Spline-based Transformers Supplementary Material

Prashanth Chandran[1*], Agon Serifi[2,3*], Markus Gross[1,3], and Moritz Bächer[2]

[1] DisneyResearch|Studios, Switzerland
[2] Disney Research, Switzerland
{prashanth.chandran, moritz.baecher}@disneyresearch.com
[3] ETH Zurich, Switzerland
{agon.serifi, grossm}@inf.ethz.ch

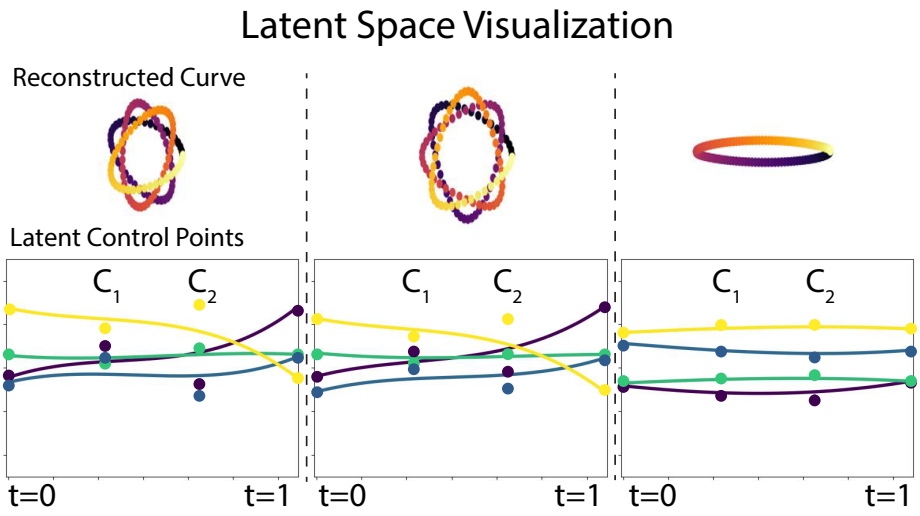

Fig. 1: We show Hypotrochoid curves generated by our method along with their corresponding latent control points (shown as dots) and the latent trajectories obtained using a cubic Bézier interpolator. Curves that look similar in Cartesian space (Cols 1, 2) seem to have similar latent controls and trajectories, while smoother curves (Col 3) seem to have smoother latent trajectories.

1 More About Control Points

Role In Downstream Applications In our work, we evaluate Spline-based Transformers mainly from the point of view of reconstruction tasks as they are a common proxy task for pre-training feature backbones for various downstream

* equal contribution.

applications (such as classification, segmentation, etc). Even when trained to reconstruct input sequences, Spline-based Transformers already seem to capture the similarities and differences between the input sequences in their latent control points. In Fig. 1, we show the predicted control points of two similar-looking curves (Cols 1, 2) alongside a different curve (Col 3) with their corresponding latent trajectories in the second row. Complex curves appear to have more non-linear latent trajectories (Cols 1, 2), while the smoother curve (Col 3) has a more linear latent trajectory. While further experimentation is certainly required, our early results indicate that Spline-based Transformers will find use in various other applications, including representation learning, classification *etc*.

Latent Control Manipulation The number of control tokens/points depends on the type of B-Spline (cardinal B-Splines, cubic Bézier, etc.) used. So far we have used cubic Bézier splines, which are defined by four control tokens across all data modalities. These control points determine the trajectory of the input sequence in the latent space of a Spline-based Transformer. For cubic Bézier splines, the first and the last latent control points determine the start and end of the latent trajectory, while the second C_1 and third C_2 control points define the shape of the latent spline. Fig. 2 demonstrates the effect of modifying the control point C_2 on 2D Hypotrochoids and the effect this has on reconstruction.

