CQF6

CQF6 FYP-Implicit Models for Scene Representations

FYP Final Report

Implicit Models for Scene Representations

by

Huang Yiming, Lin Yiqiao, Tang Zichen

CQF6

Advised by

Prof.CHEN Qifeng

Submitted in partial fulfillment of the requirements for COMP 4981

in the

Department of Computer Science

The Hong Kong University of Science and Technology

2022-2023

Date of submission: April 20, 2023

Table of Contents

Acknowledgments

We would like to extend our heartfelt gratitude to everyone who has contributed to the successful completion of this project.

First and foremost, We would like to thank our supervisor, Prof.CHEN Qifeng, for the guidance, feedback, and encouragement throughout the project. His expertise and insights have helped us to navigate the challenges and complexities of the project and have helped us to achieve our objectives.

We would also like to thanks Prof.LI Bo, for being the reviewer from our proposal to the final report.

Additionally, we would like to express our sincerest gratitude to our tutor, Ms.Noor Liza Daveau, for her invaluable help in polishing our project report. Her expertise and guidance helped us to clarify and improve the structure of our report, and the feedback was instrumental in ensuring that our ideas were communicated clearly and effectively. We feel incredibly fortunate to have had the opportunity to work with such a knowledgeable and skilled tutor.

Finally, we would like to express our appreciation to our families and friends, who have provided us with the emotional and moral support that we needed to persevere and succeed.

Once again, we extend our heartfelt thanks to everyone who has contributed to the success of this project.

Abstract

Despite the popularity of Neural Radiance Field (NeRF) [1], the necessity of dense covers prevents it from a wider application. The single-image input challenge remains unsolved since overfitting on a single-view input will result in a collapsed neural radiance field, corrupting the scene's geometry. Inspired by Wu *et al.* [2], we propose two new methods to tackle this problem. 1.*Texture Decoder Network*. 2.*Neural Texture Field*. Both methods can learn textures from the single-view input and generate meshes with textures. While *Texture Decoder Network* is jointly trained with the geometry network and decodes the textures from the image feature, *Neural Texture Field* relies on a pretrained geometry network and learns the textures from a combination of vertices prior and the image features. Particularly, our *Neural Texture Field* pipeline demonstrates a novel implicit-explicit reconstruction approach, combining the strength of the neural field and meshes. Our approaches overcome the issue of collapsed geometry while constructing high-quality texture simultaneously. The implementation of this project is open source and available at: https://github.com/lastbasket/HKUST-CSE-FYP-CQF6.

1 Introduction

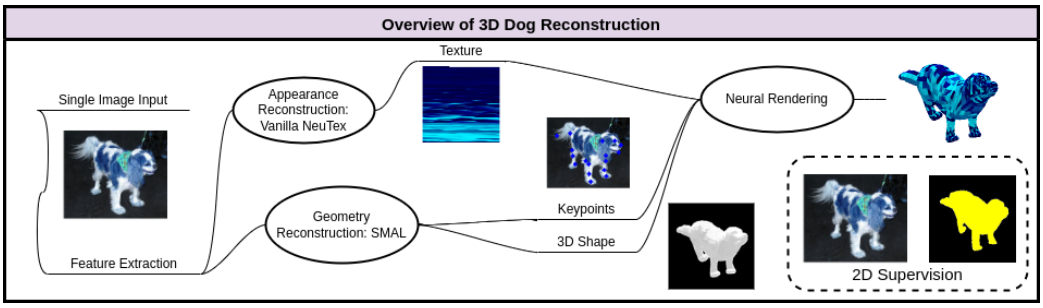


Figure 1: **Overview**. Our method has three components: 1. Geometry reconstruction module using SMAL, 2. Appearance reconstruction module (Vanilla texture network or Vanilla Field) , 3. Neural Renderer using SoftRas.

1.1 Overview

3D Reconstruction

With the growing interest in AR/VR, 3D reconstruction is becoming popular in recent years. Under the existing 3D reconstruction technology, people can transform any object in the real world into a 3D object in the virtual world through 3D scanning [3], multi-angle photography [4]. On the other hand, people can also use various 3D modelling software to create 3D models by themselves and achieve very realistic effects through mature rendering algorithms. Unreal and Unity are two powerful examples of existing virtual game engines, and developers have created several AR/VR interactive platforms through them. However, these measures are unsuitable for the vast system and rapid development. Manual construction methods consume human resources and time and depend on the creators' inspiration and proficiency in tool operation. To overcome the artificial obstacles, computer scientists have developed 3D reconstruction methods based on neural networks to reduce the burden of human labour, including NeRF [1], Neural Voxel Field [5], and other recent popular methods. With the help of neural networks, people can quickly reconstruct 3D objects from multiple or even one photo and translate, rotate, enlarge, shrink, and transform the 3D model. In this report, we reconstruct the

3D dog model using polygon mesh. As shown in Figure 1, we demonstrate our methods for reconstructing the textured 3D dog model using a Deep learning pipeline for predicting the mesh parameters and texture map implicitly and thus reconstructing the 3D dog from the single image input.

Challenge Definition: Algorithm Design
Although the deep learning algorithm can be more convenient for 3D reconstruction, this method still has many defects. The reconstruction of 3D animals remains a considerable challenge. In current traditional & deep learning algorithms, 3D reconstruction of animals encounters the following problems:

- The reconstruction quality is still insufficient, and the current reconstruction model still has general shape confusion and is missing.

- After reconstruction with explicit, we can hardly obtain the 3D contour model with texture and cannot reconstruct the texture and colour information simultaneously.

- The reconstructed model from implicit reconstruction cannot change the pose freely and can only obtain the pose from the input picture.

Challenge Definition: Algorithm Application
In addition to the defects in the algorithm level, the 3D reconstruction algorithms based on artificial intelligence also face many challenges in application. The first one is the lack of robustness caused by the algorithm's quality. In real-world application scenarios, we require algorithms to adapt well to different environments. However, due to the lack of generalization of data sets, artificial intelligence algorithms can only be applied to single scenes similar to the data sets and will significantly reduce their performances in other scenes. Secondly, due to the increasingly complex neural network structure, the production and deployment often exist artificial intelligence algorithms that must spend many computing resources.

To produce a better quality model, engineers must undergo much repetitive training to gain, a process that requires continuously using powerful graphics cards at total capacity. Even some training will continue for a few days and nights. However, the challenge of 3D reconstruction applications requires the practicability and resource cost of the algorithms and demands the mutual demand between the 3D reconstruction and the hardware. Deploying an artificial intelligence algorithm on tiny mobile devices will require the optimization of the algorithm by engineers and also the sufficient computing power support provided by the equipment hardware. However, for most mobile devices such as mobile phones and mini pcs, running neural networks takes up almost all the running resources, and the effect is still very late. Moreover, these difficulties indirectly become obstacles to realising the metaverse [6]. Due to the quality concern of existing 3D reconstruction algorithms, deployment problems, and a large amount of consumption of constructing the metaverse manually, restoring and interacting with animals in the metaverse has become a big problem. Therefore, the main target of this project is how to reconstruct 3D digital creatures in the virtual world, increase quality details, and give biological reconstruction in the virtual world the ability to transform and deform freely. In the following, we will briefly introduce and analyze the two categories of 3D reconstruction algorithms of artificial intelligence so far.

Possible Solution: Explicit Modeling

Explicit scene representations, i.e., triangular meshes, are the main methods to reconstruct heads, in which the morphable models are prior to reconstructing the face from incomplete or noisy data. Those models provide geometry head shapes and facial movements and a statistical linear model for the texture used to reconstruct faces from RGB data. Three representative models that explicitly reconstruct the surface will be introduced below.

3DMM [7] is a classical statistical model of 3D facial shape and texture widely used in facial analysis. 3DMM generalizes two key ideas to re-

construct face, shape, and appearance, respectively; all faces are in dense point-to-point correspondence, separating facial shape and colour and disentangling these from external factors.
FLAME [8] model makes the head model significantly more accurate and expressive by separating the representation of identity, pose, and facial expression. Besides, it also explicitly models head pose and eyeball rotation in contrast with existing models.

Possible Solution: Implicit Modeling
In parallel, implicit models that represent the geometry of subjects using implicit surface functions or volumetric representation are also common. In recent years, neural scene representation networks have been the key to neural rendering and 3D reconstruction. Neural Radiance Fields (NeRF) [1] combined with volumetric rendering allows for novelly synthesising a static object with a sequence of observed images as input. This method can represent complex real-world models and does not suffer from limited resolution. NerFace [9] inherits the benefits of NeRF and models the appearance and dynamics of a human face. Given a monocular portrait video sequence of a person as input, a head avatar is synthesized by conditioning the NeRF on expressions. Later, IMavatar [10] leverages the fine-grained expression control provided by the 3D morphable model and the state-of-the-art implicit head model provided by NerFace to learn personalized, generalized, and 3D-consistent facial avatars from monocular videos.

Proposed Method

According to our research, we must rely on the above two algorithms for better animal reconstruction in the 3D virtual world. Therefore, we refer to the current 3D animal reconstruction algorithm: Coarse-to-fine [11] and choose to improve its prominent architecture for proposing a 3D animal reconstruction method with the advantages of two algorithms and

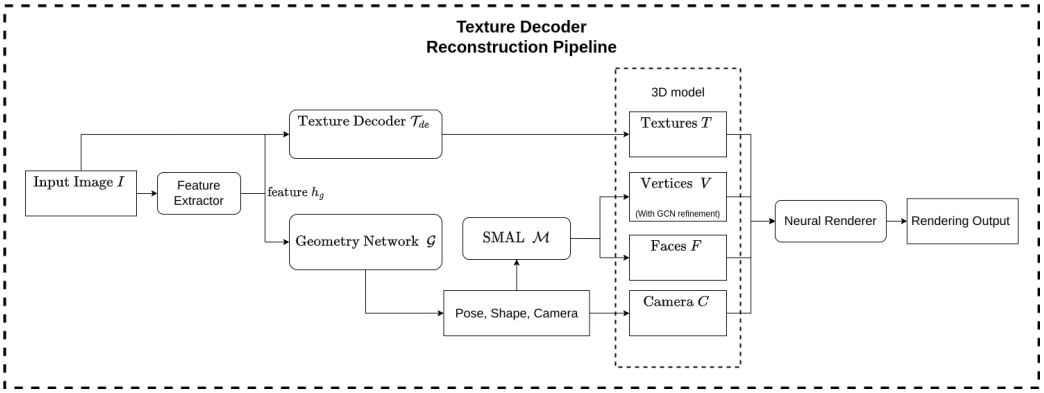


Figure 2: *Texture Decoder* reconstruction pipline.

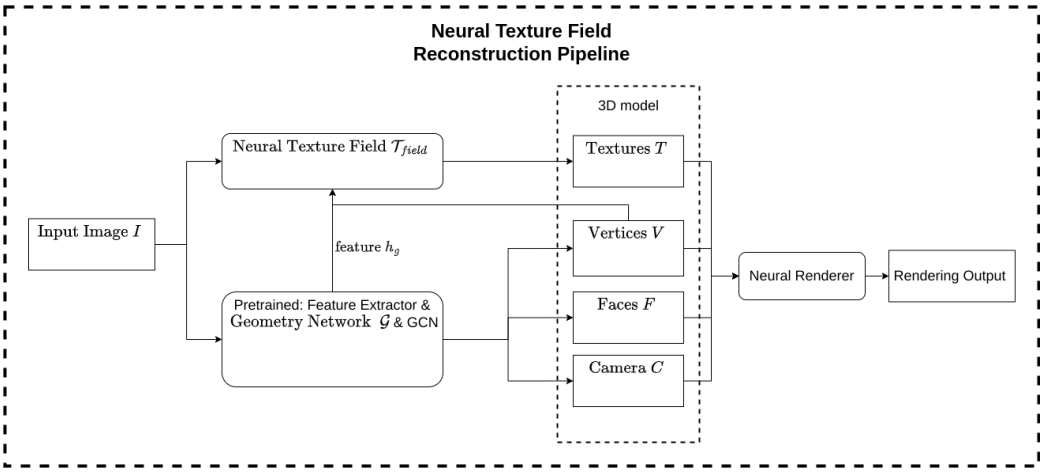


Figure 3: *Neural Texture Field* reconstruction pipline.

is more inclined toward the explicit method. We will take a single animal image as input and predict the vertices, faces and camera pose of the captured animal image through the geometry neural network. Then, we will apply it to a known artificial animal model to regress its 3D shape. Through the change, the standard artificial model will be deformed into the animal shape in the input image to successfully migrate the 3D animal shape from the real world to the virtual world and realize the digital twin between the animal and the metaverse. Different from the multi-view synthase method like NeRF [1], we have an explicit representation as pseudo ground truth geometry, and we can regress the texture from the single image without the distortion of shape. Also, with the explicit reconstructed

geometry representation, we can export the 3D animal for reusing in any Software. We can also control the animal's action input with the explicit representation of the skeleton in the virtual world. It is also possible to input another video of a single animal holding the camera still as a source of action and pose and then extract the information to make the previously reconstructed 3D animal model exhibit the same action and pose.

In this work, we utilize a SOTA geometry reconstruction method Coarse-to-fine [10] and improve the method for single-image animal reconstruction. We propose two pipelines to reconstruct the 3D dog:

1. Texture Decoder Network: adopts joint training of geometry network (explicit) and vanilla texture decoder network (implicit).

2. Neural Texture Field: a pretrained geometry network (explicit) for accurate vertices and faces estimation + vanilla texture field (implicit) for colour extraction.

