbers with mean 1 and standard deviation 0.2. Similarly, the normalized leaf lengths were scaled by `LeafLen` = 60 cm and multiplied by normally distributed random variables with mean 1 and standard deviation 0.2. The exponent relating internode length to radius was constant, `ExpIntRad` = 1.5, as was the exponent for leaf width, `ExpLeafWid` = 0.25. Leaf shapes were randomized as described in the text. The angles of leaf insertion were scaled by `BrAngle` = 90° and multiplied by normally distributed random numbers with mean 1 and standard deviation 0.15.

Apart from the phytomers, which are immediately decomposed into their constitutive components, all components age by a fixed increment `dt` in each time step, as in the case of the maize model. For example, the time-advancing production for the apices is:

```
A(od): { od.age += dt; produce A(od); }.
```

The respective productions for the remaining modules are similar. In addition, apices, buds and flowers are subject to decomposition rules, which change the module state at some point in time.

The apex `A` is initially in the vegetative state, producing rosette leaves. After producing a predefined number of leaves (specified by an L-system parameter that can be manipulated interactively using a control panel), the apex switches from the vegetative to the reproductive state, and eventually becomes inactive. While active, it produces a new phytomer `M` when the time interval between successive production of phytomers—the plastochron, measured in days—elapses. The process is implemented as the following decomposition rule:

```
// define apical states
#define vegetative 0
#define reproductive 1
#define inactive 2
// define the apex behaviour
A(od): {
     // model change of apex state
     if (od.state == vegetative && od.n >= VegPhytomers)
          od.state = reproductive;
     else if (od.state == reproductive && od.n >= MaxPhytomers)
          od.state = inactive;
     // check if apex should produce a phytomer
     if (od.state != inactive) {
          // calculate plastochron given apex state
          float p;
          if (od.state == vegetative)
               p = VegPlastochron*plastochron(od.n)
          else
               p = RepPlastochron;
          // if plastochron has elapsed
          if (od.age >= p) {
               od.age = od.age - p;
               nproduce M(od) // produce a phytomer
                        RollL(137.5); // simulate phyllotaxis
               od.n += 1; // then increase the phytomer count
               produce A(od);
          }
     }
}
```

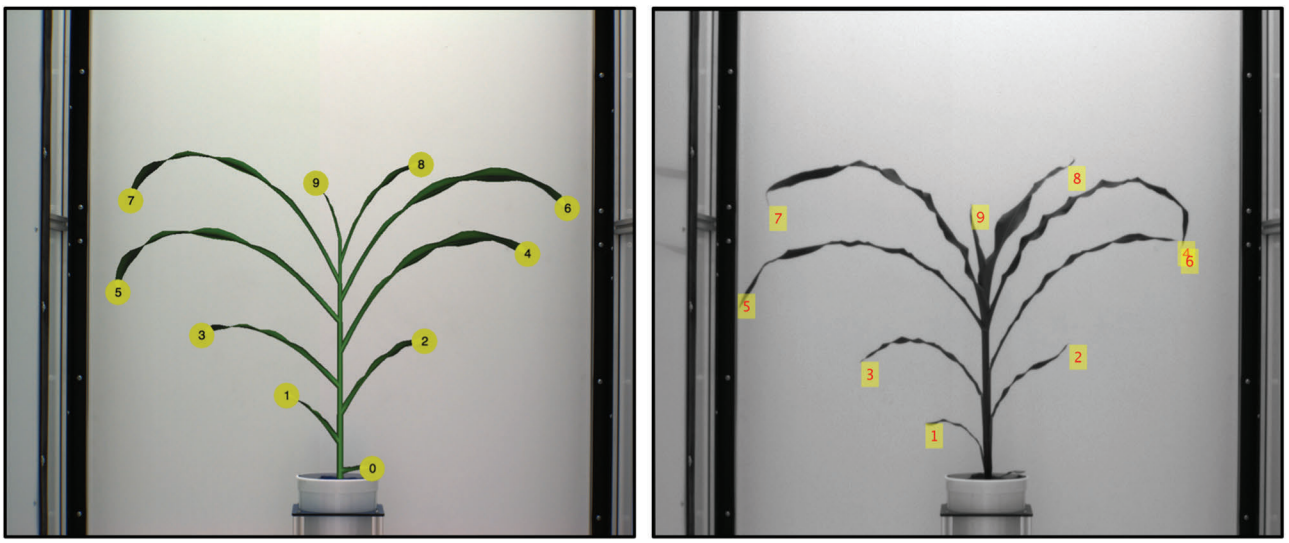

Figure 6. A comparison of an annotated image of a synthetic plant (left) to the image of a real plant (right) from the UNNL-CPPD data set, CC BY-SA 4.0, Das Choudhury *et al.* (2018).

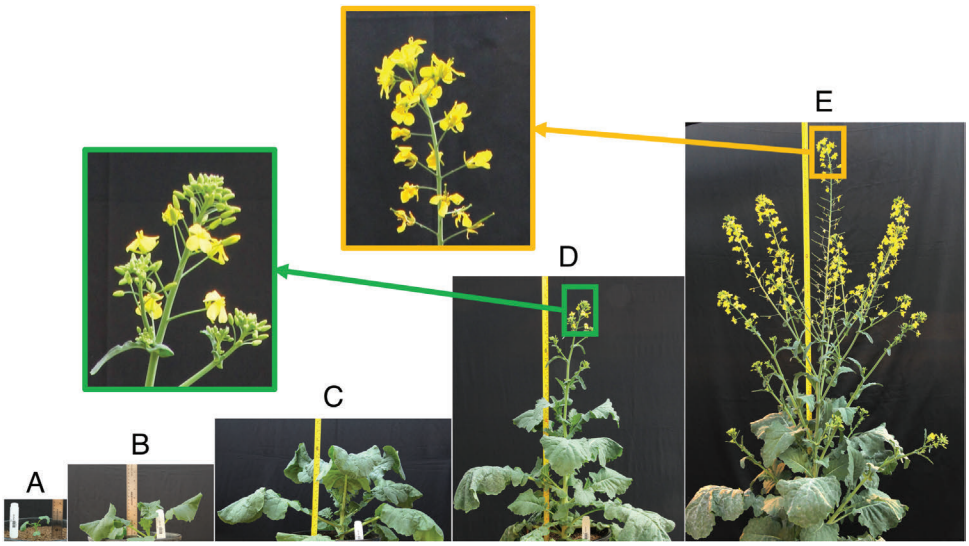

Figure 7. Developmental sequence of a canola plant illustrating the stages considered in the model: (A) germination, (B) rosette, (C) stem elongation, (D) flower bud development and (E) flowering and pod development. The insets show the tip of a developing inflorescence, highlighting an initial cluster of flower buds and a later flowering stage.

The parameters `VegPhytomers` and `MaxPhytomers`, defined by the modeller, set the maximum number of vegetative and total phytomers on the given axis. In the vegetative state, the time interval between the production of successive phytomers—the plastochron—depends on the number of phytomers it has produced. This dependence is captured by defining the plastochron using a (graphically specified) function of the phytomer count, `plastochron(od.n)`, scaled by the parameter `VegPlastochron`. The change in plastochron based on growth stage has been observed in canola (Jullien *et al.* 2011, 2012) and *Arabidopsis* (Mündermann *et al.* 2005). The latter paper also reported a dependence on phytomer number in the early vegetative growth of *Arabidopsis*. For the reproductive state, we assume the plastochron to be the constant `RepPlastochron` (Jullien *et al.* 2011, 2012). The leaves and branches on the stems are arranged in a spiral phyllotaxis (Polowick and Sawhney 1986), simulated by rotating the coordinate frame, defined along a stem, by the golden angle of 137.5° between consecutive phytomers.

