CurveCodec 2: Skeleton-Agnostic Animation Compression with a Learned Entropy Model

MINGYI SHI, The University of Hong Kong, Hong Kong
HUANCHENG LIN, The University of Hong Kong, Hong Kong
XUELIN CHEN*, Adobe Research, United Kingdom
TAKU KOMURA*, The University of Hong Kong, Hong Kong

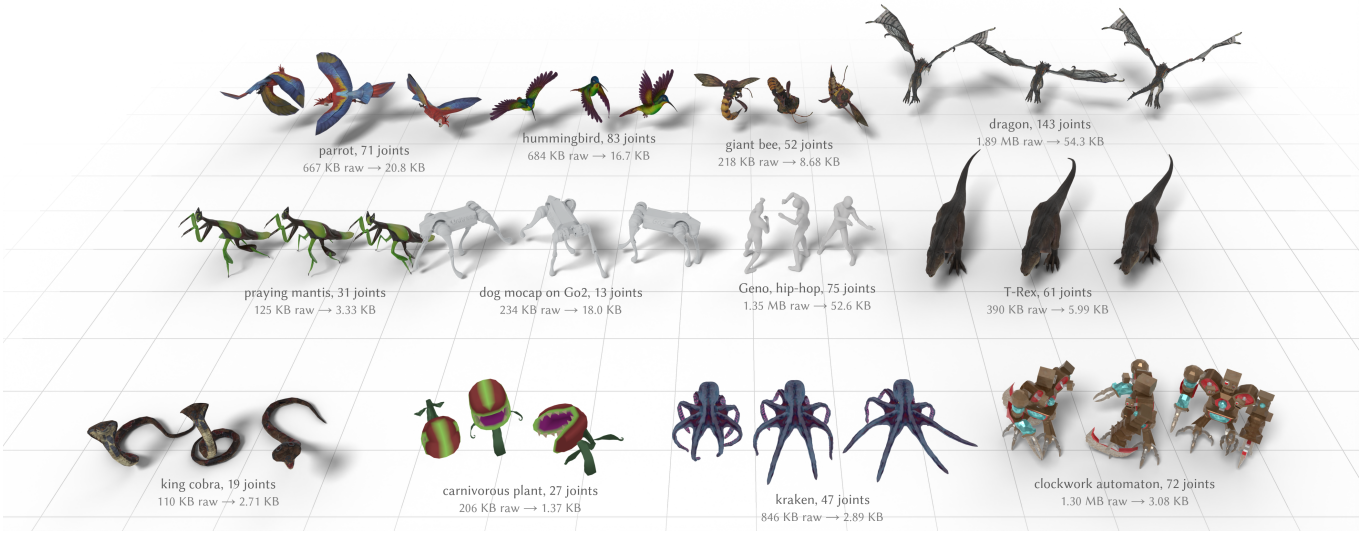


Fig. 1. CurveCodec 2 is a skeleton-agnostic codec for skeletal animation. Twelve rigs from the example set of our online demo, each at three frames of its clip as decoded by CurveCodec 2 at ACL's default precision of 0.01 cm, with the clip's raw size and the size of our stream under each. One model, trained once, codes all of them without retraining. On the held-out test side CurveCodec 2 needs 0.37× ACL's bytes at this precision, from a bit-exact bitstream that decodes on one CPU core. Project page, online demo, and supplementary material: https://rubbly.cn/publications/curvecodec/

Skeletal motion is stored as every joint's transform at every frame, yet most of it is implied by the body rather than by what the motion is about. Compression is one way to ask what a motion must still say once the body is known, and a production codec is where that question has to be answered for any skeleton, with a stated bound on the error. Our earlier codec, CurveCodec, reconstructed each joint curve from sparse anchors with a learned prior. It matched the mean error of ACL, the production library of modern game engines, but not its worst case, and it counted its payload as floating-point values rather than as bits. Here we ask where the redundancy of skeletal motion lies, and which part of a codec a learned model should take over. A measured anatomy of ACL and a series of controlled experiments give three answers. At production precision the largest saving comes from predicting each quantized curve from its own past, and the second from choosing per joint, in closed loop through the hierarchy, which samples *not* to code. On the gaps such an encoder leaves, interpolation has little left to find: a nearest-neighbour oracle with millions of training samples as memory is no better than linear interpolation there, and none of the learned in-betweeners we tried paid for itself. What a network does learn is the distribution of the residuals the codec must send. CurveCodec 2 puts these findings into one codec. Every sub-track becomes a curve, rotations in the log map and translations in centimetres, quantized in closed loop and thinned to rate–distortion-selected keys, and its prediction residuals are entropy-coded under a small learned model whose integer inference is bit-exact across platforms. Two contracts are verified on every decoded clip: ACL's own worst case per joint, within a stated tolerance, or ACL's mean error per clip. On a held-out test side of 4,472 clips from 33 datasets, CurveCodec 2 needs 0.37× ACL's bytes at ACL's default precision of 0.01 cm under the worst-case contract and 0.22× at 0.1 cm under the mean contract. It decodes whole clips on one CPU core, typically in a fraction of a second, and transfers without retraining to a species absent from training.

CCS Concepts: • **Computing methodologies** → **Motion processing**; • **Information systems** → *Data compression.*

Additional Key Words and Phrases: animation compression, skeletal animation, rate–distortion optimization, entropy coding, learned probability model

1 Introduction

A skeletal animation clip stores rotations and translations for tens to hundreds of joints at every sample, yet much of this is not what the motion is about. A performer decides what to do; the body, its

*Co-corresponding authors.

Authors' Contact Information: Mingyi Shi, The University of Hong Kong, Hong Kong, Hong Kong, irubbly@gmail.com; Huancheng Lin, The University of Hong Kong, Hong Kong, Hong Kong, lamws233@gmail.com; Xuelin Chen, Adobe Research, London, United Kingdom, xuelin.chen.3d@gmail.com; Taku Komura, The University of Hong Kong, Hong Kong, Hong Kong, taku@cs.hku.hk.

morphology and its dynamics, determines most of how each joint gets from one place to the next. Compressing these data matters in production, where animation libraries face strict memory and streaming budgets, and in motion research, where corpora reach hundreds of hours and a model should spend its capacity on the decisions rather than on kinematics the body already implies. A codec is one way to probe this separation: what it must still send under a stated error bound is at least what the body did not predict. This is our motivation, not a claim the bit counts prove. If the body is only a prior, the representation should not be tied to one body: a production codec must serve arbitrary skeletons and motion without retraining, and it must bound its error, because a single visible pop on one joint is a defect regardless of how small the average error is.

ACL [Frechette 2023] is the industry standard. It folds the sub-tracks that do not move, reduces the range of the rest per clip and per segment, and quantizes each joint with one of a fixed set of bit widths under a per-joint error threshold. It decodes a pose in microseconds, but it models neither time nor the statistics of its own symbols. Learned motion encoders [Guo et al. 2024; Ling et al. 2020; Yao et al. 2024] take the opposite trade: their latent priors couple the representation to a training skeleton and domain, and they do not reach production precision.

Our previous work, CurveCodec [Shi et al. 2026], placed the learned prior on the smallest skeleton-independent unit of motion, the local joint curve, and reconstructed each curve from sparse anchors with a shared transformer. It reached three times ACL's ratio at matched *mean* error, but its worst case exceeded ACL's, its payload was counted in floating-point values rather than entropy-coded bits, and its decoder needed a GPU rather than a CPU core.

This paper asks a different question: *where is the redundancy in skeletal motion, and which part of a codec should a learned model take over?* We answer it by measurement, treating the codec as an instrument: under an explicit error contract and an exact bit count, every candidate structure of motion either pays or does not. We dissect ACL stage by stage and test every place where a learned component could enter. Three findings shape the result. First, ACL's ratio comes almost entirely from folding and from range reduction with variable bit widths. Its widths are nearly tight for independent samples, so an entropy coder on top of ACL's own symbols would gain little; a substantially smaller stream needs its own quantizer, one whose integers remain predictable in time. Second, at production precision the largest saving comes from predicting each curve from its own past, and the next from choosing, per joint and in closed loop through the hierarchy, which samples to omit. Leaving a sample out is by far the cheapest way to spend error, and its share of the stream grows as the precision loosens, until the two are comparable at the loosest precision we use. Third, on the gaps such an encoder leaves, interpolation has little left to find. A nearest-neighbour oracle with millions of training samples as memory, which clearly beats cubic interpolation on random gaps, is no better than *linear* interpolation on the gaps the encoder actually leaves, and the learned in-betweeners we placed inside the codec did not pay. On our data, the in-betweener at the core of CurveCodec has little left to learn; what a network does learn is the *distribution* of the residuals that the codec still has to send.

These findings lead to CurveCodec 2. Every animated sub-track is coded as a curve: rotations in the log map, which removes the precision floor of ACL's dropped quaternion component; a quantization step and a key set chosen in closed loop against the object-space error of the whole subtree; and prediction residuals coded with rANS under a small learned model of the curve's own past, a causal transformer or, in the worst-case configuration, an MLP. The model sees no joint identity, so the codec is skeleton-agnostic, and its integer inference is bit-exact on ARM and x86. Two contracts are verified on every decoded clip: ACL's own worst case per joint, within a stated tolerance, or ACL's mean error per clip. A clip that fails is re-encoded with a margin or stored as ACL's stream, and every number we report includes this fallback.

On a corpus of 906 hours from 33 datasets with a held-out test side, CurveCodec 2 needs 0.37× ACL's bytes at ACL's default precision of 0.01 cm under the worst-case contract, and 0.22× at 0.1 cm under either contract; against ACL with its default keyframe stripping the ratios are slightly larger and the picture is the same. It decodes on a CPU, typically a clip in a fraction of a second on one core, but only whole clips: it is a storage and distribution format decoded at load time, not a replacement for ACL's constant-time sampling.

Our contributions are:

- **A quantitative anatomy of motion redundancy**, measuring where a production codec's bits go and which structure a codec exploited in our hands: temporal predictability and, increasingly as the precision loosens, per-joint sample sparsity did; interpolation from the decoded neighbourhood, cross-joint structure, and transforms under a worst-case criterion did not.
- **CurveCodec 2, a closed-loop, skeleton-agnostic rate–distortion codec** under two explicit error contracts, with a stated fallback for the clips that fail.
- **A learned entropy model in the role where a network pays**: small, causal, and bit-exact across platforms, it models the distribution of the residuals the codec must send, the one place where, in our measurements, a network consistently earns its cost.
- **A corpus-scale benchmark against ACL**, with audits of ACL's configuration and of our own contracts, and same-core speed.

2 Related Work

Skeletal animation compression is a long-standing problem in computer graphics. Early work focused on making animation data smaller and faster to decode in production systems; learned representations later showed that motion can be organized into compact spaces that support reconstruction and synthesis. Our work sits between the two: an explicit, verifiable codec in the production tradition, with a learned component placed where our measurements show that learning pays.