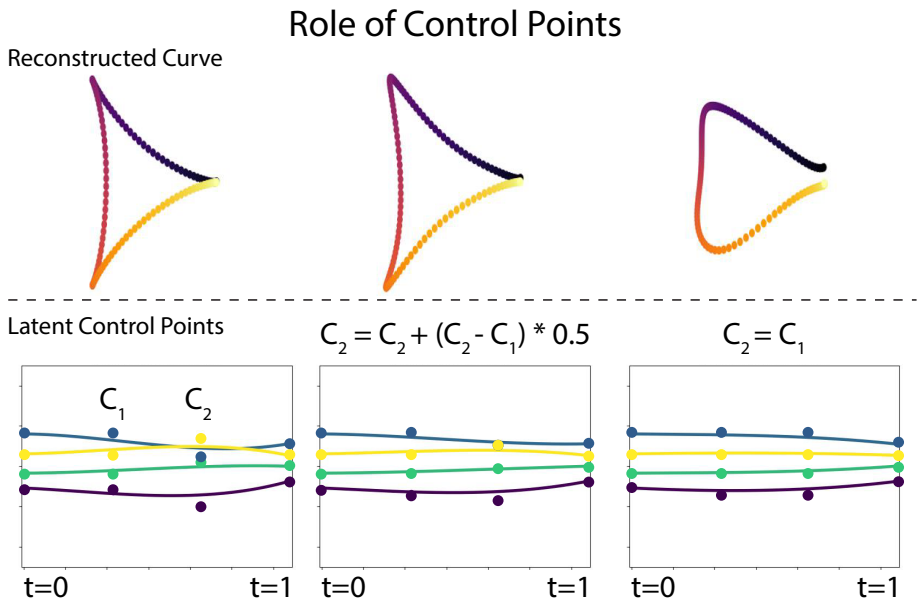

Fig. 2: Modifying the control points alters the latent trajectories according to the chosen B-Spline (cubic Bézier in our case). Here we incrementally modify control point C_2 to be closer to C_1 and visualize the change in the latent space. The corresponding reconstructed curve is shown in the top row.

2 Implementation Details

In the following, we present the implementation details for our Spline-based Transformer to help with reproducibility. In Tab. 1, we enumerate the various hyperparameters used in building the Spline-based Transformer for each of our experiments.

Algorithm 1: Spline-based Transformer

Input: Sequence x of dimensions ($seq\_len \times data\_dim$).
Param: Number of control points $controls$. Dimension of spline d.
Network layers {encoder,decoder,T5}_layers.
Output: Predicted sequence $\tilde{x}$.

```
1  /* components                                                    */
2  MLPEnc = Sequential(encoder_layers)
3  MLPDec = Sequential(decoder_layers)
4  TEnc = TransformerEncoder(T5_layers)
5  TDec = TransformerDecoder(T5_layers)
6  p = ParameterList([Parameter(d) for _ in range(controls)])
7  /* forward pass                                                  */
8  feat_in = MLPEnc(x)
9  enc_inp = cat(*p, feat_in)                    ▷ concat control points
10 enc_out = TEnc(enc_inp)

11 c_points = enc_out[:controls]
12 latents = EVALUATE(c_points, seq_len)                          ▷ Eq. (1)

13 feat_out = TDec(latents)
14 x̃ = MLPDec(feat_out)
```

Algorithm 2: EVALUATE CubicBézier

Input: Four control points c_points and sequence length seq_len.
Output: Uniformly evaluated latent spline $\mathbf{s}$.

```
1 t = linspace(0, 1, seq_len)
2 s = ((1 − t)^3 · c_points[0]
3       + 3.0 · (1 − t)^2 · t · c_points[1]
4       + 3.0 · (1 − t) · t^2 · c_points[2]
5       + t^3 · c_points[3])
6 return s
```

Spline-based Transformers only require simple modifications to a traditional transformer model and easily fit into various existing setups. Alg. 1 presents a general implementation of Spline-based Transformers using a torch-like syntax. Different layers can be chosen for MLP encoder/decoder, *e.g.*, in the case of image data, 2D convolution layers can be used. Similarly, different transformer blocks can be stacked together to build our transformer encoder/decoder. Our control tokens $\mathbf{p}$ are learnable parameters (Line 6) and are initialized from a normal distribution. These learnable control tokens are concatenated to the se-

quence of tokens after the MLP encoder (Line 9) to result in the transformer encoder's input sequence *enc_inp*. The tokens corresponding to the control tokens are interpreted as latent control points at the output of the encoder, which are then evaluated as a spline (Line 11-12). In our experiments, we used cubic Bézier splines (with four control points), and Alg. 2 shows how we evaluate them to obtain a latent token sequence for the decoder. Finally, the interpolated latents are passed through the transformer decoder and the shared MLP decoder and ultimately mapped back into their original space. An L2 loss function between the input and output sequence is used to train our system end-to-end.

