Markerless Macaque Face Reconstruction Using Synthetic Data

Anik Gupta

Markerless Macaque Face Reconstruction Using Synthetic Data

by Anik Gupta

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

JMalik

Professor Jitendra Malik
Research Advisor

May 8, 2024

Date

[illegible]
* * * * * * *

May 13, 2024

Professor Doris Tsao
Second Reader

Date

Markerless Macaque Face Reconstruction Using Synthetic Data

by

Anik Gupta

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Jitendra Malik, Chair
Professor Doris Tsao

Spring 2024

Markerless Macaque Face Reconstruction Using Synthetic Data

Abstract

Markerless Macaque Face Reconstruction Using Synthetic Data

by

Anik Gupta

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Jitendra Malik, Chair

This paper introduces a novel methodology for automating the animation of macaque monkey avatars by predicting blendshapes from live video streams and recordings. Our approach leverages advancements in computer vision, particularly in the generation of multifaceted synthetic datasets using stable diffusion techniques, tailored to a macaque avatar. Our method revolves around training a model to directly predict blendshapes from images. We create a synthetic dataset that contains fully accurate ground-truth avatar renders, depth images, blendshapes, camera parameters, and avatar vertices. Importantly, no manual annotations are needed on real imagery for training, and the entire dataset creation process is automated. Additionally, we utilize an off-the-shelf ResNet model for regressing blendshapes from our generated macaque avatar image datasets. Remarkably, due to the comprehensive nature of our dataset, the training process requires no special modifications. Experimental results demonstrate the effectiveness of our approach in accurately animating macaque monkey avatars and capturing a diverse range of facial expressions. By providing researchers with an automated and cost-effective tool for avatar animation, our paper contributes to applications in various research domains, especially neuroscience.

To my parents

Contents

List of Figures

Acknowledgments

I would like to express my deepest gratitude to Professor Jitendra Malik and Professor Doris Tsao for their valuable feedback and insights throughout my Master's.

I am immensely grateful to Shiry Ginosar and Nate Dolensek, whose mentorship and expertise were essential in guiding me through the various stages of this project. Their dedication and guidance have been pivotal in shaping the direction and outcomes of my research.

Thank you to the team at Pinscreen, particularly Hao Li and Liwen Hu, for their valuable feedback and collaboration, who significantly contributed to the development of this work by providing new ideas and suggestions.

I am extremely appreciative of my coworkers at Microsoft for their support and flexibility throughout this endeavor. Their encouragement and understanding have been invaluable during the course of this project.

To my family and friends, thank you for your unwavering belief in me, support, and encouragement. Thank you so much for the countless memories we've shared during my time here and for helping shape me to become the person I am today.

Chapter 1

Introduction

Advancements in neuroscience have long relied on the ability to understand and interpret neural responses to various stimuli. Building upon the foundational work of Hubel and Wiesel, who pioneered the understanding of receptive fields through experiments on cats [1], modern neuroscience seeks to delve deeper into the neural processes underlying complex behaviors and social interactions. One direction of exploration involves studying facial expressions in primates, particularly macaque monkeys, to gain insights into social structures and behaviors.

Traditionally, neuroscience experiments have utilized static images as stimuli, probing neural responses by presenting controlled visual inputs to subjects. However, recent efforts have aimed to incorporate more dynamic and interactive stimuli, mirroring real-world scenarios more closely. In this context, the development of 3D avatars of macaque monkeys has emerged as a promising tool for researchers, allowing for the creation of controllable and responsive stimuli while maintaining realism and avoiding the uncanny valley effect [2–4].

Neuroscience groups have been at the forefront of this research, seeking to explore the visual-neuro-social processes in macaques through computational modeling and experimentation [5, 6]. By creating computational representations of monkey faces using blendshapes, researchers can manipulate facial expressions and observe corresponding neural responses, paving the way for a deeper understanding of primate social interactions and potential applications in neuropsychiatric research.

This paper proposes a novel method for automatically animating macaque monkey avatars by predicting blendshapes from live video streams or recordings as shown in Figure 1.1. By leveraging image frames and directly regressing blendshapes, this approach eliminates the need for manual animation or data labeling and provides researchers with real-time expression parameters for downstream tasks. By correlating neural signals with measured facial expressions through blendshapes, researchers can gain insights into the neural mechanisms underlying social behaviors and interactions.

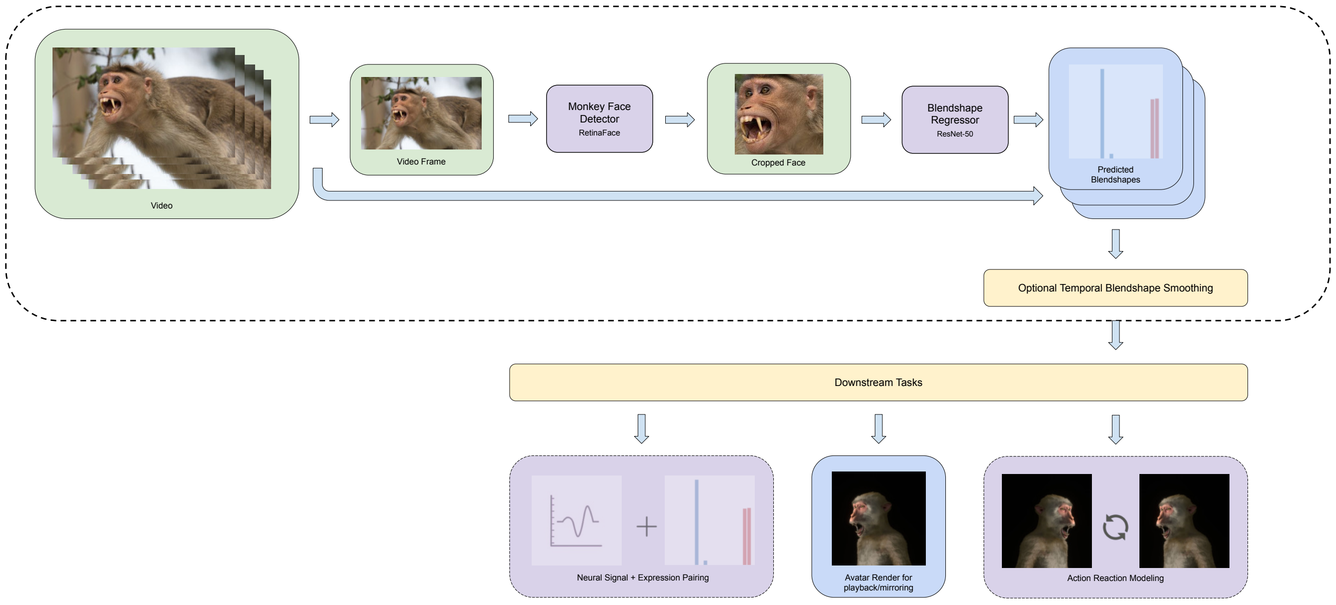


Figure 1.1: **Overall project workflow**. We demonstrate taking individual frames from a video of a macaque monkey, detecting and cropping its face, and predicting blendshapes for our avatar. These blendshapes can be used for future downstream tasks such as establishing correlations with neural signal, conducting mirroring experiments, and modeling action-reaction patterns.

The foundation of this method lies in the generation of synthetic datasets using the macaque avatar, augmented using techniques such as Stable Diffusion [7] and ControlNet [8]. These datasets enable the training of a ResNet-50 [9] to predict blendshapes directly from image frames, without the need for markers, motion capture, or manual annotation [10]. Through experimentation and validation, this paper aims to demonstrate the effectiveness of the proposed approach as shown in Figure 1.2.

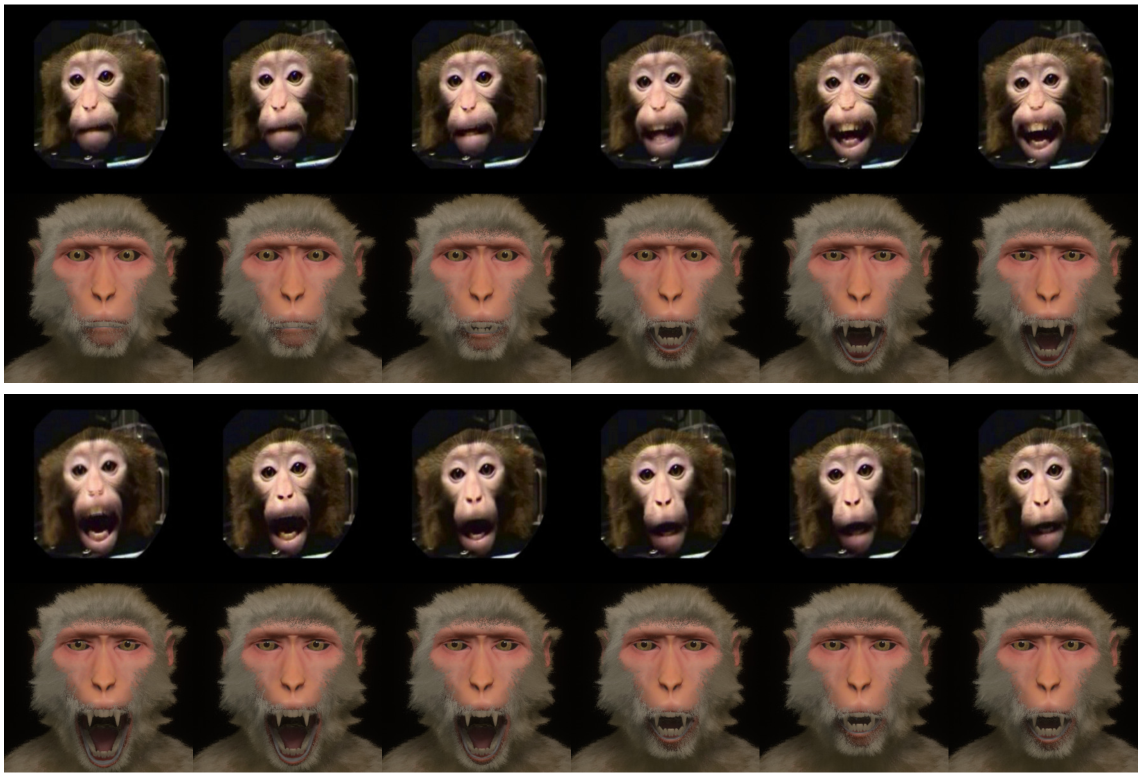


Figure 1.2: **Example clip of a real monkey and the corresponding animated avatar** rendered according to the blendshapes predicted by the final model.

Chapter 2

Related Works

2.1 Face Detection

Face detection is a fundamental task in computer vision, crucial for various applications such as face recognition, facial expression analysis, and avatar modeling. It is often used as a preprocessing step to crop images before landmark detection, pose estimation, or reconstruction. For this problem, there are two main categories of techniques used: traditional and deep-learning based. Viola and Jones [11] proposed the Haar-cascade classifier, which employs a series of classifiers trained on Haar-like features to detect faces efficiently. While less accurate compared to deep-learning based methods, Haar cascade classifiers offer fast inference speeds and can be suitable for real-time applications with limited computational resources. It is used most often due to its efficiency and accuracy. On the other hand, newer deep-learning techniques that harness the power of better GPUs have proven to be more accurate. Deng et al. [12] proposed RetinaFace, which is a multi-task deep-learning based face detector that achieves state-of-the-art performance across a wide range of face scales. It utilizes a single unified network to predict face bounding boxes, facial landmarks, and face attributes simultaneously. This multi-task loss enables it to effectively detect faces and perform well on many datasets such as WiderFace [13].

2.2 Morphable Models

3D morphable face models (3DMM) [14] provide a compact representation of facial shape and appearance variations, facilitating tasks such as avatar modeling, facial animation, and facial expression synthesis. The ICT-FaceKit [15] provides a comprehensive morphable face model and toolkit for generating realistic facial animations. It encompasses a detailed parametric model of facial geometry and appearance variations, along with tools for facial animation and expression synthesis. FLAME (Faces Learned with an Articulated Model and Expressions) [16] is another widely used morphable model that enables detailed modeling of facial shapes and expressions. It incorporates a physics-based model of facial tissue deformations, enabling realistic simulation of facial expressions and dynamics. These morphable models serve as valuable resources for avatar modeling, animation, and virtual character creation, enabling researchers to generate realistic and expressive facial animations for various applications [17]. While these are mostly used for human faces, there are a few for macaques developed by Siebert et al. [3] and Murphy and Leopold [5]. Unfortunately, these are not publicly available.

2.3 Datasets

Face datasets are critical for training and evaluating computer vision models, particularly for tasks involving facial analysis, landmark detection, and avatar modeling. There are many datasets for human faces designed for many different benchmarking tasks [13, 18–21]. Additionally, Wood et al. [22] proposed a dataset called "Fake It Till You Make It" which is a synthetic human face dataset generated using a 3DMM and differentiable renderer. It provides diverse facial expressions and poses for training avatar models, and it contains ground-truth landmarks and face segmentations. However, datasets specifically tailored to monkey faces remain limited in availability, highlighting the need for dedicated efforts in data collection and annotation for primate facial analysis tasks. Those that do exist are sparse and do not provide more than basic face detections [6, 23, 24].

2.4 Landmark Detection/Face Reconstruction

Landmark detection is one of the most fundamental tasks in computer vision. In addition, it can be used for face reconstruction. Wood et al. [25] developed a way of using their synthetic human face dataset to predict a dense set of facial landmarks with high accuracy. It utilizes a convolutional neural network (CNN) architecture trained on their large-scale synthetically annotated facial landmark datasets to predict dense landmarks with confidence values, even if they are occluded. They use these 2D landmarks to create a reprojection loss between their 3DMM vertices and the landmarks to determine blendshape values and reconstruct faces. DeepLabCut [26, 27] is another widely used tool for pose estimation and behavioral analysis, particularly in neuroscience research involving animal models. It employs a deep-learning based approach to estimate key points on animal bodies or faces from video frames, enabling researchers to analyze complex behaviors and interactions. Witham [28] implemented a custom DeepLabCut model specifically for primate faces which can predict 55 landmarks.

