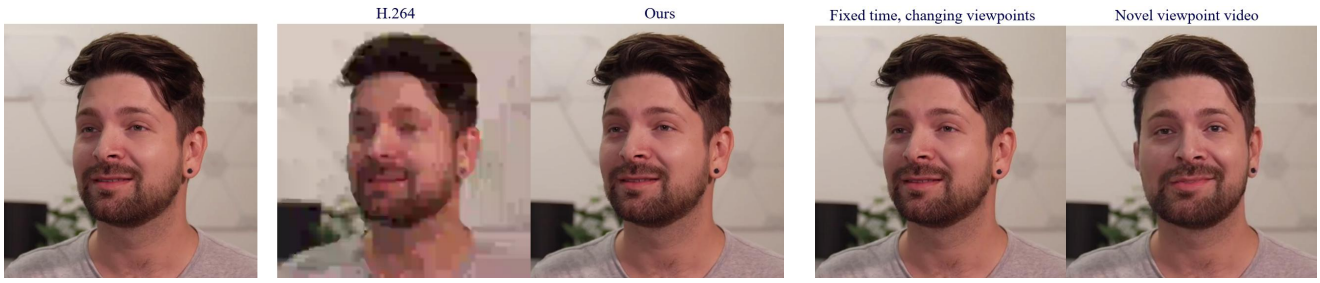
# One-Shot Free-View Neural Talking-Head Synthesis for Video Conferencing

Ting-Chun Wang Arun Mallya Ming-Yu Liu

NVIDIA Corporation




(a) Original video (b) Compressed videos at the same bit-rate (c) Our re-rendered novel-view results

Figure 1: Our method can re-create a talking-head video using only a single source image (*e.g.*, the first frame) and a sequence of unsupervisedly-learned 3D keypoints, representing motions in the video. Our novel keypoint representation provides a compact representation of the video that is 10× more efficient than the H.264 baseline can provide. A novel 3D keypoint decomposition scheme allows re-rendering the talking-head video under different poses, simulating often missed face-to-face video conferencing experiences. **Click the image to play the video in a browser.**

## Abstract

*We propose a neural talking-head video synthesis model and demonstrate its application to video conferencing. Our model learns to synthesize a talking-head video using a source image containing the target person's appearance and a driving video that dictates the motion in the output. Our motion is encoded based on a novel keypoint representation, where the identity-specific and motion-related information is decomposed unsupervisedly. Extensive experimental validation shows that our model outperforms competing methods on benchmark datasets. Moreover, our compact keypoint representation enables a video conferencing system that achieves the same visual quality as the commercial H.264 standard while only using one-tenth of the bandwidth. Besides, we show our keypoint representation allows the user to rotate the head during synthesis, which is useful for simulating face-to-face video conferencing experiences. Our project page can be found at* https://nvlabs.github.io/face-vid2vid.

## 1. Introduction

We study the task of generating a realistic talking-head video of a person using one source image of that person and a driving video, possibly derived from another person. The source image encodes the target person's appearance, and the driving video dictates motions in the output video.

We propose a *pure* neural rendering approach, where we render a talking-head video using a deep network in the one-shot setting without using a graphics model of the 3D human head. Compared to 3D graphics-based models, 2D-based methods enjoy several advantages. First, it avoids 3D model acquisition, which is often laborious and expensive. Second, 2D-based methods can better handle the synthesis of hair, beard, *etc*., while acquiring detailed 3D geometries of these regions is challenging. Finally, they can directly synthesize accessories present in the source image, including eyeglasses, hats, and scarves, without their 3D models.

However, existing 2D-based one-shot talking-head methods [68, 82, 92] come with their own set of limitations. Due to the absence of 3D graphics models, they can only synthesize the talking-head from the original viewpoint. They cannot render the talking-head from a novel view.

Our approach addresses the fixed viewpoint limitation and achieves *local free-view* synthesis. One can freely change the viewpoint of the talking-head within a large neighborhood of the original viewpoint, as shown in Fig. 1(c). Our model achieves this capability by representing a video using a novel 3D keypoint representation, where person-specific and motion-related information is decomposed. Both the keypoints and their decomposition are learned unsupervisedly. Using the decomposition, we can apply 3D transformations to the person-specific representation to simulate head pose

changes such as rotating the talking-head in the output video. Figure 2 gives an overview of our approach.

We conduct extensive experimental validation with comparisons to state-of-the-art methods. We evaluate our method on several talking-head synthesis tasks, including video reconstruction, motion transfer, and face redirection. We also show how our approach can be used to reduce the bandwidth of video conferencing, which has become an important platform for social networking and remote collaborations. By sending only the keypoint representation and reconstructing the source video on the receiver side, we can achieve a 10x bandwidth reduction as compared to the commercial H.264 standard without compromising the visual quality.

**Contribution 1.** A novel one-shot neural talking-head synthesis approach, which achieves better visual quality than state-of-the-art methods on the benchmark datasets.

**Contribution 2.** Local free-view control of the output video, without the need for a 3D graphics model. Our model allows changing the viewpoint of the talking-head during synthesis.

**Contribution 3.** Reduction in bandwidth for video streaming. We compare our approach to the commercial H.264 standard on a benchmark talking-head dataset and show that our approach can achieve 10× bandwidth reduction.

## 2. Related Works

**GANs.** Since its introduction by Goodfellow *et al*. [21], GANs have shown promising results in various areas [48], such as unconditional image synthesis [21, 23, 32, 33, 34, 49, 61], image translation [8, 12, 27, 29, 46, 47, 58, 67, 73, 84, 102, 103], text-to-image translation [62, 90, 95], image processing [17, 18, 28, 36, 39, 41, 42, 44, 72, 78, 89, 94], and video synthesis [2, 10, 35, 45, 50, 60, 63, 69, 82, 83, 101]. We focus on using GANs to synthesize talking-head videos in this work.

**3D model-based talking-head synthesis.** Works on transferring the facial motion of one person to another—face reenactment—can be divided into *subject-dependent* and *subject-agnostic* models. Traditional 3D-based methods usually build a subject-dependent model, which can only synthesize one subject. Moreover, they focus on transferring the expressions without the head movement [71, 75, 76, 77, 80]. This line of works starts by collecting footage of the target person to be synthesized using an RGB or RGBD sensor [76, 77]. Then a 3D model of the target person is built for the face region [6]. At test time, the new expressions are used to drive the 3D model to generate the desired motions.

More recent 3D model-based methods are able to perform subject-agnostic face synthesis [19, 20, 55, 57]. While they can do an excellent job synthesizing the inner face region, they have a hard time generating realistic hair, teeth, accessories, etc. Due to the limitations, most modern face reenactment frameworks adopt the 2D approach. Another line of works [15, 74] focuses on controllable face generation, providing explicit control over the generated face from a pretrained StyleGAN [33, 34]. However, it is not clear how they can be adapted to modifying real images since the inverse mapping from images to latent codes is nontrivial.

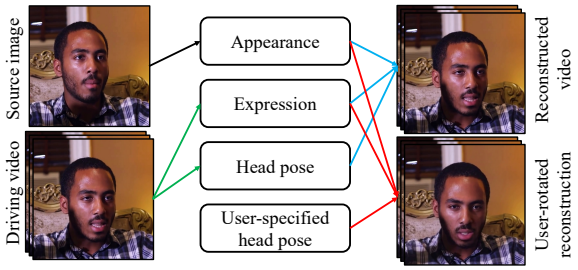



Figure 2: Combining appearance information from the source image, our framework can re-create a driving video by just using the expression and head pose information from the driving video. With a user-specified head pose, it can also synthesize the head pose change in the output video.

**2D-based talking-head synthesis.** Again, 2D approaches can be classified into subject-dependent and subject-agnostic models. Subject-dependent models [5, 88] can only work on specific persons since the model is only trained on the target person. On the other hand, subject-agnostic models [4, 9, 11, 20, 22, 24, 30, 56, 60, 68, 70, 81, 82, 86, 92, 93, 100] only need a single image of the target person, who is not seen during training, to synthesize arbitrary motions. Siarohin *et al*. [68] warp extracted features from the input image, using motion fields estimated from sparse keypoints. On the other hand, Zakharov *et al*. [93] demonstrate that it is possible to achieve promising results using direct synthesis methods without any warping. Few-shot vid2vid [82] injects the information into their generator by dynamically determining the the parameters in the SPADE [58] modules. Zakharov *et al*. [92] decompose the low and high frequency components of the image and greatly accelerate the inference speed of the network. While demonstrating excellent result qualities, these methods can only synthesize fixed viewpoint videos, which produce less immersive experiences.

