BUNRAKU: GENERATING EDITABLE LIVE2D-STYLE RIGS FROM A SINGLE ILLUSTRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

An illustrated character becomes an editable animation asset when its painted layers, deformation meshes, and parameter-dependent motion are stored together. Bunraku generates an open Live2D-style rig from one illustration and a requested layer count: layered diffusion generates RGBA slots, surviving alpha masks define triangle meshes, and a shared-frame Transformer predicts named-control vertex offsets. The asset replays in WebGL rather than native Cubism. We assess the representation with artist rigs, generated layers, and same-architecture attention controls. A post hoc 50-character artist-vertex audit favors joint over layer-blocked attention (moving error 0.0493 versus 0.0532), but a separate prospective 12-character one-seed test does not confirm this effect (0.0643 versus 0.0640). On 12 newly generated rigs, joint and blocked motion have source-mean rest-subtracted rendering errors 1.359 and 1.323 versus 1.000 for static layers; joint improves none of the 12 sources over static. A draw-order training repair lowers joint error from 1.495 to 1.196 on six new source matches, yet joint and blocked motion still trail static layers (1.196 and 1.170 versus 1.000). Only one of the original 12 rigs retains the requested number of usable layers. On a frozen 34-asset high-layer subset, after correcting the checkpoint-specific output-order convention, fine-tuning improves artist-matched alpha IoU (0.256 versus 0.173), pair-weighted order (0.623 versus 0.528), and composite PSNR (26.36 versus 20.91 dB), but reduces the count of alpha-filter-eligible layers. The historical 120-image comparison overlaps training names, and missing earlier manifests prevent certifying Stage 1 source disjointness. Bunraku yields editable controls, while reliable generated-rig animation and rights-cleared replication remain open.

1 INTRODUCTION

Live2D Cubism Live2D Inc. (2024) represents an illustrated character using ordered RGBA layers, 2D deformation meshes, and named controls that move the meshes between artist-authored keyposes. The stored layers and deformation rule make a rig editable and repeatedly drivable, unlike a rendered clip. Authoring such a rig requires separating semantic parts *with hidden regions painted in*, placing a mesh on each part, and coordinating vertex offsets across parts. All three require information absent from the flat illustration. We therefore study the structured asset itself rather than only the appearance of its rendered frames. Our target is an open Live2D-style rig played in WebGL; native Cubism export is outside the current system.

Prior work provides complementary components. Layered image generators Lin et al. (2026); Liu et al. (2025); Yin et al. (2025) predict RGBA stacks; vector-animation systems Chen et al. (2026) predict editable paths and keyframes; image-to-video systems Xing et al. (2024); Wan et al. (2025) return rendered motion. The closer image2live2d system Wzhang3912 (2026) also builds rigs from semantically named layers and offers a See-through front end. Bunraku studies jointly predicted per-vertex offsets for named controls within this asset class and audits the delivered rig; a matched motion comparison to image2live2d remains open.

Cross-layer consistency is a key challenge: an iris may move plausibly in isolation yet slide off its eye white. **Bunraku** therefore represents every vertex of every layer in one sequence with shared canvas coordinates and cross-layer attention. The name recalls the Japanese puppet theatre in which several operators move one puppet in unison. This design lets a predicted offset use other layers

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

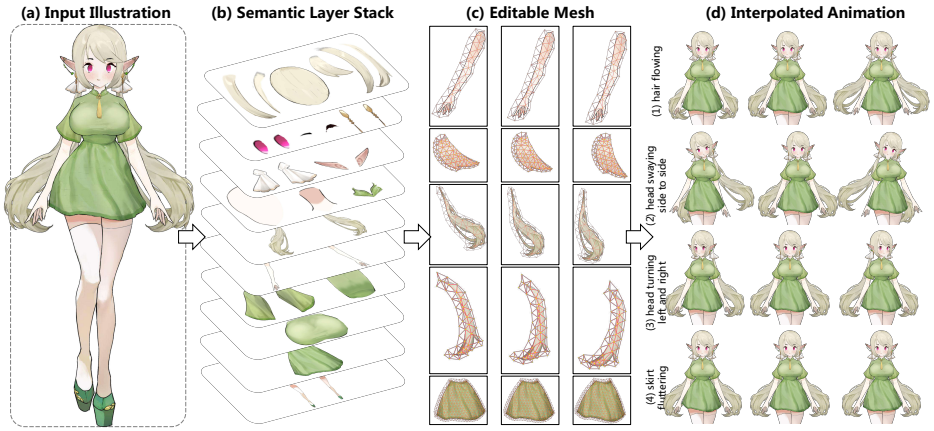

Figure 1: From a single illustration to a layer stack, editable meshes, and parameter-driven animation. This generated example shows asset structure at several control values; motion accuracy is evaluated separately.

as context. We compare the full joint implementation with an independent per-layer predictor, and separately examine a same-architecture attention control.

Bunraku has two stages (Fig. 2) and the core idea shapes both. **Stage 1** fine-tunes a layered diffusion model on artist-ordered RGBA stacks. Those targets include painted content under occluders, but the model still omits some parts. Because the layers are later reasoned about jointly, the *set* must be complete and correctly ordered, not merely clean: a missing sleeve is an inter-layer inconsistency no animation model can repair, and our small end-to-end test shows how these omissions propagate. **Stage 2** builds a content-conforming triangle mesh per layer from its alpha channel alone and, critically, places every layer's vertices in *one shared character frame* rather than a per-layer crop, so cross-layer geometric proximity exists in the input at all. A single 5.1 M-parameter Transformer then regresses per-vertex displacements over the concatenated sequence in one non-autoregressive pass, each factorised into a bounded direction and a log-magnitude because Live2D motion is heavy-tailed.

The Stage 2 evaluation uses *true generation*: one forward pass per keypose, without teacher forcing. A post hoc artist-vertex audit on 50 source-ID-disjoint characters favors joint attention over a masked same-architecture control (moving error 0.0493 versus 0.0532). A second, prospectively frozen 12-character one-seed test does not confirm the advantage (0.0643 versus 0.0640), though its wide interval includes the earlier effect. On these 12 sources, the full generated-rig test gives rest-subtracted render-motion errors 1.359 for joint, 1.323 for masked attention, and 1.000 for static layers; joint beats static on $0/12$ sources. Thus the asset and controls are editable, but the predicted animation does not outperform still layers under this motion-sensitive endpoint. A separate frozen 34-asset Stage 1 comparison, with a post hoc correction to the checkpoint-specific PNG order convention, improves matched alpha IoU, layer order, and composite PSNR versus base Qwen, but reduces the number of alpha-filter-eligible layers; historical layer rankings overlap training names. We also test a 24-control vocabulary, same-alpha re-texturing, and the complete pipeline on new illustrations.

Our contributions are summarized as follows:

1. **Task and corpus.** We formulate single-illustration generation of an editable, parameter-driven layered rig and extract $8{,}884$ Live2D models for layer and keypose supervision. Redistribution of the original artworks depends on third-party rights.
2. **Method.** Bunraku predicts ordered layer slots, meshes their nonempty alpha masks, and jointly predicts vertex offsets in a shared frame. The output is an open WebGL rig with reusable textures, geometry, and named controls.
3. **Evaluation protocol.** We separate a historical 120-image training-source diagnostic from a newly frozen layer comparison, a 50-character motion test, a prospective 12-character artist-rig replication, and generated-rig controls. We expose layer omissions, mesh-dependent vertex metrics, and the gap between artist-layer and generated-layer motion.

2 Related Work

2.1 Raster-pixel and controllable video generation

Image-to-video models can animate a reference picture, including cartoons Xing et al. (2024); Wan et al. (2025); Meng et al. (2025). Pose-driven character animation adds explicit motion controls Hu (2024); Tan et al. (2024). These approaches are valuable when the desired output is a clip, but the rendered frames do not expose a persistent per-layer mesh or a parameter-to-deformation rule. Layered image generators instead predict editable RGBA stacks Yin et al. (2025); Liu et al. (2025); Lin et al. (2026), while LayerAnimate Yang et al. (2025a) provides layer-level control over video. Such layers are a useful starting point for our Stage 1; they do not by themselves supply the meshes and keyposes needed to drive the same character interactively. We therefore evaluate decomposition as an input to rig generation, holding the motion model fixed when comparing our layers with those of See-through. This distinction matters even when a video model permits a user to revise its motion: the revision produces another sequence of images, whereas an exported rig can evaluate a control value and retain its editable texture and geometry.

2.2 Structured, editable asset generation

Autoregressive models can emit explicit mesh structure rather than pixels. MeshGPT Siddiqui et al. (2024) and EdgeRunner Tang et al. (2025) generate triangle meshes as token streams; EdgeRunner adapts EdgeBreaker connectivity coding Rossignac (1999). This motivated our initial tokenized mesh-deformation design, whose teacher-forced scores proved misleading and whose true-generation results are reported in Appendix G. For animation, LottieGPT Chen et al. (2026) and OmniLottie Yang et al. (2026) produce editable vector keyframes and easing parameters. AniClipart Wu et al. (2024) and LiveSVG Levy et al. (2026) animate vector art using video priors. These methods establish that animation can be generated as a reusable asset, but their paths and primitive transforms do not directly describe the per-vertex deformation of layered, hand-painted raster artwork. Bunraku instead takes an RGBA stack as input, predicts a continuous displacement field on each layer's mesh, and returns a named parameter interface that can be evaluated at new slider values. Its mesh vertices are already known at motion-model inference, so serializing their displacements offers no structural advantage; we test this choice against the autoregressive alternative rather than assuming it.

2.3 Live2D and mesh-based 2D deformation

An editable 3D character can be skeleton-rigged, including with automatic rigging Xu et al. (2020), but a 3D reconstruction and skin weights are a different target from preserving the original painted layers. Live2D Cubism Live2D Inc. (2024) uses ordered textured layers, deformation meshes, and named controls. Triangle-mesh warping itself is established through free-form deformation Sederberg & Parry (1986) and as-rigid-as-possible modeling Sorkine et al. (2007). Li et al. Li et al. (2021) preserve attachment, occlusion, boundary, and overlap relationships when a *given* body deformation is transferred to layered accessories; we instead predict layers and named-control motion from a flat image. Existing character systems address parts of this task: CartoonAlive He et al. (2025b) animates existing face models, Textoon He et al. (2025a) and Text2AC Sun et al. (2024) use template-based characters, while SPIRITUS Sun et al. (2025b) and Outline-and-Detail Sun et al. (2025a) study layered character generation. The open-source image2live2d pipeline Wzhang3912 (2026) authors a rig from semantically named layers and offers See-through as a flat-image front end; it also exports Cubism assets. PhysAnimator Xie et al. (2025) starts from one illustration and deforms an extracted mesh, but outputs a rendered clip. Bunraku constructs meshes deterministically from predicted RGBA layers and learns per-vertex named-control displacements jointly across the layer stack. The practical rig-authoring pipeline merits a matched motion comparison; our current quantitative controls do not include it.

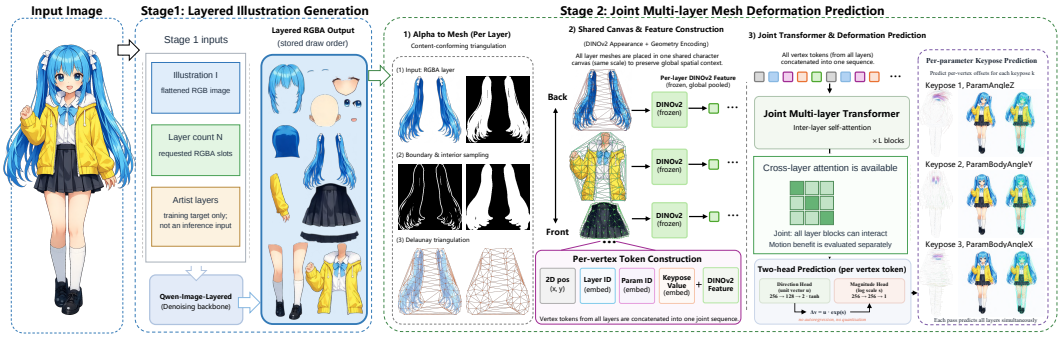

Figure 2: **Bunraku pipeline.** An illustration and requested layer count enter Stage 1; artist RGBA layers supervise training but are not inference inputs. Ordered output layers are meshed from their alpha channels. Shared-canvas vertex tokens enter Stage 2 for joint keypose prediction, and stored offsets drive the rig. The effect of cross-layer context on motion quality is evaluated separately.

3 BUNRAKU

3.1 TARGET ASSET AND LAYER DECOMPOSITION

Let $I \in \mathbb{R}^{H\times W\times 3}$ be an illustration and N a requested layer count. Stage 1 produces N ordered RGBA slots, but only a subset $\mathcal{U} \subseteq \{1,\dots,N\}$ survives the alpha and mesh filters. The actual asset is $\mathcal{A} = (\{L_i, M_i\}_{i\in\mathcal{U}}, \{D_i^{p,k}\}_{i\in\mathcal{U},p,k})$: $M_i = (V_i, F_i)$ is a triangle mesh and $D_i^{p,k}$ is a vertex offset at keypose k of named parameter p. RGB, alpha, meshes, and controls remain addressable. We learn layers and offsets, construct meshes deterministically, and replay blended offsets in stored layer order. This is an open WebGL rig bundle, not a native Cubism binary.

Stage 1 fine-tunes Qwen-Image-Layered (Yin et al., 2025) to predict ordered Live2D-style parts from (I, N). Training pairs contain flattened illustrations and artist-ordered RGBA layers, including painted hidden regions. The system has no image-only count predictor: PSD tests supply the artist count as an oracle; in-the-wild examples request 23 slots. The renderer preserves source order. Stage 2 filters slots with fewer than 30 pixels above alpha $16/255$ after resizing, then meshes the survivors; requesting N does not guarantee N usable parts. Training and split limits are in Appendix B. Stage 1 adapts an existing diffusion model.

3.2 MESHES FROM LAYER ALPHA

For a novel layer, we threshold its alpha at $4/255$, dilate the resulting silhouette by three pixels, sample every external boundary in proportion to contour length, add interior points, and triangulate with Delaunay (Barber et al., 1996). We retain all faces: faces over transparent texture do not render, whereas pruning one that covers a faint painted rim can expose a seam. The vertices share the original character canvas coordinates, normalized to $[-1, 1]^2$; separate layer crop coordinates would erase the geometric relationship between an iris and its eye white. On 151 automatically decomposed layers, this construction covers 0.9996 of pixels with alpha at least $8/255$ on average. At matched visible coverage it uses 83 vertices per layer versus 112 for a uniform quad grid, reducing the number of attention tokens. Appendix J gives the threshold and coverage controls. For artist-rig evaluation, we predict offsets on the artist's original vertices; rendering adds boundary vertices without changing the metric's vertex set.

3.3 JOINT PARAMETER-CONDITIONED DEFORMATION

One inference call specifies a parameter ID p and normalized keypose value q. If v_{ij} is vertex j of layer i, its input token combines a Fourier encoding γ of position, a learned parameter embedding e_p, a Fourier encoding of q, a learned layer-index embedding e_i, and a frozen DINOv2-small (Oquab et al., 2023) descriptor $\phi(L_i)$ of the layer crop (Figure 3):

$$h_{ij}^0 = W_v[\gamma_6(v_{ij}); e_p; \gamma_3(q); e_i] + W_a\phi(L_i). \tag{1}$$

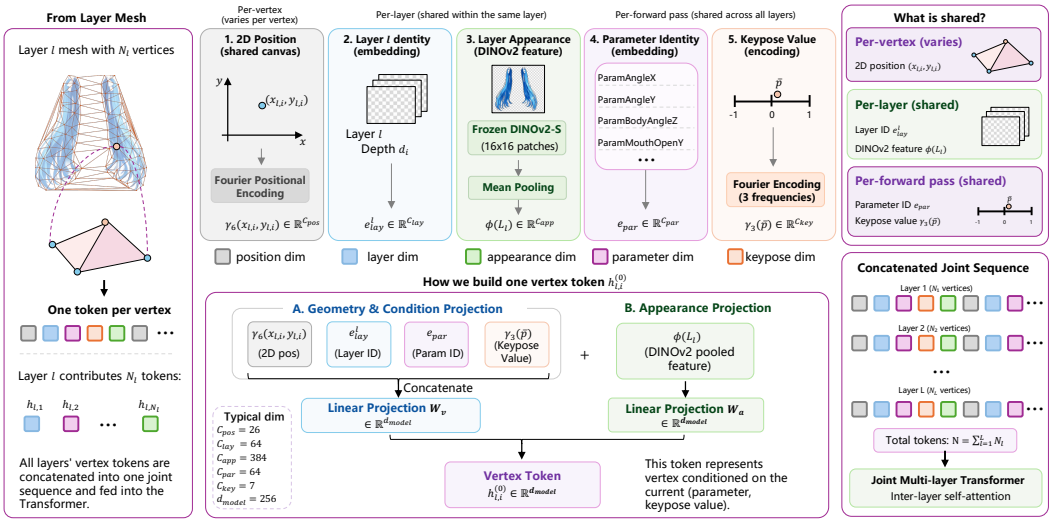

Figure 3: **Per-vertex token construction at inference.** One token is built for each mesh vertex. Position varies by vertex; layer identity and frozen DINOv2 appearance are shared within a layer; parameter identity and keypose value are shared across the forward pass. The geometry and condition features are projected together, while appearance is projected and added (Eq. 1). All vertex tokens enter one joint Transformer sequence.

The descriptor is mean-pooled across image patches and shared by vertices in a layer; the local canvas coordinate remains vertex-specific. The geometry-and-condition vector has width 161 before projection to model width 256. At inference we concatenate every layer's vertices into one sequence of $T = \sum_i |V_i|$ tokens and process it with six bidirectional Transformer blocks and four attention heads, with a padding mask but no layer mask. Training samples at most 28 vertices per layer and caps sequences at 1,200 tokens, creating an input-length shift on larger rigs (Appendix E). The trainable network has 5.1 million parameters. With index sets S_i for layers, the attention matrix contains within-layer blocks $A[S_i, S_i]$ and cross-layer blocks $A[S_i, S_j]$ for $i \neq j$. Our implemented independent predictor instead processes one layer at a time and uses DINO patch cross-attention and a pooled magnitude head; the joint predictor uses pooled layer appearance and a vertex magnitude head. Their comparison tests complete implementations, not an attention-mask-only intervention. Raw off-layer attention mass is not by itself evidence of coordination, since most matrix cells lie outside the within-layer blocks even under uniform attention.

Artist displacements have a wide dynamic range. For layer i at one keypose, define $a_i = \max(\epsilon, \max_j \|D^{p,k}_{ij}\|_\infty)$ and $u_{ij} = D^{p,k}_{ij}/a_i$. Two heads predict $\hat{u}_{ij} = \tanh f_u(h_{ij})$ and $s_{ij} = f_s(h_{ij})$; the generated offset is $\hat{D}^{p,k}_{ij} = \hat{u}_{ij} \exp(s_{ij})$. Both heads use masked L_1 supervision:

$$\mathcal{L} = \mathbb{E}_{i,j} \|\hat{u}_{ij} - u_{ij}\|_1 + \mathbb{E}_{i,j} |s_{ij} - \log a_i|. \tag{2}$$

The normalization prevents the few largest turns from dominating the direction objective. At inference the model runs once for each non-rest (p, k); no generated offset is fed back. Vertex positions at a slider value are obtained by linear interpolation between bracketing keyposes. Multiple controls add their offsets at runtime. This supports simultaneous control input, although the model does not learn interactions specific to a parameter combination. The released checkpoint uses 30 epochs of AdamW at learning rate 2×10^{-4}; the controlled attention pairs use a separate 20-epoch schedule and are compared only with each other. Appendix V.4 provides the record construction and compute details.

4 Evaluation

4.1 Data, splits, and measures

The extracted corpus has 8,884 usable artist Live2D models. The original summary reports about 50,000 augmented layer-decomposition examples; the final Stage 1 resume manifest has 3,955

Table 1: Frozen 34-asset high-layer Stage 1 comparison, with oracle artist layer count and paired inputs. Fine-tuned output order was corrected after inspecting the initial scores; both versions are audited. Bold is the favorable point estimate within each row. Eligible-layer count is a post hoc alpha-filter audit.

Metric	Base Qwen	Bunraku
Matched α-IoU ↑	.173	**.256**
Pair-weighted order ↑	.528	**.623**
Composite PSNR ↑	20.91	**26.36**
Composite SSIM ↑	.838	**.893**
Alpha-filter-eligible ↑	**21.09**	17.76

entries (3,930 normalized source names), while earlier manifests are unavailable. The current motion pack has 1,443 characters and 357,377 (character, p, k, layer) records. Our asset-level evaluation covers layers, meshes, controls, and rendered motion. The historical 120-image layer comparison is *diagnostic*: all recovered source names match the final Stage 1 training manifest; the original sample list and exact per-method pairing are unavailable (Appendix B). The earlier 50-character motion scores used checkpoint-selected validation IDs, and their 46-character subset has a different mesh/scoring pack. The new attention intervention instead freezes 50 other characters before training and excludes them from both gradient updates and checkpoint selection. Its direction cosine averages artist vertices moving more than 0.005 of canvas extent within records, then records within characters. The two attention arms use identical records and three paired seeds; all predictions are true generation without teacher forcing (Appendix E).

4.2 Layer fidelity and error propagation

Table 1 replaces the historical layer ranking in the main paper with a paired comparison on 34 frozen source composites selected by an input-only threshold of at least 14 artist layers (27 Booth and seven Nizima assets). Both Stage 1 arms receive the artist layer count as an oracle and use the same 25-step sampler. A post hoc orientation audit found that base Qwen emits PNGs back-to-front, while the fine-tuned checkpoint emits front-to-back; we reverse the latter before stacking and preserve the original scores (Appendix C). Bunraku raises artist-matched alpha IoU from 0.173 to 0.256 (paired difference $+0.083$, 95% source-bootstrap interval $[+0.059, +0.105]$; better on $28/34$), pair-weighted order from 0.528 to 0.623 (difference $+0.096$, interval $[+0.052, +0.137]$), and composite PSNR from 20.91 to 26.36 dB (difference $+5.45$ dB, interval $[+4.27, +6.67]$). Yet only $17.76/22.29$ requested Bunraku slots per asset are alpha-filter-eligible versus $21.09/22.29$ for base Qwen, with exact requested counts on $3/34$ versus $19/34$ assets. The full-slot Hungarian alpha IoUs coincide with the missing-layer-penalized values because both generators emit every oracle-count PNG slot; near-empty slots can still fail the rig filter. Filtering before the same scorer (post hoc) preserves the penalized alpha IoU gain (0.260 versus 0.173), but yields no clear order advantage (0.520 versus 0.500) and exposes a higher unmatched-layer rate (0.210 versus 0.060). Eligibility does not certify semantic correctness or successful meshing. A separate post hoc audit of artist-hidden pixels finds no clear completion gain: alpha coverage is 0.597 versus 0.653, and premultiplied RGB error is 0.265 versus 0.268 (both paired intervals cross zero; Appendix C).

Excluding three canonical source aliases found in retained data roots leaves 31 inputs: Bunraku still has higher alpha IoU (0.258 versus 0.170), order consistency (0.627 versus 0.536), and PSNR (26.22 versus 20.86 dB). A secondary completion of the originally frozen 51-input pool, after inspecting the 34-asset results, has the same image-quality directions (alpha IoU 0.257 versus 0.170; PSNR 27.84 versus 21.03 dB) but only 14.55 versus 17.51 of 18.63 requested layers pass the deployment alpha filter (exact counts 3/51 versus 27/51). The earlier Stage 1 training manifests are missing, so neither subset is certified source-disjoint; the 34-asset primary analysis is a high-layer stress test, not a general-population estimate. The archived 120-image comparison has complete source-name overlap with the final Stage 1 manifest and is retained only as a training-source diagnostic in Appendix B. Figure 1 shows the stack and motion; raw examples and failures are in Appendix F.

We animated 13 in-the-wild illustrations from Bunraku and See-through (Lin et al., 2026) layers with the same frozen motion model and mesh-construction rule. Figure 4 shows missing parts propagating

Table 2: Prospective generated-rig test. Motion residual compares pose changes after subtracting each stack's own rest render; static is exactly 1.00. All 12 matched sources are primary, and eight input-compatible sources form a predeclared sensitivity. Joint, blocked, and static share Bunraku layers and meshes. The quarter-amplitude arm was calibrated on old validation rigs and is secondary. Bold marks the lowest motion-error point estimate per row.

Cohort	Joint ↓	Blocked ↓	Static ↓	Joint ×.25 ↓
All 12, 77 poses	1.359	1.323	**1.000**	1.216
Comparable 8, 55 poses	1.471	1.414	**1.000**	1.300

through the predictor. See-through was not designed for rigging; this tests its layers as inputs to our pipeline. On three rigged characters, Bunraku layers improve animated PSNR from 21.1 to 23.4 dB, but their rest render improves by 4.6 dB (Appendix S). On eight checkpoint-selection validation rigs, released-model rest-subtracted error is 1.10 versus 1.00 for still layers. Pixel gains cannot establish motion accuracy.

The more decisive test freezes 12 source-name-matched PSD/artist-rig pairs before training a new Stage 2 attention pair. It scores 77 non-rest keyposes using an artist-motion mask (pixel change above $8/255$, dilated nine pixels) and subtracts each stack's own rest render. On identical generated layers and meshes, joint attention scores 1.359, blocked attention 1.323, and static 1.000 (Table 2); joint minus static is $+0.359$ (95% paired source-bootstrap interval $[+0.182, +0.567]$), with *none* of 12 sources improving. Joint minus blocked is $+0.036$ (interval $[-0.015, +0.094]$), lower on $6/12$. The predeclared eight-source input-compatible subset remains unfavorable (joint 1.471, blocked 1.414, static 1.000; joint below static on $0/8$). Scaling joint offsets by 0.25, calibrated on old validation rigs before these scores, reaches 1.216 but still loses to static on all 12. The fine-tuned-stack primary motion scores already used the correct output order. After correcting the secondary base-Qwen stack order post hoc, its rest PSNR (14.87 dB) and motion residual (1.350) are similar to fine-tuned layers (14.76 dB and 1.359). Only 18.25 of 22.92 requested fine-tuned slots become usable mesh layers versus 21.25 for base Qwen; just $1/12$ fine-tuned rigs retains all requested layers (Appendix E). Absolute motion-region PSNR can favor joint (10.58 versus 10.43 dB static) because it still mixes rest appearance with pose change; the residual isolates change. A post hoc artist-layer render check scores 1.033 versus 1.000 static, with a wide interval. A second post hoc check finds $28.3/255$ RMS joint motion outside the artist-motion mask, versus $0.3/255$ artist change and zero for static; within the mask, the joint RGB motion-change cosine is only 0.122 despite a mean change-norm ratio of 0.991 (Appendix E). Neither diagnostic isolates a causal error source. Two later generated-rig seeds also lose to static on every source; their three-seed source mean is 1.349 joint versus 1.000 static. A revised draw-order training convention lowers joint residual from a matched old-order 1.495 to 1.196 on six new sources, but still loses to static (1.000; one win per learned arm); blocked changes from 1.267 to 1.170. Stage 1 provenance remains incomplete. Figure 5 visualizes one post hoc selected failure.

Appendix M shows generated rigs driven along several parameter axes on illustrations without artist rigs. Each row is one input, and columns are values requested from the same stored asset rather than independent images synthesized for each pose. These examples reveal failures a vertex average can miss, such as a part separating from its neighbor, but are qualitative because no artist motion target exists for these illustrations.

4.3 Does cross-layer prediction matter?

The controlled test masks cross-layer self-attention in the same 5.08M-parameter Transformer; the three paired seeds share training records and checkpoint selection. On 50 source-ID-disjoint artist rigs, a post hoc moving-vertex audit favors joint over blocked attention (0.04929 versus 0.05316, paired difference -0.00386, 95% interval $[-0.00723, -0.00059]$), while zero motion scores 0.07319. This uses artist meshes, not generated rigs; the initial direction-cosine interval crosses zero. Protocols and source-level results are in Appendix E.

We then froze a second 12-character source-ID-disjoint cohort, excluded it from training and checkpoint selection, and trained a new one-seed attention pair before evaluating it. Its prespecified moving-vertex error is 0.06426 for joint and 0.06396 for blocked attention; zero motion scores

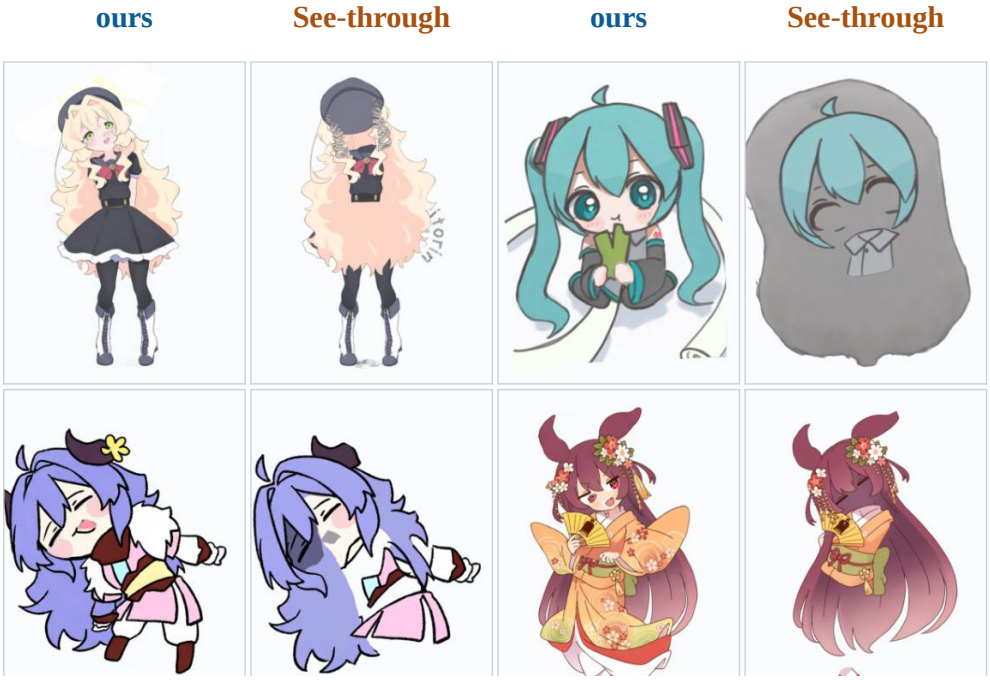

Figure 4: **Qualitative comparison with a frozen motion model.** Four paired illustrations show Bunraku layers (left) and See-through layers (right), all at `ParamAngleZ`$= -30$ with the same Stage 2 checkpoint. The top row contains `w3_goth` and `wild2_chibi`; the bottom row contains `webC_chibi2` and `wild2_kimono`. Incomplete stacks can lose or detach parts after deformation; further examples are in Appendix S.