Table 1: Parameters and Hyperparameters.

Sec	Experiment	Param.	d	n	h	c	FFN	BS	lr	PS
4.1	Lissajous (3D)	0.20M	3	4	4	64	1	256	$1e^{-3}$	-
	Hypotrochoids (4D)	0.20M	4	4	4	64	1	256	$1e^{-3}$	-
	Bézier (2D)	0.20M	2	4	4	64	1	1024	$1e^{-3}$	-
	Bézier (64D)	0.43M	64	4	4	128	1	1024	$1e^{-3}$	-
4.2	CIFAR10	0.85M	$2^{5,6,7}$	4	8	128	1	512	$3e^{-4}$	4
	AFHQ	1.65M	$2^{5,6,7}$	4	8	128	1	512	$3e^{-4}$	32
	Face images	1.65M	$2^{5,6,7}$	4	8	128	1	512	$3e^{-4}$	32
4.3	Faces	12.3M	$2^{5,6,7,8}$	4	8	256	1	32	$5e^{-5}$	-
	Motions	0.63M	$2^{4,5,6}$	4	8	128	1	1024	$3e^{-4}$	-
4.4	Strands	0.21M	$2^{3,4,5}$	4	8	64	1	128	$1e^{-3}$	-

Tab. 1 summarizes the parameters and hyperparameters used to conduct the experiments presented in the paper. The latent dimension d, number of stacked transformer layers n, number of transformer heads per layer h, and the feature size of each layer c. Internally, the transformer layers consist of Feed-Forward Networks (FFN), two fully connected layers with non-linear activation GELU. FFNs can have an inner dimension that is 1-4x larger than their outer dimension. We keep the inner structure equal and set the factor to 1x. BS is the batch size, and lr is the learning rate. For the image results, we used a patch size PS. The number of parameters corresponds to the model with the largest latent dimension.

3 Additional Results

Additional results for reconstructing synthetic curves, facial images, and hair strands are shown in Figs. 3, 4, and 5, respectively.