**Video compression.** A number of recent works [3, 16, 26, 43, 52, 65, 87] propose using a deep network to compress arbitrary videos. The general idea is to treat the problem of video compression as one of interpolating between two neighboring keyframes. Through the use of deep networks to replace various parts of the traditional pipeline, as well as techniques such as hierarchical interpolation and joint encoding of residuals and optical flows, these prior works reduce the required bit-rate. Other works [53, 85, 91, 97] focus on restoring the quality of low bit-rate videos using deep networks. Most related to our work is DAVD-Net [97], which restores talking-head videos using information from the audio stream. Our proposed method is different from these works in a number of aspects, in both the goal as well as the method used to achieve compression. We specifically focus on videos of talking faces. People's faces have an in-

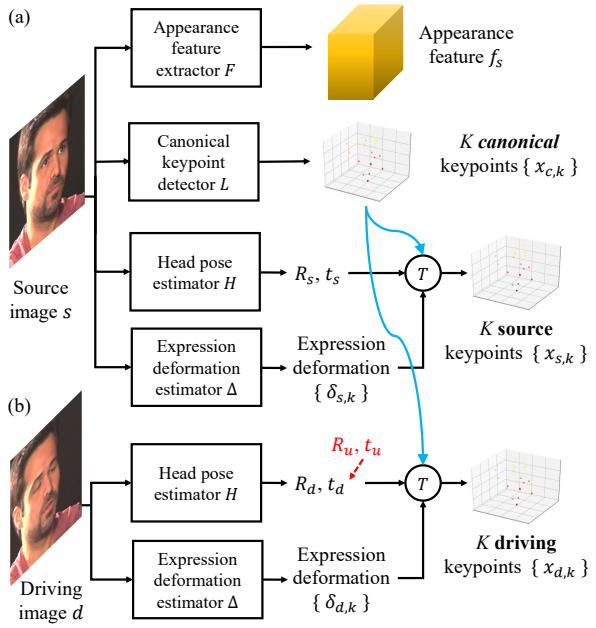



Figure 3: Source and driving feature extraction. (a) From the source image, we extract appearance features and 3D canonical keypoints. We also estimate the head pose and the keypoint perturbations due to expressions. We use them to compute the source keypoints. (b) For the driving image, we again estimate the head pose and the expression deformations. By reusing canonical keypoints from the source image, we compute the driving keypoints.

herent structure—from the shape to the relative arrangement of different parts such as eyes, nose, mouth, *etc*. This allows us to use keypoints and associated metadata for efficient compression, an order of magnitude better than traditional codecs. Our method does not guarantee pixel-aligned output videos; however, it faithfully models facial movements and emotions. It is also better suited for video streaming as it does not use bi-directional or B-frames.

## 3. Method

Let $s$ be an image of a person, referred to as the source image. Let $\{d_1, d_2, ..., d_N\}$ be a talking-head video, called the driving video, where $d_i$'s are the individual frames, and $N$ is the total number of frames. Our goal is to generate an output video $\{y_1, y_2, ..., y_N\}$, where the identity in $y_i$'s is inherited from $s$ and the motions are derived from $d_i$'s. Several talking-head synthesis tasks fall in the above setup. When $s$ is a frame of the driving video (*e.g.*, the first frame: $s \equiv d_1$.), we have a video reconstruction task. When $s$ is not from the driving video, we have a motion transfer task.

We propose a pure neural synthesis approach that does not use any 3D graphics models, such as the well-known 3D morphable model (3DMM) [6]. Our approach contains three major steps: *1) source image feature extraction*, *2) driving video feature extraction*, and *3) video generation*. In Fig. 3, we illustrate *1)* and *2)*, while Fig. 5 shows *3)*. Our key ingredient is an unsupervised approach for learning a set of **3D** keypoints and their decomposition. We decompose the keypoints into two parts, one that models the facial expressions and the other that models the geometric signature of a person. These two parts are combined with the target head pose to generate the image-specific keypoints. After the keypoints are estimated, they are then used to learn a mapping function between two images. We implement these steps using a set of networks and train them jointly. In the following, we discuss the three steps in detail.

### 3.1. Source image feature extraction

Synthesizing a talking-head requires knowing the appearance of the person, such as the skin and eye colors. As shown in Fig. 3(a), we first apply a 3D appearance feature extraction network $F$ to map the source image $s$ to a 3D appearance feature volume $f_s$. Unlike a 2D feature map, $f_s$ has three spatial dimensions: width, height, and depth. Mapping to a 3D feature volume is a crucial step in our approach. It allows us to operate the keypoints in the 3D space for rotating and translating the talking-head during synthesis.

We extract a set of $K$ 3D keypoints $x_{c,k} \in \mathbb{R}^3$ from $s$ using a canonical 3D keypoint detection network $L$. We set $K = 20$ throughout the paper unless specified otherwise. Note that these keypoints are unsupervisedly learned and different from the common facial landmarks. We note that the extracted keypoints are meant to be independent of the face's pose and expression. They shall only encode a person's geometry signature in a neutral pose and expression.

Next, we extract pose and expression information from the image. We use a head pose estimation network $H$ to estimate the head pose of the person in $s$, parameterized by a rotation matrix $R_s \in \mathbb{R}^{3\times3}$ and a translation vector $t_s \in \mathbb{R}^3$. In addition, we use an expression deformation estimation network $\Delta$ to estimate a set of $K$ 3D deformations $\delta_{s,k}$—the deformations of keypoints from the neutral expression. Both $H$ and $\Delta$ extract motion-related geometry information in the image. We combine the identity-specific information extracted by $L$ with the motion-related information extracted by $H$ and $\Delta$ to obtain the source 3D keypoints $x_{s,k}$ via a transformation $T$:

$$x_{s,k} = T(x_{c,k}, R_s, t_s, \delta_{s,k}) \equiv R_s x_{c,k} + t_s + \delta_{s,k} \quad (1)$$

The final keypoints are image-specific and contain person-signature, pose, and expression information. Figure 4 visualizes the keypoint computation pipeline.

The 3D keypoint decomposition in (1) is of paramount importance to our approach. It commits to a prior decomposition of keypoints: geometry-signatures, head poses, and expressions. It helps learn manipulable representations and differs our approach from prior 2D keypoint-based neural

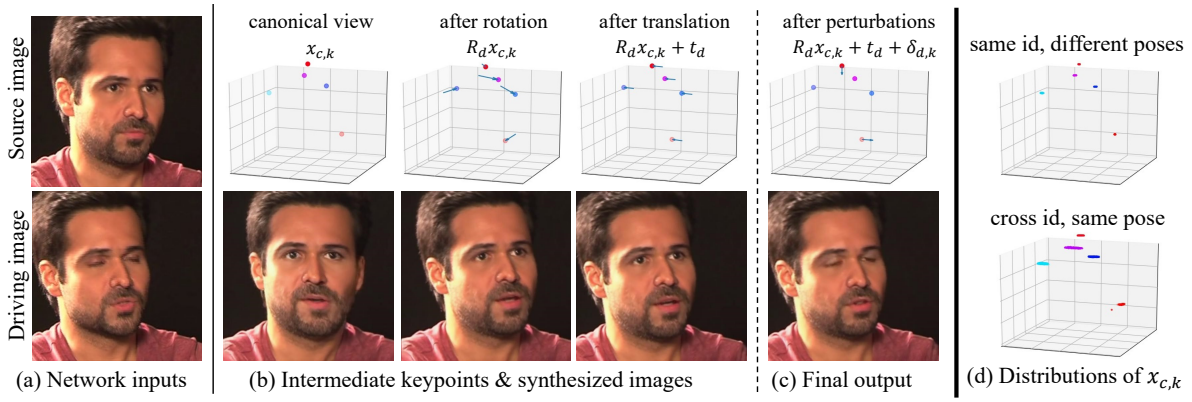



Figure 4: Keypoint computation pipeline. For each step, we show the first five keypoints and the synthesized images using them. Given the source image (a), our model first predicts the canonical keypoints (b). We then apply the rotation and translation estimated from the driving image to the canonical keypoints, bringing them to the target head pose (transformations illustrated as arrows). (c) The expression-aware deformation adjusts the keypoints to the target expression (*e.g.* closed eyes). (d) We visualize the distributions of canonical keypoints estimated from different images. Upper: the canonical keypoints from different poses of a person are similar. Lower: the canonical keypoints from different people in the same pose are different.

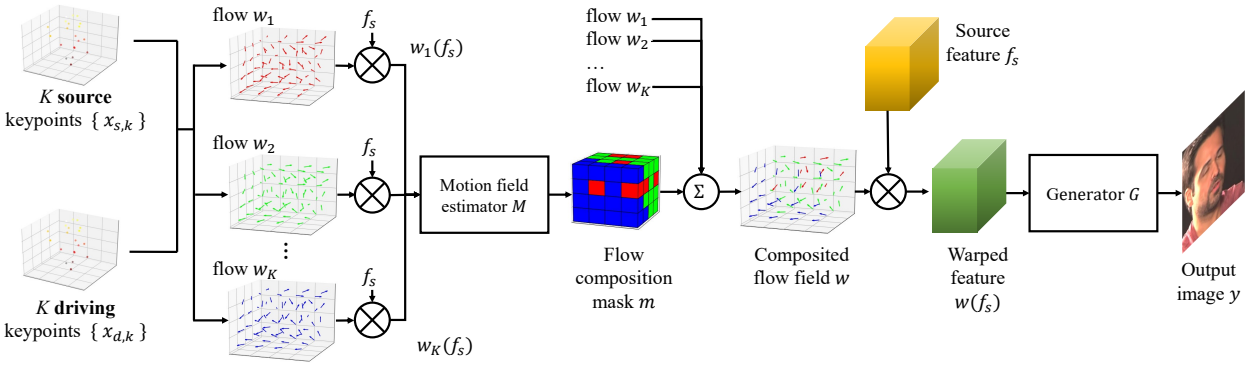



Figure 5: Video synthesis. We use the source and driving keypoints to estimate $K$ flows, $w_k$'s. These flows are used to warp the source feature $f_s$. The results are combined and fed to the motion field estimation network $M$ to produce a flow composition mask $m$. A linear combination of $m$ and $w_k$'s then produces the composited flow field $w$, which is used to warp the 3D source feature. Finally, the generator $G$ converts the warped feature to the output image $y$.

talking-head synthesis approaches [68, 82, 92]. Also note that unlike FOMM [68], our model does not estimate Jacobians. The Jacobian represents how a local patch around the keypoint can be transformed into the corresponding patch in another image via an affine transformation. Instead of explicitly estimating them, our model assumes the head is mostly rigid and the local patch transformation can be directly derived from the head rotation via $J_s = R_s$. Avoiding estimating Jacobians allows us to further reduce the transmission bandwidth for the video conferencing application, as detailed in Sec. 5.

