CURVE2MELODY: SKETCH-GUIDED MELODY GENERATION

Cristina JORDA I CUSTAL (c.j.i.custal@imv.uio.no) [1],
Tejaswinee KELKAR (tejaswinee.kelkar@imv.uio.no) [1], and
Stefano FASCIANI (stefano.fasciani@imv.uio.no) [1]

[1] *Department of Musicology*, **University of Oslo**, Oslo, Norway

ABSTRACT

In this paper, we present Curve2Melody, a multimodal AI generative system that enables the creation of new melodies based on a melodic context and a hand-drawn sketch. Our work builds upon sketch-based melodic generative models, specifically the inpainting system "Draw and Listen", optimizing its architecture by utilizing a simplified input data structure and conditioning the model on a single context measure. We introduce a novel methodology for representing melodic contours using spline curves, encoding them through their derivatives to extract relevant features. The model employs a variational autoencoder with two encoders and three decoders, predicting pitch range, rhythm, and key from the context measure, while pitch is generated from the input sketch. To evaluate the performance, we design three experiments using the same pitch-, rhythm-, and context-matching metrics as in the baseline. Results show that our approach improves contour and rhythm matching while achieving tonal consistency, outperforming the baseline model on the same dataset using a smaller model with more than thirty times fewer trainable parameters. [1]

1. INTRODUCTION

Melodies are inherently complex, exhibiting intricate pitch movements, hierarchical structures, and stylistic variations, which complicates both analysis and generation tasks, especially when melodic generation allows for user engagement [1]. Cross-correspondences of melodic shape and melodic detail are studied in several works and sketch-based music-making interfaces [2, 3]. Melodic contour has been widely employed as a key descriptor in music-related research for these tasks. By presenting the pattern of ups and downs of the melody's intervals over time, melodic contour captures the overall tendency and directional movement of a raw pitch sequence [4, p. 24].

Cognition of melodic contour is acquired early in human life, with infant-directed speech sharing melodic exaggerations across cultures. Psychological models of melodic expectation such as the IR model [5] are shown to be successful in melodic generation, but statistical approaches to melodic generation are also popular. Applications of melodic contour in analytical and generative systems range from melodic identification and similarity matching to generation. Contour can be considered a curve as it is a univariate function that represents pitch as a time-dependent variable. Recent works have taken a multimodal perspective, introducing novel interactive music generation systems that produce melodies resembling user-drawn curves [2, 3, 6–8]. Such data are particularly well-suited as inputs to generative artificial intelligence models, as they embody an initial intuitive idea that can be refined while preserving the overall contour.

This paper builds upon the music inpainting task based on user-drawn sketches [3]. One goal of sketch-based generation systems is to make the model accessible for inexperienced musicians, predicting the melody from the curve without a melodic context [2, 8], while several music inpainting works have focused on filling in a missing melodic excerpt in a given melody [3, 9, 10]. Curve2Melody approaches this framework from a different point of view, relying on just one melodic measure as past context for the generation. A user provides a sketch that functions as a melodic contour to guide the generative process. A new measure is produced following the tendency of the sketch, using the key, pitch range, and rhythmic patterns from the past measure. The generated measure then becomes the contextual input for subsequent generation. Given the recursive nature of this system, the user is free to create a complete musical piece with an arbitrary number of measures.

We accomplish this by optimizing the state-of-the-art model "Draw and Listen" [3] to our task, significantly reducing the dimensionality of the input and simplifying its architecture to obtain a model with approximately thirty times fewer trainable parameters, while still achieving good performance. Additionally, we introduce a novel spline-based approach for representing sketch-based melodic contours. Features derived from the melodic contour can be encoded geometrically, for example using cosine components as in [2]. Our approach based on first-order derivatives of the spline approximates the drawn curve to the derivative of the closest cubic spline, which carries only the slope information of the original curve. This method has been used for melody representation, ensuring its efficiency in simplifying the melodic contour while maintaining its overall tendency [11, 12]. The model

[1] https://github.com/CrisJorda/melodic-contour-generation-based-on-sketches

Copyright: © 2026. This is an open-access article distributed under the terms of the Creative Commons Attribution 3.0 Unported License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

architecture is adjusted to reflect the modifications introduced in the inputs. In contrast to the four original inputs described in [3], our model uses only a single input, which is lower in dimensionality. Consequently, we propose a model with a significantly lower number of trainable parameters, approximately *thirty times* fewer than in the state of the art. We design three experiments to evaluate the model under the same evaluation criteria as the baseline, obtaining improved performance.