2.5 Image Generation

Diffusion models and image generation techniques play a crucial role in generating realistic and diverse images, and they can be controlled to create new faces. Stable Diffusion by Rombach et al. [7] is a diffusion-based generative model that achieves high-quality image synthesis provided a text prompt. It utilizes a diffusion process to iteratively refine a noise vector into a realistic image. However, using Stable Diffusion tends to be an iterative process of writing and refining text prompts without being able to provide any visual feedback. ControlNet proposed by Zhang et al. [8] allows further control by allowing users to manipulate specific attributes of generated images with conditioning images. It employs fixed and trainable copies of Stable Diffusion and combines them at every UNet level using zero-convolutions. This allows for proper training of ControlNet and enforces the conditioning image. Additionally, inpainting methods [7] are commonly used for filling in missing regions in images. They can be used for filling in backgrounds or adding in missing details.

Chapter 3

Method

In this section, we delve deeper into the methodology we use. This involves generating a brand new multi-faceted dataset using our macaque avatar, training a blendshape predictor, augmenting our dataset to be more realistic using Stable Diffusion, finetuning our blendshape predictor, and training a monkey face detector.

3.1 Avatar Dataset

Since there are not many annotated macaque face datasets, and there is no way to map existing videos of macaque face images to the blendshapes of our custom avatar without creating large-scale manual annotations, we generate synthetic data using our avatar. Our dataset contains 100,000 rows generated using randomized blendshapes, lighting, and camera angles. To maintain that our dataset stays in the domain of realistic scenarios, we place constraints on these parameters. The camera is constrained to be randomly selected from a range of ±60 degrees along the vertical axis centered at the avatar, ±10 degrees along the horizontal axis centered at the avatar, and ±10 centimeters translation closer or further away from the avatar. In addition, the light source can vary between 1 to 3 meters away from the avatar. The avatar itself is designed to be animated using a few blendshapes at a time, and setting many blendshapes to high values leads to deformed-looking faces. To reduce the number of deformed images in the dataset, we place constraints on the 17 blendshapes as well. Each blendshape is initially set to zero with a 60% probability. The remaining nonzero blendshapes are randomly set to a value between 0 and 1. Finally, only if the sum of all blendshape values is greater than 5, we reweight them so they sum up to 5 as described in Equation 3.1.

$$b^i_{\text{target}} = \frac{5}{\sum_{j=1}^{17} b^j_{\text{random}}} b^i_{\text{random}} \tag{3.1}$$

Lastly, the rotation of the two eyes is paired, with the rotation along the vertical axis constrained to ±20 degrees and ±5 degrees along the horizontal axis. These values are selected based on which angles the eye remains visible when rendered using the avatar. Both

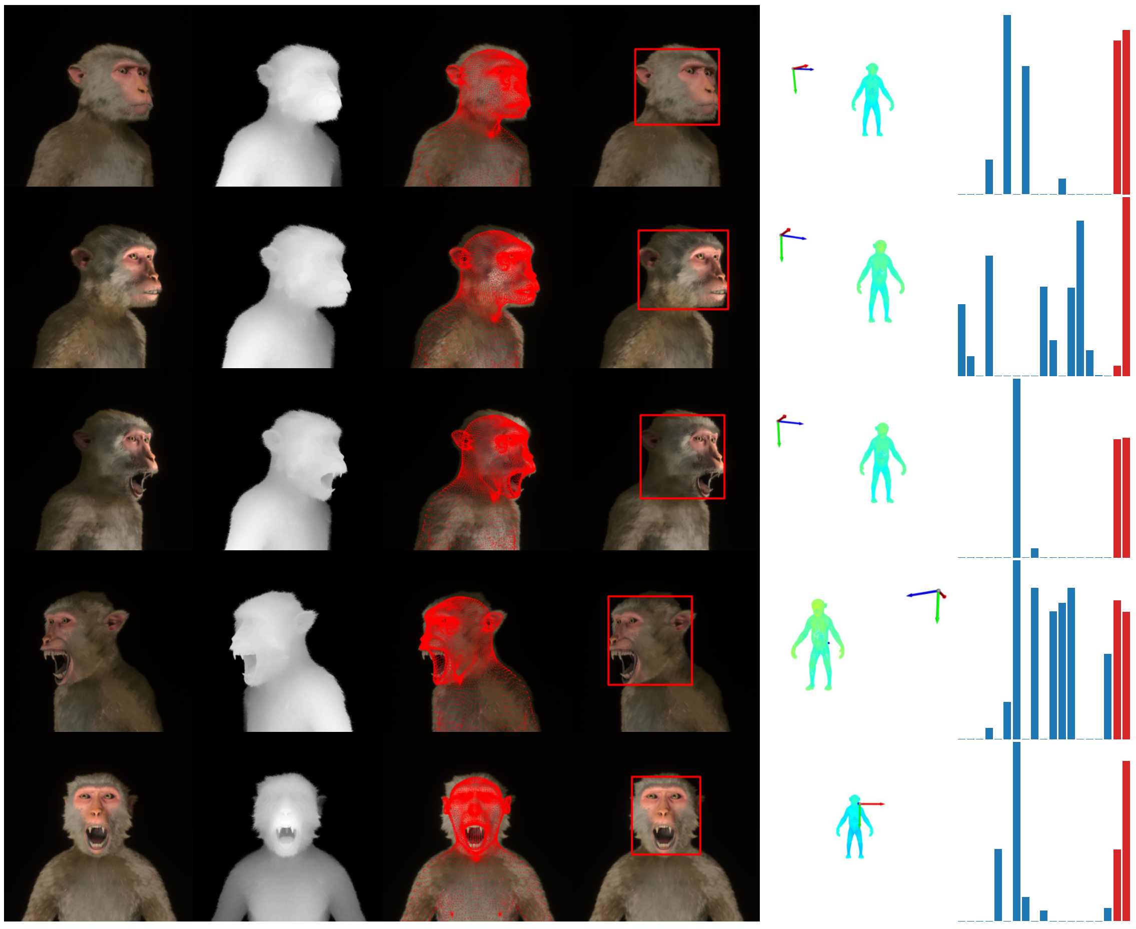


Figure 3.1: **Examples of rows in the generated avatar dataset.** The first column shows captured images of the avatar. The second column shows the depth map of the scene. The third column displays 3D mesh vertices of the avatar projected back onto the image. The fourth column shows a bounding box around the face. The fifth column displays the camera location relative to the 3D mesh of the entire monkey avatar. The blue arrow indicates the z-axis and represents the principal axis of the camera, with the x-axis in red and y-axis is green. The sixth column displays the values of the blendshapes and the eye rotations. The first 17 bars (in blue) indicate the values of the 17 blendshapes from 0 to 1. The last 2 bars (in red) indicate two axes of eye rotations scaled from 0 to 1.

eyes are guaranteed to be within 1 degree of each other along both rotational axes. All of these parameters can be configured and adjusted based on the real-world domain of monkey images. They can be easily modified for future dataset generation. The values listed here were experimentally determined based on the real videos on which the blendshape predictor will be run. After setting all the parameters, all 17 blendshapes and 2 eye rotations are normalized to 0 to 1 based on their configured maximum and minimum values.

After randomizing blendshapes, eye rotations, camera location, and lighting, our pipeline modifies the Unity scene to visualize and render these changes. With these transformations, we write out a 1024×1024 pixel image render of the monkey avatar, the blendshapes and eye rotation values, a depth map using Unity's depth (Z) buffer [29], 3D mesh vertices of the avatar in world coordinates, and camera parameters (including intrinsics and extrinsics). Using the camera parameters, 3D mesh vertices are also projected onto the image plane to create a set of dense landmarks. Because the avatar contains the monkey's entire body (containing 30,247 vertices), we can extract the 8,769 mesh vertices that only belong on the face. Projecting just these face vertices onto the image using the camera parameters also provides us with ground-truth face bounding boxes. We get the coordinates of the box using the minimum and maximum x and y coordinates of these 2D face vertices. Figure 3.1 shows a set of 5 example rows in the dataset, including the avatar render, depth render, 2D/3D keypoints, face bounding box, camera parameters, and blendshapes. Figure 3.2 shows a random subset of 30 avatar renders that belong to our training dataset. On a single RTX 3080, generating all the images took 32 hours.

3.2 Avatar Linear Skinning Model

To effectively use the blendshapes and model results in a non-Unity environment for deep learning approaches (such as PyTorch), we need to accurately model how the 3DMM avatar warps and molds based on blendshapes that are passed to it. Figure 3.3 displays the neutral face in the middle and a subset of 6 of the 17 blendshapes of our avatar, which manifest themselves as different expressions. A linear blend skinning model models the vertex locations of the avatar given the blendshapes. First, it uses the locations of the vertices in a neutral expression (all blendshapes set to 0). Each blendshape determines an offset for each vertex by specifying a specific direction and magnitude (represented by a vector in 3D), which is scaled linearly by the blendshape parameter. This linear combination can be expressed by a simplified version of the linear blend skinning (LBS) model [30]. We eliminate joint poses in our model since they are not relevant to our formulation and problem.

$$\mathbf{V}_i(\mathbf{b}) = \mathbf{N}_i + \sum_{k=1}^{|\mathbf{b}|} \mathbf{b}_k \mathbf{W}_{ki} \tag{3.2}$$

where $\mathbf{V}_i$ ($\mathbb{R}^3$) is the ith blended vertex position, and $\mathbf{N}_i$ ($\mathbb{R}^3$) represents the ith vertex position when in a neutral expression. $\mathbf{b}$ is a vector of scalars ($\mathbb{R}^{17}$ in our case) that are

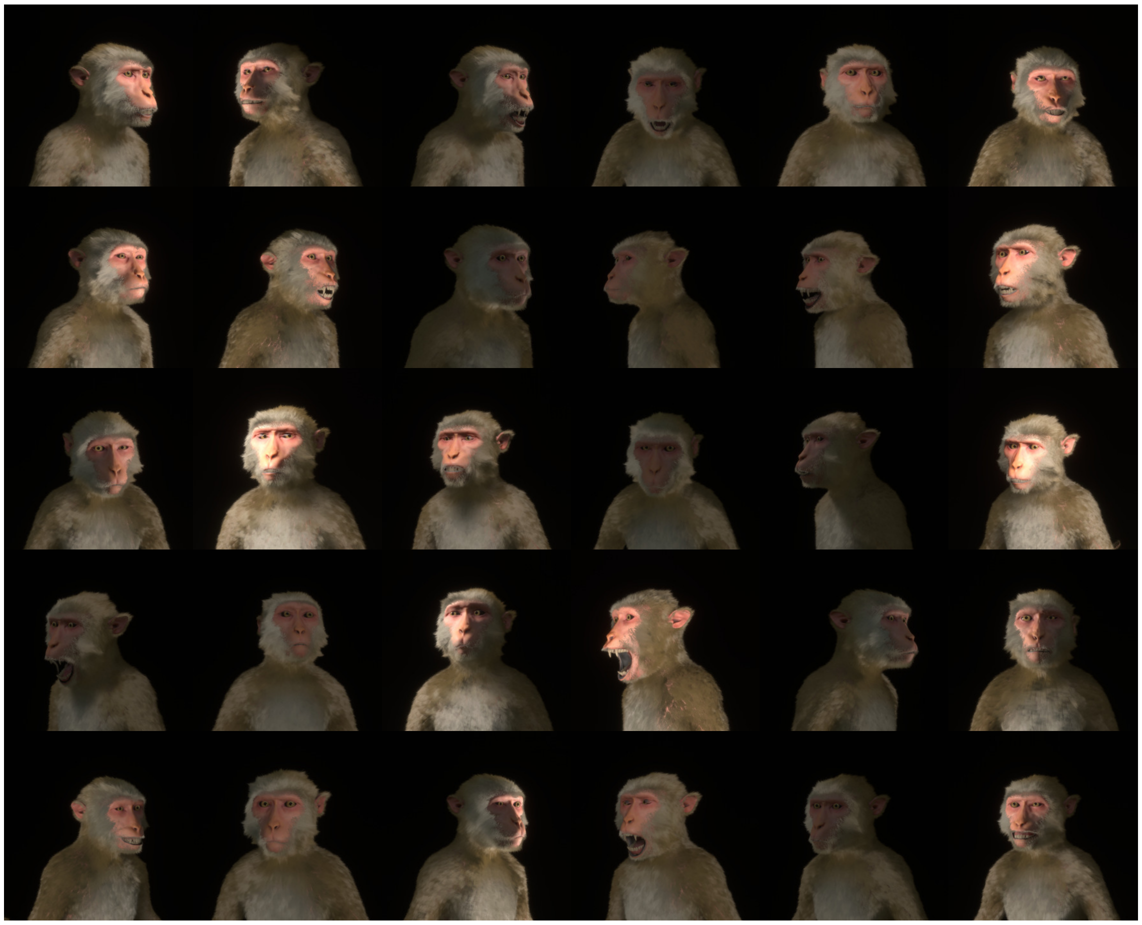


Figure 3.2: **Random subset of 30 avatar images** generated that are part of the final dataset.

used to scale each blendshape. $\mathbf{W}$ ($\mathbb{R}^{|b|\times|V|\times 3}$) represents the hardcoded blendshapes and how the ith vertex transforms scaled by the kth blendshape. This can also be represented using tensors:

$$\mathbf{V}(\mathbf{b}) = \mathbf{N} + \mathbf{b}\mathbf{W} \tag{3.3}$$

which makes this an efficient operation.

