Labelled Point Cloud Generation from L-System Models

Ayan Chaudhury
Indian Institute of Technology Kharagpur

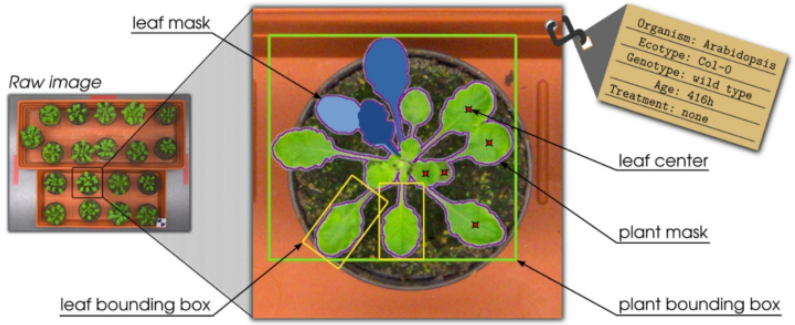
Need for labelled datasets

- Performance evaluation of an algorithm
- Comparison with other state-of-the-art
- For training the data-hungry deep learning models

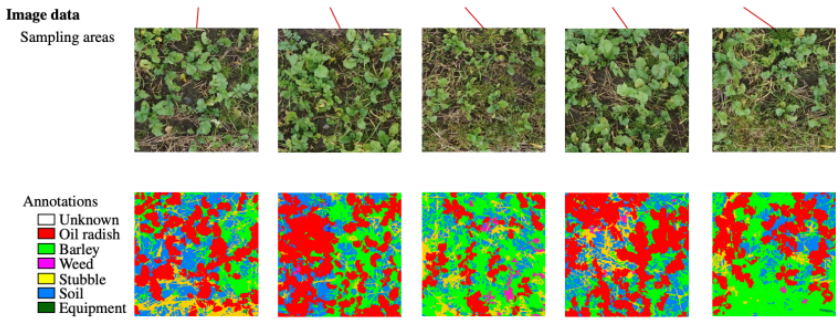
There is lack of annotated data in the plant phenotyping community

The availability of 2D datasets

- Plenty of datasets have been proposed in last few years
- Arabidopsis rosette dataset [PRL'16]



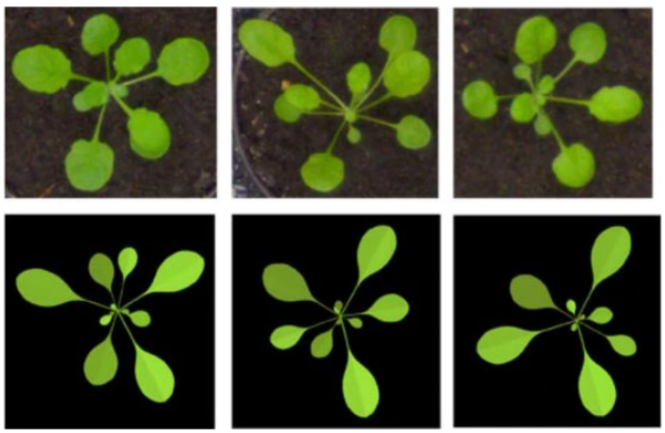
- Oil radish dataset for semantic segmentation [CVPPP'19]



Attempts in generating synthetic 2D image data

- Using Generative Adversarial Network (GAN) [CVPPP'17, '18, '19]

- Using Procedural Model (L-system) [Ubbens et al. Plant Methods'18]

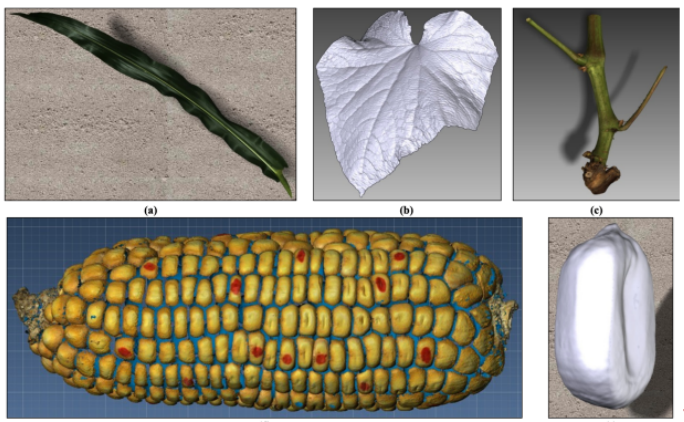


The availability of 3D point cloud datasets

Almost none !!!