Many applications can benefit from our system in creative musical contexts. It can be of special interest to professional composers as it accelerates the development of stylistically consistent musical material. It is also a great tool for amateur composers with less experience: given a melodic start, the model can recursively develop the full compositions guided only by a few sketches. Finally, our system helps to intuitively understand the influence of the contour in music and how it could potentially be used to design engaging interactive interfaces.

2. RELATED WORK

The task of generating a melody based on sketches is not a novel compositional idea, with John Cage experimenting with line-based scores, Daphne Oram's Oramics machine, and later computational algorithms [13]. Generating melodies conditioned on a previous musical context has been addressed in prior works [9, 14, 15]. Many of these studies frame the problem as music inpainting, where the goal is to generate a missing melodic segment between two existing musical passages. Such systems are typically implemented using a variational autoencoder (VAE) [16]. In this framework, the surrounding melodic context is encoded into a latent space, from which the missing segment is predicted and decoded back to symbolic music.

Contour-based multimodal music generation has also been explored in multiple works, combining symbolic musical inputs with user-drawn curves [6, 7]. A particularly relevant contribution is "Draw and Listen" [3], a music inpainting system that enables users to fill a missing musical measure by sketching control curves. In addition to the past and future context measures, the model accepts up to four different types of user inputs, primarily a pitch curve providing the tendency for the pitch contour. The user may draw one or more optional rhythm curves, indicating the zones in the pitch curve where the rhythm density should increase. Two sliders allow the user to adjust global pitch and rhythm offsets additionally. The system's architecture is also based on a VAE comprising several encoders and decoders, each producing a corresponding latent representation. Three encoders are employed: one for the pitch curve, one for the rhythm curves, and one for both the surrounding context measures. The pitch decoder maps the pitch latent vector into pitch tokens. The rhythm decoder receives as input the concatenation of the rhythm latent vector, the rhythm slider, and the context latent vector, and generates rhythm tokens. Pitch and rhythm tokens are then concatenated with the context latent vector and passed to a final decoder, which outputs the target melodic measure. A fourth decoder provides harmonic learning within the context latent space.

"MIDI-Draw" [2] tackles sketch-based melodic contour generation through a geometrical representation. Following this approach, the drawn curves are encoded as cosine components, providing a compact parametric description of the contour. Unlike "Draw and Listen" [3], this system requires only a pitch curve as input and generates the corresponding melody with fixed rhythmic structure. The underlying architecture is a conditional VAE, which predicts the melody conditioned on the sketch and the cosine components.

"Drawlody" [8] is an interactive music creation system designed to enhance usability and interpretability, allowing users without prior musical knowledge to generate melodies. The model uses only an image of the pitch curve as input, not requiring a predefined length, and generates corresponding rhythmic patterns based on structures learned during training. It utilizes a model with a VAE layout that incorporates Transformer components [17], allowing the model to better capture long-range dependencies in the generated music.

The geometric representation of curves using splines has been applied in audio and melodic analysis [18]. In particular, [11] interprets melodies as spline curves and computes melodic similarity by comparing their shapes, using derivatives of the fitted splines as descriptive features. More recently, [12] models cycles in audio signals directly as spline curves, generating new melodies that mirror the contours of selected audio waveforms segments.

3. METHODOLOGY

3.1 Task Design

The objective of Curve2Melody is to generate a one-measure melody following a hand-drawn curve provided by the user. At the same time, this new melody is conditioned on a preceding measure, which provides contextual information such as key, pitch range, and rhythmic patterns. In other words, this work introduces a multimodal music creation system for monophonic melody generation given a hand-drawn pitch curve and a single past context measure.

This formulation is similar to the "Draw and Listen" system [3], but with some key differences. While both approaches address controlled melody generation, [3] focuses on music inpainting and therefore requires both past and future context measures as inputs. In contrast, Curve2Melody is causal because the generation is conditioned only on the past context measure. Whereas [3] allows up to four user inputs, the proposed model relies exclusively on the pitch curve. As a consequence, it generates the rhythm from the contextual information rather than being explicitly specified by the user.

Furthermore, both systems share several constraints. All melodies, whether context or generated, are monophonic, in 4/4 time signature, and limited to a pitch range of approximately two octaves. These design choices are reflected in the dataset, which contains only samples satisfying these characteristics. Additionally, both context and

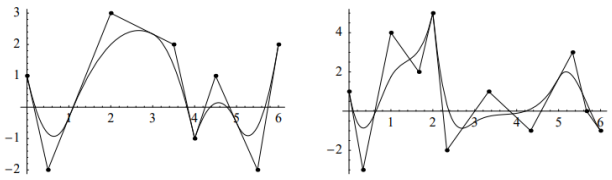


Figure 1. A quadratic spline (left) and a cubic spline (right), with their corresponding control polygon and control points. Adapted from [19, p. 99].

generated outputs are restricted to a single-measure resolution.