### 3.2. Driving video feature extraction

We use $d$ to denote a frame in $\{d_1, d_2, ..., d_N\}$ as individual frames are processed in the same way. To extract motion-related information, we apply the head pose estimator $H$ to get $R_d$ and $t_d$ and apply the expression deformation estimator $\Delta$ to obtain $\delta_{d,k}$'s, as shown in Fig. 3(b).

Now, instead of extracting canonical 3D keypoints from the driving image $d$ using $L$, we reuse $x_{c,k}$, which were extracted from the source image $s$. This is because the face in the *output* image must have the same identity as the one in the source image $s$. There is no need to compute them again. Finally, the identity-specific information and the motion-related information are combined to compute the driving keypoints for the driving image $d$ in the same way we obtained source keypoints:

$$x_{d,k} = T(x_{c,k}, R_d, t_d, \delta_{d,k}) = R_d x_{c,k} + t_d + \delta_{d,k} \quad (2)$$

We apply this processing to each frame in the driving video, and each frame can be compactly represented by $R_d$, $t_d$, and $\delta_{d,k}$'s. This compact representation is very useful for low-bandwidth video conferencing. In Sec. 5, we will introduce an entropy coding scheme to further compress these

quantities to reduce the bandwidth utilization.

Our approach allows manual changes to the 3D head pose during synthesis. Let $R_u$ and $t_u$ be user-specified rotation and translation, respectively. The final head pose in the output image is given by $R_d \leftarrow R_u R_d$ and $t_d \leftarrow t_u + t_d$. In video conferencing, we can change a person's head pose in the video stream freely despite the original view angle.

### 3.3. Video generation

As shown in Fig. 5, we synthesize an output image by warping the source feature volume and then feeding the result to the image generator $G$ to produce the output image $y$. The warping approximates the nonlinear transformation from $s$ to $d$. It *re-positions* the source features for the synthesis task.

To obtain the required warping function $w$, we take a bottom-up approach. We first compute the warping flow $w_k$ induced by the $k$-th keypoint using the first order approximation [68], which is reliable only around the neighborhood of the keypoint. After obtaining all $K$ warping flows, we apply each of them to warp the source feature volume. The $K$ warped features are aggregated to estimate a flow composition mask $m$ using the motion field estimation network $M$. This mask indicates which of the $K$ flows to use at each spatial 3D location. We use this mask to combine the $K$ flows to produce the final flow $w$. Details of the operation are given in Appendix A.1.

### 3.4. Training

We train our model using a dataset of talking-head videos where each video contains a single person. For each video, we sample two frames: one as the source image $s$ and the other as the driving image $d$. We train the networks $F$, $\Delta$, $H$, $L$, $M$, and $G$ by minimizing the following loss:

$$\mathcal{L}_P(d, y) + \mathcal{L}_G(d, y) + \mathcal{L}_E(\{x_{d,k}\}) + \mathcal{L}_L(\{x_{d,k}\}) + \mathcal{L}_H(R_d, \bar{R}_d) + \mathcal{L}_\Delta(\{\delta_{d,k}\}) \quad (3)$$

In short, the first two terms ensure the output image looks similar to the ground truth. The next two terms enforce the predicted keypoints to be consistent and satisfy some prior knowledge about the keypoints. The last two terms constrain the estimated head pose and keypoint perturbations. We briefly discuss these losses below and leave the implementation details in Appendix A.2.

**Perceptual loss $\mathcal{L}_P$.** We minimize the perceptual loss [31, 84] between the output and the driving image, which is helpful in producing sharp-looking outputs.

**GAN loss $\mathcal{L}_G$.** We use a multi-resolution patch GAN where the discriminator predicts at the patch-level. We also minimize the discriminator feature matching loss [82, 84].

**Equivariance loss $\mathcal{L}_E$.** This loss ensures the consistency of image-specific keypoints $x_{d,k}$. For a valid keypoint, when applying a 2D transformation to the image, the predicted keypoints should change according to the applied transformation [68, 98]. Since we predict 3D instead of 2D keypoints, We use an orthographic projection to project the keypoints to the image plane before computing the loss.

**Keypoint prior loss $\mathcal{L}_L$.** We use a keypoint coverage loss to encourage the estimated image-specific keypoints $x_{d,k}$'s to spread out across the face region, instead of crowding around a small neighborhood. We compute the distance between each pair of the keypoints and penalize the model if the distance falls below a preset threshold. We also use a keypoint depth prior loss that encourages the mean depth of the keypoints to be around a preset value.

**Head pose loss $\mathcal{L}_H$.** We penalize the prediction error of the head rotation $R_d$ compared to the ground truth $\bar{R}_d$. Since acquiring the ground truth head pose for a large-scale video dataset is expensive, we use a pre-trained pose estimation network [66] to approximate $\bar{R}_d$.

**Deformation prior loss $\mathcal{L}_\Delta$.** The loss penalizes the magnitude of the deformations $\delta_{d,k}$'s. As the deformations model the deviation from the canonical keypoints due to expression changes, their magnitudes should be small.

## 4. Experiments

**Implementation details.** The network architecture and training hyper-parameters are available in Appendix A.3.

**Datasets.** Our evaluation is based on VoxCeleb2 [13] and TalkingHead-1KH, a newly collected large-scale talking-head video dataset. It contains 180K videos, which are often with higher quality and larger resolution than those in VoxCeleb2. Details are available in Appendix B.1.

### 4.1. Talking-head image synthesis

**Baselines.** We compare our neural talking-head model with three state-of-the-art methods: FOMM [68], few-shot vid2vid (fs-vid2vid) [82], and bi-layer neural avatars (bi-layer) [92]. We use the released pre-trained model on VoxCeleb2 for bi-layer [92], and retrain from scratch for others on the corresponding datasets. Since bi-layer does not predict the background, we subtract the background when doing quantitative analyses.

**Metrics.** We evaluate a synthesis model on 1) reconstruction faithfulness using $L_1$, PSNR, SSIM/MS-SSIM, 2) output visual quality using FID, and 3) semantic consistency using average keypoint distance (AKD). Please consult Appendix B.2 for details of the performance metrics.

**Same-identity reconstruction.** We first compare the face synthesis results where the source and driving images are of the same person. The quantitative evaluation is shown in Table 1. It can be seen that our method outperforms other competing methods on all metrics for both datasets. To verify that our superior performance does not come from more parameters, we train another large FOMM model with doubled filter size (FOMM-L), which is larger than our model. We can see that enlarging the model actually hurts the perfor-

Table 1: Comparisons with state-of-the-art methods on face reconstruction. ↑ larger is better. ↓ smaller is better.

| | VoxCeleb2 [13] | | | | | | TalkingHead-1KH | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | L1↓ | PSNR↑ | SSIM↑ | MS-SSIM↑ | FID↓ | AKD↓ | L1↓ | PSNR↑ | SSIM↑ | MS-SSIM↑ | FID↓ | AKD↓ |
| fs-vid2vid [82] | 17.10 | 20.36 | 0.71 | Nan | 85.76 | 3.41 | 15.18 | 20.94 | 0.75 | Nan | 63.47 | 11.07 |
| FOMM [68] | 12.66 | 23.25 | 0.77 | 0.83 | 73.71 | 2.14 | 12.30 | 23.67 | 0.79 | 0.83 | 55.35 | 3.76 |
| FOMM-L [68] | N/A | N/A | N/A | N/A | N/A | N/A | 12.81 | 23.13 | 0.78 | Nan | 60.58 | 4.04 |
| Bi-layer [92] | 23.95 | 16.98 | 0.66 | 0.66 | 203.36 | 5.38 | N/A | N/A | N/A | N/A | N/A | N/A |
| **Ours** | **10.74** | **24.37** | **0.80** | **0.85** | **69.13** | **2.07** | **10.67** | **24.20** | **0.81** | **0.84** | **52.08** | **3.74** |


Figure 6: Qualitative comparisons on the Voxceleb2 dataset [13]. Our method better captures the driving motions.


Figure 7: Qualitative comparisons on the TalkingHead-1KH dataset. Our method produces more faithful and sharper results.

mance, proving that simply making the model larger does not help. Figures 6 and 7 show the qualitative comparisons. Our method can more faithfully reproduce the driving motions.

