LaTeX TITLE HERE

First Author
Institution1
Institution1 address
firstauthor@i1.org

Second Author
h Institution2
First line of institution2 address
secondauthor@i2.org

December 4, 2019

Abstract

1 Introduction

Plant architecture is a complex geometrical object. Due to both genetic and environmental variations, organs evolve in patterns which present great diversity both between species and among individuals of the same species. Automated reconstruction of plant structure from lab or in-field acquisitions remains a challenge in computer vision (ref). The most common method for complex measurements of plant structure remains handmade measurements and image analysis is a major bottleneck in plant phenotyping [**?**]. The application of automated processing of plant structure are many: precise quantifying of biomass and yield for agricultural crops, estimation of environmental response of crops [**?**], mapping genotypes to phenotypes, estimating growth parameters of species, among other. (Find many refs)

Deep convolutional networks have shown – see for example [**?**] – to be the state of the art method for many classification tasks. Several networks ([**?**], [**?**]) leverage huge classification datasets (ImageNet) to provide pixel by pixel semantic segmentation. They have been successfully used for plant organ segmentation [**?**], but are still limited to simple case with a very small diversity in test datasets. Using simulated data to augment training datasets is a method which have shown to improve neural network performance in many fields, e.g [**?**], [**?**].

Plant architecture is a well studied research topic, and many generative model of plant architecture have been developped in the last decades. Lindenmayer systems are rewriting systems specifically developped to model plant growth. They can be used to model arbitrary complex model of plants [**?**].

In this work, we propose to use plant models to train convolutional neural network for identification of different plant organs. The specific target application of our method is the identification of organs of the model plant Arabidopsis thaliana. We describe a data augmentation model using plant models and HDRI pictures to produce ground truth images, as well as a simple method for fine-tuning on real word data. We then present qualitative results in various acquisition condition to assert the robustness of our method.

2 Related works

3 Material and Methods

Virtual plants. The virtual plants were designed with the Python library OpelAlea [] developped by french research institutes of biology and mathematics to provide tools for plant architecture modelling. 3D meshes of Arabidopsis Thaliana were designed with the L-Py library from OpenAlea, which is a python implementation of L-system. L-systems were developped in 1968 by Aristid Lindenmayer [] to model plant growth. It is a generative grammar that allows to grow a virtual plant using symbols, shapes and constraints derived for plant growth observation.

Formally it is called a rewriting system, or formal grammar. It comprises:

- A vocabulary V containing the *modules* of the system. For plant generation it will represent an architectural element of the plant (apex, internode, leaf) and associated parameters (age, length, etc)
- An initial *axiom* or state s_0 corresponding to the virtual plant at t_0. It is a string of elements from the vocabulary.
- A set of *production* rules to iterate in order to model the growth. They will be applied in parallel to each variable element from the string of the previous state. They are composed of a *predecessor*, to identify the elements that will be replaced by a *successor*.

The strings can then be represented graphically in 3D using OpenAlea's PlantGL (Figure1). The A. Thaliana models comprised 5 different types of organs: fruit, stem, peduncle, leaf and flower.

Figure 1: Left: Example of an L-Py system with two modules, right: Visual representation of the first five steps of the L-system comprising apex (green dots) and internodes (brown sticks) in the vocabulary. Figure from [**?**]

Virtual scanner. An API was implemented to generate and visualise the virtual plants in Blender. The 3D models are loaded in a virtual environment and virtual cameras are used to take pictures of the model (Figure 3). The API was used to simulate a virtual scanner and generate images for the training of neural networks. The focal, position, pan and tilt of the cameras are tunable and allow to generate images from different views. It is also possible to move the plant in the scanner. The plant is placed in a virtual background. It is a 360° image and rendering it in blender reproduces the scene lightening in order to increase the complexity of the images acquired with the virtual scanner. Several sets of virtual backgrounds were used, without limitations on the lightning environement (night, daylight, sunset) or the type of scene (indoor, outdoor).

The virtual labels are acquired by rendering the plant material by material, with one material per organ (Figure 3). The organs considered for *A. Thaliana* are leaf, stem, flower, fruit and peduncle. In the pinhole model [] we use for 3D to 2D projection, a whole line in 3D projects onto a single pixel (Figure 2). As a consequence, a single pixel can belong to several classes depending on the organs crossed by the line. The labelling method takes this plurality into account by generating a ground truth image for each class.

Figure 2: Illustration of 2D overlapping due to perspective projection

Figure 3: Example of virtual training images and all labels

TODO: add change text after Christophe new plants, add random colors

3D round truth. The 3D virtual plants were used to train and evaluate the 3D reconstruction. The virtual mesh is regularly subsampled with CloudCompare [**?**] in order to get a density matching the voxel density in the volume to carve (see space carving section for explanations). Then this 3D point cloud is voxellized and each voxel of the volume to carve acquires an organ class, or class 0 (background) if it is not part of the plant. Then the vector is saved in a sparse representation to save memory. For the trainings, the data generator reads the sparse vector and encodes it in a dense representation to compare it to the predicted classes. It is needed to densify it because PyTorch can't backpropagate easily through a sparse representation.