3.2 Contour Representation

Another key difference from [3] lies in the representation of the input pitch curve. In that previous work, the curve is encoded as a relative pitch function: a 512-position normalized vector that explicitly captures the trajectory of the melodic contour. In contrast, we employ spline curves to represent the contour, adopting the geometric methodology proposed by [11] for melodic analysis.

A spline is a piecewise polynomial function defined over a partition of an interval $[a, b]$ determined by a knot vector, which is a non-decreasing sequence of parameter values $t_0 \leq t_1 \leq \cdots \leq t_m$ [12, 19].

A B-spline curve of degree k is expressed as a linear combination of basis functions, as shown in Equation 1:

$$f(t) = \sum_{i=0}^{n-1} c_i B_i^{(k)}(t) \tag{1}$$

where c_i are control points, n is the number of control points, and $B_i^{(k)}(t)$ are B-spline basis functions of degree k defined with respect to the knot vector.

Therefore, the spline is determined by three elements: the degree k, the control points $\{c_i\}_{i=0}^{n-1}$, and the knot vector $\{t_j\}_{j=0}^{n+k}$, whose length satisfies $\text{len}(t) = c + k + 1$. The control polygon is obtained by connecting the control points linearly.

To construct a clamped B-spline, the knot vector is defined such that the first $k + 1$ knots coincide with the left boundary and the last $k + 1$ knots coincide with the right boundary, leaving the internal knots to be strictly increasing, as described in Equation 2:

$$\begin{aligned} t_0 = t_1 = \cdots = t_k = a, \\ t_{k+1} < t_{k+2} < \cdots < t_{n-1}, \\ t_n = t_{n+1} = \cdots = t_{n+k} = b. \end{aligned} \tag{2}$$

Figure 1 illustrates two spline curves, one quadratic ($k = 2$) and one cubic ($k = 3$). These curves can be interpreted as approximations of the melodic contour, corresponding to the curve drawn by the user. Furthermore, the control polygon shown in the figure serves as an intermediate geometric structure, with its vertices or control points acting as parameters that are indirectly used as inputs to the model, as summarized in Table 1.

Graphical element	Spline concept	Role in the model
Curve	Parametric spline	User-drawn contour
Line segments	Control polygon (implicit)	Intermediate geometric structure
Dots	Control points	Parameters used to derive the model input

Table 1. Relationship between graphical elements in Figure 1, spline concepts, and their role in the model.

The use of splines to represent and encode melodic contours has been explored in previous works. For instance, [11] demonstrates how spline representations can simplify melodic contours while preserving their overall structural characteristics. Similarly, [2] shows that geometrical approaches, such as cubic splines or cosine components, provide effective alternatives for computing symbolic melody indexing and similarity.

In contrast to [11], in Cruve2Melody a spline is fitted to an existing melodic contour represented as a continuous curve, the relative pitch function, rather than being constructed directly from a discrete sequence of notes. The resulting control points are not used directly as model inputs, but instead they serve as geometric parameters from which shape-based features are computed to represent the contour. In [11], splines are characterized through their derivatives, which capture essential geometric properties of the curve, particularly slopes and changes in orientation. Adopting the same principle, the fitted spline in this work is represented through its derivative, producing a stable and compact representation that is robust to sampling density and local variations while preserving the overall shape of the input contour.

Figure 2 displays a melody with its relative pitch function and the corresponding fitted cubic spline. Note that the spline does not perfectly match the function but reflects the overall tendency. The first derivative of the spline is also shown, sampled at 256 equally spaced points, capturing the changes in the normalized pitch of the spline.

4. EXPERIMENTS

4.1 Model Architecture

The architecture of Curve2Melody is an optimization of the model presented in [3], modified to accommodate the proposed task and reduced input structure. As illustrated in Figure 3, the model has an overall VAE structure, combining two encoders and three decoders.