Figure 4: Reconstruction examples with the *Neural Texture Field* methods. The right sided image with background in black is the reconstructed 3D model, the left sided image is the input.

Contributions

In conclusion, our contributions are as follows:

1. We proposed a novel explicit-implicit 3D reconstruction method with the combination of mesh and neural field.

2. We demonstrate a novel pipeline to resolve the single-view challenge of neural field.

3. We provide a End-to-End method for reconstructing the 3D dog from single image.

4. We provide a method that can convert dog in single image to 3D model that can be used in 3D modeling Software.

1.2 Objectives

This project aims to rebuild real-world animals as virtual creatures in a metaverse and to support remote interaction with people. Our system will apply the neural network with an explicit method for 3D dog reconstruction from dog videos.

Here are the objectives for achieving our goal:

1. **Develop a geometry module** for 3D shape and pose reconstruction of dogs that can fit the template to the dog shape in the video more accurately.

2. **Develop an appearance module** for texture and colour reconstruction of dogs that can create a realistic appearance for the generated shape.

3. **Develop an end-to-end 3D dog reconstruction method** that can support the user in inputting the single image directly and recovering

the 3D animal shape result for demonstration.

The most challenging part we will face is the development of the appearance module. According to the previous method, geometry reconstruction and texture extraction, and mapping tasks can hardly be achieved simultaneously. To address this issue, we will first study the existing methods that propose possible solutions for explicit geometry reconstruction. Then we design our own network for mapping the texture. After selecting from the existing methods, we combine our proposed texture network with the SOTA Coarse-to-fine [11] reconstruction method for our geometry reconstruction. Finally, we will adjust the overall architecture according to the best-fitted method and fine-tune the training hyper-parameters.

1.3 Literature Survey

There is a long history of 3D reconstruction, from 3D scanning to the representation by computer graphics and then the 3D reconstruction by neural networks. In this section, we would like to review the previous works of 3D reconstruction from the perspectives of 3D animal shape, explicit modeling, and implicit modeling.

1.3.1 3D Animal Shape

Previous works for animal shapes include 3D scans and image extraction. Marr *et al.* [12] proposed a method that represents the 3D animal shape by shape primitives from the kinematic tree. Remondino [3] probes the method for retrieving the model by 3D scanning. However, there is still a lack of work focused on 3D scanning since it is difficult to handle the animals. For the animal shape extracted from the 2D images, Cashman *et al.* [13] takes images of a dolphin, a pigeon, and a polar bear to reconstruct the shape models by combining the subdivision surfaces linearly. While the above method suffers from the problems of smoothness. Zuffi *et al.* [14] propose the SMAL model, which registers the learned animal space into a standard template for representing the 3D animal model. They build a similar statistical shape representation as the SMPL [15] model for humans, focusing on four-legged mammals. This method also addresses the problem of registration with the template by introducing the novel part-based model and inference scheme that extends the "stitched puppet" (SP) [16] model. The SMAL model enables a smoother shape with less training data than the previous.

1.3.2 Explicit Modeling

FLAME

The FLAME [8] model adapts the SMPL [15] body model formulation to heads. However, SMPL lacks facial poses and expressions like the articulation of jaws or eyes. The FLAME model extends the SMPL model with a standard vertex-based linear blend skinning (LBS) with expres-

sion blend-shapes (Shape Blendshapes, Pose Blendshapes, and Expression Blendshapes).

By adopting a PCA space for identity shape, simple rotational degrees of freedom, and linear blend skinning for the neck, jaw, and global expression blend-shapes, the FLAME model is shown to be significantly more expressive and realistic compared to existing models. However, there are some drawbacks. The model still lacks the details for high-quality animation. It still cannot support fine-scale details such as wrinkles and pores.

WLDO

WLDO [17] is an end-to-end method for recovering the 3D pose and shape of dogs from monocular internet images. To address the remarkable dissimilarity in shape and texture among dog breeds, a richer prior over shapes than previous work was introduced, which helps regularize parameter estimation. The proposed method learns a detailed 3D prior through expectation maximization (EM) and achieves state-of-the-art performance on the Stanford Dog Dataset. It directly regresses to object pose and shape from a single image without a model fitting stage and uses easily obtained 2D annotations in training and none at test time. The proposed method also incorporates a new multi-modal prior and introduces new degrees of freedom to the SMAL model. The authors released a new parameterized model and annotation dataset called StanfordExtra, the dataset used in our project.

LESSIE

LASSIE [18] is a novel optimization framework for estimating animals' 3D pose and shape from a sparse set of in-the-wild images without using any 2D or 3D ground-truth annotations. The proposed method discovers 3D parts in a self-supervised manner with minimal user intervention and enforces 2D-3D part consistency using self-supervisory deep fea-

tures. Unlike existing methods that rely on pre-defined template shapes, LASSIE does not assume any form of ground-truth annotations, nor does it leverage any multi-view or temporal information. The key insight behind LASSIE is that 3D parts have much simpler shapes compared to the overall animal, and they are robust concerning animal pose articulations. The proposed method achieves considerably better 3D reconstructions, as well as both 2D and 3D part discovery, compared to prior arts, as demonstrated on Pascal-Part and self-collected in-the-wild animal datasets. The part-based representation also allows for various applications such as texture, pose transfer, and animation.

Neural Head Avatars

Based on the FLAME model, the method of Neural Head Avatars [19] optimized a photo-realistic dynamic texture with geometry that adds facial detail and hair structures. Since the FLAME model lacks facial details, hair, and photo-realism, two new MLPs (Multiple layer perceptron) are added to improve this situation. The first MLP G is to model the pose-dependent offsets concerning the template surface, and the second MLP T is to predict the colour value at any surface point of the mesh.

1.3.3 Implicit Modeling

NeRF

Neural Radiance Fields (NeRF) [1] as an implicit method requires only a set of captured images of the single object with known camera poses and achieves state-of-the-art results in view synthesis of complex objects. This approach represents an object using a fully-connected network without any convolution layers (often referred to as a multi-layer Perceptron), whose input is a single continuous 5D coordinate (spatial location (x, y, z) and viewing direction (θ, ϕ)) and whose output is the volume density σ and RGB colour c at that point. To render NeRF from a particular viewpoint, we query 5D coordinates along viewing rays and apply classic volume rending techniques to project the output densities and colour into an image. NeRF directly optimizes the parameters of the representation

network by comparing the ground-truth image and the rendered image. However, NeRF has the following limitations: 1. The original NeRF aims to represent a static scene. 2. A trained NeRF does not generalize to other objects. 3. The original NeRF cannot render a large-scale environment.

NerFace

NerFace [20] inherits the benefits of NeRF and models the appearance and dynamics of a human face. It combines volumetric rendering with Neural Radiance Fields based on the explicit pose and expression parameters estimated from a 3D morphable model. Given a portrait video with a fixed camera and a static background, this method can reproduce the photo-realistic appearance of the subject, including hair and mouth interior.

Since overfitting on a single view will result in a collapsed neural radiance field, we take advantage of the explicit pretrained geometry model and thus prevent the distortion of the geometry.

1.3.4 Implicit and Explicit Modeling

MagicPony

Given a single test image, learning the function that can estimate the 3D shape, articulation, viewpoint, and texture for an articulated animal such as a horse is critical. Dubbed MagicPony [2] as an implicit-explicit method learns this function purely from in-the-world single-view images, with minimal assumptions about the topology of deformation. For the training part, 3D shape models, keypoints, viewpoints, and any other 2D or 3D cue is not required. Only a 2D segmenter for the objects and a description of the topology and symmetry of the 3D skeleton are needed. This method utilizes the advantages of explicit and implicit methods to do 3D shape presentations. The shape of the object is represented implicitly by a neural field but converted on the fly into an explicit mesh via a fast-marching tetrahedra method of [21] and [22], a relative of cubes-like [23]. We implement the SDFs using a Multi-Layer Perceptron (MLP), taking 3D coordinates as input.

2 Methodology

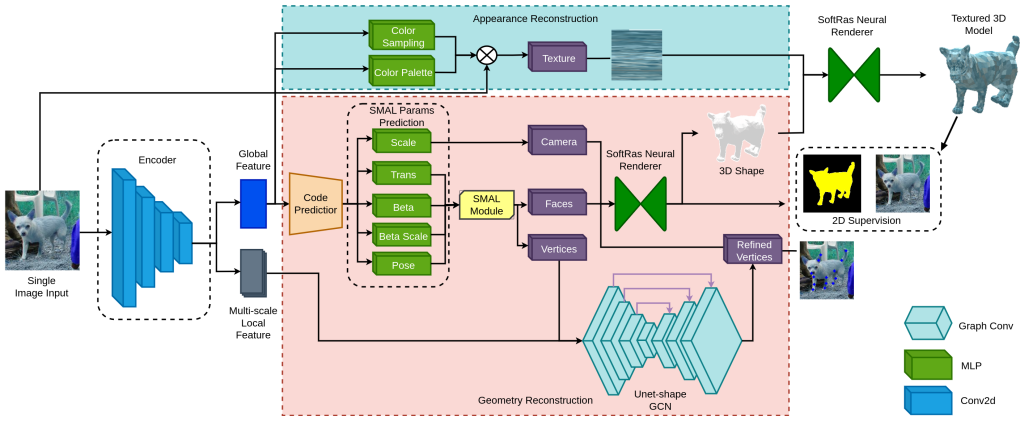


Figure 5: **Architecture**. The architecture of the 3D reconstruction network. We apply a pretrained Resnet as the feature extractor and encode with an encoder network. The Geometry network includes the SMAL decoder and the GCN refinement network. The texture network is built with MLP for predicting the colour palette and colour sampling map.

2.1 Design

Based on the previous work [11] for dog modelling, we propose a novel method that extracts the dog model from a monocular image or video. As Fig 5 shows, the proposed method is implemented in 3 parts which include a network for information extraction, the geometry network for predicting the pose, shape, and camera, and the texture network to reconstruct the corresponding texture.

2.1.1 Feature Extraction

As shown in figure 6, we implement the feature extraction network with a combination of pretrained layers from the ResNet-50 of the official Pytorch library. The extractor consists of a Conv2d layer for global feature extraction with the image first input into the network. Then we use 1 4 layers of the ResNet to extract the multi-scale local features.

The design in Figure 5 of this project starts from the Geometry reconstruction parts, in which we consider the SMAL [14] method and the SOTA 3D

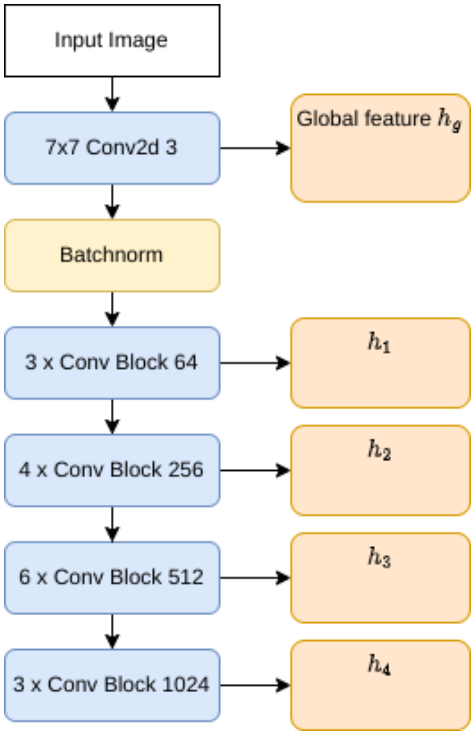


Figure 6: **Feature extractor**.

.

animal reconstruction method [11]. After the implementation and testing of the geometry network, we apply a vanilla self-designed texture generation network to recover the dog texture. Since our method is supervised by the 2D image with colour and silhouette projected from the 3D space, we use a SOTA differentiable neural renderer proposed by Liu *et al.* [24]. Our designed method can reconstruct the 3D dog with the low-res texture and 3D shape from the single image input. For video input, we further utilize a multi-frame optimization algorithm [25] to refine the bone length and shape of the predicted 3D model.

2.1.2 SMAL Animal Model

We adopt the model from Rilegg *et al.* [26] to represent the dog shape in our method. This model is based on SMAL [14]. The animal shape in SMAL consists of two steps: *1. Initial Registration* and *2. Skinned Multi-*

Animal Linear Modeling. Although the animal modeling methodology we adopt is a variant of SMAL, here we will first introduce the SMAL animal modeling measure.

Figure 7: **Template mesh**. The template is segmented into 33 parts.

.

SMAL Function The final function for the SMAL model can be represented as:

$$\mathcal{M}(\beta,\theta,\gamma) = E_{pose}(\theta) + E_s(\beta) + E_{data}(\beta,\theta), \tag{1}$$

where $E_{pose}(\theta)$ and $E_s(\beta)$ are the squared Mahalanobis distances of shape and pose distributions, β are the PCA shape coefficients, γ is the GLoSS pose, and β is the shape of GLoSS model.

2.1.3 Geometry Reconstruction

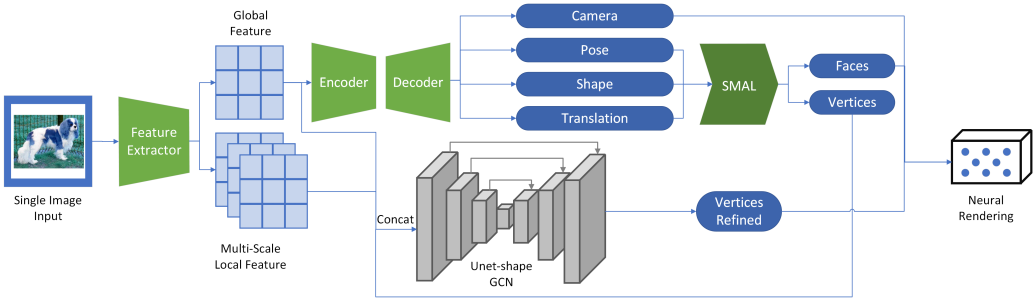


Figure 8: **Geometry reconstruction pipline**. The geometry reconstruction pipeline.

