Reichman University

Efi Arazi School of Computer Science

M.Sc. program

Image Based 4D Phenotyping

By

Eyal Friedman

Supervisor: Prof. Yael Moses

Final project, submitted in partial fulfillment of the requirements
for the M.Sc. degree, School of Computer Science
Reichman University (The Interdisciplinary Center, Herzelia)

August 2022

This work was carried out under the supervision of Prof. Yael Moses from the Efi Arazi School of Computer Science, Reichman University.

Abstract

In modern agriculture, measuring phenotypic traits helps breeders and growers to monitor plant growth and to increase yield production. Traditional phenotyping requires intensive manual work, partially being intrusive. In this paper we developed an accessible solution for calculating phenotype for use in plant sciences using multi-views images and prior assumption to reconstruct a 3D model of the plant.

The method consists of an algorithm for a 3D mesh computation of one or more objects, where the objects we considered were leaves. The mesh is computed by minimizing an energy function that reflects the computed leaves' shade with their appearance in the given set of images. We follow and extend the SoftRas work which was developed for a single object using a differential rendering function. This project extends and modifies the the energy function to deal with severe occlusions by using priors on the volume and manifold intersections. The main challenges include designing the appropriate energy terms for an efficient computation using GPUs and parallel computation, and its derivatives can be computed with respect to the mesh nodes. In addition, we explored few techniques to improve run-time and high resolution scene by introducing a resolution pyramid concept to deal with high resolution images as an input for the algorithm and the mesh complex we are expecting as a result. In this project we provide basic proof of concepts with only a qualitative evaluation. A more comprehensive treatment requires thorough experimentation, and it is left for future work.

To the best of our knowledge, there is only one article that deals with 3D reconstruction of a plant based on Differentiable Rendering [15], but it is based on neural network training learned on a large number of images and activated on a single image. As a result, there are no good reconstruction capabilities on parts that are not visible in the image.

Contents

Abstract - Hebrew (not in this file)

The hebrew version of- This work was carried out under the supervision of Prof. Yael Moses from the Efi Arazi School of Computer Science, Reichman University.

Hebrew title page as directed.

1 Introduction

Phenotyping is the activity and process of determining, analyzing or predicting all or parts of an organism [16]. In a wider view, phenotyping can refer to the set of methodologies and protocols used to measure plant growth, architecture, and composition with certain accuracy and precision at different scales of organization, from organs to canopies [7]. Phenotyping plays an essential role in plant science supporting decisions such as breeding [3, 8, 25], monitoring [11, 23], and cultivating [12]. In recent years, rapid improvements in research methods yield reliable, accurate, and autonomous phenotyping technologies, however there is still much room for improvement and advancement in the field. The aim of this project is to build the first building block for developing an accessible and efficient approach to delicate and accurate modeling of plants' parts in the dimensions (3D) and their evolvement along a time-line (4D), focusing on complex greenhouse plants. We focused on the 3D reconstruction of two leaves as shown below and explained, as a first step toward this goal.

In the past, phenotype measuring and analyzing was tedious, expensive and time consuming. Today, technology plays an important role in phenotyping, particularly in the three-dimensional reconstruction of plants and key features in plants, without causing damage to the plant itself. However, plants are complex and changing systems. They grow in diverse natural environments, so reconstruction and segmentation of three-dimensional (3D) plants and long-term changes (4D) are usually more challenging than three-dimensional reconstruction tasks of other objects (e.g., restoration of faces, objects, scenes, etc. [5, 9]). Many studies (such as [21, 22]) as well as commercial companies (such as [1, 2]) attempt to solve the phenotype task automatically and efficiently.

Three-dimensional reconstruction of a plant from a single image is an ill-posed problem, due to the loss of the depth dimension in the image and the concealment of part of the plant by other segments of the plant or the surrounding. The problem of loss of depth dimension can be overcome by

prior assumptions, by using images taken from multi-views, or by using active sensors. **In this project we focused on developing accessible solutions for calculating phenotype for use in plant sciences using multi-views images and prior assumptions.** We provide basic proof of concepts with only a qualitative evaluation. A more comprehensive treatment requires thorough experimentation, and it is left for future work.

2 Related Works

Previous works of 3D object reconstruction and, in particular, plants [21], are based on methods of extracting 3D point clouds of the object and then post processing the results to produce a continuous 3D model of the plant or drawing conclusions from the point cloud statistics. The standard 3D models are 3D pixels (Voxels) or 3D mesh. The transition from cloud points to these models is not a trivial task and therefore its results are often imperfect [18]). In order to calculate the three-dimensional point cloud from images, it is necessary to match points of interest between a large number of images (image features) and enable triangulation. This process is called Structure-from-Motion (SFM). Advanced technological approaches use active sensors such as LiDAR, laser, built-in light (SR), depth cameras (RGB-D) or CT scan to overcome these challenges (see details in reviews [21, 22]). Most successes in the field today are received by active sensors, but these are usually limited to a controlled environment, and less available due to their price. Moreover, these technologies compute point cloud and additional non-trivial post-processing is required in order to compute the desired 3D shape in a form of a mesh or voxel representation, where many assumptions required since there is no topological connection between the points (just location, color, and density [4]).

Learning-based methods, particularly neural networks (CNN), have in recent years achieved very impressive results in many computer vision tasks, including 3D reconstruction [5, 6, 9, 10] and 3D motion (scene flow) [14, 20] . However, restoring a plant in 3D is considered a very challenging

task due to sever occlusions that cause each plant part to be visible only in a subset of images, as well as repetitive structures that make it difficult to match features in different images. The task of matching two meshes (4D) of an organ that been reconstructed in different time frames is even more challenging due to the organ development or natural physical displacement in space.

3 The Scope of this work

We developed a 3D phenotype technique using new approaches from the field of computer vision, which allow direct calculation of the desired 3D representation and eliminate the need for two-stage calculation based on a point cloud. The traingle-mesh representation contains the topological representation of the plant and allows to easily and accurately measure the properties of the plant such as the surface area of the leaves or leaves' angles with respect to the sun. The input we consider is a set of images taken by a collection of high-resolution standard RGB cameras, located in different locations relative to the plant. The availability and relatively low cost of these cameras, combined with the computational capabilities currently available, will allow accessibility of the proposed technology for research and optimize greenhouse parameters.