**Cross-identity motion transfer.** Next, we compare results where the source and driving images are from different persons (cross-identity). Table 2 shows that our method achieves the best results compared to other methods. Figure 8 compares results from different approaches. It can be seen that our method generates more realistic images while still preserving the original identity. For cross-identity motion transfer, it is sometimes useful to use relative motion [68], where only motion differences between two neighboring frames in the driving video are transferred. We report comparisons using relative motion in Appendix B.3.

Table 2: Quantitative results on cross-identity motion transfer. Our method achieves lowest FIDs and highest identity-preserving scores (CSIM [93]).

| Method | VoxCeleb2 [13] FID↓ | VoxCeleb2 [13] CSIM↑ | TalkingHead-1KH FID↓ | TalkingHead-1KH CSIM↑ |
|---|---|---|---|---|
| fs-vid2vid [82] | 59.84 | 0.593 | 52.72 | 0.703 |
| FOMM [68] | 84.06 | 0.582 | 87.32 | 0.542 |
| **Ours** | **55.64** | **0.753** | **46.99** | **0.777** |

Table 3: Face frontalization quantitative comparisons. We compute the identity loss and angle difference for each method and report the percentage where the losses are within a threshold (0.05 and 15 degrees, respectively).

| Method | Identity (%)↑ | Angle (%)↑ | Both (%)↑ | FID↓ |
|---|---|---|---|---|
| pSp [64] | 57.3 | **99.8** | 57.3 | 118.08 |
| RaR [99] | 55.1 | 87.2 | 50.8 | 78.81 |
| **Ours** | **94.3** | 90.9 | **85.9** | **23.87** |

**Ablation study.** We benchmark the performance gains from the proposed keypoint decomposition scheme, the mask estimation network, and pose supervision in Appendix B.4.

**Failure cases.** Our model fails when large occlusions and image degradation occur, as visualized in Appendix B.5.

**Face recognition.** Since the canonical keypoints are independent of poses and expressions, they can also be applied to face recognition. In Appendix B.6, we show that this achieves 5x accuracy than using facial landmarks.

### 4.2. Face redirection.

**Baselines.** We benchmark our talking-head model's face redirection capability using latest face frontalization methods: pixel2style2pixel (pSp) [64] and Rotate-and-Render (RaR) [99]. pSp projects the original image into a latent code and then uses a pre-trained StyleGAN [1] to synthesize the frontalized image. RaR adopts a 3D face model to rotate the input image and re-renders it in a different pose.

**Metrics.** The results are evaluated by two metrics: identity preservation and head pose angles. We use a pre-trained face recognition network [59] to extract high-level features, and compute the distance between the rotated face and the original one. We use a pre-trained head pose estimator [66] to obtain head angles of the rotated face. For a rotated image, if its identity distance to the original image is within some threshold, and/or its head angle is within some tolerance to the desired angle, we consider it as a "good" image.

We report the ratio of "good" images using our metric for each method in Table 3. Example comparisons can be found in Fig. 9. It can be seen that while pSp can always frontalize the face, the identity is usually lost. RaR generates more visually appealing results since it adopts 3D face models, but has problems outside the inner face regions. Besides, both methods have issues regarding the temporal stability. Only our method can realistically frontalize the inputs.


Figure 8: Qualitative results for cross-subject motion transfer. Ours captures the motion and preserves the identity better.

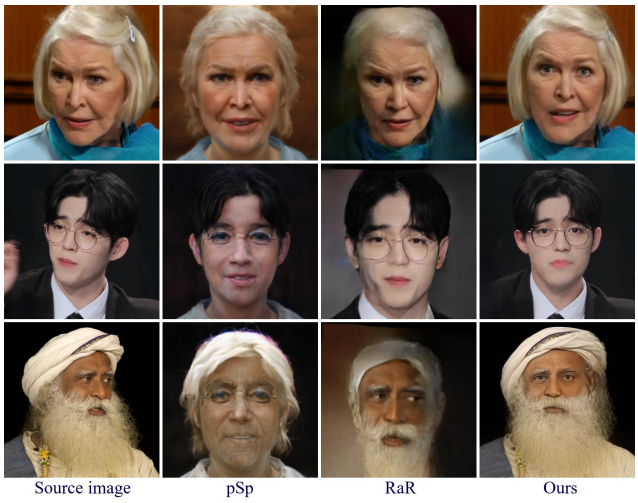



Figure 9: Qualitative results for face frontalization. Our method more realistically frontalizes the faces.

## 5. Neural Talking-Head Video Conferencing

Our talking-head synthesis model distills motions in a driving image using a compact representation, as discussed in Sec. 3. Due to this advantage, our model can help reduce the bandwidth consumed by video conferencing applications. We can view the process of video conferencing as the receiver watching an animated version of the sender's face.

Figure 10 shows a video conferencing system built using our neural talking-head model. For a driving image $d$, we use the driving image encoder, consisting of $\Delta$ and $H$, to extract the expression deformations $\delta_{d,k}$ and the head pose $R_d, t_d$. By representing a rotation matrix using Euler angles, we have a compact representation of $d$ using $3K + 6$ numbers: 3 for the rotation, 3 for the translation, and 3K for the deformations. We further compress these values using an entropy encoder [14]. Details are in Appendix C.1.

The receiver receives the entropy-encoded representation and uses the entropy decoder to recover $\delta_{d,k}$ and $R_d, t_d$. They are then fed into our talking-head synthesis framework

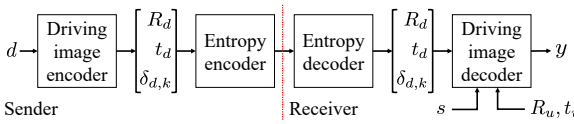



Figure 10: Our video compression framework. On the sender's side, the driving image encoder extracts keypoint perturbations $\delta_{d,k}$ and head poses $R_d$ and $t_d$. They are then compressed using an entropy encoder and sent to the receiver. The receiver decompresses the message and uses them along with the source image $s$ to generate $y$, a reconstruction of the input $d$. Our framework can also change the head pose on the receiver's side by using the pose offset $R_u$ and $t_u$.

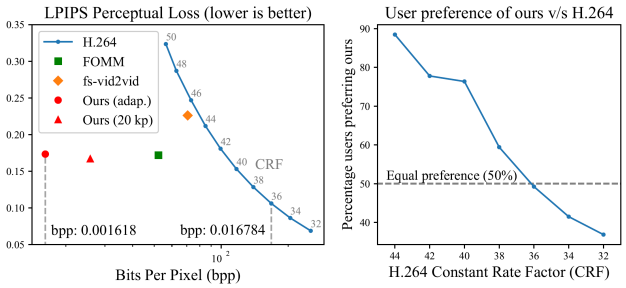



Figure 11: Automatic and human evaluations for video compression. Ours requires much lower bandwidth due to our keypoint decomposition and adaptive scheme.

to reconstruct the original image $d$. We assume that the source image $s$ is sent to the receiver at the beginning of the video conferencing session or re-used from a previous session. Hence, it does not consume additional bandwidth. We note that the source image is different from the I-frame in traditional video codecs. While I-frames are sent frequently in video conferencing, our source image only needs to be sent once at the beginning. Moreover, the source image can be an image of the same person captured on a different day, a different person, or even a face portrait painting.

**Adaptive number of keypoints.** Our basic model uses a fixed number of keypoints during training and inference. However, since the transmitted bits are proportional to the number of keypoints, it is advantageous to change this number to accommodate varying bandwidth requirements dynamically. Using keypoint dropouts at training time, we derive a model that can dynamically use a smaller number of keypoints for reconstruction. This allows us to transmit even fewer bits without compromising visual quality. On average, with the adaptive scheme, the number of sent keypoints is reduced from $K = 20$ to 11.52 (Appendix C.2).

**Benchmark dataset.** We manually select a dataset of 222 high-quality talking-head videos. Each video's resolution is $512 \times 512$ and the length is up to 1024 frames for evaluation.

**Baselines.** We compare our video streaming method with the popular H.264 codec. In order to conform to real-time video streaming cases, we disable the use of bidirectional B-frames, as this uses information from the future. By varying the constant rate factor (CRF) while encoding the ground truth input videos, we can obtain a set of videos of varying qualities and sizes suitable for a range of bandwidth availability. We also compare with FOMM [68] and fs-vid2vid [82], which also use keypoints or facial landmarks. For a fair comparison, we also compress their keypoints and Jacobians using our entropy coding scheme.

**Metrics.** We compare the compression effectiveness using the average number of bits required per pixel (bpp) for each output frame. We measure the compression quality using both automatic and human evaluations. Unlike traditional compression methods, our method does not reproduce the input image in a pixel-aligned manner but can faithfully reproduce facial motions and gestures. Metrics based on exact pixel alignments are ill-suited for measuring the quality of our output videos. We hence use the LPIPS perceptual similarity metric [96] for measuring compression quality [7].

