A. Implementation details of FaCo-Net

This section introduces the network structure and implementation details of FaCo-Net. As shown in Fig. 7, given the mouth keypoints P^M, eye keypoints P^E, and the landmark of the subject $\mathbf{S}$, FaCo-Net aims to generate facial details P^F that are consistent with the input mouth and eye keypoints and keeps the target subject style. The computational process is $P^F = \text{FaCo-Net}(\mathbf{S}, P^M, P^E)$. Firstly, FaCo-Net uses three encoders to encode the mouth features, eye features, and target style features, respectively. Each encoder is implemented using an MLP. Mathematically, the calculation process is as follows:

$$P^F = \Psi_c((\mathbf{p}_m \odot \mathbf{p}_e) \oplus \mathbf{p}_f), \tag{17}$$

where $\mathbf{p}_m = \Psi^M(P^M)$, $\mathbf{p}_e = \Psi^E(P^E)$, $\mathbf{p}_f = \text{tile}(\Psi^F(\mathbf{S}))$. Ψ^M, Ψ^E, Ψ^S are encoders for mouth keypoints, eye keypoints, and landmarks of the subject, respectively. Ψ_c is the decoder of FaCo-Net. Afterward, the intermediate feature is decoded by an MLP-based decoder to obtain the overall facial keypoints. In order to make the generated facial keypoints have rich details and avoid oversmoothing, we add GAN loss as one of the objective functions. Specifically, the overall objective function of FaCo-Net is defined in Eq.(13) of the main paper. We use a discriminator implemented by an MLP to calculate the GAN loss. The loss function of the discriminator is Eq.(12) of the main paper. During the training stage, FaCo-Net and the discriminator are trained alternately. In the testing process, the discriminator will be discarded, and only FaCo-Net will be used for inference.

B. Architecture and loss functions of portrait renderer

The purpose of portrait rendering is to generate high-definition and realistic portrait videos. Fig. 8 shows the network architecture of our portrait renderer. Firstly, the network concatenates and fuses the conditional feature map of the t-th frame, a reference image, and the TPE at the t-th moment in the channel dimension. The generator of the network consists of a U-Net with skip connections. In detail, the network is an 8-layer UNet-like [37, 22] convolutional neural network with skip connections in each resolution layer. The resolution of each layer is $(256^2, 128^2, 64^2, 32^2, 16^2, 8^2, 4^2, 2^2)$ and the corresponding numbers of feature channels are $(64, 128, 256, 512, 512, 512, 512, 512)$. Each encoder layer consists of one convolution (stride 2) and one residual block. The decoder of the portrait renderer has a structure that mirrors the encoder, which consists of 8 residual convolutional modules with upsampling layers. There are skip connections between each encoder layer and its corresponding decoder layer to better propagate feature information across different levels.

The training process of portrait renderer follows a generative adversarial training strategy. We use a discriminator D with a multi-scale PatchGAN architecture. The purpose of discriminator D is to classify the results generated by generator G as fake and the real images as real. Specifically, we use the LSGAN loss as the adversarial loss to optimize discriminator D:

$$\mathcal{L}_{GAN}(D) = (p^* - 1)^2 + p^2, \tag{18}$$

where p^*, p represents the classification result of the discriminator when given a real image I_t^* and an image I_t generated by the generator, respectively. For the generator (G)'s loss function, we draw on [22] and incorporate color loss, mouth loss, perceptual loss, and feature matching loss to further optimize the generator's output. The generator's loss is defined as:

$$\mathcal{L}_G = \mathcal{L}_{GAN}(G) + \lambda_C \mathcal{L}_C + \lambda_M \mathcal{L}_M + \lambda_P \mathcal{L}_P + \lambda_{FM} \mathcal{L}_{FM}, \tag{19}$$

where $\mathcal{L}_{GAN}(G) = (p-1)^2$ is the adversarial loss, $\mathcal{L}_C$ is the color loss, $\mathcal{L}_M$ is the mouth loss, $\mathcal{L}_P$ is the perceptual loss, and $\mathcal{L}_{FM}$ is the feature matching loss. In our experiments, the hyper-parameters are set based on empirical values (50, 100, 10, 1). For the color consistency loss, we use L1 distance, *i.e.*, $\mathcal{L}_C = |I_t - I_t^*|_1$. To enhance the network's ability to generate mouth details, we use a mouth mask to compute the mouth loss, $\mathcal{L}_M = |MI_t - MI_t^*|_1$, where M is the mouth mask. For the perceptual loss, we use VGG19 to extract perceptual features and minimize the $L1$ distance between the generated image features and the ground truth image features. To improve the stability of the training process, we also add the feature matching loss $\mathcal{L}_{FM} = \sum_l^L |y - y^*|$ in the overall objective function, where L is the number of spatial layers in the discriminator. y and y^* are the intermediate predictions in D for the generated image and ground truth, respectively.