As soon as it is produced, a phytomer `M(OrganData)` is decomposed into an internode `I`, axillary bud `B` and leaf `L`, if it was produced by a vegetative apex, or an internode `I`, pedicel `P` and flower `K`, if it was

Table 1. The plant components (modules) of the canola model, a general characterization of the associated L-system rules, and the graphical interpretation of the modules. Veg.: vegetative; Rep.: reproductive; Gen.: generalized.

Component	Symbol	Production	Decomposition	Interpretation
Apex	`A`	Increment age	`A → M A`	—
Veg. phytomer	`M`	—	`M → I [L][B]`	—
Rep. phytomer	`M`	—	`M → I [P K]`	—
Internode	`I`	Increment age	—	Cylinder
Leaf	`L`	"	—	Gen. cylinder
Axillary bud	`B`	"	`B → A`	—
Pedicel	`P`	"	—	Cylinder
Flower corolla	`K`	"	`K → D`	`K → [C][C][C][C]`
Petal	`C`	"	—	Gen. cylinder
Pod	`D`	"	—	Gen. cylinder

produced by a reproductive apex. Both possibilities are captured by a single L+C rule:

```
M(od): {
     nproduce I(od); // an internode
     if (od.state == vegetative)
          produce SB B(od) EB  // an axillary bud, and
                  SB L(od) EB; // a leaf
     else // in reproductive state
          produce SB P(od) K(od) EB; // a pedicel and flower
}
```

An internode is added to the current axis, while axillary buds, leaves and flowers with pedicels are added as lateral organs, indicated by branching symbols `SB` and `EB`. All components constituting the phytomer inherit the phytomer number and branch order from the apex that produced it. The attributes of the axillary bud change, however, when it is activated and becomes a new lateral apex. This transition is controlled by bud vigour `v`: if it is above the threshold `MinBudVigour`, the bud will grow; otherwise, it will remain dormant. The decomposition rule also checks that the maximum branching order, `MaxOrder`, has not been reached.

```
B(od): {
     // ignore buds above maximum branch order
     if (od.order >= MaxOrder)
          produce;
     // calculate bud vigour
     float v = bud_vigour(od.n) * (1 - ShortV*od.order);
     if (v > MinBudVigour) {
          // set the branching angle and the shoot gravitropic
          // response before new lateral apex is produced
          nproduce Down(BranchAngle * branch_angle(od.n))
                   SetElasticity(0,Elast*branch_tropism(od.n));
          // set organ data to produce new lateral apex
          od.state = vegetative;
          od.age = BudDelay * bud_delay(od.n);
          od.order = od.order + 1;
          od.n = VegPhytomers * bud_init_node(od.n);
          produce A(od);
     }
     produce B(od);
}
```

Bud vigour controls which buds along the parent axis will produce lateral branches. It is a phenomenological parameter modelling the endogenous (e.g. hormonal) and exogenous (e.g. temperature) factors that control phenotype-dependent branching patterns (Lück *et al.* 1990; Prusinkiewicz *et al.* 1997). Its value is calculated using a graphically defined function of phytomer number, `bud_vigour(od.n)`, which is reduced by a user-defined parameter, $0 \leq$ `ShortV` ≤ 1, proportionally to branch order `od.order`. This is the first instance in our model of using repetitions in structure to reduce the number of user-defined functions (i.e. one scaled function, `bud_vigour(od.n)`, is used for all axes).

If a bud produces a branch, the functions `branching_angle(od.n)`, and `branch_tropism(od.n)` control its shape given the phytomer number `n` associated with this bud. The functions are scaled by user-defined parameters `BranchAngle` and `Elast`, respectively. The second function sets the branch's response to simulated gravitropism, which reorients the branch in the vertical (upward) direction. The branch's susceptibility to bending is controlled by the `SetElasticity` module.

The delay in outgrowth of a bud is controlled by the `bud_delay(od.n)` function, which is scaled by a user-defined parameter `BudDelay`. It is used to control the pattern of bud activation along a parent axis. Buds assigned a smaller (possibly, negative) initial age value will be delayed in their outgrowth compared to buds with a larger value. The `bud_init_node(od.n)` function sets the initial phytomer number of a lateral apex given the bud's phytomer number on the parent axis. This makes it possible to reduce the number of phytomers in a branch by initializing its apex with a number greater than one (Mündermann *et al.* 2005).

The final decomposition rule defines the fate of a flower. A parameter, `PoddingAge`, defines the number of days before a flower has the potential to develop into a pod. Upon reaching this age, the flower produces a pod with probability depending on the age of the plant, `t`, or aborts.

```
K(od): {
     if (od.age > PoddingAge)
          if (ran(1) < prob_podding(t/MaxPlantAge)) {
               od.age = 0;    // set pod age to zero
               produce D(od); // flower becomes a pod
          } else produce;     // or flower aborts
     produce K(od);           // otherwise flower remains
}
```

The function `ran(1)` generates a random number with a uniform distribution. The graphically defined function `prob_podding(t/MaxPlantAge)` sets high probability of flower development into a pod within the first 15 days after the onset of flowering and reduces this probability afterward (Jullien *et al.* 2011), so that only 40–55 % of flowers (McGregor 1981) will develop productive pods.

The geometry of all plant components is determined by interpretation rules. The internodes are modelled as cylinders, as in the maize model, but the graphically defined function for target length is scaled by two factors: (i) a parameter to scale all the internodes (i.e. stretching the target length function), and (ii), on lateral branches, a parameter to shorten the internodes (relative to the main stem). The interpretation rule is:

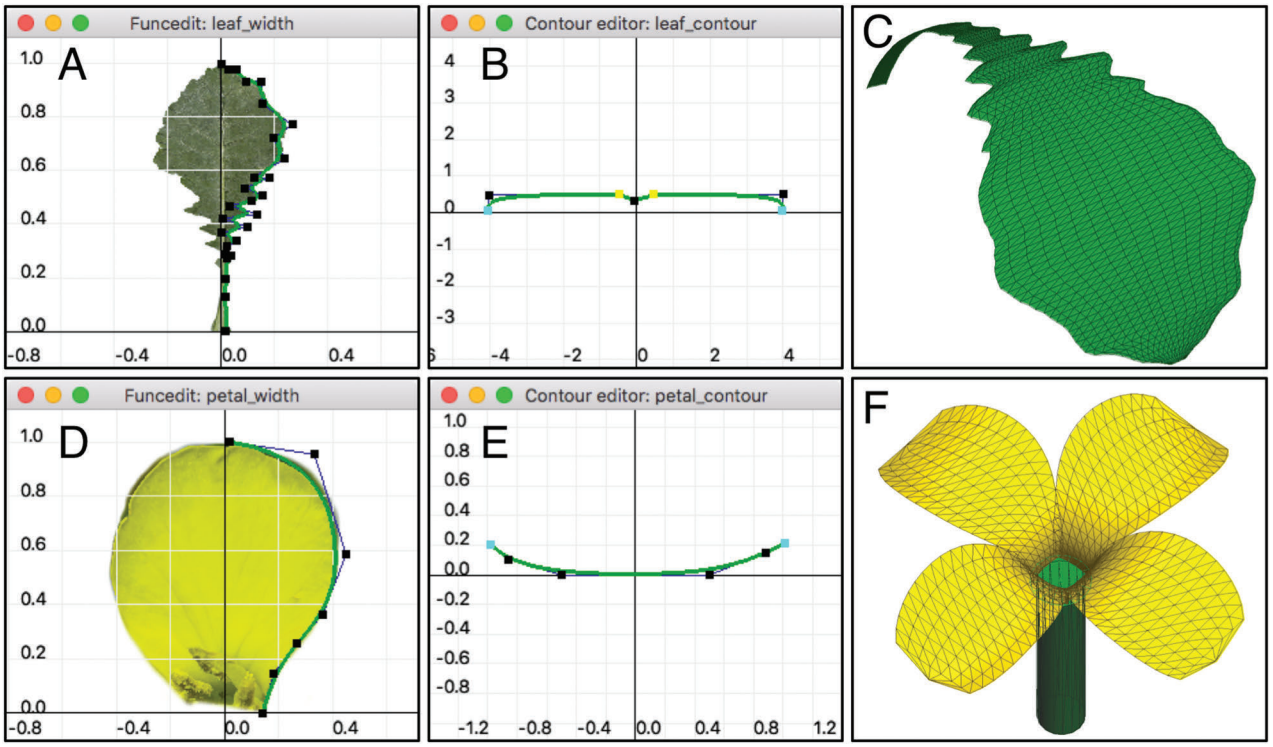

Figure 8. Modelling canola leaves and flowers. We specify the profile and cross-section of a sample leaf (A, B) and petal (D, E) using an interactive curve editor. For the profile, we use references images as a template (A, D). These contours are used to generate generalized cylinders, which are converted to triangle meshes (C, F). The template petal image has been cropped out of a photograph by Didier Descouens, CC BY-SA 4.0.

Figure 9. Sample Lemnatec Scanalyzer images of a single canola plant in different stages of development (days after seeding). In the first phase of calibration, a single reference image at the podding stage is used (e.g. Day 59), but, in the second phase, several (e.g. 8–10) images at different stages are used.

```
I(od): {
     // calculate the scaling factor for shortening
     float shorten = 1 - ShortInt * od.order;
     // calculate scaled length
     float len = IntLen * int_target_len(od.n) * shorten
               * int_length(od.age);
     // calculate scaled radius
     float rad = powf(len,ExpIntRad);
     // draw a cylinder
     produce SetWidth(rad) F(len);
}
```

The user-defined variables `ShortInt`, `IntLen` and `ExpIntRad` are chosen such that $0 \leq$ `ShortInt` ≤ 1 and `IntLen`, `ExpIntRad` ≥ 0. An internode's radius is scaled allometrically by its length raised to power `ExpIntRad`.

The leaves are modelled as generalized cylinders with a graphically defined profile and cross-section (also referred to as a contour, cf. Fig. 8). The profile determines the local leaf width as a function of position along the midrib. We specified it by tracing the image of a reference leaf, used as a

template (Fig. 8A). The cross-section is a curve scaled according to the leaf width (Fig. 8B). For simplicity, we assume that the profile and cross-section are the same (up to allometric scaling) for all leaves in the model. In contrast to the maize model, which interpolated between a closed cross-section for the sheath and an open cross-section for the leaf blade, we employ a single curve to approximate the cross-section of the entire canola leaf. Altogether, the shape of a canola leaf is captured by the following interpretation rule:

```
L(od): {
     // calculate leaf length
     float len = LeafLen * leaf_target_len(od.n)
       * (1 - ShortLeaf * od.order) * leaf_length(od.age);
     // precompute leaf width scaling factor
     float wid = LeafWid * powf(len,ExpLeafWid);
     // set leaf width at the base
     nproduce SetWidth(wid * leaf_width(0));
     // set leaf insertion angle
     nproduce Down(BranchAngle * branch_angle(od.n));
     // set leaf cross-section to curve with index 1
     nproduce CurrentContour(1);
     // start drawing the generalized cylinder
     nproduce StartGC;
     for (float x = 0; x < 1; x += DX) {
          // generate leaf segment
          nproduce F(len*DX);
          // set width and orientation of next segment
          nproduce SetWidth(wid * leaf_width(x))
                   Down(leaf_bend(x) * leaf_angle(od.n) * DX);
     }
     nproduce F(len*(1-x)); // generate last segment
     produce EndGC; // end the generalized cylinder
}
```

`LeafLen` ≥ 0 is a parameter that scales the target length for all leaves of the plant, and `ShortLeaf` is a parameter that shortens leaves on higher order branches compared to those at the corresponding positions on the main stem. The parameter `LeafWid` ≥ 0 and allometric exponent `ExpLeafWid` ≥ 0 scale the target width for all leaves on the plant. The leaf insertion angle is controlled by the same function used to set branching angles. Leaf curling is modelled by two functions: the change in angle along the midrib, `leaf_bend(x)`, and a factor dependent on leaf position, `leaf_angle(od.n)`. The first function modifies the curling of all leaves, while the second function modifies it for individual leaves.

Flowers and pods are supported by pedicels, which are modelled as cylinders using an interpretation rule similar to that for internodes. To reduce the number of graphically defined functions, we assume that a pedicel's length is related allometrically to the target length of the supporting internode:

```
P(od): {
     // calculate scaled length
     float len = PedicelLen *
                 powf(int_target_len(od.n),ExpPedicelLen) *
                 pedicel_length(od.age);
     // calculate radius
     float rad = powf(len,ExpPedicelRad);
     // draw a cylinder
     produce SetWidth(rad) F(len);
}
```

`PedicelLen`, `ExpPedicelLen` and `ExpPedicelRad` are parameters defined interactively by the modeller during model calibration.

Flowers are modelled as four petals placed in a radially symmetric pattern at the tip of the pedicel. Each petal is represented by a generalized cylinder in the same way as the leaves (Fig. 8D–F). Specifically, petal width is a graphically defined function of position along the petal midrib, traced using the image of a fully developed petal as a template (Fig. 8D). We assume that the target length of each petal is allometrically related to the target length of the internode that supports the flower. Flower opening is simulated by changing the petal length using a function of flower age; in addition, the angle of the petals with respect to the supporting pedicel is assumed to increase as a function of age. The resulting interpretation rules are as follows:

```
K(od): {
     // simulate flower opening
     nproduce Down(90 * petal_angle(od.age));
     // draw corolla (i.e., four petals)
     for (int i = 0; i < 4; i++) {
          // rotate in 90-degree increments
          nproduce RollL(90 * i) SB C(od) EB;
     }
}

C(od): {
     // calculate petal length
     float len = PetalLen *
                 powf(int_target_len(od.n),ExpPetalLen) *
                 petal_length(od.age);
     // calculate and set the initial petal width
     float wid = powf(len,ExpPetalWid);
     nproduce SetWidth(wid * petal_width(0));
     // set petal cross-section to curve with index 2
     nproduce CurrentContour(2);
     // start drawing the generalized cylinder
     nproduce StartGC;
     for (float x = 0; x < 1; x += DX) {
          // generate petal segment
          nproduce F(len*DX) SetWidth(wid * petal_width(x));
     }
     nproduce F(len*(1-x)); // generate last segment
     produce EndGC; // end the generalized cylinder
}
```

The parameters `PetalLen`, `ExpPetalLen` and `ExpPetalWid` are analogous to those for leaves. Each individual petal is visualized using an interpretation rule similar to that used for the leaves.

Finally, pods are modelled as generalized cylinders with a round cross-section and uniform thickness (similarly to the pedicels), ending with a narrower beak.

```
D(od): {
     // calculate pod length and radius
     float len = PodLen *
                 powf(int_target_len(od.n),ExpPodLen) *
                 pod_len(od.age);
     float rad = powf(len,ExpPodRad);
     // draw main segment of the pod followed by the beak
     produce SetWidth(rad) StartGC F(len*0.8)
             SetWidth(rad/10) F(len*0.2) EndGC;
}
```

The number `PodLen` scales the length of all pods in the plant, the exponent `ExpPodLen` scales the pod length allometrically by the target length of the internode it is attached to and the radius is allometrically related to pod length with the exponent `ExpPodRad`.