As shown on the left side of Fig. 16, compared to the other neural talking-head synthesis methods, ours obtains better quality while requiring much lower bandwidth. This is because other methods send the full keypoints [68, 82] and Jacobians [68], while ours only sends the head pose and keypoint deformations. Compared with H.264 videos of the same quality, ours requires significantly lower bandwidth. For human evaluation, we show MTurk workers two videos side-by-side, one produced by H.264 and the other produced by our method's adaptive version. We then ask the workers to choose the video that they feel is of better quality. The preference scores are visualized on the right side of Fig. 16. Based on these scores, our compression method is comparable to the H.264 codec at a CRF value of 36, which means our adaptive and 20 keypoint scheme obtains $10.37\times$ and $6.5\times$ reduction in bandwidth compared to the H.264 codec, respectively. To handle challenging corner cases for our video conferencing system and out-of-distribution videos, we further develop a binary latent encoding network that can efficiently encode the residual at the expense of additional bandwidth, the details of which are in Appendix C.3.

## 6. Conclusion

In this work, we present a novel framework for neural talking-head video synthesis and compression. We show that by using our unsupervised 3D keypoints, we are able to decompose the representation into person-specific canonical keypoints and motion-related transformations. This decomposition has several benefits: By modifying the keypoint transformation only, we are able to generate free-view videos. By transmitting just the keypoint transformations, we can achieve much better compression ratios than existing methods. These features provide users a great tool for streaming live videos. By dramatically reducing the bandwidth and ensuring a more immersive experience, we believe this is an important step towards the future of video conferencing.

**Acknowledgements.** We thank Jan Kautz for his valuable comments throughout the development of the work. We thank Timo Aila, Koki Nagano, Sameh Khamis, Jaewoo Seo, and Xihui Liu for providing very useful feedback to shape our draft. We thank Henry Lin, Rochelle Pereira, Santanu Dutta, Simon Yuan, Brad Nemire, Margaret Albrecht, Siddharth Sharma, and Eric Ladenburg for their helpful suggestions in presenting our visualization results.

## References

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2StyleGAN: How to embed images into the StyleGAN latent space? In *ICCV*, 2019.

[2] Kfir Aberman, Mingyi Shi, Jing Liao, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Deep video-based performance cloning. *Computer Graphics Forum*, 2019.

[3] Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *CVPR*, 2020.

[4] Hadar Averbuch-Elor, Daniel Cohen-Or, Johannes Kopf, and Michael F Cohen. Bringing portraits to life. *TOG*, 2017.

[5] Aayush Bansal, Shugao Ma, Deva Ramanan, and Yaser Sheikh. Recycle-GAN: Unsupervised video retargeting. In *ECCV*, 2018.

[6] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, 1999.

[7] Yochai Blau and Tomer Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. In *ICML*, 2019.

[8] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017.

[9] Egor Burkov, Igor Pasechnik, Artur Grigorev, and Victor Lempitsky. Neural head reenactment with latent pose descriptors. In *CVPR*, 2020.

[10] Caroline Chan, Shiry Ginosar, Tinghui Zhou, and Alexei A Efros. Everybody dance now. In *ICCV*, 2019.

[11] Lele Chen, Ross K Maddox, Zhiyao Duan, and Chenliang Xu. Hierarchical cross-modal talking face generation with dynamic pixel-wise loss. In *CVPR*, 2019.

[12] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *CVPR*, 2018.

[13] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

[14] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[15] Yu Deng, Jiaolong Yang, Dong Chen, Fang Wen, and Xin Tong. Disentangled and controllable face image generation via 3D imitative-contrastive learning. In *CVPR*, 2020.

[16] Abdelaziz Djelouah, Joaquim Campos, Simone Schaub-Meyer, and Christopher Schroers. Neural inter-frame compression for video coding. In *ICCV*, 2019.

[17] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *TPAMI*, 2015.

[18] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *ECCV*, 2016.

[19] Ohad Fried, Ayush Tewari, Michael Zollhöfer, Adam Finkelstein, Eli Shechtman, Dan B Goldman, Kyle Genova, Zeyu Jin, Christian Theobalt, and Maneesh Agrawala. Text-based editing of talking-head video. *TOG*, 2019.

[20] Jiahao Geng, Tianjia Shao, Youyi Zheng, Yanlin Weng, and Kun Zhou. Warp-guided GANs for single-photo facial animation. *TOG*, 2018.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. In *NeurIPS*, 2014.

[22] Kuangxiao Gu, Yuqian Zhou, and Thomas S Huang. FLNet: Landmark driven fetching and learning network for faithful talking facial animation synthesis. In *AAAI*, 2020.

[23] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein GANs. In *NeurIPS*, 2017.

[24] Sungjoo Ha, Martin Kersner, Beomsu Kim, Seokjun Seo, and Dongyoung Kim. MarioNETte: Few-shot face reenactment preserving identity of unseen targets. In *AAAI*, 2020.

[25] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

[26] Zhihao Hu, Zhenghao Chen, Dong Xu, Guo Lu, Wanli Ouyang, and Shuhang Gu. Improving deep video compression by resolution-adaptive flow coding. In *ECCV*, 2020.

[27] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *ECCV*, 2018.

[28] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *TOG*, 2017.

[29] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017.

[30] Amir Jamaludin, Joon Son Chung, and Andrew Zisserman. You said that?: Synthesising talking faces from audio. *IJCV*, 2019.

[31] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[32] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018.

[33] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019.

[34] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020.

[35] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *TOG*, 2018.

[36] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 2016.

[37] Davis E. King. Dlib-ml: A machine learning toolkit. *JMLR*, 2009.

[38] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[39] Orest Kupyn, Volodymyr Budzan, Mykola Mykhailych, Dmytro Mishkin, and Jiří Matas. DeblurGAN: Blind motion deblurring using conditional adversarial networks. In *CVPR*, 2018.

[40] Glen G Langdon. An introduction to arithmetic coding. *IBM Journal of Research and Development*, 1984.

[41] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017.

[42] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *CVPR Workshop*, 2017.

[43] Jianping Lin, Dong Liu, Houqiang Li, and Feng Wu. M-lvc: Multiple frames prediction for learned video compression. In *CVPR*, 2020.

[44] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. Image inpainting for irregular holes using partial convolutions. In *ECCV*, 2018.

[45] Lingjie Liu, Weipeng Xu, Michael Zollhoefer, Hyeongwoo Kim, Florian Bernard, Marc Habermann, Wenping Wang, and Christian Theobalt. Neural rendering and reenactment of human actor videos. *TOG*, 2019.

[46] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *NeurIPS*, 2017.

[47] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *ICCV*, 2019.

[48] Ming-Yu Liu, Xun Huang, Jiahui Yu, Ting-Chun Wang, and Arun Mallya. Generative adversarial networks for image and video synthesis: Algorithms and applications. *Proceedings of The IEEE*, 2021.

[49] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In *NeurIPS*, 2016.

[50] Wen Liu, Zhixin Piao, Jie Min, Wenhan Luo, Lin Ma, and Shenghua Gao. Liquid Warping GAN: A unified framework for human motion imitation, appearance transfer and novel view synthesis. In *ICCV*, 2019.

[51] Steven R Livingstone and Frank A Russo. The ryerson audio-visual database of emotional speech and song (ravdess): A dynamic, multimodal set of facial and vocal expressions in north american english. *PloS one*, 13(5):e0196391, 2018.

[52] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *CVPR*, 2019.

[53] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Zhiyong Gao, and Ming-Ting Sun. Deep kalman filtering network for video compression artifact reduction. In *ECCV*, 2018.

[54] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018.

[55] Koki Nagano, Jaewoo Seo, Jun Xing, Lingyu Wei, Zimo Li, Shunsuke Saito, Aviral Agarwal, Jens Fursund, and Hao Li. paGAN: real-time avatars using dynamic textures. *TOG*, 2018.

[56] Yuval Nirkin, Yosi Keller, and Tal Hassner. FSGAN: Subject agnostic face swapping and reenactment. In *ICCV*, 2019.

[57] Kyle Olszewski, Zimo Li, Chao Yang, Yi Zhou, Ronald Yu, Zeng Huang, Sitao Xiang, Shunsuke Saito, Pushmeet Kohli, and Hao Li. Realistic dynamic facial textures from a single image using GANs. In *ICCV*, 2017.

[58] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[59] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *BMVC*, 2015.

[60] Albert Pumarola, Antonio Agudo, Aleix M Martinez, Alberto Sanfeliu, and Francesc Moreno-Noguer. GANimation: Anatomically-aware facial animation from a single image. In *ECCV*, 2018.

[61] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2015.

[62] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, 2016.

[63] Yurui Ren, Xiaoming Yu, Junming Chen, Thomas H Li, and Ge Li. Deep image spatial transformation for person image generation. In *CVPR*, 2020.

[64] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. *arXiv preprint arXiv:2008.00951*, 2020.

[65] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G Anderson, and Lubomir Bourdev. Learned video compression. In *ICCV*, 2019.

[66] Nataniel Ruiz, Eunji Chong, and James M. Rehg. Fine-grained head pose estimation without keypoints. In *CVPR Workshop*, 2018.

[67] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.

[68] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. In *NeurIPS*, 2019.

[69] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *CVPR*, 2019.

[70] Yang Song, Jingwen Zhu, Dawei Li, Andy Wang, and Hairong Qi. Talking face generation by conditional recurrent adversarial network. In *IJCAI*, 2019.

[71] Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing Obama: learning lip sync from audio. *TOG*, 2017.

[72] Ying Tai, Jian Yang, Xiaoming Liu, and Chunyan Xu. MemNet: A persistent memory network for image restoration. In *ICCV*, 2017.