The model requires two inputs: a hand-drawn curve and a context measure. The curve is first processed to extract meaningful contour features. Specifically, a B-spline is fitted to the curve by minimizing the least squares (LSQ) criterion [20]. This produces a spline whose overall shape closely approximates the user-drawn curve, minimizing the global error. For this fitting, 12 control points are used, with cubic B-splines and 8 internal knots, resulting in a

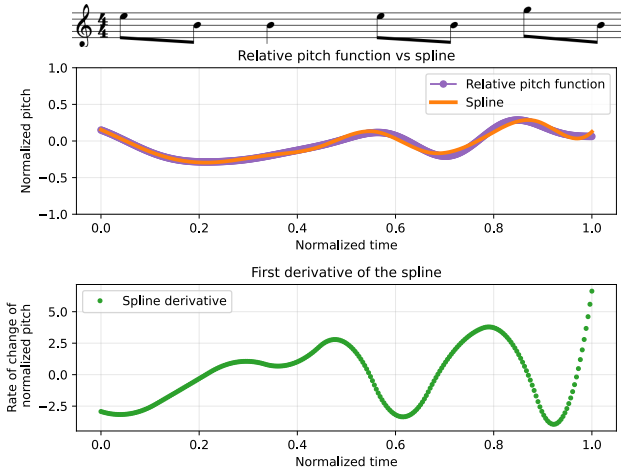


Figure 2. Three contour representations of a melody. The first plot shows the relative pitch function of the melody in purple, and the spline fitted to this curve in yellow. The second plot displays the first derivative of the spline in green.

knot vector of length 16. Once the spline is obtained, the model input vector `Xcurve` is defined as the first-order derivative of the fitted spline with length according to the parameter `spline_derivative_len`.

The encoders transform the inputs, `Xcurve` and `Xcontext`, into their corresponding latent vectors `Zp` and `Zc`, each of dimension 85. The curve encoder is implemented as a 1D convolutional neural network (CNN) [21] composed of residual blocks [22], with the number of blocks determined by the parameter `n_layers_encoder`. Each residual block contains three 1D convolutional layers with kernel sizes 1, 3, and 1, each followed by a LeakyReLU activation layer [21]. A residual connection sums the input of the block with its output, which then passes through another LeakyReLU activation layer.

The context encoder consists of an embedding layer of size 128, followed by a two-layer bidirectional gated recurrent unit (GRU) [23] with hidden dimension specified by `hidden_dim_gru_encoder`. At the end of both encoders, the mean and variance are reparametrized to obtain the latent vectors.

The pitch decoder takes the pitch latent vector `Zp` as input, passing it through a two-layer unidirectional GRU with hidden size `hidden_dim_gru_decoder`, followed by a linear layer that outputs the pitch token sequence `Tpitch` over 24 even positions. Each position encodes the pitch interval relative to the previous position, approximating a linear tendency along the curve. These intervals are quantized into 7 levels to keep this representation only as a guide for the generation process.

The MIDI decoder has the same architecture as the pitch decoder but receives the concatenation of `Tpitch` and `Zc` as input. It produces the final MIDI token sequence `Tmidi`, consisting of 130 possible tokens: 0-127 for MIDI pitches, 128 for sustained notes, and 129 for rests. These tokens are arranged on a grid of 24 positions, with each beat subdivided into six uneven ticks [14].

Finally, the chroma decoder generates a 12-dimensional vector `Vchroma` from the context latent vector `Zc`, indicating the set of pitch classes likely to appear in `Tmidi` given the harmonic context. This multitask harmonic learning approach for the context latent space is achieved with a linear layer of 12 dimensions.

4.2 Dataset

The dataset used for training, validation, and testing the model is mainly composed of samples from the Irish Folk Music dataset [24], which contains music transcriptions of traditional Irish melodies. Although it includes a variety of tunes, it was filtered to retain only those satisfying the same criteria as in [3], following the constraints described in Section 3.1. From the filtered tunes, non-overlapping two-measure segments were extracted, resulting in 455,766 samples. Each segment is represented with a resolution of 128 possible pitch values and a minimum note duration corresponding to a sixteenth note.

Following the methodology of [3], a synthetic dataset was created to address the scarcity of segments spanning a full two-octave range and to introduce more drastic pitch fluctuations. In this process, pitch values were assigned randomly within the two-octave MIDI notes from C4 to B5 to all sixteenth notes in the two-measure segments. By combining this synthetic dataset of 1,000 samples with the filtered Irish Folk Music dataset, the model is better prepared to handle a wide variety of user-drawn pitch curves.

The samples from the combined dataset are in MIDI format and require preprocessing to serve as inputs for the model. First, the note onsets of both measures are quantized onto a regular time grid of 48 positions per measure. The first measure is retained as the context melody, while the second measure is used to extract a relative pitch function to be the user's input curve. This extracted curve corresponds to the melody of the second measure, and it is used as the ground truth for the generated melody, allowing the model to be trained as supervised.