In order to take the Unity avatar and create a Python model, we need to determine $\mathbf{N}$ and $\mathbf{W}$, so we can parameterize the model using our 17 blendshape scalars $\mathbf{b}$. To get $\mathbf{N}$, we extract mesh vertices after setting all blendshapes ($\mathbf{b}$) to 0, representing the neutral state. Then, we set each of the 17 blendshapes to 1 independently and save the mesh vertices in each of these 17 configurations. In these cases, $\mathbf{b}$ becomes a one-hot vector, and we have determined $\mathbf{V}(\mathbf{b}_k = 1, \mathbf{b}_{j\neq k} = 0) = \mathbf{N} + \mathbf{W}_k$ for each blendshape index k. Knowing $\mathbf{N}$, we determine $\mathbf{W}_k = \mathbf{V}(\mathbf{b}_k = 1, \mathbf{b}_{j\neq k} = 0) - \mathbf{N}$, which provides us vertex directional vectors for

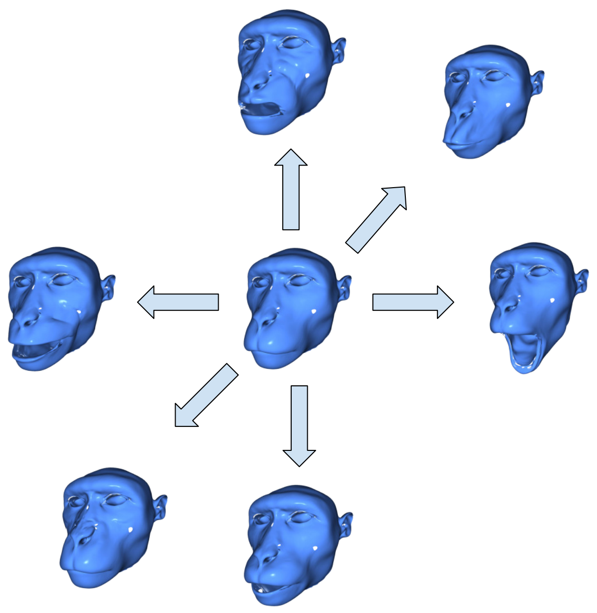


Figure 3.3: **Subset of 6 of the 17 blendshapes** that show the wide variety of expressions the avatar can make. The middle avatar displays the neutral expression, while the other faces demonstrate a set of expressions.

each blendshape. We then stack these rows of

$$\mathbf{W} = \begin{bmatrix} - & \mathbf{W}_0 & - \\ - & \mathbf{W}_1 & - \\ & \vdots & \\ - & \mathbf{W}_{|b|} & - \end{bmatrix} \tag{3.4}$$

to model the blendshapes. With this model, we can perform a simple matrix multiplication and addition to dynamically extract vertex locations for new blendshape configurations not represented in the preprocessed dataset.

3.3 Selecting Keypoints

Although we are provided with 8,769 vertices around the face, their geometric density varies significantly, and many of them are uninfluenced by changing blendshapes. Furthermore, using all of vertices directly can prove to be computationally expensive (and dilute the loss

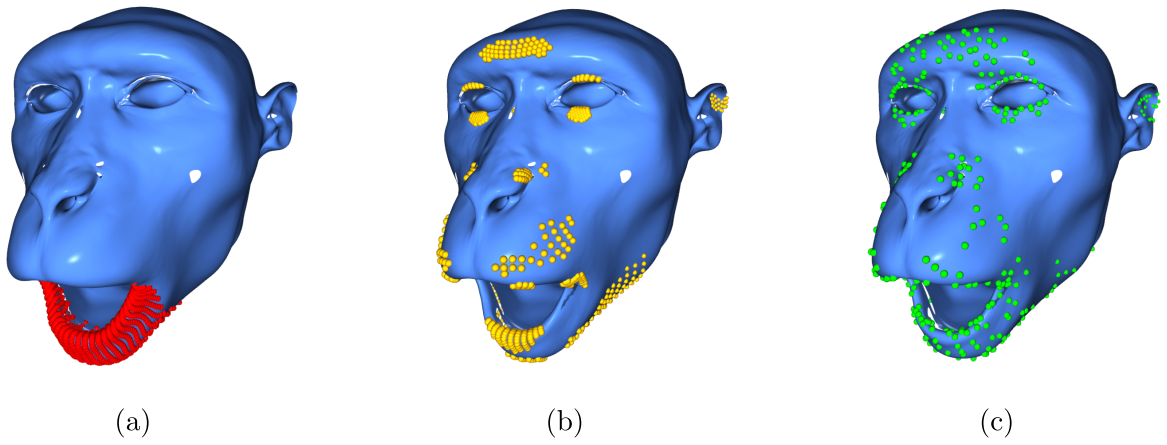


Figure 3.4: **Keypoint selection process**. Keypoints were selected using (a) top 1000 vertices across all blendshapes (b) top 100 vertices for each blendshape or (c) donwsampling and using top 50 vertices for each blendshape and downsampling again.

and gradients for any reprojection error we employ). To combat these issues, we establish several ways to downsample these vertices into keypoints as shown in Figure 3.4. Typically, face landmark detections are done with a standard set of 68 keypoints [31], but to represent a larger set of possible expressions, we select more. We want to ensure that the vertices that are geometrically transformed the most by the blendshapes are represented as the keypoints. Initially, we sampled the top 1000 vertices whose blendshape transformation norms are the largest across all blendshapes. This can be done with

$$\text{top1000}_i(\sum_{k=1}^{|\mathbf{b}|} ||\mathbf{W}_{ki}||) \tag{3.5}$$

However, this leads to selecting a dense set of vertices only on the jaw as shown in Figure 3.4a, ignoring the rest of the face. The second iteration involved using the top 100 vertices per blendshape:

$$\{\forall k : \text{top100}_i ||\mathbf{W}_{ki}||\} \tag{3.6}$$

While it better represents the face, within each blendshape, it still suffers from repeatedly oversampling the same moving segment of the face as seen in Figure 3.4b. The final iteration includes two uniform downsampling steps. First, the vertex list is downsampled by 5 times by selecting every 5th vertex. Then, for each blendshape, the top 50 vertices based on distance moved are selected and merged into a set. This new point cloud is downsampled once more using a voxel grid of 5 mm voxels, resulting in 354 selected vertices as keypoints. Figure 3.4c shows a much sparser point cloud which prevents over-redundant features while spanning the entire face and blendshape space.

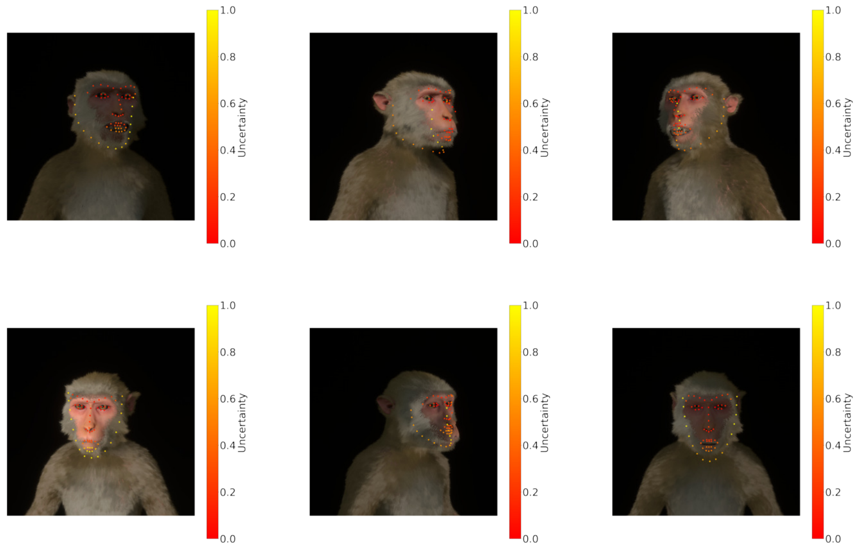


Figure 3.5: **Example landmark detections** on the avatar using a pretrained ResNet-50.

3.4 Training the Blendshape Predictor

In order to predict blendshapes from image frames, we retrain a ResNet-50 model that was pre-trained on predicting 68 human face landmarks. Given the similarities between humans and monkeys, the model already seemed to have an understanding of monkey faces and could accurately predict landmarks for some head poses and expressions as shown in Figure 3.5. To take advantage of the pre-trained model, we simply replace the final linear layer which had 204 outputs (68 xy coordinates with 68 confidence values), with 19 output values (17 blendshapes and 2 eye rotations) and a sigmoid nonlinearity to normalize values between 0 and 1.

Loss Function

The entire model is trained end-to-end with the following loss:

$$L(\mathbf{b}, \mathbf{e}) = \lambda_{\text{blend}} L_{\text{blend}} + \lambda_{\text{eyes}} L_{\text{eyes}} + \lambda_{\text{keypts2D}} L_{\text{keypts2D}} + \lambda_{\text{keypts3D}} L_{\text{keypts3D}} \tag{3.7}$$

where $\mathbf{b}$ and $\mathbf{e}$ represent the 17 blendshapes and 2 eye parameters to be predicted.

Blendshape Loss (L_{blend})

We use an L2 loss over the predicted and target blendshape values which are between 0 and 1.

$$L_{\text{blend}} = ||\mathbf{b} - \hat{\mathbf{b}}||_2 \tag{3.8}$$

Eyes Loss ($L_{\textbf{eyes}}$)

We also use the same loss function on the predicted and target eye parameters. Since eye rotations in the dataset are heavily constrained, we can regress these values directly without needing to use quaternions.

$$L_{\text{eyes}} = ||\mathbf{e} - \hat{\mathbf{e}}||_2 \tag{3.9}$$

Keypoints Reprojection Loss ($L_{\textbf{keypts2D}}$)

Several blendshapes can end up warping the same vertices and move them in the same direction. Strictly speaking, the blendshapes are not linearly independent, leading to multiple different blendshape configurations resulting in similar facial expressions. In addition, not all blendshapes lead to the same amount of physical change in the avatar. For example, the jaw is the most mobile part but is only moved using 3 of the 17 blendshapes. To inherently weigh some blendshapes more than others based on their visual impact and to regress to blendshape configurations that are visually accurate but not necessarily the exact target blendshape match, we introduce a keypoint reprojection loss. We take the predicted and target blendshapes and using our implementation of the linear skinning model determine the predicted and target 3D keypoint locations. These are then projected into the image plane using the camera parameters for the training example and a reprojection error is calculated.

$$\mathbf{V} = \mathbf{N} + \mathbf{b}\mathbf{W} \tag{3.10}$$

$$\hat{\mathbf{V}} = \mathbf{N} + \hat{\mathbf{b}}\mathbf{W} \tag{3.11}$$

$$L_{\text{keypts2D}} = ||\Pi X(\mathbf{V} - \hat{\mathbf{V}})||_2 \tag{3.12}$$

where $X = [R|t] \in \mathbb{R}^{3\times4}$ and $\Pi \in \mathbb{R}^{3\times3}$ representing extrinsic and intrinsic camera parameters.

Keypoints Euclidean Loss ($L_{\textbf{keypts3D}}$)

In addition to maintaining visual consistency by projecting points into the image plane, we also want to maintain geometric consistency to ensure no deformation along the principal axis of the camera. To account for this, we also include a 3D Euclidean distance error between the 3D keypoint locations generated using the linear skinning model with the predicted and target blendshapes.

$$L_{\text{keypts3D}} = ||\mathbf{V} - \hat{\mathbf{V}}||_2 \tag{3.13}$$

Weights

Due to the different units of these loss terms and their different contributions to the final visual output, we weigh these loss terms separately. Blendshapes and eyes values are scaled between 0 and 1, reprojection losses are in pixels, and Euclidean error is in meters in world space. Therefore, the configuration of weights are:

- **Blendshape Loss Weight** $\lambda_{\text{blend}} = 0.75$
- **Eye Loss Weight** $\lambda_{\text{eyes}} = 0.5$
- **Reprojection Loss Weight** $\lambda_{\text{keypts2D}} = 0.015$
- **Euclidean Loss Weight** $\lambda_{\text{keypts3D}} = 10$

Training Process

Data Augmentation

To add more variability to the training data, we use random transformations to the images. We randomly add affine transforms, perspective transforms, Gaussian blurs, color jitter, and grayscale. In addition, to address occlusions and inconsistent face crops, we add random boxes filled with noise onto the image and crop the image as well. Before using any data augmentations, we use the ground truth face bounding boxes to crop the images to just the monkey's faces. After all the augmentation, every image is resized to be 224×224.

Training Parameters

The training process ran for 250 epochs with both training and validation losses consistently decreasing the entire time. The AdamW optimizer was used with a learning rate of 0.0001 and a batch size of 16. It took 45 hours to train on a single NVIDIA GeForce RTX 3080.

3.5 Generating Realistic Data

While the avatar renders were effective at training the model to predict the blendshapes and eye rotations, the renders suffered from poor lighting, unrealistic fur and skin, and grainy hair follicles. Also, all the training renders were generated from a monkey with the same general visual appearance, which limits the trained model's generalizability. We harness the power of Stable Diffusion to get more realistic renders to train a more generalizable model.