0.08266. Joint minus blocked is $+0.00029$ (95% source-bootstrap interval $[-0.00865, +0.00838]$), lower on $6/12$ sources. The predeclared eight-source input-compatible subset also has no observed joint advantage (0.06602 versus 0.06420). Direction cosine is 0.6696 joint and 0.6787 blocked. This test does not confirm the 50-character post hoc effect, but its wide interval contains that earlier effect and cannot establish equivalence or a reversal. The cohort and training split differ. Two more seeds, trained after seeing seed 1, have joint-minus-blocked means $+0.00335$ and $+0.00371$; their three-seed source-mean difference is $+0.00245$ (interval $[-0.00422, +0.00815]$), a post hoc stability check rather than new confirmation. Even the original joint-versus-zero interval crosses zero. Stage 2 source-ID exclusion is audited, while Stage 1's earlier training manifests remain incomplete (Appendix E).

The original nearby cross-layer relative-offset proxy favors joint versus blocked but favors zero motion over both on the full test; 69.61% of active pairs have artist relative motion below 0.005. On eight checkpoint-selection validation rigs, seed-1 joint and blocked generated-rig motion residuals are 1.168 and 1.106 versus 1.000 static. These diagnostics do not establish useful attachment or generated-rig motion (Appendix E).

4.4 Editability, cost, and limitations

An instruction-guided editor (Qwen Team, 2025) can repaint a layer while preserving its alpha and dimensions. Across 16 such edits, mesh topology and predicted offsets are byte-identical; changing a silhouette voids this guarantee (Appendix W.6). The asset has static draw order, no physics or visemes, and is a WebGL bundle rather than native Cubism. Missing eyelids can move a whole head during blinks. A matched motion comparison with image2live2d (Wzhang3912, 2026) remains unavailable because its layer-directory mode requires semantic-role filenames absent from our stacks. Stage 2 construction takes 0.71–2.84 s for 10–63 layers, excluding Stage 1; replay reaches 30 fps (Appendices O and X).

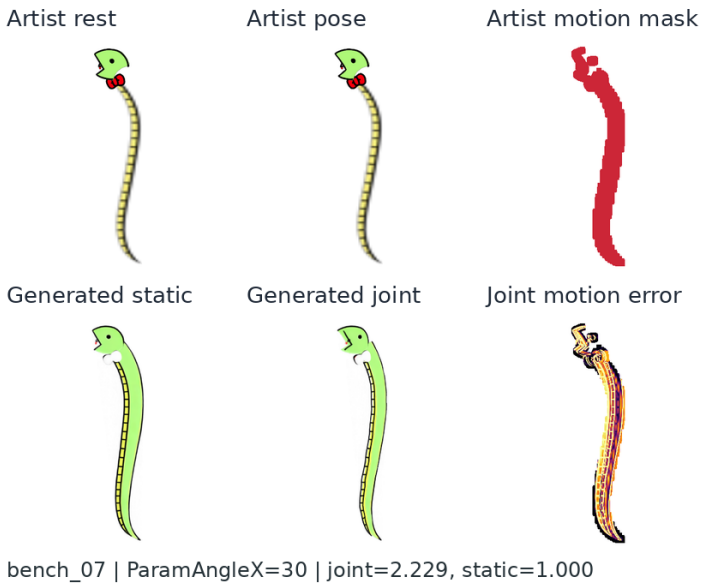

Figure 5: **Generated-rig failure on a new test source (post hoc visual selection).** For `bench_07` at `ParamAngleX`$=30$, artist rest and pose define the red motion mask. Generated static and joint frames share the same layer stack; the error map shows joint pose-minus-rest error within the artist mask (RGB root-mean-square difference, clipped at $100/255$). The source-mean motion-residual ratio is 2.229 joint versus 1.000 static. Candidate and artist rigs are separately fitted to their rest boxes; all 12 source-level scores, not this selected image, determine Table 2.

Table 3: Artist-vertex attention controls on source-ID-disjoint Stage 2 tests. The 50-character moving-error audit is post hoc and averages three seeds within each character. The separate 12-character one-seed test prespecified this endpoint. Both use vertices with artist motion > 0.005; means weight sources equally. Bold marks the best point estimate within a cohort, not a statistically established winner.

Artist-rig test	Attention	Cosine ↑	Moving L_2 error ↓
50 characters, 3 seeds	Layer-blocked	.7273	.05316
	Joint	**.7373**	**.04929**
	Zero motion	–	.07319
12 characters, 1 seed	Layer-blocked	**.6787**	**.06396**
	Joint	.6696	.06426
	Zero motion	–	.08266

5 Conclusion and Limitations

Bunraku creates editable WebGL rigs from one illustration. Draw-order training reduces generated-rig motion error on six new sources, but those rigs and 12 earlier prospective rigs still have worse mean motion residual than static layers; an artist-vertex test does not confirm an earlier attention gain. On 34 high-layer inputs, fine-tuning improves alpha IoU, order, and composite PSNR after correcting output order, but reduces eligible layer count. Reliable animation remains open.

The next test should freeze source-disjoint illustrations before either stage is trained, retain complete training manifests, and score delivered rigs against both artist motion and their own static stacks. Rig-usable part coverage, rest fidelity, motion in the artist-change region, and unwanted motion outside it should be reported per source. A draw-order ablation should hold the selected vertices fixed, since sorting can also change which tokens survive the sequence cap. These controls would separate image decomposition, token indexing, and animated-motion gains.

Reproducibility Statement

The evaluation distinguishes the historical 120-example layer diagnostic, the primary 34-of-51 high-layer comparison and secondary completion of the frozen 51, the 50-character three-seed attention test, the prospective 12-character one-seed replication, the older 46/50-character validation results, and 12- and six-source generated-rig controls. The six-source training repair followed inspection of the 12-source failure; its matched old-order training control was added before any six-source motion score. Archived names for the 120-case pool all match the final Stage 1 resume manifest by source name; exact historical per-method pairing and image identity remain unrecovered. Both new Stage 2 test cohorts were excluded from gradient training and checkpoint selection; a pre-score appearance screen flagged one possible training-source alias in the 50-character cohort, so a 49-character sensitivity is reported. Appendix D describes data construction; Sections V.3–V.4 and Appendix E give training and evaluation details. An anonymous visual supplement shows every source in the 12- and six-source generated-rig cohorts and supplies anonymized per-source score tables. Training code, weights, raw artwork, and split manifests are not part of the submitted supplement, so the experiments cannot be reproduced from the submission alone. The original artwork corpus cannot currently be redistributed in full.

Ethics Statement

The research corpus contains third-party illustrated character assets with heterogeneous license terms. Experimental access to those assets does not itself confer redistribution rights, so the full artwork corpus may not be redistributable; this limits exact reproduction. The animation system can also be used to create derivative character content, for which users must respect source rights and consent.

AI Use Statement

Generative AI tools were used solely for grammar correction and minor language polishing of text written by the authors. No generative AI tools were used for any research tasks requiring disclosure under the ICLR 2027 AI Policy. All AI-assisted edits were reviewed by the authors, who take full responsibility for the final content of this work.

References

C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483, 1996.

Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches*, 10(1):1, 2007.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

Junhao Chen, Kejun Gao, Yuehan Cui, Mingze Sun, Mingjin Chen, Shaohui Wang, Xiaoxiao Long, Fei Ma, Qi Tian, Hao Zhao, and Ruqi Huang. Lottiegpt: Tokenizing vector animation for autoregressive generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 31639–31651, June 2026.

Chao He, Jianqiang Ren, Yuan Dong, Jianjing Xiang, Xiejie Shen, Weihao Yuan, and Liefeng Bo. Textoon: Generating vivid 2d cartoon characters from text descriptions. *arXiv preprint arXiv:2501.10020*, 2025a.

Chao He, Jianqiang Ren, Jianjing Xiang, and Xiejie Shen. Cartoonalive: Towards expressive live2d modeling from single portraits. *arXiv preprint arXiv:2507.17327*, 2025b.

Li Hu. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8153–8163, 2024.

Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing diffusion-based image generators for monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9492–9502, 2024.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Matan Levy, Ran Margolin, Bar Cavia, Dvir Samuel, Yael Pritch, Shmuel Peleg, Alex Rav Acha, Ariel Shamir, and Dani Lischinski. Livesvg: Zero-shot svg animation via video generation, 2026. URL https://arxiv.org/abs/2605.30174.

Jingyi Li, Wilmot Li, Sean Follmer, and Maneesh Agrawala. Automated accessory rigs for layered 2d character illustrations. In *Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology*, 2021. doi: 10.1145/3472749.3474809.

Jian Lin, Chengze Li, Haoyun Qin, Kwun Wang Chan, Yanghua Jin, Hanyuan Liu, Stephen Chun Wang Choy, and Xueting Liu. See-through: Single-image layer decomposition for anime characters. *arXiv preprint arXiv:2602.03749*, 2026.

Cheng Liu, Yiren Song, Haofan Wang, and Mike Zheng Shou. Omnipsd: Layered psd generation with diffusion transformer. *arXiv preprint arXiv:2512.09247*, 2025.

Live2D Inc. Live2d Cubism editor: Mesh generation tool documentation, 2024. https://docs.live2d.com/cubism-editor-manual/mesh-generation/.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

Yihao Meng, Hao Ouyang, Hanlin Wang, Qiuyu Wang, Wen Wang, Ka Leong Cheng, Zhiheng Liu, Yujun Shen, and Huamin Qu. Anidoc: Animation creation made easier. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18187–18197, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

Qwen Team. Qwen-image-edit-2511: Instruction-guided image editing. Model release, https://huggingface.co/Qwen/Qwen-Image-Edit-2511, 2025. Instruction-guided image editing model built on the Qwen-Image backbone.

Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics*, 5(1):47–61, 1999.

Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pp. 151–160, 1986.

Jonathan Richard Shewchuk. Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Workshop on applied computational geometry*, pp. 203–222. Springer, 1996.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19615–19625, 2024.

Olga Sorkine, Marc Alexa, et al. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pp. 109–116, 2007.

Qirui Sun, Qiaoyang Luo, Yunyi Ni, and Haipeng Mi. Text2ac: a framework for game-ready 2d agent character (ac) generation from natural language. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pp. 1–7, 2024.

Qirui Sun, Yunyi Ni, Haixin Qiao, Jingjing Zhang, Fan Yang, Teli Yuan, Zhihao Yao, and Haipeng Mi. Outline and detail: A semantic-driven framework for layered 2d character generation. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–14, 2025a.

Qirui Sun, Yunyi Ni, Teli Yuan, Jingjing Zhang, Fan Yang, Zhihao Yao, and Haipeng Mi. Spiritus: An ai-assisted tool for creating 2d characters and animations. *arXiv preprint arXiv:2503.09127*, 2025b.

Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

Shuai Tan, Biao Gong, Xiang Wang, Shiwei Zhang, Dandan Zheng, Ruobing Zheng, Kecheng Zheng, Jingdong Chen, and Ming Yang. Animate-x: Universal character image animation with enhanced motion representation. *arXiv preprint arXiv:2410.10306*, 2024.

Jiaxiang Tang, Max Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. In *International Conference on Learning Representations*, volume 2025, pp. 35913–35934, 2025.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

Ronghuan Wu, Wanchao Su, Kede Ma, and Jing Liao. Aniclipart: Clipart animation with text-to-video priors. *International Journal of Computer Vision*, 2024. doi: 10.1007/s11263-024-02306-1.

Wzhang3912. image2live2d: Convert separated character-art layers into a reusable, riggable 2d puppet. GitHub repository, 2026. URL `https://github.com/Wzhang3912/image2live2d`. Commit 714e7cc; accessed September 2026.

Tianyi Xie, Yiwei Zhao, Ying Jiang, and Chenfanfu Jiang. Physanimator: Physics-guided generative cartoon animation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10793–10804, 2025.

Jinbo Xing, Hanyuan Liu, Menghan Xia, Yong Zhang, Xintao Wang, Ying Shan, and Tien-Tsin Wong. Tooncrafter: Generative cartoon interpolation. *ACM Transactions on Graphics (TOG)*, 43(6):1–11, 2024.

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020.

Yiying Yang, Wei Cheng, Sijin Chen, Honghao Fu, Xianfang Zeng, Yujun Cai, Gang Yu, and Xingjun Ma. Omnilottie: Generating vector animations via parameterized lottie tokens. *arXiv preprint arXiv:2603.02138*, 2026.

Yuxue Yang, Lue Fan, Zuzeng Lin, Feng Wang, and Zhaoxiang Zhang. Layeranimate: Layer-level control for animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10865–10874, 2025a.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations*, volume 2025, pp. 83048–83077, 2025b.

Shengming Yin, Zekai Zhang, Zecheng Tang, Kaiyuan Gao, Xiao Xu, Kun Yan, Jiahao Li, Yilei Chen, Yuxiang Chen, Heung-Yeung Shum, et al. Qwen-image-layered: Towards inherent editability via layer decomposition. *arXiv preprint arXiv:2512.15603*, 2025.

Table 4: Historical 120-image layer diagnostic with source-name overlap against the final Stage 1 training manifest. Composite metrics score the rest render; layer metrics match artist layers with missing-layer penalties. Bold indicates the best value *within this pool*.

	Depth seg.	SAM seg.	Qwen	See-through	Ours
Composite PSNR ↑	**28.24**	25.58	16.58	19.88	24.28
Composite SSIM ↑	**.977**	.953	.767	.890	.813
Matched LPIPS ↓	.210	.066	.143	.064	**.055**
Penalized α-IoU ↑	.081	.137	.175	.265	**.298**
Pair-weighted order ↑	.391	.302	.431	.266	**.674**

A Additional Details and Results

The appendix begins with the current submission's split audits, paired comparisons, and evaluation protocols, then gives extended method details and earlier experiments. Historical validation-selected Stage 2 results and autoregressive or video protocols must not be compared directly with the current motion and rendering results. The archived 120-case Stage 1 baseline pool overlaps the final training manifest by source name and is diagnostic only. Four anonymized per-character and per-asset CSV score tables accompany the visual supplement; no code or model weights are submitted. Other named audit files and scripts in the appendix are retained internally and are not part of the submission.

B Stage 1 Architecture

Stage 1 initialises from Qwen-Image-Layered Yin et al. (2025). The recorded full-parameter training invocation passes target RGBA images (`image`), the flattened input (`layer_input_image`), and `layer_num` to the standard flow-matching supervised fine-tuning routine, at learning rate 10^{-5} with gradient checkpointing. The deployed checkpoint is `step-6000.safetensors`; the run resumed from earlier checkpoints, and its filename is not a verified count of optimizer updates across all runs. We found no implemented explicit occlusion-weighted loss, class-token hierarchy, or per-layer caption dropout in that invocation. The $8{,}884$ models are the extracted corpus, not a verified count of distinct Stage 1 training characters. The final resume metadata file is retained on an internal data mount and contains $3{,}955$ entries ($3{,}930$ normalized source names) in the 10–35-layer range. Earlier resume manifests and the original `random_120.json` are unavailable. Two archived baseline output directories independently contain the same 120 names from May 9, before 350 new cases were appended; a retained launch script identifies the older set as the `random_120` pool. All 120 normalized names occur in the final Stage 1 resume manifest. This is source-name overlap, not a pixel-identity check, and exact per-method pairing behind Table 4 cannot be reconstructed. The layer table is consequently a training-source diagnostic, not evidence of unseen-source generalization. A separate eight-case rigged diagnostic has four name matches, and a separate step-500 preliminary 20-case evaluation has 15; these are not substitutes for the 120-case pool. The names and provenance are retained internally in `stage1_recovered120_audit.json`, `stage1_split_audit.json`, and `stage1_legacy20_audit.json`.

C Frozen Stage 1 Comparison Protocol

Before running the new Stage 1 models, we froze every asset in the retained `psd_layers` root with a composite, metadata, and 10–35 PNG layers whose URL-decoded, case-folded whole source name was absent from both retained Stage 1 image roots and ordered manifests. This gave 51 mixed character and accessory assets. A subsequent input-only audit exposed a weakness in that rule: stripping the one-letter Nizima variant prefix reveals three matching source names in a retained image root, one of which occurs in the final ordered training manifests. An exact decoded-pixel comparison against 3,464 retained training composites found no matches but cannot detect resized or recolored duplicates. We retained the full 51-asset manifest, but the measured comparison uses its 34 assets with at least 14 artist layers (27 Booth, seven Nizima), a threshold set from input metadata and observed runtime before any paired model scores were inspected. Fourteen base-model outputs had been generated; no fine-tuned output or paired score had been examined. At selection time the 51-asset two-

arm run was projected to exceed the available compute window; the lower-layer 17 were completed only after the high-layer scores and orientation diagnosis were inspected. The selected 34-asset pool deliberately stresses more complex layer stacks and is not a representative estimate for all layer counts. We report all 34 selected assets and a pre-result sensitivity excluding the three canonical aliases; neither group is certified never-seen because earlier training manifests are unavailable. The alias audit and original manifest are retained internally as `fresh_stage1_source_alias_audit.json` and `fresh_stage1_split_audit.json`.

Output-order correction (post hoc). Our first scorer stacked both checkpoints' PNGs in file-index order. An orientation diagnostic after inspecting those results showed that base Qwen emits back-to-front PNGs, whereas the fine-tuned checkpoint emits front-to-back PNGs. Reversing the fine-tuned order raises its composite PSNR from 18.06 to 26.36 dB on the same 34 assets (higher on $33/34$ when paired against its own unreversed output); reversing base Qwen instead lowers its PSNR from 20.91 to 15.94 dB (higher on $0/34$). This convention matches the already implemented generated-rig builder, which reverses the fine-tuned outputs. We therefore report base Qwen in native order and Bunraku in reversed order. The decision and corrected metrics are post hoc, not a new prespecified endpoint. Both original and corrected source-level scores, orientation diagnostics, and code are retained in the audit archive.

The base Qwen-Image-Layered model and the frozen full-parameter Stage 1 checkpoint receive the identical flat composite and the *artist layer count* as an oracle input. Both use seed 777, 25 denoising steps, a 640-pixel generation resolution, classifier-free guidance scale 4, and the same layer scorer. The reported comparison is thus conditional on knowing the correct number of layers; it does not test automatic count prediction. Each metric is paired by asset, and uncertainty uses 10,000 asset-bootstrap resamples. A base-model arm uses a full-GPU path; the fine-tuned arm uses a full-GPU path for most assets and CPU offload for a scheduling shard. In fixed-seed equivalence checks, full-GPU and offload outputs were bitwise identical for all 15 layers of one base-model asset and all 14 layers of one fine-tuned asset. This verifies those two assets, not every mixed-device case. Both paths use the same `diffusers` 0.38.0 and `transformers` 4.57.1 environment and its default fast Qwen2-VL image processor. This processor differs from the historical environment, so new scores are not pooled with the archived 120-case table. A GT-only pre-inference audit found that back-to-front compositing better reconstructs the supplied input for 49/51 assets; two accessories have RGB RMSE above 10/255 and both remain in the 34-asset subset. We also report a predeclared sensitivity excluding those two, without removing them from the primary 34-asset comparison. The initial run plan used 50 denoising steps. One base-model asset was generated at that setting, but no new model scores were inspected before the first-call runtime led us to reduce both arms to 25 steps while retaining the frozen 51-asset candidate pool. That 50-step pilot is excluded from the paired analysis. This compute-budget amendment was made before comparing the base and fine-tuned models on the new pool.

Post hoc layer-count audit on the same 34 inputs. All 758 requested PNG slots are present and have some nonzero alpha in each arm, but the inference alpha/plate filter retains a mean 17.76 fine-tuned layers versus 21.09 base-Qwen layers, from 22.29 requested per input. Only $3/34$ fine-tuned outputs versus $19/34$ base outputs retain the full requested count. Excluding the three canonical aliases yields 18.19 versus 21.39 eligible layers, with exact counts on $3/31$ versus $16/31$. Eligibility here means passing the alpha and plate filter; it does not certify a semantically correct layer or a successful triangle mesh. This structural audit was run after generation and is separate from the prespecified missing-layer-penalized overlap score. Per-asset counts are in `fresh_stage1_layer_count_posthoc_ge14_s25.json`.

Paired 34-asset Stage 1 result. With all oracle-count PNG slots in the frozen scorer, the source-equal artist-matched alpha IoU is 0.25596 for Bunraku versus 0.17335 for base Qwen (paired difference $+0.08262$, 95% asset-bootstrap interval $[+0.05947, +0.10498]$; Bunraku higher on $28/34$). Both arms have zero unmatched-slot rate by construction, so the scorer's matched and penalized alpha IoUs coincide even when a slot's alpha is too small to enter a rig. With the checkpoint-specific order convention applied, pair-weighted order consistency is 0.62336 versus 0.52764 (difference $+0.09573$, interval $[+0.05248, +0.13698]$; Bunraku higher on $26/34$). Composite PSNR is 26.3555 versus 20.9096 dB (difference $+5.4460$ dB, interval $[+4.2684, +6.6650]$; Bunraku higher on $31/34$). Composite SSIM is 0.8934 versus 0.8384 (difference $+0.0550$, interval $[+0.0415, +0.0695]$), while

Table 5: All 17 lower-layer inputs completed after the primary 34-asset scores and output-order diagnosis. Each Δ is Bunraku minus base Qwen; positive alpha IoU, order, PSNR, and alpha-filter-eligible count favor Bunraku. IDs follow the original frozen 51-input manifest order restricted to this remainder. Anonymized IDs and reported per-source measures are in the visual supplement's `stage1_51.csv`; original source names are withheld. This table is secondary, not a separate prespecified test.

ID	Type	Requested	$\Delta\alpha$-IoU	Δorder	ΔPSNR (dB)	Δeligible
R01	B	11	+0.198	+0.109	+5.30	-2
R02	B	13	+0.102	+0.077	+8.36	-1
R03	B	11	-0.023	+0.200	+6.48	-2
R04	B	12	+0.040	+0.121	-2.16	-4
R05	B	11	+0.012	+0.382	+6.12	-6
R06	B	12	+0.281	+0.045	-0.23	+0
R07	B	12	+0.124	-0.045	+18.33	-1
R08	B	11	+0.048	-0.273	+4.76	-4
R09	B	10	+0.041	+0.133	+24.33	-1
R10	B	11	+0.088	+0.091	+24.65	-4
R11	B	11	+0.164	-0.255	+5.70	-2
R12	B	11	-0.034	-0.164	+14.50	-5
R13	B	10	+0.091	+0.356	+2.09	-2
R14	B	10	+0.024	-0.089	+6.03	+0
R15	B	12	+0.176	+0.167	+16.08	-1
R16	B	12	+0.163	+0.258	+17.99	-1
R17	N	12	+0.161	+0.182	+3.90	-2

LPIPS is 0.1836 versus 0.2699 (difference -0.0863, interval $[-0.1097, -0.0660]$). Matched-layer RGB L_1 is 0.2914 versus 0.2766 (lower is better), with an interval crossing zero. This high-layer, oracle-count subset supports rest-render and order gains after a post hoc orientation correction, while the separate rig-eligibility audit shows worse count compliance.

Secondary completion of the full frozen 51-asset pool. We subsequently generated and scored the 17 previously unmeasured inputs with 10–13 artist layers under the same seeds, 25-step sampler, oracle counts, and corrected checkpoint-specific order convention. The original 51-asset manifest remained fixed, but this analysis is secondary because generation and scoring of these 17 followed inspection of the 34-asset results. Across all 51, Bunraku versus base Qwen scores 0.2574 versus 0.1698 matched alpha IoU (paired difference $+0.0876$, 95% source bootstrap interval $[+0.0676, +0.1077]$), 0.6243 versus 0.5351 pair-weighted order ($+0.0892$, $[+0.0491, +0.1298]$), and 27.84 versus 21.03 dB composite PSNR ($+6.81$ dB, $[+5.29, +8.47]$). These favorable image-level directions hold within the lower-layer 17, where PSNR is 30.82 versus 21.28 dB and alpha IoU is 0.2602 versus 0.1628; its order interval crosses zero. However, only 14.55 Bunraku layers per asset pass the deployment alpha filter versus 17.51 base-Qwen layers out of 18.63 requested; exact counts are $3/51$ versus $27/51$. In the lower-layer 17, the eligible counts are 8.12 versus 10.35 of 11.29 requested, with exact counts on $0/17$ versus $8/17$. The image-quality improvement and structural attrition therefore coexist across the entire frozen candidate list. The full per-asset scores, paired intervals, and filter counts are in `fresh_stage1_all51_ordercorrected_secondary_summary_s25.json` and `fresh_stage1_all51_layer_count_posthoc_s25.json`.

Excluding the three canonical aliases leaves 31 assets. Bunraku remains higher in alpha IoU (0.25814 versus 0.17004; paired $+0.08810$, interval $[+0.06361, +0.11140]$), order (0.62673 versus 0.53628; paired $+0.09045$, interval $[+0.04500, +0.13497]$), and rest PSNR (26.2171 versus 20.8557 dB; paired $+5.3614$ dB, interval $[+4.0958, +6.6865]$), but lower in eligible layer count (18.19 versus 21.39). Excluding the two preflagged GT-composite anomalies as well leaves 29 assets and the same directions; exact rows and all secondary metrics are in `fresh_stage1_ordercorrected_paired_summary_ge14_s25.json`.

Post hoc eligible-layer scorer sensitivity. We repeated Hungarian matching after removing slots rejected by the exact Stage 2 alpha/plate filter from both arms. The resulting rectangular match penalizes absent artist layers: Bunraku's penalized alpha IoU is 0.26016 versus 0.17294 for base (paired $+0.08722$, interval $[+0.06412, +0.11027]$, higher on $29/34$). The higher IoU survives this check, but matched-only IoU (0.33509 versus 0.18483) overstates usable coverage because the unmatched rate is 0.21021 versus 0.05969. Penalized order is 0.52013 versus 0.49957 (paired

Table 6: Per-asset Stage 1 paired differences on all 34 frozen high-layer inputs after the post hoc output-order correction. Δ is Bunraku minus base Qwen. Positive alpha IoU, order, PSNR, and usable-layer differences favor Bunraku. IDs follow the frozen manifest order; anonymized IDs and reported per-source measures are in the visual supplement's `stage1_51.csv`, while source names and audit flags are withheld. Alias and GT flags were found from input-only audits before paired scoring. Usable-layer differences are a post hoc filter audit.

ID	Type	Requested	$\Delta\alpha$-IoU	Δorder	ΔPSNR (dB)	Δusable
S01	B	17	+0.094	+0.206	+5.17	-2
S02	B	16	-0.032	+0.158	+6.63	-1
S03	B	16	-0.049	+0.075	-0.80	+0
S04	B	15	+0.052	+0.248	+5.09	-3
S05	B	32	+0.152	+0.163	+4.78	-8
S06	B	23	+0.136	+0.099	+4.63	-4
S07	B	29	+0.134	+0.185	+3.29	-6
S08	B	15	+0.117	+0.371	+0.53	-3
S09	B	15	-0.046	-0.105	+7.73	-5
S10	B	27	+0.133	-0.011	+10.65	-3
S11	B	20	+0.012	+0.026	+7.66	-9
S12	B	14	+0.191	+0.011	-0.99	-2
S13	B	16	+0.107	-0.267	+15.08	-4
S14	B	24	+0.152	-0.022	+5.36	-2
S15	B	34	+0.064	+0.012	+8.39	-8
S16	B	20	+0.095	-0.016	+6.31	-2
S17	B	28	+0.057	+0.167	+2.31	-4
S18	B	29	+0.076	+0.057	+8.66	-4
S19	B	31	+0.068	+0.114	+5.43	-3
S20	B	16	+0.058	-0.067	+2.74	+2
S21	B	34	+0.135	+0.271	+2.83	-6
S22	B	23	+0.097	+0.123	+5.12	-2
S23 (G)	B	29	+0.096	+0.113	+13.08	+0
S24 (G)	B	14	-0.017	-0.044	+5.26	-3
S25	B	29	+0.136	+0.227	+7.02	-4
S26	B	15	+0.065	+0.210	-1.48	-1
S27	B	33	+0.067	+0.218	+7.75	-3
S28 (A)	N	22	+0.089	+0.242	+7.49	-6
S29 (A)	N	17	-0.007	+0.132	+6.46	-3
S30 (A)	N	15	-0.005	+0.076	+5.01	-5
S31	N	23	+0.121	-0.020	+1.32	+0
S32	N	24	+0.200	+0.174	+5.95	-3
S33	N	23	+0.034	+0.075	+4.52	-5
S34	N	20	+0.225	+0.053	+6.16	-1

$+0.02056$, interval $[-0.01906, +0.05754]$), so this sensitivity does not establish an order advantage after filtering. The three-alias-excluded subset retains the alpha IoU gain. This filter choice was evaluated after the primary score; it is a structural sensitivity, not an independently frozen endpoint. The filtered-score audit was retained internally; this sensitivity is not separately reproducible from the visual supplement.

