School of Computer Science
Faculty of Science and Engineering
University Of Nottingham
Malaysia

UG FINAL YEAR DISSERTATION REPORT

Learning 3D Representation of Germinated Parts for View-Angle-Robust Classification

Student's name : Gabriel Hoh Chao Jie

Student Number : 20311860

Supervisor Name : Dr. Iman Yi Liao

Year : 2023

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE IN COMPUTER SCIENCE (HONS)
THE UNIVERSITY OF NOTTINGHAM

Learning 3D Representation of Germinated Parts for View-Angle-Robust Classification

Submitted in May 2023, in partial fulfillment of the conditions of the award of the degrees B.Sc.

- **Gabriel Hoh** -
School of Computer Science
Faculty of Science and Engineering
University of Nottingham
Malaysia

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date 1 / 5 / 2023

Contents

Introduction 1

Motivation 3

Related works 4

3D Representation 4

Single-view 3D Reconstruction 8

GAN Inversion 10

Methodology 13

Image and semantic segmentation 13

Pose estimation using Textured3DGAN 14

Generation of pseudo-ground truth using ConvMesh 15

MeshInversion pre-training and GAN inversion 17

Experiments and results 20

Dataset 20

Image and semantic segmentation 21

Pose estimation 24

Generating pseudo-ground truth data 25

Training of generator and performing GAN inversion 26

Qualitative results 28

Sample outputs 28

Analysis and discussions 30

Analysis 30

Discussions 30

Conclusion and future works 32

References 33

Introduction

Southeast Asia has been a major source of palm oil production in the world, with countries like Malaysia and Indonesia producing 85% of the world's supply of palm oil [1]. The uses of palm oil ranges from being used to make edible foodstuff such as doughnuts and chocolates, to everyday necessity such as toothpaste, shampoo and more. With it being versatile and widely used, it has a huge demand that needs to be fulfilled, and coupling with rising demands, presents the need to expand the production of palm oil. However, the manual production of oil palm seeds is by itself very laborious. The usage of good oil palm seeds is preferable to bad oil palm seeds as much as possible to produce the highest yield with minimal cost of operations. With this, choosing good oil palm seeds while discarding the bad seeds is critical in ensuring quality production. The convention method for determining inspecting seeds is manual labour inspection. This presents two main problems, the first being it requires personnel that is highly trained, as such it consumes a huge amount of resources and manpower to train individuals capable of performing such task, the second problem is even a highly trained worker is still subjected to judgement biases and human error, decreasing the overall accuracy of the classification of good oil palm seeds.

The rapid development of technology has made acquiring and processing of data quite easy and coupling with recent emerging breakthroughs and development of algorithms employed in machine learning, it has made machine learning a rather feasible approach in solving problems in the modern day. One of such machine learning algorithm, called Convolutional Neural Networks (CNNs), as already shown results in being able to perform better than traditional methods in classifying Crambe seed quality based on X-ray images [2]and differentiating corn species[3]. As such CNNs has been chosen as one of the methods in classification of the quality of germinated oil pam seeds, and while data was collected under controlled environment, it was able to achieve a testing classification accuracy of 97.78%[4], making it a promising method of being employed to be used industrially. Therefore, the attention was thus turned towards building a robust, generalisable, and edge-device-compatible model for a quality classification of germinated oil palm seeds.

This, of course, opened to a lot of problems that would require extensive research to dealt with. One of such would be the view-angle dependent ambiguity of 2D images.

During the process of classifying the quality of a germinated oil palm seed, one of the main factors that determines the quality of a oil palm seed depends on the shape and the angle between the plumule and the radicle of the germinated oil palm seed. If the plumule grows normally but the radicle is stunned, it is classified as a bad seed, and vice versa. However, if a good seed is positioned in such an angle that either its plumule or radicle points directly towards the camera, the 2D image appearance of the germinated part would look similar to that of a bad quality seed where either plumule or radicle is stunned. Such ambiguity is a result of the representation of the seed in 2D images. Thus, it would be feasible to represent the germinated part of the seed as a 3D image instead of a 2D one.

In the 10 years or so, the field of computer vision has had significant development in the proposal of novel and robust frameworks/architecture to perform learning of information from objects and the natural environment. One of such appearance are

Generative Adversarial Networks, also known as GANs. The framework of GAN was proposed in 2014 by Goodfellow et. al[4]. GANs are composed of two main networks, a Generator, and a Discriminator. The idea is that, Generator "generates" images, and the discriminator "discriminates" images, whether they came from the dataset or not. Thus, the images generated from the Generator are considered "fake", while images from the dataset are considered "real". The goal of the Generator is to fool the Discriminator into thinking that its images are the real deal, while the goal of the discriminator is to discern whether the images are part of the dataset or not. This becomes a minmax game between the Generator and the Discriminator, and as such they are adversaries (thus adversarial networks). The critical part of this is that the Generator does not have access to the "real" dataset, while only the discriminator has access to the "real" dataset. The Generator has to blindly generate images while sampling from a distribution of latent code.

One of the popular and well-known deep learning frameworks is called the Artificial Neural Network (or ANNs). Its goal is to predict based on the inputs a target output, and then the calculated loss is fed back into the neural network to tune the weights. As such, their main objective is to model and estimate the function of the data based on their inputs and outputs. In GANs however, it instead treats data as observations obtained from a wider basket of possible values and distribution. With the Generator unable to access the "real" dataset, it instead only has access to a loss signal obtained from the discriminator and is forced to learn a distribution, i.e. instead of learning a function of the dataset, compared to an ANN, it learns a distribution of the dataset.

Since its inception in 2014, GANs has seen major development in its usage in the field of computer vision. In 2015, DCGAN was developed by Radford et. al.[5] which employs the use of convolutional neural networks in the generation and discrimination of the images. In 2016, Wasserstein GAN was introduced by Arjovsky et. al.[6] that takes advantage of a different loss function, called Wasserstein distance to train and optimize GANs. A major breakthrough in GANs was made in 2019 when StyleGAN [7] was introduced, and besides being able to generate high-quality images, it has also has controllable abilities on features of generated image like shape, texture and color. Just 1 year later, in 2020, StyleGAN2[8] was introduced with addition by employing new strategy called progressive growing scheme, which allows itself to "grow" according to the amount of images available that is fed into it, adapting by changing its complexity to increase its representation power.

3D reconstruction from single-view images is a difficult task, owing to the fact that some surface are occlude, while lightning conditions and pose can affect the information that can be extracted from the image itself. With such difficult and complex task, each image can be considered as an observation of the possible shape distribution sampled randomly. Since the responsibility of GANs is to learn complex distribution, as such GANs can be seen as a good framework that to learn and reconstruct the germinated parts of the oil palm seed, and since only individual picture of seeds are available, this has become a single-view 3D reconstruction problem.

Motivation

One of the motivations that made me choose this topic as my final year project is that a provides me a chance to apply my machine learning knowledge in the field of industry. As a soon-to-be graduate and a future employee of society, this provides me with a first-hand experience to engage in industry problems through the means of computer, with artificial intelligence being an emerging technology that will soon be employed en masse in solving future problems.

Next, this research has the capability to assist other researcher in their field of study and their exploration of new ideas. The current study that I am researching on is actually a small part of a bigger research performed by a group of professionals in which my supervisor is part of, and with it are several challenges that needs to be addressed before any experimentation can be attempted, with this study trying to engage and solve one of the challenges. Other research that is related to image processing may also benefit from this study as it is about exploring the usage of Generative Adversarial Networks in image processing, which has already shown promising results in early exploration of others.

Besides that, this research also allows me to explore image processing as a potential future career. As my research is closely about 2D images and their transformation to 3D images, this provides with me insight about how image processing can be employed to solve the problems posed in the industry when the need to automate production is significantly increasing due to influx of demands from rising population count.

On top of that, the problem itself is an ill-posed one, meaning to say there are a lot of ways to approach it. While the idea of 3D image generation from 2D images is not a new one, however the idea itself has a lot of challenges that makes it seemingly just unfeasible to be done. However, recent development in machine learning algorithms has made not so impossible as initially thought of. As such, this research creates a challenging yet interesting opportunity for me to not just engage in such problem, but to also offer a new perspective to other problems that also looks seemingly impossible to solve.

Lastly, I am also interest in the mathematical reasoning to implement such idea. As with the main reason why I choose Computer Science as my choice of university course, I am interest on how mathematical reasoning can be applied to solve such seemingly unfeasible problem. I understand that the idea of mathematical thinking is not just important the jobs and work that we do, but is also in our everyday life, that is to be able to make common decision with proper logical thinking instead of by gut or by common census.

Related works

Before we can identify how GAN can be leveraged in single-view 3D reconstruction of 2D images of germinated palm oil seed, there are several choices of design that must be identified and their corresponding challenges.

3D Representation

To perform 3D reconstruction of 2D images, we must first identify the choice of 3D object representation. This is crucial because it affects our choice of architecture that can be used to perform 3D reconstruction.

It can be summarised as two of the following:

- Volumetric-based representation
- Surface-based representation

There can also be an intermediate layer between 2D image and the 3D images, which can be utilised to guide the reconstruction process, resulting in better convergence and stability of the model, as opposed as a direct 2D to 3D approach.

In volumetric-based representation of 3D objects, voxels are used to represent the 3D object itself by discretising the volume space of and around the object into grids of individual cubes (this is how the popular game *Minecraft* represent objects as blocks). The goal of 3D reconstruction is to recover a grid such that the resulting 3D shape is as closed as possible to the real object in the image.

To reconstruct the object as 3D voxels, a latent representation of the object in the form of a vector is obtained by encoding with a trained encoder, after that, the 3D shape of the object is regressed through a decoder, typically a deconvolutional neural network that decodes the latent code into a grid of voxels. A grid occupied with a cube means that part of the space belongs to the objects, while an unoccupied grid means that it instead belongs to the background. One of the earliest work of regressing the 3D volumetric grid of an object was by Wu et. al. in 2014 called 3D ShapeNets[9]. It utilises 2.5D information of the object in the form of depth maps that can be easily extracted using depth sensors. Later, in 2017, Wu et. al. introduced the idea of MarrNet[10], which aims to regress a normal map, a depth map and a silhouette map from the 2D image to reconstruct the volumetric 3D grid. To ensure that the reconstructed object is consistent with the 2D image, it uses a reprojection consistency loss consisting of a depth reprojection loss and a surface normal reprojection loss to ensure that the 3D objects explains well the 2.5D information.

While the aforementioned works treats voxels as a binary occupancy grid, there are works that instead directly regresses the 3D volumetric grid by predicting the voxel occupancy probability, where each grid block has a probability of belonging to the object, instead of the foreground, which were done by Tulsiani et. al. [11], [12] and Wu et. al. [13]

The mentioned voxel-based approaches have inherent problems that comes from the representation of 3D objects in volumetric voxel grid, that is they are computationally complex and has high memory requirements. This results in voxel grids with low resolution,

typically around size of 32^3 or size of 64^3, which does not perform well for objects with fine details and with thin structures.