ControlNet

Stable diffusion on its own provides little control over generation other than through text prompts. However, in our case, we want to generate images of monkeys that would follow

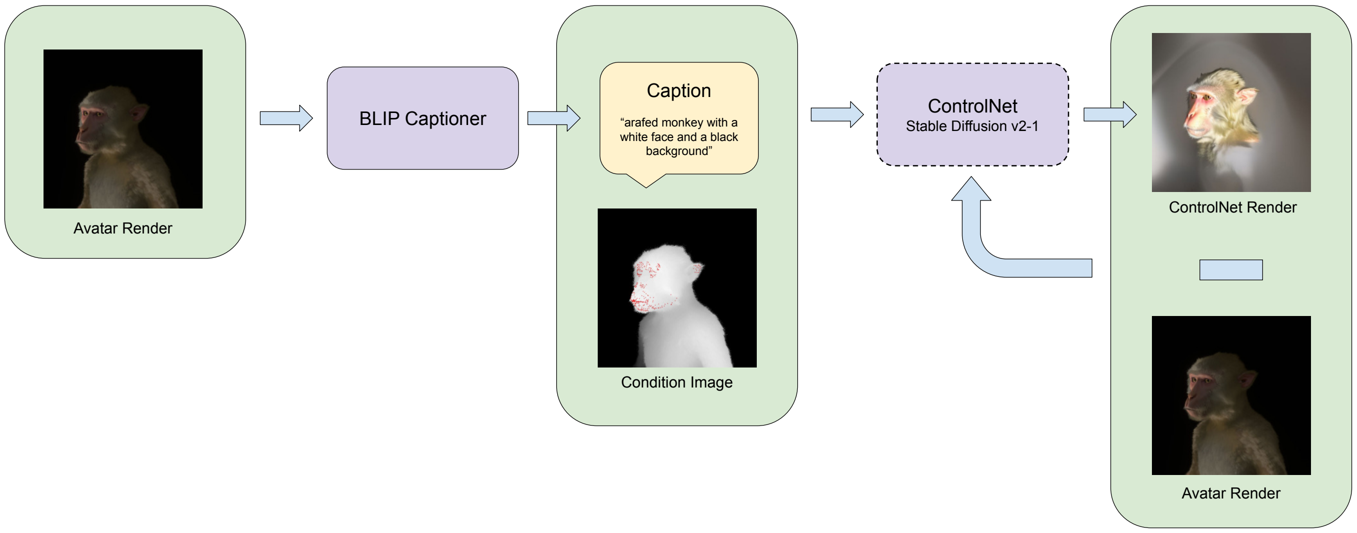


Figure 3.6: **ControlNet training process**. A condition image is formed using depth and keypoints and ControlNet is trained with BLIP captions and the true avatar render.

facial shape and structure given known blendshapes. ControlNet has been shown to provide that configurability using conditioning images in conjunction with text prompts. By adding a trainable copy of the stable diffusion model with zero convolutions, ControlNet can be finetuned to generate images given poses, Canny-edge detections, sketches, normal maps, and depth maps, among others.

We train ControlNet to use combined images using depth maps and keypoints as our conditioning images. To generate training data for ControlNet, we use the depth maps from the original synthetic avatar dataset and projected our 354 keypoints as red dots onto the image. While the depth map provides the general pose and shape of the face, the keypoints provide detailed control over particular moveable features on the face. The red dots are also scaled in intensity according to their depth in the image.

Figure 3.6 displays our training process for ControlNet. We use an out-of-the-box BLIP captioner [32] to generate text captions for all our training images. We train ControlNet using a batch size of 4 with an AdamW optimizer with default parameters for 3 epochs. Images are also resized to 512×512 to speed up training. Figure 3.7 shows some of the generated images using naive text prompts on validation conditioning images at various stages of the training process. Given the similar depth values at various parts of the avatar, the model tends to repeat faces multiple times in the image, particularly around the jaw. The model seems to rely on local context to determine what to generate instead of the global face. To address this problem we use a ControlNet checkpoint that was trained for 3 epochs, despite its overfitting. We also finetune our positive and negative prompts to ensure proper anatomy. However, ControlNet still struggles to add new backgrounds to the scene due to

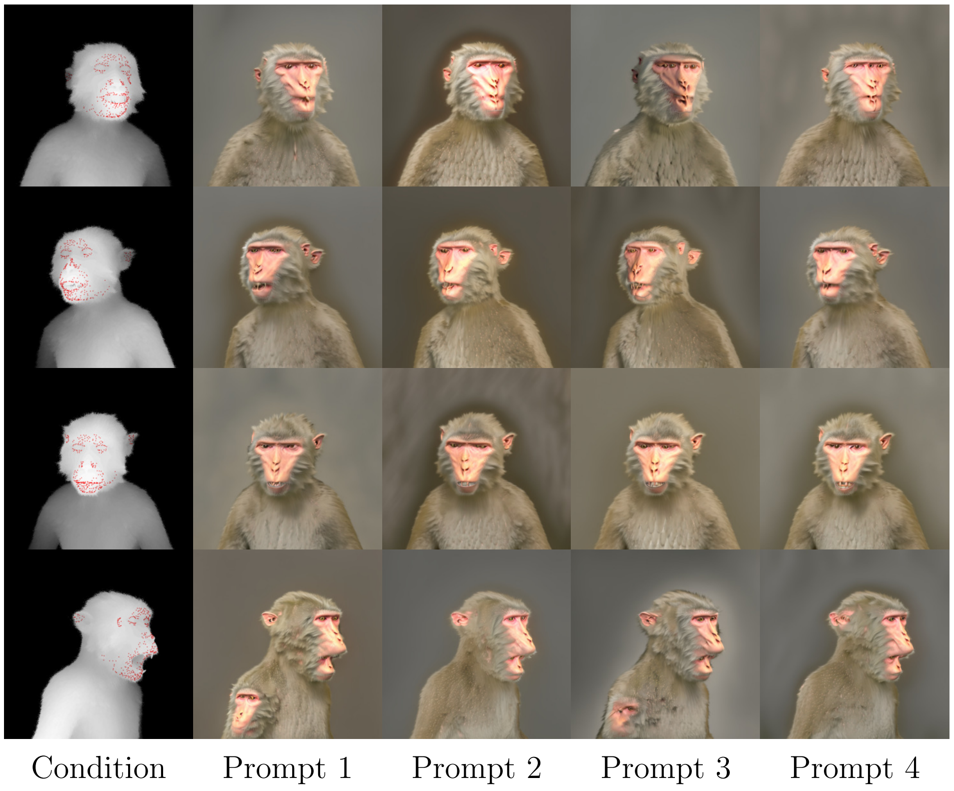


Figure 3.7: **ControlNet failures**. Shows different renders across a set of condition images using multiple prompts. Prompt 1:"High quality dslr photo of a macaque monkey". Prompt 2: "macaque monkey, dslr photograph, natural environment". Prompt 3: "High-quality photo of a monkey in a natural environment". Prompt 4: "macaque monkey". Images show multiple disfigured faces, while maintaining overall low-frequency shape.

the untextured depth in both the conditioning and training images.

Inpainting

To add backgrounds, we rely on an out of the box inpainting model. Figure 3.8 demonstrates the entire image generation pipeline. To allow for more flexibility of the rest of the body and background, we generate a mask using the depth map cropped based on the bounding box of the face. The text prompt is randomly selected to choose between natural backgrounds, human environments, and headgear that would be found in lab environments. We can see a comparison of the original avatar renders with the new realistic renders that maintain facial expressions in Figure 3.9. Figure 3.10 shows 30 random renders from this new stable diffusion-generated realistic dataset.

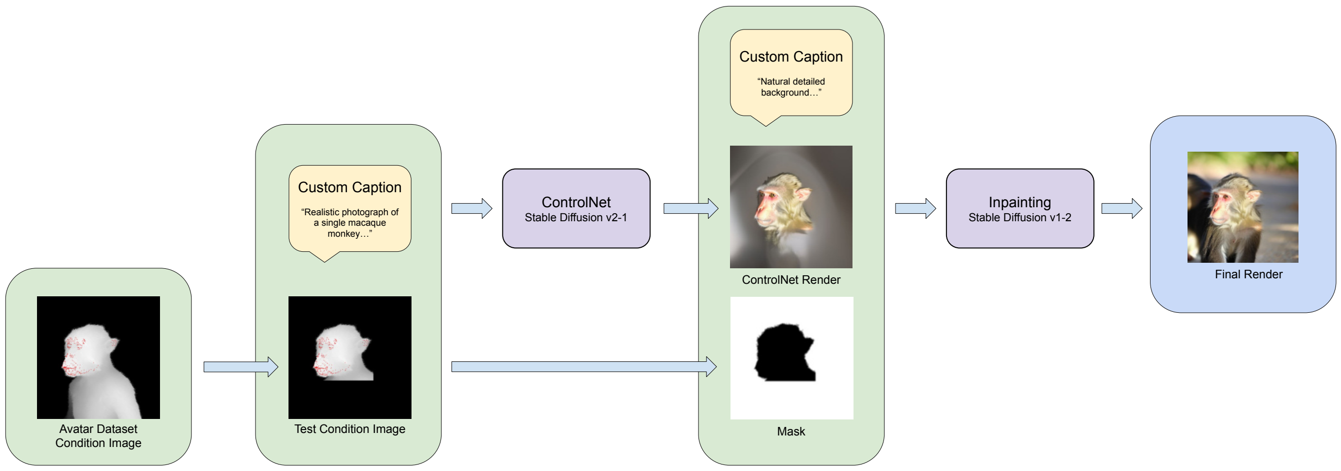


Figure 3.8: **Stable Diffusion image generation**. Condition images are used to create new monkey face renders which are then combined with inpainting to add background context.

3.6 Finetune Blendshape Predictor

We finetune our ResNet-50 using our generated realistic data to help it generalize to new backgrounds, lighting, and texture. We create a dataset of 80,000 images that contains an even mix of images from the original avatar dataset and realistic image dataset, to make sure that the model continues to train using both image domains and learn a more generalizable representation. Since ControlNet is not trained to be conditioned on eye poses, we set the eye loss weight (λ_{eyes}) to 0. In addition, the original model seems to do more poorly with certain jaw blendshapes compared to other blendshapes. To address that, we multiplied the losses for those particular blendshapes by 2. We start from the final checkpoint of the previously trained model, and the rest of the training process remains identical.

3.7 Face Detector

Before running on videos, we need a way to crop the monkey's face so it can be passed to the blendshape prediction model. RetinaFace is good at detecting monkey faces even though it is trained for human faces; however, it struggles to detect some faces, especially with side profiles or occlusions. Its detections also vary in how tightly the bounding boxes encapsulate faces, sometimes cutting off the jaw when the mouth is open. Since our blendshape regression model uses tight ground truth bounding boxes as a preprocessing step, it is important to ensure that RetinaFace provides consistently accurate boxes as well.

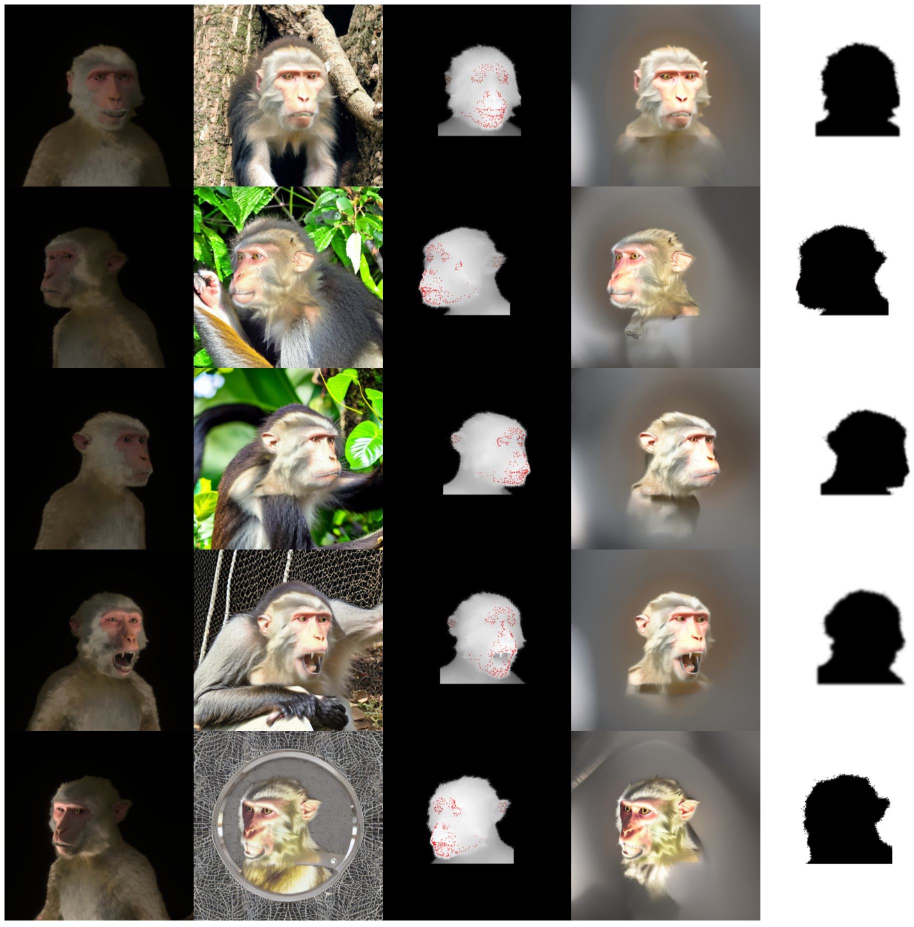


Figure 3.9: **Stable Diffusion realistic image generation pipeline and comparisons**. We can compare the original avatar render with the final Stable Diffusion generated realistic image which maintains pose and expression. We add the condition image that was used for ControlNet, the ControlNet render, and the mask used for inpainting.