[73] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. In *ICLR*, 2017.

[74] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zöllhofer, and Christian Theobalt. StyleRig: Rigging StyleGAN for 3d control over portrait images. In *CVPR*, 2020.

[75] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *TOG*, 2019.

[76] Justus Thies, Michael Zollhöfer, Matthias Nießner, Levi Valgaerts, Marc Stamminger, and Christian Theobalt. Real-time expression transfer for facial reenactment. *TOG*, 2015.

[77] Justus Thies, Michael Zollhofer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2Face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016.

[78] Tong Tong, Gen Li, Xiejie Liu, and Qinquan Gao. Image super-resolution using dense skip connections. In *ICCV*, 2017.

[79] Yi-Hsuan Tsai, Ming-Yu Liu, Deqing Sun, Ming-Hsuan Yang, and Jan Kautz. Learning binary residual representations for domain-specific video streaming. In *AAAI*, 2018.

[80] Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popovic. Face transfer with multilinear models. *TOG*, 2005.

[81] Konstantinos Vougioukas, Stavros Petridis, and Maja Pantic. Realistic speech-driven facial animation with GANs. *IJCV*, 2019.

[82] Ting-Chun Wang, Ming-Yu Liu, Andrew Tao, Guilin Liu, Jan Kautz, and Bryan Catanzaro. Few-shot video-to-video synthesis. In *NeurIPS*, 2019.

[83] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018.

[84] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *CVPR*, 2018.

[85] Xintao Wang, Kelvin CK Chan, Ke Yu, Chao Dong, and Chen Change Loy. EDVR: Video restoration with enhanced deformable convolutional networks. In *CVPR-W*, 2019.

[86] Olivia Wiles, A Sophia Koepke, and Andrew Zisserman. X2Face: A network for controlling face generation using images, audio, and pose codes. In *ECCV*, 2018.

[87] Chao-Yuan Wu, Nayan Singhal, and Philipp Krahenbuhl. Video compression through image interpolation. In *ECCV*, 2018.

[88] Wayne Wu, Yunxuan Zhang, Cheng Li, Chen Qian, and Chen Change Loy. ReenactGAN: Learning to reenact faces via boundary transfer. In *ECCV*, 2018.

[89] Wei Xiong, Jiahui Yu, Zhe Lin, Jimei Yang, Xin Lu, Connelly Barnes, and Jiebo Luo. Foreground-aware image inpainting. In *CVPR*, 2019.

[90] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, 2018.

[91] Ren Yang, Mai Xu, Zulin Wang, and Tianyi Li. Multi-frame quality enhancement for compressed video. In *CVPR*, 2018.

[92] Egor Zakharov, Aleksei Ivakhnenko, Aliaksandra Shysheya, and Victor Lempitsky. Fast bi-layer neural synthesis of one-shot realistic head avatars. In *ECCV*, 2020.

[93] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *ICCV*, 2019.

[94] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. Learning pyramid-context encoder network for high-quality image inpainting. In *CVPR*, 2019.

[95] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[96] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[97] Xi Zhang, Xiaolin Wu, Xinliang Zhai, Xianye Ben, and Chengjie Tu. Davd-net: Deep audio-aided video decompression of talking heads. In *CVPR*, 2020.

[98] Yuting Zhang, Yijie Guo, Yixin Jin, Yijun Luo, Zhiyuan He, and Honglak Lee. Unsupervised discovery of object landmarks as structural representations. In *CVPR*, 2018.

[99] Hang Zhou, Jihao Liu, Ziwei Liu, Yu Liu, and Xiaogang Wang. Rotate-and-render: Unsupervised photorealistic face rotation from single-view images. In *CVPR*, 2020.

[100] Hang Zhou, Yu Liu, Ziwei Liu, Ping Luo, and Xiaogang Wang. Talking face generation by adversarially disentangled audio-visual representation. In *AAAI*, 2019.

[101] Yipin Zhou, Zhaowen Wang, Chen Fang, Trung Bui, and Tamara L Berg. Dance dance generation: Motion transfer for internet videos. In *ICCV Workshop*, 2019.

[102] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[103] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *NeurIPS*, 2017.

# A. Additional Network and Training Details

Here, we present the architecture of our neural talking-head model. We also discuss the training details.

## A.1. Network architectures

The implementation details of the networks in our model are shown in Fig. 12 and described below.

**Appearance feature extractor $F$.** The network extracts 3D appearance features from the source image. It consists of a number of downsampling blocks, followed by a convolution layer that projects the input 2D features to 3D features. We then apply a number of 3D residual blocks to compute the final 3D features $f_s$.

**Canonical keypoint detector $L$.** The network takes the source image and applies a U-Net style encoder-decoder to extract canonical keypoints. Since we need to extract 3D keypoints, we project the encoded features to 3D through a $1 \times 1$ convolution. The output of the $1 \times 1$ convolution is the bottleneck of the U-Net. The decoder part of the U-Net consists of 3D convolution and upsampling layers.

**Head pose estimator $H$ and expression deformation estimator $\Delta$.** We adopt the same architecture as in Ruiz *et al*. [66]. It consists of a series of ResNet bottleneck blocks, followed by a global pooling to remove the spatial dimension. Different linear layers are then used to estimate the rotation angles, the translation vector, and the expression deformations. The full angle range is divided into 66 bins for rotation angles, and the network predicts which bin the target angle is in. The estimated head pose and deformations are used to transform the canonical keypoints to obtain the source or driving keypoints.

**Motion field estimator $M$.** After the keypoints are predicted, they are used to estimate warping flow maps. We generate a warping flow map $w_k$ based on the $k$-th keypoint using the first-order approximation [68]. Let $p_d$ be a 3D coordinate in the feature volume of the driving image $d$. The $k$-th flow field maps $p_d$ to a 3D coordinate in the 3D feature volume of the source image $s$, denoted by $p_s$, by:

$$w_k : R_s R_d^{-1}(p_d - x_{d,k}) + x_{s,k} \mapsto p_s. \tag{4}$$

This builds a correspondence between the source and driving.

Using the flow field $w_k$ obtained from the $k$-th keypoint pair, we can warp the source feature $f_s$ to construct a candidate warped volume, $w_k(f_s)$. After we obtain the warped source features $w_k(f_s)$ using all $K$ flows, they are concatenated together and fed to a 3D U-Net to extract features. Then a softmax function is employed to obtain the flow composition mask $m$, which consists of $K$ 3D masks, $\{m_1, m_2, ..., m_K\}$. These maps satisfy the constraints that $\sum_k m_k(p_d) = 1$ and $0 \leqslant m_k(p_d) \leqslant 1$ for all $p_d$. These $K$ masks are then linearly combined with the $K$ warping flow maps, $w_k$'s, to construct the final warping map $w$ by $\sum_{k=1}^{K} m_k(p_d) w_k(p_d)$. To handle occlusions caused by the warping, we also predict a 2D occlusion mask $o$, which will be inputted to the generator $G$.

**Generator $G$.** The generator takes the warped 3D appearance features $w(f_s)$ and projects them back to 2D. Then, the features are multiplied with the occlusion mask $o$ obtained from the motion field estimator $M$. Finally, we apply a series of 2D residual blocks and upsamplings layers to obtain the final image.

## A.2. Losses

We present details of the loss terms in the following.

**Perceptual loss $\mathcal{L}_P$.** We use the multi-scale implementation introduced by Siarohin *et al*. [68]. In particular, a pre-trained VGG network is used to extract features from both the ground truth and the output image, and the $L_1$ distance between the features is computed. Then both images are downsampled, and the same VGG network is used to extract features and compute the $L_1$ distance again. This process is repeated 3 times to compute losses at multiple image resolutions. We use layers `relu_1_1`, `relu_2_1`, `relu_3_1`, `relu_4_1`, `relu_5_1` of the VGG19 network with weights $0.03125, 0.0625, 0.125, 0.25, 1.0$, respectively. Moreover, since we are synthesizing face images, we also compute a single-scale perceptual loss using a pre-trained face VGG network [59]. These losses are then summed together to give the final perceptual loss.

**GAN loss $\mathcal{L}_G$.** We adopt the same patch GAN implementation as in [58, 84], and use the hinge loss. Feature matching [84] loss is also adopted to stabilize training. We use single-scale discriminators for training $256 \times 256$ images, and two-scale discriminators [84] for $512 \times 512$ images.

**Equivariance loss $\mathcal{L}_E$.** This loss ensures the consistency of estimated keypoints [68, 98]. In particular, let the original image be $d$ and its detected keypoints be $x_d$. When a known spatial transformation $\mathbf{T}$ is applied on image $d$, the detected keypoints $x_{\mathbf{T}(d)}$ on this transformed image $\mathbf{T}(d)$ should be transformed in the same way. Based on this observation, we minimize the $L_1$ distance $\|x_d - \mathbf{T}^{-1}(x_{\mathbf{T}(d)})\|_1$. Affine transformations and randomly sampled thin plate splines are used to perform the transformation. Since all these are 2D transformations, we project our 3D keypoints to 2D by simply dropping the $z$ values before computing the losses.