To model a volumetric grid voxel in high resolution, there have been attempts to increase the output resolution while keeping computational and memory requirements low. One of the approaches uses octrees to represent 3D voxel grid of objects, where each internal node has exactly 8 children, partitioning the voxel space into big chunks if it does not include the 3D object, into smaller chunks if it does include the 3D object. The main motivation behind using octrees is the fact that, besides the surface of the 3D object, the voxel space is mostly empty. Wang et. al manages to utilise this principle and designed O-CNN [14], in which they manage to employ a novel octree data structure to store the volumetric voxel grid as an octree and CNN features, which can be efficiently stored in memory and then can be trained with a GPU. As such, this concentrates the storage of info on the 3D surface of the object, instead of having to store the empty voxel grids too.

While the above introduced methods can significantly reduce the memory requirements in modelling volumetric voxel grids of objects, they are however complex to implement, and the resulting voxel resolution is still relatively small (256^3). Instead, there have been researches that represents the volumetric grid as a neural network. A 3D voxel grid can be considered as a function where the input is the xyz coordinate and the output is whether it is occupied or not. The approaches then attempt to have a neural network learn and estimate the underlying function, and such networks are called occupancy networks. In [15], Chen and Zhang introduced a decoder that takes in a latent representation of the shape and a 3D point, and then return a value showing whether the point is inside or outside the shape. In a way, they managed to reconstruct a high-resolution volumetric grid. However, its weakness is that every point in the voxel grid needs to pass through the model in order to obtain the whole 3D object, making reconstruction during runtime long and inefficient. Tatarchenko et. al.[16] instead proposes an occupancy network which is a continuous decision boundary represented by a neural network classifier to represent the 3D surface of the object. This allows the model to be trained without requiring high memory usage and can be evaluated at any resolution.

Another approach is to use geometric parts to represent the individual part of a 3D shape, and then have the model learn how to stitch them together. Li et. al. [17] proposed GRASS (Generative Recursive Autoencoder for Shape Structure), which voxel representation is used in the parts of the shape instead. To generate the 3D object, it first generates the structure of the shape, and then generate the geometry shape of the parts. It uses a Recursive Neural Nets (RvNN) encoder-decoder architecture trained using a Generative Adversarial Network. Even though each part has a resolution of 32^3 , as each shape is generated separately, this allows reconstruction at a high resolution. Zou et. al. [18] instead proposes 3D-PRNN, which uses primitive shapes to build the model of the object. To generate the next set of primitives to use, it uses a latent representation vector which is encoded from the object, and the previously used primitives. This method produces the structure of the reconstruction as an abstract set of cuboids. With further refining, however, could lead to more detailed 3D reconstruction.

Another method for producing high resolution 3D volumetric representation is with a set of orthogonal basis. This becomes a matter of predicting the corresponding coefficients based on the latent variable, and then reconstruct the 3D volumetric voxel using the inferenced coefficients. Johnston et. al.[19] uses a Discrete Cosine Transform-II (DCT-II) to represent the orthogonal basis, and then a convolutional decoder to predict the low frequency DCT-II coefficients.

To increase the efficiency of producing high resolution volumetric grid, there are techniques that starts with modelling a coarse low-resolution voxel grid, and then adding more details to become a fine high-resolution voxel grid. Yang et. al.[20] managed to use an up-sampling module to up-sample a rough 3D voxel grid into 256^3, meanwhile Wang et. al. [21]approached from a slice-by-slice method, where each slice is a generated image and then stitched together to form a coarse voxel grid, and then each slice is upsampled into higher resolution. To ensure consistency and continuity of each slice with the next slice, five consecutive slices are encoded using a 3D encoder, and then passed as a vector into an LSTM (Long Short-Term Memory, a recurrent generator which can generate a feature vector), and the output is fed into a 2D decoder producing a fine-detail high resolution image, forming a high-resolution 3D volume.

Another approach in modelling high resolution voxel grid while keeping computation and memory requirements relatively low is by first producing the coarse voxel gird, and then refining regions of the voxel that requires further detail refinement. Dai et. al. [22]does this by copying voxels from a 3D database while computing a k-nearest neighbour. Han et. al.[23] modifies this approach by adding a local 3D CNN to perform patch-level surface refinement. On the other hand, Cao et. al. [24] first predicts a coarse volumetric grid of 128^3, and then take a block region of 16^3 from each region in the coarse volumetric grid to predict whether they need further up-sampling. Of course, these methods are time consuming as they need to search region-by-region to decide whether they require further refinement or not, and then search the corresponding matching voxel from a 3D database.

The goal of reconstructing a volumetric representation is, in the end, to produce a 3D mesh that represents the surface of the object. The Marching Cube algorithm was proposed Lorensen and Cline [25] as a post processing step to create the actual 3D surface mesh. However, as the algorithm works on an intermediate representation, the whole model cannot be trained end-to-end. Liao et. al. [26] aims to overcome this with a Deep Marching Cube model, by using a modified differentiable representation and separating the mesh topology from the objects geometry, allowing the model to be trained end-to-end.

It is clear that there is an overwhelming amount of research that aims to reconstruct and represent objects in a 2D image as volumetric voxel grids. They have also have high representation power as they discretise the space into individual space, and as such they can model objects of any geometric shape and topology, resulting in a complex representation. However, by principle of Occam's razor, volumetric voxel grid representation may not be suitable for the purpose of this project as germinated palm oil seeds has simple topology and geometric shape. We are also interested in capturing the texture of the germinated seeds and also utilising texture prior for 3D reconstruction of germinated palm oil seeds. As such, we turn to a different method of 3D representation.

Instead of modelling a 3D object as a volumetric space, we can instead model the surface of the 3D object, since information about the object is concentrated on the surface themselves. Attempting to model the surface of an object results in a mesh where it is made up of vertices connected together to form a triangle, as such they do not have regular structure and is hard to be modelled by deep learning networks. To address this problem, three main approach has been proposed to allow a deep network to learning the underlying surface of the 3D object, which are parameterisation-based representation, template deformation-based representation and point clouds.

In parameterisation-based representation, it becomes a problem of trying to perform regular parameterisation of the surface of the 3D object. [27] used geometry images and spherical parameterisation to parameterise the surface of the object, and are suitable only for objects with 0-genus topology and disk-like surfaces. By representing the geometry of the 3D surface using parameters, it can significantly reduce memory footprint and computation requirement for processing and reconstructing the 3D object. The main challenge of parameterisation-based representation in 3D reconstruction however is trying to obtain the accurate representation of the object. While a simple set of geometric primitives can be used, its difficult to obtain a quality parameterisation of the object, and real time parameterisation can be computationally expensive.

Instead of parameterisation, another rather simpler way of representing 3D objects surfaces is through deformation-based representation. To reconstruct the 3D object's surface, a deformation field is produced from the 2D image, and then applying the deformation field to a template 3D shape. The types of deformation-based 3D representation for reconstruction that can be applied is vertex deformation, using morphable models and free-form deformation.

For vertex deformation, a deformation map entails the linear displacement of each vertex, with the assumption that the shape of the object has the same topology as the template of the object. Its feasibility has been demonstrated in [28][29][30]

For morphable models, a mean shape of the object is obtained and it is deformed using a set of orthonormal basis. A way to apply morphable models would be using PCA on a set of 3D exemplars. Techniques that used morphable models for 3D reconstruction managed to use only 2D annotations, such as 2D silhouettes or 2D images [31] [32], while requiring the object to be segmented and camera pose be estimated.

As for free-form deformation, the set of parameters that can be employed are control points that are located around the template, instead on the vertexes themselves, which when applied to the shape, can deform the space around the shape, thus it is able to form the shape without needing the use of templates that have a one-to-one correspondences with the shape. This has successfully been applied by Pontes et. al. [33], Kuryenkov et. al. [34] and Jack et al. [35].

To perform 3D reconstruction by deforming a 3D template, the 3D template to be used must first be defined. Several deformation-based 3D reconstructions use a common 3D shapes such as a sphere [28] and an ellipse [29]. Henderson et. al. instead proposes two types of templates, using cuboidal primitives to create the abstract structure of the complex

shape, and a cube subdivided into multiple vertices. Instead of settling on one type of template, Kuryenkov et. al. [34] proposes DeformNet, which instead performs searching in a database a shape that matches the image of the object, and then deforms it using the free-form deformation model. Another method of defining the template is to learn the template from a set of templates that belong to a specific class, and then learn the mean template, which was demonstrated by Tulsiani et. al. [32], whom they use a deep neural network to predict the class of the input, and then the deformation field that needs to be applied.

Another rather popular method of representing 3D objects through the surface of the 3D object is with point clouds. The 3D shape is comprised of a set of points that are marked with their corresponding coordinates in the 3D space. The fact that points are used to represent the objects make them simple but rather efficient in terms of memory usage.

To represent 3D objects as a point cloud, approaches that have been used to stores these point information are, the most straightforward one, as a set of point representation with N x 3 matrix [36], a 3-channel grid of size H x W x 3 where each pixel encodes the coordinates of a 3D point [37], and depth maps that represents the depth of the object viewed from multiple viewpoints [38]. Mandikal et. al. [36] uses fully connected layers to regress the set of points clouds of the object, by training a 3D point cloud-encoder and then learn the mapping between the 2D image and the corresponding latent embedding. Fan et. al. [37] instead trains a conditional shape sampler, which is able to predict several plausible 3D point clouds from a single input image. The main challenge of utilising point cloud representation is the ability to recover the texture of the object, as it depends very well on the quality of the recovered point cloud. As the surface of an object may be irregular or complex, it is difficult to map the texture of the 2D image onto the surface of the object, while ensuring the recovered object is natural and realistic.

Single-view 3D Reconstruction

There has been numerous works of single-view 3D reconstruction in the last 10 years. However, especially early works, rely on 3D supervision (such as 3D ground truth) to facilitate reconstruction. **Acquiring 3D ground truths is an expensive and tedious task**, especially on natural objects. As such, much was done using synthetics datasets. The main challenge with this is that it does not work well when inferring real world dataset, since real world datasets have a rather natural shape and contains details such as texture, which results in a domain adaptation problem where models train on synthetics need to adapt to real world natural objects.

Instead, much work has been shifted towards using 2D information to supervise the reconstruction of the 2D images. As a substitute for ground truth 3D models, much can be relied on annotations such as keypoints, poses or 2.5D images of the object, since they are rather easier and cheaper to obtain. The assumption is that, if the reconstructed 3D model is as closed as possible to the actual object, then viewing the 3D object at a viewpoint would be the same viewing the actual object from the same viewpoint.