3.2 Model calibration

To show the capability of the L-system model to capture different canola phenotypes, we calibrated it to four different plants with contrasting phenotypes, grown in controlled environments in a greenhouse at the University of Nebraska high-throughput phenotyping facility. Starting at seedling emergence, each plant was photographed at intervals ranging from 1 to 4 days until the podding stage using the Lemnatec Scanalyzer (Fig. 9). The photographs were taken from different angles, but, for calibration, we used the front, side and top views only. Phenotypical differences included growth rates, the overall height, plant posture (erect vs. bent main stems), branching angles, and the number and sizes of organs. For each phenotype, the model was calibrated first to a target developmental stage, which was about

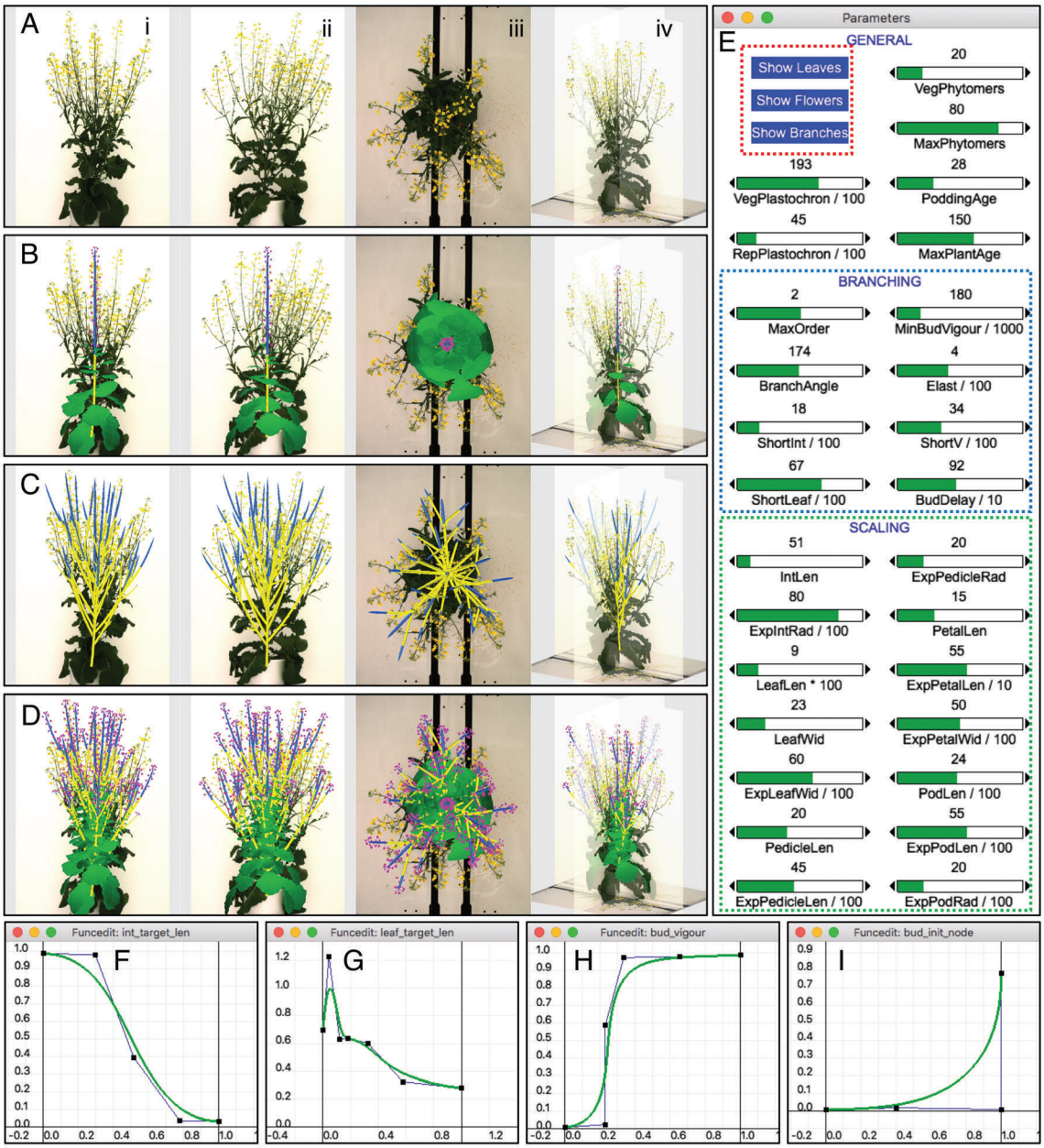

Figure 10. Example of the canola model calibration to a target plant developmental stage. The plant simulator window is shown four times (A–D) to illustrate the process, whereas the settings in the parameters window (E) and function editors (F–I) correspond to the final calibration (D). (A) Initially, all components of the model are turned off with the 'Show…' buttons enclosed within the red rectangle in (E). The reference plant is shown in front (i), side (ii), top (iii) and perspective (iv) views. (B) After turning on the visibility of the main stem, leaves and flowers, the modeller specifies the number of phytomers, the length of the main stem and the size of the leaves, flowers and pods (with their pedicels). The internodes are coloured yellow in the vegetative section and blue in the reproductive section. During the calibration, the flowers and pods are coloured magenta to distinguish them from the yellow flowers in the reference images. The sliders in the SCALING section, within the green rectangle of the control panel (E), modify the model parameters used in the interpretation rules for internodes, leaves, pedicels, flowers and pods. The number above each slider indicates the current value. For example, the `IntLen` parameter scales the function for target internode length (F), and `LeafLen` scales the target leaf length (G). (C) Next, the modeller sets the number and lengths of lateral branches. The visibility of leaves and flowers is temporarily turned off, while the visibility of the branches is enabled. The sliders in the BRANCHING section, within the blue rectangle in (E), set the bud activation threshold and scale the graphically defined functions controlling branching. The functions for the bud vigour (H) and the initial phytomer number (I) are shown. With all model components visible, the final calibrated model is shown in (D).

Figure 11. Sample canola plants with contrasting architectures (top) and their calibrated models (bottom). Phenotype (A, D) has short branches curving upward and large branching angles, (B, E) has long branches and small branching angles and (C, F) has large branching angles and straight branches.

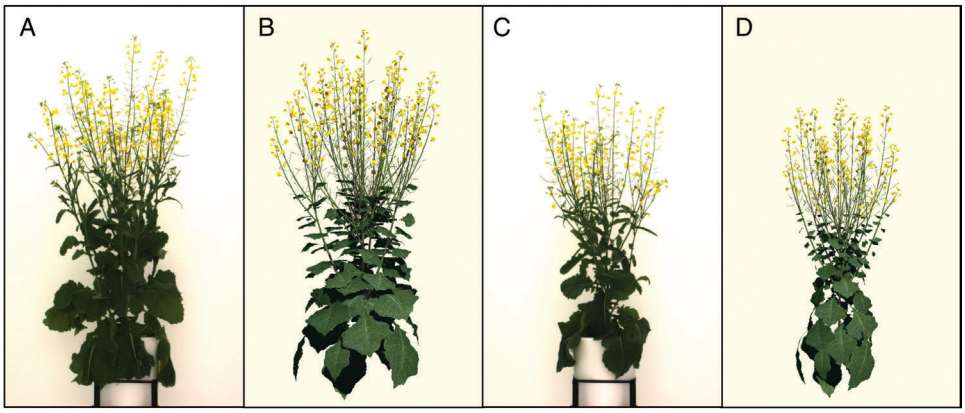

Figure 12. Images (A, C) and models (B, D) of canola plants with the same genotype, grown under controlled (A, B) and water-stressed (C, D) conditions. Compared to the control, in the water-stressed model the internode target length as well as leaf target length and width were reduced, leaves on higher order branches were shortened further, branching threshold was increased and the axillary bud vigour and activation delay functions were modified. Finally, the position-dependent leaf curling function was changed so that the leaves drooped more in the water-stressed plant.