3D Reconstruction - The basic idea of our approach is an efficient method for deforming a template 3D mesh such that its projection to the set of images will be as similar as possible to the original images under few assumptions and regulations. Note that this approach is similar to the bundle adjustment (BA) method used for Structure-from-Motion of a point cloud. However, we propose to directly compute the 3D mesh rather than a point cloud, and perform the task efficiently. Of course each model of plant has a huge number of parameters that need to be calculated. Since the whole plant model is very complex, we intend to work on several resolution. At the grossest level, we would like to divide the plant into its parts by segmenting each of the images and matching

the segments between the images. At a more detailed level, we would like to compute the triangle-mesh of each of the segments. We assume at this project that the segmentation of two leaves is given. Previous attempt to compute directly the 3D mesh was proposed by Liu at al (2019) [13] as described next in Section 4. Our method allows to overcome some of the main limitations of their approach. In particular, extended the method to reconstruct more than a single object, we improved its efficiency and altered the loss function by addition of a volume regulator and an manifold intersection regulator. We describe our method in Section 5.

4 Differential Rendering

In the next section we describe the 3D reconstruction groundbreaking work, SoftRas, by Liu et al. (2019) [13]. As described in the paper, an efficient way of optimizing the model is carried out by computing a differential function with respect to the location of each of the mesh nodes, that reflects the similarity between the projected model and a given image for each of the given images (Soft Differentiable Rendering). This function allows differentiable rendering framework that is able to directly render colorized mesh. Doing so by using differentiable functions and back-propagate efficiently, the rendered gradient to mesh vertices and their attributes, and can operate on various forms of image representations, including silhouette and color images, while maintaining the topological structure of the mesh. In our project we consider only the silhouettes.

4.1 SoftRas - Soft Rasterizer

The key to image-based 3D reasoning is to find sufficient support from the pixels to the 3D properties. To obtain image-to-3D correlations, prior approaches to SoftRas mainly rely on the matching losses based on 2D key points/contours or shape/appearance priors. However, those approaches are either limited to task-specific domains or can only provide weak supervision due to the sparsity of the 2D features. In contrast, as the process of

producing 2D images from 3D assets, rendering relates each pixel with the 3D parameters by simulating the physical mechanism of image formulation. Hence, by inverting a renderer, one can obtain *dense* pixel-level supervision for *general-purpose* 3D reasoning tasks, which cannot be achieved by conventional approaches. **However, the rendering process is not differentiable in conventional graphic pipelines**. In particular, standard mesh renderer involves a discrete sampling operation, called *rasterization*, which prevents the gradient to be flowed into the mesh vertices. Since the forward rendering function is highly non-linear and complex, to achieve differentiable rendering, the paper suggested a new way to tackle the problem by proposing a *truly differentiable* rendering framework that is able to render a colorized mesh in the forward pass (Fig. 1).

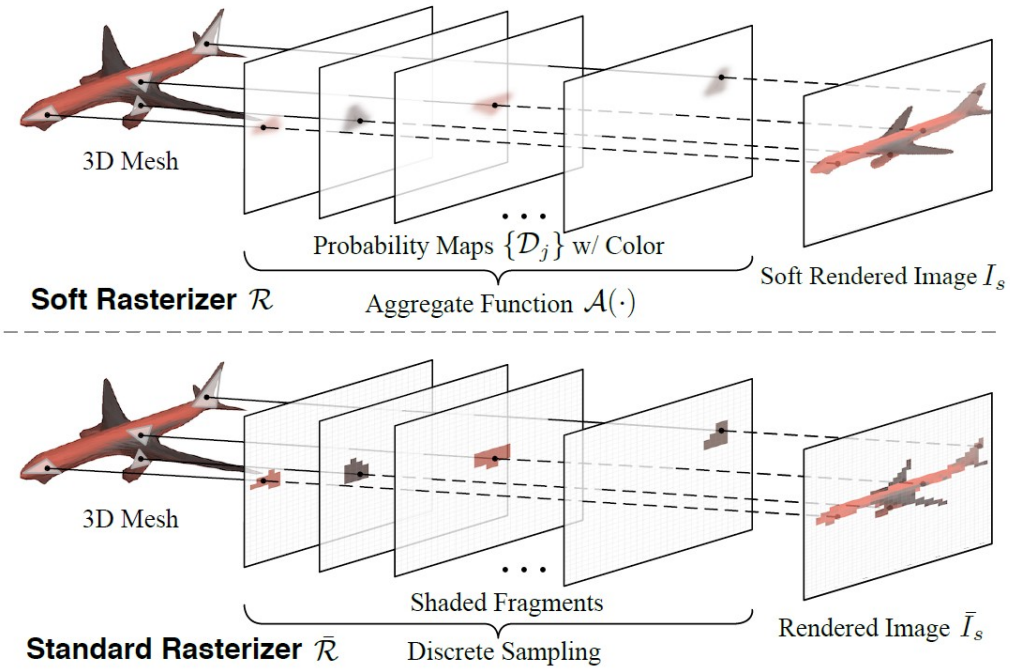

Figure 1: Soft Rasterizer $\mathcal{R}$ (upper), a truly differentiable renderer, which formulates rendering as a differentiable aggregating process $\mathcal{A}(\cdot)$ that fuses per-triangle contributions $\mathcal{D}_i$ in a "soft" probabilistic manner. SoftRas approach attacks the core problem of differentiating the standard rasterizer, which cannot flow gradients from pixels to geometry due to the discrete sampling operation (below). As published at [13].

In addition, SoftRas framework can consider a variety of 3D properties, including mesh geometry, vertex attributes (color, normal etc.), camera

parameters and illuminations and is able to flow efficient gradients from pixels to mesh vertices and their attributes. The key to the approach is the novel formulation that views rendering as a "soft" probabilistic process. Unlike the standard rasterizer, which only selects the color of the closest triangle in the viewing direction (Fig. 1 below), the article propose that all triangles have probabilistic contributions to each rendered pixel, which can be modeled as probability maps on the screen space. While conventional rendering pipelines merge shaded fragments in a one-hot manner, SoftRas propose a differentiable aggregation function that fuses the per-triangle color maps based on the probability maps and the triangles' relative depths to obtain the final rendering result (Fig. 1 upper). The novel aggregating mechanism enables SoftRas renderer to flow gradients to all mesh triangles, including the occluded ones. The framework been called Soft Rasterizer (SoftRas) as it "softens" the discrete rasterization to enable differentiability.