To enforce the similarity of the reconstructed 3D model and the object when viewing from the same viewpoint, the 3D model must first be projected and rendered onto the 2D view. To enable end-to-end training however, the projection operation must also be differentiable. Fortunately, recent advancement in **differentiable renderers** has allowed the gap between 3D object domain and 2D reprojection domain to be bridge, using different techniques that are suitable for reconstruction task of different representation. Kato et. al. [28] proposes a neural 3D mesh renderer (MNR), which can be applied on mesh and is differentiable by approximating the gradients. Another notable differentiable renderer is DIB-R [39], which instead approximates the rendering by utilising neural networks, to perform the rendering process by simulating the way lights interacts with the mesh of the object, and it divides the task by processing two different region of the image independently, which is the foreground and the background. Gadelha et. al. [40] instead proposes a differentiable renderer that can be utilised on 3D voxel grids. OpenDR [41], introduced by Loper and Black, is one of the first open source general-purpose differentiable renderer and it can be utilised on different 3D representations. Instead of using a fixed rendering algorithm Rezende et. al [42] aims to learn the projection of 3D to 2D image by utilising 3D and 2D convolutions.

After an 3D object is reprojected onto the 2D image space, there is a need to **calculate the discrepancy (loss) between the 3D reprojection and the 2D image**. A common method of calculating the loss is through utilising the silhouette of the object, where the projected silhouette of the reconstructed object, under certain lightning condition and pose/viewpoints parameter, should be as close as possible to the silhouette of the actual object in the image. The difference between the projected silhouette and the silhouette of the actual object can be calculated using several distance metric such as **Euclidian distance** (L2-norm), **Intersection over Union** (IoU), and **binary cross-entropy loss** (BCE Loss). Kundu et. al. [43] utilises both IoU and L2-norm in the loss of the reconstructed object by using IoU to calculate the similarity between the object and rendered silhouette, and using L2 distance to calculate loss between depth image of the image and the depth image of the rendered reconstructed volume. Yan et. al. [44] and Zhu et. al. [45] used a form of binary cross entropy loss that is differentiable. In [46], Zhang et. al. mentions that obtaining the mask of the reconstructed object involves rasterization that discretises the mesh into image pixels, thus resulting loss of information and bad supervision signals. By taking inspiration from point clouds data structure, Zhang et. al. used Chamfer Distance to calculate a Chamfer Mask Loss and a Chamfer Texture Loss, thus providing a robust form of measuring loss and distance of the silhouette mask and texture map respectively. In addition to the use of silhouette map, Wu et. al. [10] introduces MarrNet that uses additional cues such as normal map and depth map to guide the training process.

To project and render the 2D image projection from the reconstructed 3D model and compare it to the ground truth image, the camera parameters and the viewpoint of the ground truth image must be also identified. Some of the 3D reconstruction method assumes that the corresponding pose of the image is already known in advance. For the current dataset that is available, the pose of the corresponding image is not known. As such, there is a need to perform pose estimation on the dataset.

To apply pose estimation on the dataset, it can be done using a pose predictor model, or initialise the camera pose and then jointly optimise with reconstruction. Gadelha et. al. [40] used a fully connected layer to encode the pose as a latent pose code, besides encoding the image into a latent representation. Then the latent pose code is input into a 2D projection module to render the reconstructed 3D object. Kendall et. al. [40] proposed PoseNet, a CNN that is tasked to predict the pose of the object from a single image. In Pavllo et. al.'s work [47], while assuimng a weak-perspective camera perspective, they perform segmentation on the images to obtain a foreground mask, while also dividing up the foreground mask into several parts, where each part corresponds to a semantic part of the object, and then optimised a category-specific mesh template to obtain a semantically coloured 3D mesh template of the corresponding category, which is used to estimate the pose of the images by maximising the IoU between the silhouette of the images (and semantic segmentations) and the coloured mesh template. In CMR [30], they utilised annotated key points of the dataset and apply structure-from-motion to obtain a rough estimate of the weak perspective camera.

GAN Inversion

GANs has seen significant application in the field of computer vision. It has been applied on image synthesis that are of high quality and diversity, while being close to the data image that it has been trained on. As such, GANs has also been applied to 3D reconstruction from single 2D images.

How can generative models such as GANs be utilised for 3D reconstruction? The answer lies in the intrinsic properties of a GAN. We can think of 3D objects as having made up of a certain geometric shape and texture. We can employ the use of encoders such as VAE to encode the object into a latent space, where heading in a certain direction of the latent space changes only the geometric properties of the object, while heading in another direction only changes the texture of the object. A GAN can be taught to learn the distribution of the object in the latent space and learn the mapping from a latent code to the corresponding object.

However, as we need to deal with 2D images, this adds an additional extrinsic parameter that needs to be also dealth with, which is the pose of the object. A 2D image dataset can be thought as the previously conceived distribution of shapes and texture plus the pose of the camera. Since we do not have any methods to explicitly teach the GAN model what shape in a 2D image it corresponds to without using ground truth 3D data, we shall instead assume that a category in the dataset has its own random distribution of shape and texture for that particular category, and have the generative model of GAN learn that distribution. The main idea is that, even when the generator of the GAN samples from the random distribution a random latent code, once the generator has learnt the distribution, the generated data is as closed as possible to the real dataset.

To have a generator learn the distribution of the dataset, GANs employ an adversarial training scheme to force the generator to learn the distribution of possible 3D object of category in the dataset. In GANs, the generator model does not have direct access to the

dataset itself, rather it must learn blindly while being guided by the discriminator. While sampling from randomness, the generator initially learns to generate random, out of place data that is nowhere close to data in the dataset. Unlike in the rather commonly known Artificial Neural Network (ANN), where an input into an ANN produces an output and it must learn the target output, a discriminator in GAN serves to penalise the generator if the generator generates data that are far from being similar to data in the dataset. A discriminator in GAN can learn the distribution of the dataset fast since it has been explicitly taught to accept images from the dataset and reject images from the generator, and as such it can be used to steer the generator towards learning a good distribution of the dataset.

So how can GANs be used to learn 3D reconstruction then? It has been shown in several research that GANs can indeed learn the rich underlying semantic information [48] and thus implicitly learn the 3D shape of the object from 2D images. As such, instead of heavily rely on 2D supervision, we can instead rely on a handful of 2D information about the 3D object to have the GAN learn priors, and then use those priors to reconstruct the 3D object faithfully. After having learnt the latent space of the 3D manifold, by changing and varying the latent code of the input to fit our corresponding image that is needed to be reconstructed, we can indeed utilise GANs for reconstruction. Therefore, there is a new method to employ a pretrained GAN and perform an inverse mapping of GAN so that the 2D image is encoded into its matching, corresponding latent code, such method is called GAN inversion.

Given a generator in a GAN that has learnt how to map a latent code to an output, GAN inversion aims to invert an image to a corresponding latent code, so that the generator can use the latent code to output the intended 3D reconstruction or immediate representation of the 3D reconstruction. Specifically, given a latent code z that corresponds to an image, such that when latent code z is decoded and resulting output is similar to the image or the reconstructed 3D object, GAN inversion aims to train or optimise an encoder that it is able to convert the same image to a latent code z* and said latent code z, when decoded by the generator, is as close as possible to the input image or the 3D object model, normally through gradient descent. The encoder in this case learns the inverse mapping of the GAN.

There are two main ways on how GAN inversion can be utilised on 3D reconstruction, a learning-based inversion method, and an optimisation-based inversion method. While optimisation-based inversion result can faithfully reconstruct a quality reconstructions, they however consume more time since they require searching through latent space for the matching latent code, and meanwhile a learning-based inversion is pretrained to predict for the optimum latent code in the latent space, therefore trading quality for faster time. Pan et. al. [49] utilised a pretrained GANStyle2 as the GAN model to perform GAN inversion. Using an ellipsoid 3d shape, they render several unnatural images that are sampled from random viewpoints and lightning to generate pseudo-samples. After that they are used to guide the image towards reconstructing the same images, while being look like as if they were sampled from the same viewpoints and lightning conditions in the pseudo-samples, resulting in natural-looking images which can be adopted as ground truth multi-view image of the input image, and then are used as priors to perform reconstruction. Zhang et. al. [50] instead

trained from scratch the generator on point cloud-based 3D shapes, and then the generator uses the encoded latent code to reconstruct the complete shape of the object. In a different paper, Zhang et. al. [51] instead uses a convolutional mesh estimator called ConvMesh to generate pseudosamples of deformation map and texture map of input image to train a GAN, after that GAN inversion is performed on the pretrained GAN to produce the 3D reconstruction.

Methodology

The 3D representation of the current approach uses a mesh representation that is formed by deforming a UNV sphere mesh template, using a deformation map and texture map that produced from the input image.

The methodology employed for training and reconstruction is made up of 4 parts:

- Image and semantic segmentation
- Pose estimation framework from Texture3DGAN [47]
- Convolutional mesh estimation from Texture3DGAN [47], which is built upon ConvMesh [51] to generate pseudo-ground truth mesh and texture
- MeshInversion [46] that is trained on the pseudo-ground truth mesh and texture map, and then performing GAN inversion on the test dataset

Image and semantic segmentation

For image and semantic segmentation, the segmented mask of the seed needs to be obtained, which will be used for both pose estimation and reconstruction, while two semantic mask which corresponds to different parts of the seed i.e. the germinated part and seed body respectively also need to be obtained, which will be used only for pose estimation. In each of the seed image, the germinated seed is placed on a white background, with the seed projecting its shadow onto the white background.

Rich saturation information is generally focused on the germinated part of the seed, with some sprinkled on the seed body. After computing the histogram of the saturation image, the histogram exhibits a unimodal shape, with the frequency of the saturation values focused on lower parts of the histogram, since there are a lot of saturation-less pixels. Otsu's thresholding [52] method is applied on the saturation image to segment the germinated part of the seed. While the segmented saturated image does include the whole of the germinated part, it also includes areas around the seed body that were also high in saturation. To clean up the area around the segmented saturation image, a morphological opening operation was performed with a kernel of seven, and then contour searching was performed to obtain the largest area in the binary mask, which should correspond to the germinated part.

Given the segmented binary mask that covers the germinated part, we perform Otsu thresholding [52] on the grayscale image of the seed to segment the white background. However, the germinated part is also high in brightness, so the previously obtained germinated part mask is used to remove the area of the segmented white background mask. This results in a background binary mask which does exclused the seed, however it excludes contains the shadow of the seed projected onto the white background ,since they had lower brightness value than the thresholded brightness. The obtained mask is negated to reflect the foreground mask (while still including the shadow).

To remove the shadow from the segmented seed as well, the snake model active contour segmentation algorithm [53] is employed here to obtain the foreground seed binary mask. By using the previously obtained seed mask which erroneously includes the non-seed shadow part, it is used to initialise the area for active contour segmentation [53] to be

performed. This results in a seed mask that is well segmented from the white background, with the shadow properly removed from the segmented seed image.