60 days after seeding depending on the plant, then extended to the entire developmental sequence by setting the functions of time.

The first phase of the calibration is illustrated in Fig. 10 (and **Supporting Information—Movie 2**). The model and the reference images are shown in front, side and top views, as well as in a view combining all three reference images and the model (see Fig. 10A). In the latter case, the reference images are semi-transparent, so that they do not obscure the model when the modeller rotates it. Using all views, the modeller interactively manipulates functions of position along the axes, as in the model of maize, and numerical parameters, such as scaling factors associated with functions and exponents associated with allometric relations, until a satisfactory agreement with all reference images is reached.

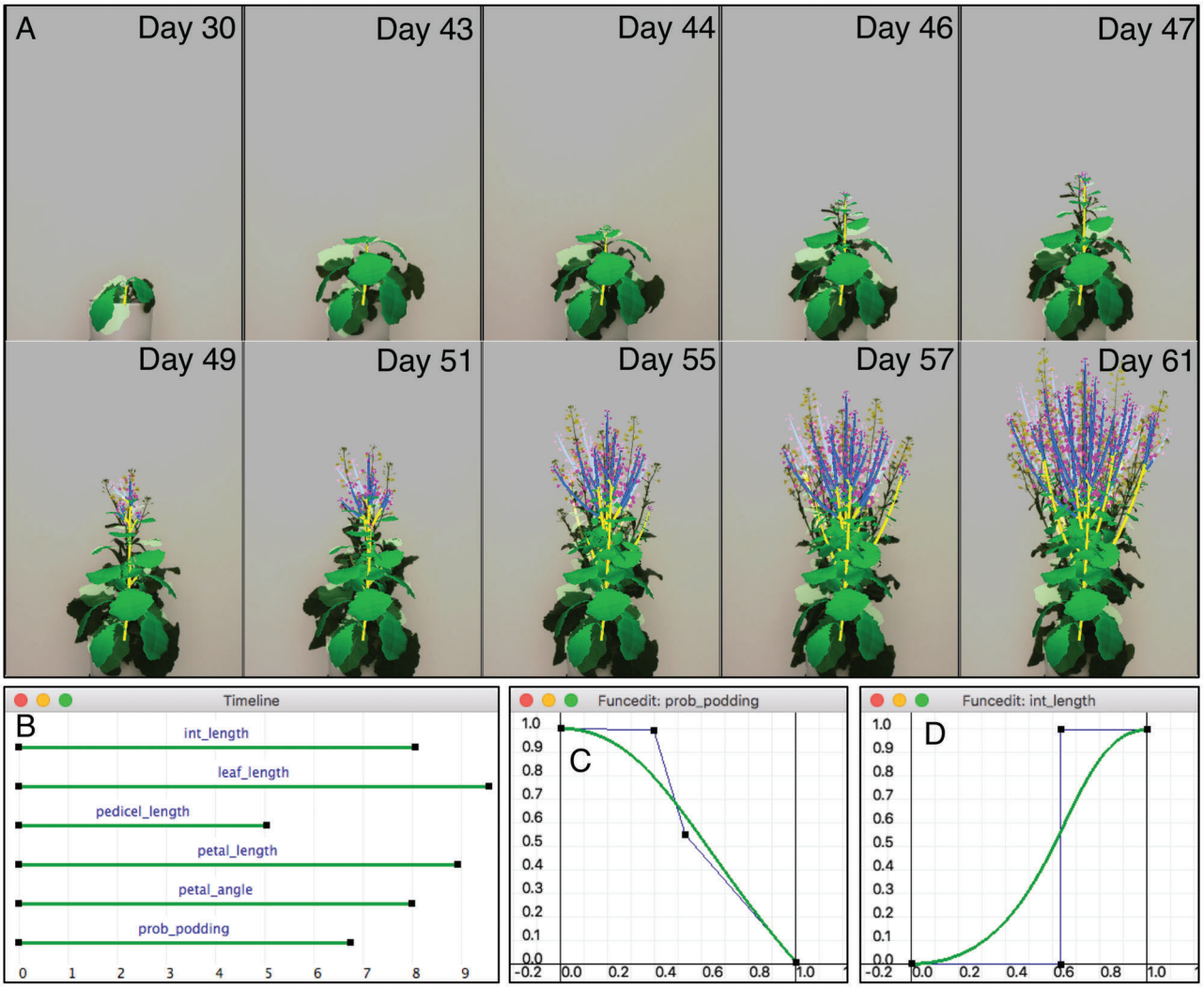

Figure 13. Calibration of the canola model to a developing plant. (A) The plant simulator shows images of 10 selected stages of the plant development and the corresponding states of the model (days after seeding). Semi-transparent reference images are positioned in the plane of the main stem of the simulated plant. Colours of simulated organs are the same as in Fig. 10. Some lateral organs appear darker in the visualization because they are behind the reference images. (B) The timeline editor specifying time scales for functions of organ age (the first five timelines) or overall plant age (the last timeline) in days. (C) Example definition of the probability of podding function. (D) The internode elongation function. Other growth functions (not shown) have a similar sigmoidal shape.

The first step is the calibration of the main stem. The numbers of phytomers in the vegetative and reproductive stem segments are specified interactively using sliders in a control panel configured by the modeller prior to the calibration. The modeller then sets the target length and width of leaves, flowers and pods supported by the main axis (Fig. 10B), using graphically defined functions (Fig. 10F and G) and their scaling factors (Fig. 10E). To facilitate calibration, different components of the model can be made visible or invisible using on/off buttons included in the control panel.

In the next step, branching is enabled while leaves and flowers are hidden (Fig. 10C). The number of branches is controlled by modifying the bud vigour function (Fig. 10H) and its scaling factor, as well as the branching threshold (Fig. 10E). Subsequently, the modeller calibrates the lengths of the lateral branches. This involves manipulating the initial phytomer number to control the number of leaves supported by the lateral branches (Fig. 10I), and a scaling parameter that shortens internode length as the branch order increases (Fig. 10E). The maximum order of branching is controlled by a slider in the panel. The functions for branching angle and elasticity, together with their scaling factors, are calibrated last (Fig. 10D). Three examples of calibrated models are shown in Fig. 11, and a model calibrated to controlled and water-stress conditions for the same genotype is shown in Fig. 12.

Figure 14. Eight simulated stages of the development of an individual plant (days after seeding).