C. Data pro-processing pipeline

The purpose of data pre-processing is to extract facial keypoints, head pose, and other information from videos to train networks at different stages. The data pre-processing process is shown in Fig. 9. For the input video frame I, we first use Mediapipe[‡] to extract 478 3D facial keypoints (b). Then we use WHENet [53] to estimate the head pose H of the person. By utilizing the head pose, we align the facial keypoints with a rigid transformation (*i.e.*, Ⓣ in Fig. 9) to standard space to align the facial keypoints of different frames in the video, which are denoted as P^F. We extract

[‡] https://google.github.io/mediapipe/

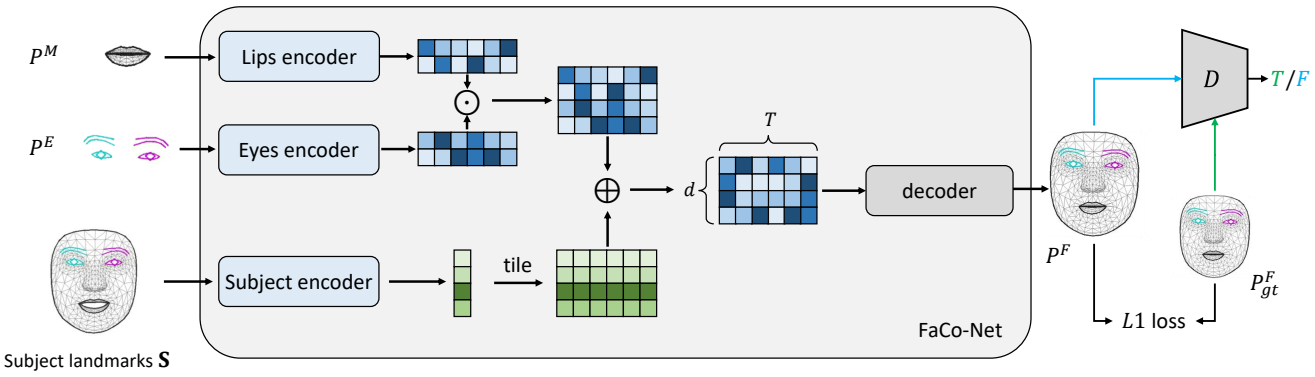

Figure 7: Architecture of FaCo-Net.

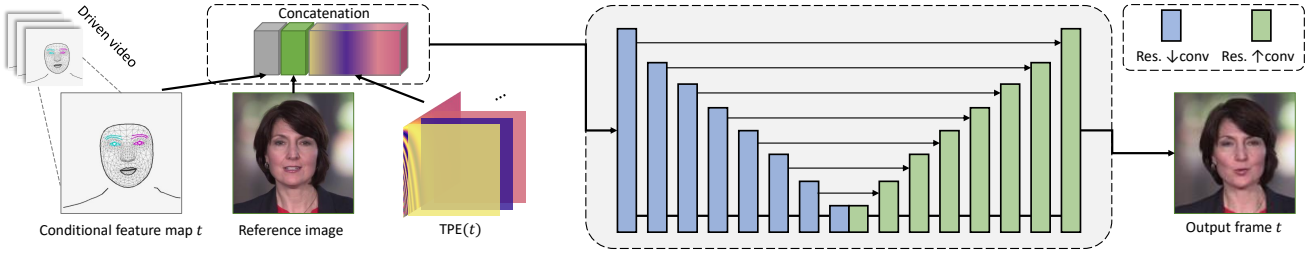

Figure 8: Architecture of portrait renderer.

the keypoints in the eye area and mouth area of P^F as the ground truth for training MODA. The eye keypoints P^E and mouth keypoints P^M are illustrated at (e) and (d) in Fig. 9, respectively.