To produce the binary mask that corresponds to the seed body, the germinated parts binary mask is negated and a bitwise AND operation is performed on the foreground mask. In the end this will produce 3 mask, the binary mask of the whole seed, the binary mask of the germinated part and the binary mask of the seed body.

Pose estimation using Textured3DGAN

The pose estimation framework that is utilised is based on the approach in Textured3DGAN [47], where the estimated pose is required to initialise the mesh estimation process. The camera projection model that used is a weak-perspective camera model, which is parameterised by a rotation $q \in \mathbb{R}^4$ (quarternion), a scale $s \in \mathbb{R}$, a translation $t \in \mathbb{R}^2$ and a perspective correction term $z_0 \in \mathbb{R}$.

The method adopted in Textured3DGAN combines the idea of mesh template-based and semantic-based pose estimation techniques. Initially, given a mesh template that belongs to a specific object category (such as a seed), several camera hypothesis (in this case 40 camera) are initialised and optimised to minimised the difference between the silhouette mask of the rendered view and the silhouette mask of the seed. Since there are poses that will be ambiguous, an v_{agr} agreement score is calculated and any image having a score higher than the threshold is discarded. This would lead to significantly shrinking of available data, and as such the next step is performed to possibly reinstate the discarded images. By using images with unambiguous pose, a 3D semantic template is inferred on the mesh template by using the segmented semantic masks. After that the poses estimation step is repeated on the ambiguous ones but this time using the semantic mask and a new score is calculated, with, again, the ones with score higher than the thresholded value is discarded.

A mesh template is utilised in both pose estimation, and generation of pseudo-ground truth deformation and texture map. To obtain a seed mesh template, a suitable seed mesh template can borrowed from the internet, or modelled using Blender. For this experiment, the mesh templated was commissioned and produced by a student from UNMC. As mentioned in [47], an important part of preparing the mesh template is remeshing the mesh template so that it topologically aligns with the UV sphere, which is the main mesh template object that will be used during reconstruction. It has also the effect of reducing the complexity of the mesh template and thus speeding up optimisation. More details about mesh template remeshing can found in the supplementary materials of [47].

In the initial silhouette optimisation step, creating several camera hypotheses with different initialisation is required to avoid failing into local minima. The objective function is then to minimise the mean squared error between the silhouette rendered from the mesh template and the silhouette of the image, specifically:

$$\min \left\| \mathcal{R}\left(V_{tpl}, F_{tpl}; q, t, s, z_0\right) - x \right\|^2$$

Where $\mathcal{R}$ is the differentiable renderer DIB-R [39].

V_{tpl} representing the fixed vertices of the mesh template.

F_{tpl} representing the mesh faces of the mesh template.

The camera hypothesis is optimised using a variant of Adam [54] from Textured3DGAN[47].

Since badly estimated poses can significantly affect and lower the training and reconstruction performance, there is a need to identify and discard bad data. It mainly stems from silhouette mask that are highly matching, but are viewed from opposite viewpoints, making such poses ambiguous and harmful to the training scheme. A pose estimate is considered confident if the intersection-over-inion (IoU) between the rendered and target silhouette mask are high, while considered unambiguous if the viewpoint that it describes is not significantly different than another pose estimate that is also high on IoU. Each camera hypothesis is describe by a V_{conf} confidence score based on their IoU, and the geodesic distance is calculated for each possible camera pair. In the end a v_{agr} agreement score is computed based on the confidence score of each camera pose and their geodesic distance with other cameras, with details of the calculation can be found in section 3.1-Scoring and ambiguity detection in [47]. If the resulting score is 0, it means all camera poses are confident and has the same camera parameters. If the score is 0.5 instead, this means the camera poses are rotated by 180 degrees from one another. It is empirically established in [47] that only images with v_{agr}<0.3 should be accepted.

To reinstate discarded data from the initial pose optimisation step, a 3D semantic template is inferred from the given mesh template, and with semantic masks of images with low v_{agr} score. The top 100 image that has the highest IoU will be employed to compute the semantic on the mesh template, dividing parts of the mesh template that belongs to the germinated part, and the parts belonging to the seed body. This becomes another optimisation problem where the goal is to minimise the MSE between the rendering of the semantically coloured mesh template and the 2D image semantics, details about calculation can be found in 3.1-Semantic template inference in [47].

Now that a 3D semantic template is computed, the v_{agr} agreement score is recomputed for the discarded image, with this time computing the mIoU (weighted Jaccard similarity) between the rendered 3D semantic templated and semantic mask of a discarded image. The worst 10% images in terms of IoU are discarded, and those with v_{agr} < 0.3 are reinstated back into the usable dataset.

Generation of pseudo-ground truth using ConvMesh

The purpose of generating pseudo-ground truth data is to assist in the training of GAN. The generation of pseudo-ground truth in [47] is based on the convolutional mesh estimation framework from ConvMesh [51], while adding extra channels that also predicts the semantic map of the object. The 3D mesh representation is based on a deformable mesh template, and then a deformation map is applied on the main mesh template to produce the mesh of the object, while a texture map is also predicted that applied on the deformed mesh to obtain the final 3D textured mesh. The reason why this form of 3D representation is

applied is because, it allows texture mapping between the output texture map and deformation map by sharing the same UV map, thus ensuring that both maps are topologically aligned which is critical in allowing the discriminator to jointly discriminate against the texture and deformation map. Producing a mesh by reconstructing the deformation map and then apply it on a mesh template also ensures that the resulting mesh is smooth since it takes advantage of the spatial correlation of the employed convolutional networks. To allow UV mapping without resulting in major distortions and gaps, a UV sphere is used as the deformable mesh template.

The mesh generation model utilised for producing the pseudo-ground truth data in [47] is based on a reconstruction model (from [51]) that aims to predict a mesh in the form of a deformation map, a texture map and a semantic map given an input image. To perform training on the mesh-estimating reconstruction model, the image of the seed, segmentation mask of the seed, the estimated pose and semantics masks of the seed image is used. Once training is finished, the generated deformation maps are used as part of the pseudo ground truth data. The generated texture map however is discarded. Instead, the original image is used as the pseudo texture map by projecting the pixels of the image onto the UV mapping of the mesh through an approach called inverse rendering. To compensate for the occluded surface in the image, a visibility mask is produced and applied on the inversely rendered pseudo texture map so that backpropagation is calculated using only the non-occluded surface.

While both the generated texture map and semantic map is discarded during training of the GAN model, they serve as a regulariser for the generation of the pseudo deformation map and are thus are included during the calculation of the mesh-estimating reconstruction loss.

The mesh estimating model used in [47] is based on ConvMesh[51], where an input image is encoded using a 2D convolutional encoder, compressed into a latent code, and then decoded by a 2D convolutional decoder that branches into 2 main output, with the output being a texture map, displacement map, and an extra k channels added by [47] (where K = 2 in this case since there is only 2 semantics, the germinated part and the seed body) that outputs semantics mask highlighting which part of the image belongs to what semantic. The details of the architecture for the main mesh estimation model can be found in appendix A.1 of [51].

The loss function of the mesh estimation model is calculated using the Mean Squared Error, between the rendered silhouette and target silhouette (segmentation mask) of the seeds, the predicted texture and the texture from the 2D seed image, and a predicted semantic texture of K = 2 channels and the semantic masks of the seed. The camera model used is the same as the one mentioned in pose estimation, with s, t and z_0 allowed to be fine-tuned by the optimizers. A smoothness loss is also used to ensure neighbouring faces on the mesh have similar normals.

Once the mesh generation model is trained, the pseudo-ground truth deformation map is obtained by performing a single forward pass on the seed images. Instead of using the predicted texture, the texture from the image is projected onto the UV map by

performing inverse mapping to obtain the pseudo-ground truth texture and the associated visibility mask, refer to [51] section 3.1 – pose-independent representation for more information about performing inverse mapping to obtain the pseudo-ground truth texture map.

The pseudo-ground truth deformation map and texture map is used for adversarial training of the generator in ConvMesh [51].

MeshInversion pre-training and GAN inversion

The current framework used for performing 3D reconstruction of the germinated parts of the seed is MeshInversion [46], where given a pretrained 2D convolutional GAN which is based on ConvMesh, the 3D reconstruction task is converted into optimising the latent code that best recovers the textured map and deformation map that can well described the single-view image.

Before GAN inversion, the 2D convolutional generator must be pre-trained in order to capture prior knowledge of deformation and texture map. The pre-trained GAN needs to be able to reconstruct the geometric shape and texture appearance of the germinated seeds while only the single-view seed image and its associated mask is supplied. In MeshInversion [46], they obtained the silhouette (segmentation) mask of the image by using a pre-trained PointRend [55] which is trained the COCO dataset [56]. However, as the current object that is being reconstructed is not part of the COCO dataset, we instead perform our own segmentation technique, as described in the first step of the methodology.

The goal of pre-training a textured GAN i.e. ConvMesh [51] is to learn the required prior knowledge for performing 3D reconstruction on the seeds, and as such are trained on the generated pseudo-ground truth data, since there is no ground truth data available for the germinated seeds. In the discriminative training of ConvMesh [51], the generator generates a deformation map and a texture map, where the deformation map is used to deform the same deformable mesh template in Textured3DGAN i.e. an UV sphere, and is used to discriminate against the pseudo ground truth dataset in the UV space. Besides performing discrimination in the UV space, MeshInversion [46] introduces a discriminator in the image space to reinforce the realism of the texture and deformation map generated from the generator, with the architecture of the image space discriminator being based on PatchGAN [57]. The calculation of the loss function is based on the least-square losses in [58]. The loss function is as follows:

- Deformation map and texture map generator

$$\mathcal{L}_G = \lambda_{uv}\mathbb{E}_{z\sim P_z}\left[\left(D_{uv}\big(G(z)\big)-1\right)^2\right] + \lambda_I\mathbb{E}_{z\sim P_z(z)}[(D_I(R\big(G(z)\big),\pi)-1)^2]$$

- UV space discriminator

$$\mathcal{L}_{D_{uv}} = \mathbb{E}_{S,T\sim P_{pseudo}}[(D_{uv}(S,T)-1)^2] + \mathbb{E}_{z\sim P_z(z)}\left[\left(D_{uv}\big(G(z)\big)\right)^2\right]$$

- Image space realism discriminator

$$\mathcal{L}_{D_I} = \mathbb{E}_{I\sim P_{data}}[(D_I(I)-1)^2] + \mathbb{E}_{z\sim P_z(z)}\left[\left(D_I(R\big(G(z)\big),\pi\right)^2\right]$$

Where D_{uv} refers to UV space discriminator

D_I refers to image space discriminator

And λ_{uv} and λ_I are the corresponding weights

The camera model used is also a weak-perspective camera projection, same as the one used during pose estimation of the dataset, where the camera pose π is parameterised by rotation in the form of a quaternion $r \in \mathbb{R}^4$, scale $s \in \mathbb{R}$ and translation $t \in \mathbb{R}^2$, with the perspective correction term is dropped since we can know assume the camera is "far away from the object", due to the invidual seeds being small and the camera that is used to capture the seeds is relatively far from the seed objects. Once pre-training is done, GAN inversion can be performed to obtain the 3D reconstruction of the seed image.