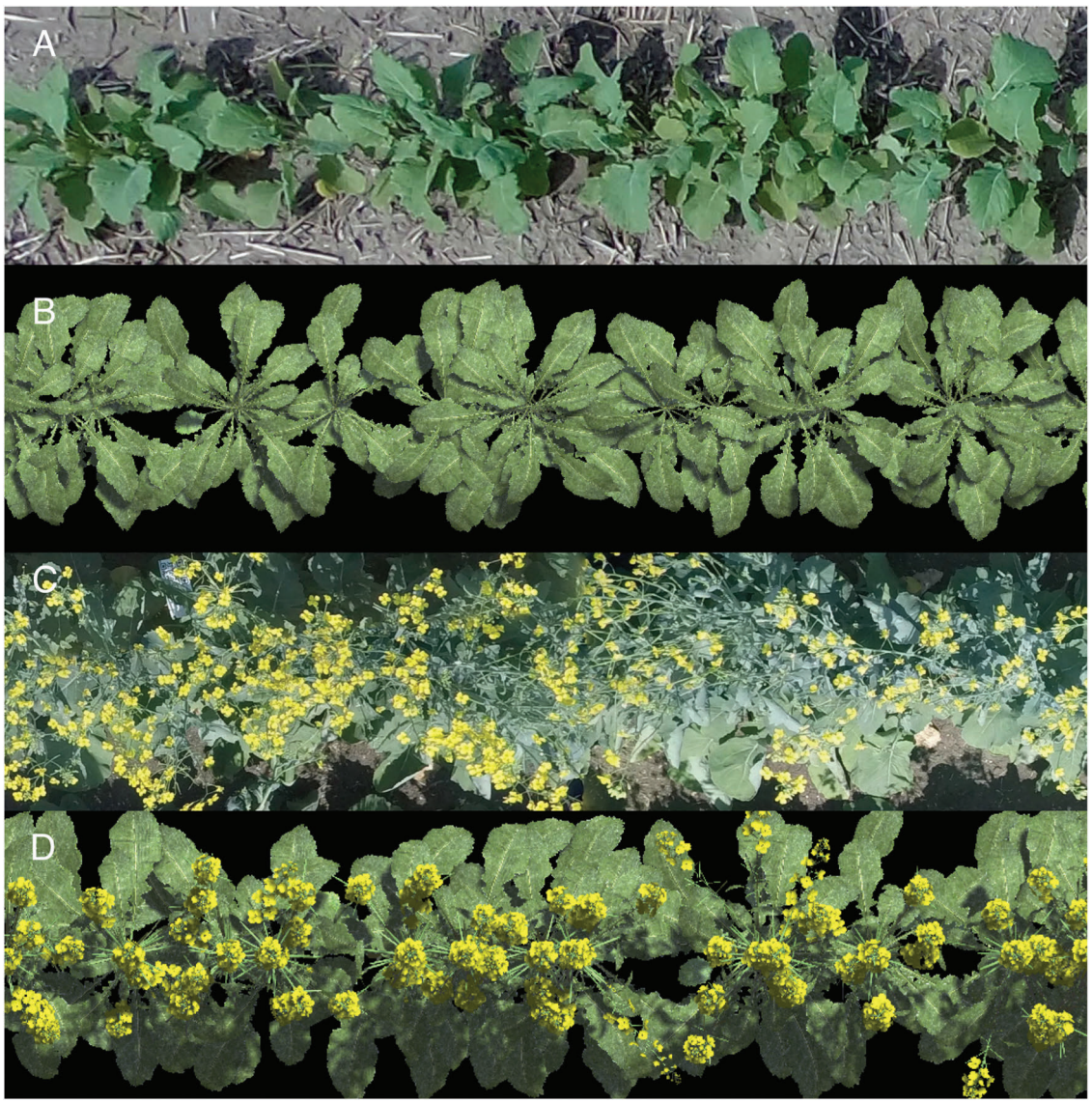

Figure 15. Photographs (A, C) and models (B, D) of a canola plot. Images acquired as described by Higgs *et al.* (2019), courtesy of Ian Stavness (University of Saskatchewan) and Sally Vail (Agriculture and Agri-Food Canada).

Model calibration is then extended to the development of the plant over time (Fig. 13). This extension typically requires iterative adjustments of functions, timelines and individual model parameters until the reference images are adequately matched. We found that the following workflow is effective in practice. First, the modeller calibrates the development of the main stem (with the visibility of the branches

turned off). This includes setting the `plastochron(od.n)` function (not shown) and its scaling factor `VegPlastochron` (Fig. 10E) to pace vegetative growth, the `RepPlastochron` plastochron (Fig. 10E) to pace reproductive growth and the internode elongation function `int_length(od.age)` (Fig. 13D), together with the time points marking the beginning and end of the elongation (Fig. 13B), to match the observed pattern of the main axis development. With these aspects settled, the modeller calibrates the leaf growth rate function `leaf_length(od.age)`, together with its start and end time points, as was done for the internodes. Then, the parameters controlling the allometric scaling of higher order branches and the initial phytomer number of axillary buds are calibrated. Finally, the bud activation delay, function `bud_delay(od.n)` (not shown), and its scaling parameter `BudDelay` (Fig. 10E) are set. With the smallest delay at the base of the plant and the largest at the apex, the model simulates a basipetal branching pattern that closely matches the timing of flowering recorded in the images.

The calibrated plants can be used to visualize canola plants at different developmental stages individually (Fig. 14) or be assembled into models of entire canola plots. In the models of canola plots shown in Fig. 15 the interactions between plants have been ignored. L-system extensions that allow for simulating different types of interaction between plants exist (Měch and Prusinkiewicz 1996) and are supported by L+C, but the resulting models are more complex and computationally expensive than the models discussed in this paper.

4. CONCLUSIONS

The emerging applications of neural networks to image-based phenotyping have created a demand for large sets of annotated, visually realistic plant images needed to train these networks. Computer-generated images can capture genetic diversity, the influence of the environment and individual variation of plants, and can be annotated in an inherently correct manner. Consequently, they provide a reliable source of ground truth for training and evaluating image-based phenotyping algorithms, which can be successfully used in addition to—or even as a replacement of—real plant images (Ubbens *et al.* 2018; Miao *et al.* 2019; Jiang and Li 2020). The need to efficiently create models that generate these images thus arises. With maize and canola serving as examples, we have shown how modelling methods based on L-systems can be combined with interactive visual calibration to provide a solution.

The key elements of the presented method are as follows:

1) The modelling process is divided into two stages: (i) the construction of an L-system, capturing the essential elements of the plant species of interest qualitatively, and (ii) model calibration to a set of photographs of reference plants. The calibration itself proceeds in two substages: calibration towards a target, fully developed plant, and incorporation of development over time.
2) To facilitate calibration, the L-system model is organized around the concept of positional information, which means that the key quantitative aspects of the target plant form, such as the distribution of branches, leaves and reproductive organs, are expressed as intuitive, easy to manipulate functions of position on their supporting axes. Developmental processes are simulated by multiplying functions of positional information by functions of time.
3) The number of functions requiring independent calibration is reduced through the use of similarities and allometric relations between model components. These characteristics are controlled by numerical parameters.
4) Quantitative aspects of the model are manipulated using graphically defined function, contour and timeline editors, as well as control panels configured according to the structure of the model. Select functions and the resulting models are superimposed on the reference images to facilitate the visual calibration. All changes to parameters and functions are reflected immediately in the displayed model, providing instantaneous visual feedback.
5) A practically unlimited number of images reflecting individual variation of plants can then be generated by randomizing model parameters. As the images are based on underlying 3D models, they can be automatically annotated with any features of interest, including those difficult to measure in actual plants, such as the length of individual internodes, and the branching and phyllotactic (divergence) angles.

The usefulness of annotations extends beyond the applications of generated images to train neural networks. In particular, calibrating models to real plants provides a quantitative estimate of the architectural parameters of these plants without measuring them directly. While the traditional process of creating descriptive plant models proceeds bottom-up, from detailed measurements of architectural parameters to their integration into the models, our modelling method provides a top-down method for estimating these parameters based on the overall similarity of the models to reference images.

In the context of machine learning, the images can be used in two modes: by pre-generating a (potentially large) number of images and placing them in a database along with photographs of real plants, or by providing generative models that synthesize and output annotated images on demand. The former approach has the advantage of being immediately usable within the standard process of training artificial neural networks. The latter approach avoids storing the images, and may be particularly useful in active learning. In that case there is a feedback between image generation and the learning process, such that image generation is affected by the output produced by the network being trained. For example, if the objective is to count pods, but the network is sensitive to the pod size, the image-generating algorithm could increase the size variance. This increase may allow the network to more efficiently learn that size differences should be discounted. A detailed analysis of the use of the proposed method for training neural networks according to various scenarios is an important topic requiring further research.