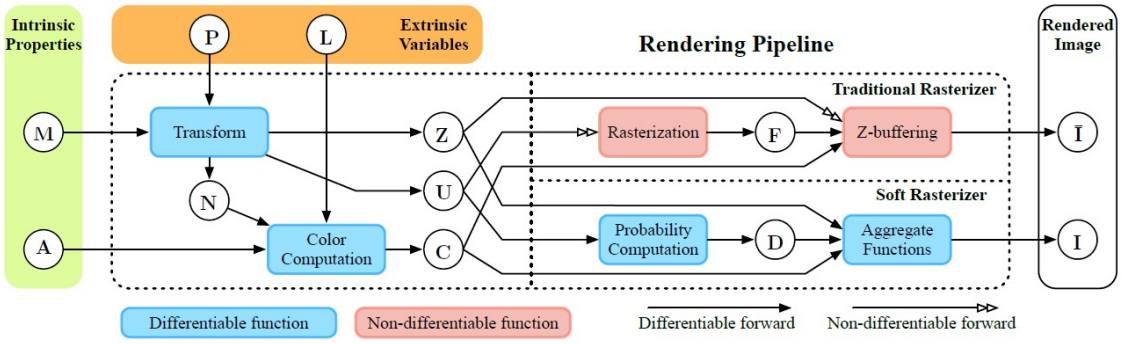

Figure 2: Comparisons between the standard rendering pipeline (upper branch) and SoftRas rendering framework (lower branch), as published at [13].

4.1.1 Differentiable Rendering Pipeline

As shown in Fig. 2, SoftRas consider both extrinsic variables (camera **P** and lighting conditions **L**) that define the environmental settings, and intrinsic properties (triangle meshes **M** and per-vertex appearance **A**, including color, material *etc.*) that describe the model-specific properties. Following the standard rendering pipeline, one can obtain the mesh normal

N, image-space coordinate **U** and view-dependent depths **Z** by transforming input geometry **M** based on camera **P**. With specific assumptions of illumination and material models (e.g., Phong model), color **C** can be computed given {**A, N, L**}. These two modules are *naturally differentiable*. However, the subsequent operations including the *rasterization* and *z-buffering* in the standard graphics pipeline (Fig. 2 red blocks) are not differentiable with respect to **U** and **Z** due to the discrete sampling operations. As mentioned above, in our project we use silhouettes only in our solution hence no colors are involved.

Differentiable formulation - SoftRas takes different perspective that the rasterization can be viewed as *binary masking* that is determined by the relative positions between the pixels and triangles, while z-buffering merges the rasterization results **F** in a pixel-wise *one-hot* manner based on the relative depths of triangles. The problem is then formulated as modeling the discrete binary masks and the one-hot merging operation in a soft and differentiable manner. To achieve this, SoftRas propose two major components, namely probability maps $\mathcal{D}_j$ that model the probability of each pixel staying inside a specific triangle f_j and aggregate function $\mathcal{A}(\cdot)$ that fuses per-triangle color maps based on $\mathcal{D}_j$ and the relative depths among triangles.

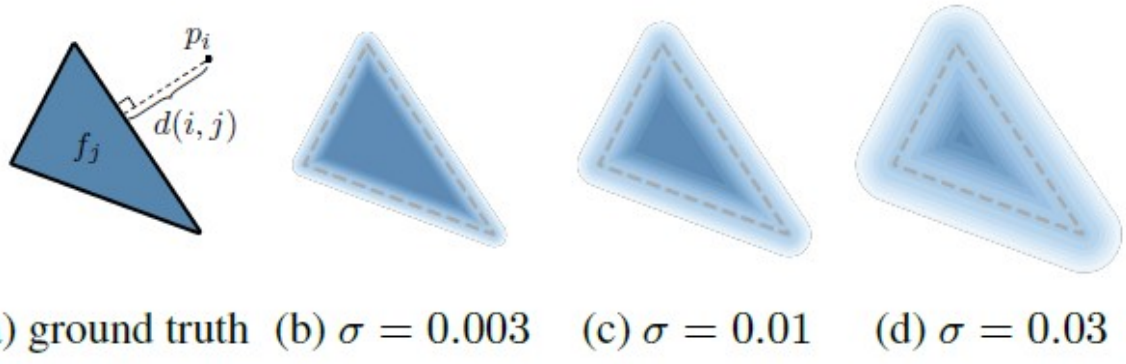

Figure 3: Probability maps of a triangle under Euclidean metric. (a) definition of pixel-to-triangle distance; (b)-(d) probability maps generated with different σ, as published at [13].

4.1.1.1 Probability Map Computation

SoftRas model the influence of triangle f_j on image plane by probability map $\mathcal{D}_j$. To estimate the probability of $\mathcal{D}_j$ at pixel p_i, the function is required to take into account both the relative position and the distance between p_i and $\mathcal{D}_j$ and defines as follows:

$$\mathcal{D}_j^i = sigmoid(\delta_j^i \cdot \frac{d^2(i,j)}{\sigma})$$

where σ is a positive scalar that controls the sharpness of the probability distribution while δ_j^i is a sign indicator $\delta_j^i = \{+1, if\ p_i \in f_i; -1, otherwise\}$. The article set σ as 1×10^{-4}. The value $d(i,j)$ is the closest distance from p_i to f_j's edges. By using the *sigmoid* function, the equation above normalizes the output to (0, 1), which is a faithful continuous approximation of binary mask with boundary landed on 0.5. In addition, the sign indicator maps pixels inside and outside f_j to the range of (0.5, 1) and (0, 0.5) respectively. Fig. 3 shows D_j of a triangle with varying σ using Euclidean distance. Smaller σ leads to sharper probability distribution while larger σ tends to blur the outcome. This design allows controllable influence for triangles on image plane. As $\sigma \to 0$, the resulting probability map converges to the exact shape of the triangle, enabling the probability map computation to be a generalized form of traditional rasterization.