.

We introduce a geometry reconstruction architecture $\mathcal{G}$ based on a shape-branch and a pose-branch network to refine the details of the SMAL model:

$$\mathcal{G} : (I) \rightarrow V, F, C, \tag{2}$$

where I is the input image of the dog, V is the output vertices in the shape of $\mathbb{R}^{n\times 3}$, F is the output faces in the shape of $\mathbb{R}^{m\times 3}$, where n, m are the number of the sampling vertices and faces respectively. C is the predicted camera in shape $\mathbb{R}^3$.

To learn the parameters that the SMAL needed for reconstructing 3D dog shapes, we apply a pretrained ResNet for image feature extraction:

$$h_g = Conv1(I), \tag{3}$$

where h_g is the global feature, $Conv1$ is the 1^{st} convolutional layer in ResNet.

$$h_l = cat(h_1, h_2, h_3, h_4), \tag{4}$$

where h_l is the extracted multi-scale local feature, $cat(\cdot)$ is the concatenation operation, h_1, h_2, h_3, h_4 are the feature with the input h_g to the first $1-4$ layers in the pretrained ResNet.

$$Encoder - Decoder : h_g \rightarrow \beta, \theta, \gamma, \tag{5}$$

here we deploy an Encoder-Decoder network to predict the shape β, pose θ, and translation γ from the global feature h_g.

$$\mathcal{M}(\beta, \theta, \gamma) : V, F, C, \tag{6}$$

Applying the SMAL reconstruction algorithm, we can retrieve the vertices V, faces F, and camera C.

$$3D\ Shape,\ \hat{I}_{Sil} = SoftRas(V_{ref}, F, C), \tag{7}$$

we use the SoftRas renderer [24] and the predicted geometry information to recover the silhouette $\hat{I}_{Sil}$ of the dog in the original image.

Encoder-Decoder Network We utilize the Encoder-Decoder network as

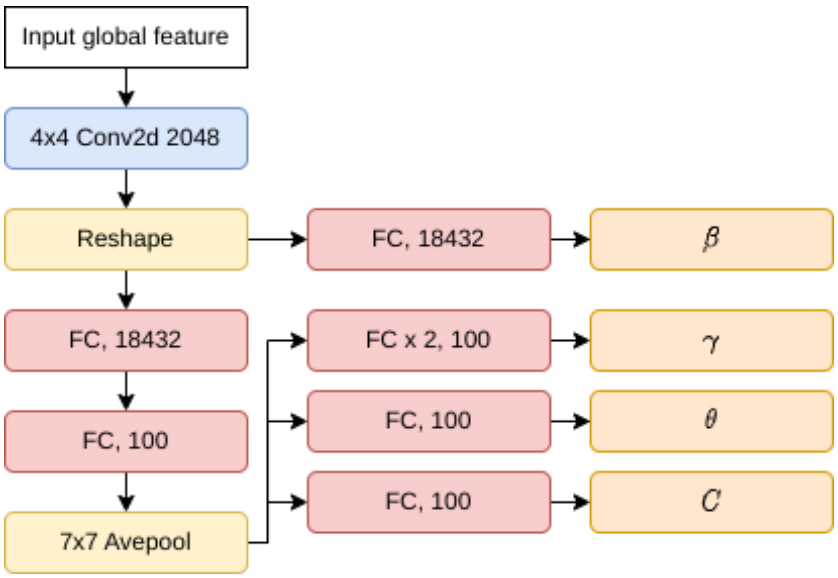


Figure 9: **Encoder-Decoder**.

.

shown in Fig 9 to predict the parameters needed by the reconstruction of SMAL [14] mesh (Shape β, Pose θ, Translation γ, and Camera C). The Encoder-Decoder network takes the extracted global feature as input and outputs the decode parameters with a decoder for each of term. The encoder consists of a convolutional layer for reducing the global feature. The encoded feature is then reshaped and passed through several fully-connected layers to decode the corresponding parameters.

2.1.4 Geometry Refinement

To refine the estimated vertices, we adopt the method from [11] with the local feature from the ResNet feature extractor. Since the vertices and faces are estimated from the global feature of the input image, it lacks the focus of the local feature in the corresponding pixel from the image. Therefore, we apply a UNet-shape graph convolutional neural network for the vertices refinement.

$$GCN : cat(h_g, h_l, V) \rightarrow V_{ref}, \tag{8}$$

we concatenate the global and local features and the vertices as input of the graph neural network where the input node is corresponding to each vertex in the vertices vector and refine the vertices as V_{ref}.

UNet-shape GCN The predicted 2D keypoints (vertices) from the Encoder-

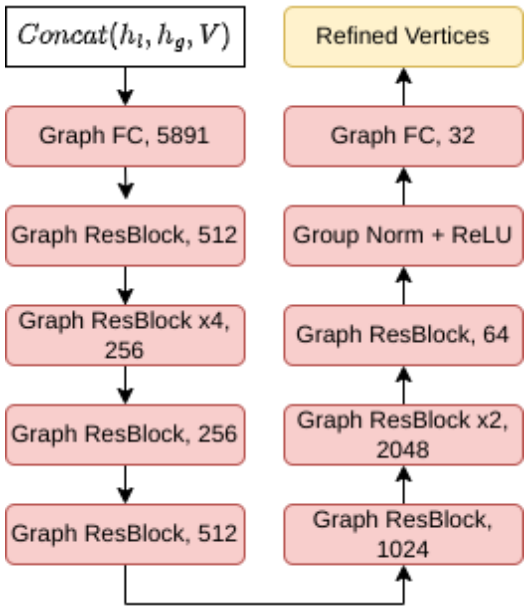


Figure 10: **UNet-shape Graph Convolutional Network**.

.

Decoder network constitute the inputs for estimating the dog's 3D shape, which is modeled in the SMAL algorithm [14]. Due to the insufficient expression power of the Encoder-Decoder Network, we leverage a graph convolutional network in the UNet Shape for refining the vertices. The node in the graph network consists of vertex-wise global and multi-scale local features. We implement the GCN as figure 10 shows. The GCN consists of 4 pairs of encode-decode graph residual blocks (ReLU + Graph linear + ReLU + Graph convolution + ReLU + Graph linear + Graph linear + Skip connection) and two fully connected layers at the beginning and the end.

2.1.5 Appearance Reconstruction

In this section, we will introduce two texture reconstruction methods. The first is a vanilla FCN (fully connected network) for decoding the dog texture directly from the input single image. The second method is a Neural texture field that takes the vertices and image features as input and implicitly generates the colour for each face.

Vanilla Texture Decoder

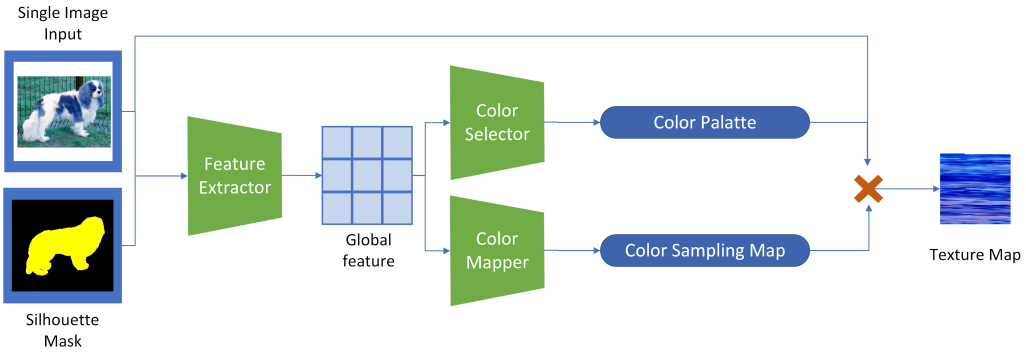


Figure 11: Texture reconstruction pipeline with decoding the texture from the image features extracted from the pretrained ResNet

.

We add a Multi-layer Perceptron (MLP) called Vanilla Texture Decoder

Network $\mathcal{T}_{de}$ inspired by Liu *et al.* [24] to decoder the texture information from the RGB image. The whole pipeline is shown as Fig 11. The estimated texture is fitted to the SMAL model to add fine appearance details and reconstruct the 3D model with colour. The appearance network outputs the predicted colour value as a texture map corresponding to the faces by taking the 3D shape of the faces on the SMAL mesh, the local features from the feature extractor as input:

$$\mathcal{T}_{de} : (I,\ h_g) \rightarrow \mathrm{T}, \tag{9}$$

where I is the input image of $\mathbb{R}^{3\times h\times w}$, where h, w are the height and width of the input image, respectively. h_g is the global feature extracted from the pretrained ResNet $\mathcal{R}$. The output T is the final texture in $\mathbb{R}^{m\times 1\times 3}$.

In the vanilla texture estimation, we first jointly estimate a colour palette I_p and a colour sampling map I_s from the extracted global feature, and then we multiply the sampling map with the original image and then the colour palette to generate the final texture:

$$I_p = MLP_{select}(h_g), \tag{10}$$

where MLP_{select} is the MLP network for estimating the colour palette, I_p is the output colour palette with size $\mathbb{R}^{d\times m}$, where d is the number of colours in the colour palette, for the current vanilla version of texture estimation network, we fix the d as a constant within the range of $[10, 20]$, here in this report we use d as 15.

$$I_s = MLP_{sample}(h_g), \tag{11}$$

where MLP_{sample} is the MLP network for estimating the colour sampling map, I_s is the output sampling map with size $\mathbb{R}^{h^2\times d}$,

$$\mathrm{T} = I \times I_s \times I_p, \tag{12}$$

The Final texture T is calculated by using the multiplication of the sampling map I_s, original image I, and the colour palette I_p.

Texture Decoder Network $\mathcal{T}_{de}$

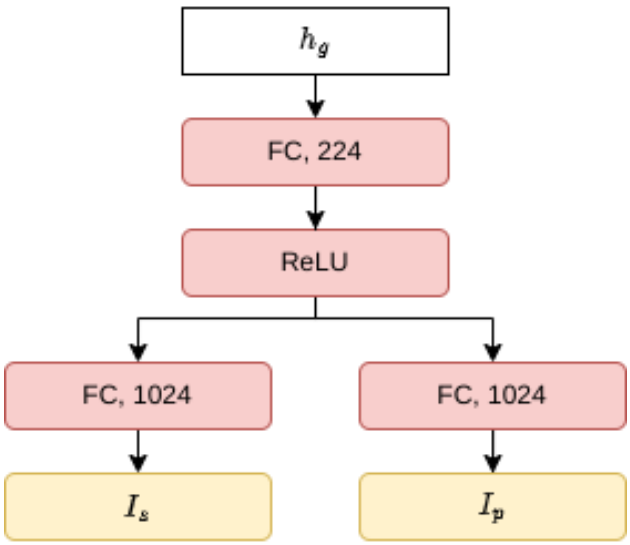


Figure 12: **Texture Decoder Network**

The Texture decoder network $\mathcal{T}_{de}$ predicts the texture of the single image input formulated as the colour value of a point on the mesh template. We have implemented the vanilla version of the texture prediction network by using the fully connected layer for predicting the colour sampling map and the colour palette. Due to the training resources restriction, we use 15 colours for the colour palette, which might be insufficient for recovering the full detail of the original 3D animal model. Our colour sampling map and colour palette are generated from two separated, fully-connected layers, which are then multiplied with the input image masked by the silhouette from the geometry network to reconstruct the texture map.

Vanilla Texture Field

To reconstruct the texture from a single image, inspired by the previous work for implicit modeling [1,27], we propose another method called *Neural Texture Field.* Our neural texture field requires explicit geometry representations from the pretrained network. We consider the geometry estimated from the geometry network as pseudo ground truth for the 3D object shape, and then we regress the texture with the network by overfitting the ground truth 2D view into our neural texture field $\mathcal{T}_{field}$.

The texture reconstruction process of the neural texture field is represented as follows:

$$\mathrm{T} = \mathcal{T}_{field}(I, h_g, V), \tag{13}$$

where I is the input image, h_g is the global feature from the feature extractor, and V is the vertices estimated by the geometry network $\mathcal{G}$. T is the output texture.

The general pipeline of the neural texture is similar to the previous decoder, where we predict the colour palette I_p and the colour sampler I_s. The only difference is that for the input of MLP, we use a combination of vertices and the global feature.

$$input = concat(V, MLP(h_g)), \tag{14}$$

where the $concat()$ is the concatenation operation, MLP is a MLP network that transfer the h_g to the size of $\mathbb{R}^{n\times 1}$, the $input$ will be a 4 channel tensor with shape as $\mathbb{R}^{n\times 4}$.

Since we adopt the vertices and the extracted global feature, our texture

field can generate the texture considering spatial information from the vertices. Thus, the texture field method can generate more detail than the previous decoder-based method.

