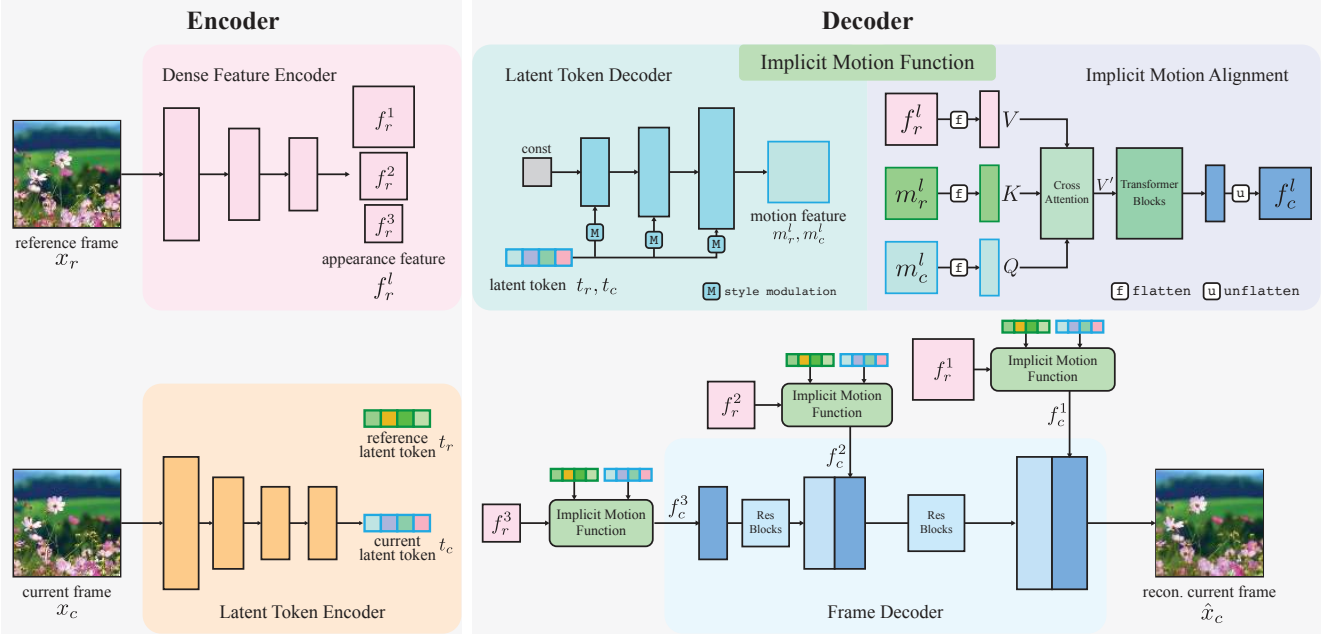

Figure 3. The illustration of our framework: a) the encoder contains the dense feature encoder E_F and the latent token encoder E_T; b) the decoder contains the implicit motion function IMF and the feature decoder D_F. IMF includes the latent token decoder IMF_D and the implicit motion alignment IMF_A. The motion features m_r^l and m_c^l are produced by IMF_D from t_r and t_c, independently.

the tokens pair (t_c, t_r) extracted respectively by E_T from x_c and x_r. The decoder D_F aims to faithfully reconstruct the $\hat{x}_c$ with (t_c, t_r, f_r). The pipeline can be formulated as:

$$\begin{aligned} &\hat{x}_c = D_F(IMF(t_c, t_r, f_r)) \quad \text{with} \\ &t_c = E_T(x_c), \quad t_r = E_T(x_r), \quad f_r = E_F(x_r). \end{aligned} \tag{1}$$

Note that the reference frame x_r is not necessarily selected from the input sequence $\{x_c\}$. Taking the talking head generation as an example, our pipeline can support the reference having a different ID with the input. Furthermore, different from the former works such as optical flow, the correlation modeling by (t_c, t_r) does not require the reference input and the current input has the same dimension, not even the same modality.

In our framework, to learn sparse yet essential information, we will constrain the information flow in t_c and t_r. The optimization target can be formulated as:

$$\begin{aligned} \min \quad & L(\hat{x}_c, x_c) \\ \text{s.t.} \quad & |t| < \epsilon, \; t \in \{t_c, t_r\}, \end{aligned} \tag{2}$$

where L is the function that measures the similarity between $\hat{x}_c$ and x_c. $|t|$ is the size of the latent tokens, and ϵ is the size limitation. To achieve this objective, we specially design the IMF at decoder to obtain high-fidelity $\hat{x}_c$. A detailed illustration of our full pipeline is shown in Fig. 3. The remaining part of this section will be organized as follows: In Sec. 3.2, we briefly introduce the encoder details of E_F and E_T. In Sec. 3.3, we describe the proposed IMF. Sec. 3.4 demonstrates the editability of the latent tokens on talking head generation. In Sec. 3.5, we delve into the design and underlying rationale of our proposed framework.

3.2. Dense Feature & Latent Token Encoder

Dense Feature Encoder. Given a reference frame $x_r \in \mathbb{R}^{H\times W\times 3}$, the dense feature encoder E_F extracts multi-scale feature $f_r^l \in \mathbb{R}^{h_l\times w_l\times d_l}, l \in \{1, L\}$, where l is the layer index and L is number of layers. h_l, w_l is spatial size of the feature. d_l is the depth dimension of the feature. The E_F is only performed on the reference frame, which means the features f_r^l are shared across the whole video if the reference frame is constant.