Post hoc painted-hidden-region audit. We define a hidden artist pixel as a pixel whose layer alpha exceeds $16/255$ and which later artist layers occlude by accumulated alpha above 0.5. The same full-layer Hungarian match used by the primary scorer pairs each artist layer with a predicted layer. Across 34 source-equal assets, predicted alpha covers 0.5972 of these hidden artist pixels for Bunraku versus 0.6531 for base Qwen (paired difference -0.0559, 95% interval $[-0.1493, +0.0384]$). Hidden-pixel premultiplied RGB L_1 is 0.2649 versus 0.2680 (lower is better; paired -0.0031, interval $[-0.0469, +0.0419]$). Neither interval excludes zero. The 31-asset alias-excluded sensitivity has the same inconclusive directions. This metric is measured after seeing the primary score and does not establish accurate semantic completion of occluded parts. Its definition, all source rows, and the scoring script are in `fresh_stage1_hidden_region_posthoc_ge14_s25.json`.

D DATASET

Corpus statistics. After deduplication, our corpus contains $8{,}884$ usable Live2D models ($7{,}773$ human/humanoid and $1{,}111$ non-human, tagged via Qwen3.6-Plus). These binary labels solely guide balanced benchmark sampling; training is supervised entirely by the extracted Live2D structural data.

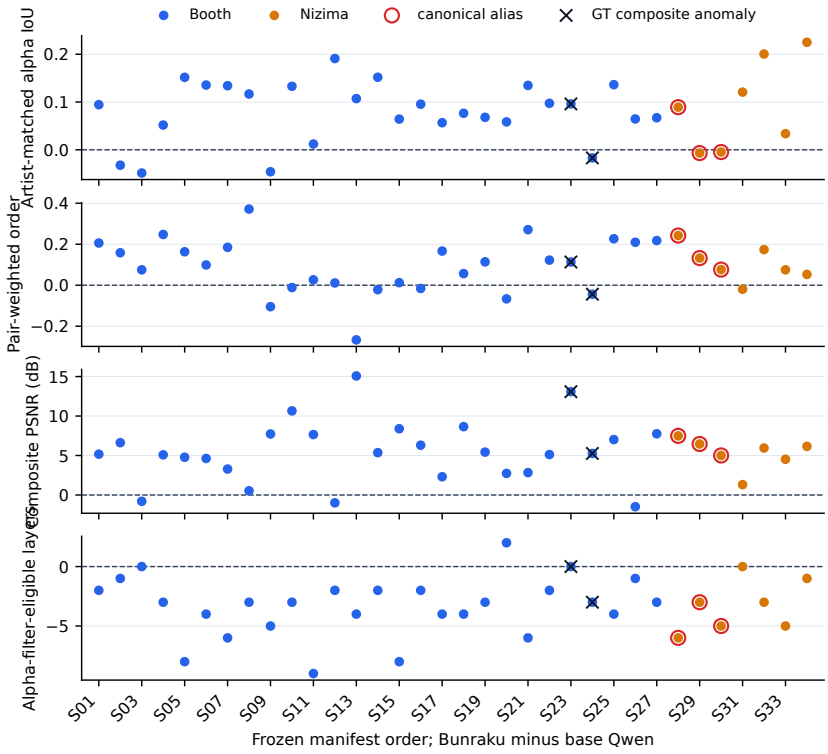

Figure 6: **Paired Stage 1 differences by source after the post hoc output-order correction.** Each point is one of the 34 frozen inputs in manifest order; positive values favor Bunraku for all four rows. Blue and orange denote Booth and Nizima sources, red rings mark three canonical aliases found in retained data roots, and black crosses mark two input-only GT-composite anomalies. The bottom alpha-filter count is a post hoc structural audit. The plot shows source variability without treating layers within an asset as independent samples.

Figure 7 shows the raw layer and group count distributions.

Data augmentation. The original data summary describes two structure-preserving augmentations (Figure 8): *motion replacement*, which applies alternative parameter curves and keyposes to a character's canonical layer stack; and *colour/texture replacement*, which alters RGBA appearance while retaining alpha masks, meshes, and keypose offsets. It reports approximately 50K Stage 1 and 35K Stage 2 augmented examples. These totals are not a verified count of distinct source characters or of the final resume manifest.

E BENCHMARK

Matched audit of independent and joint motion predictors. We re-evaluated the two saved checkpoints on the same 7,404 eligible layer–keypose records from 46 characters excluded from gradient training. They are a subset of the 50 characters used for per-epoch checkpoint selection and architecture sweeps, so this is a validation audit, not an untouched test. A record is eligible when its artist mesh has at most 80 vertices and at least one artist vertex moves more than 0.005 of canvas extent; the joint model still sees all layers as context. For both models, direction cosine is averaged over vertices whose *artist* displacement norm exceeds 0.005, then over records within each character, then over the 46 characters. This avoids the earlier both-moving gate, which selected a different scored vertex set for each predictor, and avoids scoring the joint model on records omitted

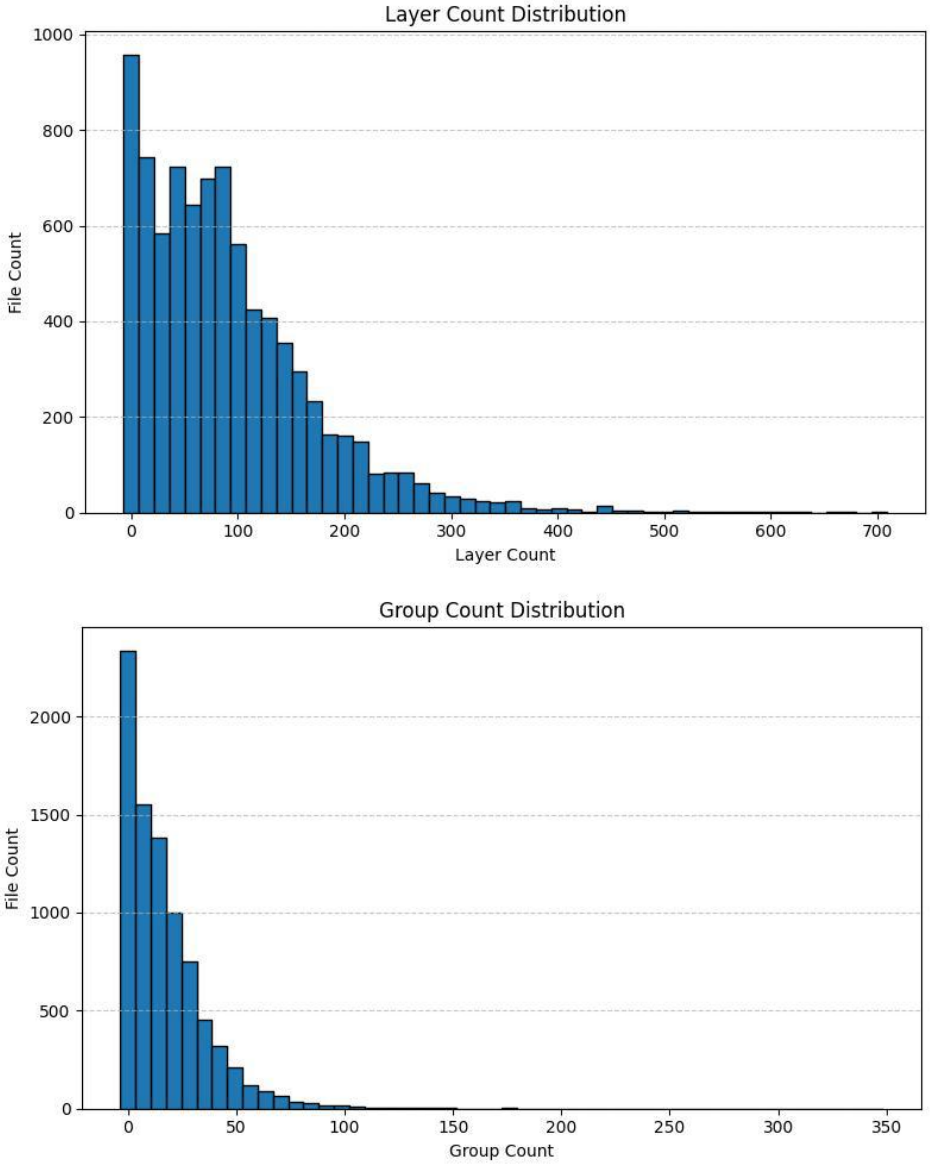

Figure 7: **Dataset statistics.** The distribution of layer counts (top) and group counts (bottom) across the dataset.



Figure 8: **Illustration of our data augmentation method.** (a) The original input data. (b) Pose and expression variations driven by underlying Live2D parameters. (c) Color and texture replacements applied to individual layers to enhance diversity.

by the per-layer implementation. The independent and joint per-character means are 0.6634 and 0.7109; medians are 0.7281 and 0.7493. The joint implementation improves $34/46$ characters. A character-level paired bootstrap with $10{,}000$ resamples and seed 2027 gives a mean difference of 0.0474 with 95% percentile interval $[-0.0117, 0.1165]$. The median paired difference is 0.0244 with interval $[0.0099, 0.0501]$. One positive outlier has only one eligible record and contributes a paired difference of $+1.219$. As an exploratory sensitivity check, restricting to the 43 characters with at least 10 eligible records leaves 32 joint improvements and a mean difference of $+0.0178$ (95% bootstrap interval $[-0.0247, 0.0499]$). Thus improvement is common but the mean effect is uncertain and sensitive to sparse characters. The original record-averaged both-moving scores (0.693/0.736) are retained below for provenance, but are not the primary comparison. This audit still compares different model architectures, so a same-architecture attention-mask control below addresses the cross-layer mechanism separately. Per-character scores are retained internally as `paired_ablation_audit.json`; rerunning checkpoint inference requires the underlying artist-rig records, which are subject to the rights limitation in the ethics statement.

Frozen 50-character attention intervention. Before fitting the revised pair, we selected 50 listing-ID-singleton characters from the existing motion pack using only source IDs and artist-motion coverage: each has at least 20 moving records under at least two of the eight controls. The fixed random seed 20270925 selected 12 Booth and 38 Nizima characters. These IDs are excluded from both gradient training and validation in the revised runs. A post-freeze, input-only screen of five frozen DINO layer embeddings per character finds one test character with cosine 0.996 to a gradient-training character. Similarity does not establish source identity, but we predeclared an additional 49-character sensitivity excluding this possible visual alias before reading any test scores.

Each of three paired seeds trains the same 5.08M-parameter, six-block `V5JointLM` for 20 epochs with AdamW, batch size eight, learning rate $2 \cdot 10^{-4}$, vertex cap 28, layer cap 52, and sequence cap 1,200. The implementation takes the first 52 artist layers in stored order, samples up to 28 vertices per retained layer with a fixed layer-index seed, and truncates any remaining sequence after token 1,200. An audit of the 11,660 active training pose groups for this split found 1,943 (16.7%) with more than 52 layers and 3,036 (26.0%) whose retained vertices exceed 1,200 tokens. At evaluation and in the released generated-rig inference path, all available mesh vertices are passed without that subsampling, so model behavior on longer sequences is a distribution shift. This cap is a practical limitation of the current predictor, not a property of the rig format. The only architecture switch restricts self-attention to vertices of the same layer; a synthetic perturbation check confirms that changing another layer's appearance cannot affect the blocked model's output. Both arms in each seed use identical training records, initialization seed, optimizer, schedule, and historical 50-character

validation pool for per-epoch checkpoint selection. This is a new pair of models and must not be compared numerically to the separately trained 30-epoch released checkpoint.

Once each checkpoint is selected, the frozen test is scored on artist layers and vertices under true generation. Moving-vertex direction cosine averages vertices with artist displacement above 0.005 of normalized canvas extent within records, then records within characters. Static-part displacement norm uses artist displacement at most 0.005. A nearby cross-layer proxy searches each vertex's 12 nearest vertices overall, selects the first on another layer within 0.025 normalized canvas units when one exists, and compares their predicted relative offset with the artist's. This approximation can miss a cross-layer vertex outside the 12-candidate set and does not measure rendered seams. Unlike cosine, which uses only records with at most 80 artist vertices, the static and relative-offset measures include larger records. All pairs use the same test records and scoring gates. For the three-seed summary, we average the seeds within each character and bootstrap *characters*, not 150 separate character–seed observations, for 10,000 paired resamples. The full 50 and possible-alias-excluded 49 character summaries are both reported.

Frozen-test result. On the 50 test characters, the three-seed within-character mean cosine is 0.73733 for joint attention and 0.72733 for the layer-blocked control. Their paired mean difference is $+0.01000$, with a 10,000-resample character-bootstrap 95% interval $[-0.00022, 0.02253]$ and $28/50$ positive character differences. The static-part predicted-offset norm is 0.02145 versus 0.02213 (difference -0.00068, interval $[-0.00435, 0.00398]$). The nearby cross-layer relative-offset error is 0.01716 versus 0.02550 (difference -0.00834, interval $[-0.00985, -0.00698]$), with the joint model lower on all 50 characters. Excluding the one pre-screened possible visual alias leaves a relative-error difference of -0.00820 (interval $[-0.00967, -0.00682]$) and a cosine difference of $+0.01042$ (interval $[-0.00004, 0.02305]$). Table 11 shows the paired seed means. These are artist-layer, artist-vertex tests; the local relative-offset proxy is not a rasterized seam metric or evidence that the generated-layer motion control succeeds.

Post hoc zero-motion control for the local proxy. An independent review prompted us to score the same nearby cross-layer pairs with every predicted offset set to zero. The zero-motion field has relative-offset error 0.01138, lower than joint attention's 0.01716 and blocked attention's 0.02550. Joint minus zero is $+0.00578$ (95% paired character-bootstrap interval $[0.00465, 0.00702]$); joint attention beats zero motion on only $1/50$ characters. All $999{,}897$ active directed pair instances match the saved attention-audit pair counts. Of these, $696{,}039$ (69.61%) have artist *relative* displacement below 0.005 canvas units despite at least one vertex meeting the individual motion gate. A common translation also cancels in this metric. Consequently, the joint-versus-blocked improvement is a property of the learned variants, not evidence that this proxy establishes useful relative motion. This control was added after the original test scores were seen; the executable audit and all per-character values are retained internally in `fresh_attention_zero_motion_audit.json`.

Post hoc relative-motion sensitivity. To test whether nearly static cross-layer pairs explain this failure, we reused the same directed candidate pairs and retained those with artist relative displacement above 0.005 or 0.02 canvas units. The first gate retains $303{,}858$ pair instances across all 50 characters: character-mean relative errors are 0.02956 (joint), 0.03529 (blocked), and 0.02839 (zero). Joint minus zero is $+0.00117$ with a paired character-bootstrap 95% interval $[+0.00010, +0.00230]$; joint beats zero on $26/50$ characters. The stricter 0.02 gate retains $114{,}043$ instances: the corresponding errors are 0.04707, 0.05056, and 0.05147. Joint minus zero is -0.00440 (interval $[-0.00588, -0.00289]$, lower on $40/50$ characters), and joint minus blocked is -0.00349 (interval $[-0.00482, -0.00223]$, lower on $38/50$). We selected these gates after seeing the original proxy and zero-motion results, so this is an exploratory sensitivity rather than a new confirmatory endpoint. It does not replace the full-pair failure or measure a visible seam. The per-character and per-seed audits are retained internally in `fresh_motion_strong_pairs_3seed_summary.json` and the corresponding seed files.

Post hoc absolute-displacement control. We then reran all six frozen checkpoints to score $\|\hat{D} - D\|_2$ directly on artist vertices. This audit was specified after the attention and zero-motion results and is exploratory on the same 50-character test. For vertices with artist displacement greater than 0.005 canvas units, the three-seed character-mean errors are 0.04929 (joint), 0.05316 (blocked), and 0.07319 (zero). Joint minus blocked is -0.00386 with paired character-bootstrap 95% interval

$[-0.00723, -0.00059]$ and $35/50$ lower character errors; joint minus zero is -0.02390 with interval $[-0.03223, -0.01586]$ and $43/50$ lower errors. All-vertex errors are 0.04746, 0.05129, and 0.06845, respectively. The pre-screened possible-alias exclusion leaves a 49-character moving-vertex difference of -0.00297 (interval $[-0.00580, -0.00005]$). Mean predicted moving-vertex norm is 0.05852 for joint attention versus 0.05151 for blocked attention; the artist target mean is 0.07319. This indicates undersized motion even on artist vertices. Unlike the cosine audit, these absolute errors include records with more than 80 artist vertices. Changing from the original native multi-head-attention inference path to scaled-dot-product attention reduced peak memory and reproduced every archived per-character cosine and relative score within 2×10^{-6}. Anonymized per-character moving-error values appear in the visual supplement; the executed scripts and full audit JSON are retained internally. None of these artist-rig scores supersedes the negative generated-rig motion-residual control.

The new attention arms on generated layers. We also applied the actual freshly trained seed-1 joint and layer-blocked checkpoints to the same eight Stage 1 generated layer stacks used by the historical renderer diagnostic. The stacks, meshes, textures, artist target poses, and GT-only motion masks are identical between arms. For each stack we subtract its own rest render before comparing pose changes; leaving the stack still therefore scores exactly 1.000. Character-mean normalized motion-residual error is 1.168 for joint and 1.106 for blocked attention. Joint minus static is $+0.168$ (paired character-bootstrap 95% interval $[+0.060, +0.273]$), with joint below static for $1/8$ characters; joint minus blocked is $+0.062$ (interval $[-0.040, +0.171]$), lower for $2/8$. All eight characters were in the historical Stage 2 checkpoint-selection validation set, so this is a diagnostic of the deployed generated stacks, not an independent generated-rig test. It is unfavorable even under this validation setting and does not show a cross-layer benefit at the generated rig output. The executable inference and renderer are retained internally, with per-character scores in `fresh_generated_rig_validation_seed1_cap0_summary.json`. As a post hoc check of the training/inference token-count shift, we held the full rendering meshes fixed but passed at most 28 fixed-seed sampled vertices per layer through each network, then interpolated offsets back to every mesh vertex with inverse-distance weighting. The eight-character residual means remain 1.168 (joint) and 1.109 (blocked), versus 1.000 static; joint improves on only $1/8$ characters. This single-seed diagnostic does not repair generated-rig motion, and it does not validate the full-vertex model on a fresh source pool. The exact character rows are in `fresh_generated_rig_validation_seed1_cap28_summary.json`. We further projected each joint predicted field onto either one rigid translation for the whole character or separate mean translations for each layer, without changing meshes or the rest render. Their eight-character motion-residual means are 1.311 and 1.172, respectively, versus 1.168 for the full field and 1.000 static; each beats static for only $1/8$. These are post hoc projections of the same predictor, not independent learned or retrieval baselines. They suggest that simply suppressing nonrigid variation does not repair the generated-stack failure. The per-character audit is `fresh_generated_rig_rigid_projection_summary.json`.

Generated-mesh proximity contacts. As another post hoc check on those eight validation rigs, we pair each rest mesh boundary vertex with the nearest boundary vertex on a different layer within 10 canvas pixels. This is a proximity proxy, not verified semantic attachment or a rendered seam measure. The source-equal mean positive increase in pair separation is 2.44 pixels for joint attention and 5.03 for blocked attention; the fractions with more than five pixels of new gap are 0.157 and 0.282. A proximity-Laplacian smoothing step reduces the mean gap proxy to 0.234 pixels but changes artist-target rest-subtracted motion error from 1.168 to 1.174, still worse than static at 1.000. Thus fewer nearby boundary gaps do not imply more accurate animation. The exact scores and script are retained internally as a post hoc pilot. We fixed this proximity audit and its 10-pixel radius before scoring the new 12-source generated-rig test; its protocol and script are retained internally as `fresh_generated_rig_attachment_protocol12.json`.

Validation-calibrated amplitude control. On those same eight validation rigs, multiplying each predicted displacement by 0.25, while holding the layer stack, mesh, and rest render fixed, reduces joint motion-residual error from 1.168 to 1.034 and increases the number of characters beating the static same-stack control from 1/8 to 4/8. The validation-calibrated scale remains worse than static on the source-equal mean. We froze 0.25 as a *secondary* control for the new 12-source test before its scores were examined; the original full-amplitude joint versus blocked comparison remains primary.

A half-amplitude check on the old eight scores 1.084 and was not selected. Exact rows, both protocol versions, and the scaling script were retained internally; the summary result is reported here.

Prospective generated-rig cohort and endpoints. Before fitting the next attention pair or seeing its test scores, we matched the preselected high-layer PSD pool to artist-motion rigs by exact vendor and source-name suffix, yielding 12 sources. All 12 are the primary cohort. An input-only contact sheet identified four visually mismatched or incomplete PSD/artist-rest pairs, so the remaining eight form a predeclared sensitivity analysis; none is removed from the primary result. The new pair excludes all 12 and the prior fresh 50 from gradient training, leaving 1,332 training characters. It selects checkpoints on 49 historical validation characters; one former validation ID was moved to the new test. A frozen-DINO screen found no new test character whose nearest training-layer embedding cosine reached 0.99, although appearance similarity cannot certify source identity. Historical Stage 1 manifests remain incomplete, so this cohort certifies Stage 2 source-ID disjointness only. Both Stage 1 variants receive the artist layer count and the same 25-step sampling settings. The generated layer stack and mesh are fixed when joint, blocked, and static motion are compared. Primary motion error subtracts each stack's own rest render, uses a mask defined only from artist motion, and averages by source; static scores one. The validation-calibrated 0.25-amplitude joint control, rest-pose fidelity, motion-region PSNR, and rest-defined proximity contacts are secondary. A separate one-seed artist-vertex test uses moving-vertex absolute error as its prespecified primary endpoint. Source manifests, split checks, scripts, and the pre-score protocol were retained internally. The visual supplement supplies anonymized source-level scores but not the training code, weights, or artwork.

Prospective generated-rig rendering result. The prespecified primary score averages the rest-subtracted motion residual within each source and then gives equal weight to all 12 sources (77 non-rest keyposes). On identical fine-tuned generated stacks and meshes, the source means are 1.35883 for joint attention, 1.32310 for layer-blocked attention, and 1.00000 for the same-stack static control. Joint minus static is $+0.35883$ (paired source-bootstrap 95% interval $[+0.18176, +0.56695]$), with joint lower for $0/12$ sources. Blocked minus static is $+0.32310$ (interval $[+0.15557, +0.53757]$), also lower on $0/12$. Joint minus blocked is $+0.03574$ (interval $[-0.01525, +0.09383]$), lower on $6/12$; the attention comparison is inconclusive. The predeclared eight-source input-compatible sensitivity (55 poses) gives 1.47056 joint, 1.41391 blocked, and 1.00000 static, with neither learned arm below static on any source. The validation-calibrated quarter-amplitude joint arm reaches 1.21603 on all 12 and 1.30003 on the comparable eight, but still improves no source over static. These rendering results are motion-sensitive and cannot be replaced by an absolute frame PSNR or artist-vertex metric.

Post hoc three-seed generated-rig stability. After inspecting the seed-1 prospective result, we trained seeds 2 and 3 with the same frozen 12 sources, generated layers, meshes, pose list, and renderer. These later seeds are a stability check, not independent prospective confirmations. Averaging seeds within each source before a source bootstrap gives motion-residual ratios of 1.34943 joint, 1.30861 blocked, and 1.00000 static. Joint minus static is $+0.34943$ (95% interval $[+0.19223, +0.53104]$), with no source below static; joint minus blocked is $+0.04082$ (interval $[-0.00610, +0.09203]$), lower on $2/12$ sources. Each individual seed also has $0/12$ joint and $0/12$ blocked sources below static. On the predeclared eight-source input-compatible subset, the three-seed means are 1.45077 joint and 1.38894 blocked versus 1.00000 static, with neither learned arm below static for any source. The full source-level rows and seed values are in `fresh_generated_rig_test12_render_3seed_posthoc_summary.json`.

The primary fine-tuned-stack motion scores above already used its correct front-to-back output convention. In the secondary base-Qwen comparison, an erroneous reversal of base PNGs had depressed its rest-pose PSNR to 12.0473 dB; the original manifest and scores are retained. We rerendered that arm in its native back-to-front order after inspecting the error. Corrected base rest PSNR is 14.8732 dB versus 14.7619 dB for fine-tuned layers on the same 12 illustrations and supplied artist layer counts. The paired fine-minus-base source difference is -0.1112 dB (95% interval $[-0.8176, +0.7093]$; fine higher on $6/12$). With the same joint checkpoint, the corrected base-layer motion residual is 1.34996 versus 1.35883 for fine-tuned layers (difference $+0.00887$, interval $[-0.08099, +0.09043]$; fine lower on $6/12$). The base stack also loses to static on every source. This is a post hoc correction of a secondary arm, not a new independent experiment, and rest fidelity does not establish a motion benefit. The rest-defined nearby-boundary gap-increase proxy is smaller for joint than blocked motion

Table 7: All 12 prospective generated-rig sources. The compatibility flag was set using input images only, before motion scoring. Rest PSNR compares the fine-tuned and base-Qwen stacks against the artist rest render; the three motion columns use the fine-tuned stack. Every same-stack static control scores exactly 1.000. Anonymized source IDs and full-precision values for the reported motion measures are in the visual supplement's `generated_rigs_18.csv`; the source-name mapping is withheld.

Rig	Compatible	Poses	Rest fine	Rest base	Joint	Blocked	Joint $\times.25$
bench_00	Y	7	15.03	14.99	1.447	1.237	1.285
bench_01	N	6	9.56	9.25	1.266	1.287	1.114
bench_02	Y	6	16.27	16.93	1.339	1.341	1.202
bench_03	Y	7	26.40	23.00	1.059	1.028	1.039
bench_04	Y	4	17.98	19.11	1.599	1.674	1.302
bench_05	N	7	8.30	10.90	1.153	1.146	1.045
bench_06	Y	11	18.98	19.12	1.365	1.219	1.080
bench_07	Y	2	11.15	11.08	2.229	2.265	2.029
bench_08	N	3	7.07	7.03	1.081	1.039	1.023
bench_09	Y	7	14.68	14.48	1.003	1.042	1.018
bench_10	N	6	16.84	17.64	1.040	1.094	1.010
bench_11	Y	11	14.88	14.95	1.724	1.505	1.445

(1.928 versus 4.561 pixels; paired interval for the difference $[-4.648, -1.128]$, lower on $11/12$), yet this is geometric proximity, not a labeled seam or attachment endpoint, and both variants lose to static on the primary render-motion measure. Exact source-level rows and source-bootstrap code are in `fresh_generated_rig_test12_base_ordercorrected_posthoc_summary.json`, the original `fresh_generated_rig_test12_summary.json`, and the retained analysis scripts. Stage 2 source IDs are excluded from training and selection; Stage 1's earlier manifests are incomplete, so these results do not certify entire-pipeline source disjointness.

With the corrected base order, the paired rest-PSNR median is near zero and the source-level interval crosses zero. The generated-rig motion failure is not driven by an individual source: joint motion exceeds the static score on every source, including the eight input-compatible pairs.

Requested versus usable layers (post hoc structural audit). Both Stage 1 arms produced every requested PNG slot for these 12 sources (275 slots per arm), and every PNG has at least one nonzero-alpha pixel. After resizing to a 600-pixel long side, however, 56 fine-tuned slots and 20 base-Qwen slots contain fewer than 30 pixels with alpha above $16/255$; Stage 2 discards them before meshing. No full-frame plate is removed in this cohort. The fine-tuned rigs retain a mean 18.25 usable layers from a mean 22.92 requested, with the exact count on $1/12$ sources; the base-Qwen figures are 21.25 and $4/12$. The mean source-proportional shortfall is 22.1% fine-tuned versus 9.4% base. These are count and opacity audits, and the fine-minus-base usable-layer difference is -3.00 layers (paired source-bootstrap 95% interval $[-4.33, -1.58]$; fine-tuned fewer on $10/12$ sources). This is a post hoc structural endpoint, not a prespecified Stage 1 quality test. The counts are not judgments about whether the surviving layers are semantic, correctly ordered, or complete behind occluders. The dropped slots may remove needed parts. The per-source audit and exact inference filter are in `fresh_generated_rig_test12_layer_count_posthoc_audit.json` and the internally retained `infer_joint.py`. For example, `bench_10` requests 23 parts but retains 14; seven discarded PNGs have maximum alpha no greater than $6/255$ in their saved images. Lowering the mesh threshold would not recover opaque painted parts from those slots.