The presented method exploits the synergy between a human and a computer (Licklider 1960) to efficiently create calibrated plant models: the computer generates and renders candidate models, and the operator adjusts them based on visual comparisons to reference images. As a result, relatively complex models, here exemplified by canola, can be calibrated quickly, in the order of minutes, by an experienced modeller.

A user study quantifying the time that modellers with different levels of experience need to calibrate sample models—and the differences between the resulting models—would be an interesting complement to the work presented here. Another intriguing question is whether the human operator could be eliminated altogether, making the calibration entirely automatic. This could be achieved if a suitable metric for evaluating discrepancies (distances) between the model and reference images could be found. The definition of such a metric, and algorithms for the efficient minimization of these distances, is also an interesting area of further research.

SUPPORTING INFORMATION

The following additional information is available in the online version of this article—

Movie 1. Calibration of a maize model.

Movie 2. Calibration of a canola model.

ACKNOWLEDGEMENTS

We thank Jana Ebersbach for clarifications regarding the canola data set, and Richard Morris, Jo Hepworth and the reviewers for constructive comments on the manuscript.

CONFLICT OF INTEREST

None declared.

CONTRIBUTIONS BY THE AUTHORS

M.C., P.P.: modelling and writing; M.C., P.F., P.P.: software development and testing; N.K., R.S., S.J.R., I.P., I.M.: data acquisition and analysis; M.C., N.K., P.F., R.S., S.J.R., I.P., I.M., P.P.; manuscript reviewing and editing.

SOURCES OF FUNDING

The support of our research by the Plant Phenotyping and Imaging Research Centre – Canada First Research Excellence Fund (M.C./P.P. and I.M.), the Natural Sciences and Engineering Research Council of Canada (Discovery Grants 2016-06172 to I.M. and 06279-2019 to P.P.) and X, the moonshot factory (P.P.), is gratefully acknowledged.

DATA AVAILABILITY

The models were implemented using the Virtual Laboratory 4.5.1 plant modelling software (algorithmicbotany.org/virtual_laboratory) on macOS High Sierra v.10.13.6, and are available at the Algorithmic Botany website (algorithmicbotany.org/papers/l-phenomics2021.html).

LITERATURE CITED

Abelson H, DiSessa AA. 1986. *Turtle geometry: the computer as a medium for exploring mathematics*. Cambridge: MIT Press.

Artzet S, Chen TW, Chopard J, Brichet N, Mielewczik M, Cohen-Boulakia S, Cabrera-Bosquet L, Tardieu F, Fournier C, Pradal C. 2019. Phenomenal: an automatic open source library for 3D shoot architecture reconstruction and analysis for image-based plant phenotyping. *bioRxiv*.

Bartels R, Beatty J, Barsky B. 1995. *An introduction to splines for use in computer graphics and geometric modeling*. Los Alamos, NM: Morgan Kaufmann.

Brichet N, Fournier C, Turc O, Strauss O, Artzet S, Pradal C, Welcker C, Tardieu F, Cabrera-Bosquet L. 2017. A robot-assisted imaging pipeline for tracking the growths of maize ear and silks in a high-throughput phenotyping platform. *Plant Methods* **13**:1–12.

Cabrera-Bosquet L, Fournier C, Brichet N, Welcker C, Suard B, Tardieu F. 2016. High-throughput estimation of incident light, light interception and radiation-use efficiency of thousands of plants in a phenotyping platform. *The New Phytologist* **212**:269–281.

Canola Council of Canada. 2020. Canola growth stages. https://www.canolacouncil.org/canola-encyclopedia/growth-stages/ (3 May 2021).

Das Choudhury S, Bashyam S, Qiu Y, Samal A, Awada T. 2018. Holistic and component plant phenotyping using temporal image sequence. *Plant Methods* **14**:1–21.

Drouet JL, Pagès L. 2003. GRAAL: a model of GRowth, Architecture and carbon ALlocation during the vegetative phase of the whole maize plant: model description and parameterisation. *Ecological Modelling* **165**:147–173.

Edwards J. 2011. *Canola growth and development*. Orange, NSW, Australia: Department of Primary Industries.

Erickson RO, Michelini FJ. 1957. The plastochron index. *American Journal of Botany* **44**:297–305.

Fahlgren N, Gehan MA, Baxter I. 2015. Lights, camera, action: high-throughput plant phenotyping is ready for a close-up. *Current Opinion in Plant Biology* **24**:93–99.

Federl P, Prusinkiewicz P. 1999. Virtual laboratory: an interactive software environment for computer graphics. *Proceedings of Computer Graphics International* **1999**:93–100.

Ferraro P, Godin C, Prusinkiewicz P. 2005. Toward a quantification of self-similarity in plants. *Fractals* **13**:91–109.

Fournier C, Andrieu B. 1998. A 3D architectural and process-based model of maize development. *Annals of Botany* **81**:233–250.

Fournier C, Andrieu B. 1999. ADEL-maize: an L-system based model for the integration of growth processes from the organ to the canopy. Application to regulation of morphogenesis by light availability. *Agronomie* **19**:313–327.

Furbank RT, Tester M. 2011. Phenomics—technologies to relieve the phenotyping bottleneck. *Trends in Plant Science* **16**:635–644.

Galbraith C, Prusinkiewicz P, Davidson C. 1999. Goal oriented animation of plant development. In: Tigges M, Baranoski G, eds. Proceedings of the 10th Western Computer Graphics Symposium. Banff, AB, Canada, 19–32. http://algorithmicbotany.org/papers/goal-oriented-animation.html

Godin C, Costes E, Sinoquet H. 1999. A method for describing plant architecture which integrates topology and geometry. *Annals of Botany* **84**:343–357.

Godin C, Guédon Y, Costes E, Caraglio Y. 1997. Modeling and analysing plants with an AMAPmod software. In: Michaloewicz MT, ed. *Plants to ecosystems: advances in computational life sciences*. Collingwood, Australia: CSIRO, 53–84.

Guo Y, Ma Y, Zhan Z, Li B, Dingkuhn M, Luquet D, De Reffye P. 2006. Parameter optimization and field validation of the functional-structural model GREENLAB for maize. *Annals of Botany* **97**:217–230.

Guo J, Xu S, Yan DM, Cheng Z, Jaeger M, Zhang X. 2020. Realistic procedural plant modeling from multiple view images. *IEEE Transactions on Visualization and Computer Graphics* **26**:1372–1384.

Hanan JS, Room PM. 1997. Practical aspects of virtual plant research. In: Michaloewicz MT, ed. *Plants to ecosystems: advances in computational life sciences*. Collingwood, Australia: CSIRO Publishing, 28–44.

Higgs N, Leyeza B, Ubbens J, Kocur J, Van Der Kamp W, Cory T, Eynck C, Vail S, Eramian M, Stavness I. 2019. ProTractor: a lightweight ground imaging and analysis system for early-season field phenotyping In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. Los Alamitos, CA, USA: IEEE, 2629–2638.

Huxley JS. 1925. Constant differential growth-ratios and their significance. *Nature* **114**:895–896.

Huxley JS, Teissier G. 1936. Terminology of relative growth. *Nature* **137**:780–781.

Jiang Y, Li C. 2020. Convolutional neural networks for image-based high-throughput plant phenotyping: a review. *Plant Phenomics* **2020**:4152816.

Jullien A, Mathieu A, Allirand JM, Pinet A, de Reffye P, Cournède PH, Ney B. 2011. Characterization of the interactions between architecture and source-sink relationships in winter oilseed rape (*Brassica napus*) using the GreenLab model. *Annals of Botany* **107**:765–779.