2.1 Hand-engineered Animation Compression

Before deep learning, animation compression removed temporal redundancy, inter-joint redundancy, and perceptually insignificant

detail while keeping the motion faithful. Mesh compression began with static connectivity and geometry [Karni and Gotsman 2000; Rossignac 1999]; animated meshes were then compressed through their vertex trajectories, by principal components [Alexa and Müller 2000] or space-time prediction [Ibarria and Rossignac 2003]. Arikan [2006] moved the problem to skeletal motion: joint angles were converted into virtual marker trajectories, fitted with cubic Bézier curves, and projected into clustered PCA subspaces. Later work exploited low-dimensional structure with segment-level PCA and spline interpolation [Liu and McMillan 2006], quadratic Bézier break-and-fit curves [Khan 2016], wavelet coefficient selection [Beaudoin et al. 2007; Lee and Lasenby 2008], and wavelets guided by perception [Firouzmanesh et al. 2011]; others indexed repeated motion patterns across a database [Gu et al. 2009; Lin et al. 2011]. ACL [Frechette 2023] is the production standard. It prioritizes per-track processing and constant-time random access over compression ratio, and Sec. 5.1 quantifies where its bits go. Our keys relate to keyframe extraction [Halit and Capin 2011; Lim and Thalmann 2001] and curve simplification [Barkowsky et al. 2000], which choose samples from a local error or saliency criterion. CurveCodec 2 instead chooses them by rate–distortion optimization against the object-space error of the whole subtree, and Sec. 5.2 shows that this closed loop, not the key representation itself, carries the gain. Unlike transform codecs, which we find lose to sparse keys under a worst-case criterion, CurveCodec 2 keeps the time domain and predicts each sample from its own past.

2.2 Learned Motion Representations and Codecs

Motion manifolds and deep motion representations show that valid character motion occupies a much smaller space than raw joint rotations [Holden et al. 2016, 2015]. VAE-style priors support character control, action-conditioned synthesis, and pose estimation [Ling et al. 2020; Petrovich et al. 2021; Rempe et al. 2021]; tokenizers with vector or residual quantization compress motion into discrete codes [Guo et al. 2024; Yao et al. 2024]; phase-based models organize motion by periodic timing [Shi et al. 2023; Starke et al. 2022]; and Learned Motion Matching [Holden et al. 2020] amortizes an animation database into networks. Cross-skeleton methods learn representations for retargeting [Aberman et al. 2020; Lee et al. 2023; Villegas et al. 2018]. These representations target synthesis, control, or transfer at pose level, not a decodable bitstream at a stated precision; SAME [Lee et al. 2023] in particular learns an embedding shared across skeletons, which is a different goal from ours rather than a competing codec. The learned encoders measured in CurveCodec [Shi et al. 2026] under the shell metric reconstruct with errors two orders of magnitude above the 0.01 cm of a production codec, and the residual-VQ tokenizer we built has a precision floor of about 1.3 % of a track's RMS (Sec. 5.4); we do not extend these bounds to methods we did not measure. Implicit neural representations fit signals with coordinate networks [Chen et al. 2021; Dupont et al. 2021; Park et al. 2019; Sitzmann et al. 2020; Tancik et al. 2020], and NeMF [He et al. 2022] applies the idea to kinematic motion. We find the same fidelity floor for per-window neural fields. CurveCodec [Shi et al. 2026] is our direct predecessor. It made the learned prior skeleton-agnostic by working on single sub-tracks, and reconstructed each from a first-frame state, a Fourier descriptor, and extrema anchors with a masked transformer, in the spirit of learned in-betweening [Harvey et al. 2020]. CurveCodec 2 keeps the per-sub-track, skeleton-agnostic unit but moves the learned prior from the interpolator to the probability model, for reasons that Sec. 5.3 measures on our data.

2.3 Entropy Coding and Learned Probability Models

Lossless coding separates modelling from coding: an arithmetic coder [Witten et al. 1987] or an asymmetric numeral system [Duda 2013] spends $-\log_2 P$ bits per symbol, so the compression depends only on how well the model P predicts the data. Predictive lossless coders such as LOCO-I / JPEG-LS [Weinberger et al. 2000] and CALIC [Wu and Memon 1997] combine a fixed predictor with a context model of the residual, the structure CurveCodec 2 adopts per curve. Learned image compression [Ballé et al. 2017, 2018; Minnen et al. 2018] showed that much of the gain of a learned codec comes from the learned probability model, and that decoding with a network requires deterministic, integer inference to keep encoder and decoder in sync across platforms [Ballé et al. 2019]. General-purpose neural compressors use recurrent or transformer models as the probability model of a byte stream [Bellard 2021; Delétang et al. 2024; Goyal et al. 2019]. Learned probability models have entered animation compression before: Chen et al. [Chen et al. 2022] compress motion-capture sequences with a learned continuous-time latent model, and Qin et al. [Qin et al. 2025] code mesh animations with a deep entropy model over vertex trajectories; neither holds an object-space worst-case bound through a skeleton. CurveCodec 2 combines per-curve learned residual probabilities with hierarchy-aware quantization and key selection under explicitly verified object-space error constraints, and our experiments indicate that for skeletal motion the probability model is the role where a network pays, while the learned interpolators, transforms, tokenizers, and neural fields we tried did not pay for themselves.

3 Preliminaries

3.1 Skeletal Animation and Error

A skeleton has J joints connected as a rooted tree; joint j has parent $\pi(j)$. A clip of N uniformly spaced samples stores, per joint and sample, a local rotation $\mathbf{q}_j(t)$ (a unit quaternion) and a local translation $\mathbf{t}_j(t) \in \mathbb{R}^3$; BVH data carry no scale. Following ACL, the time series of one transform component of one joint is a *sub-track*. Skeleton-agnostic means that the same codec and weights apply to any joint count and tree topology without retraining. For comparability with ACL we use its raw size of ten 32-bit floats per joint-sample, but report rate mainly in *bits per joint-sample* and as our bytes divided by ACL's on the same clips. Sizes include every header; shared model weights are reported separately.

Shell error. We use ACL's shell-distance metric [Frechette 2023]. Forward kinematics gives the object-space transform of every joint,

$$\mathbf{q}^o_j = \mathbf{q}^o_{\pi(j)} \otimes \mathbf{q}_j, \qquad \mathbf{x}^o_j = \mathbf{q}^o_{\pi(j)}\, \mathbf{t}_j\, \overline{\mathbf{q}^o_{\pi(j)}} + \mathbf{x}^o_{\pi(j)}, \tag{1}$$

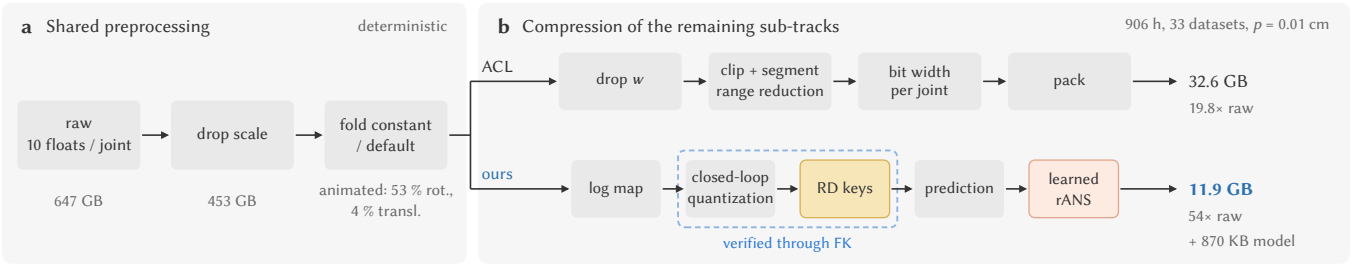


Fig. 2. The two-stage view of animation compression, with the estimated size of our 906-hour corpus after each stage at $p = 0.01$ cm (extrapolated from the test side). (a) Deterministic preprocessing shared by ACL and CurveCodec 2: dropping the absent scale sub-tracks and folding sub-tracks that do not move. (b) ACL quantizes the remaining sub-tracks with per-segment range reduction and fixed bit widths (32.6 GB); CurveCodec 2 codes them as curves with a closed-loop quantizer, RD-selected keys verified through forward kinematics, prediction, and a learned entropy coder (11.9 GB). The 870 KB model is fixed after training and shared by all clips; it is not counted in the stream.

and the error of joint j at sample t is the largest displacement of three virtual vertices at distance $\delta = 3$ cm along its axes,

$$e_{j,t} = \max_{a \in \{x,y,z\}} \left\| T_j^o(t)\,(\delta \hat{\mathbf{e}}_a) - \hat{T}_j^o(t)\,(\delta \hat{\mathbf{e}}_a) \right\|, \tag{2}$$

where T_j^o and $\hat{T}_j^o$ are the original and decoded object-space transforms, so the error of a joint contains the errors of all its ancestors. The vertices sample the sphere, so Eq. (2) can understate the largest displacement on it (by up to a factor $\sqrt{2/3}$ for a pure rotation error); the ACL commit we benchmark uses this approximation, which its development branch has since replaced with the exact maximum (supplementary Sec. A). Both codecs are measured with Eq. (2), at the sample times only as ACL does; our implementation matches the clip maximum our ACL build reports within 10^{-4} cm.

3.2 Two-Stage Compression Pipeline

As in CurveCodec, we view compression as two stages (Fig. 2). The first, deterministic preprocessing shared by ACL and CurveCodec 2, drops the scale sub-tracks that ACL's raw convention reserves and *folds* every sub-track that does not move within p into a single default or constant value. Production skeletons contain many such joints: on our corpus, dropping scale reduces ACL's raw 647 GB to 453 GB, and at $p = 0.01$ cm folding leaves only 53 % of the rotation and 3.6 % of the translation sub-tracks animated. CurveCodec 2 applies ACL's folding rule unchanged, so both codecs start the second stage from the same animated sub-tracks.

ACL's compression stage. ACL drops the w component of each quaternion and reconstructs it from the unit norm. It normalizes every animated sub-track to its clip range and to the range of each 16-sample segment, and gives each joint one of 25 bit widths, chosen per segment by a search that raises widths until the object-space error of the joint and its descendants meets p. We use ACL's development branch after release 2.1.0 (commit `3ee5685`), whose newer search does not read ACL's compression level; supplementary Sec. A describes it and compares the release. ACL performs no prediction and no entropy coding: every sample of a segment costs the same number of bits, which gives constant-time random access. It can also strip whole frames that linear interpolation of their neighbours reproduces within p (on by default). Our corpus becomes an estimated 32.6 GB, extrapolated from the test side. CurveCodec 2 replaces this stage (Sec. 4) with log-map rotations, steps and keys chosen in closed loop through forward kinematics, prediction, and learned entropy coding; the corpus then becomes an estimated 11.9 GB, plus one 870 KB model shared by all clips.

3.3 Error Contracts

ACL compresses to a precision p (default 0.01 cm) but does not guarantee it: its widths come from a fixed set and its dropped w has a precision floor, so a clip's maximum error stays within p on only 1.5 % of our test clips at 0.01 cm. Comparing codecs at the same p is fair only if both are held to the same *measured* error. We check every clip against ACL's result on that clip, with $\bar{e}_j^{\mathrm{ACL}}$ and μ_j^{ACL} the maximum and mean error ACL reaches on joint j, μ^{ACL} its clip mean, and f^{ACL} the fraction of joint-samples it leaves above p.