To accurately extract shoulder information as a condition for the torso, we design a semantic-guided 3D torso points estimation method. Specifically, we first use BiSeNet [47] to segment semantic information (d) from the image I. Furthermore, we design a torso points extraction algorithm to estimate key points information for the upper body. The algorithm consists of the following steps:

1. We first calculate the semantic boundary of the upper body by computing the boundary between the upper body semantics and the background/hair semantics.

2. Then, we use morphological operations on the semantic boundary to expand its range, and we extract key points from the semantic contour using a polygon fitting algorithm.

3. Next, we use a k-nearest neighbors algorithm to constrain the number of key points for each side of the shoulder. k is set to 9 in our experiments.

4. After obtaining the 2D key points of the torso, we use the average depth information of the face mesh (b) as the depth information of the torso keypoints.

The visualization result of the extracted body keypoints is shown in Fig. 9(h). By adding the face mesh (b) and upper body key points P^T (h) projected onto the image coordinate, we obtain the condition image I^c (i) of the portrait image.

For the training of the proposed system, given a reference image of a subject I_r, we extract the face mesh obtained from a face mesh detector as the style $\mathbf{S}$. The audio information $\mathbf{A}$ and $\mathbf{S}$ are used as input, P^M, P^E, H, P^T in Fig. 9 are used as the target, to train the first stage of MODA. The goal of the second stage, FaCo-Net, is to learn the mapping from S, P^E, P^M to P^F, so that the generated P^F contains rich details. Finally, the condition image I^c, input image I, and reference image I_r are combined to form the training data for the portrait renderer.

D. Additional experimental results

D.1. Additional visual comparison results

In this section, we provide additional visual comparison results among different methods in Fig. 10. MakeItTalk [54] generates low-resolution videos without head/torso motions. The results from LSP [22] have some warping effects and are not 3D consistent. Wav2Lip [29] can generate accurate mouth motions. However, their mouth areas usually have blurry boundaries and artifacts, which make the video unnatural. The results of AD-NeRF [10] have blurry bound-

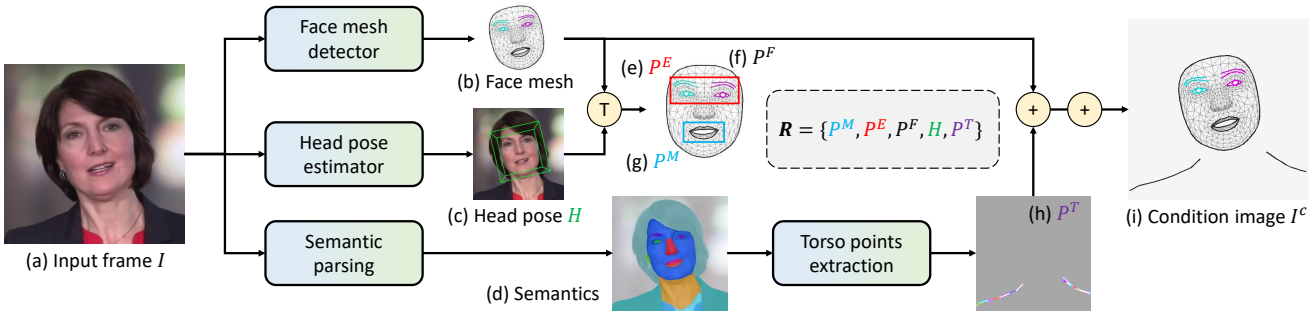

Figure 9: Pipeline of data pre-processing.