**Keypoint prior loss $\mathcal{L}_L$.** As described in the main paper, we penalize the keypoints if the distance between any pair of them is below some threshold $D_t$, or if the mean depth value deviates from a preset target value $z_t$. In other words,

$$\mathcal{L}_L = \sum_{i=1}^{K} \sum_{j=1}^{K} \max(0, D_t - \|x_{d,i} - x_{d,j}\|_2^2) + \|Z(x_d) - z_t\| \tag{5}$$

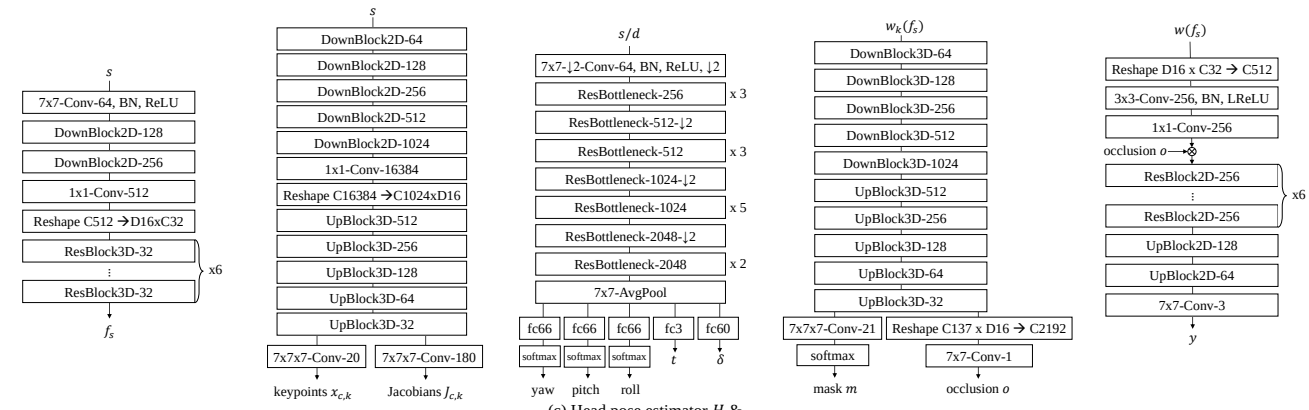



Figure 12: Architectures of individual components in our model. For the building blocks, please refer to Fig. 13

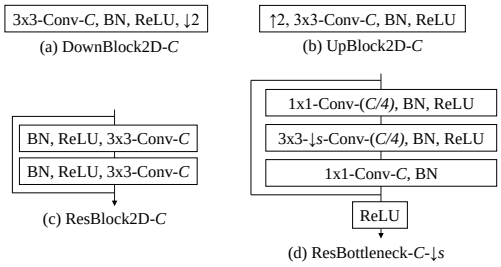



Figure 13: Building blocks of our model. For the 3D counterparts, we simply replace 2D convolutions with 3D convolutions in the blocks.

where $Z(\cdot)$ extracts the mean depth value of the keypoints. This ensures the keypoints are more spread out and used more effectively. We set $D_t$ to 0.1 and $z_t$ to 0.33 in our experiments.

**Head pose loss $\mathcal{L}_H$.** We compute the $L_1$ distance between the estimated head pose $R_d$ and the one predicted by a pre-trained pose estimator $\bar{R}_d$, which we treat as ground truth. In other words, $\mathcal{L}_H = \|R_d - \bar{R}_d\|_1$, where the distance is computed as the sum of differences of the Euler angles.

**Deformation prior loss $\mathcal{L}_\Delta$.** Since the expression deformation $\Delta$ is the deviation from the canonical keypoints, their magnitude should not be too large. To ensure this, we put a loss on their $\mathcal{L}_1$ norm: $\mathcal{L}_\Delta = \|\delta_{d,k}\|_1$.

**Final loss** The final loss is given by:

$$\begin{aligned}\mathcal{L} =&\lambda_P \mathcal{L}_P(d, y) + \lambda_G \mathcal{L}_G(d, y) + \lambda_E \mathcal{L}_E(\{x_{d,k}\}) + \\ &\lambda_L \mathcal{L}_L(\{x_{d,k}\}) + \lambda_H \mathcal{L}_H(R_d, \bar{R}_d) + \lambda_\Delta \mathcal{L}_\Delta(\{\delta_{d,k}\})\end{aligned} \tag{6}$$

where $\lambda$'s are the weights and are set to $10, 1, 20, 10, 20, 5$ respectively in our implementation.

## A.3. Optimization

We adopt the ADAM optimizer [38] with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is set to $0.0002$. We apply Spectral Norm [54] to all the layers in both the generator and the discriminator. We use synchronized BatchNorm for the generator. Training is conducted on an NVIDIA DGX1 with 8 32GB V100 GPUs.

We adopt a coarse-to-fine approach for training. We first train our model on $256 \times 256$ images for 100 epochs. We then finetune on $512 \times 512$ images for another 10 epochs.

# B. Additional Experiment Details

## B.1. Datasets

We use the following datasets in our evaluations.

**VoxCeleb2 [13].** The dataset contains about 1M talking-head videos of different celebrities. We follow the training and test split proposed in the original paper. where we use 280K videos with high bit-rates to train our model. We report our results on the validation set, which contains about 36k videos.

**TalkingHead-1KH.** We compose a dataset containing about 1000 hours of videos from various sources. A large portion of them is from the YouTube website with the creative common license. We also use videos from the Ryerson audio-visual dataset [51] as well as a set of videos that we recorded with the permission from the subject ourselves. We only use videos whose resolution and bit-rate are both high. We call this dataset *TalkingHead-1KH*. The videos in the TalkingHead-1KH are in general with higher resolutions and better image quality than those in the VoxCeleb2.

## B.2. Metrics

We use a set of metrics to evaluate a talking-head synthesis method. We use $L_1$, PSNR, SSIM, and MS-SSIM for

Table 4: Cross-identity transfer using relative motion.

| Method | VoxCeleb2 FID↓ | VoxCeleb2 CSIM↑ | TalkingHead-1KH FID↓ | TalkingHead-1KH CSIM↑ |
|---|---|---|---|---|
| fs-vid2vid [82] | 48.48 | 0.928 | 44.83 | 0.955 |
| FOMM [68] | 48.91 | 0.954 | 42.26 | 0.961 |
| **Ours** | **46.43** | **0.960** | **41.25** | **0.964** |

quantifying the faithfulness of the recreated videos. We use FID to measure how close is the distribution of the recreated videos to that of the original videos. We use AKD to measure how close the facial landmarks extracted by an off-the-shelf landmark detector from the recreated video are to those in the original video. In the following, we discuss the implementation details of these metrics.

$L_1$**.** We compute the average $L_1$ distance between generated and real images.

**PSNR** measures the image reconstruction quality by computing the mean squared error (MSE) between the ground truth and the reconstructed image.

**SSIM/MS-SSIM.** SSIM measures the structural similarity between patches of the input images. Therefore, it is more robust to global illumination changes than PSNR, which is based absolute errors. MS-SSIM is a multi-scale variant of SSIM that works on multiple scales of the images and has been shown to correlate better with human perception.

**FID [25]** measures the distance between the distributions of synthesized and real images. We use the pre-trained InceptionV3 network to extract features from both sets of images and estimate the distance between them.

**Average keypoint distance (AKD).** We use a facial landmark detector [37] to detect landmarks of real and synthesized images and then compute the average distance between the corresponding landmarks in these two images.

### B.3. Relative motion transfer

For cross-identity motion transfer results in our experiment section, we transfer absolute motions in the driving video. For completeness, we also report quantitative comparisons using relative motion proposed in [68] in Table 4. As can be seen, our method still performs the best.

### B.4. Ablation study

We perform the following ablation studies to verify the effectiveness of our several important design choices.

**Two-step vs. direct keypoint prediction.** We estimate the keypoints in an image by first predicting the canonical keypoints and then applying the transformation and the deformations. To compare this approach with direct keypoint location prediction, we train another network that directly predicts the final source and driving keypoints in the image. In particular, the keypoint detector $L$ directly predicts the final source and driving keypoints in the image instead of the canonical ones, and there is no pose estimator $H$ and deformation estimator $\Delta$. Since there is no pose estimator, we do not need any pose supervision (i.e., head pose loss) for this direct prediction network. Note that while this model is only slightly inferior to our final (two-step) model quantitatively, it has no pose control for the output video since the head pose is no longer estimated, so a major feature of our method would be lost.

**3D vs. 2D warping.** We generate 3D flow fields from the estimated keypoints to warp 3D features. Another option is to project the keypoints to 2D, estimate a 2D flow field, and extract 2D features from the source image. The estimated 2D flow filed is then used to warp 2D image features.

**Number of keypoints.** We show that our approach's output quality is positively correlated with the number of keypoints.

As can be seen in Table 5, our model works better than all the other alternatives on all of the performance metrics.