- Very recent rosebush dataset [Dutagaci et al. Plant Methods'20]

- Very recent rosebush dataset [Wen et al.'17]



Challenges

- Challenges of creating 3D point cloud dataset:
  - Expensive hardware to perform scanning
  - Problem with occlusion while scanning
  - Still needs multi view reconstruction for full 3D model
  - SfM methods are often noisy
  - Tedious manual labelling of the data (chances of error)

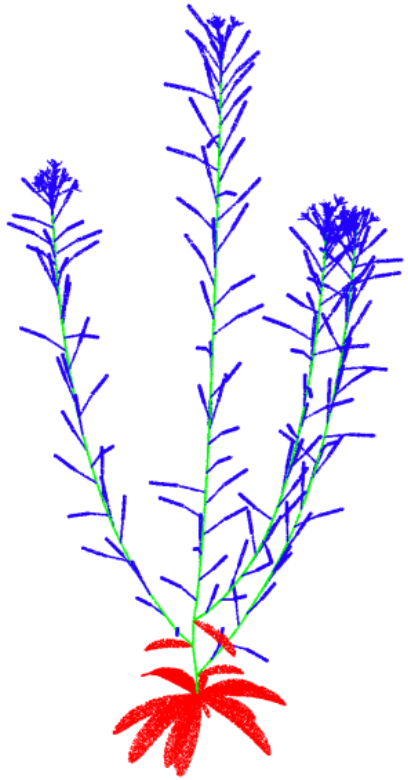
There is a need to find an alternate solution

Virtual Plants

- Virtual Plant models can come to the rescue:
  - No need of real plants
  - Can generate large number of diversities
  - No need of manual annotation
  - No chance of labelling error (100% reliable)

Virtual Plants

(Surprisingly) No attempts have been made to create point cloud from virtual models !!



Getting started from Virtual Plants

- What do we need to create labelled point clouds?
  - A virtual plant model of the desired species is available (in L-Py)
  - We know the lstrings of the model (even if we don't have it, can be printed easily)

That's all what we need!