Max gate. Every joint stays within 5 % of the larger of p and ACL's own maximum on that joint, and the fraction of samples above p within half a percentage point of ACL's:

$$\begin{aligned} \max_t e_{j,t} &\le 1.05 \max\!\left(p, \bar{e}_j^{\mathrm{ACL}}\right) \quad \forall j, \\ \tfrac{1}{JN}\left|\{(j,t) : e_{j,t} > p\}\right| &\le f^{\mathrm{ACL}} + 0.005. \end{aligned} \tag{3}$$

This bounds a joint by $1.05\,p$ where ACL meets p and by ACL's own maximum where it does not; a joint ACL codes far below p may be coded up to $1.05\,p$. It holds ACL's worst case *within tolerance*, not identically; Sec. 6.3 measures what the tolerances are worth.

Mean gate. The clip keeps ACL's mean $m_p = \max(\mu^{\mathrm{ACL}}, \phi p)$, each joint stays near its share, and a guard bounds the worst case:

$$\begin{aligned} \tfrac{1}{N}\textstyle\sum_t e_{j,t} &\le \rho \max\!\left(m_p, \mu_j^{\mathrm{ACL}}\right) \quad \forall j, \\ \tfrac{1}{JN}\textstyle\sum_{j,t} e_{j,t} &\le \tau\, m_p, \\ \max_t e_{j,t} &\le \max\!\left(\gamma\, m_p,\ 1.05\,\bar{e}_j^{\mathrm{ACL}}\right) \quad \forall j, \end{aligned} \tag{4}$$

with $\rho = 1.05$, $\tau = 1.02$, $\gamma = 10$, and $\phi = 0$. The guard is ten times ACL's clip mean, not a multiple of p: at $p \le 0.1$ cm the resulting worst case is below ACL's, but at loose precision it admits large isolated errors, so the mean gate suits storage at tight precision.

Verification and fallback. The encoder decodes its own stream and checks Eq. (3) or (4) on every joint. A failing clip is re-encoded with its internal limits scaled by 0.98, then 0.95, then 0.90; if none passes, or if the final stream is larger than ACL's, ACL's own stream is stored, which satisfies either contract by definition. All test-side byte counts include this fallback.

4 CurveCodec 2

Fig. 3 gives an overview. As in CurveCodec, the unit of coding is one animated sub-track, a *curve*: one joint's rotation (three log-map components) or translation (three components) over time; each coded component becomes one integer *column* of the entropy coder. The skeleton enters the codec in one place: the encoder evaluates the shell error through forward kinematics to decide how coarsely each curve may be coded. The learned model never sees joint identity; its only hierarchy-derived input is a four-class depth bucket that the decoder computes from the skeleton's parent array. Sections 4.1 to 4.3 describe the lossy part, which determines the decoded clip, and Sec. 4.4 the lossless part, which determines only the number of bits, so the network cannot violate the error contract.

4.1 Representation

Folding. Folding follows ACL's rule. On long chains (tails, finger rigs) small folding errors can accumulate below each joint's threshold yet push a descendant over its limit; when the unquantized state already violates the contract, we un-fold sub-tracks on the offending root paths, largest error first. In the benchmarked max-gate codec this runs only in the first quantization stage, which causes most loose-precision failures (Sec. 6.3).

Log-map rotations. ACL stores the (x, y, z) part of a quaternion and reconstructs w, which amplifies quantization error near $|w| \approx 0$ and leaves a precision floor: on our curated set ACL's maximum error is 0.071 cm at every precision from 0.01 down to 0.001 cm. We instead code the rotation vector [Grassia 1998]

$$\mathbf{v} = \operatorname{Log}(\mathbf{q}) = 2\,\operatorname{atan2}\big(\|\mathbf{q}_{xyz}\|, q_w\big)\,\frac{\mathbf{q}_{xyz}}{\|\mathbf{q}_{xyz}\|}, \tag{5}$$

after choosing quaternion signs so that consecutive samples lie in the same hemisphere ($\mathbf{q}_t \cdot \mathbf{q}_{t-1} \geq 0$, with $q_w \geq 0$ at $t = 0$), so that $\|\mathbf{v}\| \in [0, 2\pi]$. Where $\|\mathbf{q}_{xyz}\|$ vanishes (the identity, or $\mathbf{q} = -1$ after a full turn) the direction is undefined and the limit depends on the direction of approach; by convention we take $\mathbf{v} = 0$, a branch choice that Exp maps back to the same rotation in both cases. The map needs no side information, bounds the rotation error by $\|\Delta\mathbf{v}\|$, and lowers the curated maximum at 0.01 cm from 0.071 to 0.024 cm. It is continuous only short of a full turn: when a joint's rotation relative to its parent passes 2π, $\mathbf{v}$ jumps from about $+2\pi\,\mathbf{u}$ to $-2\pi\,\mathbf{u}$. Since $\mathbf{v}$ and $\mathbf{v} + 2\pi\,\mathbf{u}$ are the same rotation in $SO(3)$ (their quaternions differ only in sign) and the contract is checked on the decoded clip, this costs a few keys, not correctness; it occurs on 0.33 % of the test side's rotation tracks and costs below 0.2 % of the bytes (supplementary Sec. C, which also evaluates an unwrapping rule). Translations are coded in centimetres without any transform.

4.2 Closed-Loop Quantization

Each curve is quantized uniformly, $n_t = \operatorname{rint}(v_t/s)$, $\hat{v}_t = n_t s$, with the step on a logarithmic grid $s = 2^{i/G}$, $i \in \mathbb{Z}$, $G = 16$, so that the header stores a small integer. Because a joint's error is amplified by the chain it moves, we start from

$$s_{\text{rot}} = \frac{2\,p}{\sqrt{3}\,R_j\sqrt{c_j}}, \qquad s_{\text{trans}} = \frac{p}{\sqrt{3}\,\sqrt{c_j}}, \tag{6}$$

where R_j is the shell distance plus joint j's longest child chain and c_j the number of joints on the longest root-to-leaf chain through j. A closed loop then grows each track's step while the contract holds: every candidate is decoded, forward kinematics is recomputed for the joint's subtree, and Eq. (3) or (4) is tested on every joint of the subtree. The search passes leaves to root and back, offering $1, 2, 4, \ldots$ grid units and bisecting on refusal; because the error is not monotone in the step, it looks up to half an octave past a refusal, which alone saves 1 to 7 % under the mean gate. Under the max gate the three rotation components may then grow separately. The root keeps the finest steps; leaves end 2 to 5× coarser than Eq. (6).

4.3 Rate–Distortion Keys

Each track keeps a key set $\mathcal{K} \subseteq \{0, \ldots, N-1\}$ with the first and last sample, and the decoder fills the rest by Catmull–Rom interpolation [Catmull and Rom 1974] of the log-map values,

$$\begin{aligned} \hat{\mathbf{v}}(t) &= h_{00}(u)\,\hat{\mathbf{v}}_k + h_{10}(u)\,\Delta_k\mathbf{m}_k + h_{01}(u)\,\hat{\mathbf{v}}_{k'} + h_{11}(u)\,\Delta_k\mathbf{m}_{k'}, \\ \mathbf{m}_k &= \frac{\hat{\mathbf{v}}_{k^+} - \hat{\mathbf{v}}_{k^-}}{t_{k^+} - t_{k^-}}, \qquad u = \frac{t - t_k}{\Delta_k}, \end{aligned} \tag{7}$$

with consecutive keys k, k', $\Delta_k = t_{k'} - t_k$, neighbouring keys $k^{\pm}$, and cubic Hermite basis functions $h..$; rotations are recovered with Exp. The cubic saves 0.5 to 2.6 % over linear interpolation in the same rotation-vector coordinates; this is not spherical linear interpolation [Shoemake 1985], which it matches only for coaxial rotations, and every interpolation baseline in this paper is log-linear. Keys, like steps, are chosen in closed loop through the hierarchy.

Max gate. From the all-samples state, the encoder removes keys in greedy rounds: it proposes every fourth interior key of every track, decodes the affected window, recomputes the subtree's error, and accepts the removal if every joint stays within its limit and the clip within its over-p allowance. A narrow failure may be rescued by moving the two neighbouring key values up to two steps towards their least-squares fit of the window. Rounds alternate leaf-to-root and root-to-leaf passes until nothing is accepted.

Mean gate. Under the iso-mean contract the budget is a sum, so moves can trade between curves. A Lagrangian ladder [Sullivan and Wiegand 1998] lets two moves compete: removing a key, with distortion ΔD the increase of the summed error over the window and subtree and rate ΔR estimated from the residuals it removes, and growing a step by g grid units, with $\Delta R = \beta \cdot 3\,|\mathcal{K}|\,g/G$ and $\beta = 2.5$ calibrated against measured bits. At level λ every move with $\Delta D/\Delta R \leq \lambda$ that keeps Eq. (4) is accepted; λ starts at $10^{-4} m_p$ per bit and grows by $\sqrt{2}$ until the budget is spent, about 60 % on steps and 40 % on keys.

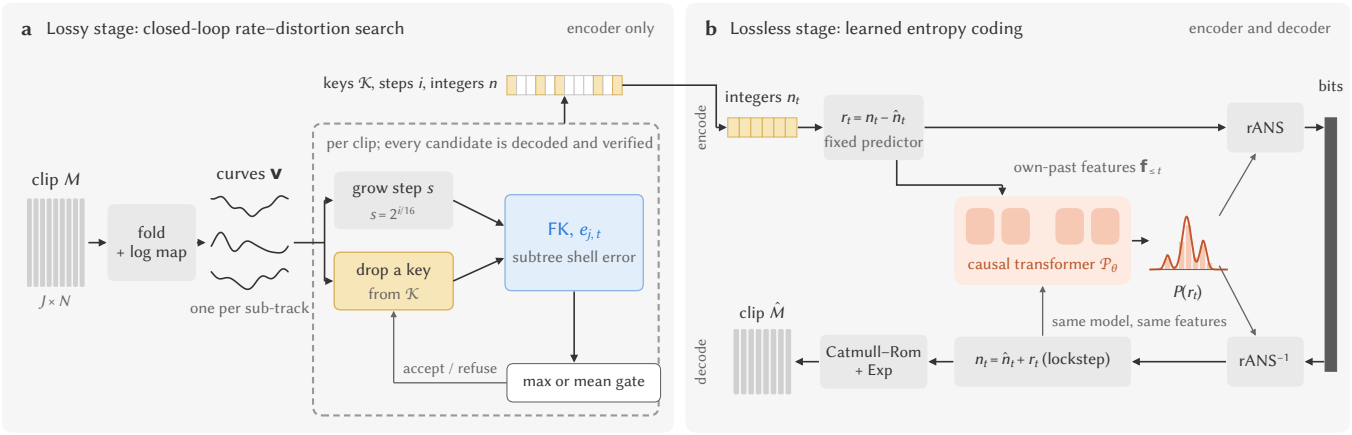


Fig. 3. Overview of CurveCodec 2. (a) The lossy stage runs once per clip in the encoder. Every animated sub-track becomes a curve in the log map; a closed loop then proposes two kinds of moves, growing a track's quantization step or dropping one of its keys, decodes the candidate, evaluates the shell error of the affected subtree through forward kinematics, and accepts the move only if the error contract still holds. The skeleton's geometry is used here and nowhere else; the lossless stage reads only each joint's depth in the hierarchy. (b) The lossless stage codes the resulting integers. A fixed predictor turns them into residuals, and a small learned model, the only learned component (a causal transformer in the product, an MLP under the max gate), reads features of each curve's own past and outputs a mixture distribution that drives the rANS coder. The decoder replays the same model in lockstep over all curves, so it reproduces every probability exactly, and reconstructs the clip by Catmull–Rom interpolation between keys.