Given a pretrained ConvMesh [51] that can generate a deformation map and a texture map, the purpose of GAN inversion is to find a latent code z such that the generated textured 3D mesh is as close as possible to the given input image $\mathbf{I}_{in}$ and silhouette mask M_{in}. To perform the search in the latent space, gradient descent is utilised so that a latent code z can be found that minimalizes the reconstruction loss L_{inv}:

$$Z^* = \arg\min_z L_{inv}(R(G(z),\pi),\mathbf{I}_{in})$$

To compute the reconstruction loss L_{inv} however, the reconstructed 3D needs to be projected into a 2D image, which requires knowing the ground truth camera pose in advance. As mentioned in [46], concurrently optimizing the camera pose with the latent code may lead to camera pose that are ambiguous, as described in the pose estimation step. While the pose estimation step that is used during pose estimation uses semantics and thus gives a better pose approximation, it does not ensure near-perfect pose estimation, and there are images that have a relatively high v_{agr} score owing to their poorly segmented or ambiguous silhouette mask. As such, [46] proposes a robust form of calculating mask and texture loss while also allowing the camera pose to be optimised along with the latent code:

$$Z^*,\pi^* = \arg\min_{z,\pi} L_{inv}(R(G(z),\pi),\mathbf{I}_{in})$$

Taking inspiration from point cloud data structures, [46] employs Chamfer distance to perform a Chamfer Texture Loss and a Chamfer Mask Loss.

As mentioned in [46], they treat a 2D image as a set of 2D coloured points, where each point has an appearance attribute (RGB values) and spatial attribute (coordinates in the image grid). To calculate a Chamfer Texture Loss of the rendered image and the target 2D image, they used Chamfer distance to calculate a dissimilarity measure between two set of coloured points that are taken from the two images, which considers both the appearance and the location. The main motivation behind the idea is that they allow tolerance if the pixels slightly misaligns, while heavily penalise if a large misalignment takes place. During experimentation only 8096 randomly selected pixels are considered. Besides applying on the

rendered and input images themselves, they use a pre-trained VGG-19 [59] network to extract the feature maps of the foreground of the rendered image and the input image, and apply the same Chamfer Texture Loss on the obtained feature maps, thus allowing them to calculated a contextual loss. More details about the Chamfer Texture Loss calculation can be found in section 3.2 in [46]. This results in a pixel-level Chamfer Texture Loss L_{pCT} and a feature-level Chamfer Texture Loss L_{fCT} to calculate the appearance loss of the 3D reconstruction.

Besides Chamfer Texture Loss, a Chamfer Mask Loss [46] is also employed to calculate the dissimilarity of the silhouette of the reconstructed 3D mesh and the target silhouette. Their argument is that a small change in the latent code z should correspond to a small change in the 3D shape of the mesh. However, during the rasterization process to obtain a binary mask for computing the similarity between the predicted silhouette and target silhouette, that small change may not necessarily be reflected in the rendered binary mask itself, and this results in loss of information and thus reduce the effectiveness of silhouette supervision. Instead of rendering the binary mask of the reconstruction, the vertices of the 3D reconstructed mesh are project onto the image plan using the current camera pose, resulting in a set of points that corresponds to the reconstructed mesh's silhouette mask, while the foreground pixels of the target silhouette is also project onto the image plane, normalised to a range of [-1, 1] to obtain a set of points that corresponds to the target silhouette. After that Chamfer distance is computed between the two set of points to obtain a Chamfer Mask Loss L_{CM}. More details can be found in section 3.3 in [46].

Beside Chamfer Texture Loss L_{pCT} and Chamfer Mask Loss L_{CM}, a smooth loss $L_{smooth}[28]$ is also computed to encourage neighbouring faces to be smooth by having similar normals, while a latent space loss L_z is also computed to encourage the latent code z to have a Gaussian distribution (which serves as a regulariser).

The resulting inversion loss is calculated with the 5 previously mentioned reconstruction loss:

$$L_{inv} = L_{pCT} + L_{fCT} + L_{CM} + L_{smooth} + L_z$$

Experiments and results

Dataset

The germinated oil palm seeds dataset that is acquired has three batches collected from AAR, with different light conditions and using different camera models. The first batch of seeds were taken with a digital camera Samsung NX2000 and a Sanoto MK40 Photo Light Box was used as a tool to emit light under the light box condition, with the camera attached to a tripod fixed 40 cm from the light box and the setting set to autofocus. It contains 110 images of good quality seeds and 105 images of bad quality seeds, and each image contains around 10-11 individual seeds with a clear gap between seeds. 90 out of 110 images of the good quality seeds and 85 out of 105 images of bad quality seeds were used for training. In the end there were 901 good quality individual seeds and 851 bad quality individual seeds in the training seed, while in the test set there were 201 good quality individual seeds and 200 bad quality individual seeds. In total there are 1752 individual seeds in the training set and 401 seeds in the training seed.

The second batch and the third batch were collected under two different lightning conditions. For the second batch, it was collected under normal room light, with the source of light emitted from Phillips Lifemax TLD 36W/54-765 fluorescent bulb. For the third batch, it was collected under light box condition, where the Sanoto MK40 Photo Light Box was used as a tool to emit light. Both batches were instead captured using a digital compact camera Canon IXUS 285 HS, with the camera fixed 26 cm above a tabletop where the dataset were positioned, and its setting was set at autofocus function. In total there are 900 individual seeds for batch 2, with 450 good quality individual seed and 450 bad quality individual seed, while in batch 3 there was 1198 invidual seeds, with 605 good quality individual seeds and 693 bad quality individual seeds.

Good quality seeds from batch-1

Bad quality seeds from batch-1

Good quality seed from batch-2

Bad quality seeds from batch-2

Good quality seeds from batch-3

Bad quality seeds from batch-3

Image and semantic segmentation

To perform pose estimation using semantics, the binary mask of the whole seed, the binary mask of the germinated part and the binary mask of the seed body need to be obtained. In Textured3DGAN [47], they used PointRend [55] that is pretrained on the COCO dataset to identify and segment the foreground from the background. However, in this case germinated palm oil seeds are not part of the category in the COCO dataset, and thus are not able to segment the seed off the background. As for the semantics, they [47] used a semi-supervised object detector from [60] to segment object parts in the foreground, and while the object detector was trained from 3000 classes that are available in Visual Genome (VG), it did not contain a “germinated part of oil palm seed” class, nor did it include a “seed body of oil palm seed” class in the dataset. As such it was required to experiment with various segmentation techniques to obtain the binary masks.

As various segmentation techniques were experimented on the batch-1 seed to separate the germinated oil palm seed off the background, and the germinated part from

the main oil palm seed body, in the end it was settled with using active contour segmentation [53] and saturation image thresholding.

To obtain a mask of the germinated part, first the saturation image of the image is obtained by converting the image to HSV, and Otsu thresholding [52] was performed. Next a simple morphological operation of opening was performed to remove any saturation "noise". Finally a simple contour searching was performed to obtain the largest contour.

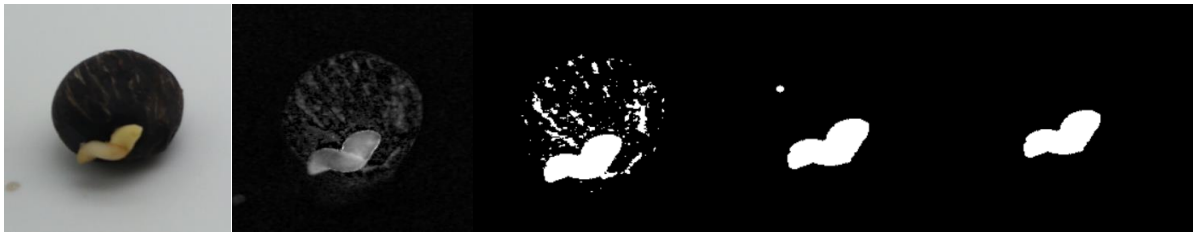
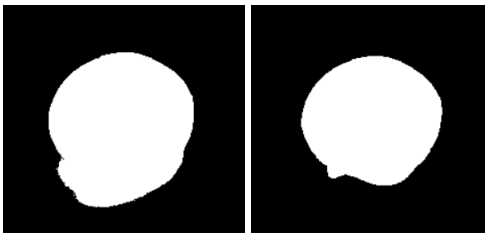


Image -> Saturation -> Thresholded saturation -> Morphological opening -> Search for biggest contour

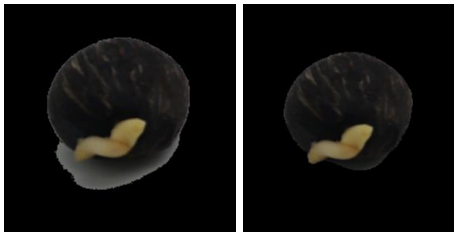
To separate the background from the image, Otsu thresholding [52] was performed on the grayscale image of the seed. This however erroneously includes the germinated part and excludes the shadow. To deal with this, the previously obtained germinated binary mask can be used to filter out the germinated part from the background. First the obtained background is negated to obtain the supposedly background mask, and then a bitwise OR operation is performed between the supposed foreground mask and germinated part mask, this leaves the shadow left to be removed.

Image -> Gray scale -> Background (includes germinated part, excludes shadow) -> Negated background -> Foreground (includes shadow)

To remove the shadow, the snake model active contour segmentation technique [53] is employed here. The current foreground mask (which erroneously includes the shadow) is used as the initial contour. As for the parameters, some experiments were done on tuning the alpha, beta and gamma parameters. In the end, the alpha was chosen at 0.5, beta at 1.0 and gamma at 0.01, the maximum iteration was set at 500.