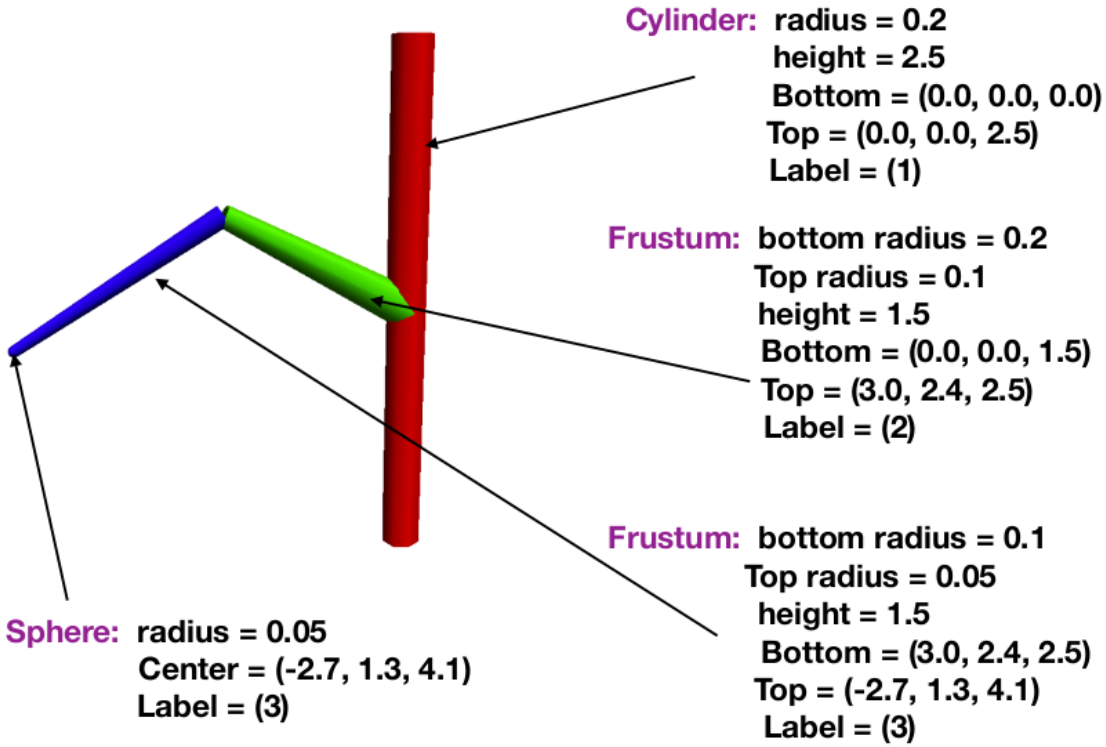
Labelling technique

- Consider the lstring of a model
  - I, I, I, @O, Leaf, I, I, @O, Leaf, I, I, ...
- Keep a dictionary of possible labels in your model
  - Dict = {"L": 1, "I":2, "A": 3, "B": 4, "@O": 5, "Petals": 6, "Leaf": 7, "[": 8, "Carpel": 9, "@Gc": 10, ... }
- Then simply assign label the lstring from the dictionary (start from 1)
- If you don't have a dictionary, then default labells will be assigned
- You can also write your own function for the labels

Given any virtual plant model in L-Py, it is possible to generate labelled point clouds from it

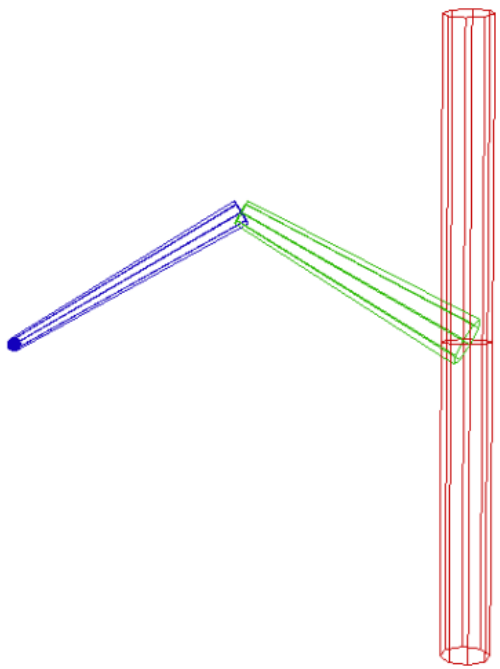
Labelling steps

- Extract Primitives (along with their labels & geometry)



Labelling steps

- Tesselate each Primitive & Obtain triangles (along with their labels & geometry)