Track modes. Each track is finally coded in the cheapest of four modes: static, all samples, one key set shared by its components, or per-component sets that may only *thin* the shared set, with a context-coded presence flag per component and key.

4.4 Prediction and Learned Entropy Coding

After the lossy stage a clip is a set of integer columns, the quantized values of each coded component at its keys (at most six per joint, 468 for a 78-joint clip before folding).

Prediction and alphabet. Each column is predicted from its own past, $r_t = n_t - \hat{n}_t$, with a predictor chosen per column among first and second differences, a second difference scaled by the ratio of key gaps, and an order-4 integer NLMS filter [Haykin 2002] on first differences. A residual is split into a symbol (zero or the sign, the bit length of $|r_t|$, and the first bit below its leading one; 159 symbols) and raw low bits. Each symbol covers an interval of residual values, so a continuous distribution over r_t induces symbol probabilities through its cumulative distribution.

Learned probability model. A network $\mathcal{P}_\theta$ sees only the column's own decoded past through 36 integer features $\mathbf{f}_t$: the last eight residuals as signed log magnitudes, five exponential averages of $|r|$, first and second differences of the decoded integers, the key gaps around the position, the depth class (root, one to two, three to five, six and deeper), the sub-track kind, predictor, log step, frame rate, precision, and position in the column; none identifies a joint, a skeleton, or a dataset. $\mathcal{P}_\theta$ is a causal transformer, two pre-norm blocks of width 64 with four heads over the last $W = 128$ positions (108 K parameters), whose head outputs a mixture of $K = 3$ logistics [Salimans et al. 2017]; the symbol covering $[a, b]$ has probability

$$P(a \le r_t \le b \mid \mathbf{f}_{\le t}) = \sum_{k=1}^{K} w_k \left[S_k\left(b + \tfrac{1}{2}\right) - S_k\left(a - \tfrac{1}{2}\right) \right], \qquad (8)$$

ALGORITHM 1: Lossy encoder of CurveCodec 2. Every acceptance test decodes the candidate and evaluates the shell error of the affected subtree exactly.

Input: clip M, precision p, contract (max or mean), ACL's per-joint statistics
Output: key sets $\mathcal{K}$, step indices i, integers n
fold constant and default sub-tracks (ACL's rule); un-fold on chains that break the contract;
convert rotations to log maps with sign continuity along time (Eq. 5);
initialize steps (Eq. 6); $\mathcal{K} \leftarrow$ all samples;
repeat `// object-space step search`
  for *joints j leaf-to-root, then root-to-leaf* **do**
    grow i_j by $1, 2, 4, \ldots$ grid units with lookahead; accept if the subtree satisfies the contract;
  end
until *no step changes*;
if *max gate* **then**
  repeat
    remove candidate keys whose subtree stays within its limits (with neighbour refit);
  until *no key removed*;
else
  for $\lambda \leftarrow 10^{-4} m_p,\ \sqrt{2}\lambda,\ \ldots$ **do**
    accept key removals and step growths with $\Delta D / \Delta R \le \lambda$ that keep Eq. (4);
  end
end
choose per-track mode and component key thinning by coded cost;

with $S_k(x) = \mathrm{sig}\big((x - \mu_k)/\sigma_k\big)$, quantized to 15-bit frequencies and coded with rANS [Duda 2013]. Residuals are ordered time-major over all columns, so the model runs once per index as a batch over

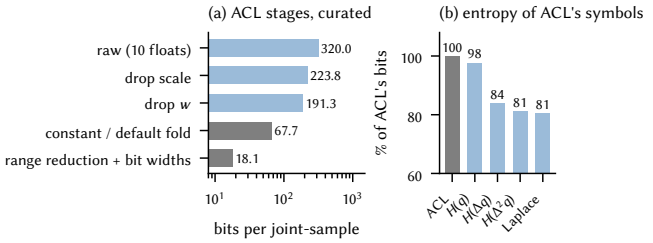


Fig. 4. Where ACL's bits go on the curated set. (a) Bits per joint-sample after each ACL stage: almost all of the 17.7× ratio comes from folding sub-tracks that do not move (2.84×) and from the two-level range reduction with variable bit widths (3.74×). (b) Entropy of ACL's own quantized symbols relative to the bits ACL spends, for zeroth- to second-order models: an entropy coder with these models on top of ACL saves at most 16 to 20 %, so a substantially smaller stream needs a different quantizer, not just a better back end.

the active columns, and the decoder reconstructs each integer before the next features are computed. For the max-gate codec we use a 22 K-parameter MLP (36 → 128 → 128 → 9) over the same features, 4× cheaper to decode.

Bit-exact inference. Encoder and decoder must compute identical probabilities. We export the network to fixed point: 16-bit weights with power-of-two scales, 64-bit integer accumulators, integer layer normalization, and lookup tables for softmax, sigmoid, and exponential. The residual stream is clipped to $\pm 2^{28}$, which bounds every operand downstream: linear-layer products stay below 2^{43} and their sums over at most 256 terms below 2^{53}, so even a float64 matrix library adds exact integers [Ballé et al. 2019]; the attention logits and the layer-norm variance are accumulated in 64-bit integers within the bounds given in Appendix H. The guarantee is that every probability and every decoded integer is identical on any platform. The float reconstruction (dequantization, interpolation, Exp) runs on those integers and reproduced x86 results bit for bit on an ARM machine in our tests, but it is deterministic rather than exact by construction (Appendix H).

Training. The model minimizes the code length of Eq. (8) on residual streams of our training split (886 h, 41.2 G symbols), on 40 M windows with 128 context positions and 384 scored ones, with AdamW [Loshchilov and Hutter 2019] and a one-cycle schedule, in about 1.5 hours on one RTX 4090. Supplementary Appendix G summarizes the bitstream layout; the reference decoder defines the details it leaves out.

5 Where Is the Redundancy in Motion?

These measurements motivate the design. Unless stated otherwise, they use two development sets that predate the full corpus: *curated*, 70 clips from 14 mocap datasets (15.9 M joint-samples), and *dev*, 15 clips. Every variant was verified against its contract on every clip; Sec. 6 tests the final codec on held-out data.

5.1 ACL's Bit Budget

Fig. 4a follows ACL's stages on the curated set. Two stages do the work: folding sub-tracks that do not move (2.84×, from 191 to 68 bits per joint-sample) and range reduction with variable bit widths (3.74×, to 18.1 bits); the 4× spread of ACL's ratio across datasets comes from folding. Of the remaining bytes, 81 % are animated rotations and 13.5 % per-segment range metadata, a share that grows to 24 % at 0.1 cm on the test side because it does not shrink with the precision. Fig. 4b asks what a lossless back end could add. The zeroth-order entropy of ACL's quantized samples is 97.7 % of the bits it spends, so its widths are nearly tight for independent samples; first and second differences reach 83.8 and 81.1 %, and real context coders 0.80×. These bounds hold for the models we tried, up to second-order temporal context; little is left because range reduction stretches every 16 samples to fill their integer range.

Consequence. A substantially smaller stream needs its own quantizer. Ours is close to ACL in zeroth-order entropy (0.92 to 1.07×), but its first differences cost 0.57 to 0.62× ACL's bits against 0.84× for ACL's own symbols: the gain appears once time is modelled.

5.2 Samples Predicted and Samples Not Coded

Temporal predictability. Predicting each integer from its own past reduces the log-map stream by 39 to 55 % against its zeroth-order cost. The best predictor depends on the joint and capture: second differences beat first differences by 18 % at the root but only 4 to 7 % at depth six and below, where noise at the scale of the step dominates, and an adaptive (NLMS) predictor wins on 37 to 44 % of the curves, which motivates a per-curve choice.

Per-joint sparsity. Not every sample needs coding. Under the max gate the encoder keeps 87 % of the root's samples but only 57 % at depth six and below at 0.01 cm, and 51 against 22 % at 0.1 cm (Fig. 5a; five development clips, an earlier version of the encoder); the root keeps the most because its error reaches every descendant. ACL can only strip whole frames whose every joint is linearly interpolable, 0.38 % of the curated frames at 0.01 cm. Where keys pay, the gain comes from the closed loop: removing keys by a local per-curve error bound saves only 0 to 0.6 %.

The price of a bit. At the margin, keys are the cheapest way to spend error. Fig. 5c compares the distortion the iso-mean encoder adds per bit it saves at 0.1 cm: growing a step by one grid unit costs 0.17 step units, a dead zone 0.31, and removing an RD-selected key about 0.03. This says which move to prefer, not how much each move contributes in total, which depends on how many cheap moves the precision leaves: on held-out data, removing keys from the final codec costs 8 % at 0.01 cm and 69 % at 1 cm, where it matches prediction (Sec. 6.5). Codecs that code every sample lose accordingly: a dyadic hierarchical codec with a learned in-betweener matches ours where keys keep 83 % of the samples (1.01×) but needs 1.55 to 1.78× the bytes where they keep 7 to 15 %.

Consequence. Prediction is the largest saving at $p \leq 0.1$ cm and the choice of which samples to omit the second, with a share that grows as p loosens. The encoder needs both, with steps and keys competing for one budget.

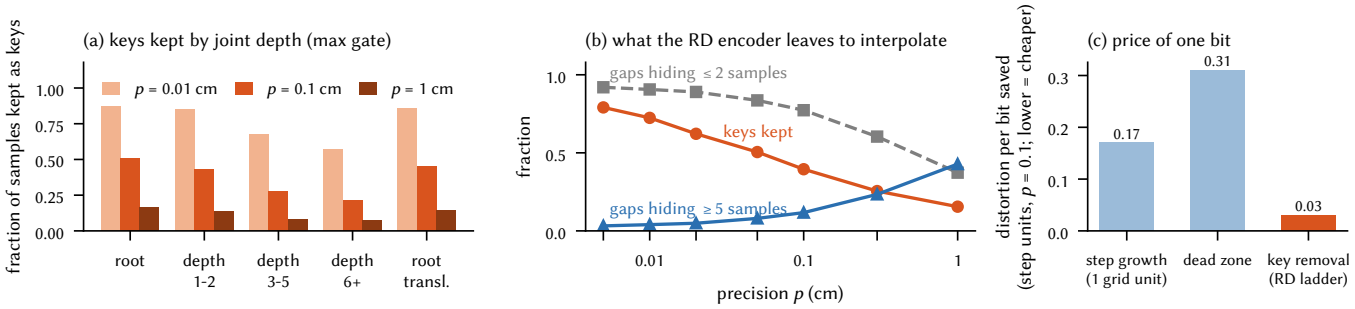


Fig. 5. How the encoder spends distortion. (a) Fraction of samples kept as keys by joint depth (five development clips, an earlier version of the encoder): the root, whose error reaches every descendant, keeps the most keys, and leaves the fewest. (b) The keys the encoder keeps fall from 79 % to 16 % of the samples as p loosens, yet at $p \le 0.1$ cm more than three quarters of the remaining gaps hide at most two samples. (c) The marginal distortion an encoder move adds per bit it saves, at $p = 0.1$ cm: removing a key is 5 to 10× cheaper than growing the quantization step or adding a dead zone, which is why the encoder prefers to omit samples once the precision leaves room to.