The relative pitch curve is obtained following the same approach as in [3]. A piece-wise linear interpolation is applied to obtain a 512-point sequence, with the Y-axis representing MIDI pitch values. The resulting sequence is then filtered using a Butterworth low-pass filter [25] with a normalized cut-off frequency of 1/50 and order 5. Finally, the curve is normalized by subtracting the midpoint of its range and dividing by 12.

Therefore, the inputs to the model consist of the 512-point relative pitch function and the context measure represented as a 48-position vector of MIDI values.

4.3 Experimental design

The model proposed was implemented in Python, using the PyTorch library [26] for training and validation. For data processing, the music21 package [27] was employed to handle symbolic musical data, while scipy [28] was used to compute the spline interpolation for the contour representation. The dtaidistance library [29] was used for evaluation (see Section 4.5).

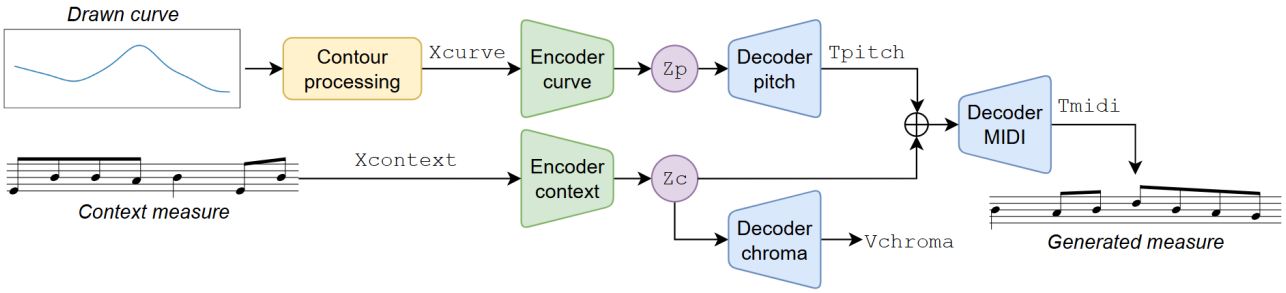


Figure 3. Diagram of the model architecture, showing two encoders, one for the processed input curve and the other for the context measure, and three decoders for the pitch tokens, the MIDI tokens, and the chroma vector, respectively.

The processed dataset was split into 70% for training, 20% for validation, and the remaining 10% for testing. Each experiment was trained for up to 100 epochs with a batch size of 128, using the Adam optimizer [30] with a fixed learning rate of 0.00003. Early stopping with a patience of 10 epochs was applied, interrupting training if the validation loss did not improve for 10 consecutive epochs. The experiments were conducted on a workstation equipped with an Intel Core i9-14900KF (24 cores) CPU, 192GB of DDR5 5200MHz RAM, and an NVIDIA GeForce RTX 4090 (24GB) GPU, all operating on Ubuntu 24.04 LTS.

Three experiments were designed to train and evaluate the model. Table 2 summarizes their configurations.

Experiment 1 preserves the configuration of [3], using the relative pitch function as the contour representation and the adapted architecture with the original parameter values. This results in a model with a significant number of trainable parameters and an approximated total training time of one week.

Experiment 2 uses the same model parameters as Experiment 1 but replaces the contour representation with the spline derivative. This introduces a new architectural parameter, `spline_derivative_len`, which is the length of the derivative vector. In this experiment, it is set to 512 uniformly spaced positions, so the only difference from Experiment 1 is the contour representation, which introduces a geometric regularization through the spline fitting process. It was trained for approximately three days before the early stopping mechanism interrupted the training at 60 epochs.

Experiment 3 also utilizes the spline derivative but incorporates additional architectural modifications. The derivative vector length is halved to 256 positions, and the number of residual blocks in the curve encoder is decreased accordingly to avoid overfitting. The hidden dimension of the context encoder's GRU, which in [3] was designed to encode two measures, is also reduced to match the single-measure input. Having both inputs reduced, the GRU hidden dimensions in the decoders are also halved. These adjustments result in a more efficient model with significantly fewer parameters, reducing the total number of trainable parameters by more than thirty times while preserving functionality and training in approximately three days.