Jullien A, Mathieu A, Ney B, Qi R, Allirand JM, Richard-Molard C. 2012. Use of a structure-function plant model to assess the morphogenetic plasticity. How does variation in phyllochron modify plant growth and development of *Brassica napus* in the GreenLab model? In: Kang M, Dumont Y, Guo Y, eds. Proceedings—2012 IEEE 4th International Symposium on Plant Growth Modeling, Simulation, Visualization and Applications, PMA 2012. New York, USA: IEEE, 180–187.

Karwowski R, Prusinkiewicz P. 2003. Design and implementation of the L+C modeling language. *Electronic Notes in Theoretical Computer Science* **86**:134–152.

Khan NA, Lyon OAS, Eramian M, McQuillan I. 2020. A novel technique combining image processing, plant development properties, and the Hungarian algorithm, to improve leaf detection in maize. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. Los Alamitos, CA, USA: IEEE, 330–339.

Kurth W. 1994. *Growth grammar interpreter GROGRA 2.4: a software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modelling*. Göttingen, Germany: Forschungszentrum Waldökosysteme der Universität Göttingen.

Licklider JCR. 1960. Man-computer symbiosis. *IRE Transactions on Human Factors in Electronics* **HFE-1**:4–11.

Lindenmayer A. 1968a. Mathematical models for cellular interactions in development. I. Filaments with one-sided inputs. *Journal of Theoretical Biology* **18**:280–299.

Lindenmayer A. 1968b. Mathematical models for cellular interactions in development. II. Simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology* **18**:300–315.

Lindenmayer A. 1971. Developmental systems without cellular interactions, their languages and grammars. *Journal of Theoretical Biology* **30**:455–484.

Lintermann B, Deussen O. 1999. Interactive modeling of plants. *IEEE Computer Graphics and Applications* **19**:56–65.

Lück J, Lück HB, Bakkali M. 1990. A comprehensive model for acrotonic, mesotonic and basitonic branchings in plants. *Acta Biotheoretica* **38**:257–288.

Ma Y, Wen M, Guo Y, Li B, Cournède PH, de Reffye P. 2008. Parameter optimization and field validation of the functional-structural model GREENLAB for maize at different population densities. *Annals of Botany* **101**:1185–1194.

McGregor DI. 1981. Pattern of flower and pod development in rapeseed. *Canadian Journal of Plant Science* **61**:275–282.

Měch R, Prusinkiewicz P. 1996. Visual models of plants interacting with their environment. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques—SIGGRAPH'96. New York, NY: ACM Press, 397–410. https://dl.acm.org/doi/proceedings/10.1145/237170

Mercer L, Prusinkiewicz P, Hanan J. 1990. Concept and design of a virtual laboratory. In: Proceedings of Graphics Interface'90. Toronto, ON, Canada: Canadian Information Processing Society, 149–155.

Miao C, Hoban TP, Pages A, Xu Z, Rodene E, Ubbens J, Stavness I, Yang J, Schnable JC. 2019. Simulated plant images improve maize leaf counting accuracy. *bioRxiv*.

Moulia B, Sinoquet H. 1993. Three-dimensional digitizing systems for plant canopy geometrical structure: a review. In: Varlet-Grancher C, Bonhomme R, Sinoquet H, eds. *Crop structure and light microclimate: characterization and applications*. Paris: INRA, 183–193.

Mündermann L. 2003. *Inverse modeling of plants*. *PhD Thesis*, University of Calgary, Calgary, AB, Canada.

Mündermann L, Erasmus Y, Lane B, Coen E, Prusinkiewicz P. 2005. Quantitative modeling of Arabidopsis development. *Plant Physiology* **139**:960–968.

Niklas KJ. 1994. *Plant allometry: the scaling of from and processes*. Chicago, IL: University of Chicago Press.

Polowick PL, Sawhney VK. 1986. A scanning electron microscopic study on the initiation and development of floral organs of *Brassica napus* (cv. Westar). *American Journal of Botany* **73**:254–263.

Prusinkiewicz P. 1986. Graphical applications of L-systems. In: Proceedings of Graphics Interface'86/Vision Interface'86. Toronto, ON, Canada: Canadian Information Processing Society, 247–253.

Prusinkiewicz P. 1998. Modeling of spatial structure and development of plants: a review. *Scientia Horticulturae* **74**:113–149.

Prusinkiewicz P. 2004a. Art and science for life: designing and growing virtual plants with L-systems. *Acta Horticulturae* **630**:15–28.

Prusinkiewicz P. 2004b. Self-similarity in plants: integrating mathematical and biological perspectives. In: Novak MM, ed. *Thinking in patterns: fractals and related phenomena in nature*. Singapore: World Scientific, 103–118.

Prusinkiewicz P, Cieslak M, Ferraro P, Hanan J. 2018. Modeling plant development with L-systems. In: Morris RJ, ed. *Mathematical modelling in plant biology*. Cham, Switzerland: Springer, 139–169.

Prusinkiewicz P, Hammel M, Hanan J, Měch R. 1997. Visual models of plant development. In: Rozenberg G, Salomaa A, eds. *Handbook of formal languages*. Berlin: Springer-Verlag, 535–597.

Prusinkiewicz P, Hanan J. 1990. Visualization of botanical structures and processes using parametric L-Systems. In: Thalmann D, ed. *Scientific visualization and graphics simulation*. Chichester: Wiley, 183–201.

Prusinkiewicz P, Hanan J, Měch R. 2000. An L-system-based plant modeling language. In: Nagl M, Schürr A, Münch M, eds. *Applications of Graph Transformations with Industrial Relevance.*

AGTIVE 1999. Lecture Notes in Computer Science. Berlin: Springer, 395–410.

Prusinkiewicz P, Karwowski R, Lane B. 2007. The L+C plant-modelling language. In: Vos J, Marcelis L, De Visser P, Struik P, Evers J, eds. *Functional-structural plant modelling in crop production*. Dordrecht: Springer, 27–42.

Prusinkiewicz P, Lindenmayer A. 1990. *The algorithmic beauty of plants*. New York, NY: Springer-Verlag.

Prusinkiewicz P, Mündermann L, Karwowski R, Lane B. 2001. The use of positional information in the modeling of plants. In: *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001*. Los Angeles, CA: Association for Computing Machinery, 289–300. https://dl.acm.org/doi/10.1145/383259.383291

Prusinkiewicz PW, Remphrey WR, Davidson CG, Hammel MS. 1994. Modeling the architecture of expanding *Fraxinus pennsylvanica* shoots using L-systems. *Canadian Journal of Botany* **72**:701–714.

Quan L, Tan P, Zeng G, Yuan L, Wang J, Kang SB. 2006. Image-based plant modeling. *ACM Transactions on Graphics*. **25**:599–604.

Richards OW, Kavanagh AJ. 1945. The analysis of growing form. In: Le Gros Clark W, Medawar P, eds. *Essays on growth and form presented to d'Arcy Wentworth Thompson*. Oxford: Clarendon Press, 188–230.

Room P, Hanan J, Prusinkiewicz P. 1996. Virtual plants: new perspectives for ecologists, pathologists and agricultural scientists. *Trends in Plant Science* **1**:33–38.

Singh AK, Ganapathysubramanian B, Sarkar S, Singh A. 2018. Deep learning for plant stress phenotyping: trends and future perspectives. *Trends in Plant Science* **23**:883–898.

Tardieu F, Cabrera-Bosquet L, Pridmore T, Bennett M. 2017. Plant phenomics, from sensors to knowledge. *Current Biology* **27**:R770–R783.

Ubbens J, Cieslak M, Prusinkiewicz P, Stavness I. 2018. The use of plant models in deep learning: an application to leaf counting in rosette plants. *Plant Methods* **14**:1–10.