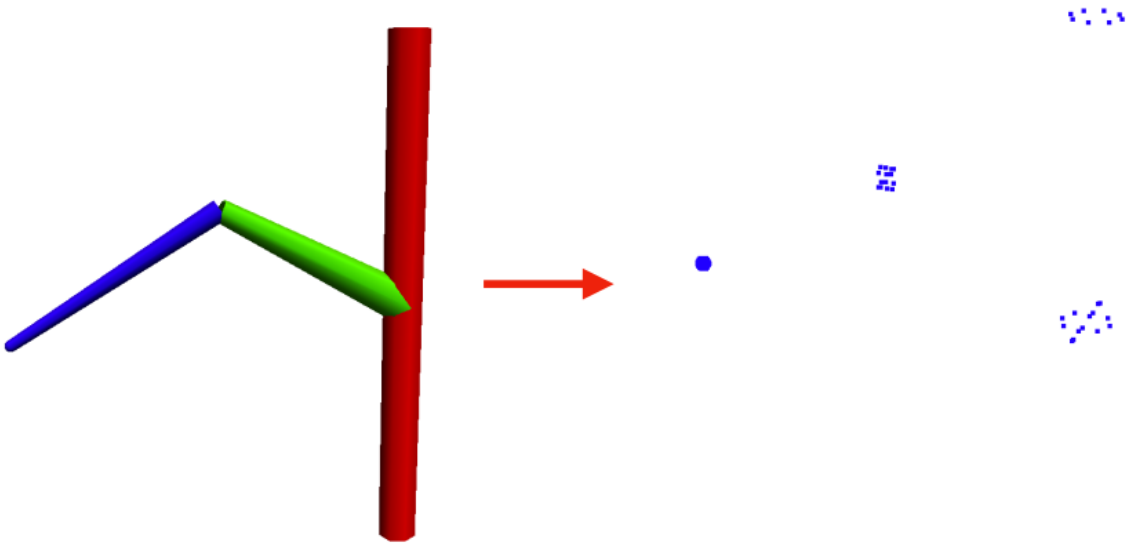
T1 (v1, v2, v3) -> Label (1)
T2 (v1, v3, v4) -> Label (1)
.
.
.
Tn (v54, v60, v59) -> Label (3)

These are “local” vertices

Create a vertex list of global vertices

Labelling steps

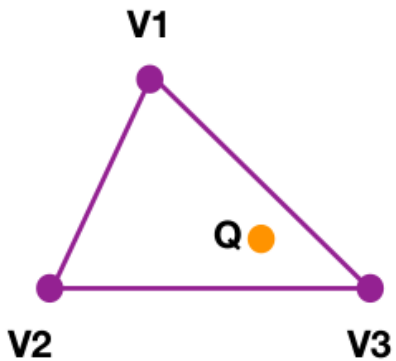
- At this point, the mesh vertices (point cloud) looks like the following



- Could use successive cylinders, but still that's not sufficient
- We need to do point sampling

Point sampling

We exploit the idea of barycentric coordinates



For the case of triangle

Given the three vertices, any random point can be expressed as:

$$\mathbf{Q} = \alpha_1 \mathbf{V}_1 + \alpha_2 \mathbf{V}_2 + \alpha_3 \mathbf{V}_3$$

Where

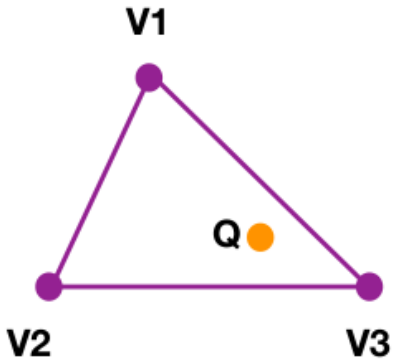
$$\alpha_1, \alpha_2, \alpha_3 \geq 0$$

$$\alpha_1 + \alpha_2 + \alpha_3 = 1$$

This simple idea can be used to generate as many points as we wish!!

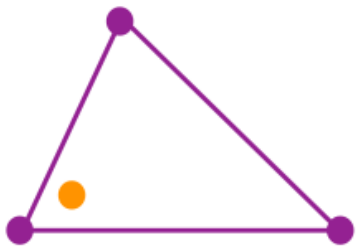
Point sampling

Random Point Sampling:

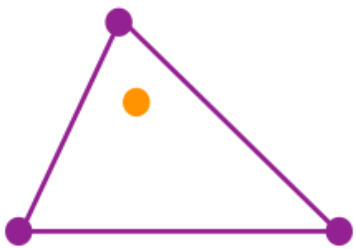


1. Randomly select a triangle $T_i \in \mathcal{T}$
2. $\{\mathbf{v}_1^i, \mathbf{v}_2^i, \mathbf{v}_3^i\} \leftarrow vertex(T_i)$
3. $\alpha_1 \leftarrow rand(0,1)$
4. $\alpha_2 \leftarrow rand(0,1)$
5. **If** $(\alpha_1 + \alpha_2) > 1$
   $\alpha_1 \leftarrow 1 - \alpha_1$
   $\alpha_2 \leftarrow 1 - \alpha_2$
6. $\alpha_3 \leftarrow 1 - (\alpha_1 + \alpha_2)$
7. $\mathbf{Q} \leftarrow \alpha_1 \mathbf{v}_1^i + \alpha_2 \mathbf{v}_2^i + \alpha_3 \mathbf{v}_3^i$

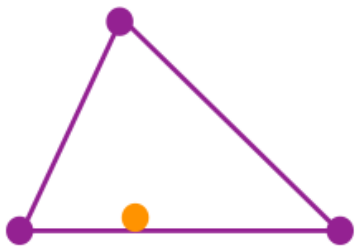
$\alpha_1 = 0.21,\ \alpha_2 = 0.42$



$\alpha_1 = 0.76,\ \alpha_2 = 0.15$

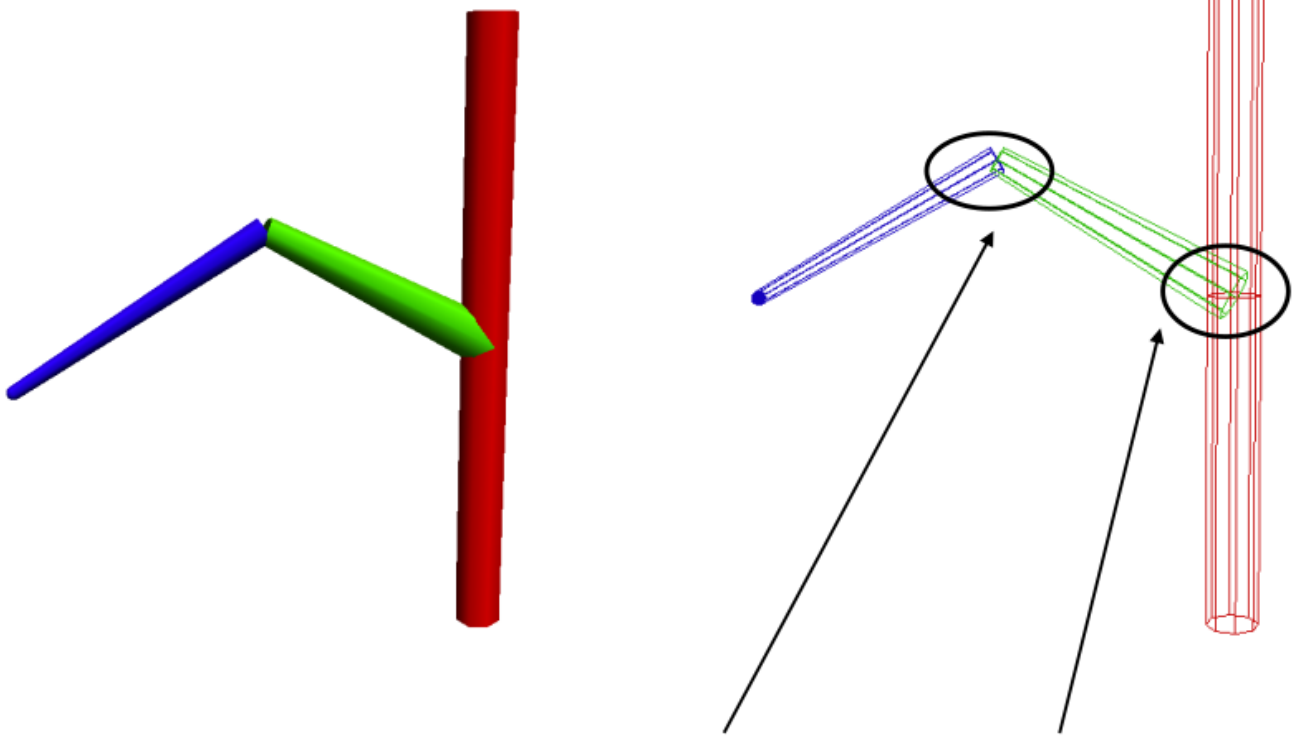


$\alpha_1 = 0.33,\ \alpha_2 = 0.49$



...