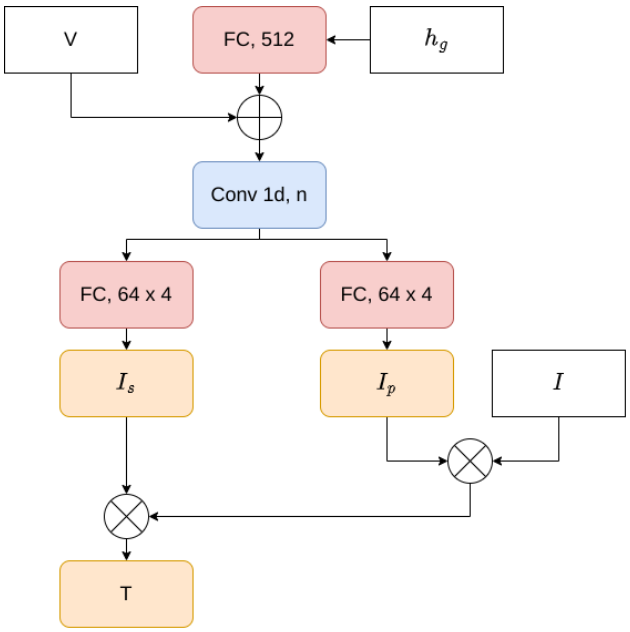


Figure 13: **Neural Texture Field.**

Our implementation for the neural texture field includes three MLP layers and one one-dimensional convolution layer. The input global feature is first upsampled with an MLP to the same size as the number of vertices and then concatenated with the vertices. And then, the combined features will pass through the one-dimension convolution to shrink the hidden size. The output of the convolution layer will be used for generating the colour pallet and the colour sampling embedding using two MLPs as the *Texture Decoder Network* $\mathcal{T}_{de}$, which will be multiplied with the input image I to generate the final texture T.

2.1.6 Losses

2D Keypoint Loss

$$\mathcal{L}_{kp} = \|J_{2D} - \prod(J_{3D}, f)\|^2, \tag{15}$$

where J_{2D} is the ground truth 2D keypoints and $J_{3D} \in \mathbf{R}^{N\times 3} = \mathcal{W} \times V(\mathcal{W}$ is a linear regressor provided with the SMAL model), is the predicted 3D keypoints. f is the camera intrinsic parameters and $\prod$ denotes the projection from 3D to 2D.

Silhouette Loss

$$\mathcal{L}_{Sil} = 1 - \mathcal{T}(S, \mathcal{R}(V, f), \alpha, \gamma), \tag{16}$$

where $\mathcal{T}$ is the Tversky loss S represents the ground truth silhouette and $\mathcal{R}$ is the mesh renderer. α and γ are two hyper-parameters.

Tversky Loss

$$\mathcal{T}(S, \mathcal{R}(V, f), \alpha, \beta) = \frac{|PG|}{|PG| + \alpha|P - G| + \beta|G - P|}, \tag{17}$$

where $|PG|$ is the foreground pixel in both predicted mask P and ground truth mask G, $|P - G|$ is the background pixel predicted as foreground and $G - P$ vise-versa.

Shape and Pose Losses

$$\mathcal{L}_{\beta} = (\beta - \mu_{\beta})^T \Sigma_{\beta}^{-1} (\beta - \mu_{\beta}), \tag{18}$$

$$\mathcal{L}_{\theta} = (\theta - \mu_{\theta})^T \Sigma_{\theta}^{-1} (\theta - \mu_{\theta}), \tag{19}$$

where $\mathcal{L}_{\theta}$, and $\mathcal{L}_{\beta}$ are the pose and shape loss correspondingly. μ_{β} and μ_{θ} are the mean for the shape and pose prior. Σ_{β}^{-1} and Σ_{θ}^{-1} are the variance of the shape and pose prior.

Colour Loss

$$\mathcal{L}_{colour} = \|\hat{I} - I\|_1, \tag{20}$$

where the $\hat{I}$ and I are the prediction and the ground truth images respectively.

We also use the pose limit prior loss $\mathcal{L}_{lim}$ from the previous work [28].

Total Loss

Our final loss can be represented as:

$$\mathcal{L} = \lambda_{colour}\mathcal{L}_{colour} + \lambda_{lim}\mathcal{L}_{lim} + \lambda_{\theta}\mathcal{L}_{\theta} + \lambda_{\beta}\mathcal{L}_{\beta} + \lambda_{kp}\mathcal{L}_{kp} + \lambda_{sil}\mathcal{L}_{sil}, \tag{21}$$

where all the λ terms here represent the weight for each loss term.

2.2 Implementation

2.2.1 Training configuration

To train the whole pipeline, we set the learning rate for the geometry baseline(Encoder-Decoder network, GCN) and the vanilla texture network as 0.0001, batch size 32, and use Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$ as the optimizer. We use a cosine decay schedule [29] to control our learning rate. We trained the pipeline with two strategies for our two proposed methods on the StanfordExtra dataset [30], where we use the 2D segmentation, 2D RGB image, and the keypoints as our ground truth for supervising the network training.

2.2.2 Training Process

Our training processes for the *Texture Decoder Network* and *Neural Texture Field* are shown below:

For the training pipeline of the Texture decoder method, we have three stages. In stage one, we train the texture network together with the feature extractor and the geometry network for 200 epochs. In stage 2, we only update the GCN for 10 epochs for initialization. In stage 3, we update the whole pipeline together for 200 epochs.

For the training process of the Neural Texture Field, we use a pretrained geometry network, feature extractor and GCN for geometry estimation, and train as the 3 stages in the training pipeline of the Texture decoder method but without the texture decoder. In stage 4, we overfit the neural texture field with 60 epochs.

Training Log

2.2.3 Validation

We summarized the overall performance of Encoder-Decoder method (both before and after refinement) in stage 1,2,and 3 in the figures below. Note

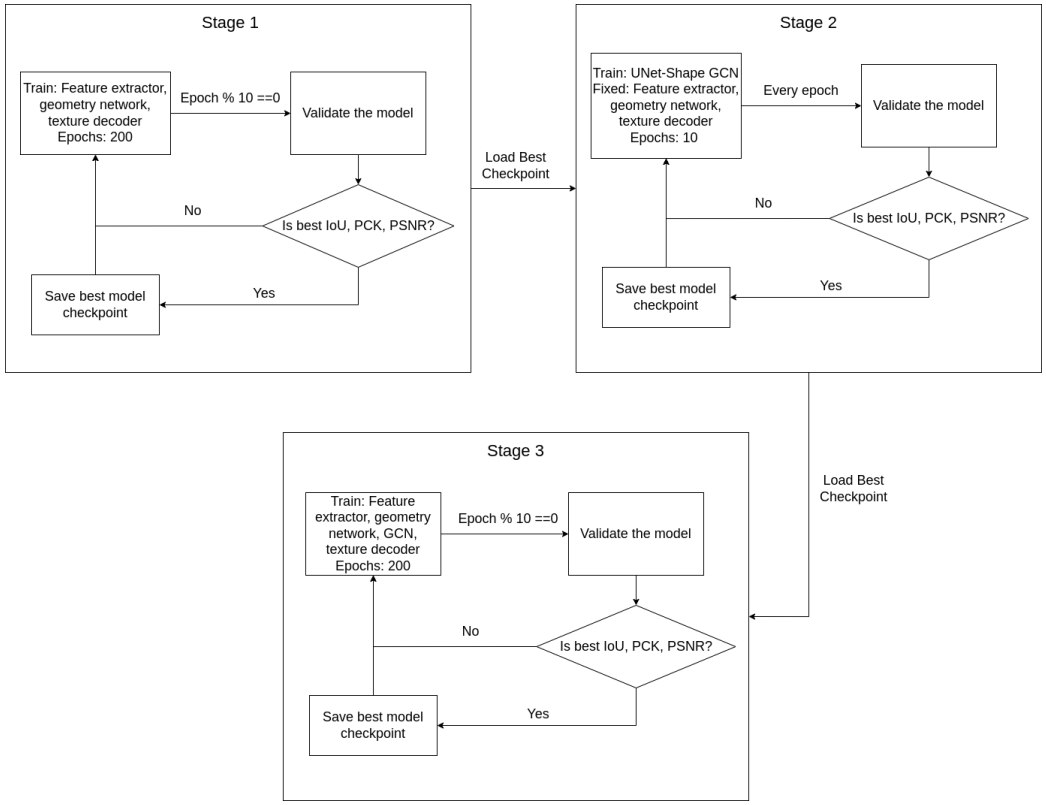


Figure 14: Training process of Texture decoder pipeline.

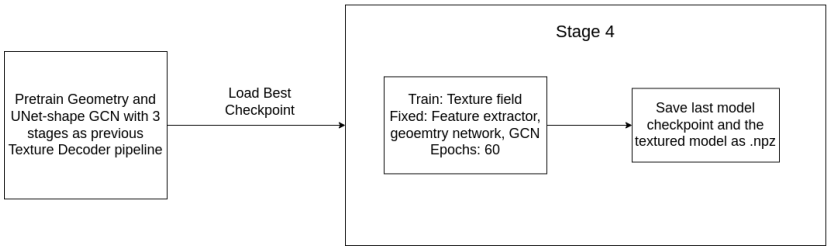


Figure 15: Training process of Neural Texture Field pipeline.

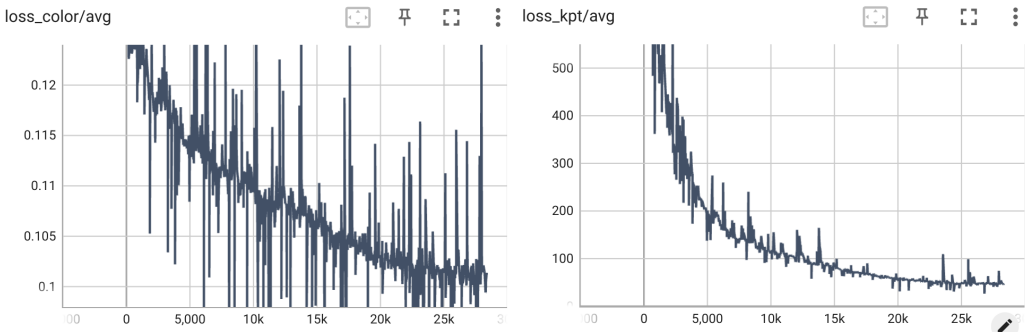


Figure 16: Left: loss_color, right: loss_kpt in stage 1

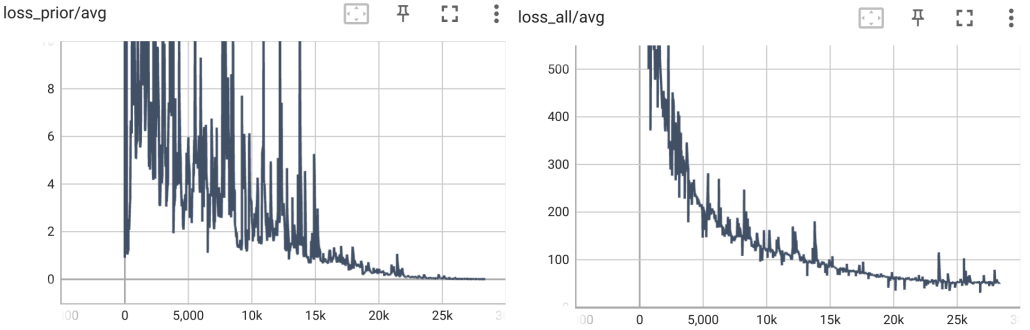


Figure 17: Left: loss_prior, right: loss_all in stage 1

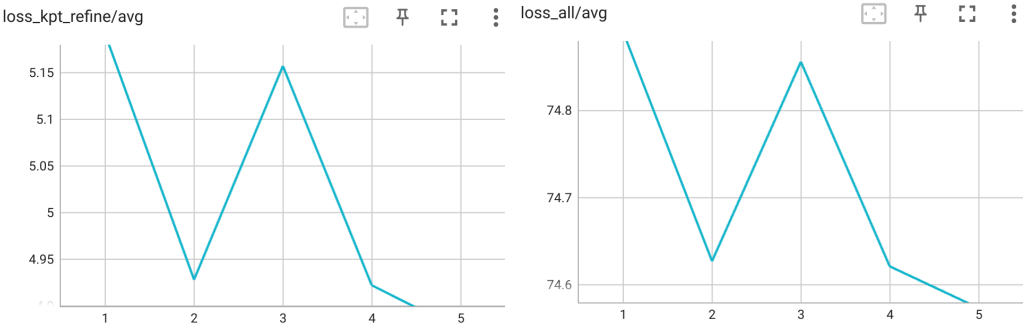


Figure 18: Left: loss_kpt_refine, right: loss_all in stage 2

that we use the performance of Coarse-to-fine as ground truth for comparison. One may find out that the accuracy of Encoder-Decoder for Geometry reconstruction becomes worse than Coarse-to-fine due to the addition of Vanilla Texture Network. Therefore, we need to consider an alternative solution, which is Vanilla Texture Field.