4.4 Loss Function

The training of the supervised model is guided by a loss function defined as the sum of four individual losses [3], as shown in Equation 3.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Tmidi}} + \mathcal{L}_{\text{Tpitch}} + \mathcal{L}_{\text{Vchroma}} + \mathcal{L}_{\text{VAE}} \quad (3)$$

In this equation, $\mathcal{L}_{\text{Tmidi}}$ denotes the loss of the final token sequence, which is computed as the cross-entropy loss between the predicted MIDI token sequence `Tmidi` and its ground truth. Similarly, $\mathcal{L}_{\text{Tpitch}}$ corresponds to the cross-entropy loss of the intermediate pitch token sequence `Tpitch` and its ground-truth sequence. $\mathcal{L}_{\text{Vchroma}}$ is the binary cross-entropy between the predicted chroma vector `Vchroma` and its ground truth. Finally, $\mathcal{L}_{\text{VAE}}$ is the Kullback-Leibler (KL) divergence loss for the VAE latent vectors, defined in Equation 4 [31].

$$\mathcal{L}_{\text{VAE}} = \beta_{\text{var}} \left(\mathcal{L}_{\text{KL}}^{(z_{\text{p}})} + \mathcal{L}_{\text{KL}}^{(z_{\text{c}})} \right) \quad (4)$$

The total VAE loss is computed as the sum of the KL divergence loss of pitch latent vector `Zp` and the context latent vector `Zc`. Each KL divergence loss is calculated from the mean and variance of its respective latent vector. The combined VAE loss is weighted by the variational beta parameter β_{var}, which is set to 0.1.

4.5 Evaluation metrics

The model is evaluated to assess pitch and rhythm contour similarity, as well as the contextual fitness [3].

For evaluation, two random samples of two measures each are drawn from the test dataset, resulting in four measures: `M_11` and `M_12` for the first sample and `M_21` and `M_22` for the second. `M_11` serves as the context for the generation, while `M_22` is used as the user input. M_22 is preprocessed to obtain the relative pitch function, which serves both as input to the model and as the ground truth for evaluation. The objective is to measure the similarity between the generated melody `M_3` and the `M_22` pitch and rhythm contours while ensuring alignment with the `M_11` context. Following the methodology proposed by Benetatos et al. [3], we repeat this process 1,000 times and report the averaged results.

Dynamic time warping (DTW) is used to evaluate the pitch and rhythm contour matching. DTW is a robust algo-

Trainable Parameter	Experiment 1	Experiment 2	Experiment 3
Contour representation	Relative pitch function	Spline derivative	Spline derivative
`spline_derivative_len`	-	512	256
`n_layers_encoder`	7	7	2
`hidden_dim_gru_encoder`	2048	2048	512
`hidden_dim_gru_decoder`	2048	2048	512
Total trainable parameters	581,220,741	581,220,741	17,571,973

Table 2. Experiment configurations according to the contour representation used and the values of its architectural parameters. See Section 4.1 for an explanation of the parameters.

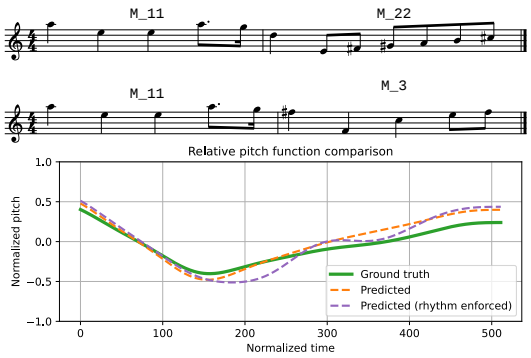


Figure 4. Example of the evaluation process. The samples used are `M_11` as context measure and `M_22` as ground-truth measure. The next score shows the same context measure but with the generated measure `M_3`. On the bottom, the extracted relative pitch function is displayed from the ground-truth measure (green), the predicted measure (orange), and the predicted measure with the rhythm enforced (purple).

rithm for comparing time series with potential shifts or distortions in time, capturing similarity in overall shape [32].

In our work, rhythm is fully generated from the context, and pitch alignment costs are calculated both with and without rhythm enforcement to allow direct comparison with prior results. The ground-truth rhythm is represented as a dense vector of 16 sixteenth-note positions, each one labeled as '0' for rhythm onsets, '1' for holds, and '2' for rests. Unlike [3], the current model does not include a rhythm decoder, so rhythm is enforced during evaluation by masking the generated pitch token sequence according to the rhythm vector. Therefore, DTW aligns the relative pitch function of the generated melody with that of the input curve, computing the alignment cost as a reference for the matching of the contours. It is important to note that this metric evaluates the melodic shape, not exact note values. For the rhythm contour matching, a relative rhythm function is extracted from both the predicted and ground-truth measures, which encodes the note density instead of the exact note durations. The DTW algorithm is then applied to these rhythm functions to compute the rhythm alignment cost.