4.1.1.2 Aggregate Function

For each mesh triangle f_j , SoftRas defines its color map C_j at pixel p_i on the image plane by interpolating vertex color using barycentric coordinates. Then clip and normalize the barycentric coordinates to [0; 1] for p_i outside of f_j . Then using an aggregate function $\mathcal{A}\cdot$ to merge color maps $\{C_j\}$ to obtain rendering output I based on $\{\mathcal{D}_j\}$ and the relative depths $\{z_j\}$. Inspired by the softmax operator, SoftRas defines an aggregate function $\mathcal{A}_S$ as follows:

$$I^i = \mathcal{A}_S(\{\mathcal{C}_j\} = \sum_j w_j^i C_j^i + w_b^i C_b$$

where $\mathcal{C}_b$ is the background color; the weights $\{w_j\}$ satisfy $\sum jw_j^i + w_b^i = 1$ and are defined as:

$$w_j^i = \frac{\mathcal{D}_j^i \cdot \exp(z_j^i/\gamma)}{\sum_k \mathcal{D}_k^i \cdot \exp(z_k^i/\gamma) + \exp(\epsilon/\gamma)}$$

where z_j^i denotes the normalized inverse depth of the 3D point on f_i whose 2D projection is p_i; ϵ is small constant that enables the background color while γ controls the sharpness of the aggregate function.

5 Testing and extending SoftRes

In our work we used the degraded form of the aggregated function for working with shading images (silhouettes) when the intrinsic vertices' color are set to one. The proposed dedicated aggregation function in the article $\mathcal{A}_O$ for the silhouette based on the binary occupancy is:

$$I_s^i = \mathcal{A}_O(\{\mathcal{D}_j\}) = 1 - \prod_j (1 - \mathcal{D}_j^i)$$

Intuitively, this equation models silhouette as the probability of having at least one triangle cover the pixel p_i.

5.1 Data

For the development of algorithms, as well as for evaluation, a vast amount of images and videos are needed. In this work we used only synthetic static data that been produced in animation software - Blender (as shown in fig. 4). In this software one can control the amount of leaves, the number of images, the parameters of the cameras, and of course the tagging is automatic from the software. A future step which has not been achieved in this work, is using a dynamic simulation that describes the development of plants. In the virtual world today, advanced models and animation tools are added daily [19], including generators for the synthetic production of plants and their growth.

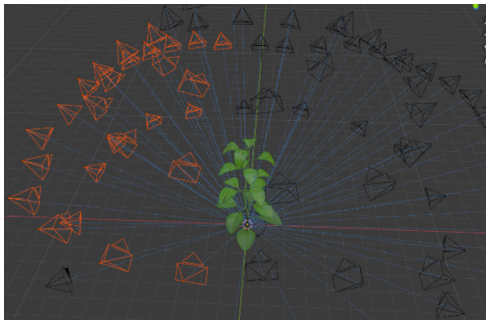

Figure 4: Synthetic simulation of a plant model in Blender

5.2 SoftRes on a single leaf

SoftRas as published, deals with a single object in the scene with the ability to render shading and colors. We hence tested its performance on a single leaf using silhouette images only.

Experiment 1 - We applied the published SoftRas algorithm on our dataset. For doing so we created an artificial model of a leaf as shown in Fig. 5 in Blender software and wrote a script to extract images with specific intrinsic and extrinsic parameters which we could inject into the SoftRas algorithm and to reload back to the Blender software for ground truth comparison. We produced 120 images of the leaf, evenly distributed around the object and down-sampled it to 64x64 pixels, due to the input requirements of the original SoftRas algorithm. Next we used a template mesh of a sphere with 642 verices as shown in Fig. 6 on the left image. After 600 iterations of the SoftRas algorithm we achieved a leaf shape model that was almost exact as the original leaf as can be seen on the right image.

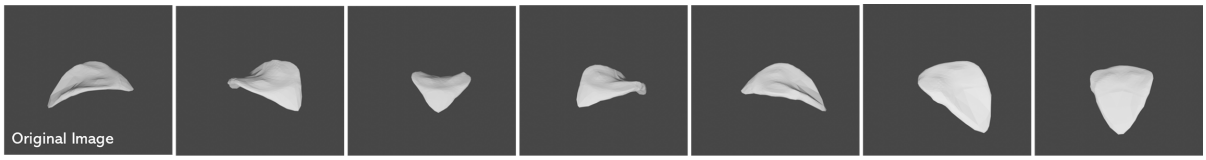

Figure 5: The original 3D model object in different views

Figure 6: SoftRas algorithm 3D model convergent. The projection of the calculated model for the view of a left image in Figure 1, as a function of the number of iterations. In this run 120 original images were incorporated. It can be seen that after 600 iterations the leaf model was restored in full with a very high level of accuracy.

5.3 Multiple Objects (centroids and activation function)

The algorithms that we encountered within the research handle a single object in the scene, wherein the scene holds one centroid. We redesigned the SoftRas algorithm to handle several objects, where each object will maintain its topological properties and centroid, while maintaining the independency of each object as a distinct mesh in the scene, without collapsing or exploding the activation function and the regression process. As seen below, we encountered few challenges during the research, where stable algorithm yield undesired results. We will show some of the problems that rose and the way we decided to tackle it.

Experiment 2 - In Fig. 7 we ran the original SoftRas algorithm on a scene that contains a sphere and a cube, one on top of the other. We initialized the template mesh model as a single sphere. Since there is only one centroid to the scene, and a single connected template mesh, the original mesh been stretched towards those two objects, maintaining a narrow "neck" to minimize the overall loss with the given images.

Figure 7: Run of the SoftRas algorithm, initialing the template model to a big sphere in the center of the scene, and it converges into a combined mesh of a cube and a sphere. That been done in 280 iterations.

At Fig. 8 we used the same input, but rather than one sphere, two spheres are used for initialization. The meshes blended in each other not converging into a global minimum. Hence, a better solution is required,

as proposed below, which improve the initialization and prevent shape intersections.

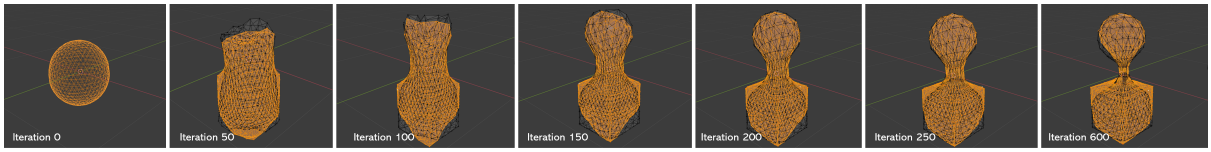

Figure 8: Same as in Fig 7, but inhere we initiate the template mesh with two spheres in the center of the scene, which converge into a loomed two meshes. We highlighted in orange one of the two meshes.