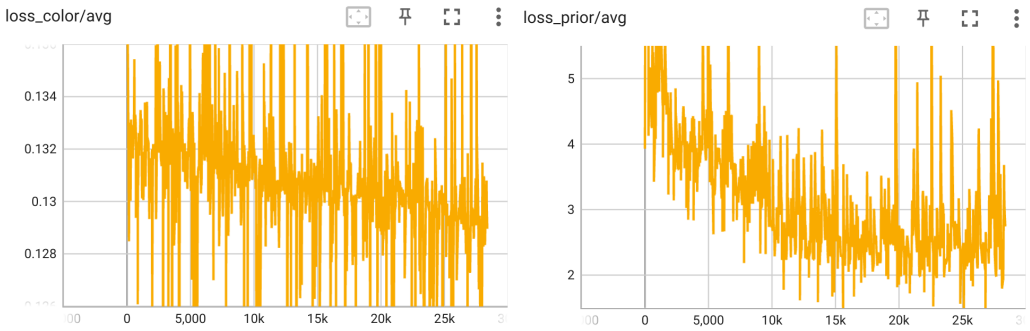


Figure 19: Left: loss_color, right: loss_prior in stage 3

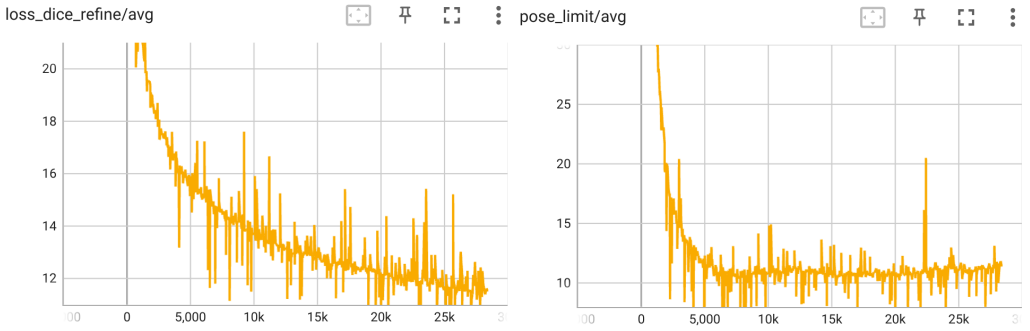


Figure 20: Left: loss_dice_refine, right: pose_limit in stage 3

2.3 Testing

The two vital and testable components of our project are the geometry model and the appearance model. We can test each of them separately using 2D supervision. Here in this section, we will introduce the detail of the dataset and the testing results for both models.

2.3.1 Datasets

StanfordExtra Dog Dataset

We trained and tested both models with the StanfordExtra Dog Dataset (StanExt dataset) [30]. The StanExt dataset contains 20,580 images, where each image has 20 keypoints and segmentation masks. These 20 keypoints are 2 per tail, 3 per leg, 2 per ear, nose, and jaw. We further modify the keypoints by adding the keypoints for withers, throat, and eyes. These extra keypoints are obtained by the pre-trained stacked hourglass network

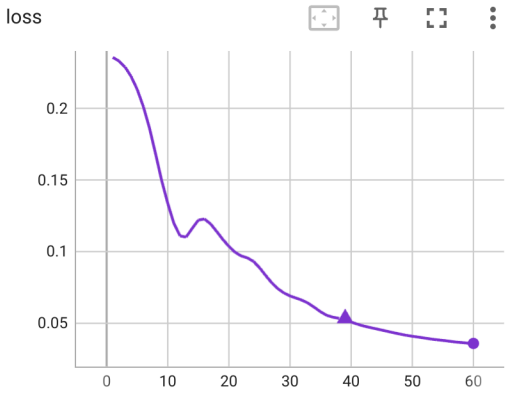


Figure 21: Loss_all of Vanilla Texture Field

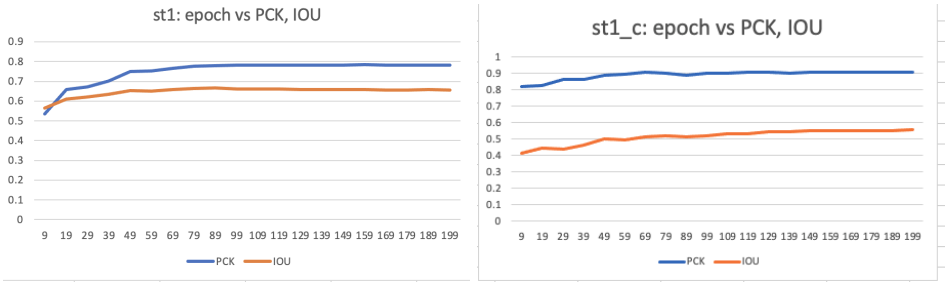


Figure 22: Left: Ground truth. Geometry reconstruction performance only in stage 1. Right: Encoder-Decoder Geometry performance in stage 1.

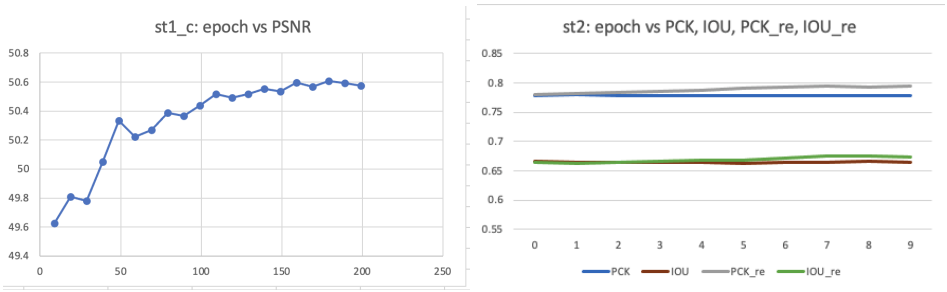


Figure 23: Left: our Encoder-Decoder Texture performance in stage 1. Right: Ground turth. Geometry reconstruction performance before and after refinement in stage 2. PCK_re and IOU_re represent performance after refinement.

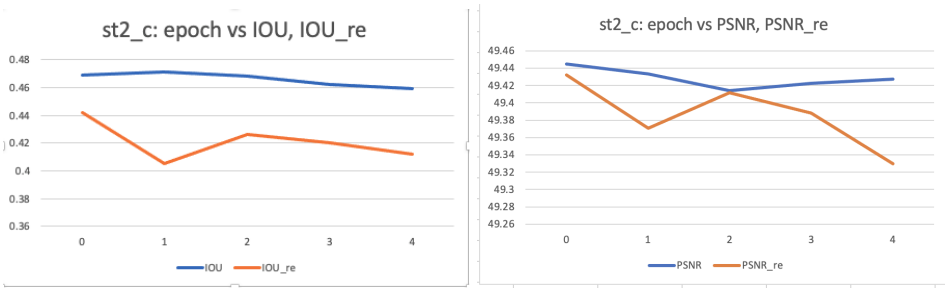


Figure 24: Left: our Encoder-Decoder Geometry performance before and after refinement in stage 2. IOU_re represent performance after refinement. Right: Encoder-Decoder Texture performance before and after refinement in stage 2.

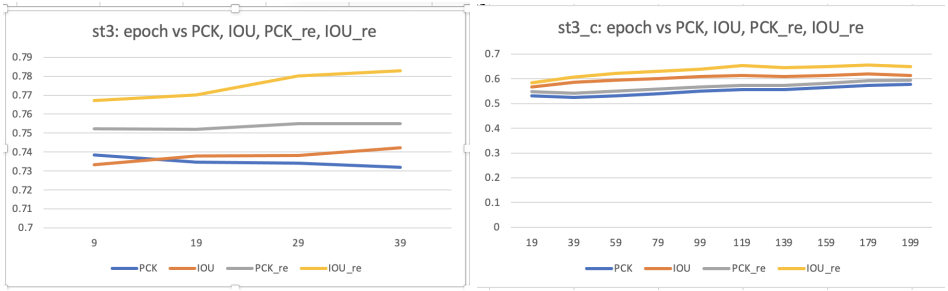


Figure 25: Left: Ground truth. Geometry reconstruction performance only before and after refinement in stage 3. Right: Encoder-Decoder Geometry performance before and after refinement in stage 3.

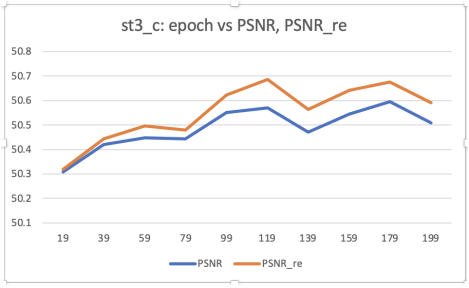


Figure 26: our Encoder-Decoder Texture performance before and after refinement in stage 3.

on the Animal Pose dataset [31].

2.3.2 Testing the Geometry model

We evaluated the accuracy of the Geometry model with standard 2D measures like PCK and IoU.

1. **Intersection over Union (IoU)**
   Here we will evaluate the 2D projection of the animal shape by IoU of the predicted and ground truth silhouettes. Let S^{pred} and S^{gt} be the predicted and ground truth silhouettes, we compute the IoU as:

$$IoU = \frac{S^{pred} \cap S^{gt}}{S^{pred} \cup S^{gt}}. \tag{22}$$

2. **Percentage of Correct Keypoints (PCK)**
   We also adopt PCK for measuring the correctness of the predicted keypoints:

$$PCK^D = \frac{\sum_D \delta(d_{pi}/d_p^{def} \leq T_D)}{\sum_D 1}, \tag{23}$$

   where D represent the set of keypoints of the dog, i is the $i^{th} keypoint$, δ is the indicator function, d_{pi} is the Euclidean distance between the predicted keypoints and ground truth, d_p^{def} is the scale factor of the animal shape, T_D is the threshold.

We summarized the geometry reconstruction performance of BARC [26], Coarse-to-Fine [11] and WLDO [30] in table 1, and concluded that Coarse-to-fine [11] performed the best as our Geometry model among the three.

	BARC	Coarse-to-Fine	WLDO
IoU	0.7232	0.81637	0.74195
PCK	0.7894	0.83405	0.78768
PCK_legs	0.7989	0.81873	0.76353
PCK_tail	0.5346	0.63616	0.63895
PCK_ears	0.7785	0.84435	0.78061
PCK_face	0.8717	0.94415	0.92110

Table 1: Comparison of accuracy between BARC, Coarse-to-Fine and WLDO

We also conducted a qualitative comparison of BARC (upper half) with Coarse-to-Fine (lower half). For each method, we show the input image, the keypoint labelled, the 3D reconstruction projected on the input image, and the 3D reconstruction (from left to right).

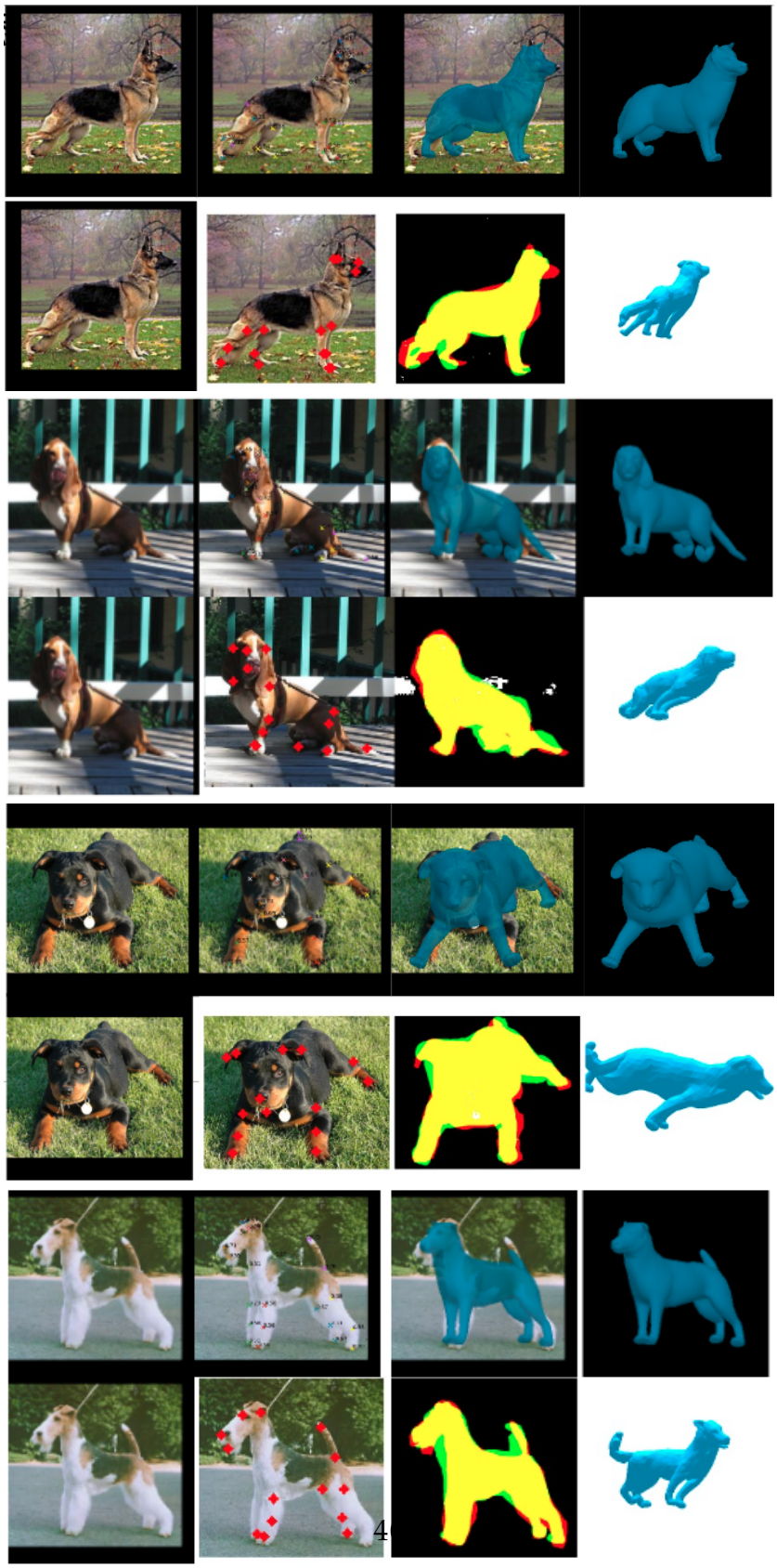


Figure 27: Sample testing results of BARC and Coarse-to-Fine(Our geometry baseline). From left to right: **input image, the keypoint labelled, segmentation map, 3D shape without textures.**

2.3.3 Geometry Model Evaluation

As stated in 2.3.2, we compared three well-established Geometry models (BARC, Coarse-to-Fine, WLDO) and came up with the conclusion that Coarse-to-Fine is the state-of-the-art while the IoU and PCK of such method can achieve 0.82 and 0.84 respectively. Therefore, we adopt Coarse-to-fine as the backbone method in our system and adapt it on this basis.