Table 5: Ablation study. Compared with all the other alternatives, our model (the preferred setting) works the best.

| Method | L1 | PSNR | SSIM | MS-SSIM | FID | AKD |
|---|---|---|---|---|---|---|
| Direct pred. | 10.84 | 24.00 | 0.80 | 0.83 | 58.55 | 4.26 |
| **Ours (20 kp)** | **10.67** | **24.20** | **0.81** | **0.84** | **52.08** | **3.74** |
| 2D Warp | 11.64 | 23.38 | 0.79 | 0.82 | 58.75 | 4.20 |
| **Ours (20 kp)** | **10.67** | **24.20** | **0.81** | **0.84** | **52.08** | **3.74** |
| 10 kp | 11.49 | 23.36 | 0.79 | 0.82 | 56.27 | 4.31 |
| 15 kp | 11.35 | 23.53 | 0.79 | 0.82 | 54.36 | 4.50 |
| **Ours (20 kp)** | **10.67** | **24.20** | **0.81** | **0.84** | **52.08** | **3.74** |

Source image Driving frame Synthesized result

Figure 14: Example failure cases. Our method still struggles when there are occluders such as hands in the image.

### B.5. Failure cases

While our model is in general robust to different situations, it cannot handle large occlusions well. For example, when the face is occluded by the person's hands or other objects, the synthesis quality will degrade, as shown in Fig. 14

Table 6: Size of per-frame metadata in bytes for talking-head methods before and after arithmetic compression.

| Method | Before Compression | After Compression | | | |
|---|---|---|---|---|---|
| | | Mean | Min | Max | Median |
| fs-vid2vid [82] | 504 | 231.42 | 158 | 599 | 238 |
| FOMM [68] | 240 | 171.09 | 159 | 210 | 169 |
| Ours (20 kp) | 132 | 84.44 | 78 | 104 | 84 |
| Ours (adaptive) | 81.16 | 53.03 | 25 | 102 | 45 |

### B.6. Canonical keypoints for face recognition

Our canonical keypoints are formulated to be independent of the pose and expression change. They should only contain a person's geometry signature, such as the shapes of face, nose, and eyes. To verify this, we conduct an experiment using the canonical keypoints for face recognition.

We extract canonical keypoints from 384 identities in the VoxCeleb2 [13] dataset to form a training set. For each identity, we also pick a different video of the same identity to form the test set. The training and test videos of the same subject have different head poses and expressions. A face recognition algorithm would fail if it could not filter out pose and expression information. To prove our canonical keypoints are independent to poses and expressions, we apply a simple nearest neighbor classifier using our canonical keypoints for the face recognition task.

Overall, our canonical keypoints reaches an accuracy of 0.070, while a random guess has an accuracy of 0.0026 (Ours is 27× better than the random guess.). On the other hand, a classifier using the off-the-shelf dlib landmark detector only achieves an accuracy of 0.013, which means our keypoints are 5× more effective for face recognition.

## C. Additional Video Conferencing Details

### C.1. Entropy encoder

We represent each rotation angle, translation, and deformation value as an fp16 floating-point number. Each number consumes two bytes. Naively transmitting the $3K + 6$ floating numbers will result in transmitting $6K + 12$ bytes. We adopt arithmetic coding [40] to encode the $3K + 6$ numbers. Arithmetic coding is one kind of entropy coding. It assigns different codeword lengths to different symbols based on their frequencies. The symbol that appears more often will have a shorter code.

We first apply the driving image encoder to a validation set of 127 videos that are not included in the test set. Each frame will give us $6K + 12$ bytes. We treat each of the bytes separately and build a frequency table for each byte. This gives us $6K + 12$ frequency tables. When encoding the test set, we encode each byte using the associated frequency table learned from the validation set. This results in a varying-length representation that is much smaller than $6K + 12$ bytes on average.

Table 6 shows the sizes of the per-frame metadata in bytes that needs to be transmitted for various talking-head methods before and after performing the arithmetic compression for an image size of 512×512. Our adaptive scheme requires a per-frame metadata size of 53.03 B, which corresponds to $(53.03 \times 8/512^2) = 0.001618$ bits per pixel.

### C.2. Adaptive number of keypoints

Our basic model uses a fixed number of keypoints during training and inference. However, on a video call, it is advantageous to adaptively change the number of keypoints used to accommodate varying bandwidth and internet connectivity. We devise a scheme where our synthesis model can dynamically use a smaller number of keypoints for reconstruction. This is based on the intuition that not all of the images are of the same complexity. Some just require fewer keypoints. Using fewer keypoints, we can reduce the bandwidth required for video conferencing because we just need to send a subset of $\delta_{d,k}$'s. To train a model that supports a varying number of keypoints, we randomly choose an index into the array of ordered keypoints, and dropout all values from that index till the end of the array. This dropout percentage ranges from 0% to 75%. This scheme is also helpful when the available bandwidth suddenly drops.

### C.3. Binary encoding of the residuals

When the contents of the video being streamed change drastically, *e.g.* when new objects are introduced into the video or the person changes, it becomes necessary to update the source frame being used to perform the talking-head synthesis. This can be done by sending a new image to the receiver. We also devise a more efficient scheme to encode and send only the residual between the ground truth frame and the reconstructed frame, instead of an entirely new source image. To encode a residual image of size $512 \times 512$, we use a 3-layer network with convolutions of kernel size 3, stride 2, and 32 channels, similar to the network proposed by Tsai *et al*. [79]. We compute the sign of the latent code of size $32 \times 64 \times 64$ to obtain binary latent codes. The decoder also consists of 3 convolutional layers of 128 filters and uses the pixel shuffle layer to perform upsampling. After arithmetic coding, the binary latent code requires 13.40 KB on average. Note that we do not need to send the encoded binary residual every frame. We just need to send it when the current source image is not good enough to reconstruct the current driving image. In the receiver side, we will use the encoded residual to improve the quality of the reconstructed image. The reconstructed image will become the new source image for decoding future frames using the encoded rotation, translation, and deformations. Example improvements after adding the residual are shown in Fig. 15.


Figure 15: Fixing artifacts in compressed images using our binary residual encoder. We are able to fix artifacts caused due to the introduction of new objects, changes in background, as well as extreme poses by transmitting the residuals encoded as binary values. Each residual binary latent code requires only about 13.40 KB and can replace sending new source images.

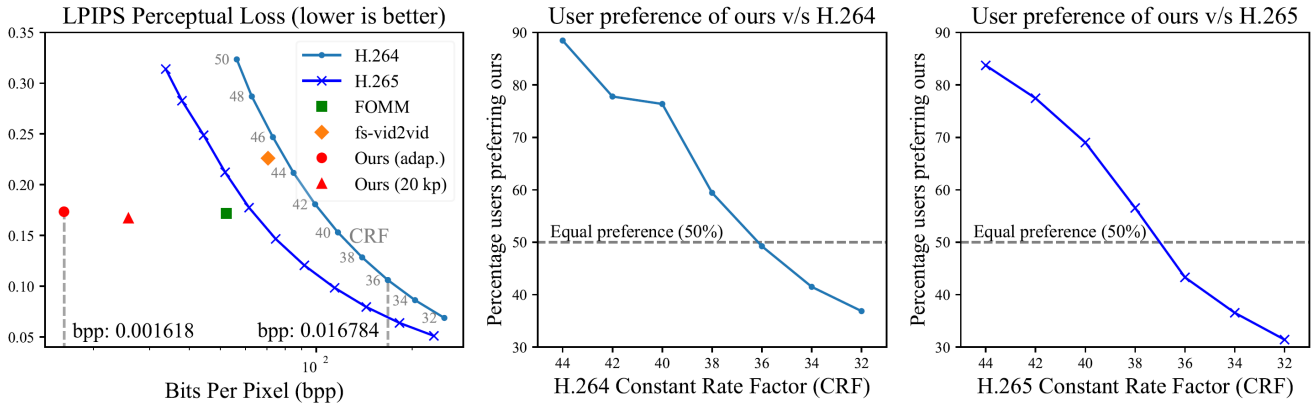



Figure 16: Automatic and human evaluations for video compression. Ours requires much lower bandwidth than the H.264 and H.265 codecs and other talking-head synthesis methods thanks to our keypoint decomposition and adaptive scheme.

### C.4. Dataset

For testing, we collect a set of high-resolution talking-head videos from the web. We ensure that the head is of size at least 512×512 pixels and manually check each video to ensure its quality. This results in a total of 222 videos, with a mean of 608 frames, a median of 661 frames, and a min and max of 20 and 1024 frames, respectively.

### C.5. Additional experiment results

In Fig. 16(a), we show the achieved LPIPS score by our approach under the adaptive setting (red circle), our approach under the 20 keypoint setting (red triangle), FOMM (green square), fs-vid2vid (orange diamond), H.264, and H.265 using different bpp rates. We observe that our method requires much lower bandwidth than the competing methods.

**User study.** Here, we describe the details of our user study. We use the Amazon Mechanical Turk (MTurk) platform for the user preference score. A worker needs to have a lift-time approval rate greater than 98 to be qualified for our study. This means that the requesters approve 98% of his/her task assignments. For comparing two competing methods, we generate 222 videos from each method. We show the corresponding pair of videos from two competing methods to three different MTurk workers and ask them to select which one has better visual quality. This gives 666 preference scores for each comparison. We report the average preference score achieved by our method. We compare our adaptive approach to both H.264 and H.265. The user preference scores of our approach when compared to H.264 and H.265 are shown in Fig. 16(b) and (c), respectively. We found that our approach renders comparable performance to H.264 with CRF value 36. For H.265, our approach is comparable to CRF value 37. Our approach was able to achieve the same visual quality using a much lower bit-rate.