Experiment 3 - At Fig. 9 we tried the same experiment but now the initiation of the template spheres is as shown, far from each other, and the convergence is correct and quick. This was a great improvement towards our goal. The solution requires to determine how and where to initialize the centroids and how to choose the number of objects that are in the scene. More on that, when objects are close to each other,it is required to prevent the meshes to intersect (as shown in the Fig. 8).

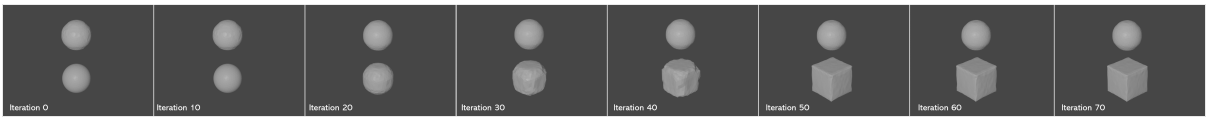

Figure 9: In this example we showed that in some initiation, the algorithm converge into the absolute minimum, where in this case is the cube and the sphere.

In terms of the number of objects in the scene we used the average of the amount of silhouettes' segmentation in the input images. For the initialization of the centroids we considered few possibilities, finally we used a few randomize initialization with a full run of the algorithm, and rerun it again with the converged centroids. In case those runs yield different results (meaning the centroids located in different positions at the end of the run) then we try all the centroids' setups. The output with the lowest Loss is the one to be chosen.

For some inputs, the algorithm so far produced great results, although in some cases, mostly with complex and occluded shapes, the algorithm's results were far off the ground truth. Whether after over-fitting the template shape started to folded around itself in attempt to reduce the loss,

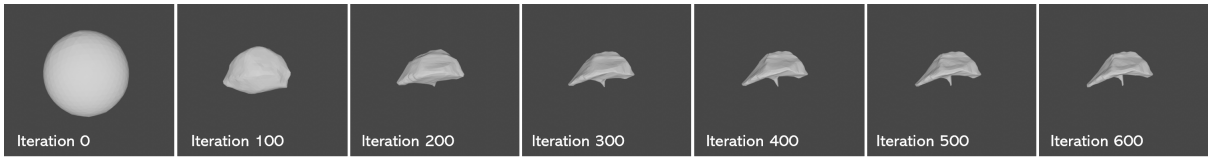

Figure 10: The projection of the calculated model as shown in Figure 5, but this time with a small subset of the source images taken from a top view. It can be seen that the bottom of the leaf, which is missing in the original photos, gets distorted due to lack of data, and some roughness at the top of the leaf.

or two object would ended up intersecting at a local minima even with the fact real objects are water-tight can't intersect themselves. To overcome those defects we altered the loss function as described next.

5.4 Loss function modification

The loss function in [13] is:

$$\mathcal{L} = \mathcal{L}_s + \lambda \mathcal{L}_c + \mu \mathcal{L}_g$$

Where $\hat{I}_s$ and I_s denote the predicted and the observed silhouette respectively. The silhouette loss is intersection over union (IoU) and defined as $\mathcal{L}_s = 1 - \frac{||\hat{I}_s \otimes I_s||_1}{||\hat{I}_s \oplus I_s - \hat{I}_s \otimes I_s||_1}$, where $\bigotimes$ and $\bigoplus$ are the element-wise product and sum operators respectively. The geometric loss $\mathcal{L}_g$ which contains Laplacian regularizes for both shape and color.

The obtained qualitative results using SoftRas on our input images are good, as can be seen in Fig. 6. However, when we are dealing with more than one object that induce occlusions on some of the objects in some of the views, the invisible parts been reconstructed with some artifacts and imperfections (see Fig. 10). In this example the projection of the calculated model as shown in Fig. 6, but this time been calculated with small subset of the source images taken from a top view. It can be seen that the bottom of the leaf, which is missing in the original subset images, becomes distorted due to the lack of data.

Hence, we propose to modify the loss function and to add a volume and a manifold intersection regulators as described next. Since we are dealing with tangible plants, we have a lot of prior knowledge about it,

as the leaves shape, size, thickness, stems behavior and many more. Let's focus on leaves - in our experiments, we could colidly assume the area of the leaf, as long as there were no occlusions. The main problem occurred when we do not have information on the back of the object, artifacts could show up, as seen in Fig. 10. We propose to use the assumption that the leaves are thin, almost completely flat, to solve the shape ambiguity due to missing views. In particular, we modify the energy function to force minimization, we could force the mesh to converge into a flat manifold. One problem that arose with this solution was a self-intersection of object mesh intersection - as shown in Fig. 11 (top two examples). To avoid that, we introduced our intersection loss.

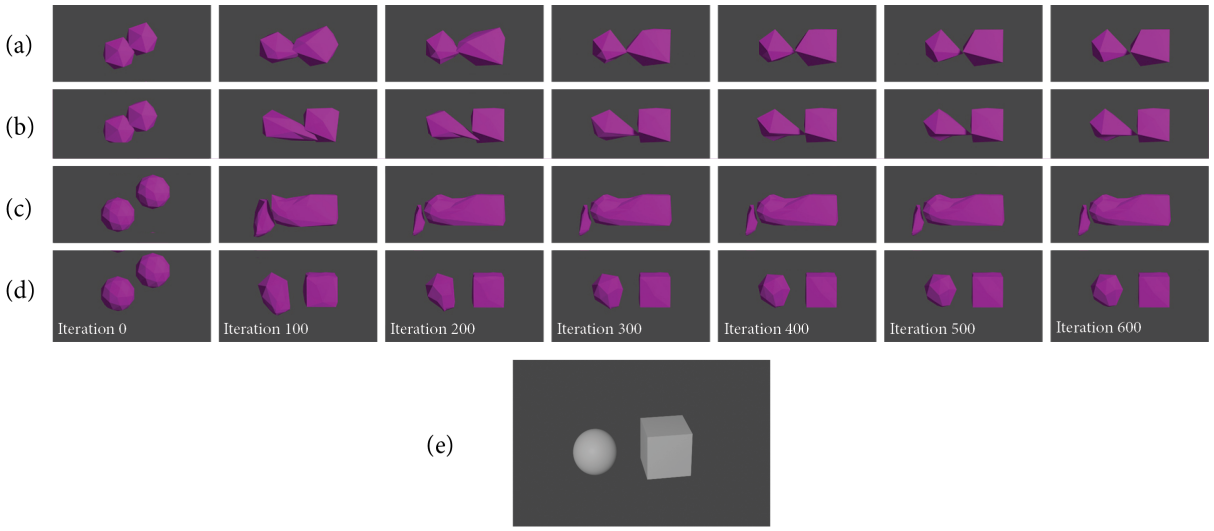

Figure 11: Each row, (a)-(d), is a result of various version of the algorithm in the same set of images. Each column describes to evolution of the mesh within x iterations. Bottom two runs have the same mesh templates - two simple sphere templates with 42 verteces each. The difference between those runs affected by the volume regulator. In example (c) we don't use the volume regulator hence in this setup the right mesh takes over the scene, while in example (d) the volume regulator prevent that to occur, and we accept the desire sphere and cube.
On the top two examples, we can see a simple 12 vertices' spheres, second from top (b) is a run without any intersections constrains, and the two spheres intersect each other, while at the top example, using our intersection loss, the spheres end up without any intersections.
Image at row (e) is a figure of the target objects - a simple sphere and a simple cube, one next to each other.