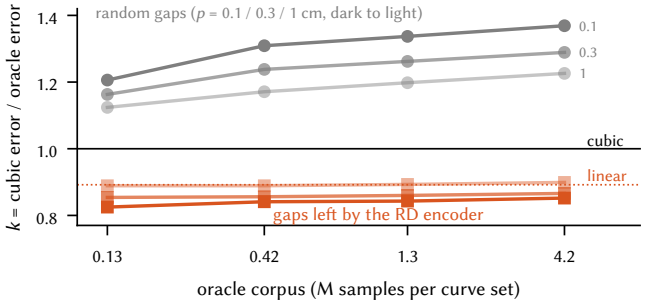


Fig. 6. Interpolation from the decoded neighbourhood gains little on the gaps an RD encoder leaves. A nonparametric nearest-neighbour oracle searches every window of the training corpus for the best match of a gap's own-curve context (k is cubic error divided by oracle error; $k > 1$ is better than cubic). On random gaps (grey) it beats cubic interpolation by up to 1.37× and keeps improving with the corpus. On the gaps the RD encoder actually leaves (orange) it is no better than linear interpolation and is flat in the corpus size over the range we could measure.

5.3 Interpolation Gains Little on the Gaps the Encoder Leaves

Where keys carry the bytes, a better interpolator should buy more: one that halved the cubic error everywhere ($k = 2$) would save 3 to 27 %. This is the bet of CurveCodec and of learned in-betweening; we tested it, and the results below bound what the estimators we tried achieved on our data, not what any interpolator could. The gaps the encoder leaves are short: under the max gate at $p \le 0.1$ cm, 77 to 92 % hide at most two samples (Fig. 5b), whereas published in-betweeners train on gaps of 5 to 45 frames at errors 60× above our default precision. In our gaps the cubic residual is linearly unpredictable from the curve's own context (ridge $R^2 \le 0.02$).

A retrieval oracle. As a strong nonparametric reference, an oracle searches every window of 510 training clips (4.2 M samples, held-out skeletons) for the best match of a gap's decoded context and copies the matched interior as a residual over cubic interpolation. It is a nearest neighbour in context space over those 4.2 M samples, a 510-clip subset of the 886-hour training split the entropy model sees. On random gaps it beats cubic interpolation by 1.37, 1.29, and 1.23× at $p = 0.1$, 0.3, and 1 cm and keeps improving with the corpus size (Fig. 6). On the gaps the iso-mean RD encoder leaves, it reaches only 0.85 to 0.90, no better than *linear* interpolation on the same gaps, and is flat over the 32-fold range of library size we measured. The encoder removes exactly the samples a smooth prior already predicts and keeps contacts, impacts, and direction changes with no precursor in the curve's past. A better interpolator might also let the encoder drop keys it now keeps; we placed four learned in-betweeners inside the encoder, including a 3.4 M-parameter transformer with keys re-chosen for it. None paid (supplementary Sec. F): they changed the size by −2.5 to +5.7 %, no more than the null option of choosing per gap between cubic and linear interpolation would give.

Consequence. Under an explicit error contract and an explicit bit count, none of the interpolators we built found enough information in the transmitted context to pay for itself. These experiments ran under the iso-mean contract; for the max gate, whose gaps are shorter still, the conclusion is an inference we did not test. CurveCodec 2 therefore keeps a fixed interpolator and moves the learned prior to the probability model.

Table 1. Every learned role we tested, each as a complete codec under the iso-mean contract and verified on every clip (dev set; bytes relative to our codec with the MLP entropy model). Only the learned probability model of the residuals pays.

Network role	Representation	0.01	0.1	1 cm
Entropy model	keys + residuals	**0.984**	**0.982**	**0.983**
Correction	keys + fixes	1.00 to 1.04 (unused)		
Transform + model	lifting wavelet	1.03	1.09	1.15
In-betweener + model	dyadic grid	1.10	1.35	1.94
Tokenizer	residual VQ	1.10	1.39	2.09
Neural field	per-window INR	–	2.0	3.3

5.4 Where a Network Pays

Tab. 1 summarizes every role we gave a network. Dense learned representations have a precision floor: a residual-VQ tokenizer cannot represent a track below about 1.3 % of its RMS and a meta-learned neural field not below 1.6 to 4.8 % of the window's amplitude, while the contract asks for 0.01 to 0.1 %, so both degenerate into scalar residual codecs. In a lifting-wavelet codec of ours, learning the transform changed the size by ±1 % while learning its entropy model saved 7 to 12 %; learned image codecs report a similar division of gains [Ballé et al. 2018; Minnen et al. 2018], an analogy rather than evidence for our numbers. Over a hand-designed context coder, the MLP entropy model saves 3 to 7 % on mocap and 13 % on game assets. It does not predict the next residual's value ($R^2 \leq 0.02$) but its distribution, from a slowly varying per-curve regime: the residual's scale relative to the step, periodicity, and whether a curve is held, linear, or noisy. Eight positions of context give 60 to 70 % of the gain. Supplementary Sec. F reports that cross-joint structure (mirroring, pose-space PCA, interpolation context) and transforms under a worst-case contract did not pay on our data.

Consequence. The network belongs in the probability model, where it models a distribution the codec must send anyway, and it can be small, causal, and local.

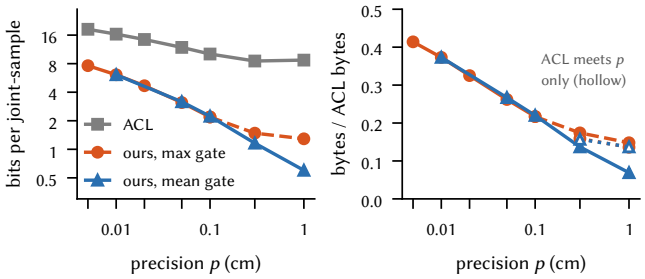


Fig. 7. Rate against precision on the held-out test side (4,472 clips, 20.1 h, 33 datasets). Left: bits per joint-sample. Right: our bytes divided by ACL's on the same clips. Under the max gate (orange), which holds every joint within 5 % of the larger of p and ACL's maximum, CurveCodec 2 needs 0.41× to 0.22× ACL's bytes for $p \leq 0.1$ cm; under the iso-mean criterion (blue) the ratio keeps falling to 0.07× at 1 cm. All ratios include the fallback for clips that fail their contract. At 0.3 and 1 cm ACL misses its own precision on part of the data (dashed lines, whole test side); hollow markers on dotted lines repeat the ratio on the clips where ACL keeps at least 99 % of its joint-samples within p.

6 Experiments

6.1 Setup

Corpus and split. We assembled 192,392 clips (906 h, 16.17 G joint-samples) from 33 datasets, including HiPHI [Ji et al. 2026], a large mocap dataset released online (*large online mocap* below), Motion-X [Lin et al. 2023], BEAT [Liu et al. 2022], AMASS [Mahmood et al. 2019], MotionPersona [Shi et al. 2025], 100STYLE [Mason et al. 2022], CMU [CMU Graphics Lab 2019], LAFAN1 [Harvey et al. 2020], ZeroEGGS [Ghorbani et al. 2023], and PFNN [Holden et al. 2017], retargeted and converted versions of several of them, hand capture, and 549 game-asset clips of 196 skeletons, at 20 to 120 Hz. Because the corpus holds retargeted, cropped, and converted versions of the same performances, we split by *take group*, a key that follows the original performance across datasets and conversions, holding out 2 of every 100 clips per dataset: the test side has 4,472 clips, 20.1 h, and 342.7 M joint-samples, and no take group appears on both sides. A quadruped dataset [Zhang et al. 2018] (dog) is held out of training entirely. The test side also contains the 71 development clips of Sec. 5 (5.1 % of its joint-samples); without them the ratios change by less than one point.

Baselines and metrics. ACL [Frechette 2023] (commit `3ee5685`) runs with its default settings at p = 0.005, 0.01, 0.02, 0.05, 0.1, 0.3, and 1 cm, with keyframe stripping off because our contracts take their targets from the no-strip result; we report our ratio against ACL's default stripping as well. CurveCodec 2 runs under the max gate with the MLP entropy model and under the mean gate with the transformer, our product configuration. Learned motion encoders are not included; their errors are two orders of magnitude above ACL's at these precisions [Shi et al. 2026]. We report bits per joint-sample, the ratio of our bytes to ACL's on the same clips, and the mean, 99th-percentile, and maximum shell error over all joint-samples. Timing uses one pinned core of an AMD Threadripper PRO 7975WX for both codecs.

6.2 Compression against ACL

At ACL's default precision of 0.01 cm, CurveCodec 2 needs 6.1 bits per joint-sample where ACL needs 16.4, 0.37× ACL's bytes under either contract (Tab. 2, Fig. 7). The ratio improves as p loosens because ACL's segment metadata and widths have a floor while our keys thin out. Under the max gate our P99 is within 1.08× of ACL's and our mean error 1.4× higher; under the mean gate our mean matches ACL's on every clip and our maximum is lower at $p \leq 0.1$ cm (0.094 against 0.213 cm at 0.01 cm). At 0.3 and 1 cm ACL itself exceeds p on part of the data, which both contracts inherit. Against ACL with its default keyframe stripping, which removes 4.8 % of the frames at 0.01 cm and 29 % at 0.1 cm without changing its error, our ratios are 0.380× and 0.25 to 0.26×. ACL's compression level changes no byte in our build. The released ACL 2.1.0 writes 2 to 7 % more bytes at a tighter worst case, a comparison of bytes only, since our contracts were not re-matched to its errors. xz or zstd over ACL's stream saves 2 to 5 % (supplementary Sec. A). Extrapolated per dataset (Fig. 8), our 906-hour corpus would take an estimated 11.9 GB at 0.01 cm, against the 32.6 GB that ACL would need.

Fig. 9 shows decoded motion for a humanoid, a game character, a creature, and the held-out dog. At 0.01 cm, decoded and original skeletons coincide for all three codecs while our streams are 2 to 3× smaller; at 1 cm the largest joint displacement in the frames shown is about half a centimetre for all three (the clip maxima are in the caption), at 4.4 to 4.7× fewer bytes than ACL. A static frame cannot show popping; the evidence about the worst frame is the contracts, checked on every sample.

Table 2. Compression and error on the held-out test side (4,472 clips, 342.7 M joint-samples). Ratio is our bytes divided by ACL's, after the fallback (Sec. 3.3); errors are shell errors in cm over all joint-samples of the delivered streams. The max gate holds every joint within 5 % of the larger of p and ACL's maximum and spends the room below the worst case, so its mean error is higher; the mean gate holds ACL's mean with a flatter distribution (higher P99) and a lower maximum for $p \leq 0.1$ cm. At 0.3 and 1 cm ACL misses its own precision on 6.7 and 13.5 % of its joint-samples, and the mean gate's guard admits the large maxima shown; ratios in parentheses are on the clips where ACL keeps at least 99 % of its joint-samples within p.