Post hoc unwanted-motion diagnostic. To address whether the renderer makes parts move where the artist remains still, we measure each candidate's pose-minus-own-rest RGB RMS outside the GT-only motion mask used for the primary endpoint. The static same-stack control is exactly zero by definition. Averaging keyposes within sources and then sources, joint attention changes these outside-mask pixels by 28.31 8-bit levels (95% source-bootstrap interval $[21.12, 36.48]$); blocked attention scores 26.85 ($[19.98, 33.91]$). The artist's pose-minus-rest change in that region is only 0.31 levels. Joint minus blocked is $+1.46$ levels (interval $[-2.59, +5.26]$); the attention comparison is inconclusive. This metric was requested after review of the primary motion result and is not a replacement endpoint. The large outside-mask change shows unwanted motion in both learned

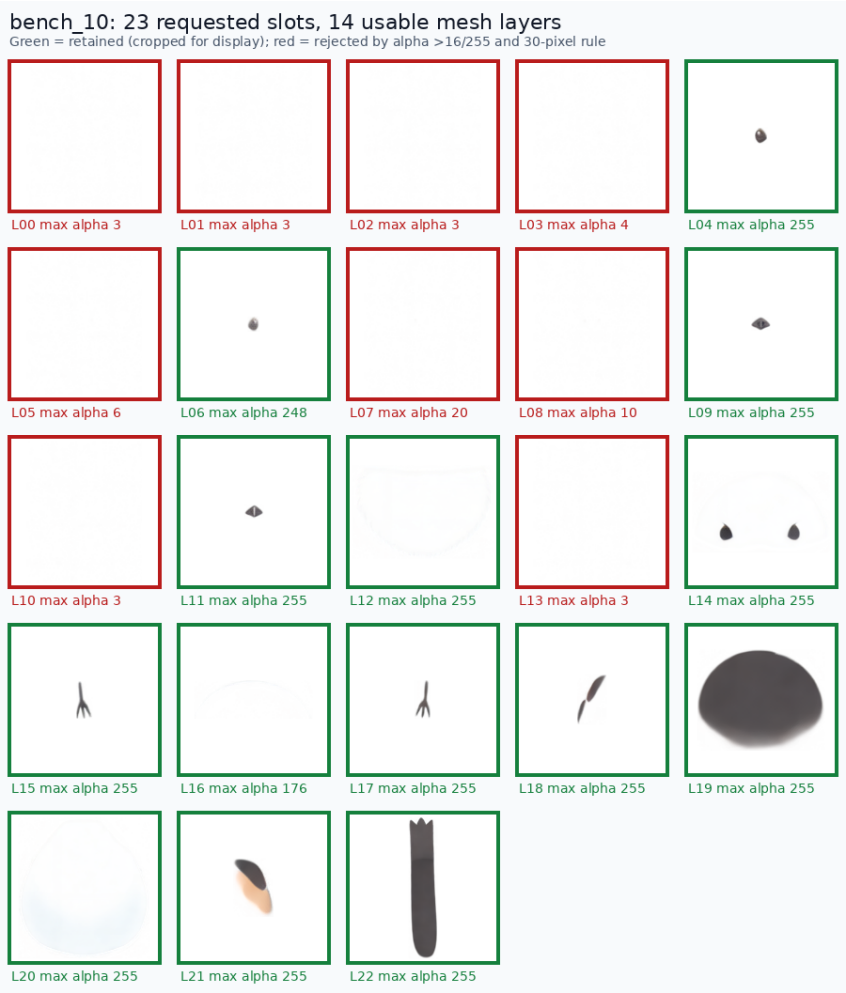

Figure 9: **Layer-slot attrition example from the prospective cohort.** The fine-tuned Stage 1 model was asked for 23 layers on `bench_10`; nine PNGs (red) fall below the inference alpha-size threshold, leaving 14 meshed layers (green). Retained layers are cropped to their alpha support for legibility; the dropped, nearly transparent slots are shown on their full canvas. The figure illustrates count failure, not whether the remaining painted parts are semantically correct. Selection of this example was post hoc; Table 8 reports all 12 sources.

arms under the common rest framing; differences in generated silhouette and part placement can contribute, so the measure does not isolate a motion-network defect. In these 12 sources, missing fine-tuned layer count has Spearman correlation -0.34 with the prespecified joint motion residual; with only 12 sources and different art complexity, this descriptive association does not isolate layer attrition as the cause of motion failure. Exact masks, per-pose rows, and source summaries are in `fresh_generated_rig_test12_unwanted_motion_posthoc_audit.json`. Table 9 places the source-level attrition and motion measurements side by side.

Post hoc control-level failure diagnosis. We rerendered the same 77 keyposes and decomposed the primary residual by named control, retaining the artist-only mask and each candidate's own rest render. The recomputed source means exactly match the frozen primary scores. Within the artist-motion mask, the source-mean RGB pose-change cosine with the artist is only 0.122 for joint and 0.140 for blocked attention, although the corresponding change-norm ratios are 0.991 and 0.954. Thus the predicted change has approximately the same image-space norm on average but is poorly aligned with the target change. Per-control source-equal joint residuals are 1.326 for `ParamAngleX` (11 sources), 1.371 for `ParamAngleY` (10), 1.136 for `ParamAngleZ`

Table 8: Layer-slot attrition on the 12 generated rigs. All requested PNG slots have some nonzero alpha, but only masks meeting the inference alpha threshold become mesh layers. F and B denote fine-tuned and base-Qwen Stage 1 outputs.

Rig	Requested slots	Nonempty F/B	Meshed F	Meshed B
bench_00	23	23/23	17	21
bench_01	29	29/29	23	29
bench_02	15	15/15	9	12
bench_03	15	15/15	8	13
bench_04	16	16/16	14	12
bench_05	34	34/34	28	34
bench_06	29	29/29	25	29
bench_07	15	15/15	11	12
bench_08	33	33/33	29	32
bench_09	23	23/23	23	23
bench_10	23	23/23	14	19
bench_11	20	20/20	18	19
Mean	22.92	22.92/22.92	18.25	21.25

Table 9: Post hoc failure diagnostics on the same 12 prospective generated rigs. Missing means requested minus alpha-filter-eligible fine-tuned layers. Motion residual is the prespecified joint/static ratio (lower is better; static is 1). Outside-mask RMS is RGB pose-minus-own-rest change outside the artist-motion mask, in 8-bit levels; the artist outside-mask source mean is 0.31. The diagnostics were defined after the primary result and do not identify a causal failure mechanism.

Rig	Missing layers	Motion residual	Outside-mask RMS
bench_00	6	1.447	27.8
bench_01	6	1.266	37.4
bench_02	6	1.339	19.5
bench_03	7	1.059	11.0
bench_04	2	1.599	29.8
bench_05	6	1.153	26.3
bench_06	4	1.365	12.2
bench_07	4	2.229	56.6
bench_08	4	1.081	21.1
bench_09	0	1.003	27.6
bench_10	9	1.040	19.8
bench_11	2	1.724	50.6
Source mean	4.67	1.359	28.3

(10), 2.249 for ParamBodyAngleX (2), 1.225 for ParamBodyAngleZ (2), and 1.340 for ParamMouthOpenY (7); all exceed the static control's 1.000. These are descriptive post hoc subsets, not independent source samples or a diagnosis of the model alone: missing or misplaced layers also alter the rendered RGB change. Source, control, and keypose rows are in fresh_generated_rig_test12_control_failure_posthoc_audit.json.

Post hoc layer-ID convention audit. The original Stage 2 loader numbers layers in their first-record order, whereas deployment assigns IDs after sorting the numerical draw-rank suffix for back-to-front compositing. Among 9,692 training pose groups with at least five ranked layers, 7,640 have a negative Spearman correlation between record index and draw rank (median -0.456). This is a plausible train–deployment mismatch, rather than proof that it caused the motion failure. Reassigning model layer IDs in reverse order at inference while holding weights, generated layers, meshes, and poses fixed is a post hoc diagnostic on the inspected 12 sources. Joint residual changes from 1.359 to 1.335 (paired interval for the change $[-0.078, +0.018]$), with just $1/12$ sources below the 1.000 static control. Blocked residual changes from 1.323 to 1.313 (interval $[-0.016, -0.005]$), with $0/12$ below static. Thus reversing IDs in the existing checkpoint does not resolve the rendered-motion failure. The training and rerender audits are retained internally; the visual supplement gives source-level scores for the main rendered-motion endpoint.

Table 10: New six-source generated-rig test of the revised draw-order training convention. O denotes first-record-order training; C denotes canonical back-to-front training. All arms share the same generated layers, meshes, artist-only motion mask, and static control. The four-source subset passed an input-only visual comparability screen. Bold is the lowest motion-residual point estimate within each row. The intervention was motivated by the prior 12-source failure; this is a new test of that intervention, not an untouched test of the research program.

Rig	Meshed	Joint O	Joint C	Blocked O	Blocked C	Static
bench_00	7	0.907	0.900	0.868	**0.863**	1.000
bench_01	8	1.322	1.199	1.262	1.194	**1.000**
bench_02	5	1.417	1.090	1.222	1.118	**1.000**
bench_03	11	2.823	1.489	1.820	1.498	**1.000**
bench_04	11	1.053	1.046	1.065	1.040	**1.000**
bench_05	10	1.447	1.453	1.367	1.310	**1.000**
All 6	–	1.495	1.196	1.267	1.170	**1.000**
Comparable 4	–	1.550	1.131	1.244	1.130	**1.000**

To test an intervention in training rather than only at inference, we froze all six additional exact source-name matches from the 10–13-layer remainder of the original 51-input manifest. These six had not contributed to Stage 2 training or validation in the revised split; four passed an input-only rest-image comparability screen, while all six remain the primary cohort. The revised training loader sorts each pose's layer records by numeric back-to-front draw rank before assigning model IDs. We train joint and layer-blocked models with the same 5.08M-parameter architecture, 20 epochs, seed, data split, and validation-cosine checkpoint rule as matched first-record-order controls. Because the loader applies its 52-layer and 1,200-token caps after ordering, sorting can also change which training tokens survive a cap; this comparison tests the full draw-order convention, not the layer embedding alone. The intervention was motivated by the previously inspected 12-source failure. Source selection, the same-stack static endpoint, and the old-order control amendment were fixed before any new six-source motion score was inspected. A structural count audit before motion scoring finds that fine-tuned stacks have 8.67 alpha-filter-eligible layers on average out of 11.67 requested, versus 11.33 for base Qwen; exact requested counts occur on $0/6$ versus $5/6$ inputs. On the six newly generated rigs and 26 non-rest keyposes, canonical-order joint attention has source-mean motion residual 1.196 (paired joint-minus-static 95% source-bootstrap interval $[+0.030, +0.368]$), layer-blocked attention has 1.170 (interval $[+0.016, +0.334]$), and the same-stack static control is 1.000. Each learned arm is below static on only $1/6$ sources. Joint minus blocked is $+0.026$ (interval $[-0.010, +0.075]$), so cross-layer attention has no established advantage here. On the four pairs passing the input-only comparability screen, joint and blocked residuals are 1.131 and 1.130 versus static 1.000, each below static on $1/4$; both source-bootstrap intervals include zero for the difference from static. These six test sources were frozen before their motion scores, but the training repair was motivated by the earlier 12-source failure. The two visibly unmatched rest-image pairs remain in the primary six-source aggregate. Exact source rows are in `fresh_generated_rig_remainder6_canonical_eval_summary.json`. The matched first-record-order pair scores 1.495 joint and 1.267 blocked on these same six sources. Sorting the training records into canonical draw order therefore lowers joint residual by 0.299 (paired source-bootstrap 95% interval for canonical minus old order $[-0.723, -0.022]$; $5/6$ sources lower) and blocked residual by 0.097 (interval $[-0.191, -0.031]$; $6/6$ lower). The visually comparable four sources show the same direction: joint 1.550 to 1.131 and blocked 1.244 to 1.130. This matched intervention supports the importance of the draw-order training convention, but neither repaired arm beats the same-stack static control on average. As caps are applied after sorting, the experiment does not isolate layer IDs from token selection. The training-pair audit verifies identical split exclusions, architecture, schedule, and checkpoint rule; the evaluation audit verifies identical artist targets and generated rest renders for the two training conventions. Per-source values and the fixed amendment are in `fresh_generated_rig_remainder6_order_ablation_summary.json` and the visual supplement's per-source score table. A post hoc replay of the same six rigs finds source-mean candidate pose-minus-rest RGB RMS outside the artist-motion mask of $20.11/255$ for joint and $18.86/255$ for blocked, versus $0.52/255$ artist motion and zero for the static control. As in the original 12, rest-silhouette mismatch can

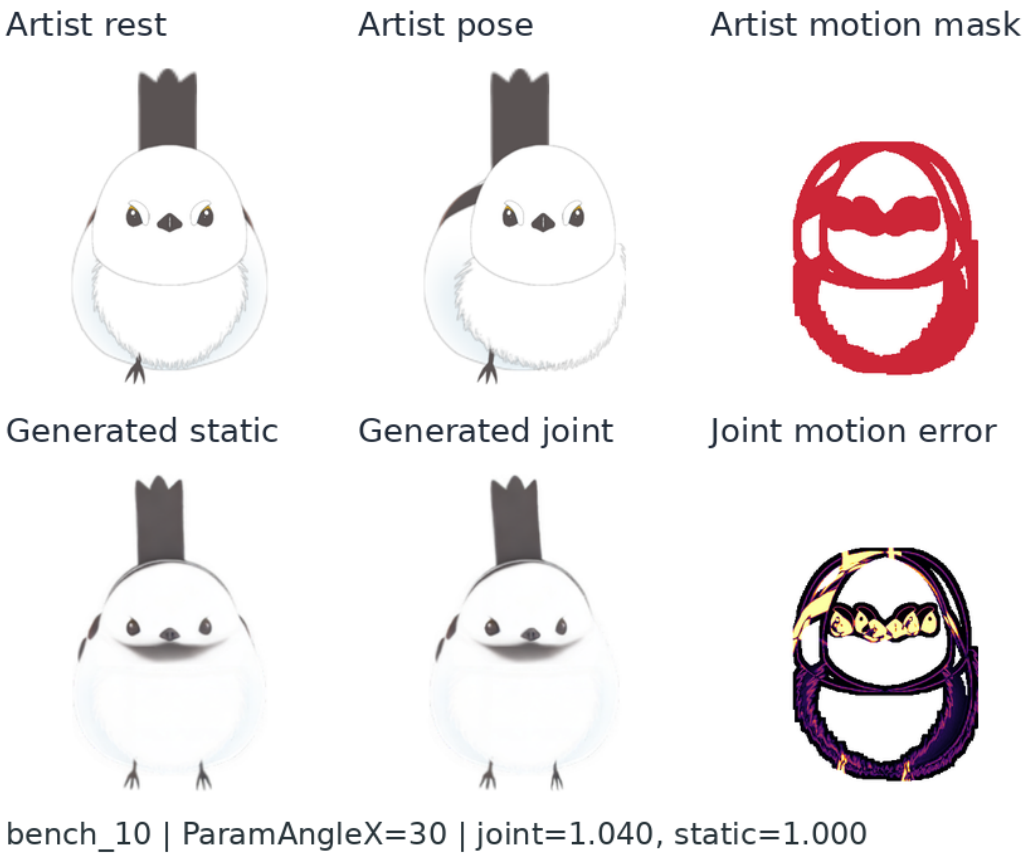

Figure 10: **Additional prospective generated-rig failure (post hoc visual selection).** `bench_10` at the positive `ParamAngleX` extreme. The artist rest/posed frames define the target motion and red artist-only mask; generated rest is the same-stack static control. The heatmap shows the joint pose-minus-rest error within that mask, clipped at RGB root-mean-square difference $100/255$. The displayed source-mean ratio aggregates all scored keyposes. Candidate and artist rigs are separately fitted to their rest boxes, so this residual measures rendered target alignment rather than semantic attachment quality. The full 12-source analysis determines the quantitative conclusion.

contribute. This diagnostic was added after the six motion scores and is recorded in `fresh_generated_rig_remainder6_unwanted_motion_posthoc_audit.json`.

Prospective artist-vertex replication result. The one-seed checkpoints were selected without the 12 test characters. On the prespecified moving-vertex L_2 endpoint, source-equal means are 0.06426 (joint), 0.06396 (layer-blocked), and 0.08266 (zero motion). Joint minus blocked is $+0.00029$ with a 10,000-resample paired source-bootstrap 95% interval $[-0.00865, +0.00838]$; joint is lower for $6/12$ sources. Joint minus zero is -0.01841 with interval $[-0.03793, +0.00206]$, so this smaller test does not establish even that comparison. The corresponding mean direction cosines are 0.66960 and 0.67872. The predeclared eight-source input-compatible sensitivity has moving errors 0.06602 (joint), 0.06420 (blocked), and 0.08337 (zero), with a joint-minus-blocked interval $[-0.00945, +0.01177]$ and $4/8$ sources lower. These are artist-vertex scores; they do not test generated layer stacks. The source-level rows, eligibility audit, fixed protocol, and summarizer are retained internally in `fresh_generated_rig_test12_artist_vertex_audit.json` and `fresh_generated_rig_test12_artist_vertex_summary.json`. Training seeds 2 and 3 were started *after* inspecting seed 1, so any later multi-seed result is a post hoc stability analysis rather than a second untouched replication.

Table 11: Fresh 50-character attention intervention: joint minus layer-blocked per-seed means. Absolute motion error was audited after the original test. A higher cosine and lower errors are better. The confidence intervals above resample characters after averaging seeds.

Seed	Cosine	Static norm	Nearby relative error	Moving L_2 error
1	$+0.01747$	$+0.00002$	-0.01056	-0.00550
2	-0.00012	$+0.00120$	-0.00820	-0.00217
3	$+0.01265$	-0.00325	-0.00628	-0.00392

Post hoc three-seed artist-vertex stability. After both later seeds finished, averaging all three paired seeds within each source gave moving-vertex errors 0.06581 joint and 0.06336 blocked, with zero motion at 0.08266. The source-equal paired joint-minus- blocked difference is $+0.00245$ (95% source-bootstrap interval $[-0.00422, +0.00815]$), with joint lower on $4/12$ sources. The three per-seed mean differences are $+0.00029$, $+0.00335$, and $+0.00371$; none shows an observed joint advantage. The predeclared eight-source input-compatible subset has a three-seed difference $+0.00608$ (interval $[+0.00030, +0.01148]$, joint lower on $2/8$ sources). This later-seed exercise is a stability check on the same test IDs after seed 1 inspection, not a second prospective replication or an independent test of a reversal. It still uses artist vertices, not generated-rig render motion. The full source-level and per-seed rows are in `fresh_generated_rig_test12_artist_vertex_3seed_posthoc_summary.json`.

Post hoc artist-asset rendering check. We also rendered the seed-1 predictor's offsets on the original artist layers and artist meshes for these same 12 sources. This uses the artist-authored motion as the target and leaves the candidate's layers and rest render identical to the target, isolating learned offsets from Stage 1 and automatic meshing. Source-mean rest-subtracted error is 1.033 for predicted motion versus 1.000 for static; the predicted-minus-static paired source-bootstrap 95% interval is $[-0.122, +0.232]$, with prediction lower on $8/12$ sources. The predeclared input-compatible eight-source subset scores 1.105 versus 1.000 static, lower on $6/8$. Two sources have notably large residuals (1.861 and 1.531), so the point mean is sensitive to source heterogeneity. This diagnostic was added after the seed-1 artist-vertex result and is not the prospective generated-layer endpoint. It shows that a lower vertex-error point estimate than zero motion does not guarantee improved source-mean rendered motion even with artist textures and meshes. Exact rows and the renderer are retained internally in `fresh_generated_rig_test12_artist_render_posthoc_summary.json`.

Metric protocol. Live2D-Bench evaluates a predicted ordered RGBA stack $P = \{(R_j^p, \alpha_j^p)\}$ against a ground-truth stack $G = \{(R_i^g, \alpha_i^g)\}_{i=1}^m$, where the number of predicted layers is allowed to differ from the number of ground-truth layers. RGB values are normalized to $[0, 1]$ unless stated otherwise, alpha values are continuous in $[0, 1]$, and $\epsilon = 10^{-8}$ avoids zero denominators.

Full-image metrics. For composite-level scores we first alpha-composite each stack into a single RGB image and also collapse its layer alphas by max pooling,

$$\alpha^g_{\text{full}}(x, y) = \max_i \alpha_i^g(x, y), \qquad \alpha^p_{\text{full}}(x, y) = \max_j \alpha_j^p(x, y). \tag{3}$$

The full-image alpha score (Eq. 4) is the soft IoU over the continuous alpha maps,

$$\text{IoU}^{\text{full}}_\alpha = \frac{\sum_{x,y} \min(\alpha^g_{\text{full}}(x, y), \alpha^p_{\text{full}}(x, y))}{\sum_{x,y} \max(\alpha^g_{\text{full}}(x, y), \alpha^p_{\text{full}}(x, y)) + \epsilon}. \tag{4}$$

The reported full-image RGB-L1 (Eq. 5) uses the ground-truth foreground support, $M_g(x, y) = \mathbf{1}[\alpha^g_{\text{full}}(x, y) > 0]$,

$$\text{RGB-L1}^{\text{full}} = \frac{\sum_{x,y} M_g(x, y) \frac{1}{3} \sum_{c \in \{R,G,B\}} |I^p_c(x, y) - I^g_c(x, y)|}{\sum_{x,y} M_g(x, y) + \epsilon}. \tag{5}$$

PSNR and SSIM are computed on the entire composited RGB image; PSNR uses the standard 255-range MSE, and SSIM uses a 7×7 uniform window (`scikit-image` defaults, i.e. no Gaussian weighting) and averages the three color channels. LPIPS is computed on the full composited RGB images with the standard pretrained AlexNet LPIPS features.

Hungarian layer matching. For every ground-truth layer i and predicted layer j, we compute a pairwise alpha IoU and foreground RGB error (Eqs. 6 and 7),

$$a_{ij} = \frac{\sum_{x,y} \min(\alpha_i^g(x,y), \alpha_j^p(x,y))}{\sum_{x,y} \max(\alpha_i^g(x,y), \alpha_j^p(x,y)) + \epsilon}, \tag{6}$$

$$r_{ij} = \frac{\sum_{x,y} \mathbf{1}[\alpha_i^g(x,y) > 0] \, \frac{1}{3} \sum_{c \in \{R,G,B\}} |R_{j,c}^p(x,y) - R_{i,c}^g(x,y)|}{\sum_{x,y} \mathbf{1}[\alpha_i^g(x,y) > 0] + \epsilon}. \tag{7}$$

The assignment cost combines the two (Eq. 8),

$$c_{ij} = \frac{r_{ij} + (1 - a_{ij})}{2} \in [0,1]. \tag{8}$$

We then solve the rectangular linear assignment problem of Eq. 9,

$$M = \underset{\substack{M' \subseteq \{1,\dots,m\} \times \{1,\dots,n\} \\ |M'| = k, \text{ one-to-one}}}{\arg\min} \sum_{(i,j) \in M'} c_{ij}, \qquad k = \min(m,n). \tag{9}$$

The implementation uses the standard Hungarian / LAPJV solver and thus directly supports rectangular cost matrices. If the two stacks have different sizes, $u = \max(m,n) - k$ layers are unmatched. For matched-only metrics we average over the k assigned pairs. For penalized metrics, unmatched layers receive the phantom values $a = 0$, $r = 1$, and $c = 1$, and the denominator is $\max(m,n)$:

$$\bar{a}_{\text{pen}} = \frac{\sum_{(i,j) \in M} a_{ij}}{\max(m,n)}, \tag{10}$$

$$\bar{r}_{\text{pen}} = \frac{\sum_{(i,j) \in M} r_{ij} + u}{\max(m,n)}, \tag{11}$$

$$\bar{c}_{\text{pen}} = \frac{\sum_{(i,j) \in M} c_{ij} + u}{\max(m,n)}. \tag{12}$$

Matched-layer LPIPS uses the same matching M: each matched RGBA layer is alpha-composited over a white background, LPIPS is computed for the pair, and the result is averaged over matched pairs.

Layer-order consistency. To score whether depth order is preserved, we sort the matched pairs by ground-truth layer index and read the corresponding predicted indices $p_1, \dots, p_k$. The number of inversions

$$I = \#\{(s,t) : s < t, \; p_s > p_t\} \tag{13}$$

is exactly the number of crossing edges in the bipartite matching diagram. The matched order score is a Kendall-style normalized agreement,

$$O_{\text{match}} = 1 - \frac{I}{\binom{k}{2}}, \qquad k \geq 2, \tag{14}$$

so a perfectly ordered stack scores 1 and a completely reversed stack scores 0. For $k < 2$ there is no ordering decision to make, so the score is left undefined and the case is excluded from the aggregate rather than being credited with a perfect score. The penalized order score also accounts for missing or extra layers,

$$O_{\text{pen}} = O_{\text{match}} \frac{\min(m,n)}{\max(m,n)}, \qquad \text{unmatched-rate} = \frac{u}{\max(m,n)}. \tag{15}$$

When aggregating the fair order number across benchmark cases, we use pair-weighted averaging with weight $\binom{k}{2}$ so that dense layer stacks contribute in proportion to the number of ordering decisions.

Coverage and mask metrics. Cov-MAE measures whether the predicted stack has the same per-pixel layer coverage count as the ground truth. With threshold $\tau = 0.5$,

$$\mathrm{cov}(x, y) = \left| \{k : \alpha^{(k)}(x, y) > \tau\} \right|, \tag{16}$$

$$\text{Cov-MAE} = \frac{1}{HW} \sum_{x,y} \left| \mathrm{cov}_g(x, y) - \mathrm{cov}_p(x, y) \right|. \tag{17}$$

We additionally track the signed decomposition

$$\mathrm{over} = \frac{1}{HW} \sum_{x,y} \max(\mathrm{cov}_p - \mathrm{cov}_g, 0), \tag{18}$$

$$\mathrm{under} = \frac{1}{HW} \sum_{x,y} \max(\mathrm{cov}_g - \mathrm{cov}_p, 0), \tag{19}$$

whose sum equals Cov-MAE. Over counts excess predicted layer coverage, whereas under counts missing predicted coverage.

Mask Dice is computed on the same Hungarian pairs for compatibility with layer-decomposition baselines. For pair (i, j) we use the squared soft Dice form with smooth term $s = 1$,

$$\mathrm{Dice}_{ij} = \frac{2 \sum_{x,y} \alpha_i^g(x, y) \alpha_j^p(x, y) + s}{\sum_{x,y} (\alpha_i^g(x, y))^2 + \sum_{x,y} (\alpha_j^p(x, y))^2 + s}, \tag{20}$$

$$\mathrm{MaskDiceLoss}_{ij} = 1 - \mathrm{Dice}_{ij}. \tag{21}$$

The matched Mask Dice loss averages this loss over M; the penalized version gives each unmatched layer loss 1 and divides by $\max(m, n)$.

F Stage 1 Qualitative Results

This section presents additional qualitative results for our Stage 1 layer decomposition. Visual comparisons against existing baselines are detailed in Figures 11 and 12. Additionally, Figures 13 to 16 highlight our multi-layer generation capabilities, specifically illustrating our method's flexibility in supporting layer separation with controllable granularity.

G Superseded Stage 2 Design: Autoregressive Mesh-Token Stream

Status of this appendix. The Stage 2 model of the main paper (§V.4) is a joint continuous-regression network: it predicts per-vertex displacements for all layers of a character in one forward pass and never discretises geometry. This appendix documents an *earlier* Stage 2 formulation that we no longer use, in which mesh connectivity and keypose offsets were serialised into a single autoregressive token stream over quantised coordinates. We retain it for two reasons: the layer-to-mesh construction it describes is still the basis of the content-conforming mesh used throughout the paper, and its compression statistics may be of independent interest to readers building token-based mesh models. Two caveats must be read with everything below. First, the accuracy figures reported for this design (cosines above 0.99) were obtained under *teacher forcing*, with the ground-truth prefix fed back at every decoding step; they measure next-token accuracy, not the quality of a generated rig, and they are not comparable with the true-generation numbers in the main text. Second, no number in this appendix supports any claim in the main paper.

Operator alphabet. For a single connected component of a 2D triangle mesh with V vertices and F faces, the tokeniser emits a byte stream over the alphabet $\Sigma = \{\langle \texttt{COMP} \rangle, \langle \texttt{C} \rangle, \langle \texttt{L} \rangle, \langle \texttt{R} \rangle, \langle \texttt{E} \rangle, \langle \texttt{S} \rangle, \langle \texttt{B} \rangle\}$ mixed with quantised vertex coordinates from the alphabet $\{0, \ldots, Q-1\}$. The semantics follow EdgeBreaker Rossignac (1999) as adapted to neural mesh generation by EdgeRunner Tang et al. (2025):

- $\langle \texttt{COMP} \rangle$ starts a component and is followed by six integers, the three quantised (x, y) pairs of the seed triangle's three vertices.



Figure 11: **Stage 1 layer decomposition against the segmentation and layering baselines**, sheet 1. Eight illustrations, one per row. Each method is given the same flattened illustration; within a method's column the left half is the re-composited result and the right half is its layer stack exploded. Left to right: ground-truth artist layers, ours, See-through, Qwen-Image-Layered, SAM, Marigold-depth. Note that Marigold, having only depth to work with, splits by depth band rather than by part, and SAM segments visible pixels without completing what a layer hides.

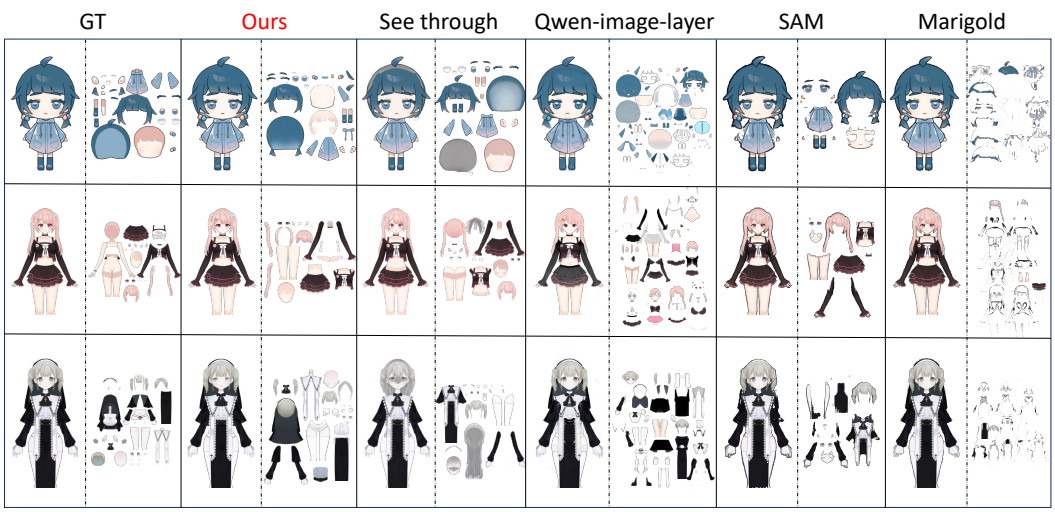

Figure 12: **Stage 1 layer decomposition against the baselines**, sheet 2. Eight further illustrations; format exactly as Fig. 11.