2.3.4 Testing the Texture model

We tested our two texture models, Vanilla Texture Decoder (top) and Vanilla Texture Field (bottom), using a subset of the StanfordExtra Dog Dataset [30]. Below is a set of test cases. For each row, from left to right are the original image, textured shape, keypoints, segmentation map, refined textured shape, refined keypoints, refined segmentation map, novel view. Other testing cases refer to Gallery 10.

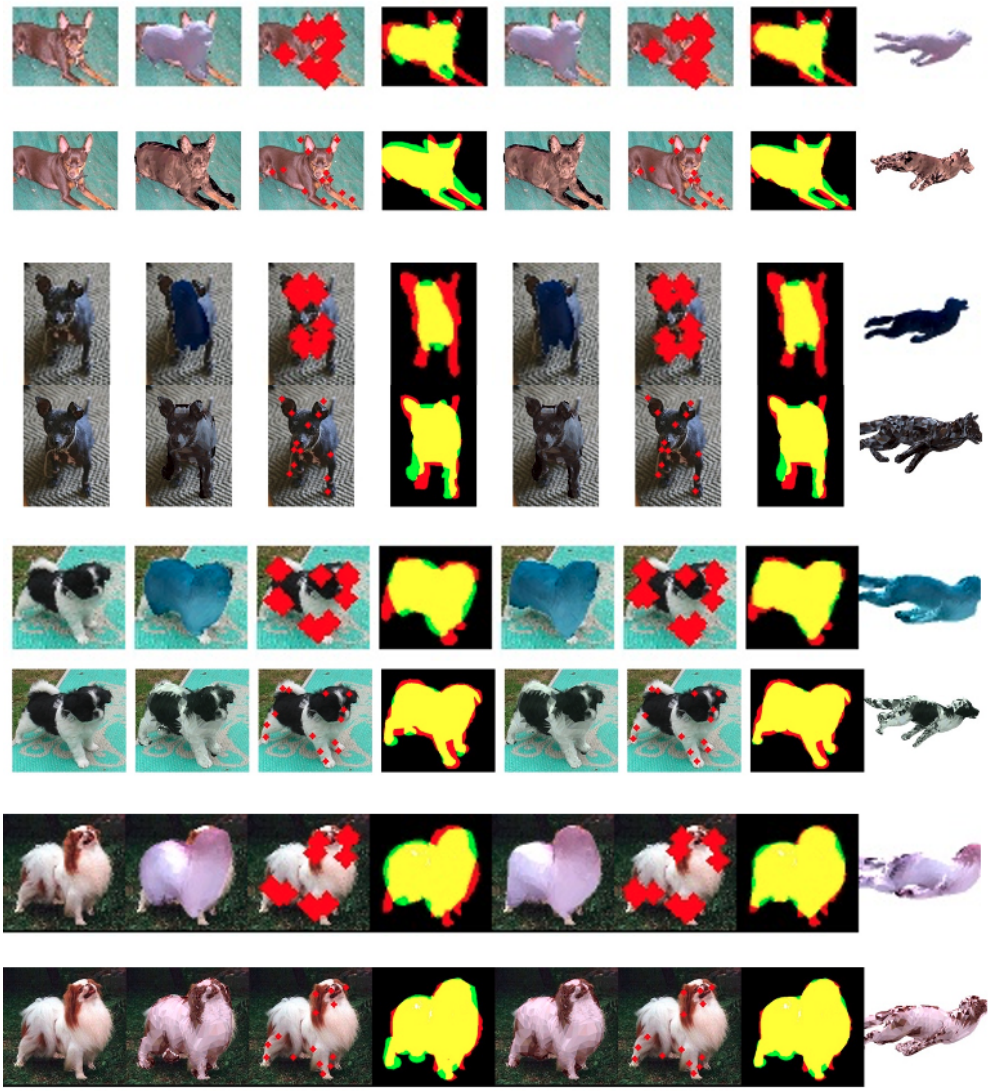


Figure 28: Sample testing results of Vanilla Texture model. From left to right are: **input, textured shape, keypoints, segmentation map, refined textured shape, refined keypoints, refined segmentation map, novel view.**

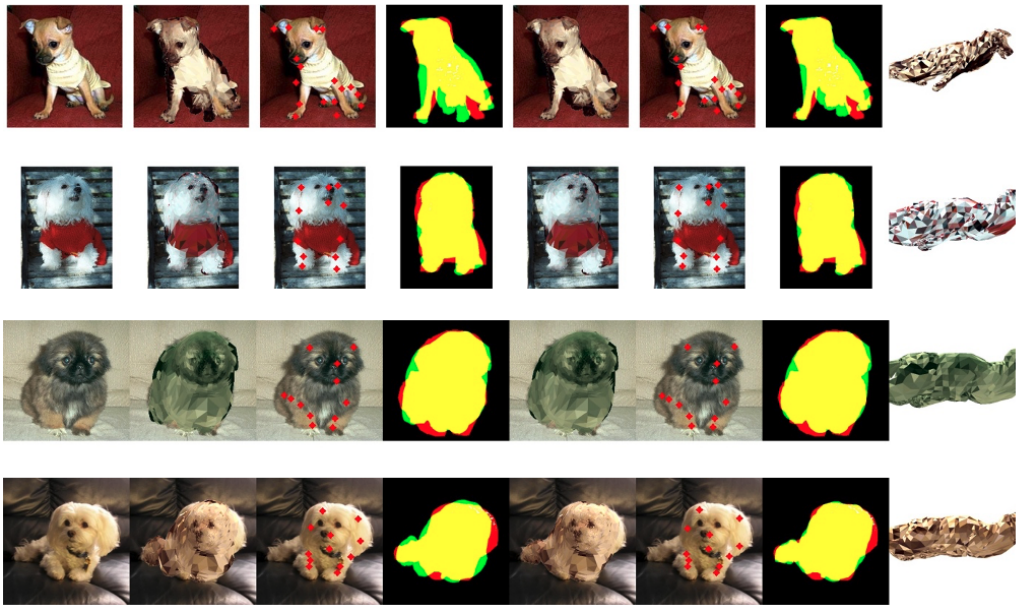


Figure 29: Failure Cases for Vanilla Field. From left to right are: **input, textured shape, keypoints, segmentation map, refined textured shape, refined keypoints, refined segmentation map, novel view.**

2.3.5 Texture Model Evaluation

From our testing results in section 2.3.4, one can observe that the performance of the Vanilla Texture Field is generally better than the Vanilla Texture Decoder. By comparing the texture before the refinement of the Vanilla Texture Field with the original image, we are surprised to find out that the texture on the front side is already pretty good and the heatmap loss is small. Although in some cases, such as poor lighting, images with filters, and distracting accessories, the colour extracted can be slightly different from the ground truth, see Figure 30. On the other hand, the texture information on the sides is not sufficient due to the limitation of a single front image input, thus leading to some inconsistency throughout the final model. However, we are satisfied with the final results because the two Vanilla models are not too complex and the computation time is acceptable. In the future, one can also consider trying several image inputs to address this problem.

3 Interaction Interface & 3D Model Application

As mentioned in the previous design, our method enables end-to-end 3D reconstruction from the single image input. Compared with the implicit-based method, our explicit-implicit can conveniently output the 3D meshes with colour and save them as *.obj* or *.pkl* files which can be imported to any 3D Software for 3D games and 3D CG development such as *Unity 3D Max*, providing enormous potential in the movie, game design, and AR/VR industries. For example, in movie production, the ability to quickly and accurately create realistic 3D models of animals can significantly reduce production costs and time. Similarly, in game design, 3D models of dogs can be created in real-time, enhancing the gaming experience for players. In the AR/VR industry, using 3D models of dogs can provide users with an immersive and interactive experience.

Interaction Interface

To visualize and modify the 3D model from the estimation and adjust the detail according to the user's favour, we also provide software based on the visualization tools for SMAL [14]. Here are the demonstrations of visualizing and modifying the 3D model from our deep learning method:

Our visualizing tool will also be available at https://github.com/lastbasket/HKUST-CSE-FYP-CQF6.

Figure 30: Visualization tool for visualizing the 3D model and adjusting the pose and shape detail of the 3D model.

4 Discussion

4.1 On the Promising Performance of the Geometry Reconstruction Network

When trained on the BARC [26] for the first time, we were surprised to find that the final results were not as satisfying as stated in the paper. There were even some extreme errors such as inverse dog (recognizing the head as the back) in our testing cases. Therefore, we started to look for other alternative methods like Coarse-to-fine [11]. After multiple rounds of testing, we finally confirm ourselves that Coarse-to-fine perform has the highest accuracy in 2D measures like PCK and IoU and should be the backbone algorithm for our system.

4.2 On the Fair Performance of the Texture Reconstruction Network

This should be the first time that 3D texture reconstruction comes to dogs and we are excited that we finally make it. We have two pipelines for Texture reconstruction, Encoder-Decoder and Vanilla Field. Encoder-Decoder has a relatively simple architecture, and in fact gives an acceptable output given that the input is only a single image. But we are not satisfied with the accuracy and losses, and that is why we also brings in Vanilla Field, a more powerful texture reconstruction model. As we can see from the test results, Vanilla Field is able to restore the basic color, but sometimes it may also mix with some other unwanted colors. After some experiments, we believe that this result is due to two possible reasons:

1. Given that the input is only a frontal image, the deduced side texture information is relatively limited. For invisible area, Vanilla field may mix into inaccurate colors, leading to a less satisfactory 3d model.

2. We also notice that the ground truth segmentation map in the StandfordExtra dataset is not accurate enough, which means it will also

include some background color. Since our Vanilla Field is implicit, relying on the output from Geometry network, it is inevitable that the final model will mix in some background color.

4.3 On the Development of Interaction Interface and Applications

During testing, our software is able to output 3D meshes with color in 0.07 seconds, including upload time. The interation interface is also user-friendly, as the users can easily stretch the sliders and modify the 3D model in real-time.

5 Conclusions

5.1 Summary

In this project, we demonstrate how to combine the advantage of the existing explicit geometry modeling methods and the implicit texture extraction methods. Here we demonstrate two explicit-implicit methods for reconstructing 3D dog from single image input. While the *Texture Decoder* method does not perform well on the texture extraction. We get inspiration from the previous work [2] and proposed the *Neural Texture Field*, where we use a similar geometry representation to explicitly generate robust meshed and use the vertices and image feature as latent code to extracting the fine texture. We overcome the single-view challenge of corrupt geometry with the explicit geometry representation and successfully build an End-to-End framework for single-view 3D reconstruction.

However, there still exist disadvantage for our methods. Since the colours of texture map in our methods are supervised by single-view, there exists more distortion and artifacts on the back side of the texture. This suggest that our method is poor in the inpainting ability of the unseen area. Furthermore, due to the inaccuracy of the ground truth segmentation map,

there exist some error in the colour pallet sampled by our methods, especially in the *Texture Decoder* method. At the current stage, the quality of our reconstructed model is still far from the multi-view reconstruction methods.

5.2 Future Works

In the future, to solve the inpainting ability problem, we may try to utilize some generative model such as GAN [32], VAE [33], and Diffusion [34] models. We may also try to develop some self-supervise strategies to increase the generative ability of our methods. We hope that our methods can be further extend and thus provide a more realistic result that can be widely used in other industries.

6 Project Planning

6.1 Tasks

- Do the literature survey
- Decide the FYP topic
- Improve existing models
- Write the FYP proposals
- Model training and testing
- Result evaluation
- Write the monthly report
- Write the progress report
- Write the final report
- Prepare the oral presentation and demo

6.2 Updated GANTT Chart

Task	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May
Decide the FYP topic										
Write the FYP proposal										
Do the literature survey										
Improve existing models										
Model training and testing										
Result evaluation										
Write the monthly report										
Write the progress report										
Write the final report										
Prepare the presentation										

7 Required Hardware & Software

7.1 Hardware

Development PC:	Personal PC with MS Windows 10 / MacOS.
Minimum Display Resolution:	1920 * 1080 with 16-bit color.
Server:	CSE servers with CentOS Linux 7.
Graphic Cards:	RTX 2080Ti x 16 RTX 3090 x 8

7.2 Software

Pytorch	Framework for implementing the neural network.
Pytorch3D	For processing 3D learning data.
OpenCV	For normal image processing.
...	

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[2] S. Wu, R. Li, T. Jakab, C. Rupprecht, and A. Vedaldi, "MagicPony: Learning articulated 3d animals in the wild," 2023.

[3] F. Remondino, "Heritage recording and 3d modeling with photogrammetry and 3d scanning," *Remote sensing*, vol. 3, no. 6, pp. 1104–1138, 2011.

[4] H. Chen and J. Zheng, "Dynamics of photography guiding based on multi-angle virtual image reconstruction technology," in *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1232–1235, IEEE, 2022.