Image segmentation. The objective of image segmentation is to convert an input image in a stack of probability map of the same dimension as the image. For C classes there will be C probability maps, each corresponding to one class. The value of a pixel in the probability map of class k is the probability that this pixel belongs to the organ-type k in the original image.

The image segmentation was performed using segmentation convolutional neural network [**?**]. Such segmentation networks are based on a contracting structure from

an image to a low dimension feature space, which encodes the content of the image. It is branched to a symmetric expanding structure that translates the information from the feature space to an image similar to the original one, but with only the content of interest reconstructed. The contracting structure is directly inspired from classification neural networks, and made of convolution layers, non-linearities and down-pooling layers. The convolution kernels allow to filter the information to emphasise the content, and the down-pooling allows to reduce the dimension of the information. The expanding structure reproduces the path in the other direction, with up-pooling layers and convolutions. The spatial information is reinjected to reconstruct the image properly, by providing the down-pooling coordinates from the contracting phase.

Figure 4: Example of segmentation network: the U-Net architecture [**?**]

The structure tested was inspired from U-Net [**?**], however the contracting structure, or encoder, was replaced by a classification network trained on ImageNet, and with 6 classes to segment. This classification structure has already been trained to encode the semantic representation of an image in the latent space. We tested ResNet [**?**] which is a deep neural network where inputs from previous layers are regularly reinjected into deeper layers in order to maintain the geometry and avoid vanishing gradients [**?**].

Deep learning training The network was trained with images from the virtual scanner. TODO(Describe the path, dimension of the images). The dataset comprised TODO(Number of images). To make the network robust to images and lignting conditions that are not in the dataset, we artificially augment the dataset by adding TODO(add gaussian noise, change contrast, rotation, déformation, more?). The dataset was split into 3 sets:

- a training set to train the network (50% of the dataset)
- a validation set on which the network is not trained and is used to evaluate and compare the different networks (25% of the dataset)
- a test set used at the very end on the selected architecture (25% of the dataset)

To evaluate the segmentation of the images, a combination of metrics were used. As it is a multiclass problem, a sigmoid is applied to each output in order to contain the numerical range of the predictions. First crossentropy was used, it is a per-pixel metric. It represents the uncertainty of the prediction compared to the ground-truth. If the class of a pixel has a very low predicted probability, it will highly penalise the loss. If the probability is close to one, the contribution to the loss is close to zero. The notion of entropy represents this discrepancy between the ground truth distribution and the predicted distribution. It is naturally translated at the mathematical level with the negative of the logarithm.

$$L = -\sum_{i}^{n}\sum_{k}^{C} y_{i,gt}^{k} \ln(y_{\text{i,pred}}^{k}) \tag{1}$$

Where n is the number of pixels in the image, C the number of possible classes, $y_{gt}^{k} = 1$ if pixel i is in class k, 0 otherwise, and $y_{\text{i,pred}}^{k} = p(\text{label}(y_{\text{i,gt}}) = k)$.

The second loss we used was the Dice coefficient, which theoretically writes as:

$$s = 1 - \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{C} y_{\text{i,gt}} y_{\text{i,pred}}^{k} \tag{2}$$

This coefficient compares the number of right predictions to the number of samples, for each class. In practice, the product of the ground truth by the predictions is summed, so that only the prediction of the right class contributes to the sum. For example for point n, the class is k, and the prediction for class k is p_k, the loss for point n will be $1 - \frac{2p_k}{2} = 1 - p_k$. The total loss is computed vector-wise, and lies between 0 and 1.

We used the mean of these two losses for the training, as the crossentropy has more stable gradients, but the Dice loss is what we want to minimise, and is less sensitive to class imbalance.

Fine-tuning. Our network was trained on virtual Arabidopsis Thaliana model but aims at reconstructing real plants. Therefore when the virtual plants fail at reproducing the complexity of real plants, the network can perform poorly. the network also fails when confronted to other plant species with very different traits. Therefore we conceived a simple interface that allows the user to manually label a few images of interest (three images is enough), then run the training of the network on this small dataset. This way, the network will be able to segment similar images with correct classes.

Space Carving. We used space-carving to reconstructi the palnts in 3D from the 2D images. The space carving [**?**] is a photogrammetric approach which uses pictures of an object from different point of views to reconstruct an object in 3D. To simplify the explanation we will take the example of the original space carving paper [**?**]. Let the objective be to reconstruct a sculpture in 3D. The cameras circle a volume of interest that contains the cube and take N pictures at regular intervals. Each picture will be preprocessed, with a masking step: on each image the mask will give the probability that the pixel belongs or not to the sculpture. Then, the surrounded volume is computed virtually, and each voxel of the volume is projected onto each masked view. If a voxel has projected onto the sculpture for each 2D-view, it is part of the 3D-sculpture. Otherwise it is part of the background: voxel after voxel, the volume is carved into the shape of the object. The main limitation of this technique is that the problem is ill-posed and the reconstructed object will be a *photo hull* [**?**] of the desired object, that is an object which is the union of all objects derivable from the multi-view observation. It can cause problems for highly curved or hollow objects, but in the case of plants it is not a major limitation.

4 Results and Discussions

5 Conclusion and perspectives