5.4.1 Volume Loss

We based on [24] work of efficiently calculating area and volume of a 2D and a 3D meshes respectively. By breaking it into elementary shapes of triangular and tetrahedrons shapes. Doing so allowed us to compute the area/volume of the single elementary shape individually, and then to add up all the values for the complete mesh. This algorithm is simple and can be calculated in parallel, we could therefore code it to run on GPU makes it super efficient.

In a nutshell, we will provide the 2D calculation, that can be extended into the 3D space, as suggested in [24]. The assumption for this algorithm is the polygon is close - and since we are dealing with real life objects we assume this is the case for all the calculations. Since we know all the vertices location, we can calculated the edges segments of the mesh. Once we got it, we will extract the *normals* for each edge easily. For example, edge AB in Fig 12 has the normal:

$$N_{AB} = \frac{-(y_2 - y_1)\hat{x} + (x_2 - x_1)\hat{y}}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$

Where (x_1, y_1) and (x_2, y_2) are the coordinates of vertices A and B respectively, and $\hat{x}$ and $\hat{y}$ are the unit vectors for axes. The normals are then used to construct a set of triangles that connects all the polygons verices with the origin. Each edge and the origin form an elementary triangle, which is the smallest unit for computation. We define the *signed area* for each of those triangles as below: The magnitude of this value is the area of the triangle, while the sign of the value is determined by checking the position of the origin with respect to the edge and the direction of the normal. Take the triangle OAB in Fig. 12 as an example. The area of OAB is:

$$|S_{OAB}| = |\frac{1}{2}(-x_2 y_1 + x_1 y_2)|$$

The sign of S_{OAB} is the same as the sign of the inner product $\overrightarrow{OA} \cdot \vec{N_{AB}}$, which is positive in this case. The total area of the polygon can be com-

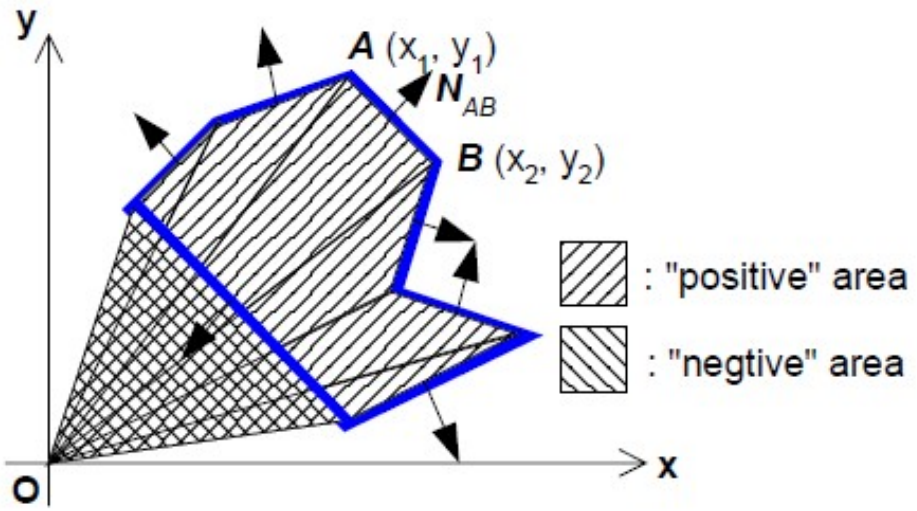

Figure 12: The calculation of a 2D polygon area as published at [24]

puted by summing up all the *signed areas*. That is,

$$S_{total} = \sum_i S_i$$

where i goes through all the edges or elementary triangles. We can skip the sign check by looping through all the edges, so that the inside part of the mesh is always kept at the same side. The final result should be the magnitude of S_{total} The 3D case is an extension of the algorithm above into the 3D space, where:

$$V_i' = \frac{1}{6}(-x_{i_3}y_{i_2}z_{i_1} + x_{i_2}y_{i_3}z_{i_1} + x_{i_3}y_{i_1}z_{i_2} - x_{i_1}y_{i_3}z_{i_2} - x_{i_2}y_{i_1}z_{i_3} + x_{i_1}y_{i_2}z_{i_3}$$

$V'_{total} = \sum V_i'$, where V_i stands for the index of the elementary tetrahedron. We implemented this algorithm with Pytorch to run on a Cuda-GPU framework. The code for this section attached at the work git-hub site.

5.4.2 Manifold Intersection Loss

Based of Tomas Möller's work [17] of a Fast Triangle-Triangle Intersection Test, to compute whether or not two triangles intersect, we could re-code the algorithm to run on a GPU paralleling it through the entire mesh. Since the algorithm calculates the results for every two triangles, all we

need is to match the upper right table of all the triangles sets. Each calculation is exclusive, and returns a binary value whether there is an intersection or not. A vertices summary table accumulates the number of intersections for each vertex. This table at the end being injected to the *Loss Function* as a punishing regulator.

Briefly, let us denote the two triangles T_1 and T_2, and V_0^1, V_1^1, V_2^1 and V_0^2, V_1^2, V_2^2 respectively; and the planes in which the triangles lie π_1 and π_2. First we build the planes' equation for both triangles and check the signed distances for the vetices of T_1 to π_2. Now, if all $d_{v_i^1} \neq 0$, $i = 0, 1, 2$ and all have the same sign, then T_1 lies on one side of π_2 and the overlap rejected. The same is done for T_2 and π_1. Indeed, for a pair to pass this test there must be some line of direction $N_1 \times N_2$ that meets both.