- ⟨C⟩ (*capture*) declares that the next face has a previously-unseen vertex as its third corner; it is followed by two integers giving its quantised coordinates.
- ⟨L⟩ and ⟨R⟩ (*previous-twin left / right*) declare that the third corner is already visited and that its respective side edge is already on the frontier; they carry no payload.
- ⟨E⟩ (*end*) closes a face whose two side edges are both already on the frontier; no payload.
- ⟨S⟩ (*split*) declares that the third corner is a visited vertex whose side edges are not on the frontier; it is followed by one integer, the compact identifier of that vertex.
- ⟨B⟩ (*boundary advance*) consumes a frontier edge that has no face behind it; no payload.

Encoder and decoder maintain the same frontier stack (a stack of edges with an opposite-corner annotation), so they always agree on which gate is being processed and can round-trip exactly.

Layer-to-mesh. At training time the input mesh is the artist-authored Live2D mesh, whose vertex count ranges from a few tens to several hundreds and whose face count grows roughly twice as fast. At inference on a novel illustration (no artist mesh), we synthesise a base mesh from the layer's alpha channel in six steps: (1) threshold the alpha channel at $8/255$ to obtain a silhouette; (2) morphologically dilate the silhouette by approximately 8 px to recover the artist's safety margin used by Live2D Cubism for mesh padding; (3) walk the dilated boundary counter-clockwise and sample $n_b = 24$ uniform points; (4) inside the silhouette, draw $n_i = 8$ Poisson-disk interior points with minimum spacing 20 px; (5) run constrained Delaunay triangulation Barber et al. (1996) on the combined point set and discard triangles whose centroid lies outside the dilated silhouette; (6) quantise the resulting vertices to $Q = 128$ bins per axis and emit them in the order required by the operator alphabet above. The full procedure together with the EdgeRunner per-face traversal that follows it is illustrated across multiple character / layer combinations in Figs. 20, 21, 22, and 23 below; the mesh deformation across keyposes that the resulting tokens drive is shown across three multi-character composites in Figs. 17, 18, and 19.

Across the nine rows of Figs. 17 and 19 the superseded design was exercised on seven characters (elf-girl, fox-girl, goth, pink-lolita, schoolgirl, military maid, VTuber) and nine distinct layer types, and its EdgeRunner compression ratio stayed in the band 0.507 to 0.523 throughout, so the token savings were independent of character identity, mesh size and layer shape.

Token stream size. The total token count of a single-component mesh is $T = 1 + 6 + 3\,N_{\mathrm{C}} + N_{\mathrm{L}} + N_{\mathrm{R}} + N_{\mathrm{E}} + 2\,N_{\mathrm{S}} + N_{\mathrm{B}}$, where N_x counts the number of ⟨x⟩ operators emitted. The 6 accounts for the seed-triangle vertex payload, each ⟨C⟩ adds 1 operator plus 2 coordinate integers, and each ⟨S⟩ adds 1 operator plus 1 identifier integer. Crucially, every interior vertex of the mesh is emitted by exactly one

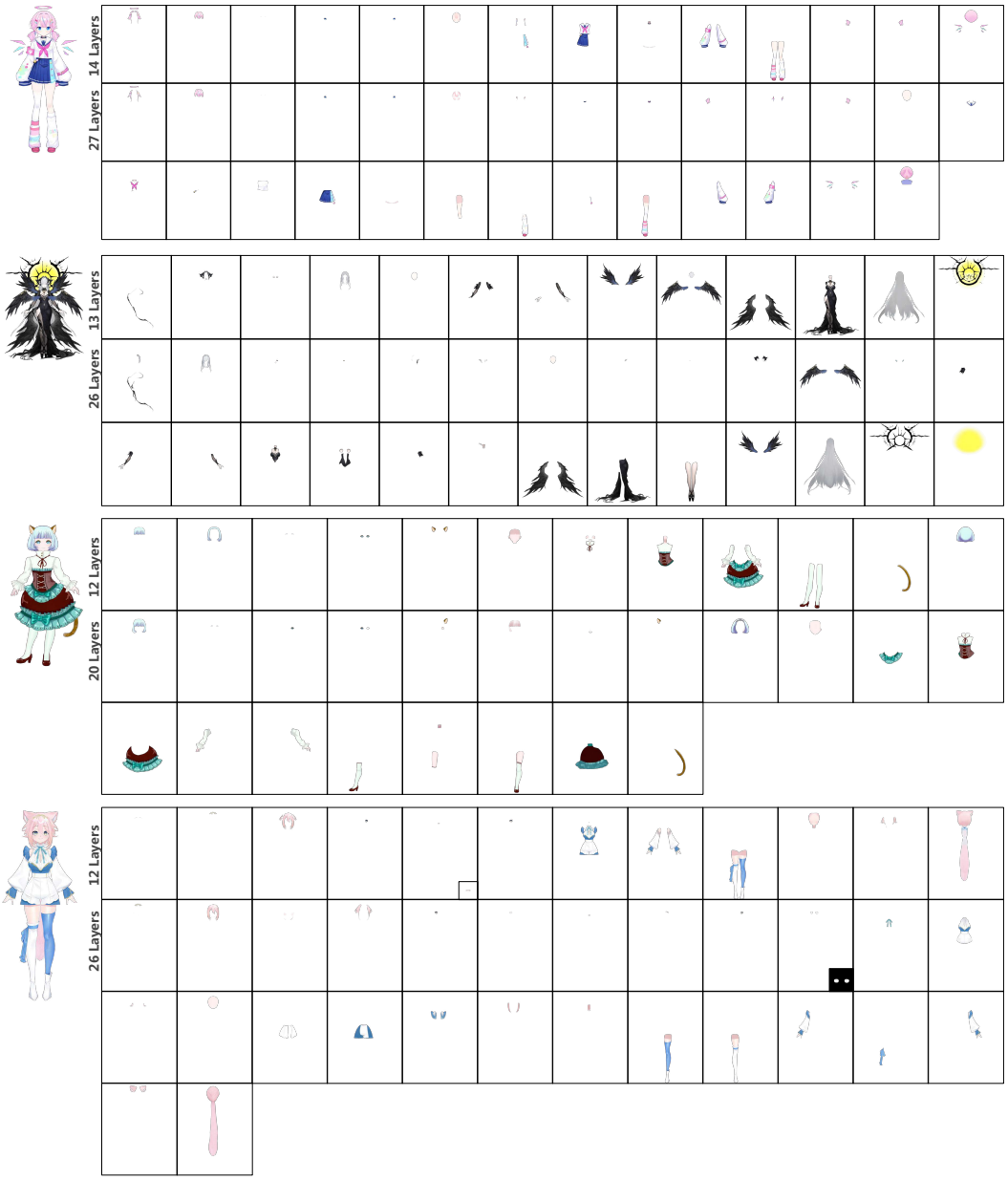

Figure 13: The reconstructed characters (left) are shown alongside their decomposed layer textures (right) under different specified layer counts. Our approach naturally adapts to free layer numbers, supporting both coarse and fine-grained separation.

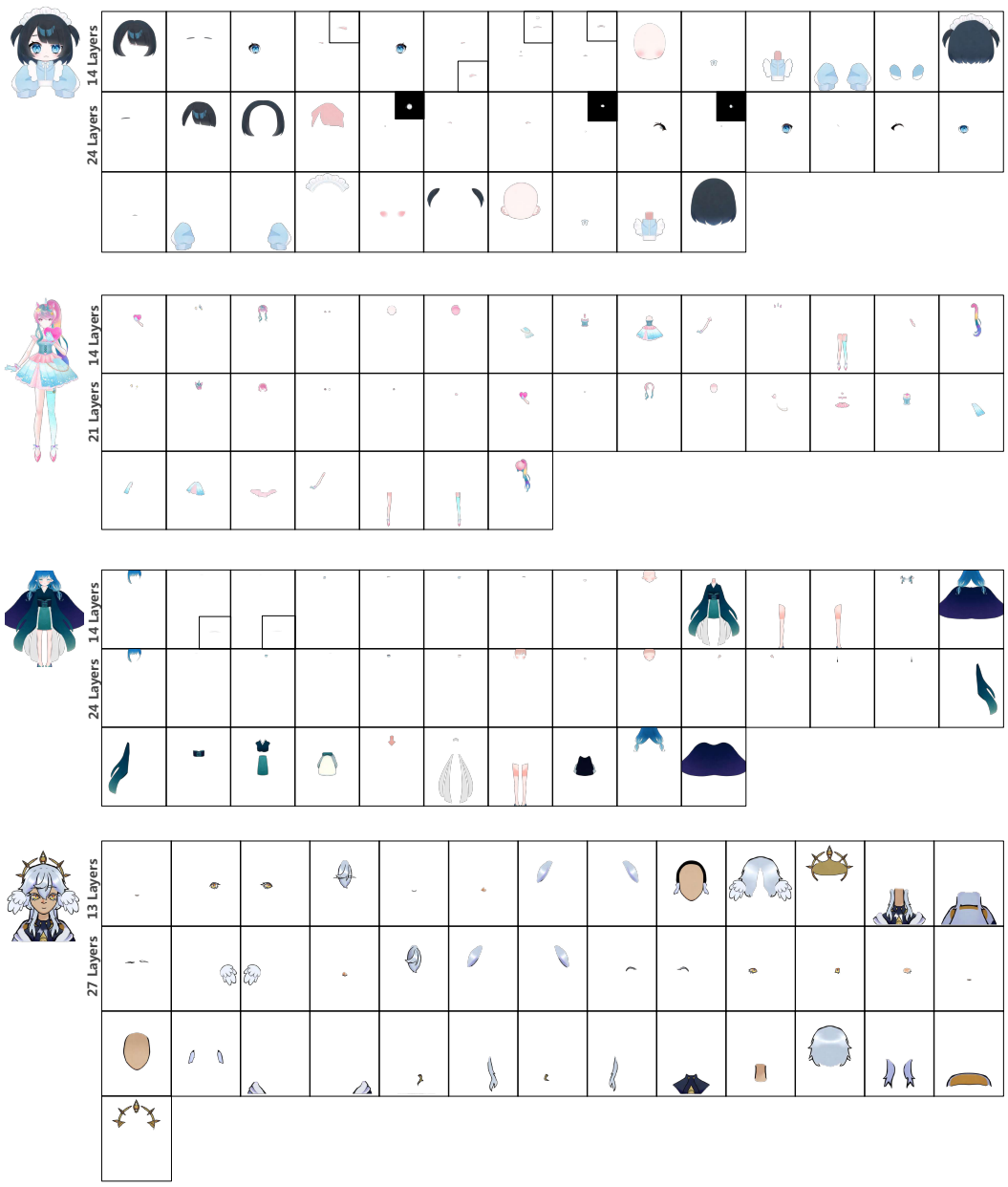

Figure 14: The reconstructed characters (left) are shown alongside their decomposed layer textures (right) under different specified layer counts. Our approach naturally adapts to free layer numbers, supporting both coarse and fine-grained separation.

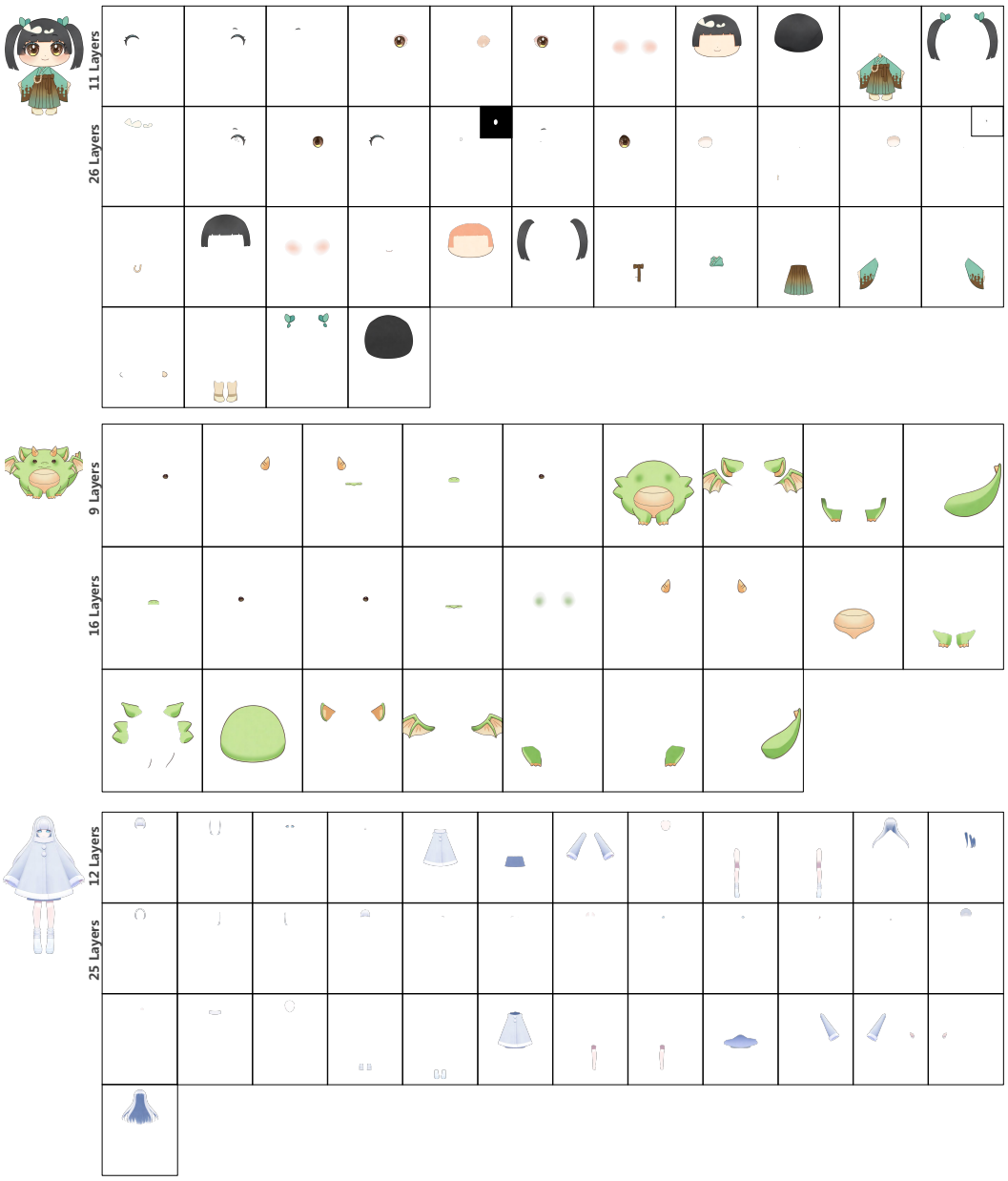

Figure 15: The reconstructed characters (left) are shown alongside their decomposed layer textures (right) under different specified layer counts. Our approach naturally adapts to free layer numbers, supporting both coarse and fine-grained separation.

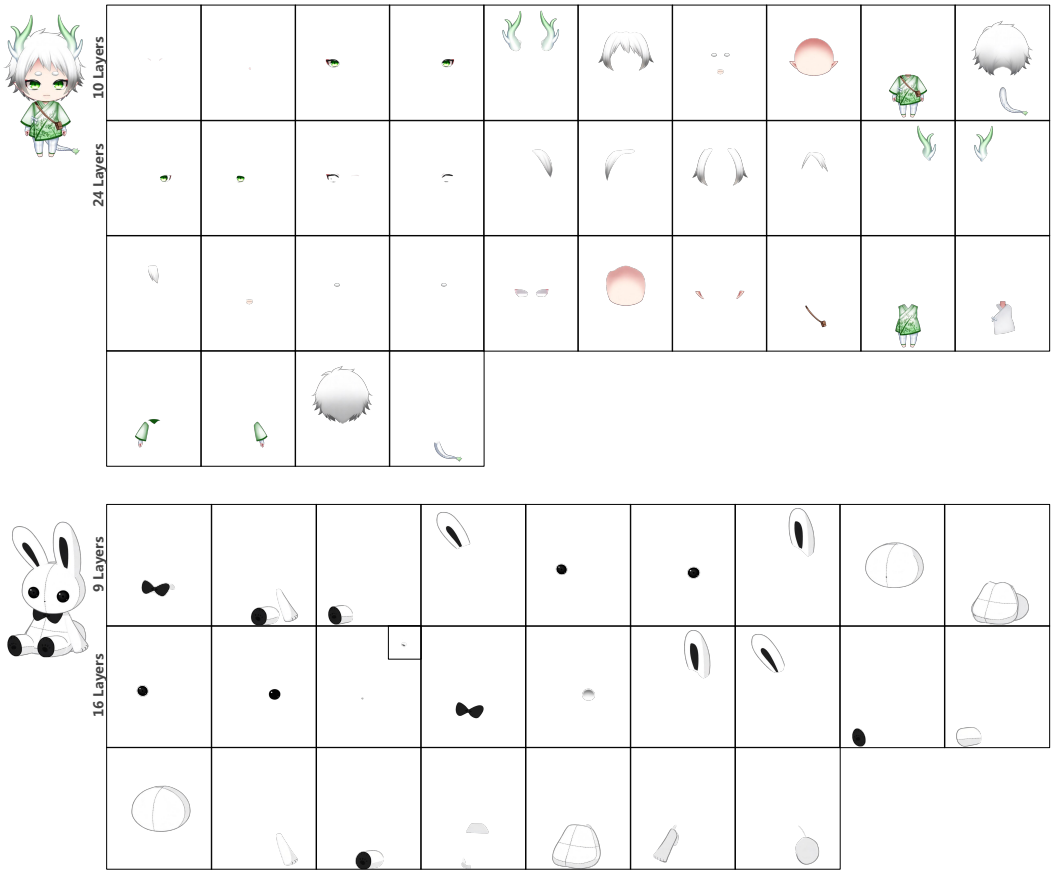

Figure 16: The reconstructed characters (left) are shown alongside their decomposed layer textures (right) under different specified layer counts. Our approach naturally adapts to free layer numbers, supporting both coarse and fine-grained separation.

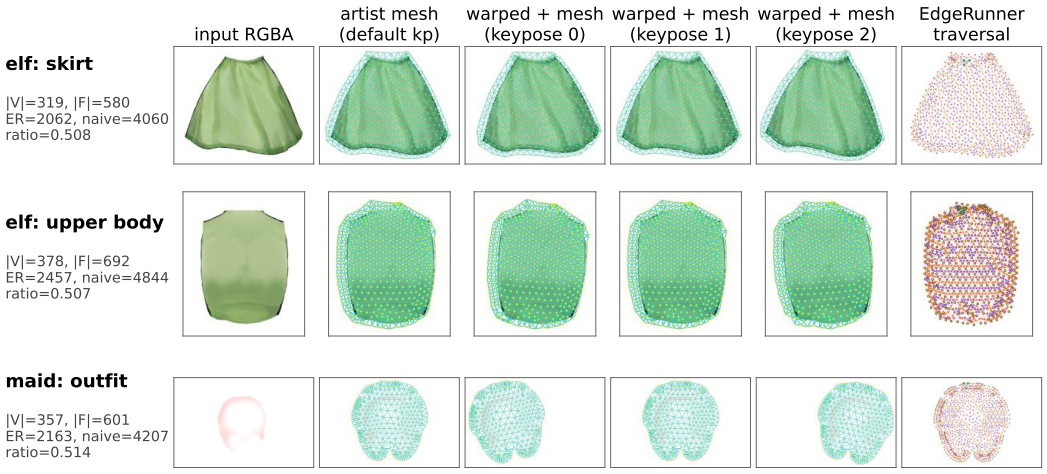

Figure 17: **Multi-character demonstration on garment layers** (superseded design). Three characters, all on full-body garment layers. Columns: the raw RGBA layer; the artist mesh at the default keypose; the same mesh re-warped to keyposes 0/1/2 along the listed parameter axis; and the EdgeRunner per-face traversal emitted by the superseded Stage 2 (green seed triangle, orange capture arrows, purple split arrows, boundary operators trailing). Per-row token statistics are printed beside each row label.

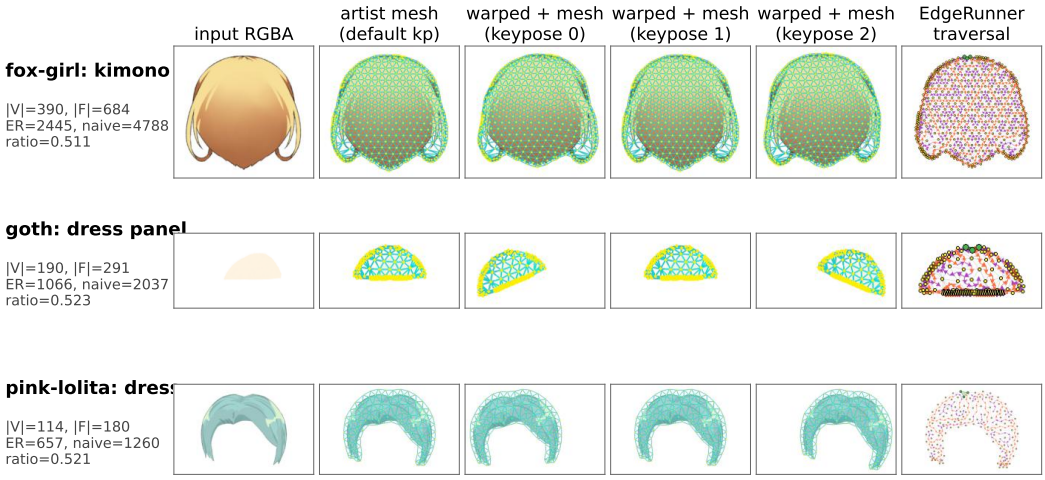

Figure 18: **Multi-character demonstration on body-component layers**, formatted identically to Fig. 17. Three rows show a fox-girl's kimono front panel, a goth-styled character's dress panel, and a pink-lolita's dress; together they span three new characters and three new layer shapes.

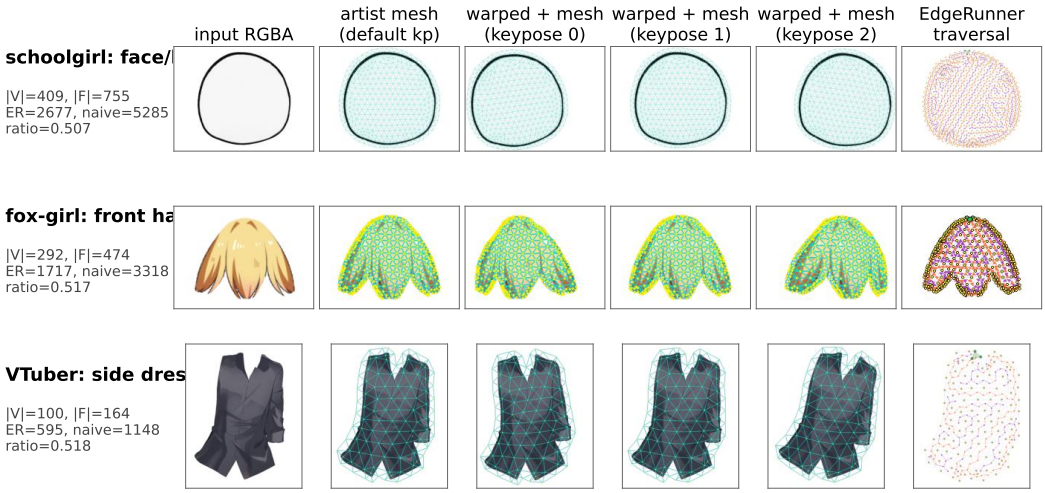

Figure 19: **Multi-character demonstration on head / hair layers** (superseded design), formatted as Fig. 17: a schoolgirl's full face/hair shape (the largest layer in the demo, $|V| = 409$, $|F| = 755$), a fox-girl's front hair, and a VTuber's side dress.

$\langle$C$\rangle$, so $N_{\mathrm{C}} = V - 3$ for a fully connected component, and every boundary edge contributes exactly one $\langle$B$\rangle$. A naive baseline that simply writes each triangle as $\langle$F$\rangle$ followed by 3 (x, y) pairs uses $7F$ tokens. The compression ratio between EdgeRunner and naive is thus a near-constant function of the boundary-to-face ratio and is largely independent of mesh size.

Compression statistics on real Live2D data. We measured EdgeRunner and naive token counts on every layer of the Tsumugi character (100 layers, $|V| = 4156$, $|F| = 5941$). The aggregate compression is reported in Tab. 12. EdgeRunner produces 22,297 tokens against 41,587 for naive, a 0.536 ratio (46.4% savings), with 3.75 tokens per face on average versus 7.0 for naive. The breakdown of the operator alphabet is dominated by $\langle$B$\rangle$ (boundary advances, 50.9%) and $\langle$C$\rangle$ (interior captures, 32.3%), with $\langle$S$\rangle$ (splits, 16.8%) accounting for the rest. The $\langle$P$\rangle$ operator was never triggered on this character, consistent with the under-2% P-fraction we observe across the full 662-character training set.

Table 12: Mesh-tokenisation statistics on a representative Live2D character (Tsumugi, 100 layers). EdgeRunner tokens correspond to our operator-plus-payload count; the naive baseline encodes each triangle as a fresh $\langle$F$\rangle$ marker plus three quantised (x, y) pairs.

Quantity	Value
Layers (with non-empty mesh)	100
Total vertices $\lvert V\rvert$	4,156
Total faces $\lvert F\rvert$	5,941
EdgeRunner tokens	22,297
Naive XYZ tokens	41,587
Compression ratio (EdgeRunner / Naive)	0.536
Bytes saved relative to naive	46.4%
Average EdgeRunner tokens per face	3.75
Average Naive tokens per face	7.00
Operator $\langle$B$\rangle$ (boundary)	50.9%
Operator $\langle$C$\rangle$ (capture)	32.3%
Operator $\langle$S$\rangle$ (split)	16.8%
Operator $\langle$P$\rangle$ (pop / backtrack)	0.0%

Cross-character examples. We exercise the full layer-to-mesh-to-token pipeline on four character / layer combinations drawn from *four different characters* spanning distinct anime aesthetics, summarised in Tab. 13 and shown panel-by-panel in Figs. 20, 21, 22, and 23. Each figure is a single 2×4 grid in which panels 1 to 7 step through the layer-to-mesh procedure and panel 8 overlays the EdgeRunner per-face traversal that the autoregressive model emits on the same artist mesh. The seed triangle is drawn in green, $\langle$C$\rangle$ (capture) arrows in orange, $\langle$S$\rangle$ (split) arrows in purple, and $\langle$B$\rangle$ (boundary advance) operators are folded into the stream tail. Across all four cases the compression ratio against the naive baseline stays in the narrow band 0.512 to 0.523, which matches the constant savings predicted by the per-face cost analysis and confirms that the savings are independent of character identity, mesh size, and layer shape.

Table 13: Per-figure token statistics for the four cross-character demos.

Character	Layer description	$\lvert V\rvert$	$\lvert F\rvert$	EdgeRunner	Naive	Ratio
elf-girl	side hair	249	432	1,548	3,024	0.512
fox-girl (kimono)	front hair	292	474	1,717	3,318	0.517
goth (with harness)	half-circle bust	190	291	1,066	2,037	0.523
pink lolita	front dress	147	244	874	1,708	0.512

Putting it all together. In this superseded design, the full per-layer record stitched together (i) a mesh block tokenised as above, (ii) a single $\langle$SCALE$\rangle$ token whose hidden state was read by a magnitude-regression head, and (iii) two keypose blocks each emitting $|V|$ quantised $(\Delta x, \Delta y)$ offsets. For a typical 60-layer character, the token stream compressed from approximately 90 K

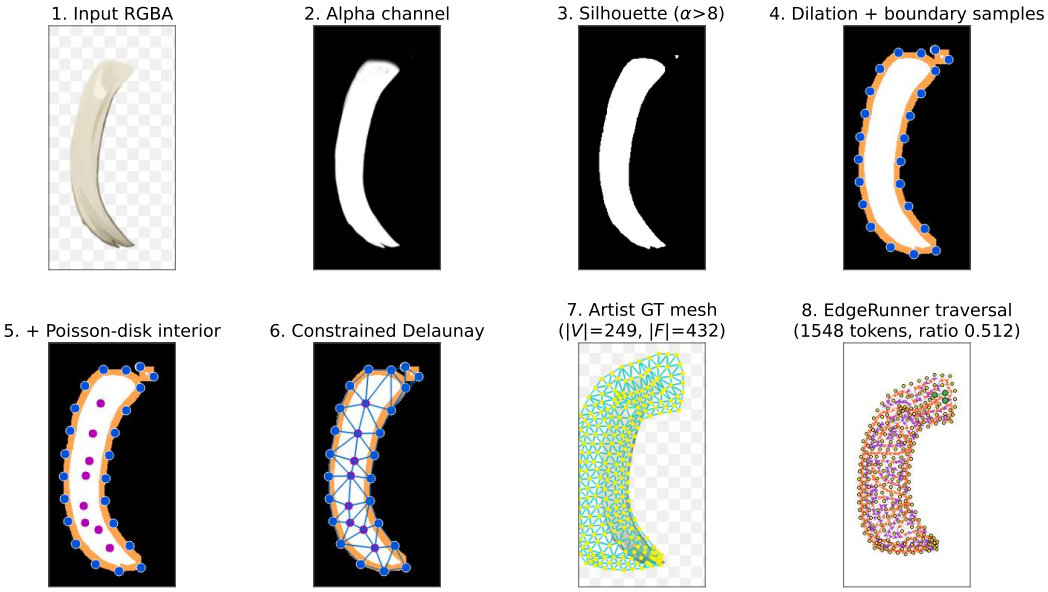

Figure 20: Layer-to-mesh pipeline and EdgeRunner traversal on a long flowing side hair layer of an elf-girl character ($|V| = 249$, $|F| = 432$). Panels 1 to 7 step through the auto-mesh procedure; panel 8 shows the per-face traversal that produces a $1{,}548$-token EdgeRunner stream, $0.512\times$ the $3{,}024$-token naive baseline.