	ACL				Ours, max gate					Ours, mean gate				
p (cm)	bits	Mean	P99	Max	ratio	bits	Mean	P99	Max	ratio	bits	Mean	P99	Max
0.005	18.43	–	–	–	0.414	7.62	–	–	–					
0.01	16.38	0.0034	0.0111	0.213	**0.373**	6.11	0.0048	0.0120	0.211	**0.373**	6.11	0.0033	0.0119	0.094
0.02	14.39	–	–	–	0.325	4.68	–	–	–					
0.05	11.87	0.0166	0.0508	0.539	**0.262**	3.11	0.0236	0.0540	0.536	0.267	3.17	0.0166	0.0576	0.480
0.1	10.11	0.0333	0.1003	0.916	**0.217**	2.19	0.0466	0.1054	0.845	0.220	2.23	0.0332	0.1155	0.903
0.3	8.53	0.137	0.379	2.39	0.174 (0.159)	1.48	0.173	0.395	2.38	0.137 (0.157)	1.16	0.134	0.445	7.9
1	8.72	0.560	1.47	7.17	0.148 (0.136)	1.29	0.646	1.50	6.97	0.069 (0.137)	0.60	0.542	1.73	24.7

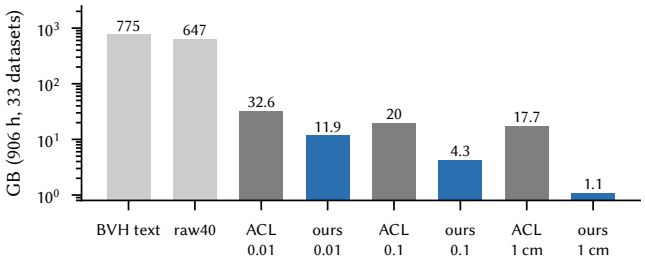


Fig. 8. Estimated storage of the full 906-hour corpus (16.17 G joint-samples), extrapolated per dataset from the test side; the full corpus was not encoded. Our bars use the mean-gate product codec after the fallback. Log scale.

Table 3. Ratio to ACL by data type and frame rate (test side, mean gate). The gain holds on every type, including game assets and a species absent from training, and grows with the frame rate.

Data type	p = 0.01	0.1	1 cm
native-skeleton mocap	0.358	0.202	0.055
retargeted mocap	0.489	0.322	0.168
converted SMPL	0.413	0.325	0.211
game assets	0.353	0.297	0.219
hands	0.373	0.239	0.129
dog (held-out species)	0.532	0.350	0.168
Frame rate (Hz)	20 / 24–30 / 60 / 90 / 120 at p = 0.1		
	0.323 / 0.341 / 0.316 / 0.200 / 0.178		

6.3 Contracts: Tolerances and Failures

The tolerances of the max gate are a convenience for the search, not a source of the gain. Removing them costs 0.7 to 1.0 % of the bytes on a 311-clip test subset, and a standalone contract that holds every joint-sample within p, which ACL itself meets on only 2 % of the clips, still gives 0.438× and 0.265× ACL's bytes at 0.01 and 0.1 cm: ACL's own overshoot of p is worth 4 to 6 % of our bytes. The fallback is a safety net, not a source of compression: at $p \leq 0.1$ cm the first encode passes on at least 99.6 % of the 4,472 clips under either contract, and before any fallback no stream is larger than ACL's. The final check fails on 16 and 3 clips at 0.01 and 0.1 cm under the max gate, nearly all hairline misses of the over-p fraction, and on 4 and 3 under the mean gate; a margin re-encode or ACL's stream resolves every one and changes no ratio by more than 0.001; the re-encodes of all failing clips cost about 7 core-hours in total. At 0.3 and 1 cm, 34 and 95 max-gate clips fail because the encoder's late step refinement cannot un-fold a folded track that already exceeds the contract, and 36 and 92 streams, nearly all of them these clips, are larger than ACL's before the fallback; after it, ACL's stream is stored for 13 and 29 clips, the counts and ratios of Tab. 2 (details and an un-folding fix in supplementary Sec. B). Under the mean gate no stream is larger than ACL's at any precision.

6.4 Generalization

Every dataset is smaller than ACL at every precision (Fig. 10, Tab. 3). The gain is largest on long, high-rate captures (large online mocap 0.236× at 0.01 cm) because smoother curves let a fixed budget buy longer key gaps, while ACL's rate per sample barely changes with the frame rate. A looser bound pays only where there are samples to omit: from 0.01 to 1 cm the ratio of 120 Hz BEAT falls 27-fold (0.399 to 0.015×) and that of 20 Hz HumanAct12 1.5-fold (0.676 to 0.444×); the frame-rate classes are partly confounded with datasets. Retargeted data gain less because ACL is already efficient on them. The dog, a species absent from training, reaches 0.532×, 0.350×, and 0.168×. The entropy model is the only trained part: on the dog it saves 5.7 to 6.9 % over the codec without a network, 91 to 95 % of what a model trained on dog data saves. A model trained on one 120 Hz dataset alone is 1.34 to 1.43× *larger* than no model on the dog, so the product model is trained on the whole corpus.

6.5 Ablation

Tab. 4 adds components one at a time and removes each from the final encoders. Prediction is the largest component at $p \leq 0.1$ cm: against the quantized integers coded with a general-purpose compressor (first row) it is worth 2.0 and 2.2×, and 2.3 to 3.0× against the same context coder without prediction. Keys are second and grow as p loosens: removing them costs 8 %, 39 %, and 69 % at 0.01,

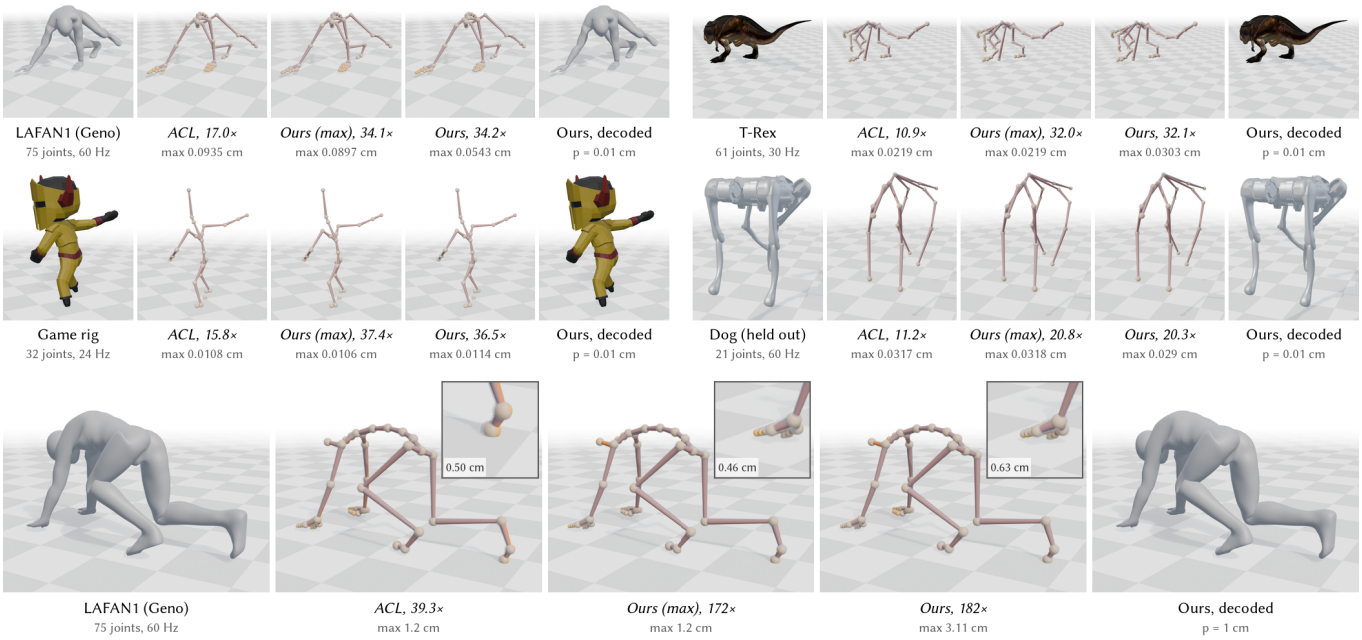


Fig. 9. Decoded motion on the held-out test side. Each half-row shows the ground-truth mesh, the skeletons decoded by ACL, by CurveCodec 2 under the max gate, and by CurveCodec 2 under the mean gate, each overlaid on the ground-truth skeleton (blue; the decoded skeleton is orange and thinner, so where they coincide the overlay reads pink), and the mesh driven by our decoded motion. Labels give the clip's compression ratio against ACL's raw size and its maximum shell error. Top two rows: $p = 0.01$ cm, where all three codecs are visually exact and CurveCodec 2 needs 2 to 3× fewer bytes than ACL. The dog capture (held out of training) has no mesh of its own; its mesh panels show a Go2 robot for orientation only and its skeleton panels are placed approximately. Bottom row: $p = 1$ cm on the same humanoid clip, with insets on the joint that moves the most; ACL reaches 39× and CurveCodec 2 172× (max gate, ACL's worst case within tolerance) or 182× (mean gate, same mean error), and the largest joint displacement in the shown frame is 0.46 to 0.63 cm for all three.

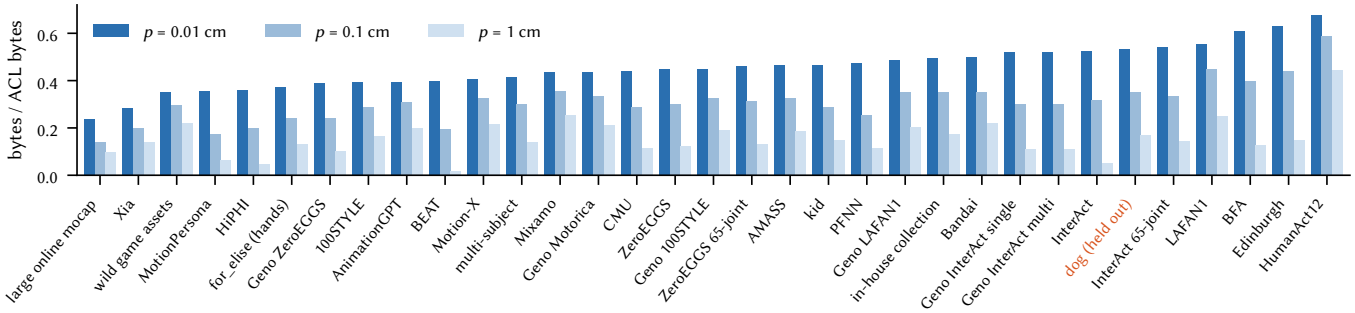


Fig. 10. Our bytes divided by ACL's on each dataset of the held-out test side, iso-mean criterion, at three precisions, sorted by the ratio at 0.01 cm. No dataset is larger than ACL. The gain is largest on long, high-rate captures (large online mocap, HiPHI, BEAT) and smallest on short, low-rate clips of small skeletons (HumanAct12, Edinburgh, BFA). The dog (orange) is a species absent from training.

0.1, and 1 cm under the max gate (2.2× at 1 cm without the five weakness clips) and 32 % to 2.2× under the mean gate, so at 1 cm keys and prediction are comparable (1.9×). Each configuration spends a different share of the error budget, so the table ranks contributions under the contract as used, not at strictly equal distortion. The final step search and container save 1 to 8 %; the entropy model saves 3 to 6 % under the max gate and 7 to 10 % under the mean gate, of which the transformer contributes 4 to 5 % over the MLP; thinning saves at most 1.6 %. Supplementary Sec. D gives the development ablation on the curated set and the capacity of the entropy model: deeper models save another 1 to 2.7 points at 1.8 to 27.5× the decode cost.