Context matching is measured as the percentage of notes in the generated melody whose pitch classes belong to the key estimated from the context melody, determined using the Krumhansl algorithm [33]. As noted in [3], this metric alone is not sufficient to fully assess the context fitness, but it provides a basic assessment of tonal consistency.

In addition, pitch and rhythm contour metrics are computed between `M_12` and the generated melody `M_3` to provide a reference for the alignment cost between unrelated melodies. Context matching is also evaluated between `M_11` and `M_12` to have the reference of the metric in two measures belonging to the same sample.

4.6 Results

The results of the three experiments described in Section 4.3, evaluated with the metrics explained in Section 4.5, are summarized in the following tables. Table 3 reports the evaluation costs for related measures, while Table 4 presents the costs for reference measures. As evident in the tables, optimizing "Draw and Listen" model for the different task had a positive impact on the metrics in Experiment 1, which serves as the baseline for comparison with Experiments 2 and 3. Replacing the relative pitch function with the spline derivative in Experiment 2 improved performance across nearly all metrics, except context matching for the related measures and rhythm matching for the reference measures. Further reducing the architecture in Experiment 3 led to additional improvements for pitch matching with rhythm enforcement and rhythm matching for the related measures. For the reference measures, pitch-matching performance improved, while rhythm-matching costs increased, and context-matching percentage decreased slightly.

Figure 4 displays an example of the evaluation process, showing how the predicted measure `M_3` follows the same contour as the ground-truth measure `M_22` but adapts the rhythm and the pitch values to the context measure `M_11`. The same figure also shows a plot where the relative pitch function curves of the ground truth melody and the predicted melody, both with and without rhythm enforcement, can be compared.

5. CONCLUSIONS

Curve2Melody is a multimodal sketch-based system for symbolic melody generation, conditioned on a drawn curve and a previous measure. Building on the "Draw and Listen" music inpainting framework, we redesigned the input structure and representation, and consequently optimized the architecture to obtain a substantially lighter

Metric	Draw and Listen	Experiment 1	Experiment 2	Experiment 3
Pitch matching without rhythm enforcing	–	1.6 ± 1.5	$\mathbf{1.3 \pm 1.3}$	1.3 ± 1.4
Pitch matching with rhythm enforcing	6.1 ± 5.2	1.8 ± 1.6	1.4 ± 1.4	$\mathbf{1.3 \pm 1.4}$
Rhythm matching	8.4 ± 13.1	0.9 ± 1.4	0.4 ± 1.0	$\mathbf{0.2 \pm 0.7}$
Context matching	86%	**95%**	94%	93%

Table 3. Evaluation results of the related measures.

Metric	Draw and Listen	Experiment 1	Experiment 2	Experiment 3
Pitch matching without rhythm enforcing	–	4.8 ± 3.0	4.6 ± 2.9	$\mathbf{4.5 \pm 2.8}$
Pitch matching with rhythm enforcing	54.6 ± 38.3	4.7 ± 3.2	4.5 ± 3.1	$\mathbf{4.5 \pm 3.0}$
Rhythm matching	53.8 ± 61.3	$\mathbf{1.2 \pm 1.5}$	1.6 ± 1.6	1.7 ± 1.6
Context matching	87%	**96%**	**96%**	95%

Table 4. Evaluation results of the reference measures.

model while maintaining the ability to recursively generate stylistically coherent melodies. We introduced a novel methodology for representing melodic contour curves by approximating the sketch with a cubic spline and using its first-order derivative directly as model input. Our results show that this geometrical approach improves performance and offers an efficient alternative for modeling melodic contours, with lower computational complexity than standard polynomial function representations.

We evaluate our model with three experiments which run on simulated curve inputs. Modifying the contour representation and reducing the architecture led to consistent improvements in evaluation metrics, particularly for pitch and rhythm matching on both related and reference measures. Context-matching performance decreased slightly across experiments, although it remains significantly higher than the baseline.

Although Curve2Melody achieves good performance, there is still room for further development. A natural next step is to train the system to produce melodies with a greater real-world relevance by incorporating rests into the training data. Moreover, it would be valuable to expand the temporal scope of both the context and the generated melodies to include musical sequences of more than one measure. In addition, the model should be evaluated using real hand-drawn sketches collected in a user study, allowing the use of subjective evaluation metrics. Another limitation is that most samples in the current dataset come from Irish folk music. Expanding the dataset to include other musical styles could increase the diversity of the model's outputs. Allowing the user to input more and different types of parameters to the system can also be beneficial, as it provides more control over the generated output. Finally, we could improve the architecture by leveraging transformer components to better capture long-range dependencies and achieve better results on the matching metrics.