[5] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, and C. Theobalt, "Neural sparse voxel fields," *Advances in Neural Information Processing Systems*, vol. 33, pp. 15651–15663, 2020.

[6] J. Joshua, "Information Bodies: Computational Anxiety in Neal Stephenson's Snow Crash," *Interdisciplinary Literary Studies*, vol. 19, pp. 17–47, 03 2017.

[7] V. Blanz and T. Vetter, "A morphable model for the synthesis of 3d faces," in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 187–194, 1999.

[8] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero, "Learning a model of facial shape and expression from 4D scans," *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, vol. 36, no. 6, pp. 194:1–194:17,

2017.

[9] G. Gafni, J. Thies, M. Zollhofer, and M. Nießner, "Dynamic neural radiance fields for monocular 4d facial avatar reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8649–8658, 2021.

[10] Y. Zheng, V. F. Abrevaya, M. C. Bühler, X. Chen, M. J. Black, and O. Hilliges, "Im avatar: Implicit morphable head avatars from videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13545–13555, 2022.

[11] C. Li and G. H. Lee, "Coarse-to-fine animal pose and shape estimation," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[12] D. Marr and H. K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 200, no. 1140, pp. 269–294, 1978.

[13] T. J. Cashman and A. W. Fitzgibbon, "What shape are dolphins? building 3d morphable models from 2d images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 232–244, 2013.

[14] S. Zuffi, A. Kanazawa, D. W. Jacobs, and M. J. Black, "3d menagerie: Modeling the 3d shape and pose of animals," *CoRR*, vol. abs/1611.07700, 2016.

[15] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM Trans. Graph.*, vol. 34, oct 2015.

[16] S. Zuffi and M. J. Black, "The stitched puppet: A graphical model of 3D human shape and pose," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2015)*, pp. 3537–3546, June 2015.

[17] B. Biggs, O. Boyne, J. Charles, A. Fitzgibbon, and R. Cipolla, "Who left the dogs out?: 3D animal reconstruction with expectation maximization in the loop," in *ECCV*, 2020.

[18] C.-H. Yao, W.-C. Hung, Y. Li, M. Rubinstein, M.-H. Yang, and V. Jampani, "Lassie: Learning articulated shape from sparse image ensemble via 3d part discovery," in *NeurIPS*, 2022.

[19] P.-W. Grassal, M. Prinzler, T. Leistner, C. Rother, M. Nießner, and J. Thies, "Neural head avatars from monocular rgb videos," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18653–18664, 2022.

[20] G. Gafni, J. Thies, M. Zollhöfer, and M. Nießner, "Dynamic neural radiance fields for monocular 4d facial avatar reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8649–8658, June 2021.

[21] T. Shen, J. Gao, K. Yin, M.-Y. Liu, and S. Fidler, "Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[22] J. Munkberg, J. Hasselgren, T. Shen, J. Gao, W. Chen, A. Evans, T. Müller, and S. Fidler, "Extracting Triangular 3D Models, Materials, and Lighting From Images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8280–8290, June 2022.

[23] Y. Liao, S. Donné, and A. Geiger, "Deep marching cubes: Learning ex-

plicit surface representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[24] S. Liu, T. Li, W. Chen, and H. Li, "Soft rasterizer: A differentiable renderer for image-based 3d reasoning," *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[25] B. Biggs, T. Roddick, A. Fitzgibbon, and R. Cipolla, "Creatures great and SMAL: Recovering the shape and motion of animals from video," in *ACCV*, 2018.

[26] N. Rueegg, S. Zuffi, K. Schindler, and M. J. Black, "BARC: Learning to regress 3D dog shape from images by exploiting breed information," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 3876–3884, 2022.

[27] D. Xu, Y. Jiang, P. Wang, Z. Fan, H. Shi, and Z. Wang, "Sinnerf: Training neural radiance fields on complex scenes from a single image," 2022.

[28] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black, "3d menagerie: Modeling the 3d shape and pose of animals," 2017.

[29] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," 2017.

[30] B. Biggs, O. Boyne, J. Charles, A. W. Fitzgibbon, and R. Cipolla, "Who left the dogs out? 3d animal reconstruction with expectation maximization in the loop," *CoRR*, vol. abs/2007.11110, 2020.

[31] J. Cao, H. Tang, H.-S. Fang, X. Shen, C. Lu, and Y.-W. Tai, "Cross-domain adaptation for animal pose estimation," 2019.

[32] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley,

S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[33] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2022.

[34] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[35] S. Geman and D. E. McClure, "Statistical methods for tomographic image reconstruction," 1987.

[36] O. Sorkine and M. Alexa, "As-rigid-as-possible surface modeling," in *Symposium on Geometry processing*, vol. 4, pp. 109–116, 2007.

[37] D. A. Hirshberg, M. Loper, E. Rachlin, and M. J. Black, "Coregistration: Simultaneous alignment and modeling of articulated 3d shape," in *European conference on computer vision*, pp. 242–255, Springer, 2012.

8 Appendix A: Meeting Minutes

8.1 Minutes of the 1^{st} Project Meeting

Date: April 30, 2022
Time: 14:00 pm
Place: Online (Zoom)
Present: All of our group members
Absent: None
Recorder: Huang Yiming

1. Approval of minutes

   This was the first formal group meeting, so there were no minutes to approve.

2. Report on progress

   (a) The list of FYP projects has been released. Each team member has picked several projects based on their preferences.

   (b) All team members have read the instructions for the Final Year Project online.

3. Discussion items

   (a) After in-depth communication, we decided to do deep learning and computer vision-related projects with Prof.Chen.

   (b) We briefly went through the hot topics in the field of computer vision, see a collection of CVPR2022.

   (c) Our group members were particularly interested in two papers: BARC and Local Radiance Fields for Human Avatar Modeling. We will further ask for Prof. Chen's opinions.

4. Goals for the coming week

   All members will explore more interesting papers for the project and prepare for the coming meeting with Prof. Chen.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 15:00 pm.

   The next meeting will be at 15:00 am on May 3^{rd} at Zoom with Prof.Chen.

8.2 Minutes of the 2^{nd} Project Meeting

Date: May 3, 2022
Time: 15:00 pm
Place: Online (Zoom)
Present: All of our group members and Prof.Chen
Absent: None
Recorder: Lin Yiqiao

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   (a) We confirmed the commitment with Prof. Chen Qifeng, who is the project supervisor.

   (b) All the group members had done some basic reading for the project.

3. Discussion items

(a) We introduced two papers (BARC and Local Radiance Fields for Human Avatar Modeling) to Prof. Chen.

(b) We basically determine that our project will focus on BARC.

4. Goals for the coming month

   All members begin to read the paper and summarize some key points.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 16:00 pm.

   The next meeting will be at 17:00 pm on June 24^{th} at Zoom.

8.3 Minutes of the 3^{rd} Project Meeting

Date: June 24, 2022,
Time: 17:00 pm
Place: Online (Zoom)
Present: All of our group members
Absent: None
Recorder: Lin Yiqiao

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   (a) The code of BARC has been released on barc release. We will do further training.

   (b) All the group members have read the paper.

3. Discussion items

   (a) We quickly went through the paper and pointed out some future directions we can work on: refinement of the current BARC model, colorization, controllable pose, model editing, style transfer, etc.

   (b) We also explored some related papers: Neural Head Avatars from Monocular RGB Videos and StyleNeRF.

   (c) Yiming gave a tutorial about how to set up the environment in CSE server.

4. Goals for the coming month

   All members continue to read papers and debug the released code.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 18:00 pm.

   The next meeting will be at 14:30 pm on July 12^{th} at Zoom with Prof. Chen.

8.4 Minutes of the 4^{th} Project Meeting

Date: July 12, 2022,
Time: 14:30 pm
Place: Online (Zoom)
Present: All of our group members, Prof.Chen
Absent: None
Recorder: Tang Zichen

1. Approval of minutes

CQF6 FYP-Implicit Models for Scene Representations

The minutes of the last meeting were approved without amendment.

2. Report on progress

   (a) We finished debugging the code of BARC.

   (b) Yiming trained the BARC model and tested the model with the StanfordExtra dataset. The overall accuracy of the model was satisfactory. However, we also noticed that the tracked key points in some images were incorrect.

3. Discussion items

   (a) We showed the results of the training to Prof. Chen.

   (b) Prof. Chen pointed out several directions we could refine the BARC model. First, we could color it and add fur details that are missing. We could also have a video input instead of a photo input.

4. Goals for the coming month

   (a) We want to try explicit modeling and implicit modeling on BARC to refine the model. Each member will read their assigned papers (BARC,Neural Head Avatars from Monocular RGB Videos, I M avatar and NerFace)and provide a detailed literature review.

   (b) We will start to write the project proposal.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 17:00 pm.

   The next meeting will be at 15:00 pm on Sept 6 th at Room 3508 with Prof. Chen.

8.5 Minutes of the 5^{th} Project Meeting

Date: Sept 6, 2022
Time: 15:00 pm
Place: Room 3508
Present: All of our group members, Prof.Chen
Absent: None
Recorder: Huang Yiming

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   (a) Each member summarizes their research findings in literature review.

   (b) We have completed the introduction and literature review of the project proposal.

3. Discussion items

   (a) We showed our literature review to Prof. Chen. Prof. Chen said that implicit modeling might be hard for our case. Therefore we would only try explicit modeling. He also suggested that we could focus on controllable pose first.

   (b) Prof. Chen also checked our project proposal and said the format was fine.

4. Goals for the coming month

   (a) Finish the project proposal.

(b) Retrain the BARC model using video input.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 16:00 pm.

   The next meeting will be scheduled after the training are done. The time and venue will be announced via WeChat.

8.6 Minutes of the 6^{th} Project Meeting

Date: Oct 25, 2022,
Time: 14:00 pm
Place: Room 3508
Present: All of our group members, Prof.Chen
Absent: None
Recorder: TANG Zichen

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   (a) We do the Video Refinement according to the SMALIFY.

   (b) We use the New energy term for regressing the geometry model

3. Discussion items

   (a) We discussed with Prof.Chen our major difficulty, which is the limited accuracy of the pose and segmentation map estimation. Prof. Chen offered us some possible solutions.

4. Goals for the coming month

(a) Doing comparison experiment on multi-frame fusion result between each method.

(b) Using 2D segmentation as a refinement for the shape prediction.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 14:35 pm.

   The next meeting will be scheduled after the training are done. The time and venue will be announced via WeChat.

8.7 Minutes of the 7^{th} Project Meeting

Date: Nov 25, 2022
Time: 17:35 pm
Place: Zoom
Present: All of our group members, Prof.Chen
Absent: None
Recorder: Lin Yiqiao

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   (a) We re-implement coarse-to-fine and WLDO networks for comparison and do testing on the StanfordExtra datasets.

   (b) We implement 2D segmentation with SOTA semantic segmentation network on StanfordExtra.

3. Discussion items

(a) We explore the feasibility of utilizing the two new networks to improve BARC with Prof. Chen.

(b) We also did the statistical analysis and got advice from Prof. Chen for further improvement.

4. Goals for the coming month

   (a) Further explore the structure of the Coarse-to-fine network.

   (b) Refine BARC.

5. Meeting adjournment and next meeting

   The meeting was adjourned at 18:00 pm.

   The next meeting will be scheduled after the training are done. The time and venue will be announced via WeChat.

8.8 Minutes of the 8^{th} Project Meeting

Date: Jan 11, 2023,
Time: 16:00 pm
Place: Zoom
Present: All of our group members, Prof.Chen
Absent: None
Recorder: Huang Yiming

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Report on progress

   (a) We analyze the training result and find common patterns of in-

accurate models and key points in both BARC and Coarse-to-fine networks.

(b) We analyze and compare the network structure of BARC and Coarse-to-fine.

3. Discussion items

(a) We showed Prof. Chen some examples of the inaccurate training result of both BARC and Coarse-to-fine.

(b) We also did the statistical analysis and got advice from Prof. Chen for further improvement.

4. Goals for the coming month

(a) Finish the project progress report.

(b) Improve based on the coarse-to-fine networks.

5. Meeting adjournment and next meeting

The meeting was adjourned at 16:45 pm.

The next meeting will be scheduled after the training is done. The time and venue will be announced via WeChat.

8.9 Minutes of the 9^{th} Project Meeting

Date: Feb 02, 2023,
Time: 19:00 pm
Place: Zoom
Present: All of our group members
Absent: None
Recorder: Huang Yiming

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Discussion items

   (a) We watched the video recording for the tutoring session in Jan 16^{th} and modified the progress report based on the tutor's advice.

3. Goals for the coming month

   (a) Finish the project progress report.

4. Meeting adjournment and next meeting

   The meeting was adjourned at 19:45 pm.

   The next meeting will be scheduled after the training is done. The time and venue will be announced via WeChat.

8.10 Minutes of the 10^{th} Project Meeting

Date: Apr 02, 2023,
Time: 14:00 pm
Place: Zoom
Present: All of our group members
Absent: None
Recorder: TANG Zichen

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Discussion items

(a) We summarized the progress and made plans to write the final report for our project.

3. Goals for the coming month

(a) Finish the final report.

4. Meeting adjournment and next meeting

The meeting was adjourned at 14:25 pm.

The next meeting will be scheduled after the training is done. The time and venue will be announced via WeChat.

8.11 Minutes of the 11^{th} Project Meeting

Date: Apr 14, 2023,
Time: 17:00 pm
Place: Zoom
Present: All of our group members, Ms.Noor Liza Daveau
Absent: None
Recorder: Zoom recording

1. Approval of minutes

The minutes of the last meeting were approved without amendment.