Initial contour -> After active contour segmentation

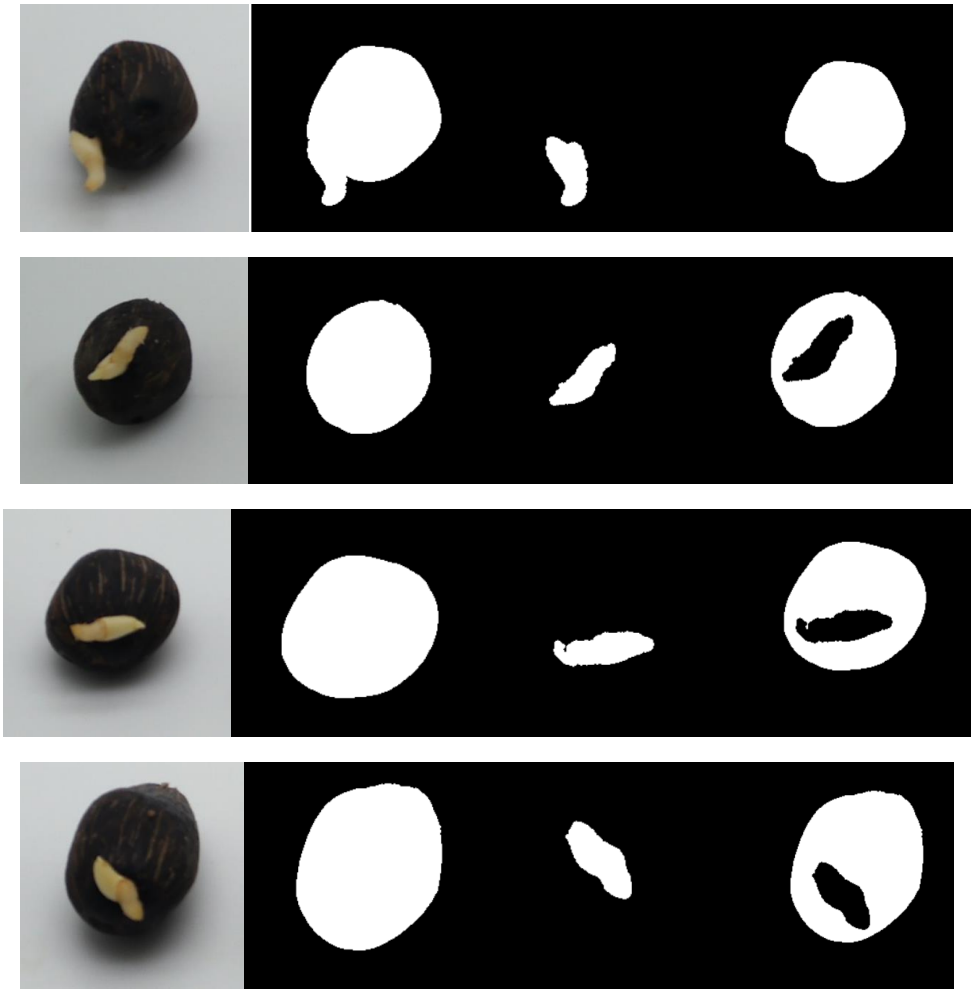


Foreground seed (with shadow) -> Foreground seed (without shadow)

As for the mask of the seed body, it can be simply obtained by negating the obtained germinated part binary mask and performing a bitwise AND operation on the foreground seed mask.

Foreground mask – germinated mask = Seed body mask

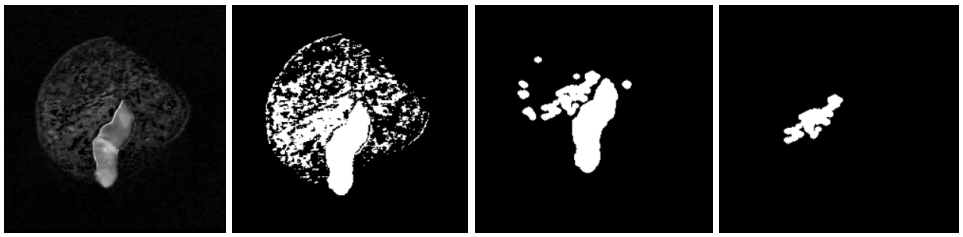
More results:



First column – Input image; Second column – Seed mask; Third column – Germinated part mask; Fourth column – Seed body mask

However, not all the seed can be successfully segmented into their corresponding mask, possibly due to a noisier saturation image.

A badly segmented image (Left to right: Input, whole seed, badly segmented germinated part, seed body)

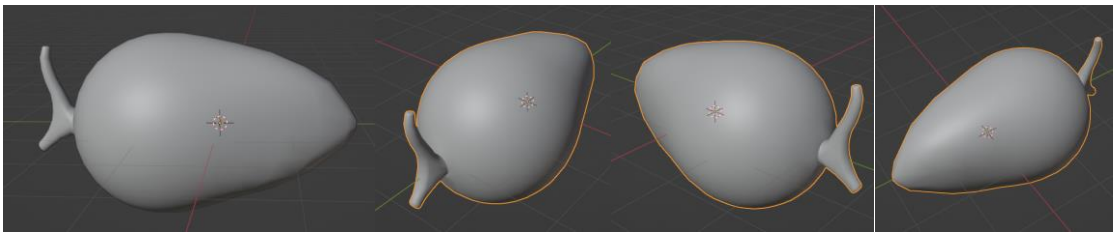


Process of the segmentation; the contour searching algorithm mistakenly picked the wrong area

The same algorithm was applied on batch 2 and 3 as well, however due to differing lightning condition and camera model used to take the pictures, the seed images were also poorly segmented. As such the decision made was to use only the batch 1 seed for training and testing.

Pose estimation

After obtaining the segmentation mask of the seeds and their corresponding semantic mask (a germinated part and a seed body), pose estimation can be performed by utilising a germinated palm oil seed mesh template.



Germinated palm oil seed template; viewed from different angles

As mentioned in methodology, the pose estimation is performed in two steps, by utilising only silhouette (segmentation) mask first, and then using the semantic masks (germinated mask and seed body mask).

Since the dataset does not have any key points to estimate pose using structure-from-motion (unlike in Textured3DGAN), the accuracy of the estimated poses is not calculated. The pose estimation framework provided by [47] does allow computing poses estimation using multiple mesh template. There is an attempt to find mesh templates of

germinated oil palm seeds (or similar ones) online yet the geometric shape of the found seed mesh templates are quite different than the ones in the dataset. With this, a student from UNMC was commissioned to create a single germinated oil palm seed template.

Pose estimation using silhouette only (whole seed segmentation mask) is performed on the 1752 invidual seeds using the method described in methodology, with 40 camera hypothesis randomly initialised for each seed image and then optimised. It was performed on the university's HPC and the whole process takes around 5 ~ 6 hours. The hyperparameters chosen were the same as the ones used in Textured3DGAN, where the optimiser chosen was Adam [Textured 3D GAN, 25], with $\beta_1 = 0.9$ and $\beta_2 = 0.95$. Each camera hypothesis is optimised for 100 iterations with a learning rate of LR = 0.1, decayed to 0.01 after the 80th iteration. More details about the silhouette-pose estimation process and hyperparameters tuning can be found in the A.1 supplementary materials [47].

Once the initial silhouette-pose estimation is done, the semantic-pose estimation process is performed to infer a semantically-coloured template on the germinated palm oil seed mesh template, and then to recalculate the v_{agr} agreement score of the images.

To infer the semantic template for the germinated palm oil seed mesh template, among images that have v_{agr} agreement score lower than 0.3, the top 100 images with the highest IoU are used. The semantic template is computed in a single pass following a closed form solution. More details about inferring semantic template can be found in A. Supplementary materials – semantic template inference from [47].

Once the germinated oil palm seed mesh template is semantically coloured, the v_{agr} scores are recalculated with this time substituting the IoU with mIoU, that is calculated using the semantic masks of the seed instead of the silhouette mask. For the training set, the worst 10% images in terms of mIoU are discarded, as well as image with v_{agr} higher than 0.3. For each training image that are not discarded, the best camera hypothesis in terms of mIoU is selected. For testing set however, the same pose estimation process is applied, with none of the images in the testing discarded.

Out of the 1752 images in the training set however, only 925 images passed the v_{agr} agreement score threshold.

Generating pseudo-ground truth data

To generate the pseudo-ground truth data, Textured3DGAN [47] adopted the same mesh estimation model from ConvMesh [51] while adding k extra channels (in this case is 2 for two of the semantic map, germinated part and seed body part) for producing semantic maps as well.

The architecture of the mesh estimation model of ConvMesh can be found in A supplementary material of [51], where the encoding block is made up of 2D convolutions and linear layers, followed by the decoding block where it is made up of residual blocks following the ResNet architecture and upscaling layers, and then branching into two main branches to produce a deformation (displacement) map and a texture map. The mesh estimation (reconstruction) model is purposedly overfitted by training for 130k iterations with batch size of 32. The optimiser used in this case is the Adam optimiser [54] with an

initial learning rate of 0.0001 and then halved at ¼, ½, and ¾ of the training iteration. More information about the hyperparameters of the model and settings can be found in section 3.2 and the supplementary material section of [47]. Training using the university HPC takes around 8~9 hours to finish.

Once training is done, a single pass is performed for each image and the predicted mesh (in the form of a displacement map) is used as the pseudo-ground truth mesh, while the predicted texture map and predicted semantic map is discarded. As for the pseudo-ground truth texture map, an inverse rendering is performed for the masked input image to project it onto the UV map, resulting in a pose-independent representation. The pseudo - ground truth texture map is masked with a visibility mask to ensure only the non-occluded surface of the image is used in backpropagation. More details about inverse mapping can be found in section 3.1 – pose-independent representation from [51].

The obtained pseudo-ground truth data is used for training the 2D convolutional GAN.

Produced pseudo-ground truth data of goodtrain.png from batch-1 training dataset of good quality

From the left: Input image, pseudo-ground truth displacement map, pseudo-ground truth texture map, visibility map, semantically segmented map (green: germinated part, pink: seed body; the segmentation however is discarded)

Training of generator and performing GAN inversion

The obtained pseudo-ground truth displacement map and texture map is used to train the same generator in ConvMesh [51], while being discriminated against the same UV space discriminator. The hyperparameters and settings follows the same as the one mentioned in MeshInversion [46], but it is trained without the class conditional settings for 600 epoch. The generator is updated once every 3 iterations with a learning rate of 0.0001, while both the UV space discriminator and image space discriminator are updated concurrently twice every three iterations with a learning rate of 0.0001. The optimizer employed here is also the Adam optimizer [54] with $\beta_1 = 0$ and $\beta_2 = 0.9$. The weights used in the objective function is set respectively to, $\lambda_{uv} = 1$ and $\lambda_{uv} = 0.04$. The batch size is set to 32.

After training is complete, GAN inversion can be computed on the testing set, reconstructing the seed in terms of a deformation map and a texture map. For the inversion process, it is done in multi-stages with the total stages being 4, and each stage has different learning rates for the latent code and camera pose. The first stage is computed for 50 iterations, with the learning rate of the latent code set to 0.1 and learning rate of the camera pose set to 0.01. For the second stage it is set to also 50 iterations, with learning rates of 0.05 and 0.005 set for the latent code and camera pose respectively. For the third stage, it is

50 iterations, learning rate of 0.01 and 0.001 respectively. For the fourth stage it is 50 iterations and learning rate of 0.005 and 0.0005, respectively. Adam optimiser is used with $\beta_1 = 0$ and $\beta_2 = 0.99$.