6. REFERENCES

[1] D. Herremans, C. Chuan, and E. Chew, "A functional taxonomy of music generation systems," in *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, 2017, pp. 1–30.

[2] T. Namgyal, P. Flach, and R. Santos-Rodriguez, "Midi-draw: Sketching to control melody generation," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2023.

[3] C. Benetatos and Z. Duan, "Draw and listen! a sketch-based system for music inpainting," in *Transactions of the International Society for Music Information Retrieval*, 2022, pp. 141–155.

[4] H. W. B and E. Selfridge-Field, *Melodic Similarity*. MIT Press, 1998.

[5] E. Narmour, *The analysis and cognition of melodic complexity: the implication-realization model*. Chicago: University of Chicago Press, 1992.

[6] T. Kitahara, S. Giraldo, and R. Ramírez, "Jamsketch: Improvisation support system with ga-based melody creation from user's drawing," in *International Symposium on Computer Music Multidisciplinary Research*, 2017, pp. 509–521.

[7] S. Löbbers, L. Thorpe, and G. Fazekas, "Sketchsynth: cross-modal control of sound synthesis," in *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, 2023, pp. 164–179.

[8] Q. Liang and Y. Wang, "Drawlody: Sketch-based melody creation with enhanced usability and interpretability," in *IEEE Transactions on Multimedia*, vol. 26, 2024, pp. 7074–7088.

[9] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 343–351.

[10] A. Marafioti, P. Majdak, N. Holighaus, and N. Perraudin, "Gacela – a generative adversarial context encoder for long audio inpainting," in *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 1, 2020, pp. 120–131.

[11] J. Urbano, J. Lloréns, J. Morato, and S. Sánchez-Cuadrado, "Melodic similarity through shape similarity," in *International Symposium on Computer Music Modeling and Retrieval*, 2010, pp. 338–355.

[12] M. Klassen and P. Lanthier, "Melodic contour generation with spline models of cycles," in *International Conference on Mathematics and Computation in Music*, 2024, pp. 210–223.

[13] M. Farbood, H. Kaufman, and K. Jennings, "Composing with hyperscore: An intuitive interface for visualizing musical structure," in *ICMC*, 2007.

[14] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019, pp. 596–603.

[15] K. Chen, C. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music sketchnet: Controllable music generation via factorized representations of pitch and rhythm," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020, pp. 77–84.

[16] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," in *Foundations and Trends in Machine Learning*, vol. 12, no. 4, 2019, pp. 307–392.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[18] D. Lolive, N. Barbot, and O. Boeffard, "Melodic contour estimation with b-spline models using a mdl criterion," in *Proceedings of the 11th International Conference on Speech and Computer*, 2006, pp. 333–338.

[19] T. Lyche and K. Morken, *Spline methods draft*. Department of Informatics, Center of Mathematics for Applications, University of Oslo, Oslo, 2008.

[20] D. D. Morrison, "Optimization by least squares," in *SIAM Journal on Numerical Analysis*, vol. 5, no. 1, 1968, pp. 83–88.

[21] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, 2021, pp. 6999–7019.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[23] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *IEEE 60th International Midwest Symposium on Circuits and Systems*, 2017, pp. 1597–1600.

[24] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Conference on Computer Simulation of Musical Creativity*, 2016.

[25] M. Shouran and E. Elgamli, "Design and implementation of butterworth filter," in *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 9, no. 9, 2020, pp. 7975–7983.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[27] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2010, pp. 637–642.

[28] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright *et al.*, "Scipy 1.0: Fundamental algorithms for scientific computing in python," in *Nature Methods*, vol. 17, no. 3, 2020, pp. 261–272.

[29] W. M. Meert, "dtaidistance: Distance measures for time series data," https://github.com/wannesm/dtaidistance, accessed: 2026-03-06.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015.

[31] J. Terven, D. M. Cordova-Esparza, J. A. Romero-González, A. Ramírez-Pedraza, and E. A. Chavez-Urbiola, "A comprehensive survey of loss functions and metrics in deep learning," in *Artificial Intelligence Review*, vol. 58, no. 7, 2025, p. 195.

[32] P. Senin, "Dynamic time warping algorithm review," in *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, vol. 855, no. 1-23, 2008, p. 40.

[33] C. L. Krumhansl and E. J. Kessler, "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys," in *Psychological Review*, vol. 89, no. 4, 1982, pp. 334–368.