6.6 Runtime and Deployment

On one core, ACL encodes 1 M joint-samples in 5.0 s and decodes them in 12.5 ms. The max-gate codec encodes in 24 s (4.8×) and

Table 4. Ablation of the final codecs on held-out data (311-clip test subset, all 33 datasets). Top of each block: components added one at a time, bytes relative to ACL. Bottom: one component removed from the final encoder, bytes relative to the final encoder. A failing clip counts at ACL's bytes. At 1 cm the max-gate rows include five clips of the encoder weakness of Sec. 6.3; without them the final codec reaches 0.110. Prediction is measured on the every-sample codec.

	$p = 0.01$	0.1	0.3	1 cm
Max gate, bytes / ACL				
log-map quantizer, every sample	0.978	0.860	–	0.596
+ prediction and context rANS	0.480	0.394	0.406	0.310
+ fine step grid, lookahead, final container	0.458	0.366	0.372	0.307
+ RD keys	0.432	0.274	0.265	0.191
+ component key thinning	0.430	0.271	–	0.187
+ MLP entropy model (final)	**0.413**	**0.256**	**0.251**	**0.182**
Max gate, final without one component, bytes / final				
without prediction (bytes / every-sample codec)	2.34	2.95	2.77	2.99
without keys	1.076	1.394	1.46	1.69
without MLP entropy model	1.040	1.062	–	1.028
without thinning	1.003	1.002	–	1.012
Mean gate, bytes / ACL				
RD ladder, no model, no thinning	0.456	0.295	–	0.126
+ component key thinning	0.455	0.287	–	0.122
+ MLP entropy model	0.440	0.272	–	0.118
MLP replaced by transformer (final)	**0.418**	**0.260**	**0.169**	**0.113**
Mean gate, final without one component, bytes / final				
without keys	1.317	1.813	2.207	2.23
without entropy model	1.090	1.104	–	1.074
transformer replaced by MLP	1.052	1.048	–	1.039
without thinning	0.994	1.006	–	1.016

decodes in 0.48 s (38×); the product encodes in 57 s (12×) and decodes in 2.1 s (168×, about 8,700 frames per second of a 55-joint skeleton), 0.76 s on four threads and 0.57 s on one GPU (pooled over all precisions). Decode time scales with the number of coded residuals; at 0.01 cm the median clip of a 93-clip timing subset decodes in 0.10 s with the max-gate codec and 0.57 s with the product, the 95th percentile in 0.86 and 4.5 s. The model weights take 870 KB (transformer) and 182 KB (MLP). Re-encoding the decoded clip into ACL (supplementary Sec. E) costs one ACL compression, which dominates the load time, and a second lossy stage (1.4 to 1.8× ACL's mean error), so a pipeline that must deliver ACL's error should run that stage at a tighter precision or keep the dense output; ACL at $p/2$ costs 10 to 16 % more bytes, and we did not measure the composite error at that setting. Our equal-error comparisons are therefore claims about storage and transport, not about runtime.

6.7 Failure Cases

CurveCodec 2 gains least on short, low-rate clips of small skeletons (HumanAct12 0.676× at 0.01 cm, Edinburgh 0.628×, BFA 0.609×; Fig. 11), where the header and the model's cold start are amortized over few samples and few samples can be omitted. At loose precision, ACL's own misses on HiPHI widen both contracts, and the mean gate's guard, ten times ACL's clip mean, admits worst-case errors up to 24.7 cm at 1 cm on a handful of clips; we do not recommend the mean gate at $p \geq 0.3$ cm. We do not compare against CurveCodec: its published numbers come from a different split with floating-point accounting and are listed in the supplement for context only.

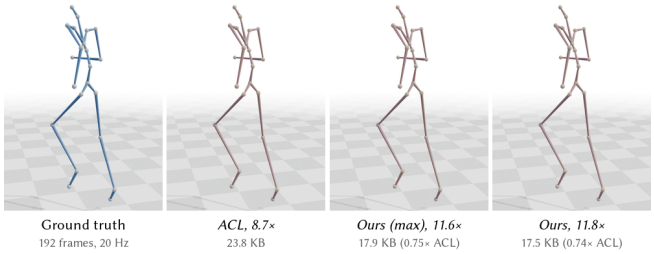


Fig. 11. The weakest case: a 192-sample, 20 Hz clip of a 27-joint skeleton (HumanAct12) at $p = 0.01$ cm. All codecs are visually exact, but CurveCodec 2 saves only a quarter of ACL's 23.8 KB (0.74× to 0.75×), against 0.37× on the whole test side: the per-clip header and the cold start of the entropy model are amortized over few samples, and at 20 Hz few samples can be omitted.

7 Limitations

No random access. CurveCodec 2 decodes whole clips: the coder, predictors, and model step are sequential in time, so a single pose cannot be read in constant time, which ACL does in about half a microsecond, and a full clip takes 38 to 168× ACL's decompression time on one core, still far faster than real time (Sec. 6.6). This is inherent to the design, not a matter of optimization: ACL is stateless and keeps the clip compressed in memory, decompressing only the poses a runtime's evaluation pass needs, whereas CurveCodec 2 is a storage and distribution format decoded at load time, and re-encoding into ACL adds a second lossy stage. Encoding is 4.8 to 12× slower than ACL and runs offline, in parallel over clips.

Contracts relative to ACL. Both contracts take their targets from ACL's result on the same clip, so the encoder needs ACL's statistics and inherits its misses at loose precision; the max gate holds ACL's worst case only within a 5 % tolerance, and errors are bounded at the sample times, not between them. A standalone contract in terms of p costs 4 to 6 % of the bytes on a test subset but was not run at corpus scale. At loose precision the max-gate encoder does not un-fold tracks (Sec. 6.3); the fix is not part of the benchmarked codec.

Not validated as a representation. We motivated the codec by the hope that what it must send is closer to the motion's decisions than to the body, but we test it only as a codec. Neither the stream nor the entropy model is evaluated on a generative or recognition task, and the findings that interpolation, cross-joint structure, and transforms did not pay hold for the estimators, data, and precisions we tried, each with one architecture and training run; they do not prove that no such model can do better. Whether the codec can serve a learned motion model is future work.

8 Conclusion

We asked where the redundancy of skeletal motion lies and which part of a codec a learned model should take over, and answered by measurement. Predicting each quantized curve from its own past is the largest saving at tight precision; omitting samples, chosen per joint in closed loop through the hierarchy, is the second, and its share grows until the two are comparable at 1 cm. On the gaps

such an encoder leaves there is little left to interpolate: none of the interpolators we tried, including a retrieval oracle over millions of training samples, beat linear interpolation, so the learned in-betweener at the core of CurveCodec had little left to learn. A network pays as the probability model of the residuals the codec must send. CurveCodec 2 puts these findings into one skeleton-agnostic, bit-exact codec that needs 0.37× ACL's bytes at ACL's default precision under ACL's own worst-case contract within tolerance, and 0.22× at 0.1 cm at its mean error, with one model that serves every rig we tried, including a species absent from training. The two parts that carry the gain, each curve's slowly varying residual regime and the hierarchy-aware choice of which samples to send, are what a learned motion model would have to capture to work across bodies; building such a model on them is the next step.

Acknowledgments

We thank Nicholas Frechette, the author of ACL, for detailed discussions of ACL's design goals and technical details. We thank Jun Xing, Tianshu Zhang and Zhixin Piao for the very early discussions on motion compression. This work was partially funded by the Research Grants Council of Hong Kong (Ref: 17216826) and by the Innovation and Technology Commission of the HKSAR Government under the ITSP-Platform grant (Ref: ITS/335/23FP, ITS/469/24FP).

References

Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. 2020. Skeleton-Aware Networks for Deep Motion Retargeting. *ACM Transactions on Graphics* 39, 4, Article 62 (2020), 14 pages. https://doi.org/10.1145/3386569.3392462

Marc Alexa and Wolfgang Müller. 2000. Representing Animations by Principal Components. *Computer Graphics Forum* 19, 3 (2000), 411–418. https://doi.org/10.1111/1467-8659.00433

Okan Arikan. 2006. Compression of Motion Capture Databases. *ACM Transactions on Graphics* 25, 3 (2006), 890–897. https://doi.org/10.1145/1141911.1141971

Johannes Ballé, Nick Johnston, and David Minnen. 2019. Integer Networks for Data Compression with Latent-Variable Models. In *International Conference on Learning Representations (ICLR)*.

Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. 2017. End-to-end Optimized Image Compression. In *International Conference on Learning Representations (ICLR)*.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. 2018. Variational Image Compression with a Scale Hyperprior. In *International Conference on Learning Representations (ICLR)*.

Thomas Barkowsky, Longin Jan Latecki, and Kai-Florian Richter. 2000. Schematizing Maps: Simplification of Geographic Shape by Discrete Curve Evolution. In *Spatial Cognition II: Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications (Lecture Notes in Computer Science)*. Springer, Berlin, Heidelberg, 41–53. https://doi.org/10.1007/3-540-45460-8_4

Philippe Beaudoin, Pierre Poulin, and Michiel van de Panne. 2007. Adapting Wavelet Compression to Human Motion Capture Clips. In *Proceedings of Graphics Interface 2007 (GI '07)*. Association for Computing Machinery, New York, NY, USA, 313–318. https://doi.org/10.1145/1268517.1268568

Fabrice Bellard. 2021. NNCP v2: Lossless Data Compression with Transformer. https://bellard.org/nncp/.

Edwin Catmull and Raphael Rom. 1974. A Class of Local Interpolating Splines. In *Computer Aided Geometric Design*, Robert E. Barnhill and Richard F. Riesenfeld (Eds.). Academic Press, New York, 317–326. https://doi.org/10.1016/B978-0-12-079050-0.50020-5

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser Nam Lim, and Abhinav Shrivastava. 2021. NeRV: Neural Representations for Videos. *Advances in Neural Information Processing Systems* 34 (2021), 21557–21568.

Ricky TQ Chen, Matthew Le, Matthew Muckley, Maximilian Nickel, and Karen Ullrich. 2022. Latent Discretization for Continuous-time Sequence Compression. *arXiv preprint arXiv:2212.13659* (2022).

CMU Graphics Lab. 2019. CMU Graphics Lab Motion Capture Database. http://mocap.cs.cmu.edu/. Accessed May 2019.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. 2024. Language Modeling Is Compression. In *International Conference on Learning Representations (ICLR)*.

Jarek Duda. 2013. Asymmetric Numeral Systems: Entropy Coding Combining Speed of Huffman Coding with Compression Rate of Arithmetic Coding. *arXiv preprint arXiv:1311.2540* (2013).

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. 2021. COIN: Compression with Implicit Neural Representations. In *Neural Compression: From Information Theory to Applications – Workshop @ ICLR 2021*. https://openreview.net/forum?id=yekxhcsVi4

Amirhossein Firouzmanesh, Irene Cheng, and Anup Basu. 2011. Perceptually Guided Fast Compression of 3-D Motion Capture Data. *IEEE Transactions on Multimedia* 13, 4 (2011), 829–834. https://doi.org/10.1109/TMM.2011.2129497

Nicholas Frechette. 2023. Animation Compression Library. https://github.com/nfrechette/acl. MIT License; used at development commit 3ee5685 (version 2.1.99), release 2.1.0 is compared in the supplement.