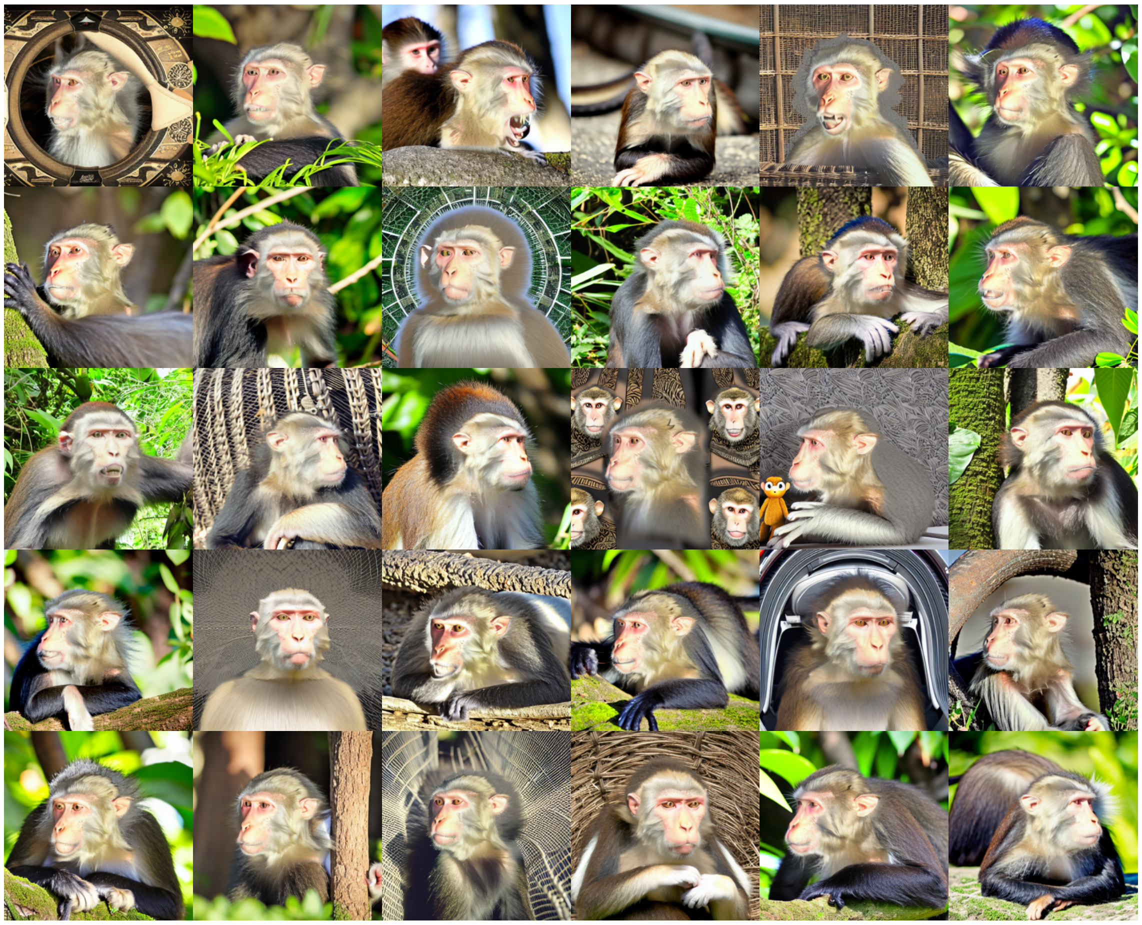


Figure 3.10: **Random subset of 30 Stable Diffusion realistic images**. Each image displayed has a corresponding avatar image using which it was generated.

Training RetinaFace

To train RetinaFace, we use our newly generated realistic images with ground-truth bounding boxes that are determined using the corresponding avatar render. RetinaFace was originally trained using a multi-task loss for classification, bounding box, keypoints, and a dense face model regression. For our purposes, we train using only classification, bounding box, and keypoints since we do not have a differentiable renderer for our avatar. Using the avatar's mesh, we selected 5 keypoints (right eye, left eye, nose, right corner of mouth, and left corner of mouth), which are used for the keypoint loss as extra supervision. To add data for classification (face versus not a face), we add two extra random bounding boxes per image with negative labels. Only two boxes are added to prevent class imbalance since each image only contains one face. For these boxes, we ensure that their IoU with the ground truth box is greater than 0.25 but less than 0.75. This ensures that we are providing valuable feedback to the model by giving it examples that overlap the ground truth but have poor precision. To

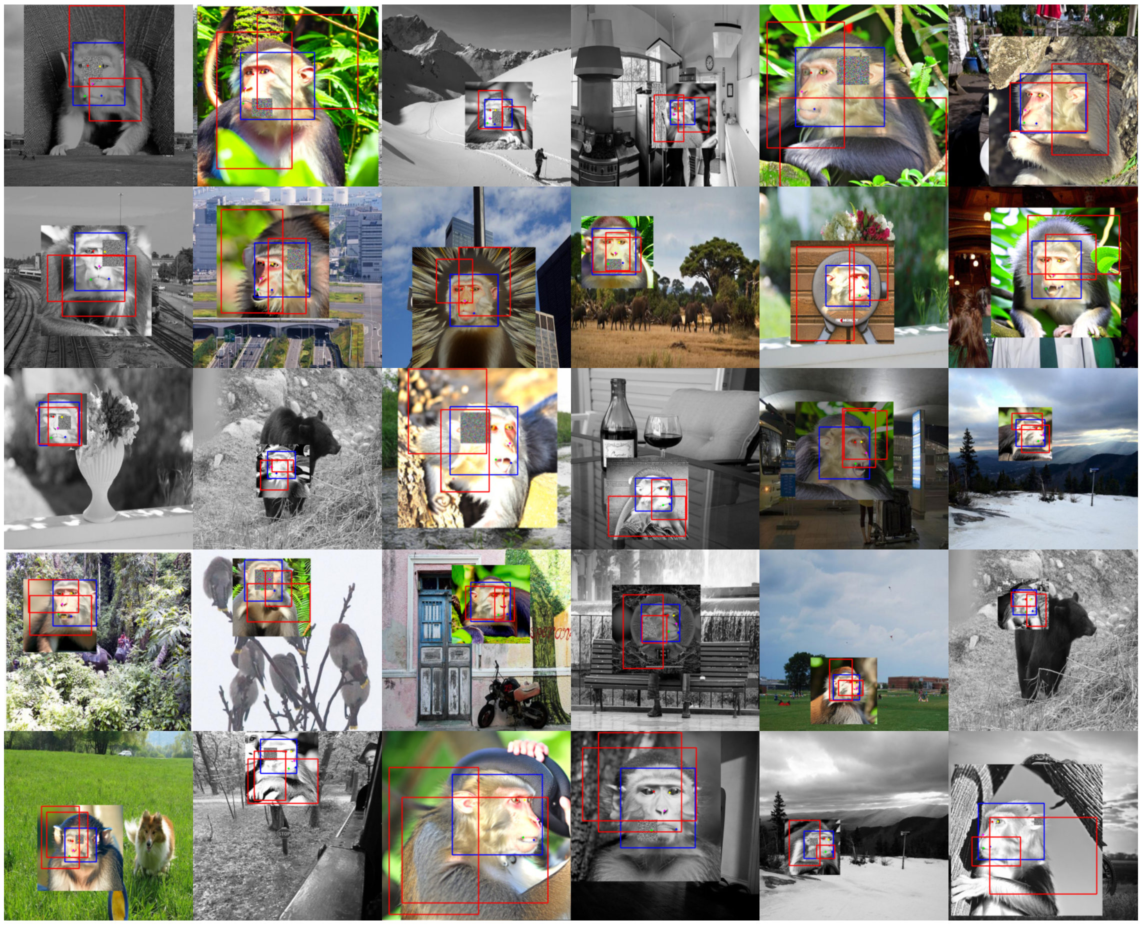


Figure 3.11: **Training data for RetinaFace**. Resized Stable Diffusion renders pasted onto background images, with ground truth boxes and keypoints in blue and synthetically added negative detections in red.

account for different-sized faces in the image, the stable diffusion renders are downsampled randomly to be 1 to 7 times smaller than their original size. These downsampled images are then pasted onto random locations on a larger 512 × 512 background image selected from the COCO dataset [33]. Additional data augmentation involves cropping, blurring, color adjustments, and mirroring. Figure 3.11 shows example images with bounding boxes and keypoints around the monkey's face. Blue bounding boxes indicate ground truth, while red boxes show artificially added false detections.

3.8 Overall Pipeline

Figure 1.1 demonstrates the overall pipeline that this paper implements, with the dotted section discussed in the scope of this paper. First, individual image frames from a video (live or offline) are processed using RetinaFace to detect and crop monkey faces. Then, using our ResNet-50 blendshape regression model, we predict avatar blendshapes for the given image. Although these predictions are done independently frame-by-frame, they are mostly temporally consistent. However, we do sometimes include an optional smoothing step to remove extra jitteriness. We employ a simple exponential moving average (EMA) on blendshapes directly:

$$\mathbf{b}^{0}_{\text{smooth}} = \mathbf{b}^{0}_{\text{predict}} \tag{3.14}$$

$$\mathbf{b}^{t}_{\text{smooth}} = \alpha \mathbf{b}^{t}_{\text{predict}} + (1-\alpha)\mathbf{b}^{t-1}_{\text{smooth}}, \qquad t > 0 \tag{3.15}$$

Although this paper does not implement any of the downstream tasks, these predicted blendshapes can be used in various applications. Time-synced recorded neural signals can be used to draw correlations with facial expressions. We can also run experiments that simultaneously capture the blendshapes of two monkeys interacting with each other to build a predictive interaction model.

Chapter 4

Results

4.1 Face Detector

To test the accuracy of the monkey face detector, we used the test set of both the avatar dataset and stable diffusion generated dataset, since we did not have any real ground truth real images. In Figure 4.1, average precision is measured with AP_{50} and AP_{75} with true positive predictions being marked with IoU = 0.5 and = 0.75, respectively. AP is AP[0.5:0.95:0.05], which is the average AP for all IoU from 0.5 to 0.95 at 0.05 intervals, derived from COCO evaluation metrics [33, 34]. AP_S is for faces with area < 322 px^2, AP_M is for $322 <$ area < 962 px^2, and AP_L is for area > 962 px^2. The fine-tuned model was trained only on the stable diffusion realistic dataset. The table shows that the fine-tuned model outperforms the original human face detector in every metric for both datasets. However, it does not perform as well on the avatar dataset as it does on the Stable Diffusion dataset. This indicates overfitting and exemplifies the lack of visual diversity even in the stable diffusion dataset.

Figure 4.2 shows pairs of detections of the original model (left) and the fine-tuned model (right) on real videos. It is noticeable that the fine-tuned model provides tighter bounding

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Avatar Dataset						
RetinaFace Original	27.8	69.0	13.1	17.8	27.7	33.3
RetinaFace Finetuned	**84.9**	**99.4**	**94.9**	**81.8**	**84.2**	**88.8**
Stable Diffusion Dataset						
RetinaFace Original	18.3	40.9	11.7	8.2	17.8	25.0
RetinaFace Finetuned	**98.2**	**99.9**	**99.9**	**94.6**	**98.2**	**99.7**

Figure 4.1: **Average Precision values** comparing the original RetinaFace and our fine-tuned model.

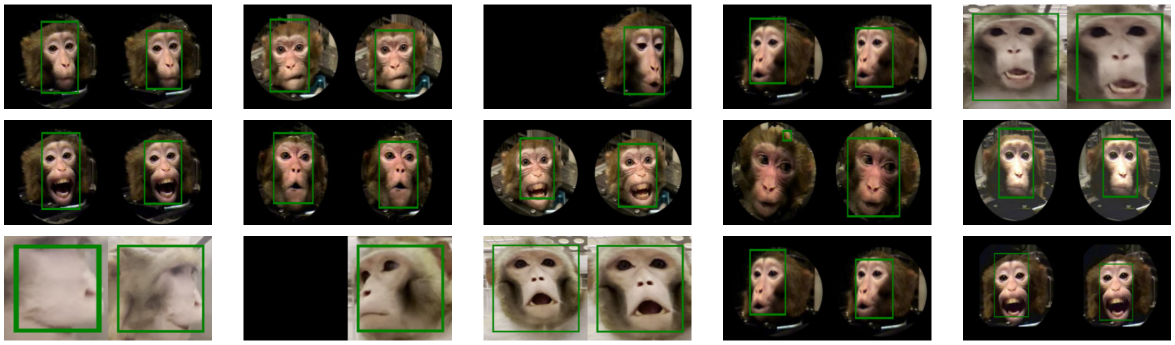


Figure 4.2: **Examples where the finetuned RetinaFace performs better** than the original RetinaFace model. The left image shows the original model, while the right image shows the finetuned model. Images that are black indicate no detection.

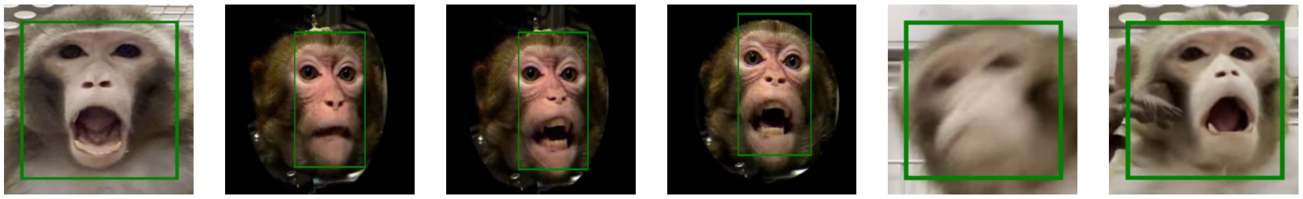


Figure 4.3: **Examples where the finetuned RetinaFace fails to detect** faces that the original RetinaFace model detects. These images show the detections from the original RetinaFace model.

boxes, particularly at the top and bottom of the faces. It also does a better job of handling profile views, which the original model sometimes misses completely. We attribute this to the fact that human and monkey faces look more similar from the front than from the side.

Sometimes, the fine-tuned model does worse and completely misses detections as shown in Figure 4.3. These are all detections found by the original model, but missed by the fine-tuned model. These examples demonstrate the overfitting of the fine-tuned model, as it seems to miss some of the more exaggerated expressions that may not have been present in the dataset. The original model seems to generalize better to unseen examples by simply detecting 2 eyes, a nose, and a mouth, no matter the shape, and therefore it can handle these cases. This could also explain why it could successfully detect the face in the example with significant motion blur, by focusing on the low-frequency information in the image.

It is also worth noting that the fine-tuned model can handle several special difficult cases due to using data augmentation in the training process. Figure 4.4 shows examples of smaller scales of motion blur, varying camera auto-focus, and occlusions from food or hands. The fine-tuned RetinaFace model still successfully detects faces here.