The segmentation (silhouette) mask of the testing set is obtained using the same methodology and the camera pose initialised using the same pose estimation method from Textured3DGAN.

The results of inversion will be reported in terms of the FID (Fréchet Inception Distance) [61] and IoU, and some renderings will be shown. While there is no 3D ground truth available to compare with the reconstruction, nor is there any other baseline that can be utilised to indicate whether the FID is good or not, it alone may provide some insight on how the model trained with only 925 pseudo-ground truths can fare well.

A mean IoU will be computed between the rendered masks and the ground truths mask.

A FID 1 reports the appearance quality of reconstruction when viewed from the optimised camera pose.

A FID 12 reports the quality of the reconstruction by using the renderings of the reconstruction from 12 different views around the reconstructed mesh, from an azimuth of 0 degrees to 360 degrees with an interval of 30 degrees.

As mentioned in [46], a FID 10 is can be computed which reports the exact back view of the reconstruction, which corresponds to the occluded surfaces.

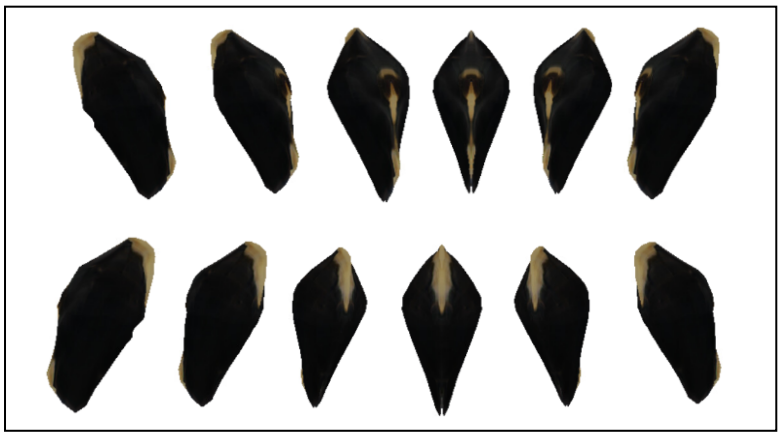
After that, the 12 viewpoints rendering of some seeds will be shown. A top-down view of the mesh will also be captured, to correspond to how the image of the seeds are captured.

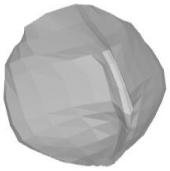
Qualitative results

Mean IoU	FID 1	FID 12	FID 10
0.842	243.61	217.50	217.83

Sample outputs

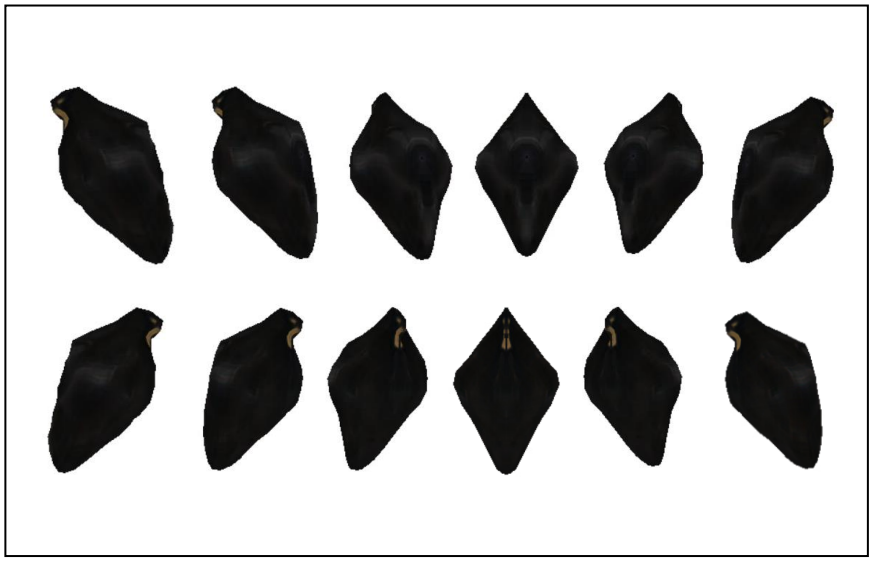
Goodtest2.png





Goodtest101.png

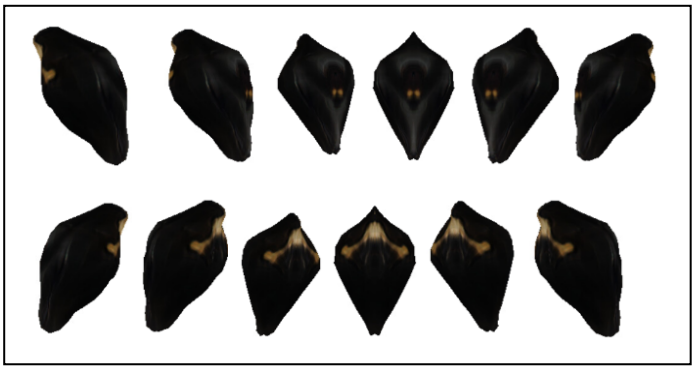
Goodtest109



Goodtest114.png

Goodtest100.png

Badtest83.png



Analysis and discussions

Analysis

From the rendering results, it seems like the germinated parts of the oil palm seed cannot be faithfully reconstructed. Nevertheless, there are some parts that is worth mentioning.

The model did learn a 3D reasoning towards the germinated parts of the seed, i.e. it recognises that the germinated part as being distinct from the seed body, as it tries to reconstruct a distinct germinated part on the mesh template.

Despite never shown the occluded back view of the seed, it has concluded from the texture priors that the occluded patch is similar to the texture patches of the seed body.

When viewed top down corresponding to the camera, the mesh still exhibit a seed-like shape similar to the silhouette of the seed mask, yet the geometric shape of the occluded back view is significantly distorted, suggesting that a variation of possible poses needs to be included in the dataset in order to properly learn the distribution of possible and natural seed shape, since a majority of the seed pictures are taken from a top down view.

An FID score of around 200 is considered relatively high and suggest that the gap between the distribution of the reconstruction and the distribution of dataset is significantly wide.

A high mean IoU of 0.842 while having an high FID score can be contributed to the fact the oil palm seeds has similar looking geometric shape as the mesh template, which is an UV sphere.

Discussions

It has been demonstrated that GANs have good generalisation learning power owning to their ability to implicitly learn the distribution of the variables, yet they have failed to generalise well in the current application. There are several possible factors that might contributed to the poor generalisation of 3D reconstruction of the germinated oil palm seeds.

One of which is the distribution of camera poses. In the dataset the image of the germinated palm oil seeds are often taken from a top-to-down, while hiding the back view of the seed from the camera. While this type of occlusion problem is solved by capturing from a frequency of unique viewpoints, in our case however the back side is seldom seen by the camera and as such the distribution of possible, natural occluded back view of the seed cannot be generalised properly by the generative model of ConvMesh. Besides the occluded back view, this also an obvious lack of geometric distinction between the germinated parts of the seed and the seed body. The results of inversion has indeed shown that the generative model has understood that the germinated part exhibits a unique geometric shape rather than being simply part of the seed body. In the dataset most images shows the germinated part lying in the foreground area of the seed body, instead of being a distinctive part of the seed. As such there is a need to capture the germinated palm oil seeds from the

sides of the seed such that a gap exists between the germinated part and the seed body for the model to learn.

The second factor that may contribute to the poor generalisation of the generative model is the segmentation technique being employed. The main part of the segmentation technique being employed is the usage of saturation thresholding. While the saturation thresholding can indeed separate the germinated part from the seed body for most of the seed image, however this may not be the case for several other seed image in batch 1, owning to the fact that different palm oil seed may exhibit different texture surface and as such may confuse the saturation thresholding process, resulting in a noisy segmentation and noisy mask, leading to some of the seeds having a high agreement score during pose estimation and is discarded from the training set. The saturation thresholding technique was also not successfully applied as well to the seeds in batch 2 and 3 as they are taken under different light condition and with different camera model. This has lead the shrinking of usable dataset to a mere handful, lowering from a good amount of 3850 individual seed image to 925 individual seeds images.

To remedy this, suggestion that has been made is to either employ a more robust method of segmentation or employ a segmentation model just like in ConvMesh [51] and Textured3DGAN [47], where they employ PointRend [55] to segment the object, and then utilised Visual Genome (VG) [60] to segment the semantics of the oil palm seed. This requires however fine-tuning the pre-trained models as the pre-trained models are not trained to identify palm oil seeds and segment the corresponding germinated parts.

Another crucial factor that has contributed to the poor generalisation of the model is the germinated oil palm seed mesh template used for pose estimation. In their experiments in [47], their results have suggested that only a single template for a single category/class is required. That cannot be said for this situation as we have an even limited amount of usable data owing to the flaw segmentation technique employed. Unlike the more regular shape object used in [47], where the category includes cars and planes, the germinated part of the seed can exhibit different geometric shape and length, with the length of the plumule and radicle of the germinated part existing in different relative size due to their differing growing environment. With this, one single template may not be enough to cover all the possible cases of different length. This may require employing germinated oil palm seed mesh template with different length of the plumule and radicle, and then picking the one that best describes the germinated oil palm seed to infer a more accurate pose estimation.

Conclusion and future works

The usage of GANs in performing 3D reconstruction of the germinated palm oil seed can significantly relaxed the supervision and data requirement. For this to be achieved however it also requires a variation of the possible input parameters to allow the generative network to better learn the distribution. Employing priors does reduce the required complexity of the framework because it assumes factors that are too complex as randomised distribution. The process comes down to using a reliable training scheme to perform adversarial training on the generative network, and diversifying data/images to ensure the corresponding latent space mapping to be learned.

In conclusion, GANs can be reliable for 3D reconstruction of the germinated parts of the palm oil seed, as shown in the results that the GAN was able to identify the germinated part of having a supposedly distinct 3D structure, while also preserving the roundness of the seed when observed from top to bottom.

For future works, another attempt on performing 3D reconstruction with the same framework can be done with this time using the germinated oil palm seed template, which was used only for pose estimation, to be used as well as the main mesh template for 3D reconstruction, instead of the UV sphere. This also means a correspondence needs to be established between the UV sphere and the seed mesh template. However, the remeshing step performed in [47] may be able to aid in such process.

Another future works that can be done to improve the initial results would be to fine-tune a pre-trained segmentation model to segment the seed, the germinated part and the seed body. The 2 plausible candidate for this is PointRend [55] and Visual Genone (VG) [60].

References

[1] D. J. Murphy, “Growing palm oil on former farmland cuts deforestation, co2 and biodiversity loss”.

[2] A. D. de Medeiros, R. C. Bernardes, L. J. da Silva, B. A. L. de Freitas, D. C. F. dos S. Dias, and C. B. da Silva, “Deep learning-based approach using X-ray images for classifying Crambe abyssinica seed quality,” *Ind Crops Prod*, vol. 164, 2021, doi: 10.1016/j.indcrop.2021.113378.