2. Discussion items

(a) The tutor analyze the structure and content of our progress report and made several suggestions to make our report more coherent and effective. We discussed the possible modifications and effective structure of the final report with the tutor.

3. Goals for the coming month

(a) Finish the final report.

4. Meeting adjournment and next meeting

   The meeting was adjourned at 18:05 pm.

   The next meeting will be scheduled after the training is done. The time and venue will be announced via WeChat.

8.12 Minutes of the 12^{th} Project Meeting

Date: Apr 18, 2023,
Time: 15:00 pm
Place: Zoom
Present: All of our group members
Absent: None
Recorder: LIN Yiqiao

1. Approval of minutes

   The minutes of the last meeting were approved without amendment.

2. Discussion items

   (a) We watched the video recording for the tutoring session in Apr 14^{th} and modified the final report based on the tutor's advice.

3. Goals for the coming month

   (a) Finish and submit the final report.

4. Meeting adjournment and next meeting

   The meeting was adjourned at 16:45 pm.

The next meeting will be scheduled after the training is done. The time and venue will be announced via WeChat.

9 Appendix B: Supplementary Details for SMAL Modeling

Global/Local Stitched Shape model. To define the GLoSS parametrization, we set n_i as the number of vertices in part i and use the vector coordinate $\hat{\mathbf{p}}_i$ in a global reference frame and $p_i^{3\times n_i}$ in a local coordinate frame as description.
For $\hat{\mathbf{p}}_i$ we have:

$$\hat{\mathbf{p}}_i(\pi_i) = R(\mathbf{r}_i)\mathbf{p}_i + \mathbf{I}_i, \tag{24}$$

where the $\hat{\mathbf{p}}_i \in \mathbb{R}^{3\times n_i}$ is the vertex coordinates for part $i \in (1...N)$ which takes the variable set $\pi_i = \{\mathbf{I}_i, \mathbf{r}_i, \mathbf{s}_i, \mathbf{d}_i\}$ of part i as input. Here, $\mathbf{I}_i \in \mathbb{R}^{3\times 1}$ is the part location, $\mathbf{r}_i \in \mathbb{R}^{3\times 1}$ is the Rodrigues vector for 3D rotation, $\mathbf{s}_i \in \mathbb{R}^{n_s\times 1}$ is the intrinsic shape variables and $\mathbf{d}_i \in \mathbb{R}^{n_d\times 1}$ is the pose deformation variables. The total set of variables for all the parts is described as $\prod = \{\mathbf{I, r, s, d}\}$. To represent $p_i^{3\times n_i}$ we have:

$$vec(\mathbf{p}_i) = \mathbf{t}_i + \mathbf{m}_{p,i} + B_{s,i}\mathbf{s}_i + B_{p,i}\mathbf{d}_i. \tag{25}$$

$\mathbf{t}_i \in \mathbb{R}^{3n_i\times 1}$ is the points from the i^{th} part of the template, $\mathbf{m}_{p,i} \in \mathbb{R}^{3n_i\times 1}$ is the average pose displacement, $B_{s,i} \in \mathbb{R}^{3n_i\times n_s}$ is a matrix where the column represents the basis of the intrinsic shape displacement, and $B_{p,i} \in \mathbb{R}^{3n_i\times n_d}$ is the pose dependent deformation matrix.

Pose Deformation Space. To define the pose deformation matrix, we first perform LBS (Linear blend skinning) on the lioness template and then apply the PCA (Principal component analysis) on the vertices of each part, retrieving the average pose deformation $\mathbf{m}_{p,i}$ and the basis matrix $B_{p,i}$.

Shape Deformation Space. For the definition of the shape deformation,

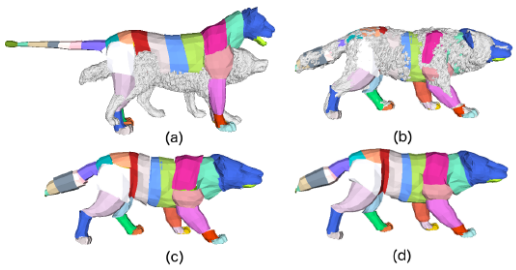


Figure 31: **GLoSS fitting result**. (a) Initial template and scanned model. (b) GLoSS initialization. (c) GLoSS with parts. (d) Merged mesh

we use 7 scales of the part template which include the scale for the whole part, scales along the x, y, and z axis, and 3 scales for the combination of any two of the x, y, and z-axis. The shape coefficients are further modeled as a Gaussian distribution with zero mean and diagonal covariance.

GLoSS-based Registration. Then we represent the GLoSS-based template registration by minimizing the following objective:

$$E(\Pi) = E_m(\mathbf{d},\mathbf{s}) + E_{stitch}(\Pi) + E_{curv}(\Pi) + E_{data}(\Pi) + E_{pose}(\mathbf{r}), \tag{26}$$

where

$$E_m(\mathbf{d},\mathbf{s}) = k_{sm}E_{sm}(\mathbf{s}) + k_s\sum_{i=1}^{N} E_s(\mathbf{s}_i) + k_d\sum_{i=1}^{N} E_d(\mathbf{d}_i). \tag{27}$$

E_s is the squared Mahalanobis distance for the GLoSS shape distribution. and E_d is a squared L2 norm for the shape coefficients. E_{sm} is for restricting the symmetric parts to have the same deformation by computing the similarity of the left and right torsos.
E_{stitch} is the sum of squared distances of the point correspondences between two surfaces. Let C_{ij} be a point correspondence for part i and j, we have:

$$E_{stitch}(\Pi) = k_{st} \sum_{(i,j)\in C} \sum_{(k,l)\in C_{ij}} \|\hat{p}_{i,k}(\pi_i) - \hat{p}_{j,l}(\pi_j)\|^2. \tag{28}$$

By minimizing this part, we can refine the surface connection.
$E_{data}(\Pi)$ is the data term, it is define as:

$$E_{data}(\Pi) = k_{kp}E_{kp}(\Pi) + k_{m2s}E_{m2s}(\Pi) + k_{s2m}E_{s2m}(\Pi), \tag{29}$$

where E_{m2s} and E_{s2m} are the distances from the model to the scan and from the scan to the model:

$$E_{m2s}(\Pi) = \sum_{i=1}^{N} \sum_{k=1}^{n_i} \rho(\min_{s\in S} \|\hat{p}_{i,k}(\pi_i) - \mathbf{s}\|^2), \tag{30}$$

$$E_{s2m}(\Pi) = \sum_{l=1}^{S} \rho(\min_{\hat{p}} \|\hat{p}(\Pi) - \mathbf{s}_l\|^2), \tag{31}$$

where S is the set of scanned vertices and ρ is the Geman-McClure robust error function [35]. $E_{kp}(\Pi)$ is the sum of the squared distances between the matching model keypoints with scan keypoints.

$$E_{curv}(\Pi) = k_c \sum_{(i,j)\in C} \sum_{(k,l)\in C_{ij}} \|\|\hat{n}_{i,k}(\pi_i) - \hat{n}_{j,i}(\pi_j)\|^2 - \|\hat{n}_{i,k}^{(t)} - \hat{n}_{j,l}^{(t)}\|^2|, \tag{32}$$

where $\hat{n}_i$ and $\hat{n}_j$ are the vector norms for part i and j. And the last term E_{pose} is the pose prior for the tail parts. The weight parameter k is manually defined and kept constant for each animal.

ARAP-based refinement
Given the GLoSS model, we further refine the vertices alignment between the scanned and template model by minimizing the data term that is the same as Eq. 29 and applying the As-Regid-As-Possible(ARAP) regularization term [36]:

$$E(\mathbf{v}) = E_{data}(\mathbf{v}) + E_{arap}(\mathbf{v}). \tag{33}$$

Skinned Multi-Animal Linear Modeling
Given the registered GLoSS model, we want to further refine it and produce the full SMAL model. Here are the detailed steps of full SMAL refinement from Pose Normalization to Refinement with Dog Specification.

Pose Normalization For GLoSS models obtained for multi-animals, we further register them into the same pose using LBS. Since the result from LBS might enlarge the asymmetry of the left and right pose, we then averaged the vertices after the re-registration. Also, the mouth pose of each registered animal might be different. To resolve this problem, a linear model is learned from the template to regress the palate and tongue points from the mouth points. The final step for the normalization is applying Laplacian smoothing to the whole mesh.

Co-registration Inspired by [37], we apply co-registration as follows:

$$E_{coup}(v) = k_o \sum_{i=1}^{V} |\mathbf{v}_i^0 - \mathbf{v}_i|, \tag{34}$$

where V is the number of vertices in the template, v_i^0 is the i^{th} vertex of template, and v_i is the corresponding vertex after the previous SMAL function. The co-registration is performed in 4 iterations. The output of the last iteration is the final SMAL model.

Refinement with Dog Specification The animal model used in our proposed method is a variant of the original SMAL, which is specifically mod-

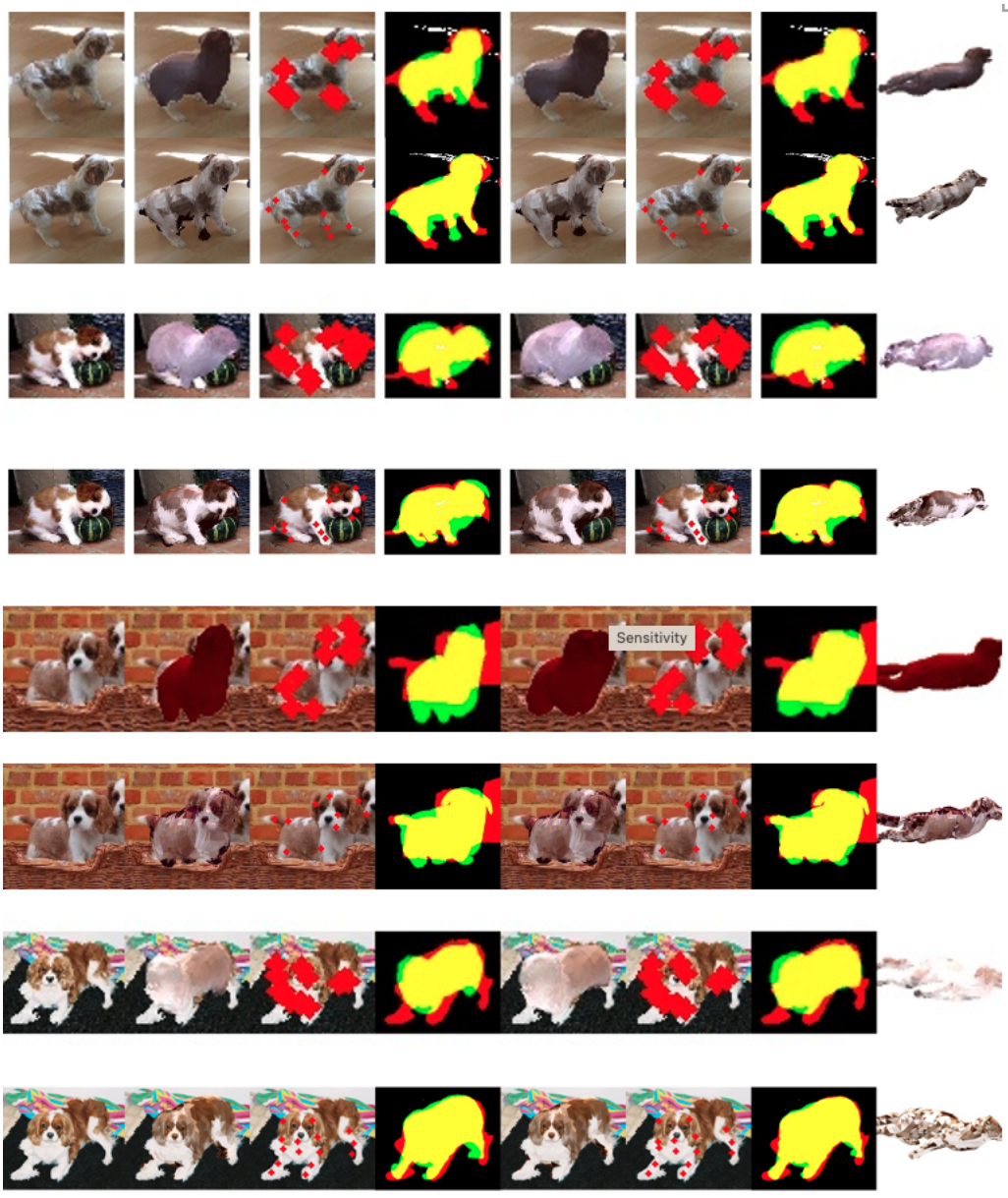
ified to fit the dog shape. Except for training with the scanned model from the SMAL dataset [14], this model further utilizes the 3D *Unity* model and enlarges the weight for dogs. Furthermore, the torso lengths are fixed to 1 and extended with the scale parameters k (actual scale is $\exp(k)$). The scale k is also merged into PCA shape coefficients β_{pca} as a new shape vector β.

10 Appendix C: Gallery for Testing Cases

10.1 Texture Models Testing Results

This section is the gallery of our testing results for Vanilla Texture Decoder (top) and Vanilla Texture Field (bottom), using a subset of the StanfordExtra Dog Dataset [30].From left to right are: **input, textured shape, keypoints, segmentation map, refined textured shape, refined keypoints, refined segmentation map, novel view.**


Sensitivity


Sensitivity