Saeed Ghorbani, Ylva Ferstl, Daniel Holden, Nikolaus F. Troje, and Marc-André Carbonneau. 2023. ZeroEGGS: Zero-shot Example-based Gesture Generation from Speech. *Computer Graphics Forum* 42, 1 (2023), 206–216. https://doi.org/10.1111/cgf.14734

Mohit Goyal, Kedar Tatwawadi, Shubham Chandak, and Idoia Ochoa. 2019. DeepZip: Lossless Data Compression Using Recurrent Neural Networks. In *2019 Data Compression Conference (DCC)*. IEEE, 575. https://doi.org/10.1109/DCC.2019.00087

F. Sebastian Grassia. 1998. Practical Parameterization of Rotations Using the Exponential Map. *Journal of Graphics Tools* 3, 3 (1998), 29–48. https://doi.org/10.1080/10867651.1998.10487493

Qin Gu, Jingliang Peng, and Zhigang Deng. 2009. Compression of Human Motion Capture Data Using Motion Pattern Indexing. *Computer Graphics Forum* 28, 1 (2009), 1–12. https://doi.org/10.1111/j.1467-8659.2008.01309.x

Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. 2024. MoMask: Generative Masked Modeling of 3D Human Motions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1900–1910. https://doi.org/10.1109/CVPR52733.2024.00186

Cihan Halit and Tolga Capin. 2011. Multiscale Motion Saliency for Keyframe Extraction from Motion Capture Sequences. *Computer Animation and Virtual Worlds* 22, 1 (2011), 3–14. https://doi.org/10.1002/cav.380

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust Motion In-betweening. *ACM Transactions on Graphics* 39, 4, Article 60 (2020), 12 pages. https://doi.org/10.1145/3386569.3392480

Simon Haykin. 2002. *Adaptive Filter Theory* (4 ed.). Prentice Hall, Upper Saddle River, NJ, USA.

Chengan He, Jun Saito, James Zachary, Holly Rushmeier, and Yi Zhou. 2022. NeMF: Neural Motion Fields for Kinematic Animation. *Advances in Neural Information Processing Systems* 35 (2022), 4244–4256.

Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. 2020. Learned Motion Matching. *ACM Transactions on Graphics* 39, 4, Article 53 (2020), 12 pages. https://doi.org/10.1145/3386569.3392440

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-Functioned Neural Networks for Character Control. *ACM Transactions on Graphics* 36, 4, Article 42 (2017), 13 pages. https://doi.org/10.1145/3072959.3073663

Daniel Holden, Jun Saito, and Taku Komura. 2016. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Transactions on Graphics* 35, 4, Article 138 (2016), 11 pages. https://doi.org/10.1145/2897824.2925975

Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning Motion Manifolds with Convolutional Autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs (SA '15)*. Association for Computing Machinery, New York, NY, USA, Article 18, 4 pages. https://doi.org/10.1145/2820903.2820918

Lawrence Ibarria and Jarek Rossignac. 2003. Dynapack: Space-Time Compression of the 3D Animations of Triangle Meshes with Fixed Connectivity. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '03)*. Eurographics Association, 126–135. https://doi.org/10.2312/SCA03/126-135

Jiahao Ji, Ji Ma, Runhan Zhang, Runyi Yu, Wenjia Wang, Weiheng Chi, Qianqian Peng, Weichao Yan, Yongfei Gu, Ye Tian, Ting Wu, Longwei Li, Chun Yuan, Ruoli Dai, and Lei Han. 2026. HiPHI: A Large-Scale Benchmark for High-Precision Human Motion and Object-Interaction. *arXiv preprint arXiv:2608.16222* (2026).

Zachi Karni and Craig Gotsman. 2000. Spectral Compression of Mesh Geometry. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press, New York, NY, USA, 279–286. https://doi.org/10.1145/344779.344924

Murtaza Ali Khan. 2016. An efficient algorithm for compression of motion capture signal using multidimensional quadratic Bézier curve break-and-fit method. *Multidimensional Systems and Signal Processing* 27, 1 (2016), 121–143. https://doi.org/10.1007/s11045-014-0293-4

Chao-Hua Lee and Joan Lasenby. 2008. An Efficient Wavelet-Based Framework for Articulated Human Motion Compression. In *Advances in Visual Computing (ISVC 2008) (Lecture Notes in Computer Science)*. Springer, Berlin, Heidelberg, 75–86. https://doi.org/10.1007/978-3-540-89639-5_8

Sunmin Lee, Taeho Kang, Jungnam Park, Jehee Lee, and Jungdam Won. 2023. SAME: Skeleton-Agnostic Motion Embedding for Character Animation. In *SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers '23)*. Association for Computing Machinery, New York, NY, USA, Article 45, 11 pages. https://doi.org/10.1145/3610548.3618206

Ik Soo Lim and Daniel Thalmann. 2001. Key-Posture Extraction out of Human Motion Data. In *Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, Vol. 2. IEEE, 1167–1169. https://doi.org/10.1109/IEMBS.2001.1020399

I-Chen Lin, Jen-Yu Peng, Chao-Chih Lin, and Ming-Han Tsai. 2011. Adaptive Motion Data Representation with Repeated Motion Analysis. *IEEE Transactions on Visualization and Computer Graphics* 17, 4 (2011), 527–538. https://doi.org/10.1109/TVCG.2010.87

Jing Lin, Ailing Zeng, Shunlin Lu, Yuanhao Cai, Ruimao Zhang, Haoqian Wang, and Lei Zhang. 2023. Motion-X: A Large-scale 3D Expressive Whole-body Human Motion Dataset. *Advances in Neural Information Processing Systems* 36 (2023), 25268–25280. Datasets and Benchmarks Track.

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. 2020. Character Controllers Using Motion VAEs. *ACM Transactions on Graphics* 39, 4, Article 40 (2020), 12 pages. https://doi.org/10.1145/3386569.3392422

Guodong Liu and Leonard McMillan. 2006. Segment-Based Human Motion Compression. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*. Eurographics Association, 127–135. https://doi.org/10.2312/SCA/SCA06/127-135

Haiyang Liu, Zihao Zhu, Naoya Iwamoto, Yichen Peng, Zhengqing Li, You Zhou, Elif Bozkurt, and Bo Zheng. 2022. BEAT: A Large-Scale Semantic and Emotional Multi-Modal Dataset for Conversational Gestures Synthesis. In *Computer Vision – ECCV 2022 (Lecture Notes in Computer Science)*. Springer Nature Switzerland, Cham, 612–630. https://doi.org/10.1007/978-3-031-20071-7_36

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 5442–5451. https://doi.org/10.1109/ICCV.2019.00554

Ian Mason, Sebastian Starke, and Taku Komura. 2022. Real-Time Style Modelling of Human Locomotion via Feature-Wise Transformations and Local Motion Phases. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 5, 1, Article 6 (2022), 18 pages. https://doi.org/10.1145/3522618

David Minnen, Johannes Ballé, and George D. Toderici. 2018. Joint Autoregressive and Hierarchical Priors for Learned Image Compression. *Advances in Neural Information Processing Systems* 31 (2018), 10794–10803.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 165–174. https://doi.org/10.1109/CVPR.2019.00025

Mathis Petrovich, Michael J. Black, and Gül Varol. 2021. Action-Conditioned 3D Human Motion Synthesis with Transformer VAE. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 10985–10995. https://doi.org/10.1109/ICCV48922.2021.01080

Tai Qin, Chunyang Fu, Ge Li, and Shan Liu. 2025. Multi-Descriptor Mesh Animation Compression. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5. https://doi.org/10.1109/ICASSP49660.2025.10889493

Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J. Guibas. 2021. HuMoR: 3D Human Motion Model for Robust Pose Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 11488–11499. https://doi.org/10.1109/ICCV48922.2021.01129

Jarek Rossignac. 1999. Edgebreaker: Connectivity Compression for Triangle Meshes. *IEEE Transactions on Visualization and Computer Graphics* 5, 1 (1999), 47–61. https://doi.org/10.1109/2945.764870

Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. 2017. PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications. In *International Conference on Learning Representations (ICLR)*.

Mingyi Shi, Huancheng Lin, Xuelin Chen, and Taku Komura. 2026. Neural Codec for Skeletal Animation Compression. In *SIGGRAPH Asia 2026 Conference Papers*. https://doi.org/10.1145/3829340.3842192

Mingyi Shi, Wei Liu, Jidong Mei, Wangpok Tse, Rui Chen, Xuelin Chen, and Taku Komura. 2025. MotionPersona: Real-Time Locomotion Control across Personas, Bodies, and Styles. *arXiv preprint arXiv:2506.00173* (2025).

Mingyi Shi, Sebastian Starke, Yuting Ye, Taku Komura, and Jungdam Won. 2023. PhaseMP: Robust 3D Pose Estimation via Phase-conditioned Human Motion Prior. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 14725–14737. https://doi.org/10.1109/ICCV51070.2023.01353

Ken Shoemake. 1985. Animating Rotation with Quaternion Curves. *ACM SIGGRAPH Computer Graphics* 19, 3 (1985), 245–254. https://doi.org/10.1145/325165.325242

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. 2020. Implicit Neural Representations with Periodic Activation Functions. *Advances in Neural Information Processing Systems* 33 (2020), 7462–7473.

Sebastian Starke, Ian Mason, and Taku Komura. 2022. DeepPhase: Periodic Autoencoders for Learning Motion Phase Manifolds. *ACM Transactions on Graphics* 41, 4, Article 136 (2022), 13 pages. https://doi.org/10.1145/3528223.3530178

Gary J. Sullivan and Thomas Wiegand. 1998. Rate-Distortion Optimization for Video Compression. *IEEE Signal Processing Magazine* 15, 6 (1998), 74–90. https://doi.org/10.1109/79.733497

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. *Advances in Neural Information Processing Systems* 33 (2020), 7537–7547.

Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. 2018. Neural Kinematic Networks for Unsupervised Motion Retargetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 8639–8648. https://doi.org/10.1109/CVPR.2018.00901

Marcelo J. Weinberger, Gadiel Seroussi, and Guillermo Sapiro. 2000. The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS. *IEEE Transactions on Image Processing* 9, 8 (2000), 1309–1324. https://doi.org/10.1109/83.855427

Ian H. Witten, Radford M. Neal, and John G. Cleary. 1987. Arithmetic Coding for Data Compression. *Commun. ACM* 30, 6 (1987), 520–540. https://doi.org/10.1145/214762.214771

Xiaolin Wu and Nasir Memon. 1997. Context-Based, Adaptive, Lossless Image Coding. *IEEE Transactions on Communications* 45, 4 (1997), 437–444. https://doi.org/10.1109/26.585919

Heyuan Yao, Zhenhua Song, Yuyang Zhou, Tenglong Ao, Baoquan Chen, and Libin Liu. 2024. MoConVQ: Unified Physics-Based Motion Control via Scalable Discrete Representations. *ACM Transactions on Graphics* 43, 4, Article 144 (2024), 21 pages. https://doi.org/10.1145/3658137

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Trans. Graph.* 37, 4, Article 145 (July 2018), 11 pages. https://doi.org/10.1145/3197517.3201366