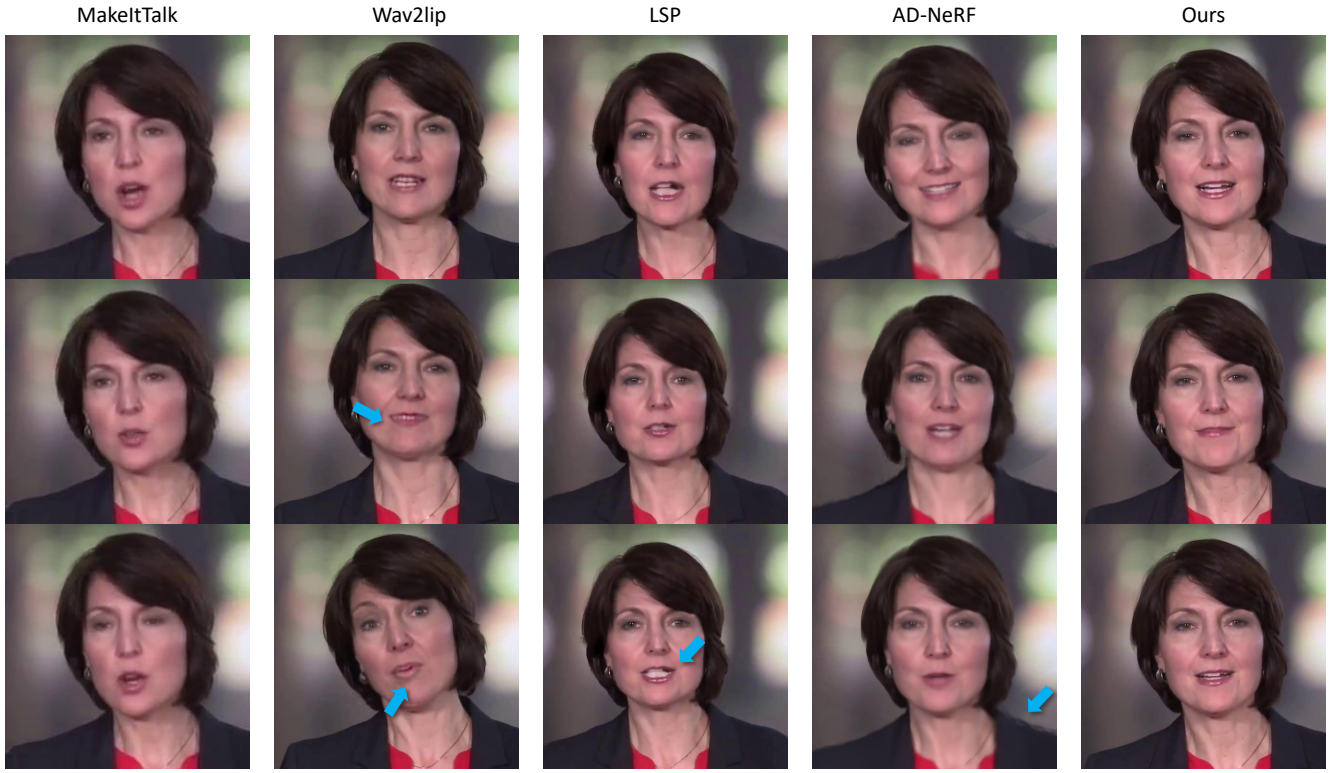

Figure 10: Additional comparisons among different methods.

aries around shoulders, and the relative movements between the head and torso are unnatural. Compared to other baselines, our system generates portrait videos with correct lip-sync, natural movements, and high visual quality.

D.2. Running time comparisons

In this section, we provide running time comparisons among different methods that can generate high-fidelity videos. All models are trained and tested under the same condition (*i.e.*, a single RTX 3090 GPU). Results are reported at Tab. 6. Since the compared methods require training the network separately for each subject, their training time increases proportionally with the number of subjects. Our method, on the other hand, can generalize across multiple individuals and therefore can be trained simultaneously on multiple subjects, resulting in significant time reduction, especially as the number of training subjects increases (*e.g.*, $\mathbf{2.5}\times, \mathbf{11.5}\times$ faster than LSP and GeneFace under 3 subjects). During the inference stage, LSP needs to use different networks to generate mouth movements and head movements separately, while our method can generate multiple features to drive the portrait through mapping once, resulting in faster overall inference time. Both AD-NeRF and GeneFace require the use of NeRF to render each frame,

Table 6: Running time comparisons between the proposed method and other methods.

Method	Training time			Inference time		
	1 subject	2 subjects	3 subjects	5s audio	10s audio	30s audio
LSP [22]	$\sim$ **14h**	$\sim$ 30h	$\sim$ 50h	15s	26s	70s
AD-NeRF [10]	$\sim$ 70h	$\sim$ 145h	$\sim$ 220h	$\sim$ 11min	$\sim$ 25min	$\sim$ 80min
GeneFace [44]	$\sim$ 85h	$\sim$ 150h	$\sim$ 230h	$\sim$ 26min	$\sim$ 92min	$\sim$ 270min
MODA (Ours)	$\sim$ 15h	$\sim$ **17h**	$\sim$ **20h**	**12s**	**25s**	**62s**

which significantly slows down the inference speed. Overall, our method achieves faster training and inference speed, demonstrating the superiority of our proposed approach.