Insideness Testing

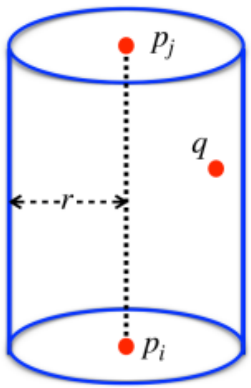


In real plants, this does not happen!

Insideness Testing

We need to consider volumetric primitives:

1. Cylinder



$$(\vec{q} - \vec{p_i}) \cdot (\vec{p_j} - \vec{p_i}) \geq 0,$$
$$(\vec{q} - \vec{p_j}) \cdot (\vec{p_j} - \vec{p_i}) \leq 0.$$

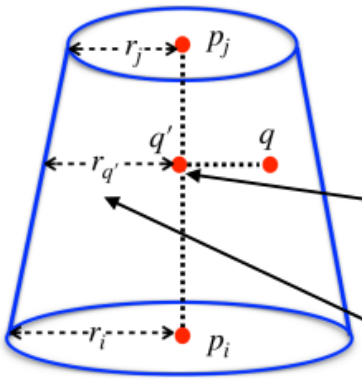
The point q lies within the two circular facets of the cylinder

$$\frac{|(\vec{q} - \vec{p_i}) \cdot (\vec{p_j} - \vec{p_i})|}{|(\vec{p_j} - \vec{p_i})|} \leq r$$

The point q lies inside the cylinder

Insideness Testing

2. Frustum



(First check if the point q lies within the two circular facets of the frustum)

$$\vec{q}' = \vec{p}_i + \left(\frac{(\vec{q} - \vec{p}_i) \cdot (\vec{p}_j - \vec{p}_i)}{(\vec{p}_j - \vec{p}_i) \cdot (\vec{p}_j - \vec{p}_i)} \right) * (\vec{p}_j - \vec{p}_i) .$$

$$r_{q'} = \frac{\mathbf{dist}(\vec{p}_i, \vec{q}')}{\mathbf{dist}(\vec{p}_i, \vec{p}_j)} * r_i + \frac{\mathbf{dist}(\vec{p}_j, \vec{q}')}{\mathbf{dist}(\vec{p}_i, \vec{p}_j)} * r_j$$

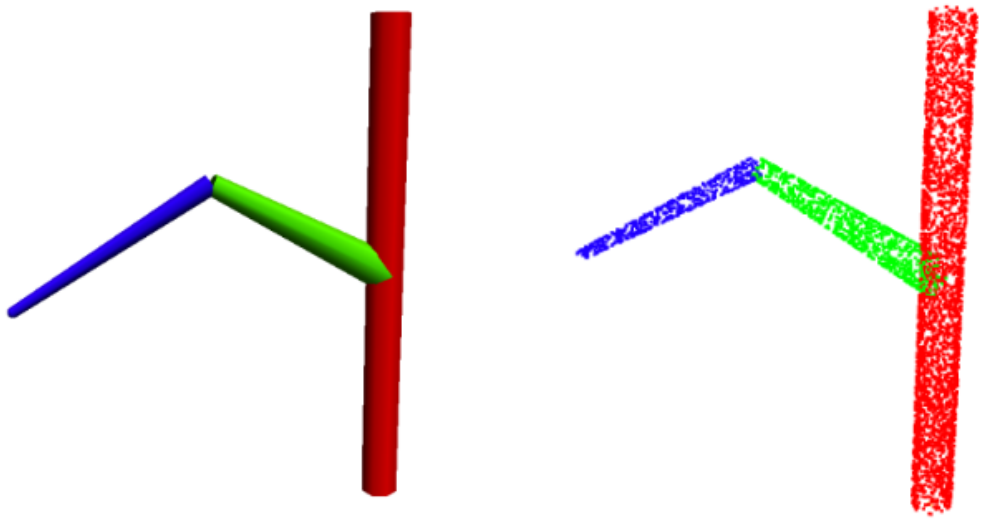
$$d = \mathbf{dist}(\vec{q}, \vec{q}')$$