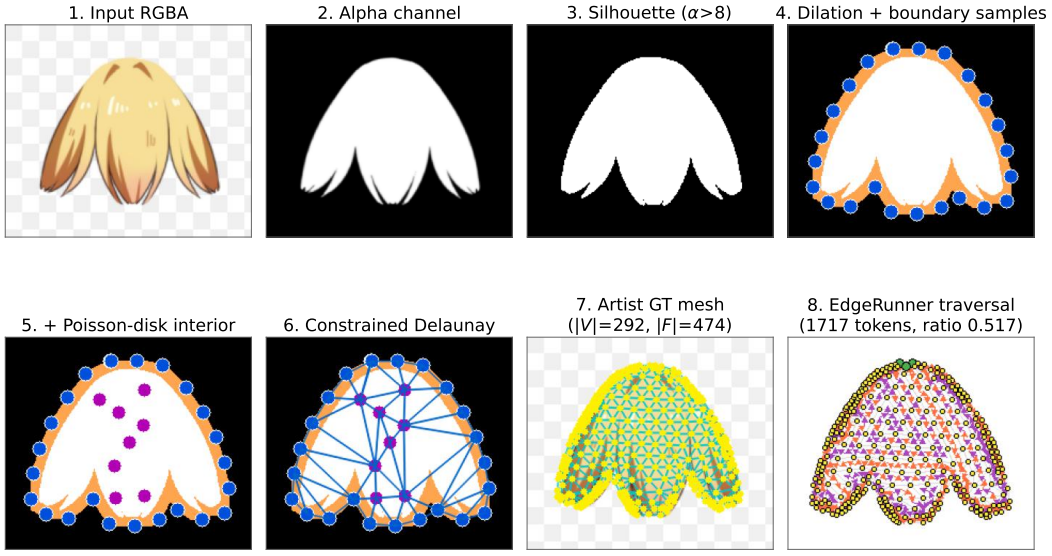

Figure 21: Layer-to-mesh pipeline and EdgeRunner traversal on a fox-girl character's front hair / fringe layer ($|V| = 292$, $|F| = 474$). Although the shape and aesthetic differ markedly from Fig. 20, the resulting compression ratio (0.517) is essentially identical.

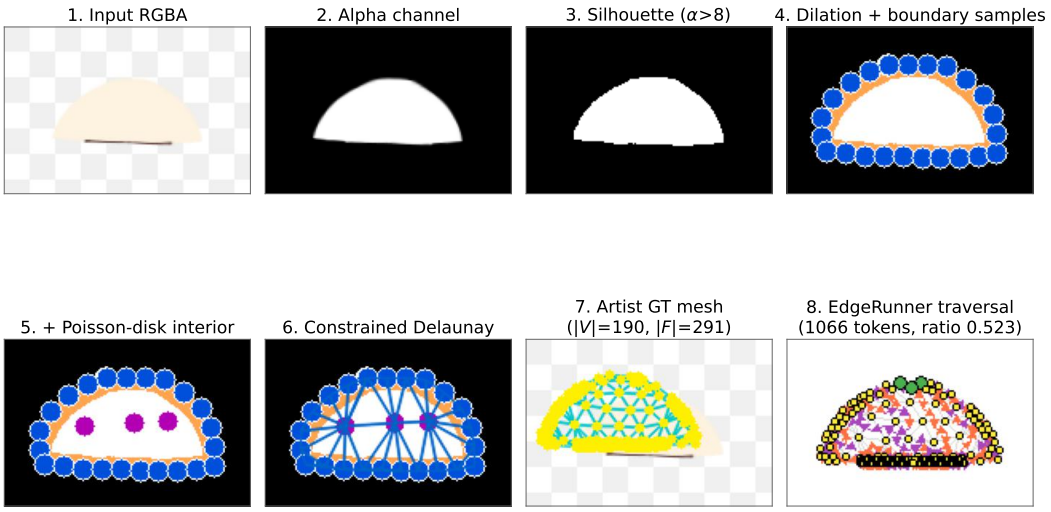

Figure 22: Layer-to-mesh pipeline and EdgeRunner traversal on a goth-styled character's half-circle bust layer ($|V| = 190$, $|F| = 291$). The smaller layer produces a $1{,}066$-token EdgeRunner stream, ratio 0.523 against the $2{,}037$-token naive baseline.

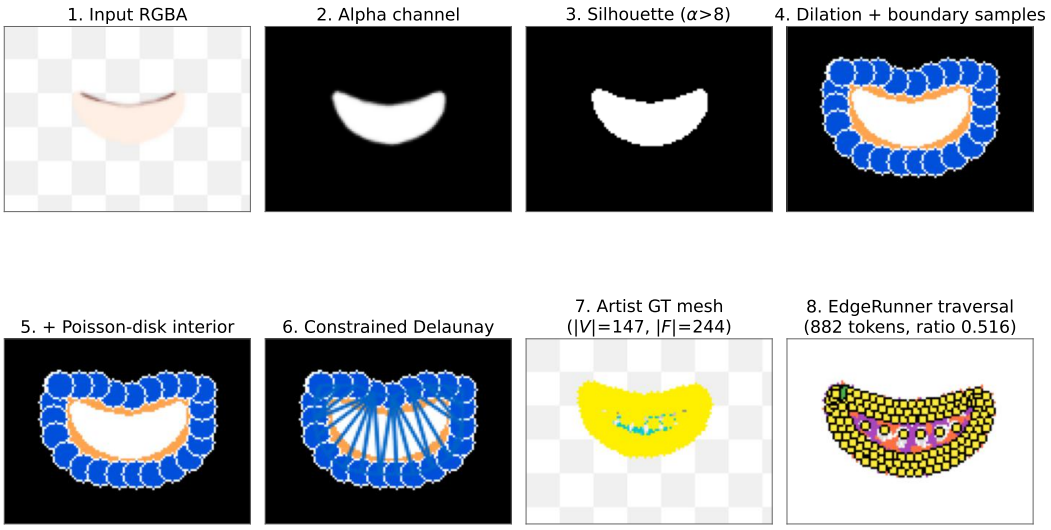

Figure 23: Layer-to-mesh pipeline and EdgeRunner traversal on a pink-lolita character's front dress layer ($|V| = 147$, $|F| = 244$). On a smaller layer the absolute token count drops to 874, but the ratio against the naive baseline stays at 0.512, demonstrating that compression savings are mesh-size invariant.

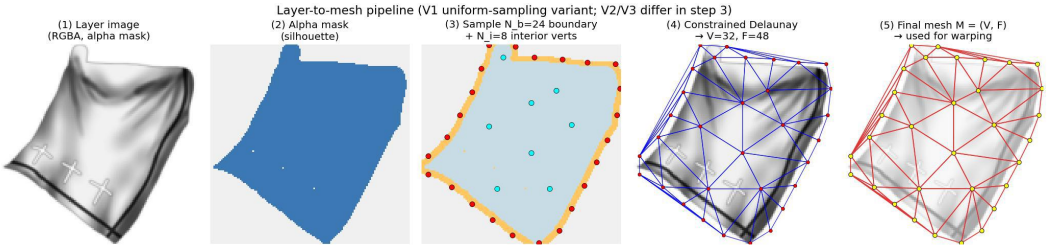

Figure 24: **Layer-to-mesh pipeline, V1 uniform-sampling variant** (superseded design), on a real square garment layer. Five steps: input RGBA and alpha mask; silhouette at $\alpha > 8$; dilation of about 8 px (orange ring) with $n_b = 24$ uniform boundary points and $n_i = 8$ Poisson-disk interior points; constrained Delaunay giving $|V| = 32$ and $|F| \approx 48$; and the final mesh in canvas-normalised $[-1, 1]^2$. Note that the shipped construction of §V.4 differs at several of these steps; the variants are listed in the text.

JSON tokens to between 18 K and 22 K tokens, which is the regime in which a from-scratch 12 M-parameter causal Transformer trained stably. The joint regression model that replaced it needs no such serialisation: one vertex is one token, and a 60-layer character is a single sequence of a few thousand tokens processed in one non-causal pass.

H Displacement-Magnitude Distribution

Stage 2 factors every displacement into a bounded shape term and a log-scale term (Eqs. 30 and 31) because the magnitude distribution is extremely heavy-tailed. Measuring the per-layer peak displacement $a = \max_v \|\Delta v\|_\infty$ over the full training corpus, 80% of (layer, parameter, keypose) records concentrate at $a < 0.05$ of canvas extent, with a long tail past $a > 0.4$; the density is only readable on a log scale. Two consequences follow. First, regressing raw displacements would let the tail dominate the loss, since a single large head turn contributes two orders of magnitude more gradient than a typical eyelid motion; normalising by a and predicting $\log a$ separately makes the two regimes contribute comparably, so that a $2\times$ magnitude error costs the same whether the true motion is 0.002 or 0.2. Second, the same statistic explains why a uniform quantisation of displacements is a poor representation for this domain: a $Q = 128$-level grid over $[-1, 1]$ would collapse the entire small-motion mode into roughly the central six bins, which is what motivated the per-record normalisation used by the superseded design of Appendix G and what the continuous formulation avoids outright.

I Stage 2 Inference and Live2D Runtime Interpolation

Stage 2 inference is one forward pass per (parameter, keypose) pair. All layers of the character are concatenated into a single sequence of vertex tokens (Eq. 28); the direction head emits $u_{i,j} = \tanh(\cdot) \in [-1, 1]^2$ and the magnitude head emits $s_{i,j}$, and the keypose offset is their product $\widehat{\Delta v}_{i,j} = u_{i,j} \exp(s_{i,j})$ (Eq. 31). There is no autoregressive rollout and therefore no exposure-bias gap between training and inference: the model is evaluated in exactly the mode in which it is trained, which is why every Stage 2 number in the main text is a true-generation number. Sweeping the modelled parameters over their keyposes produces the full displacement table of Eq. 23. At runtime the parameter slider drives per-vertex linear interpolation between the two bracketing keyposes (Eq. 33), followed by per-triangle rasterisation with premultiplied alpha and no face culling. The full deformation runs at 30 fps in-browser.

J Layer-to-Mesh Deployment Study (superseded Stage 2)

V1 to V5 differ only at the sampling step: V1 uses a random interior, V2 Poisson-disk sampling Bridson (2007), V3 curvature-aware boundary sampling, V4 V3 plus Lloyd CVT Lloyd (1982), and V5 Shewchuk's Triangle Shewchuk (1996) at `pq30`. None of these is the shipped construction: §V.4

masks at $\alpha > 4$ rather than $\alpha > 8$, dilates by 3 px rather than 8, samples the interior on a jittered lattice rather than by Poisson disk, and uses an unconstrained Delaunay triangulation.

Auto-mesh + retraining. This study was run with the superseded autoregressive Stage 2 of Appendix G, under a protocol that supplied the mesh prefix; its cosines are therefore *not* comparable with the true-generation numbers of the main text and are reported only for the relative ordering of mesh-construction algorithms, which is what it was designed to establish. At training time we use artist meshes; at deployment time the input is a Stage 1 PSD layer (pixel + alpha only), so we automatically generate a base mesh: Suzuki-Abe Suzuki et al. (1985) contour, n_b uniform boundary points, n_i Poisson-disk interior, constrained Delaunay Barber et al. (1996), drop center-outside triangles. We additionally study (i) dilation-px sweep $\{0, 8, 16, 24\}$ to mimic the artist's $\sim$17 px safety margin, and (ii) algorithm sweep V1 to V5. Tab. 14 reports 50-char OOD results.

Findings. (i) Inference-only auto-mesh: cosine drops uniformly to ~ 0.91 across all 5 algorithms (V1 to V5 within 0.001); the gap is *vertex-distribution* mismatch, not geometric quality. (ii) *Retraining* on auto-meshes (paired GT offsets via inverse-distance weighting) closes it: V1/V2/V3 retrain all reach cos ≥ 0.9998 within 5 epochs; V3 curvature-aware additionally gives the lowest mag (2.91) and best RMSE in pixel space (1.99 px). *Honest caveat:* retrained cos 0.9999 is not directly comparable to artist-baseline 0.997; V3 auto-mesh has V_u=32 uniform vertices vs V=30 to 100 sharp / non-uniform artist verts, so the task is simpler. Pixel-domain RMSE is more honest: V1 retrain 2.15 px vs artist 0.99 px (still sub-3 px); PCK@10 0.977 vs 0.993.

Table 14: Layer-to-mesh deployment study on 50-char OOD, run with the *superseded* autoregressive Stage 2 (Appendix G) under a mesh-prefix-supplied protocol. Read the relative ordering of mesh algorithms, not the absolute cosines: these are not true-generation numbers and are not comparable with Tab. 24.

Mesh source	vert. cos↑	mag→1	α-err↓	Reference
Artist mesh (training distribution)	**0.997**	**7.73**	**0.320**	,
Inference-only auto-mesh:				
V1 uniform, dilate 0	0.916	22.53	0.506	,
V1 uniform, dilate 16	0.916	19.39	0.482	,
V1 uniform (random interior)	0.908	17.81	0.485	legacy
V2 Poisson-disk interior	0.908	19.27	0.491	Bridson (2007)
V3 curvature-aware boundary	0.908	**15.31**	0.494	Live2D-style
V4 V3 + Lloyd CVT (4 iter)	0.908	18.25	0.510	Lloyd (1982)
V5 Triangle `pq30`	0.907	17.84	0.473	Shewchuk (1996)
Retrained on auto-mesh (deployment-aware, ours):				
V1 uniform retrain	**0.9999**	3.84	0.328	this work
V2 Poisson retrain	**0.9998**	3.19	0.336	this work
V3 curvature retrain	**0.9999**	**2.91**	0.324	this work
V1+barycentric interp	**0.9999**	3.34	**0.256**	this work

Multi-component layers. The single-contour formulation above (largest external contour) silently drops disconnected components that occur routinely in in-the-wild decompositions: e.g. a side-hair layer split into a main lock plus a detached tail, or a garment layer separated by an occluding arm. We therefore extend the auto-mesh to *all* external contours above a minimum-area threshold: each component contributes its own arc-length-sampled boundary loop to the PSLG, boundary-point budget is allocated proportional to component area, and constrained Delaunay is run once over the union before dropping centroid-outside triangles. This keeps every visible part meshed, while remaining identical to the single-contour procedure when a layer has one component. We note for the record that we initially justified this design by claiming a bounding-box grid wastes most of its vertices on transparent pixels; measuring it on 46 validation layers refuted that claim (on-content

vertex fraction 0.376 for the grid against 0.368 for ours, silhouette IoU 0.426 for both), because a layer's own alpha bounding box is mostly filled. The defensible advantage is cost at equal coverage: 83 vertices and 112 faces per layer against 112 and 182, which is 26% fewer tokens for the same opaque coverage, with the budget concentrated along contours rather than spread uniformly.

K Animation Results Under Many Conditions

The numbers in §W.3 and §W.4 each compress a whole condition into one scalar. This appendix shows the same conditions as pictures, because some of them are only convincing that way and one of them (capacity) is more convincing that way. Every cell below is a raw render of a real rig file through the same viewer at an *absolute* parameter value, so any cell can be checked against the released rig; nothing is hand-picked or per-example fixed.

Mesh representation (Fig. 25). Quad grid against content-conforming triangles, with the artist's own rig driven to the same parameter value as a third column. Characters are matched *by name* across the two rig sets because their internal ids differ, and the per-character cosine of each condition is printed beside the row. The visual difference between the two meshes is small at this scale, which is consistent with the 0.7676 against 0.7542 scalar gap; what the figure adds is that neither mesh produces a qualitatively different kind of error, so the choice between them is a matter of token cost and accuracy rather than of failure mode.

Parameter vocabulary (Fig. 26). The same in-the-wild illustrations and the same Stage-1 layers, animated once by the 8-parameter model and once by the 24-parameter model, followed by three parameters only the larger vocabulary has. The first two columns are the visual form of the controlled comparison in Tab. 29: on the parameter both models share, they are hard to tell apart, which is what a 0.031 cosine difference should look like. The last three columns are the reason to prefer the larger vocabulary at all, since gaze, breathing and hair sway have no counterpart in the smaller one.

Layer source (Fig. 27). One frozen 24-parameter animation model animating two decompositions of each illustration, at rest and at a large turn. Showing rest as well as the pose matters: several of the differences that look like animation errors are already present at rest, which localises them in Stage 1 rather than Stage 2. This is the qualitative companion to the quantitative three-way comparison of Appendix S, where the artist's own layers form a third condition and bound what Stage 2 can achieve.

Capacity (Fig. 28). Seven models from 1.3 M to 1.0 B parameters, trained on identical data with an identical objective and schedule, all animating the same illustration at the same parameter value. The columns are nearly indistinguishable. We consider this the most persuasive form of the capacity result, because a reader who distrusts our metric can still see that a $770\times$ increase in parameters does not visibly change the output. Note that the 1.0 B column is not empty: that run diverged in training (validation cosine -0.085), and what it produces is a small, nearly uniform displacement, which is what a collapsed model looks like rather than a crash.

L An Atlas of Every Parameter's Displacement Field

A per-parameter table of cosines (Tab. 18) says which parameters were learned but not *what* they learned. Reviewers of an earlier version raised exactly this: the animation looked single-axis and the distribution of motion types was never shown. This appendix answers it directly. One row per parameter, one column per character, and in each cell the predicted displacement field drawn as vector arrows over the rest mesh, read straight out of the released rig files with no inference run.

The figure is meant to be read in two directions. Reading *down* a column checks semantic correctness without trusting any metric: the head parameters move the head group and leave the feet alone, breathing moves the torso, the hair parameters move only hair, gaze moves two iris layers of a few hundred pixels and nothing else. Reading *across* a row checks consistency: the same parameter should behave the same way on characters with different layer counts, proportions and art styles, and it does. The parameters the model failed to learn are included and marked in red rather than omitted, and their fields look the way a failure should look, either near-zero or pointing somewhere unrelated.

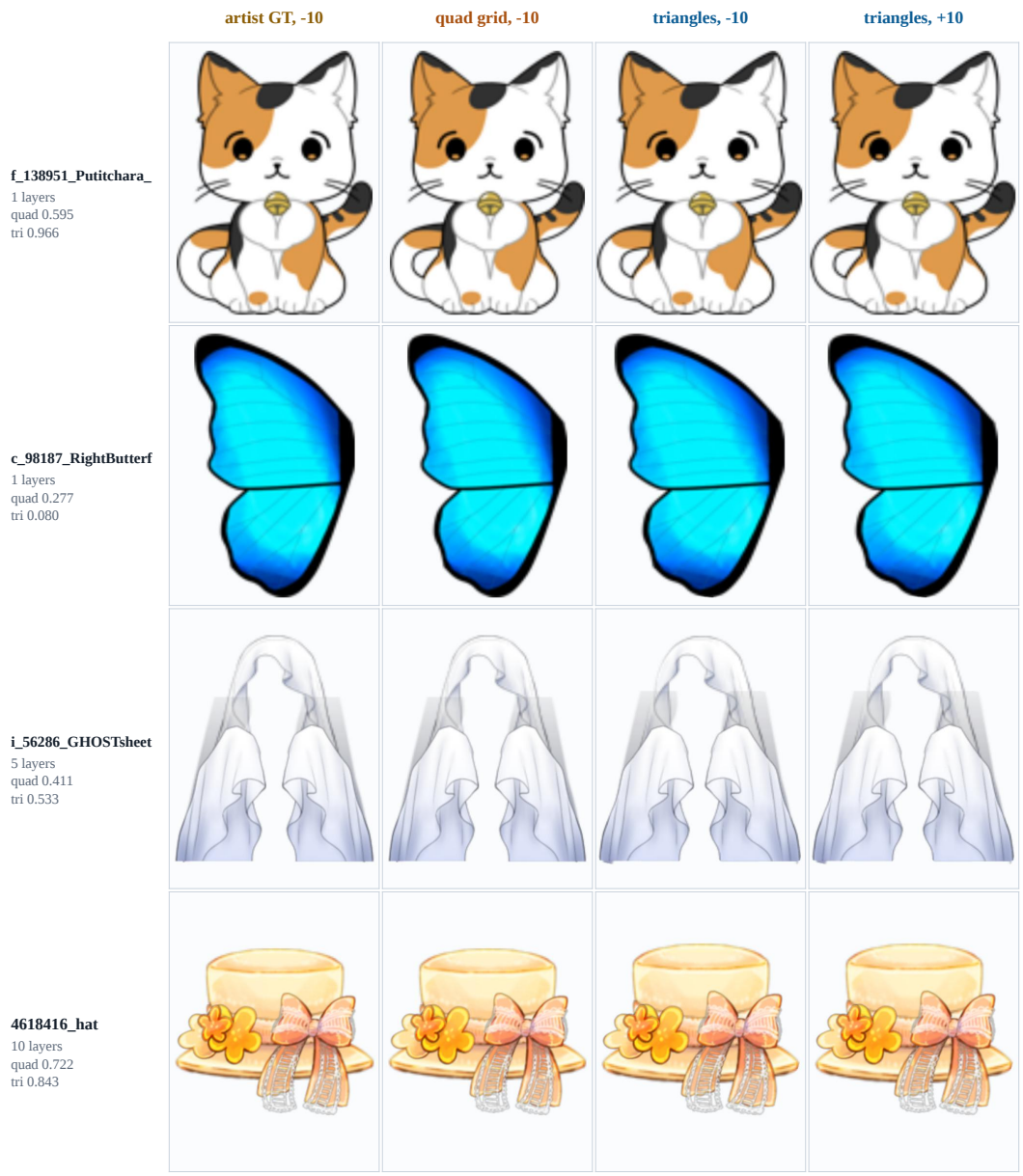

Figure 25: **Mesh representation, same character and pose.** Artist ground truth, the prediction on a quad grid, and the prediction on content-conforming triangles, at `ParamAngleX` $= +30$. Layer count and each condition's per-character direction cosine are printed beside the row. Characters are matched by name across the two rig sets.

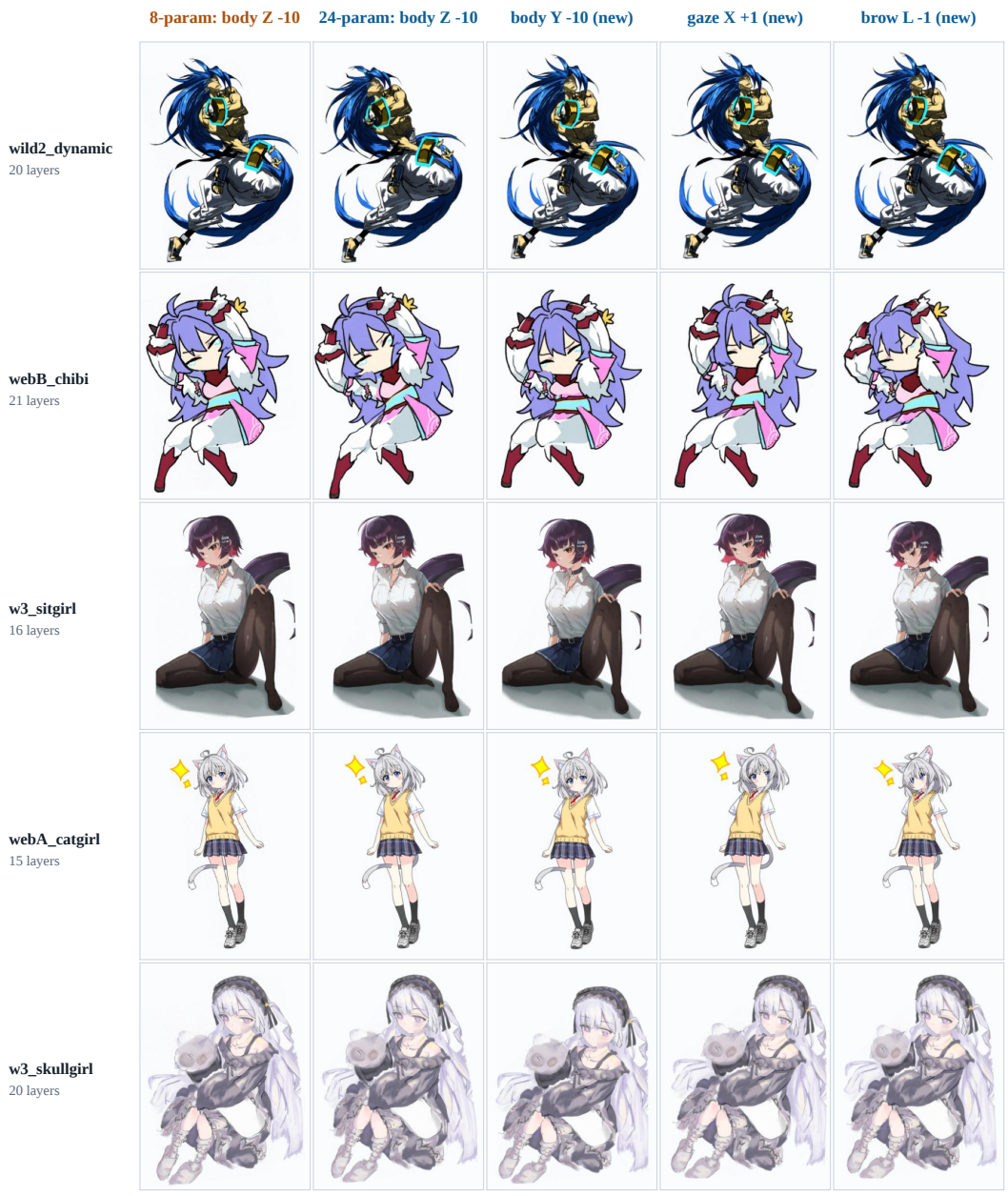

Figure 26: 8**-parameter against** 24**-parameter model.** Same illustrations, same Stage-1 layers, only the animation model differs. Columns 1–2 are the shared parameter `ParamAngleX` under each model; columns 3–5 are parameters only the 24-parameter model has (gaze, breathing, front-hair sway).

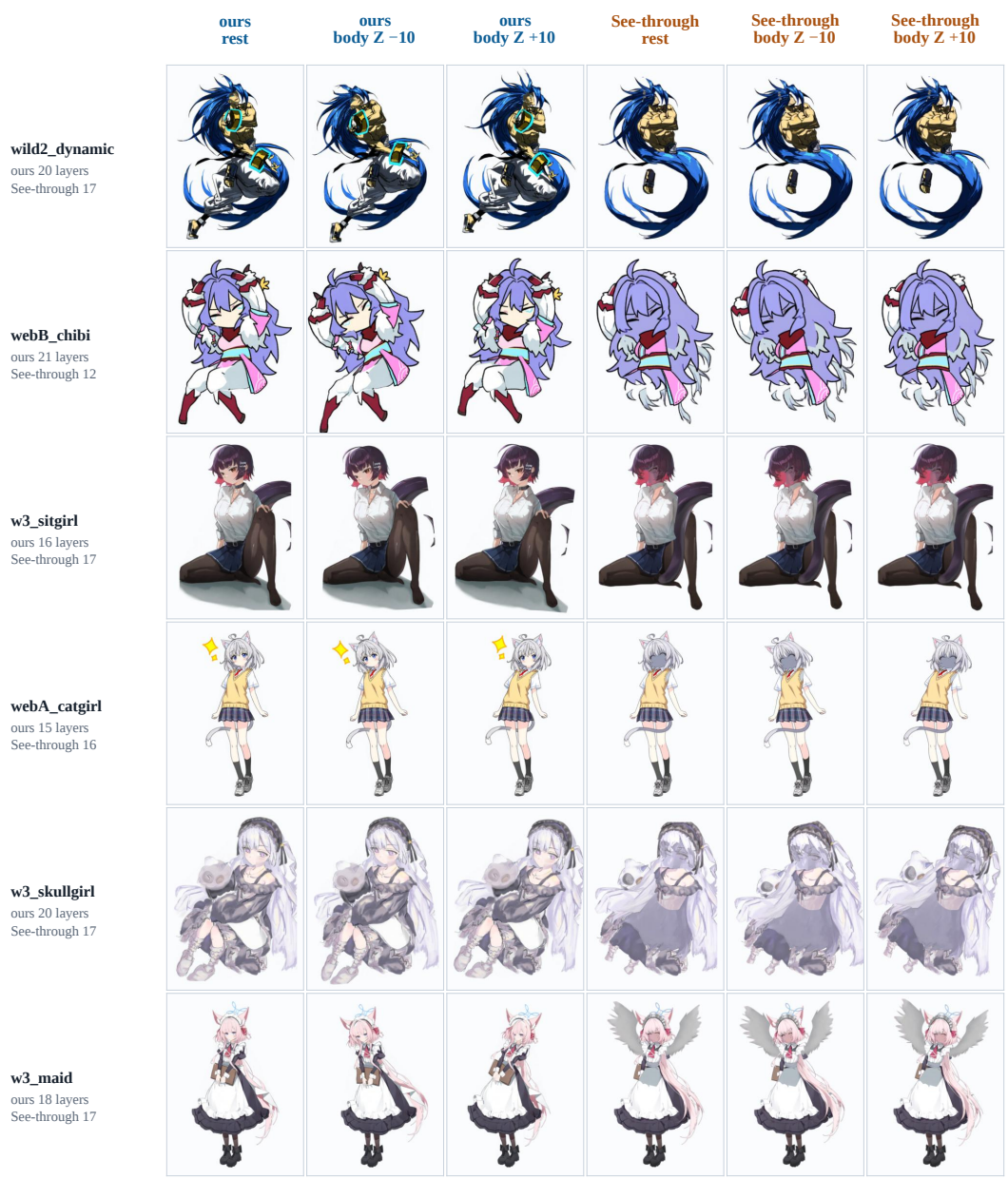

Figure 27: **Layer source is the only variable.** One frozen 24-parameter model animates our Stage-1 decomposition and the See-through decomposition of the same illustration, each shown at rest and at `ParamAngleX` $=+30$. Layer counts are printed beside the row. Differences already visible in the rest columns are decomposition errors, not animation errors.