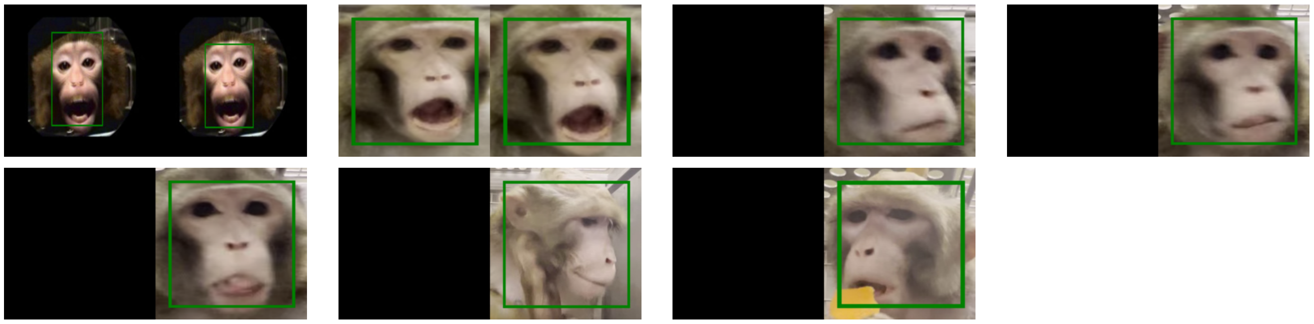


Figure 4.4: **Examples where the finetuned RetinaFace handles difficult scenarios** that the original RetinaFace model struggles with. The left image shows the original model, while the right image shows the finetuned model. Images that are black indicate no detection.

4.2 Blendshape Predictor

To quantitatively evaluate the blendshape predictor, we ran it on the test sets of the avatar dataset and the Stable Diffusion dataset independently. Using these predictions, we compared the per-blendshape errors of both models, as well as per-vertex Euclidean errors. Figure 4.5 visualizes the per-blendshape errors on the test set of the avatar dataset across both models in histograms. The model trained on just avatar data is shown in blue, while the same model fine-tuned using Stable Diffusion renders is shown in orange. Surprisingly, despite the original model primarily being trained on this avatar dataset, the fine-tuned model slightly outperforms it on every blendshape, except for some of the smaller motion blendshapes such as eye-blink. It is also worth mentioning that the differences in target and predicted blendshapes are primarily close to 0 with almost all test images having < 0.2 absolute error for every blendshape.

We can also look at the results on the Stable Diffusion test set as shown in Figure 4.6. The difference in errors is much more pronounced on this dataset compared to the difference in errors between both models on the avatar dataset. As expected, the model trained using the Stable Diffusion renders suffers from less overall loss. However, it seems to be more inaccurate with features such as lips and eyes. These features were particularly difficult for Stable Diffusion to render properly while following the conditioning image. These inaccuracies in the Stable Diffusion dataset may explain why the model could have learned incorrect representations for these features. Additionally, the target blendshapes corresponding to these Stable Diffusion renders would not be completely accurate due to the imperfect renders generated, compared to the ground truth. These inaccuracies would explain why the blendshape errors span ± 0.4 unlike on the avatar dataset where errors span ± 0.2.

We can also look at per-vertex errors as shown on the heatmaps in Figure 4.7. Overall, most of the error is primarily localized around the mouth area, as indicated by the lighter colors. Figures 4.7a and 4.7b show the results on the avatar dataset for both the original

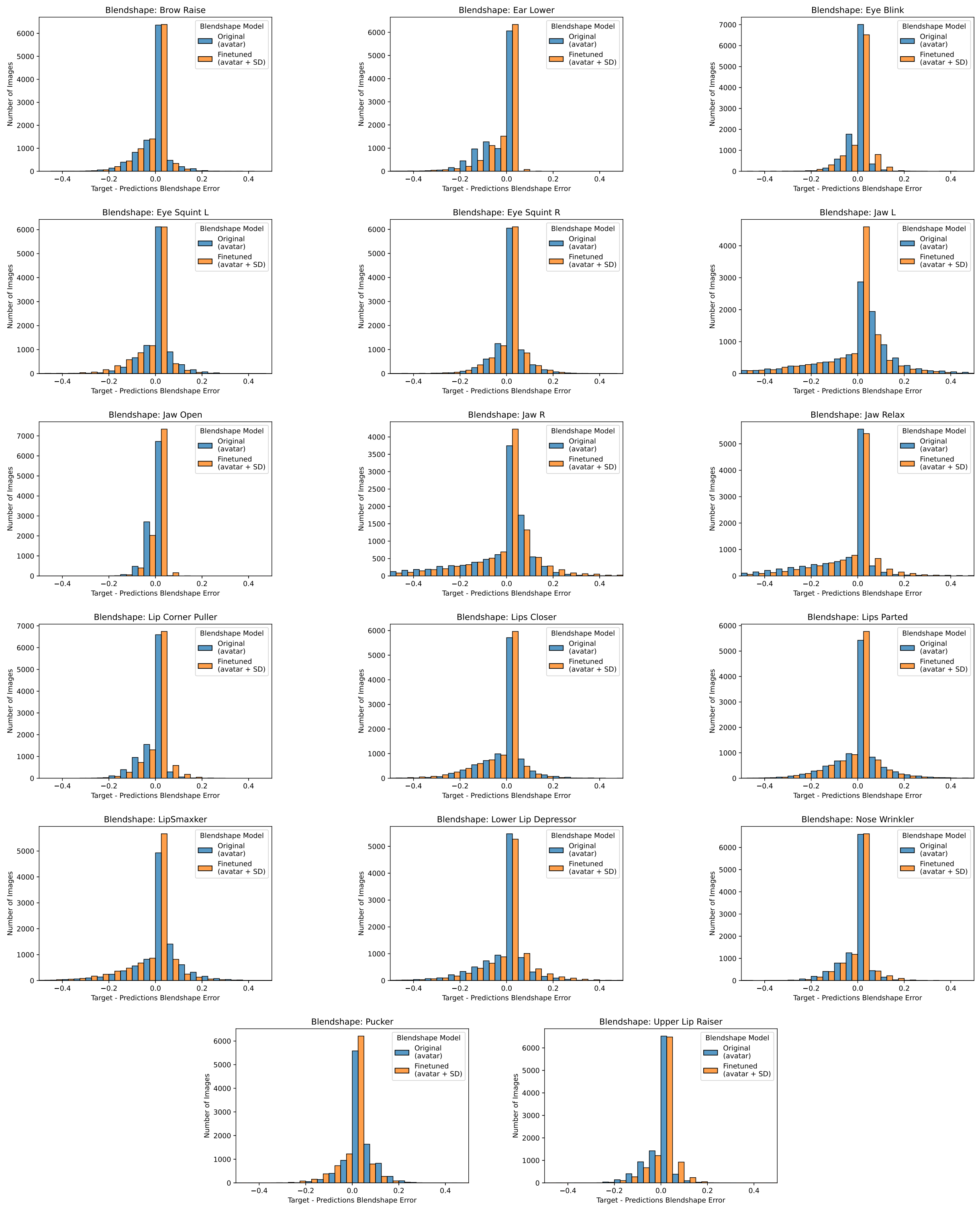


Figure 4.5: **Per blendshape errors on avatar dataset**. Compares the original trained (on avatar renders) ResNet-50 that predicts blendshapes with the updated model (trained on avatar renders and stable diffusion (SD) renders). Shows errors when run on the test set of the avatar render dataset.

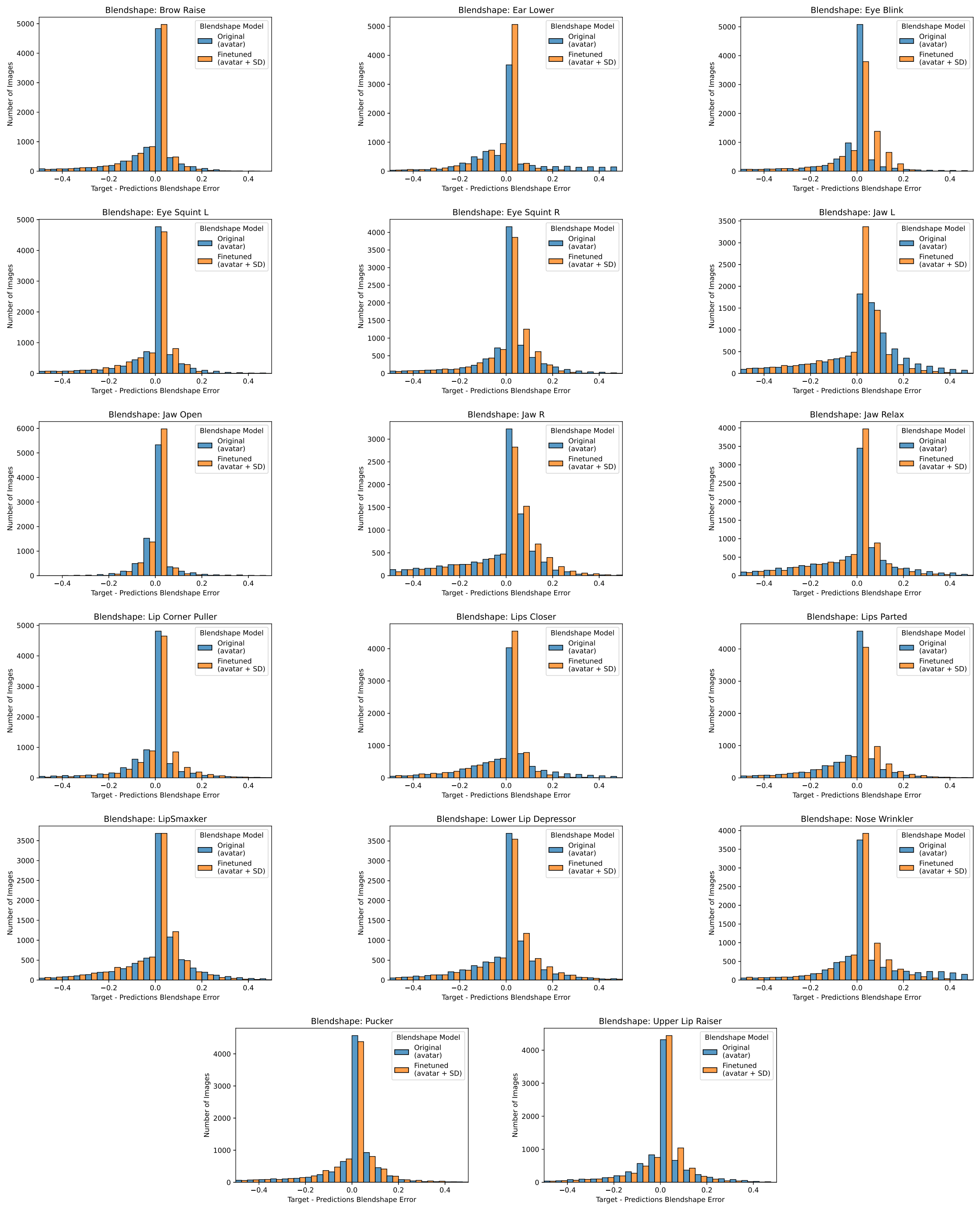
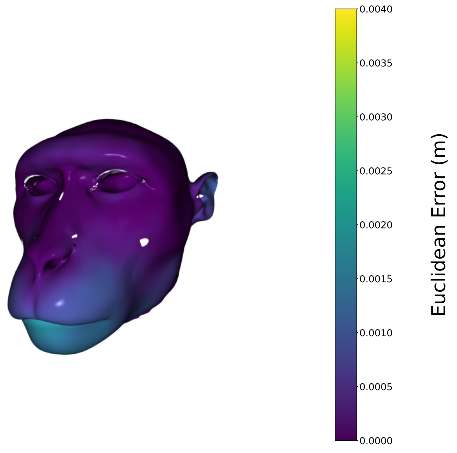
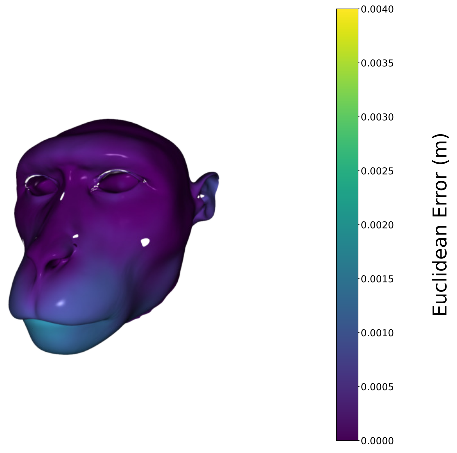


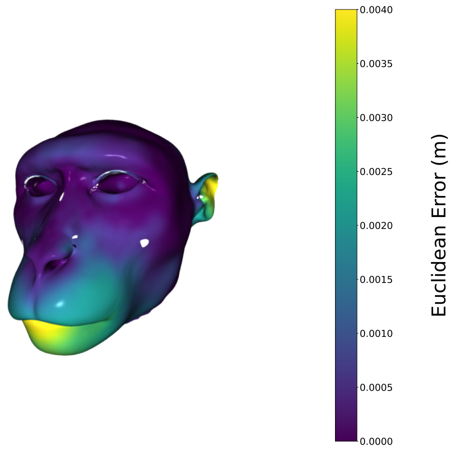
Figure 4.6: **Per blendshape errors on the Stable Diffusion dataset**. Compares the original trained (on avatar renders) ResNet-50 that predicts blendshapes with the updated model (trained on avatar renders and stable diffusion renders). Shows errors when run on the test set of the stable diffusion rendered dataset.



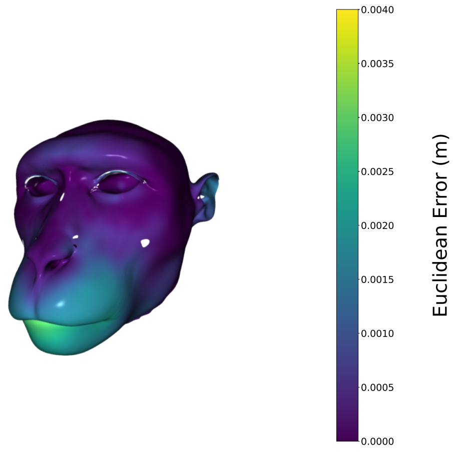
(a)



(b)



(c)



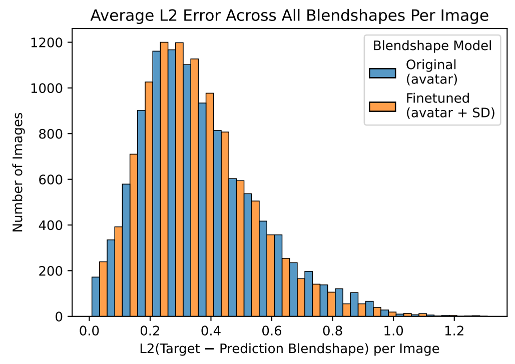
(d)

Figure 4.7: **Euclidean vertex errors**. Displays the different average per vertex errors across both models and both datasets. On the avatar dataset we visualize the vertex errors for the (a) original model trained only on avatar renders and (b) the finetuned model trained on stable diffusion renders. On the stable diffusion dataset, we visualize the error heatmaps for the (c) original model and (d) the finetuned model.