[3] S. Javanmardi, S. H. Miraei Ashtiani, F. J. Verbeek, and A. Martynenko, “Computer-vision classification of corn seed varieties using deep convolutional neural network,” *J Stored Prod Res*, vol. 92, 2021, doi: 10.1016/j.jspr.2021.101800.

[4] I. J. Goodfellow *et al.*, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2014. doi: 10.1007/978-3-658-40442-0_9.

[5] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation learning with Deep Convolutional GANs,” *International Conference on Learning Representations*, 2016.

[6] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein generative adversarial networks,” in *34th International Conference on Machine Learning, ICML 2017*, 2017.

[7] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and improving the image quality of stylegan,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. doi: 10.1109/CVPR42600.2020.00813.

[8] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, “Training generative adversarial networks with limited data,” in *Advances in Neural Information Processing Systems*, 2020.

[9] Z. Wu *et al.*, “3D ShapeNets: A deep representation for volumetric shapes,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015. doi: 10.1109/CVPR.2015.7298801.

[10] J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum, “MarrNet: 3D shape reconstruction via 2.5D sketches,” in *Advances in Neural Information Processing Systems*, 2017.

[11] S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik, “Multi-View Supervision for Single-View Reconstruction via Differentiable Ray Consistency,” *IEEE Trans Pattern Anal Mach Intell*, vol. 44, no. 12, 2022, doi: 10.1109/TPAMI.2019.2898859.

[12] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, “Multiview to novel view,” *Eccv*, 2018.

[13] S. Liu, L. Giles, and A. Ororbia, “Learning a hierarchical latent-variable model of 3D shapes,” in *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, 2018. doi: 10.1109/3DV.2018.00068.

[14] P. S. Wang, Y. Liu, Y. X. Guo, C. Y. Sun, and X. Tong, “O-CNN: Octree-based convolutional neural networks for 3D shape analysis,” in *ACM Transactions on Graphics*, 2017. doi: 10.1145/3072959.3073608.

[15] Z. Chen and H. Zhang, “Learning implicit fields for generative shape modeling,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. doi: 10.1109/CVPR.2019.00609.

[16] M. Tatarchenko, S. R. Richter, R. Ranftl, Z. Li, V. Koltun, and T. Brox, “What do single-view 3D reconstruction networks learn?,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. doi: 10.1109/CVPR.2019.00352.

[17] J. Li, K. Xu, and S. Chaudhuri, “GRASS: Generative recursive autoencoders for shape structures,” in *ACM Transactions on Graphics*, 2017. doi: 10.1145/3072959.3073637.

[18] C. Zou, E. Yumer, J. Yang, D. Ceylan, and D. Hoiem, “3D-PRNN: Generating Shape Primitives with Recurrent Neural Networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.103.

[19] A. Johnston, R. Garg, G. Carneiro, I. Reid, and A. Van Den Hengel, “Scaling CNNs for High Resolution Volumetric Reconstruction from a Single Image,” in *Proceedings - 2017 IEEE International Conference on Computer Vision Workshops, ICCVW 2017*, 2017. doi: 10.1109/ICCVW.2017.114.

[20] B. Yang, S. Rosa, A. Markham, N. Trigoni, and H. Wen, “Dense 3D Object Reconstruction from a Single Depth View,” *IEEE Trans Pattern Anal Mach Intell*, vol. 41, no. 12, 2019, doi: 10.1109/TPAMI.2018.2868195.

[21] W. Wang, Q. Huang, S. You, C. Yang, and U. Neumann, “Shape Inpainting Using 3D Generative Adversarial Network and Recurrent Convolutional Networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.252.

[22] A. Dai, C. R. Qi, and M. Nießner, “Shape completion using 3D-encoder-predictor CNNs and shape synthesis,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.693.

[23] X. Han, Z. Li, H. Huang, E. Kalogerakis, and Y. Yu, “High-Resolution Shape Completion Using Deep Neural Networks for Global Structure and Local Geometry Inference,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.19.

[24] Y. P. Cao, Z. N. Liu, Z. F. Kuang, L. Kobbelt, and S. M. Hu, “Learning to reconstruct high-quality 3D shapes with cascaded fully convolutional networks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018. doi: 10.1007/978-3-030-01240-3_38.

[25] W. E. Lorensen and H. E. Cline, “MARCHING CUBES: A HIGH RESOLUTION 3D SURFACE CONSTRUCTION ALGORITHM.,” *Computer Graphics (ACM)*, vol. 21, no. 4, 1987, doi: 10.1145/37402.37422.

[26] Y. Liao, S. Donne, and A. Geiger, “Deep Marching Cubes: Learning Explicit Surface Representations,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/CVPR.2018.00308.

[27] C. Gotsman, X. Gu, and A. Sheffer, “Fundamentals of spherical parameterization for 3D meshes,” in *ACM SIGGRAPH 2003 Papers, SIGGRAPH ’03*, 2003. doi: 10.1145/1201775.882276.

[28] H. Kato, Y. Ushiku, and T. Harada, “Neural 3D Mesh Renderer,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/CVPR.2018.00411.

[29] N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y. G. Jiang, “Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images,” in *Lecture Notes in Computer Science (including subseries*

Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2018. doi: 10.1007/978-3-030-01252-6_4.

[30] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik, "Learning category-specific mesh reconstruction from image collections," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018. doi: 10.1007/978-3-030-01267-0_23.

[31] S. Vicente, J. Carreira, L. Agapito, and J. Batista, "Reconstructing PASCAL VOC," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014. doi: 10.1109/CVPR.2014.13.

[32] S. Tulsiani, A. Kar, J. Carreira, and J. Malik, "Learning Category-Specific Deformable 3D Models for Object Reconstruction," *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 4, 2017, doi: 10.1109/TPAMI.2016.2574713.

[33] J. K. Pontes, C. Kong, S. Sridharan, S. Lucey, A. Eriksson, and C. Fookes, "Image2Mesh: A Learning Framework for Single Image 3D Reconstruction," Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.10669

[34] A. Kurenkov *et al.*, "DeformNet: Free-form deformation network for 3D shape reconstruction from a single image," in *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, 2018. doi: 10.1109/WACV.2018.00099.

[35] D. Jack *et al.*, "Learning Free-Form Deformations for 3D Object Reconstruction," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2019. doi: 10.1007/978-3-030-20890-5_21.

[36] P. Mandikal, K. L. Navaneet, M. Agarwal, and R. Venkatesh Babu, "3D-LMNet: Latent embedding matching for accurate and diverse 3D point cloud reconstruction from a single image," in *British Machine Vision Conference 2018, BMVC 2018*, 2019.

[37] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.264.

[38] K. Li, T. Pham, H. Zhan, and I. Reid, "Efficient dense point cloud object reconstruction using deformation vector fields," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018. doi: 10.1007/978-3-030-01258-8_31.

[39] W. Chen *et al.*, "Dib-R supp.," *ArXiv*, vol. 0, no. 7, 2019.

[40] M. Gadelha, S. Maji, and R. Wang, "3D shape induction from 2D views of multiple objects," in *Proceedings - 2017 International Conference on 3D Vision, 3DV 2017*, 2018. doi: 10.1109/3DV.2017.00053.

[41] M. M. Loper and M. J. Black, "OpenDR: An approximate differentiable renderer," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. doi: 10.1007/978-3-319-10584-0_11.

[42] D. J. Rezende, S. M. Ali Eslami, S. Mohamed, P. Battaglia, M. Jaderberg, and N. Heess, "Unsupervised learning of 3D structure from images," in *Advances in Neural Information Processing Systems*, 2016.

[43] A. Kundu, Y. Li, and J. M. Rehg, “3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018. doi: 10.1109/CVPR.2018.00375.

[44] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee, “Perspective transformer nets: Learning single-view 3d object reconstruction without 3D supervision,” in *Advances in Neural Information Processing Systems*, 2016.

[45] R. Zhu, H. K. Galoogahi, C. Wang, and S. Lucey, “Rethinking Reprojection: Closing the Loop for Pose-Aware Shape Reconstruction from a Single Image,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.16.

[46] J. Zhang, D. Ren, Z. Cai, C. K. Yeo, B. Dai, and C. C. Loy, “Monocular 3D Object Reconstruction with GAN Inversion,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2022. doi: 10.1007/978-3-031-19769-7_39.

[47] D. Pavllo, J. Kohler, T. Hofmann, and A. Lucchi, “Learning Generative Models of Textured 3D Meshes from Real-World Images,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2021. doi: 10.1109/ICCV48922.2021.01362.

[48] L. Goetschalckx, A. Andonian, A. Oliva, and P. Isola, “GANalyze: Toward visual definitions of cognitive image properties,” *J Vis*, vol. 20, no. 11, 2020, doi: 10.1167/jov.20.11.297.

[49] X. Pan, B. Dai, Z. Liu, C. C. Loy, and P. Luo, “Do 2D GANs Know 3D Shape? Unsupervised 3D shape reconstruction from 2D Image GANs,” Nov. 2020, [Online]. Available: http://arxiv.org/abs/2011.00844

[50] J. Zhang *et al.*, “Unsupervised 3D Shape Completion through GaN Inversion,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021. doi: 10.1109/CVPR46437.2021.00181.

[51] D. Pavllo, G. Spinks, T. Hofmann, M. F. Moens, and A. Lucchi, “Convolutional generation of textured 3D meshes,” in *Advances in Neural Information Processing Systems*, 2020.

[52] N. Otsu, “THRESHOLD SELECTION METHOD FROM GRAY-LEVEL HISTOGRAMS.,” *IEEE Trans Syst Man Cybern*, vol. SMC-9, no. 1, 1979, doi: 10.1109/tsmc.1979.4310076.

[53] M. Kass, A. Witkin, and D. Terzopoulos, “Snakes: Active contour models,” *Int J Comput Vis*, vol. 1, no. 4, 1988, doi: 10.1007/BF00133570.

[54] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[55] A. Kirillov, Y. Wu, K. He, and R. Girshick, “Pointrend: Image segmentation as rendering,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2020. doi: 10.1109/CVPR42600.2020.00982.

[56] T. Y. Lin *et al.*, “Microsoft COCO: Common objects in context,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014. doi: 10.1007/978-3-319-10602-1_48.

[57] P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.632.

[58] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, “Least Squares Generative Adversarial Networks,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017. doi: 10.1109/ICCV.2017.304.

[59] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[60] R. Krishna *et al.*, “Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations,” *Int J Comput Vis*, vol. 123, no. 1, 2017, doi: 10.1007/s11263-016-0981-7.

[61] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs trained by a two time-scale update rule converge to a local Nash equilibrium,” in *Advances in Neural Information Processing Systems*, 2017.