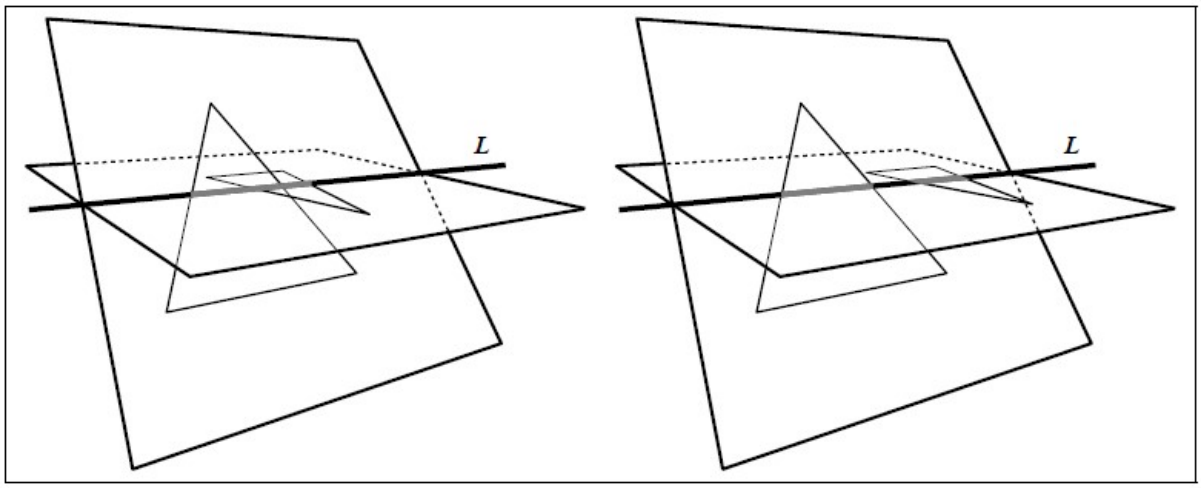

Figure 13: Triangles and the planes in which they lie. Intersection intervals are marked gray in both figures. **Left**: the intervals along L overlap as well as the triangles. **Right**: no intersection, the intervals do not overlap, as published at [17].

If all $d_{v_i^1} \neq 0$, $i = 0, 1, 2$, then the triangles are co-planar and this case is handled separately. we decided to skip this test since it is rare and not a stable situation, and it won't change much the convergence of the general solution. If not, the intersection of π_1 and π_2 is a line, $L = O + tD$, where $D = N_1 \times N_2$ is the direction of the line and O is some point on it. Note that due to our previous calculations and rejections, both triangles are guaranteed to intersect L. These intersections form intervals on L, and if

these intervals overlap, the triangles overlap as well. Two situations that can occur are depicted in Fig 13. By calculating the scalar interval (on L), as described in [17], we can reject or approve an intersection between those two triangles. The Cuda-GPU code for this section attached at the appendix.

5.4.3 Final Loss Function

Based of [24] we can efficiently compute the accumulated volume of the scene with several disconnected manifolds at once. Using prior knowledge of the object volume (such that a standard leaf is flat) we can achieve great improvement. Based on [17], we compute efficiently all the intersections within a given mesh to avoid self collapse of the mesh during the iterations as described at chapter 3.3. We decide not to implement the color channels data at that point and our final loss function will be:

$$\mathcal{L} = \mathcal{L}_s + \lambda_g \mathcal{L}_g + \lambda_v \mathcal{L}_v + \lambda_i \mathcal{L}_i$$

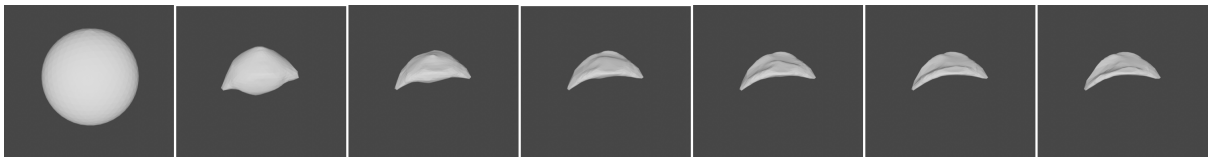

Figure 14: The projection of the calculated model as shown in Figure 3 with the same set of images, but now in combination with smoothness and volume regulators. It can be seen that the leaf surface is smooth and closer to the reconstruction of the original model (Fig. 2) and that the formed deformation at the bottom of the leaf in previous reconstruction (Fig. 3) disappeared.

5.5 Experiments

The results of the new loss function can be seen in Fig. 14. To see the effect of our loss function, compare this result with Fig. 10.

6 Accuracy, Efficiency, and Pyramid Regression

The accuracy of the results is affected by the internal and external cameras parameters. We assume that the cameras are calibrated and no distortion

is present.

The efficiency of the algorithm is affected by the input setup, where the number of observed images introduced (more images bring into more accurate reconstruction) and the higher resolution of those images are in exponent proportion to the running time. The third component to the efficiency equation is the template mesh resolution, which is in exponent proportion as well to the running time.

Therefore we altered our algorithm in a way to boost it performance according to the factors above.

6.1 Pyramid solution

The algorithm is an energy minimization function. Therefore, if we "feed" the algorithm with a stable initialization parameters, it tends to stay in that position. Harnessing this feature, we decided to stick with the original algorithm resolution parameter for the first round, of 64x64 pixels of the input images, and to let the algorithm to converge. To enhance the output resolution, we restarted a new run, doubling the resolution in both axis (x4), while the initialization is the output of the first run. Doing so, the model stayed stable, and most of the time converged after less then 10 cycles. We used this method to grow the resolution back to a full scale image of 1980x1980 pixels.

6.2 Mesh template resolution

As mentioned above, additional factor to the run time based on the template meshes that address the algorithm, and the bigger they are, and the higher number we use, weigh heavily on the algorithm's run time. For efficiency, we showed in our work, similar to the resolution pyramid above, that using simpler template meshes in the first stages of the run, and then increasing their resolution and rerun the algorithm, bring the same convergence as using more complex one in a significantly shorter time (for example, each triangle in the mesh can be split into three triangles, but

introducing an additional vertex in the middle).

7 The Project Contribution

For conclusion we summarized our work with our accomplishment of our research and direction for future work.