If $d > r_{q'}$ then the point is outside the frustum

3. Sphere

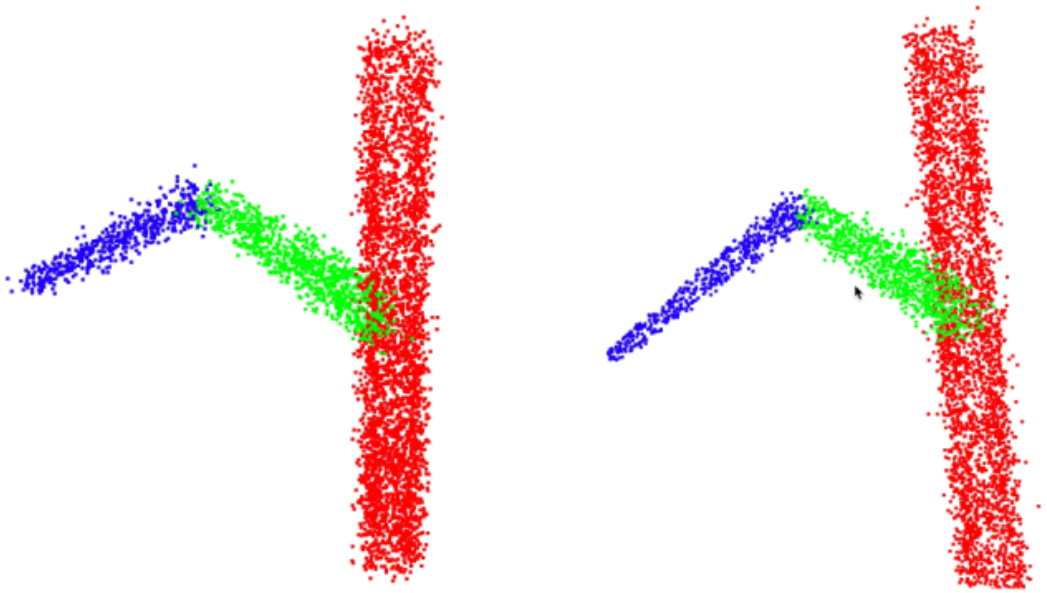
Simply check if: $\sqrt{(x_c - x_i)^2 + (y_c - y_i)^2 + (z_c - z_i)^2} \leq r$