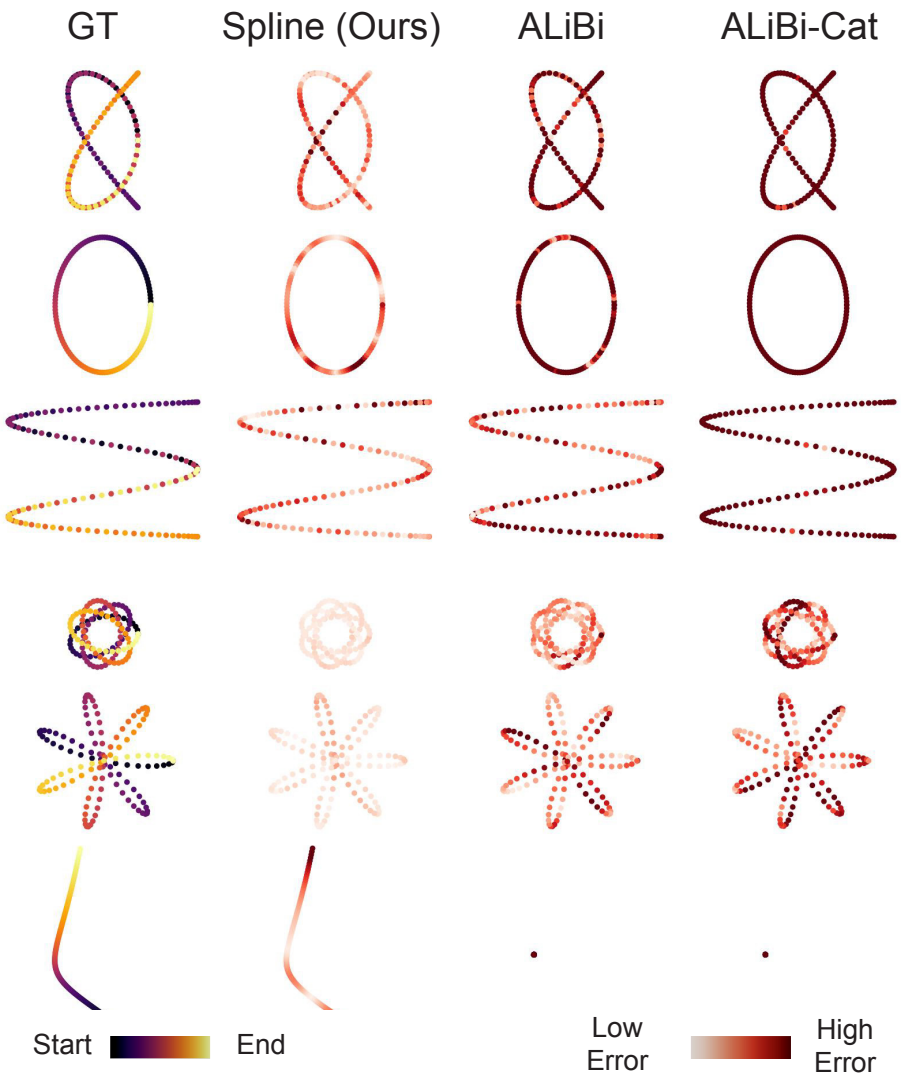

Fig. 3: Additional reconstruction results for 2D curves.

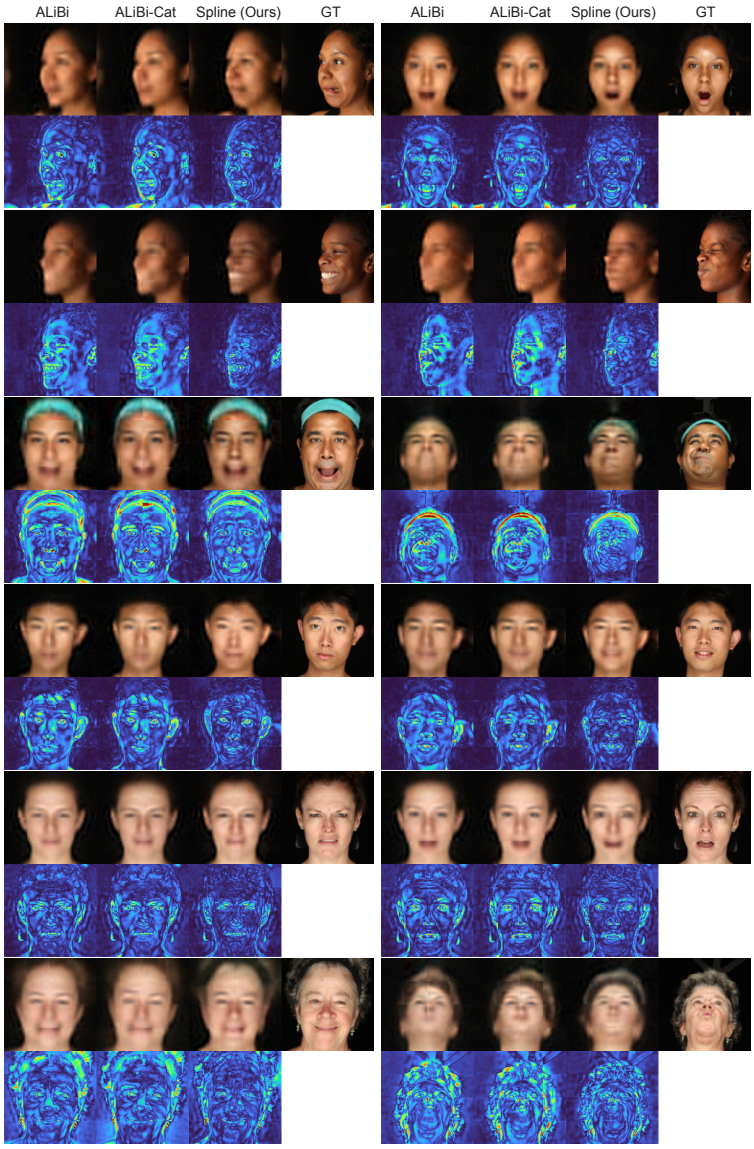

Fig. 4: Additional reconstruction results for test images.