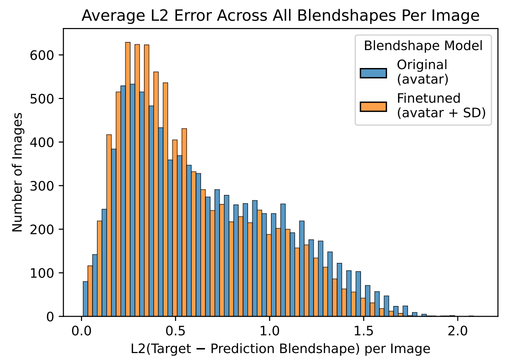
model and the fine-tuned model, respectively. Both look almost identical, but the fine-tuned model is slightly better (has a darker color). Figures 4.7c and 4.7d show the results on the Stable Diffusion dataset. While both models perform much worse on this dataset than on the avatar dataset, the fine-tuned model performs significantly better than the original model, indicating that it learned a slightly more generic representation instead of overfitting to the original avatar dataset.

We can also compare overall error using L2 error across all blendshapes and the average vertex error as shown in Figure 4.8. In Figures 4.8a and 4.8b, the fine-tuned model performs better on both avatar and stable diffusion datasets. While the range of L2 errors across all blendshapes remains the same, the distribution is much more skewed towards 0. While this is extremely evident in the stable diffusion dataset, it also presents itself in the avatar dataset. Likewise, vertex errors also present a similar story in Figures 4.8c and 4.8d. Both follow a similar pattern, where the fine-tuned model that was trained using both avatar and stable diffusion data performed better on both datasets individually.

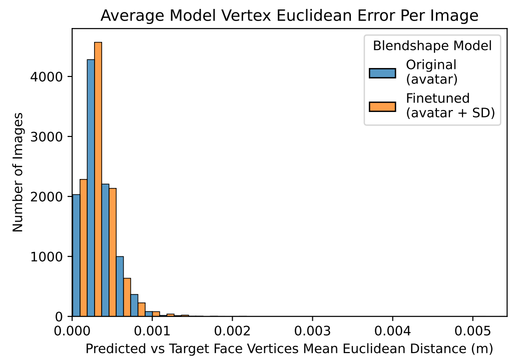
Figures 4.9, 4.10, 4.11, 4.12, and 4.13 show the avatar animated using the predicted blendshapes from real videos. Despite training on purely synthetic data, the avatar quite accurately mimics the monkey in the videos. The jaw opens and closes appropriately. We can also see the top and bottom lips moving to cover parts of the gums and teeth whenever the real monkey has its mouth open but hides its teeth. While difficult to recognize in static images, the avatar can mimic more subtle movements around the ears and smaller chewing motions of the jaw. There also are some key differences: the avatar is not able to change the overall mouth shape to exactly match the real monkey, the avatar is not controllable to hide its teeth completely when the mouth is open, and the avatar's eyes look vastly different than any of the real monkey images. These shortcomings have led to limits in both the generated training data as well as the reconstructed avatar renders using the predicted blendshapes, leaving room for improvement.



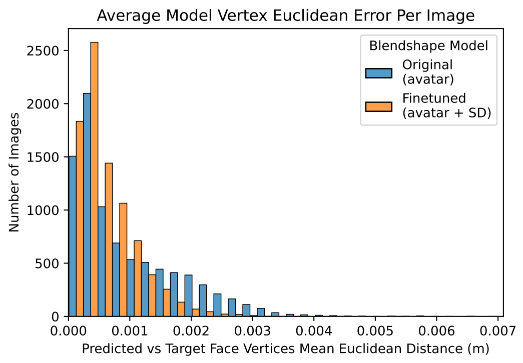
(a) Blendshape Error on Avatar Dataset



(b) Blendshape Error on Realistic Dataset



(c) Vertex Error on Avatar Dataset



(d) Vertex Error on Realistic Dataset

Figure 4.8: **Overall error comparison between models** trained on just avatar renders vs. avatar renders and Stable Diffusion (SD) renders: L2(Target - Predicted blendshapes) across all blendshapes for both models run on (a) the avatar dataset and (b) the stable diffusion (realistic) dataset. Predicted vs. Target Vertex Mean Euclidean Distance for both models run on (c) the avatar dataset (d) the stable diffusion (realistic) dataset.

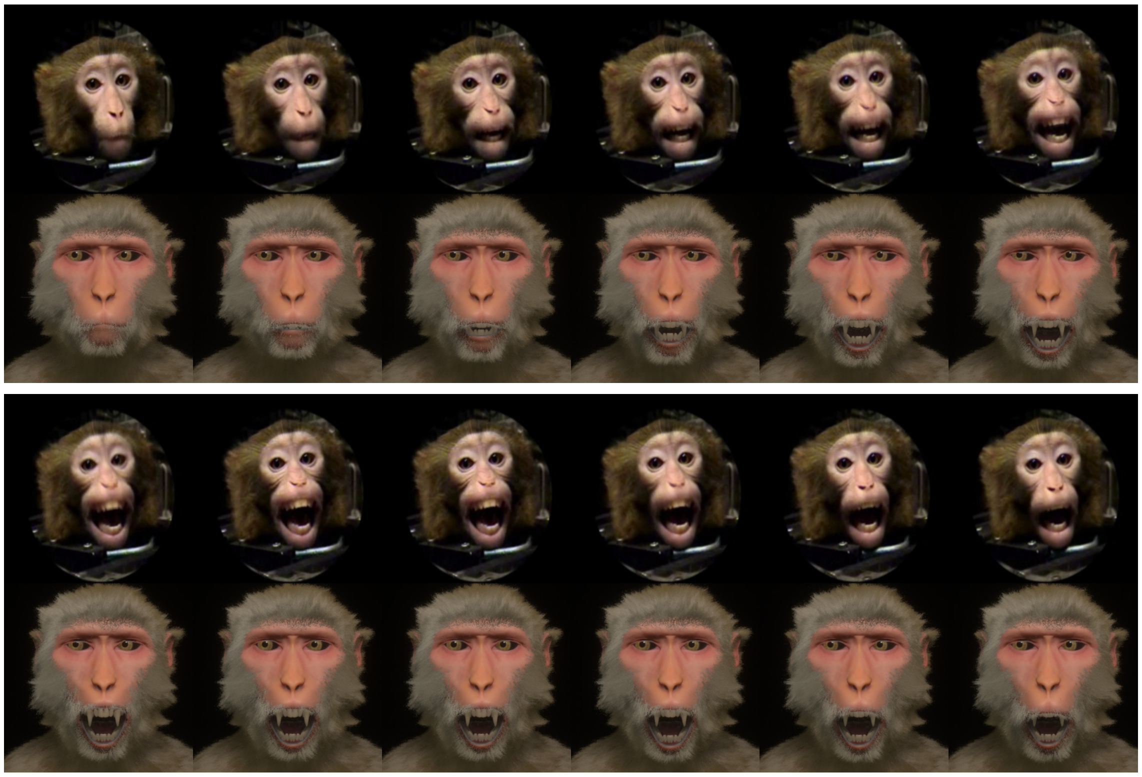


Figure 4.9: **Example clip 1 of a real monkey and the corresponding animated avatar** rendered according to the blendshapes predicted by the final model.

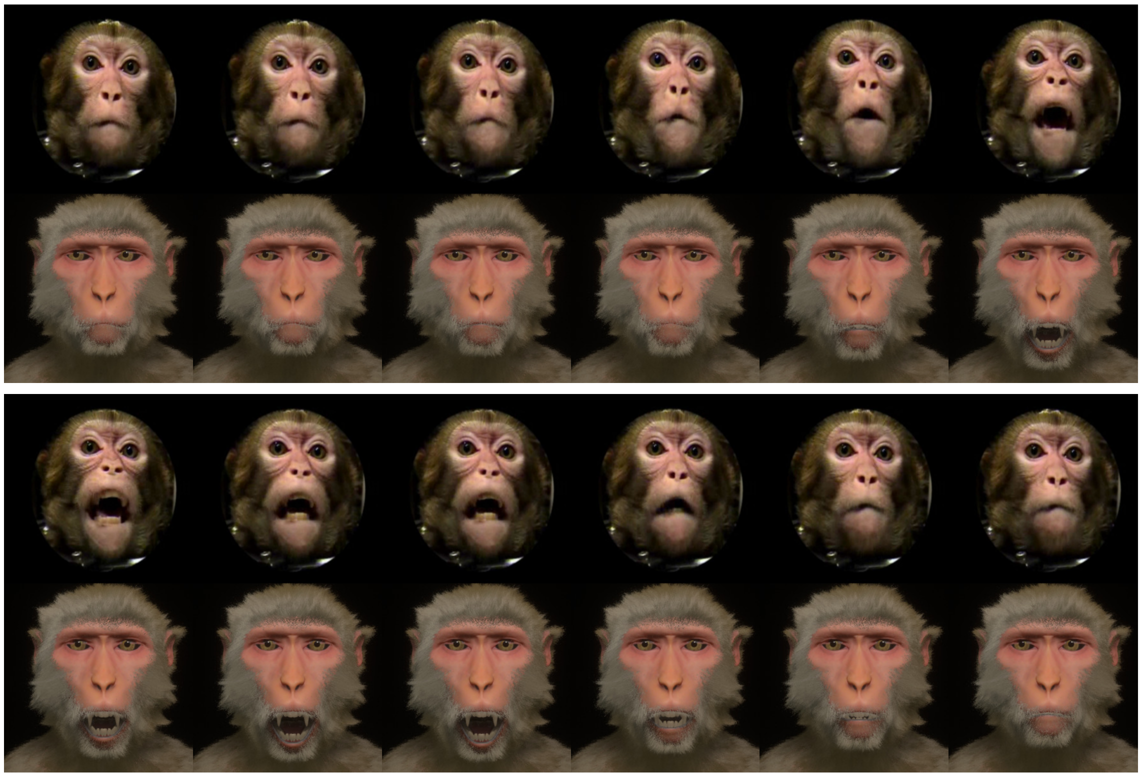


Figure 4.10: **Example clip 2 of a real monkey and the corresponding animated avatar** rendered according to the blendshapes predicted by the final model.

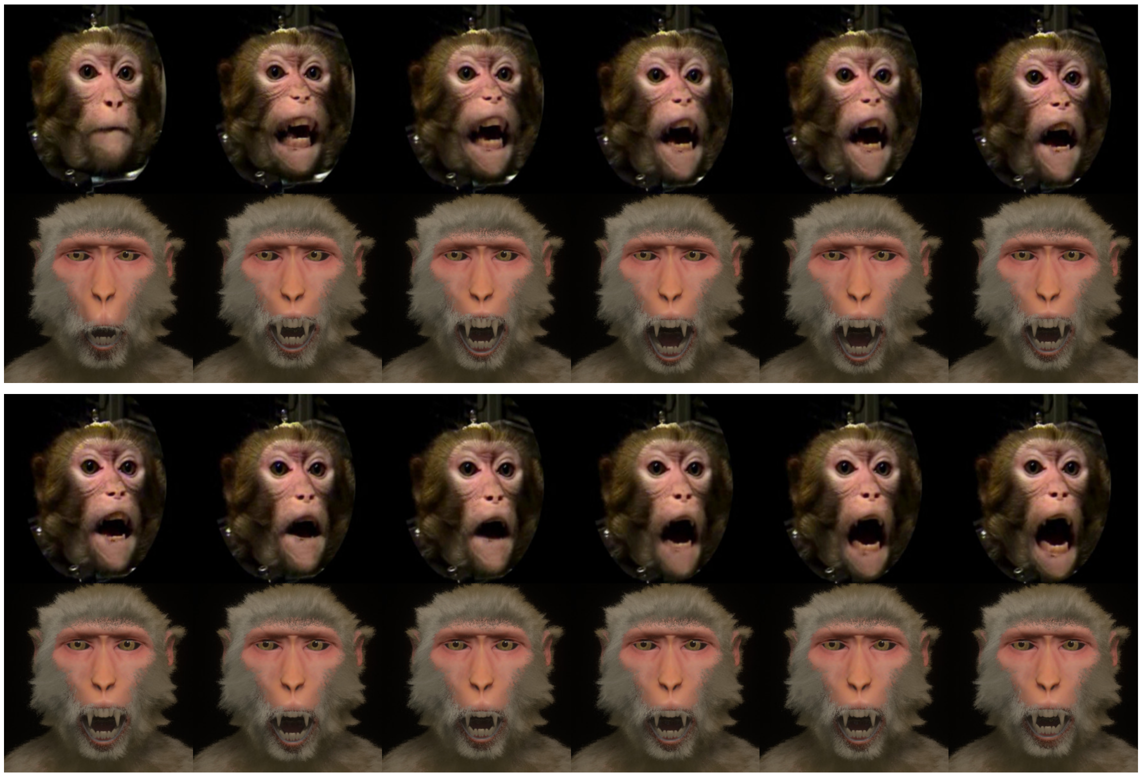


Figure 4.11: **Example clip 3 of a real monkey and the corresponding animated avatar** rendered according to the blendshapes predicted by the final model.