Figure 28: **Capacity is not the bottleneck.** Every value is a row of Tab. 26 section B: six models on identical data, objective and schedule across a $112\times$ parameter range. (a) direction cosine never improves; the band is the measured same-config seed spread of 0.024, and the 1.0 B run diverged so it has no score to plot. (b) amplitude calibration is the one thing capacity buys. (c) the count of rigs above 0.80 agrees with (a), and the smallest model ties for the most.

Two details make the atlas honest. Each cell uses that parameter's *extreme* keypose, so no row is flattered by a small value. And the arrow scale is fixed per cell by the 99th percentile of that cell's own displacement magnitudes, so a parameter with genuinely tiny motion looks tiny rather than being renormalised into looking healthy.

M In-the-Wild Qualitative Gallery

The main text shows one sheet of in-the-wild rigs (Fig. 48). This appendix shows *every* in-the-wild character we processed, because the honest way to present a generative system is to show its whole output on a fixed input set rather than a selection. The paper's appendix has no page limit, so we use it.



Figure 29: **Displacement field of every animation parameter, sheet 1: head and body rotation.** One row per parameter, one column per character, six in-the-wild characters ordered by measured motion magnitude. Arrows are the predicted displacement at that parameter's extreme keypose, exaggerated $3\times$ and coloured blue (small) to red (large), over the rest mesh in grey. The cosine beside each row is that parameter's measured quality, coloured green above 0.70, amber between 0.40 and 0.70, red below. Note the qualitative signature of each axis: turn and body X are horizontal fields, nod and body Y vertical, and tilt Z is a rotation fan about the head.

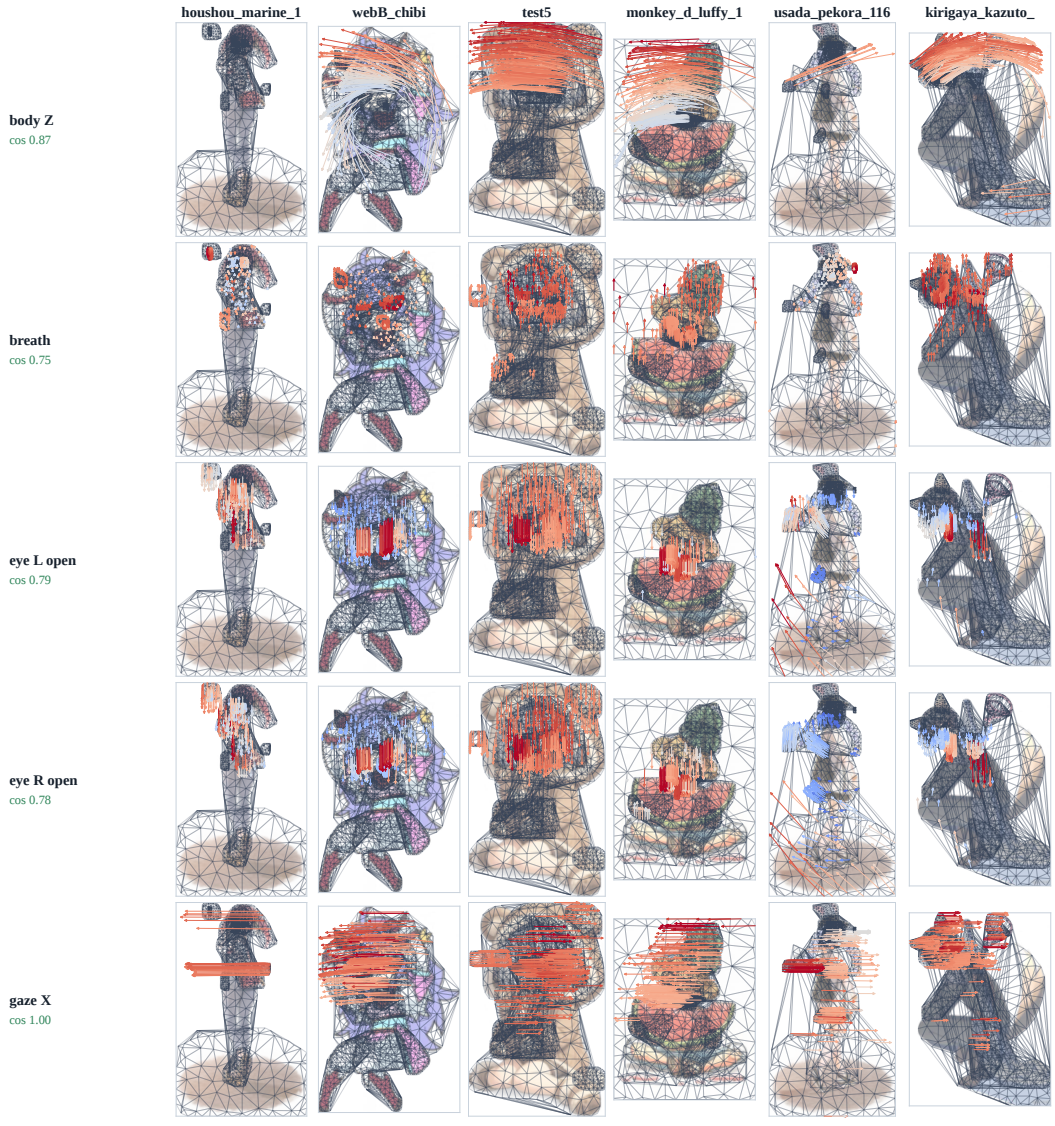

Figure 30: Sheet 2: body sway, breathing and eye opening. Format as Fig. 29. Breathing is the clearest case of correct localisation without supervision of localisation: the field concentrates on the torso and decays to nothing at the head and the feet.

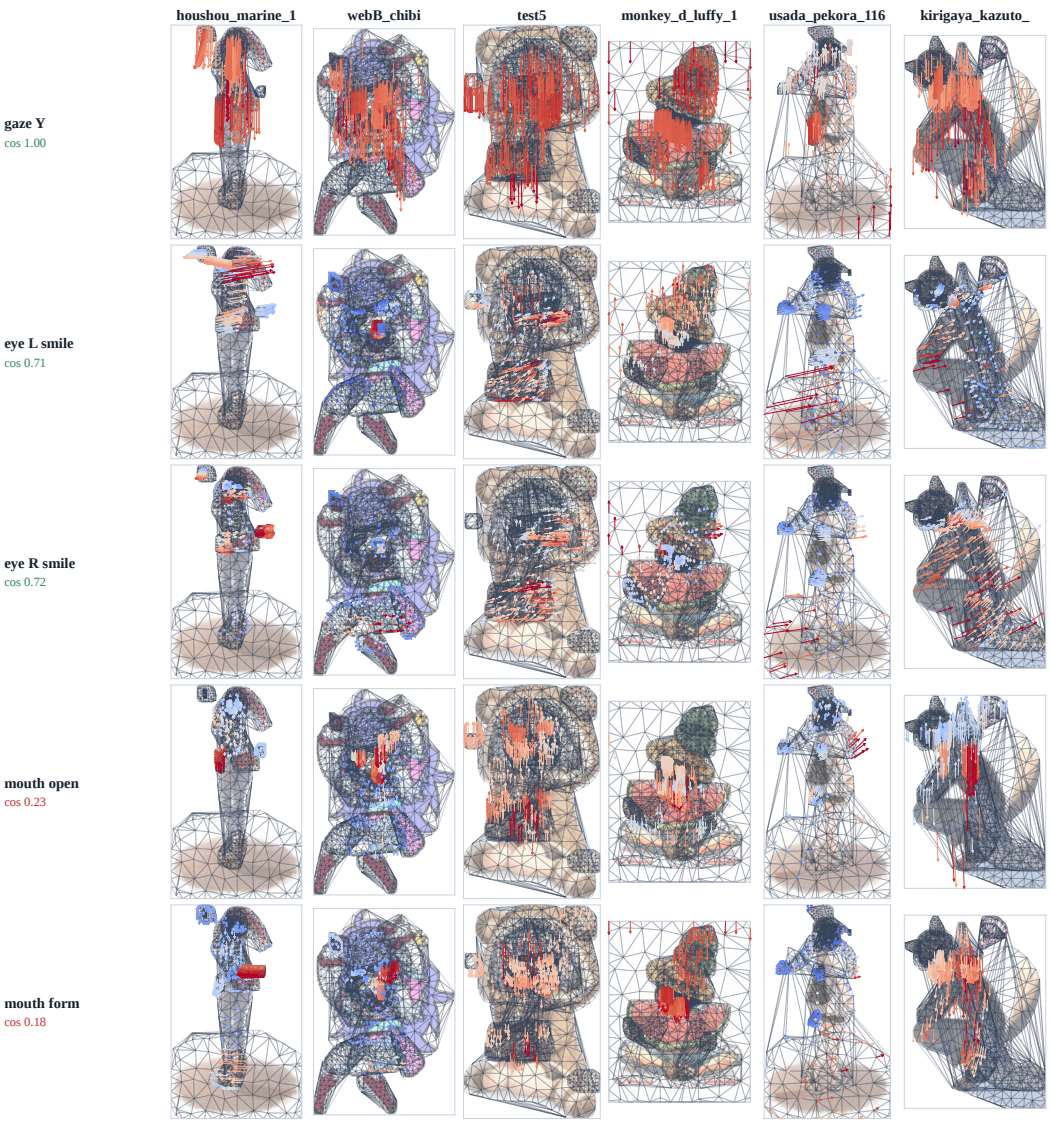

Figure 31: Sheet 3: gaze, eye smile and mouth. Format as Fig. 29. Gaze is the highest-scoring parameter in the whole vocabulary (0.997 and 0.999) and the atlas shows why the task is easy: the field is confined to two small iris layers and is close to a rigid translation. Mouth form, by contrast, is one of the failures, and its field is nearly empty.



Figure 32: Sheet 4: brow controls, which are where the vocabulary extension fails. Format as Fig. 29. Brow height is learned (0.83 and 0.84); brow form and brow angle are not (0.39, 0.34, and negative for the angles). The fields for the failed parameters are visibly unstructured, which is what a parameter trained on 17 to 48 moving-layer samples looks like.

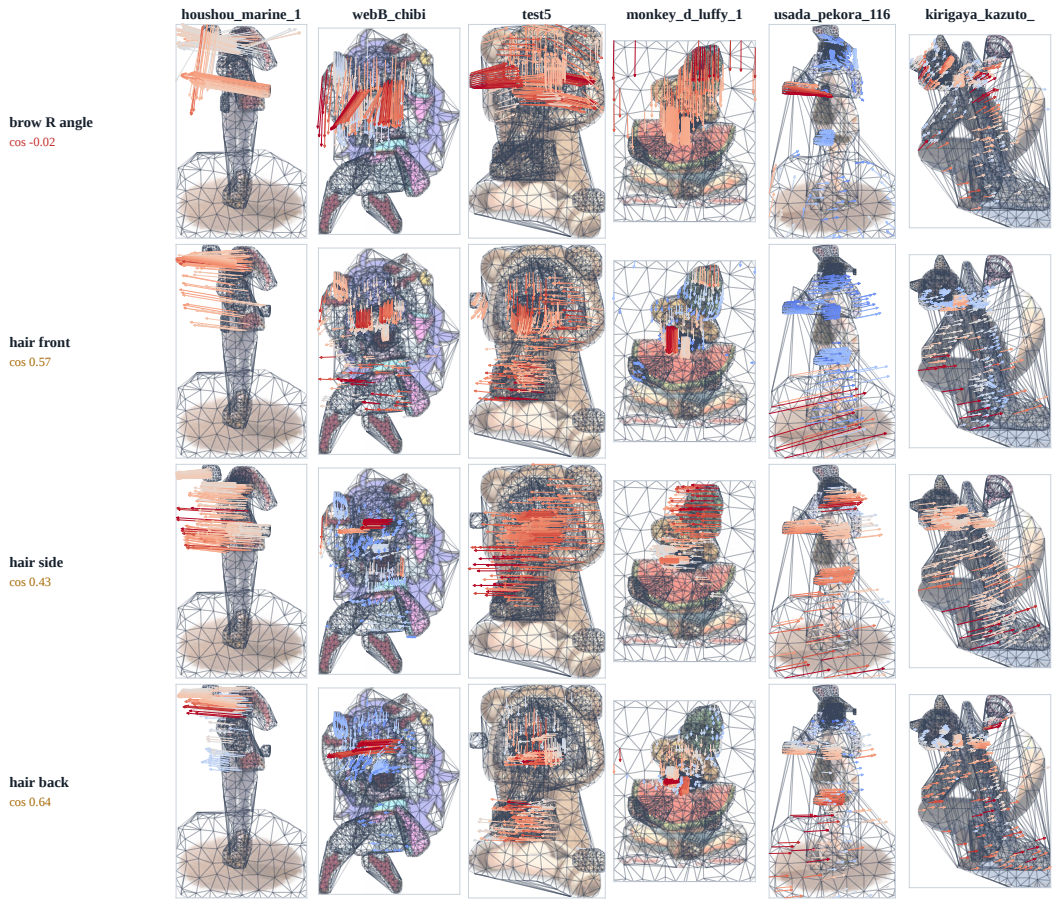

Figure 33: Sheet 5: the remaining brow angle and the three hair-sway groups. Format as Fig. 29. Front and back hair sway are marginal (0.57 and 0.64) and side hair is below the usable threshold (0.43); the atlas shows the fields are at least confined to hair layers, so the failure is one of amplitude and direction within the right support rather than of localisation.

Characters are ordered by *measured* motion magnitude, defined as mean predicted displacement at the extreme keypose divided by character span and printed beside each row. That ordering is worth reading as a result in itself: it spans 0.037 down to 0.004, roughly a factor of nine, on inputs that a human would describe as equally animatable. The low end is where amplitude compression (§X) is most visible, and it correlates with characters whose layers are large and flat, where a small displacement of many vertices is the conditional-mean answer.

Figures 34–36 continue the pose sheets. Fig. 40 shows the same rigs with the generated triangulation drawn over the artwork, which is the view a rigger inspects: every visible pixel of every layer must lie inside a triangle, or the layer tears at its silhouette during deformation. Over the 355 layers of these 18 rigs, coverage of perceptible pixels ($\alpha \geq 8$) averages 0.9996 with a median of 1.000 and a minimum of 0.9863; exactly one layer falls below 0.99. The residual is anti-aliased rim pixels at the very edge of the dilated alpha mask.

Fig. 42 then drops the renderer entirely and draws the prediction itself: the triangulation at rest, the predicted per-vertex displacement field as arrows, and the deformed triangulation, all read directly out of the released rig files. The arrows are the literal output of Eq. 31, exaggerated $3\times$ for legibility and coloured by magnitude. Three properties are visible there that a rendered frame cannot show. The field is *spatially coherent across layer boundaries* even though those layers share no connectivity, which is the joint attention doing its job. Its magnitude is *organised by depth* without depth ever being an input: head-group layers carry the long arrows, torso layers intermediate, parts resting on the

Table 15: **Stage 2 per-character wall clock**, seconds, median of nine repetitions with $[\min, \max]$. "Stage-1 output" is a real in-the-wild decomposition rather than a benchmark character. A complete rig is 13 forward passes, each covering all layers of the character simultaneously.

	small 10 lay., $T = 1052$	large 63 lay., $T = 5476$	Stage-1 output 19 lay., $T = 1873$
PNG decode + crop (CPU)	0.155	0.528	0.577
DINOv2 features (GPU)	0.115	0.614	0.183
mesh construction (CPU)	0.016	0.076	0.027
joint forwards, all 13 (GPU)	0.165	1.061	0.270
rig JSON + texture export	0.256	0.562	0.325
total per character	**0.71**	**2.84**	**1.39**
peak activation memory	189 MiB	1982 MiB	–
triangles produced	1513	7930	2637
rig JSON size	0.94 MB	4.94 MB	1.70 MB

ground almost nothing. And the triangulation is visibly *denser along contours than in flat interiors*, which is the content-conforming construction of §V.4 and the reason its vertex budget is 26% below a grid's at equal coverage.

N Assets That Predate the Method Change

This paper went through a change of Stage-2 method: an autoregressive discretised-token design (Appendix G) was replaced by the single-pass joint continuous regressor of §V.4. Most figures and tables were produced or re-measured with the current model, and the ones that were not say so in their own caption rather than in a separate index. Concretely, the token-traversal visualisations (Figs. 20–23, Tabs. 12 and 13) exist precisely to document the superseded design; the layer-to-mesh deployment study (Fig. 24, Tab. 14) was run under that design's mesh-prefix protocol and is retained only for the *relative ordering* of mesh algorithms, which is method-independent; and the failure gallery (Fig. 45) predates the change but isolates failure *categories* that persist. Everything concerning Stage 1 (decomposition, dataset statistics, mesh construction) never involved the animation model at all.

Producing this audit caught three assets that were stale without saying so, and all three are now fixed rather than annotated. The runtime discussion carried an unmeasured placeholder and is now measured (Tab. 15). The earlier image-to-video table mixed superseded and current Stage 2 protocols; we removed it because those rows cannot support a matched ranking. And the claim that a quad grid wastes "40 to 60% of its vertices on transparent pixels", which we had used to justify the mesh design, turned out to be false when measured (0.376 against 0.368 on-content vertex fraction, indistinguishable); it has been replaced by the argument the measurement does support, namely 26% fewer tokens at equal coverage. The main pipeline overview now depicts the joint Stage 2. Four image-to-video baselines would still need to be rerun under the current rendering protocol before a matched video comparison could be claimed.

O Stage 2 Runtime and Memory

Measured with a script that reproduces the reference implementation phase by phase (identical content-conforming triangulation at 110 target vertices per layer, identical 600 px canvas, identical parameter and keypose spec), with `cuda.synchronize()` bracketing every timed phase and peak memory read from `max_memory_allocated()` after resetting statistics at the start of the phase. One warm-up repetition is discarded and nine timed repetitions are kept; the median is reported with $[\min, \max]$ in Tab. 15. The card was an A800-80GB *shared with other tenants* who held 58.5 GB at the time, which is why the spread on the GPU phases is wide; the minima are the better estimate of an uncontended run. Stage 1 is a separate 20 B diffusion model and is deliberately not measured here, so no Stage-1 timing is reported or estimated.

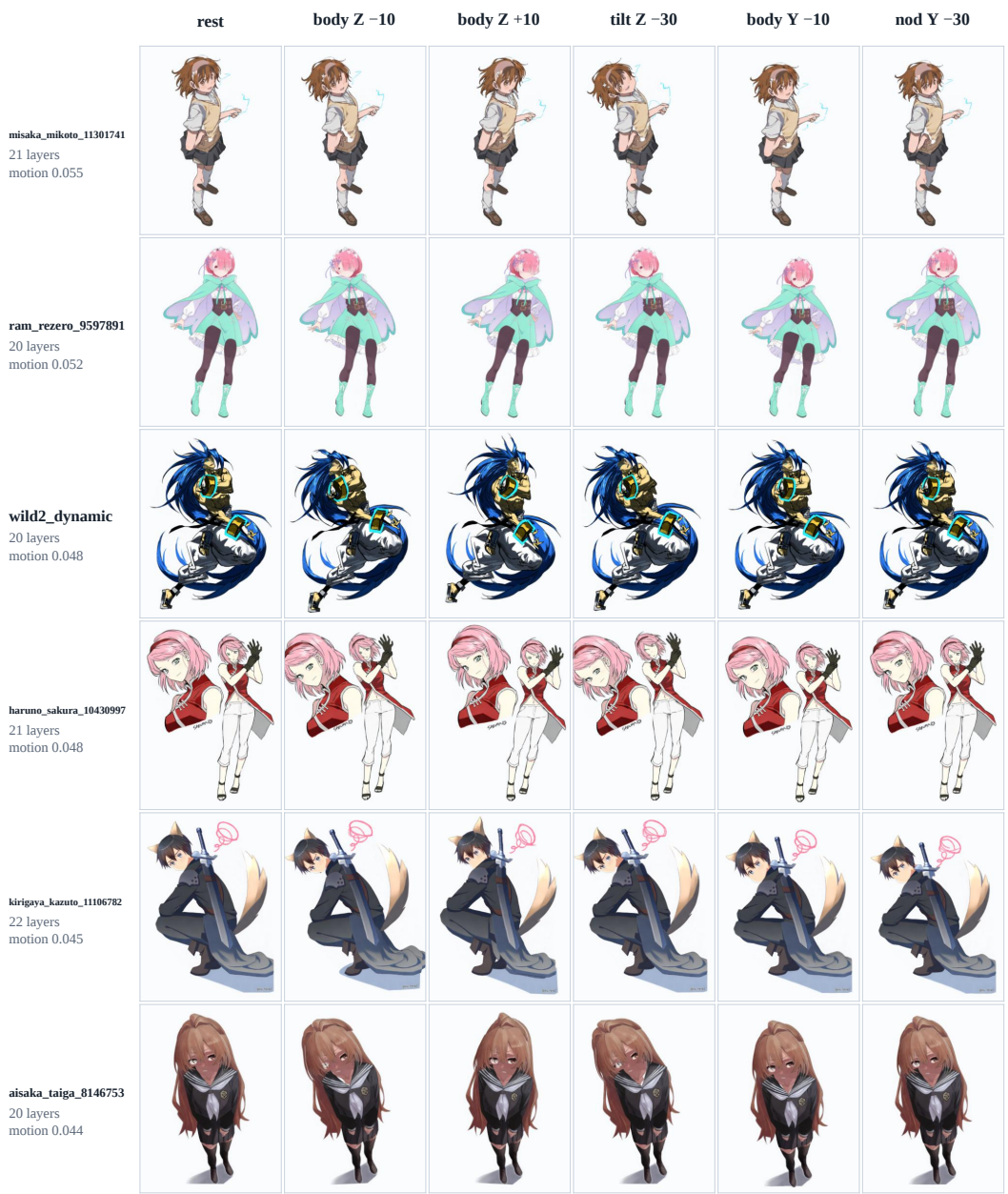

Figure 34: In-the-wild rigs, raw output, sheet 2. Format as Fig. 48.



Figure 35: In-the-wild rigs, raw output, sheet 3. Format as Fig. 48.

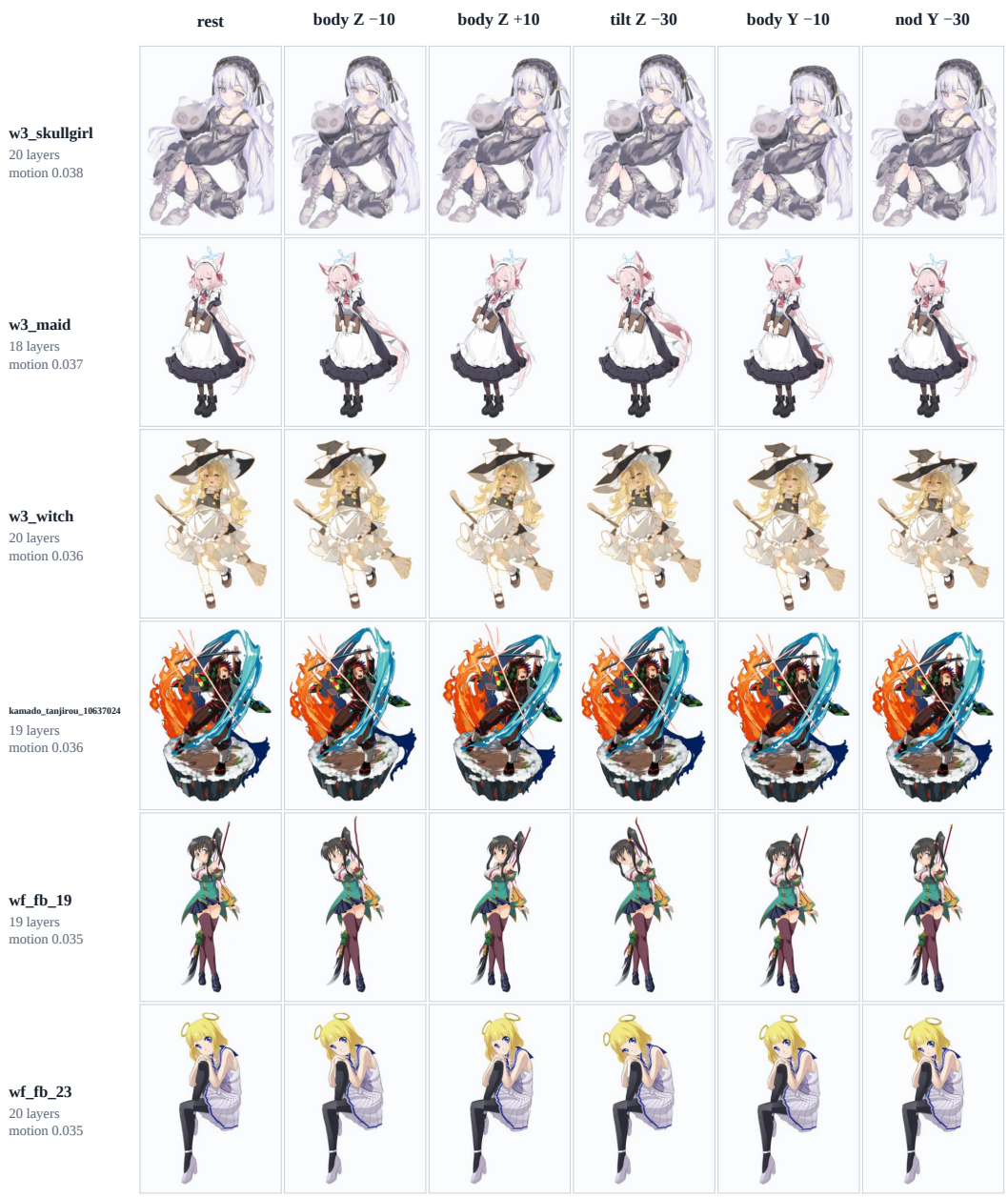

Figure 36: In-the-wild rigs, raw output, sheet 4. Format as Fig. 48.

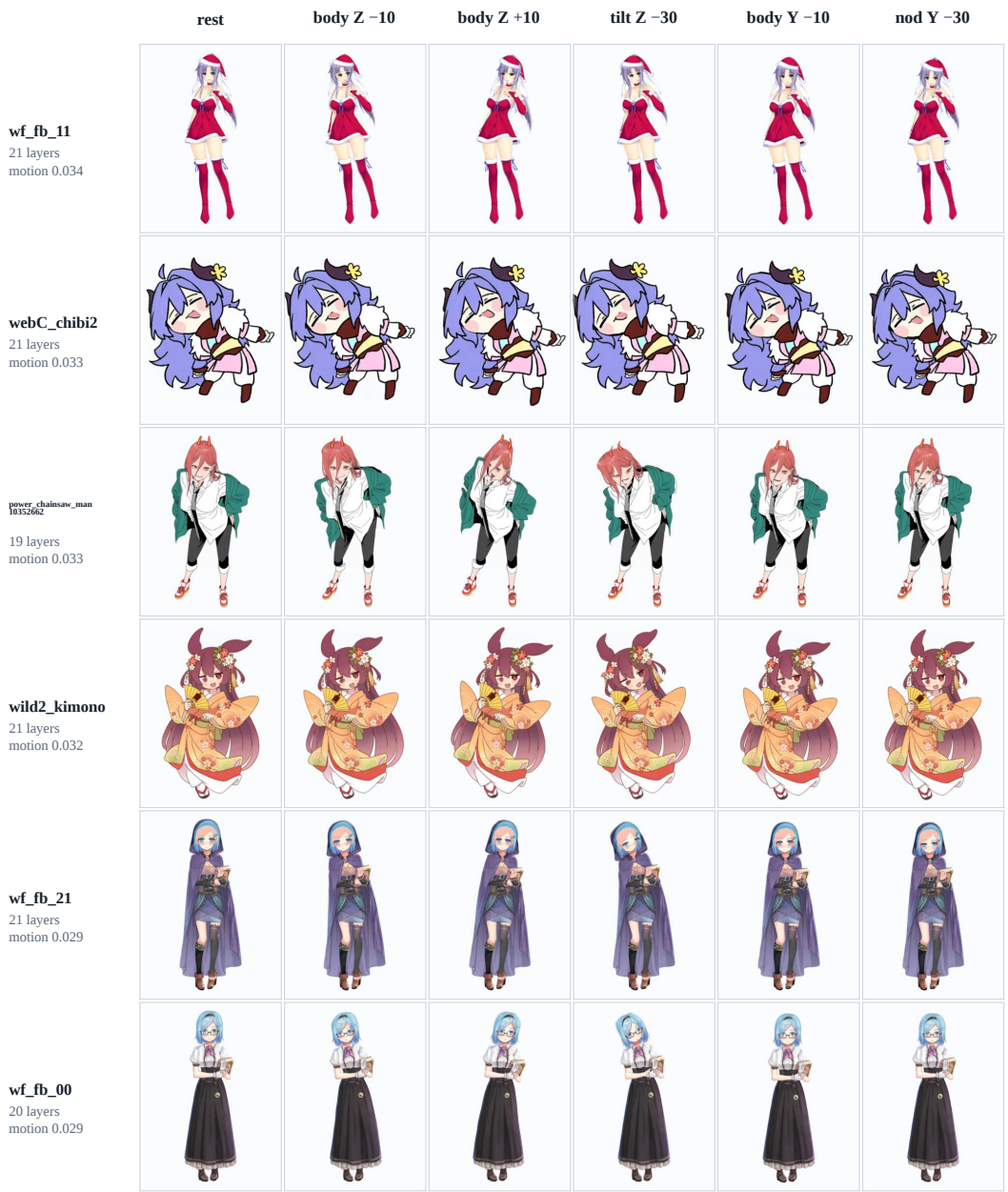

Figure 37: In-the-wild rigs, raw output, sheet 5. Format as Fig. 48.