Results



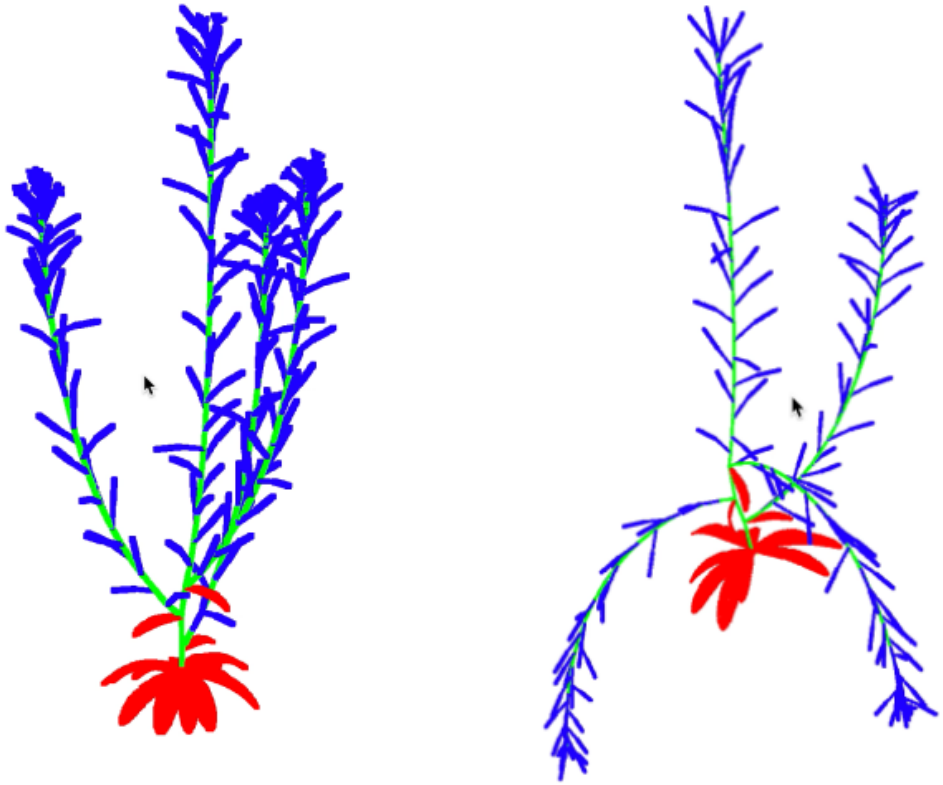
Results

It is possible to add Gaussian noise, as well as device specific noise (noise increases with the distance from the scanning position)



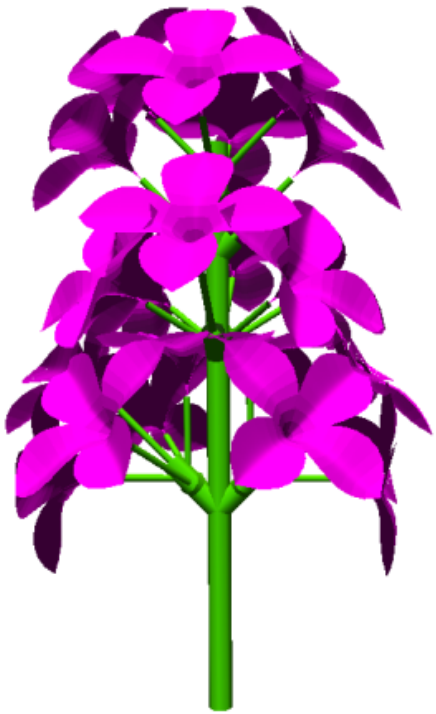
Results

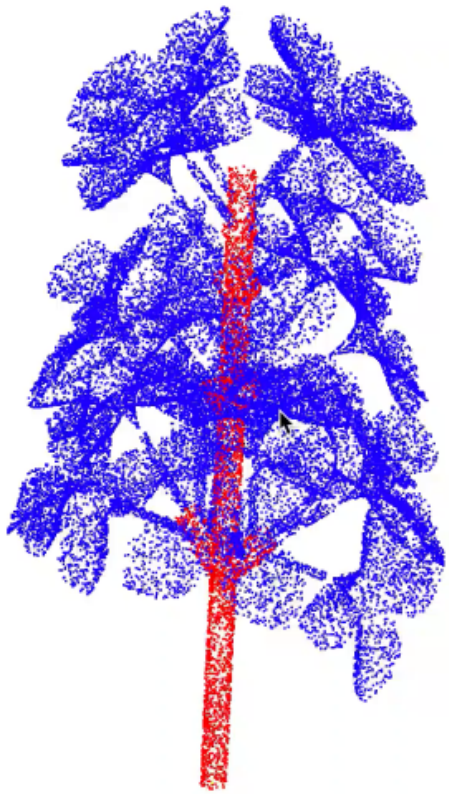
Labelled Arabidopsis



Results

Artificial plant: Lilac





Now let's see the demo of the software library