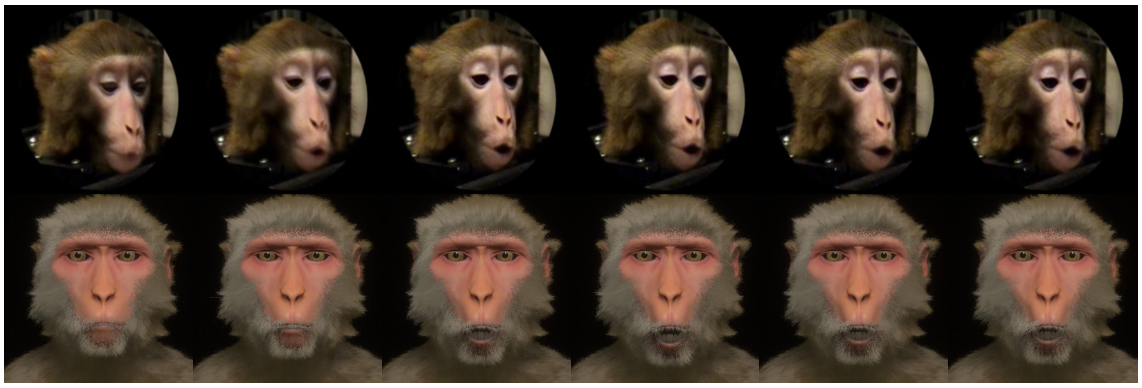


Figure 4.12: **Example clip 4 of a real monkey and the corresponding animated avatar** rendered according to the blendshapes predicted by the final model.

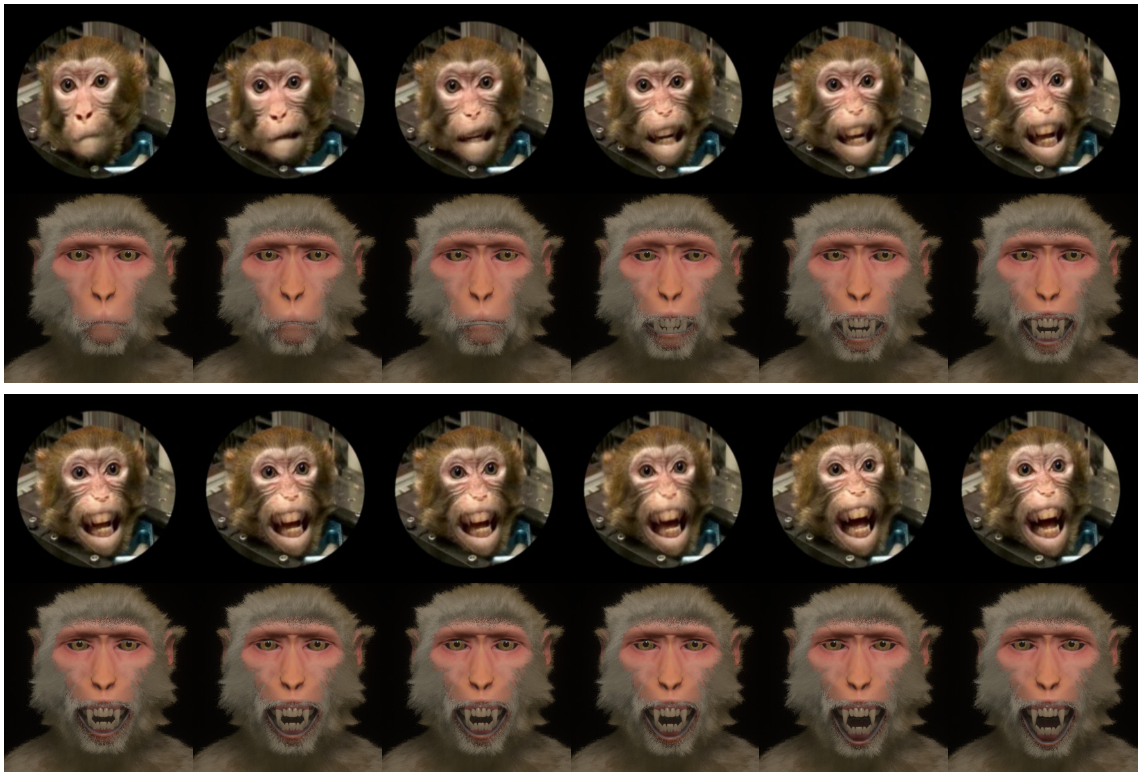


Figure 4.13: **Example clip 5 of a real monkey and the corresponding animated avatar** rendered according to the blendshapes predicted by the final model.

Chapter 5

Discussion

We propose a way of detecting monkey faces and predicting blendshapes for a custom avatar (LBS model) using only synthetic data. Our results demonstrate the effectiveness of using ground truth images generated using the avatar through low blendshape errors and Euclidean vertex errors. Furthermore, using Stable Diffusion and ControlNet on the original avatar renders to create more realistic data has proven to help the model generalize to more appearances and even boosts its performance on the original avatar dataset.

However, it is important to note that the ControlNet renders sometimes struggle to portray all expressions effectively. This is especially true for smaller movements like nose scrunching and side-to-side jaw motions. Stable Diffusion itself also struggles with some basic anatomy such as eyes, eyelids, lips, and teeth which has led to slightly inaccurate ground-truth labels for the Stable Diffusion dataset. This, in turn, does negatively impact some of the model's training and errors. However, the finetuned model still performs better on both datasets, indicating its usefulness for fostering generalizability.

Most systems that perform model registration rely on depth, landmark detections, facial motion capture with markers, or hardcoded muscle activation retargeting. Our system relies on a simple forward model that goes from raw images to predicted blendshapes directly. This removes the manual work of retargeting muscles or needing to add motion capture markers. It also reduces the technical requirements of a setup. Depth is not required, the additional preprocessing step of predicting landmarks can be skipped, and an expensive motion capture system can be replaced with a single camera. This is made possible by the quality of the synthetic datasets generated using a bare bone avatar and Stable Diffusion. That process also requires no manual labeling and is a purely automated procedure that synthesizes high-quality ground-truth data.

In the future, it would be useful to explore adjusting the avatar to include different identities which would allow using a more geometric approach for model registration, since it will be able to physically match more monkey faces. Exploring a wider variety of blendshapes to capture a broader range of facial expressions observed in real-life scenarios would be beneficial in recreating more nuanced expressions. Creating a differentiable renderer for the avatar could also be useful to match appearance and create new appearance-based loss terms to the training process. We can also augment the model to predict head poses in addition to blendshapes. Finally, we would like to explore using other generative models such as CycleGAN [35] that can be used to more accurately convert the avatar renders we generate to images that match the domain of our real-life scenarios. Stable Diffusion hallucinates what the target domain looks like using text prompts, but a visual similarity based approach controlled with cycle-consistency may prove to be more effective at both maintaining the same facial expression and matching the target domain.

Chapter 6

Conclusion

In this paper, we introduce a novel method for monkey face detection and blendshape prediction for avatar registration using synthetic data. Leveraging Stable Diffusion and ControlNet, our approach demonstrates promising results in simplifying the avatar registration process and enhancing our blendshape predictor's generalization.

We have gained valuable insights into the challenges and opportunities in avatar modeling especially in situations where data is scarce. While our approach exhibits effectiveness, challenges such as accurately portraying certain expressions highlight areas for further improvement.

Despite these challenges, our research contributes to the advancement model registration techniques and neuroscience by offering a more accessible and cost-effective approach to markerless motion capture and face reconstruction. By directly predicting blendshapes from raw images, we eliminate the need for complex preprocessing steps and costly equipment, making avatar animation more accessible to a broader audience.

Looking ahead, our research opens up exciting possibilities in various domains, including neuroscience experiments involving monkeys. By providing a more accessible and cost-effective approach to avatar modeling and real-time animation, we enable neuroscientists to run experiments that aim to understand social interactions in monkeys. Our method for detecting and tracking facial expressions offers a valuable tool for researchers, facilitating the study of complex social behaviors and emotional responses.

In conclusion, our research introduces a new technique of animating avatars trained solely on synthetic data generated from the avatar itself and using Stable Diffusion to augment it. By addressing current challenges and exploring future research directions, we can continue to push the boundaries of markerless face retargeting even for nonhuman faces, ultimately enabling more immersive virtual experiences and facilitating work in diverse fields.

References

[1] D H Hubel and T N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.*, 148(3):574–591, October 1959.

[2] Masahiro Mori, Karl MacDorman, and Norri Kageki. The uncanny valley [from the field]. *IEEE Robot. Autom. Mag.*, 19(2):98–100, June 2012.

[3] Ramona Siebert, Nick Taubert, Silvia Spadacenta, Peter W Dicke, Martin A Giese, and Peter Thier. A naturalistic dynamic monkey head avatar elicits species-typical reactions and overcomes the uncanny valley. *eNeuro*, 7(4):ENEURO.0524–19.2020, July 2020.

[4] Sarah B Carp, Anthony C Santistevan, Christopher J Machado, Alexander M Whitaker, Brittany L Aguilar, and Eliza Bliss-Moreau. Monkey visual attention does not fall into the uncanny valley. *Sci. Rep.*, 12(1):11760, July 2022.

[5] Aidan P. Murphy and David A. Leopold. A parameterized digital 3d model of the rhesus macaque face for investigating the visual processing of social cues. *Journal of Neuroscience Methods*, 324:108309, 2019. ISSN 0165-0270. doi: https://doi.org/10.1016/j.jneumeth.2019.06.001. URL https://www.sciencedirect.com/science/article/pii/S0165027019301591.

[6] Xin-He Liu, Lu Gan, Zhi-Ting Zhang, Pan-Ke Yu, and Ji Dai. Probing the processing of facial expressions in monkeys via time perception and eye tracking. *Zool. Res.*, 44(5):882–893, September 2023.

[7] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[8] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. 2023.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. 2015.

[10] Greg Welch, Gary Bishop, Leandra Vicci, Stephen Brumback, and Kurtis Keller. High-performance wide-area optical tracking. 05 2002.

[11] P Viola and M Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* IEEE Comput. Soc, 2005.

[12] Jiankang Deng, Jia Guo, Yuxiang Zhou, Jinke Yu, Irene Kotsia, and Stefanos Zafeiriou. Retinaface: Single-stage dense face localisation in the wild. *CoRR*, abs/1905.00641, 2019. URL http://arxiv.org/abs/1905.00641.

[13] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Wider face: A face detection

benchmark. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[14] Bernhard Egger, William A. P. Smith, Ayush Tewari, Stefanie Wuhrer, Michael Zollhöfer, Thabo Beeler, Florian Bernard, Timo Bolkart, Adam Kortylewski, Sami Romdhani, Christian Theobalt, Volker Blanz, and Thomas Vetter. 3d morphable face models - past, present and future. *CoRR*, abs/1909.01815, 2019. URL http://arxiv.org/abs/1909.01815.

[15] Ruilong Li, Karl Bladin, Yajie Zhao, Chinmay Chinara, Owen Ingraham, Pengda Xiang, Xinglei Ren, Pratusha Prasad, Bipin Kishore, Jun Xing, and Hao Li. Learning formation of physically-based face attributes, 2020.

[16] Tianye Li, Timo Bolkart, Michael. J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 36(6):194:1–194:17, 2017. URL https://doi.org/10.1145/3130800.3130813.

[17] William A. P. Smith, Alassane Seck, Hannah Dee, Bernard Tiddeman, Joshua Tenenbaum, and Bernhard Egger. A morphable face albedo model. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5011–5020, 2020.

[18] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. MS-celeb-1M: A dataset and benchmark for large-scale face recognition. 2016.

[19] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[20] Gwangbin Bae, Martin de La Gorce, Tadas Baltrušaitis, Charlie Hewitt, Dong Chen, Julien Valentin, Roberto Cipolla, and Jingjing Shen. Digiface-1m: 1 million digital face images for face recognition. In *2023 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2023.

[21] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. VGGFace2: A dataset for recognising faces across pose and age. In *International Conference on Automatic Face and Gesture Recognition*, 2018.

[22] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J Cashman, and Jamie Shotton. Fake it till you make it: face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3681–3691, 2021.

[23] Yuan Yao, Praneet Bala, Abhiraj Mohan, Eliza Bliss-Moreau, Kristine Coleman, Sienna M Freeman, Christopher J Machado, Jessica Raper, Jan Zimmermann, Benjamin Y Hayden, and Hyun Soo Park. OpenMonkeyChallenge: Dataset and benchmark challenges for pose estimation of non-human primates. *Int. J. Comput. Vis.*, 131(1):243–258, January 2023.

[24] Debayan Deb, Susan Wiper, Alexandra Russo, Sixue Gong, Yichun Shi, Cori Tymoszek, and Anil Jain. Face recognition: Primates in the wild. 04 2018.

[25] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Matthew Johnson, Jingjing Shen, Nikola Milosavljević, Daniel Wilde, Stephan Garbin, Toby Sharp, Ivan Stojiljković, et al. 3d face reconstruction with dense landmarks. In *European Conference on Computer Vision*, pages 160–177. Springer, 2022.

[26] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie W. Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 2018. URL https://www.nature.com/articles/s41593-018-0209-y.

[27] Tanmay Nath*, Alexander Mathis*, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie W Mathis. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature Protocols*, 2019. URL https://doi.org/10.1038/s41596-019-0176-0.

[28] Claire L Witham. Automated face recognition of rhesus macaques. *J. Neurosci. Methods*, 300: 157–165, April 2018.

[29] Edwin Earl Catmull. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, 1974. AAI7504786.

[30] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '00, page 165–172, USA, 2000. ACM Press/Addison-Wesley Publishing Co. ISBN 1581132085. doi: 10.1145/344779.344862. URL https://doi.org/10.1145/344779.344862.

[31] Ali Elmahmudi and Hassan Ugail. A framework for facial age progression and regression using exemplar face templates. *The Visual Computer*, 37, 07 2021. doi: 10.1007/s00371-020-01960-z.

[32] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation, 2022. URL https://arxiv.org/abs/2201.12086.

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[34] Rafael Padilla, Wesley L. Passos, Thadeu L. B. Dias, Sergio L. Netto, and Eduardo A. B. da Silva. A comparative analysis of object detection metrics with a companion open-source toolkit. *Electronics*, 10(3), 2021. ISSN 2079-9292. doi: 10.3390/electronics10030279. URL https://www.mdpi.com/2079-9292/10/3/279.

[35] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.