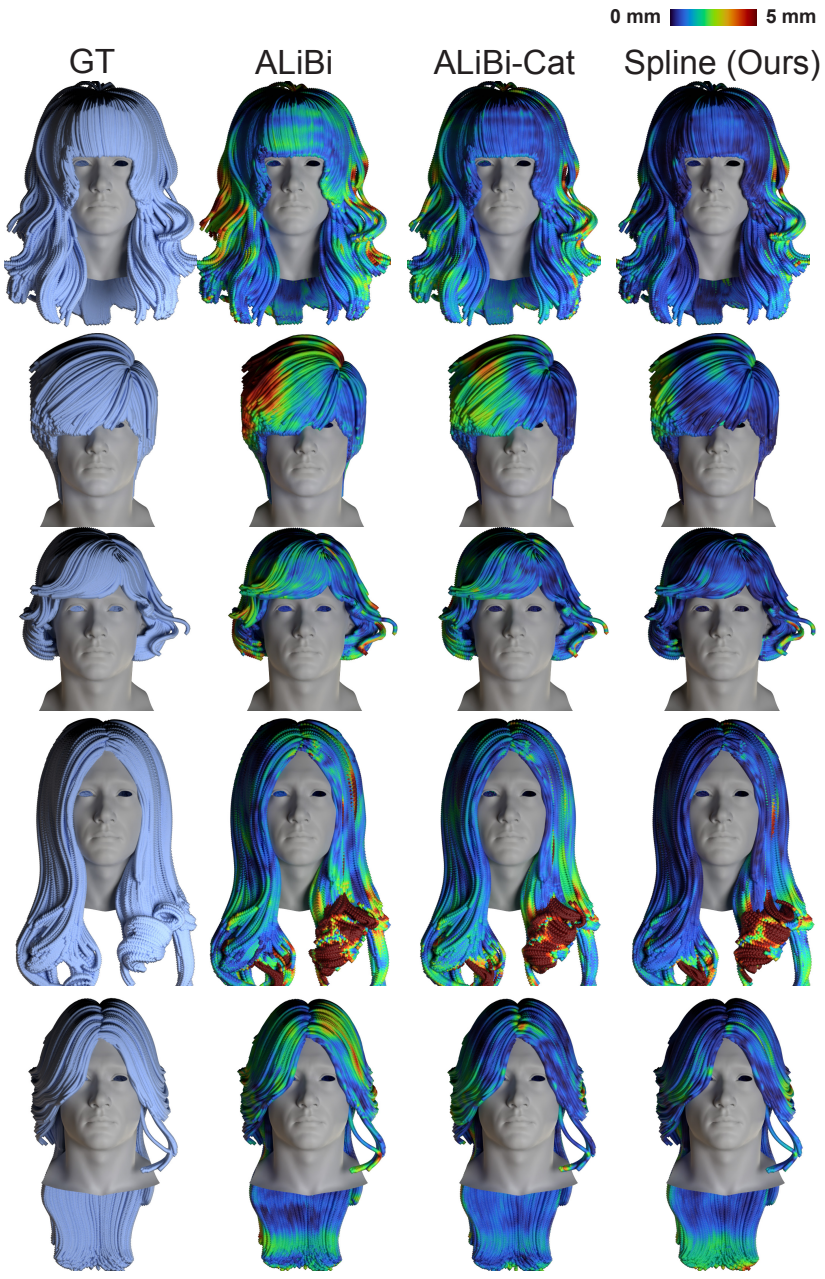

Fig. 5: Additional reconstruction results for test hairstyles.