1. The SoftRas algorithm focuses on the reconstruction of a single object (Fig. 6, the original object is shown as Fig. 5). We demonstrated that SoftRas can be extended to two deal with two or more objects.
2. We explores several possibilities to initialize several the meshes of more than a single object.
3. We extend the loss function of SoftRas to deal with mesh intersections which allows to deal with more than a single objects.
4. We extend the loss function of SoftRas to calculate the volume of the mesh, which allows to control the desire volume regulation.
5. Efficiency: We showed several way of improving computation by combining different resolutions of the input images and template mesh complexities to converge the algorithm without compromising product quality.
6. We generated a synthetic data set to test the algorithm.
7. We implemented items 2-5 on a GPU.

7.1 Future work

Listed here are more questions needed to be answered to fulfill the phenotyping task completely.

1. Tracking the transformation of a plant. This process will allow the addition of the time dimension to the plant modeling. Also, tracking changes over time can improve segmentation and three-dimensional calculation.

2. Adding the color channels to improve the geometric reconstruction.

3. Each plant and organ in the plant has known properties which in classical extraction from images are not reflected, however in the world of neural networks it is possible to refer to prior knowledge such as constraint on volume and shape. In preliminary work (as noted in the previously) we showed how the use of volume constraint improves the convergence of the model to a flat leaf. The goal is to combine a more complex feature modeling adapted to different parts of the plant, and to different plants.

4. Segmentation of plant parts: In recent years there has been a significant improvement in the segmentation of plant parts, using neural networks where most of them are based on single images.

5. In this project we provide basic proof of concepts with only a qualitative evaluation. A more comprehensive treatment requires thorough experimentation, should be performed.

8 Project Summary

The aim of the study is to create a topological phenotype of plants and reconstruct its change over time, a topic that is considered particularly challenging in the field of plant science. This project adapted and activated advanced methods of computer vision to bring closer a solution to this important and challenging problem. We aimed to develop an algorithm based on SoftRas that allows reconstruction of an entire plant as several objects in parallel, and directly handles occlusions. Taking advantage of parallel computing graphics cards for a traditional algorithm, with a combination of AI and machine learning in the future could bring a major change to the vision world, via Agri-tech and phenotyping in particular. As described above, we provide basic proof of concepts with only a qualitative evaluation. A more comprehensive treatment requires thorough experimentation, and it is left for future work.

References

[1] Lemnatec gmbh. *Nerscheider Weg 170, 52076 Aachen, Germany*, (Tel. +49 2408 981850).

[2] Psi drásov. *Czech Republic*, 2021.

[3] Geng Bai, Yufeng Ge, David Scoby, Bryan Leavitt, Vincent Stoerger, Norbert Kirchgessner, Suat Irmak, George Graef, James Schnable, and Tala Awada. Nu-spidercam: A large-scale, cable-driven, integrated sensing and robotic system for advanced phenotyping, remote sensing, and agronomic research. *Computers and Electronics in Agriculture*, 160:71–81, 2019.

[4] Luca Carlone, Jing Dong, Stefano Fenu, G Rains, and Frank Dellaert. Towards 4d crop analysis in precision agriculture: Estimating plant height and crown radius over time via expectation-maximization. In *ICRA Workshop on Robotics in Agriculture*, pages 1–8, 2015.

[5] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019.

[6] Valentin Deschaintre, Miika Aittala, Fredo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (ToG)*, 37(4):1–15, 2018.

[7] Fabio Fiorani and Ulrich Schurr. Future scenarios for plant phenotyping. *Annual Review of Plant Biology*, 64(1):267–291, 2013. PMID: 23451789.

[8] Yufeng Ge, Abbas Atefi, Huichun Zhang, Chenyong Miao, Raghuprakash Kastoori Ramamurthy, Brandi Sigmon, Jinliang Yang, and James C Schnable. High-throughput analysis of leaf physiological and chemical traits with vis–nir–swir spectroscopy: a case study with a maize diversity panel. *Plant methods*, 15(1):1–12, 2019.

[9] Yu Jiang and Changying Li. Convolutional neural networks for image-based high-throughput plant phenotyping: a review. *Plant Phenomics*, 2020, 2020.

[10] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018.

[11] Keerthy Kusumam, Tomáš Krajník, Simon Pearson, Tom Duckett, and Grzegorz Cielniak. 3d-vision based detection, localization, and sizing of broccoli heads in the field. *Journal of Field Robotics*, 34(8):1505–1518, 2017.

[12] Chris Lehnert, Dorian Tsai, Anders Eriksson, and Chris McCool. 3d move to see: Multi-perspective visual servoing towards the next best view within unstructured and occluded environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3890–3897. IEEE, 2019.

[13] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7708–7717, 2019.

[14] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019.

[15] Federico Magistri, Nived Chebrolu, Jens Behley, and Cyrill Stachniss. Towards in-field phenotyping exploiting differentiable rendering with self-consistency loss.

[16] Merriam-Webster. Pyenotyping. 2008.

[17] Tomas Möller. A fast triangle-triangle intersection test. *Journal of graphics tools*, 2(2):25–30, 1997.

[18] Gustavo Scalabrini Sampaio, Leandro Augusto da Silva, and Maurício Marengoni. 3d reconstruction of non-rigid plants and sensor data fusion for agriculture phenotyping. *Sensors*, 21(12):4115, 2021.

[19] C. Stachniss. 4d-plant-resgistration, 2021.

[20] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings*

of the IEEE conference on computer vision and pattern recognition, pages 8934–8943, 2018.

[21] Breght Vandenberghe, Stephen Depuydt, and Arnout Van Messem. How to make sense of 3d representations for plant phenotyping: a compendium of processing and analysis techniques. 2018.

[22] Manuel Vázquez-Arellano, Hans W Griepentrog, David Reiser, and Dimitris S Paraforos. 3-d imaging systems for agricultural applications—a review. *Sensors*, 16(5):618, 2016.

[23] Laura Zabawa, Anna Kicherer, Lasse Klingbeil, Reinhard Töpfer, Heiner Kuhlmann, and Ribana Roscher. Counting of grapevine berries in images via semantic segmentation using convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 164:73–83, 2020.

[24] Cha Zhang and Tsuhan Chen. Efficient feature extraction for 2d/3d objects in mesh representation. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 3, pages 935–938. IEEE, 2001.

[25] Chunjiang Zhao, Ying Zhang, Jianjun Du, Xinyu Guo, Weiliang Wen, Shenghao Gu, Jinglu Wang, and Jiangchuan Fan. Crop phenomics: current status and perspectives. *Frontiers in Plant Science*, 10:714, 2019.