Latent Token Encoder. The latent token encoder E_T encodes the reference frame and the current frame to t_r and t_c, independently. It maps the space $\mathbb{R}^{H\times W\times 3} \Rightarrow \mathbb{R}^{\mathbf{d}}$. Similar to f_r, when the reference frame is constant, t_r is also shared across the entire video. When $\mathbf{d} = K \times 2$, where K is the number of keypoints, our tokens can be viewed as the explicit coordinates representations from FOMM [57]. However, due to the limited information expressed by coordinates, the expressive capability of this kind of representation is limited, especially in terms of semantic integrity, *i.e.* the completeness to faithfully reconstruct the original frame. By contrast, we choose $\mathbf{d} = 1 \times d_m$, a latent vector to ensure both completeness and compactness. We further discuss in Sec. 3.5 the choice between explicit and implicit representations of t_c and reveal the pros and cons of each in the ablation experiments.

3.3. Implicit Motion Function

With the latent token t_r and the appearance feature f_r extracted from the reference frame, the target of IMF is to obtain the appearance feature f_c of the current frame from its latent token t_c. Specifically, the IMF is mainly composed of two parts. First, we design a *latent token decoder* IMF_D module to transform the highly compact latent tokens into spatially aligned motion features that correspond with the frame. Subsequently, we utilize *implicit motion alignment* IMF_A module to align and refine the appearance feature of the reference frame to the current frame.

Latent Token Decoder. The IMF_D decodes the compact tokens t_r, t_c to multiple motion features $m_r^l \in \mathbb{R}^{h_l \times w_l \times d_l}$ and $m_c^l \in \mathbb{R}^{h_l \times w_l \times d_l}$ where l is the layer index. The "style modulation" utilizes Weight Modulation and Demodulation technique of StyleGAN2 [25], scales the convolution weights with the latent token, and normalizes them to unit standard deviation. Owing to the varying granularities, our latent tokens effectively compress multi-scale information, serving as a comprehensive representation. Furthermore, the fully implicit nature of our representation allows for flexible adjustment of the latent token dimension to accommodate different scenarios. Different from keypoint-based explicit representation [41, 57, 66], where the motion features are Gaussian heatmaps converted from the keypoints, our design enjoys better scalability and capability due to the latent token being directly learned by the encoder instead of coordinates with a limited value range.

Implicit Motion Alignment. As shown in Fig.3, with the motion features m_r^l and m_c^l, the IMF_A will align the reference appearance features f_r^l to the current frame. The IMF_A contains two parts including the cross-attention [65] and transformer. The cross-attention module is implemented with scaled dot-product cross-attention. This attention module takes motion features m_c^l, m_r^l, and appearance features f_r^l as Q, K, and V, respectively. The features are first flattened, then the positional embeddings P_q, P_k are added to the queries and keys, We compute the dot products of the queries with keys, divide each by $\sqrt{d_k}$, and apply a `softmax` function to obtain the weights on the values. Then the output-aligned values are computed through matrix multiplication. With the aligned values V', we further refine them using multi-head self-attention and feed-forward network-based Transformer blocks [65], and finally obtain the appearance features f_c^l of the current frame.

3.4. Token Manipulation

In contrast to the explicit optical flow method, the implicit representation offers a distinct advantage in terms of editability. As the latent token is not task-specific, for a new controllable generation task, the decoder can be fixed and just train a small adapter with a small cost. Taking talking head generation as an example, in the training of the latent space, the full network is only trained to reconstruct the current frame. After the latent space is trained, we can edit learned tokens through another independent token manipulation network. Formally, with an editing module ψ, source frame x_s, and control condition h, we can rewrite Eq. 1 to obtain the edited feature map $\tilde{f}_s$ as:

$$\tilde{f}_s = IMF\big(\psi(t_s|h), t_s, f_s\big). \tag{3}$$

Passing $\tilde{f}_s$ to the frame decoder trained before, we can get the edited frame $\tilde{x}_s$. In our experiments, we implement ψ with two MLP encoders to encode the source token and control condition (*e.g.*, the 3DMM face coefficients), and one MLP decoder to output the edited token.

3.5. Discussion

Explicit or Implicit. Based on the foundational works MonkeyNet [56], FOMM [57], and MRAA [58], introduced by Siarohin *et al.*, several works employ explicit keypoints [17, 66], landmarks [11, 81], regions [58] or vectors [67] as intermediate representations for direct optical flow estimation. Given the pivotal role of optical flow in elucidating relationships between video frames, this explicit approach has become the predominant methodology in video modeling. However, explicit representations are inherently constrained. Optical flow, being a dense representation, poses optimization challenges due to the per-pixel operations in grid sampling. While keypoints offer sparsity, they compromise on detailed motion information. Additionally, keypoints are effective for quasi-rigid structures like human heads but falter in general video tasks due to the diverse shapes and motion patterns, which are difficult to encapsulate using explicit methods. Another notable drawback of keypoints is their limited scalability, where increased point usage for finer motion details often leads to redundancy and artifact-prone outputs.

Encoder-centric (EC) or Decoder-centric (DC). Here, we discuss the placement of correlation modeling, whether it is within the encoder or the decoder. Most existing video compression and video modeling methods belong to EC, which requires a complex design for encoder. For example, the functioning of a video codec encoder is contingent on the decoder output to obtain the prediction or context. Such intertwined framework necessitates meticulous tuning. For instance, the previous SOTA codec DCVC-DC [31, 55] incorporates upwards of ten distinct complicate strategies during training phases. In contrast, recent advancements in Large Language Models (LLMs) [48] usually utilize a DC architecture to support diverse tasks. This paradigm shift prompts reconsideration of similar frameworks for video codec. Actually, Wyner-Ziv coding [71] has given the theoretical support that the rate of DC framework can be identical with that of the EC framework. Our IMF design aligns with successful DC-based LLMs, achieving significant enhancements in video modeling.