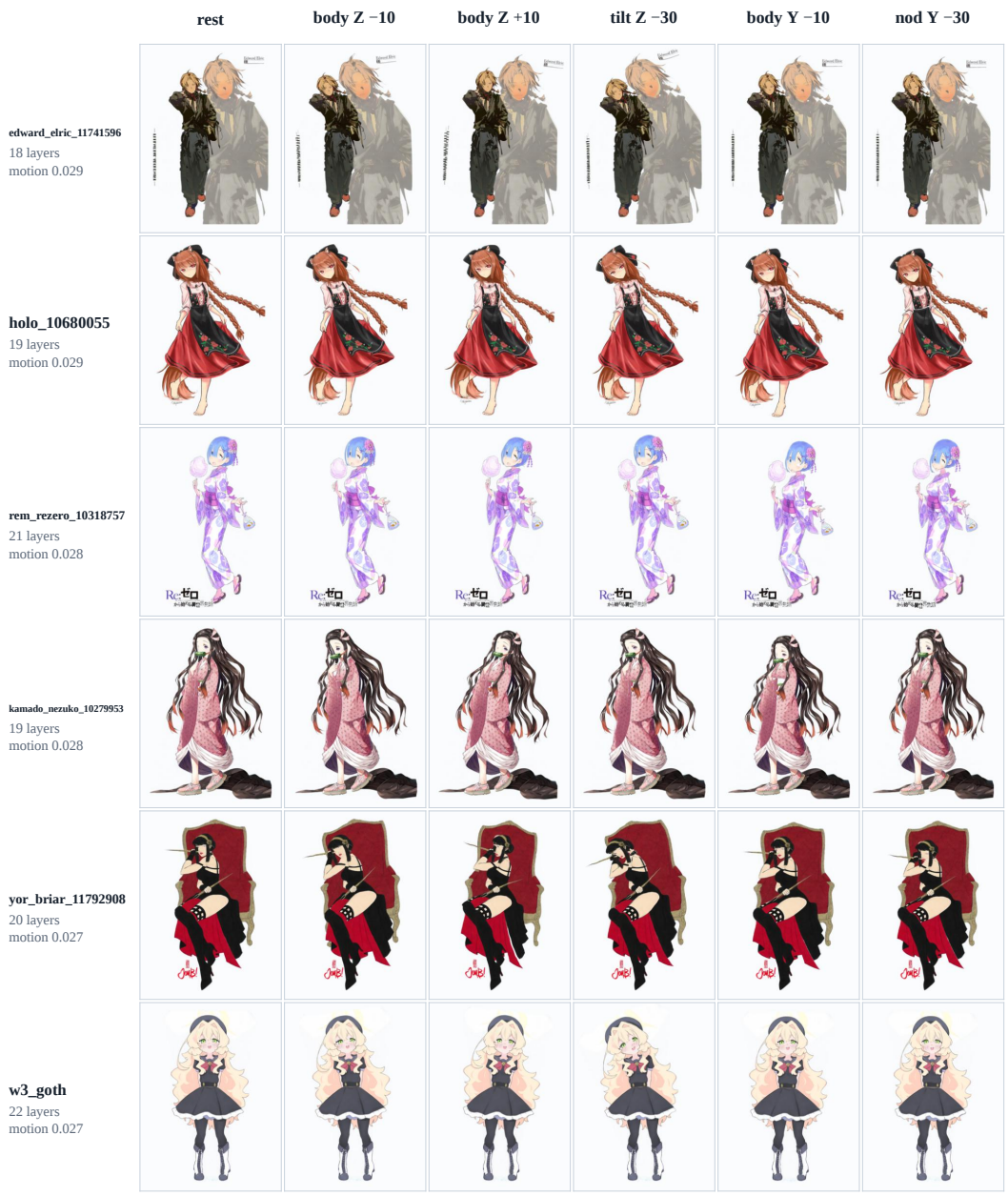

Figure 38: In-the-wild rigs, raw output, sheet 6. Format as Fig. 48.

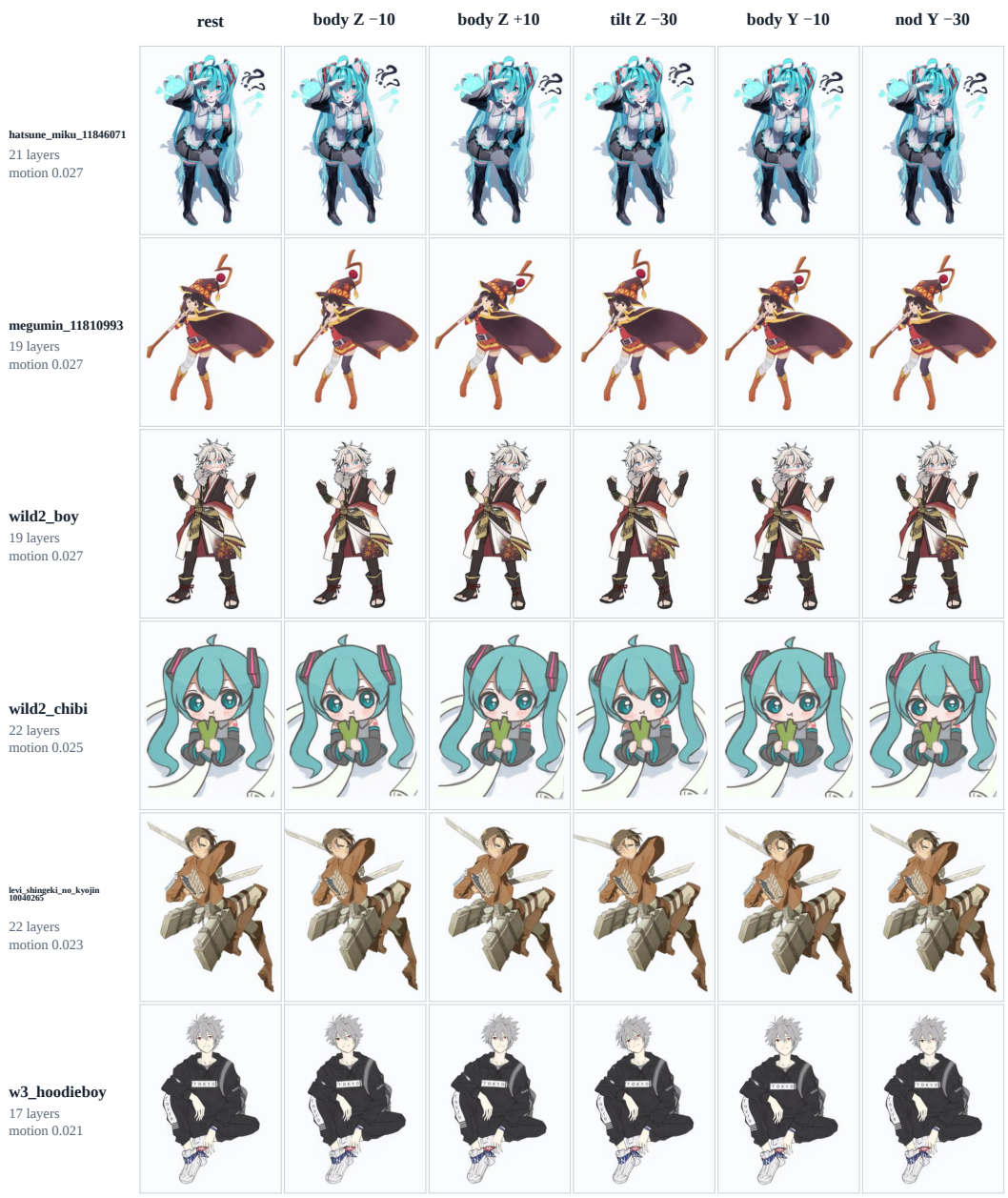

Figure 39: In-the-wild rigs, raw output, sheet 7. Format as Fig. 48.

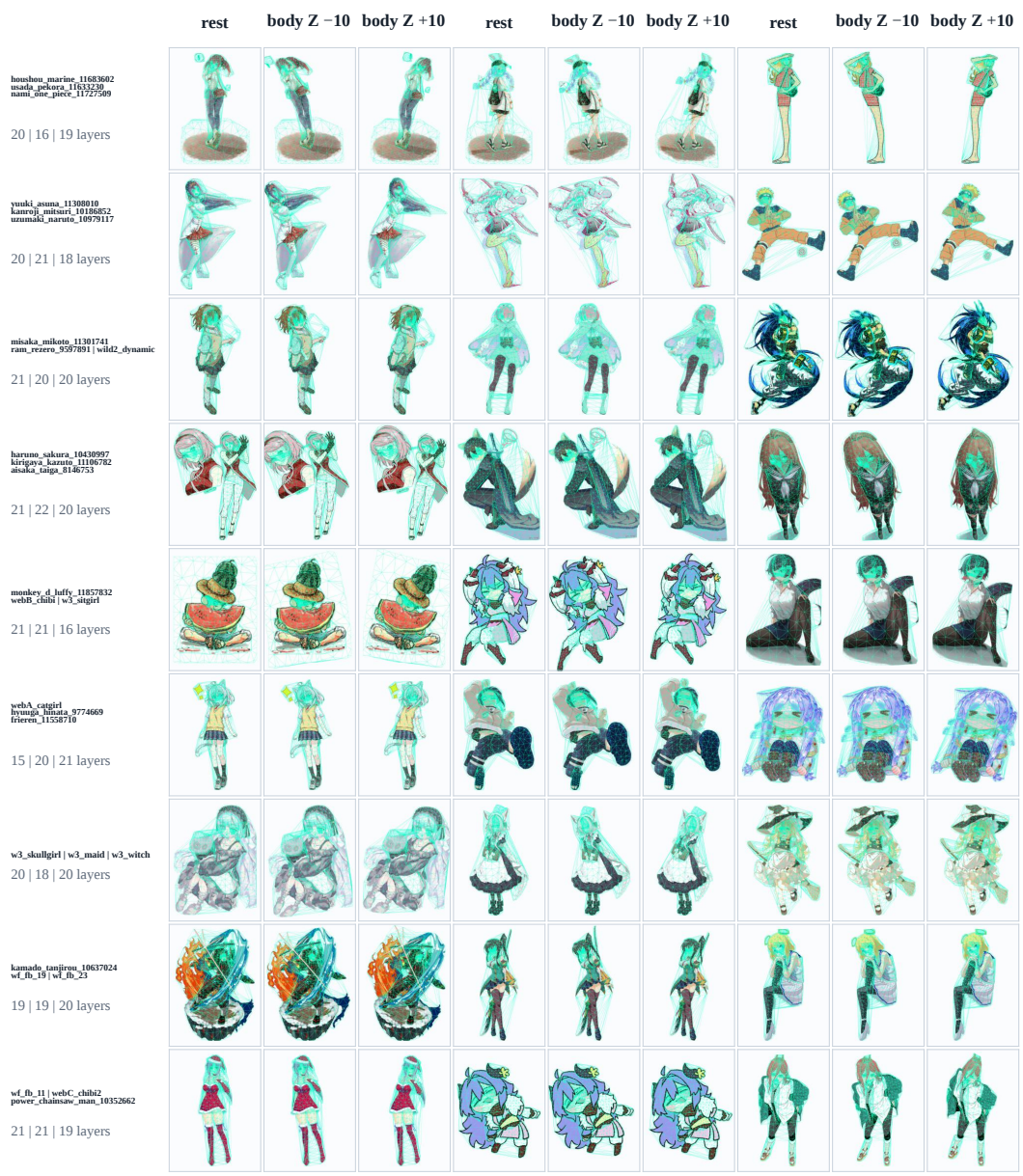

Figure 40: **The generated triangulation drawn over the artwork**, at rest and at the two poses that displace these characters most, three characters per row. Every visible pixel of every layer must lie inside a triangle or the layer tears at its silhouette during deformation. Sheet 2 is Fig. 41.

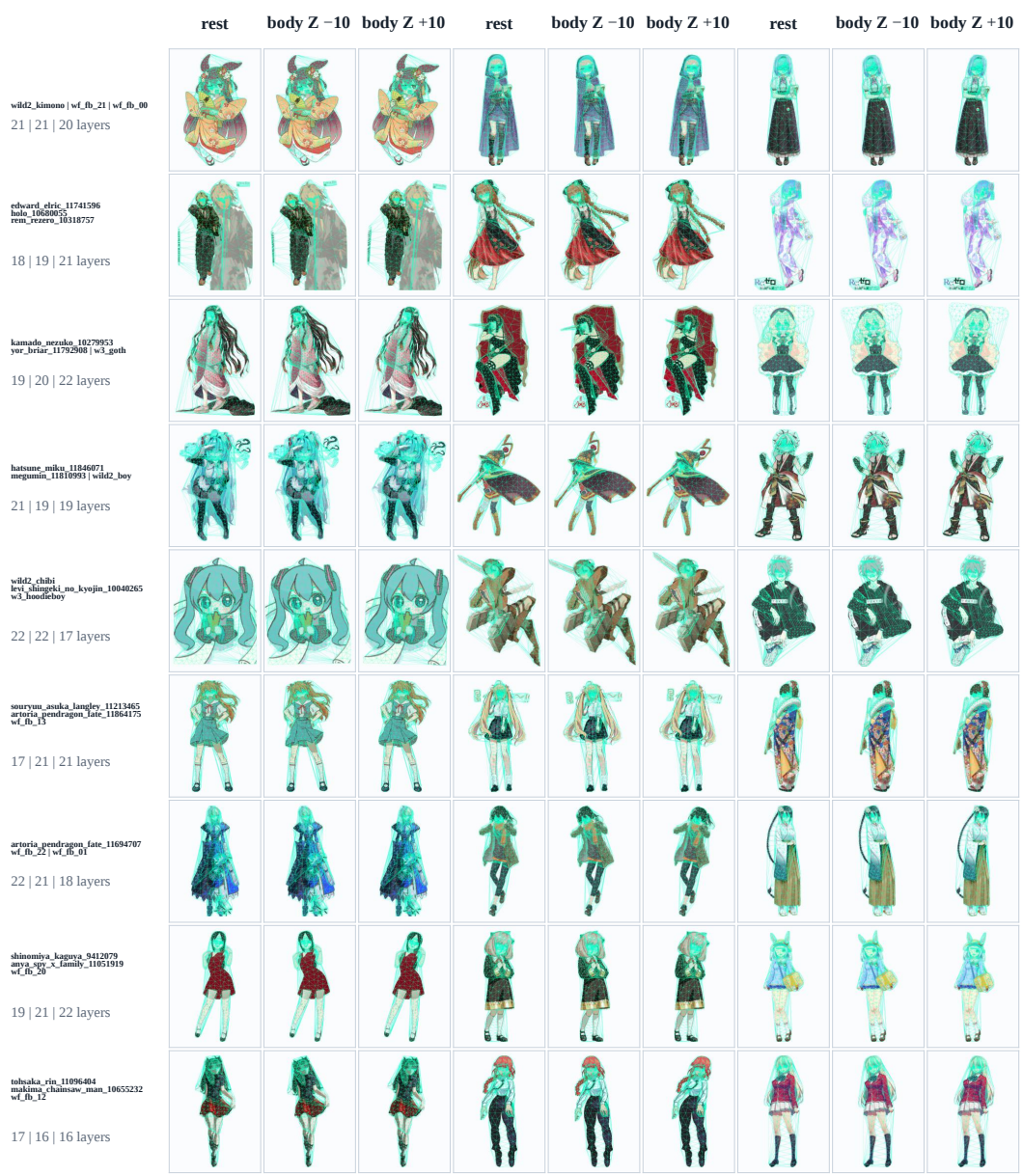

Figure 41: The generated triangulation drawn over the artwork, sheet 2. Format as Fig. 40.

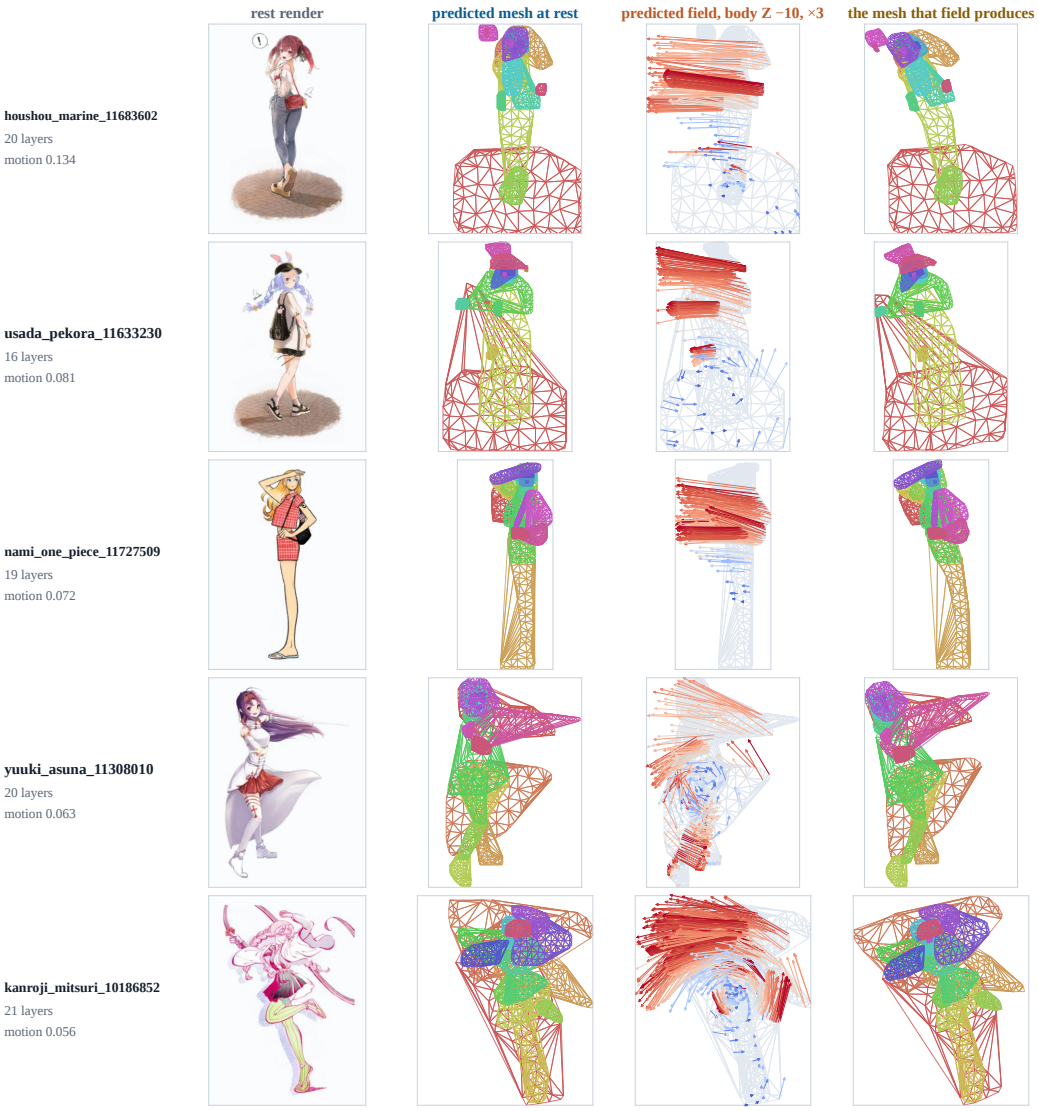

Figure 42: **What Stage 2 actually predicts, drawn from the rig rather than rendered.** Per row: the rig at rest; the predicted triangulation at rest with one hue per layer; the predicted displacement field for `ParamAngleX` $= +30$ as per-vertex arrows, exaggerated $3\times$ and coloured blue (small) to red (large); and the deformed triangulation. The field is continuous across layer boundaries although the layers share no connectivity, and its magnitude is organised by depth although depth is never an input.

What this means for deployment. Resident weights are 103.9 MiB, of which 84.5 is the frozen DINOv2 encoder and only 19.4 the model we train; the checkpoint on disk is 19.0 MiB. Loading costs 0.42 s for the joint model and 2.85 s for DINOv2, once per process rather than per character. The activation cost is the only quantity that grows steeply: $5.2\times$ more tokens costs $24.7\times$ more activation memory, exactly the $O(T^2)$ of dense attention over vertices. At the largest character we have (125 layers) this is still under 4 GB, but a character an order of magnitude denser would need attention windowing, most naturally over layer blocks, which is a change to the mechanism this paper argues for and therefore not one we make casually. Finally, mesh construction is 2.7% of the large character's time, which is worth stating because the triangulation is the part of Stage 2 that looks expensive and is not: the cost is attention over vertices, and the mesh's job is to keep the vertex count low, which is exactly what §W.3 measures it doing (83 vertices per layer against a grid's 112 at equal coverage).

P What We Tried That Did Not Work

The conditioning signals below are all plausible, several were suggested by reviewers of an earlier version, and none of them beat the plain joint model. We report them because a reader choosing what to add next is better served by knowing which additions we already paid for. Every row of Tab. 16 uses the identical 46-character clean benchmark, pack, objective and schedule, and differs from the baseline only in the stated conditioning; all numbers are true generation. The essential context is the **seed spread**: two runs of the baseline that differ only in random seed give 0.7289 and 0.7533, a range of 0.024, so a variant must move the mean by more than that to mean anything. None of them does.

Explicit per-vertex draw order (C1, C3). Reviewers observed that a generated layer stack can misorder parts relative to the artist's true z-order, and suggested conditioning Stage 2 on draw order explicitly. We did: each vertex additionally receives its layer's normalised draw-order position, front to back, Fourier-encoded and added to the token embedding, which is a strictly larger hypothesis class than the layer-identity embedding alone. It scores 0.7148, below both baseline seeds. Our reading is that layer identity plus a shared canvas already carries the ordering information the displacement field needs: layers that occlude each other are adjacent in the stack *and* overlapping in the canvas, so the ordering is recoverable from what the model already sees, and adding it as a separate channel only spends capacity. This does not resolve the reviewers' deeper point, which is that a wrong ordering *in the layer stack itself* is a Stage-1 error we cannot fix in Stage-2; that remains open and is stated as a limitation.

A 2.5D parallax prior. A head turning is not a 2D slide: near and far parts of the face should move differently, which a per-vertex 2D displacement field must learn implicitly. We added a zero-initialised residual that predicts a per-vertex depth z and a per-pose parallax gain g, contributing $z \cdot g$ to the pre-tanh direction, so the variant reduces exactly to the baseline at initialisation and can only help if the data supports it. Two seeds give 0.7225 and 0.7170, inside the baseline seed range, and extending to 40 epochs gives 0.7223: no gain on the mean. One nuance is worth recording rather than burying. Parallax combined with draw-order conditioning at 40 epochs has the *best median* of any variant we trained (0.8255 against the baseline seeds' 0.8024 and 0.7993), the *most* characters above 0.80 ($25/46$), and by far the best amplitude calibration (magnitude ratio 1.054 against 1.238 and 1.163), while its mean is lower (0.7153). That pattern is a left-tail effect: the variant improves the typical character and the amplitude, and loses on a few characters badly enough to drag the mean. We do not ship it, because we cannot show the median gain exceeds noise with two seeds, but a 2.5D prior remains the most promising of the negative results and the amplitude effect is the one signal here that points at our known amplitude-compression failure.

Per-vertex image features instead of one pooled vector per layer. Our model gives every vertex of a layer the *same* pooled DINOv2 vector. The obvious refinement is to sample the 16×16 patch-token grid at each vertex's own position, so a vertex on a sleeve edge sees the sleeve edge. It is clearly worse: 0.6918, and the amplitude degrades badly (magnitude ratio 1.563 against 1.238). We interpret this as the conditioning becoming too local. A layer's displacement under a head turn is a property of *what the layer is* (a fringe, an iris, a collar) far more than of what any single vertex looks like, and per-vertex sampling lets the model latch onto local texture that does not predict global motion, while

Table 16: **Negative and inconclusive results.** All on the clean 46-character benchmark, same pack, objective and 20-epoch schedule unless noted, true generation. The two baseline seeds bracket the noise floor: a mean difference smaller than 0.024 is not evidence. Bold marks entries that beat *both* baseline seeds on that column.

	dir-cos ↑		mag	chars
Variant	mean	med.	med. → 1	≥ 0.8
baseline, seed 1	0.7289	0.8024	1.238	23/46
baseline, seed 2	0.7533	0.7993	1.163	23/46
+ draw-order conditioning	0.7148	0.7883	1.154	21/46
+ parallax, seed 1	0.7225	0.8048	1.095	**24**/46
+ parallax, seed 2	0.7170	0.7828	1.184	22/46
+ parallax, 40 ep	0.7223	0.8004	1.193	23/46
+ parallax + order, 40 ep	0.7153	**0.8255**	**1.054**	**25**/46
+ per-vertex image features	0.6918	0.7959	1.563	23/46
8 blocks, d 256 (6.7 M)	0.7325	0.7939	1.139	23/46
6 blocks, d 384 (11.2 M)	0.7168	**0.8137**	1.072	**25**/46

the pooled vector forces a layer-level summary. This is also the reason we do not use cross-attention into image patches, and it is one of the two things §V.4 states the architecture deliberately does not do.

Width and depth at fixed data. For completeness, two intermediate capacity points inside the small-model regime behave like the larger ones in ablation B: 8 blocks at d 256 (6.7 M) gives 0.7325 and d 384 at 6 blocks (11.2 M) gives 0.7168, both inside or below the baseline seed range. The d 384 variant does reach 25/46 characters above 0.80 with a median of 0.8137, the same median-versus-mean pattern as the parallax variant.

Balanced sampling does not rescue the long tail. The obvious response to a long-tailed parameter distribution is to rebalance the sampler, and we tried it. Counting *layer records* rather than poses (a rare parameter appears in many poses but contributes two moving layers each, while head rotation contributes forty, so layer count is what sets the gradient share), we physically repeat each pose $\lceil(\max/\text{count})^{0.5}\rceil$ times, capped at 24, which grows the epoch from 26,133 to 92,068 poses and repeats the rarest parameter twelve times. It does not help, and it very slightly hurts on both halves of the vocabulary: the eight warm-started parameters fall from 0.6983 to 0.6931 and the sixteen new ones from 0.5588 to 0.5562, with validation cosine indistinguishable (0.6957 against 0.6958). We read this as informative rather than merely negative. Repetition raises a rare parameter's share of the gradient but adds no information about it, so if the failure were a matter of optimisation pressure this intervention should have moved it. It did not, which is consistent with the interpretation we give in §W.5: with 17 to 18 moving-layer examples of brow angle in the whole corpus, the constraint is the number of distinct examples, not how often the optimiser sees them. Warm-starting helps for a different reason, namely that it transfers a displacement field the rare parameter can re-index rather than manufacturing supervision.

A methodological note on how these were measured. Four of these evaluations were initially wrong in a way worth documenting, because the same trap is easy to fall into. The parallax variants store extra heads that the baseline architecture does not have; our evaluation script built the architecture from an environment flag, we forgot to set it, and `load_state_dict` silently discarded those heads. The scores that came back were of a crippled model, and they looked plausible (0.6307 to 0.7096), which is exactly what makes the failure dangerous. We now detect optional architecture branches from the checkpoint's own key set and *assert* that no checkpoint weight goes unused, so a mismatch raises instead of scoring. Every number in this table is from the corrected path.

Table 17: **Renderer anti-aliasing, before and after**, against a $16\times$-supersampled alias-free reference of the identical view (`bench_34`, 35 layers, rest pose, so all three images are pixel-comparable by construction). "edge band" is the 13.3% of pixels where the reference has a silhouette gradient. Staircase is mean$|\nabla^2|$ on that band; the reference's own value is 13.172, so the meaningful quantity is the excess over it.

	full frame		edge band	
Render path	PSNR	RMSE	PSNR	RMSE
before (`LINEAR`, 900^2 buffer)	51.99	0.641	43.49	1.707
after (mipmap + aniso, 1800^2 buffer)	**58.38**	**0.307**	**50.87**	**0.729**

Q Rendering: Where the Aliasing Came From and What Fixed It

Two reviewers reported that aliasing in our renders impeded their judgement of animation quality, and asked whether the reported metrics reflect the method or the renderer. We instrumented the viewer to answer both parts.

Multisampling was already enabled, and was the wrong tool. Reading the live WebGL2 state through the browser shows the framebuffer was created with `SAMPLES` $=4$, `SAMPLE_BUFFERS` $=1$ and `MAX_SAMPLES` $=4$, so $4\times$ MSAA was active from the start and could not be increased. MSAA anti-aliases *triangle* edges, but in this renderer the visible silhouette of hair or a skirt hem is not a triangle edge: each layer is a textured triangle fan whose outline comes from the layer's alpha channel sampled *inside* the triangles, where all 4 coverage samples of a pixel read the same texel and MSAA does nothing. Reporting "MSAA is on" would therefore have been a non-answer.

The two real causes. First, minification without a mip chain: `TEXTURE_MIN_FILTER` was `LINEAR` while a layer texture of up to 600 px is typically drawn into about 300 px of canvas, so bilinear filtering selected roughly one texel in four and the alpha edge crawled. Second, a backing store equal to the CSS box at `devicePixelRatio` $=1$, i.e. exactly one sample per output pixel, so nothing anti-aliased the alpha edge at all. We fixed both: the backing store is now supersampled $2\times$ per axis (clamped so large stages cannot exceed a 4096 px buffer), every layer texture gets a mip chain with trilinear minification, and anisotropic filtering at $8\times$ is enabled where the extension exists so strongly sheared meshes do not over-blur. Mip generation happens in premultiplied-alpha space, which is the correct space and avoids dark or bright halos at the silhouette.

Measured, against an alias-free reference. We built the reference by rendering the identical view into a 3600^2 backing store ($16\times$ area supersampling on top of MSAA) and box-filtering to 900^2. Constructing that reference from the old and the new code path gives images agreeing to 92.9 dB PSNR (RMSE $0.006/255$), so the reference is unbiased with respect to the change and can score both sides fairly. Scoring on the silhouette band (reference gradient above $8/255$, dilated 2 px, 13.3% of the frame) gives Tab. 17: edge-band error against the alias-free reference drops $2.34\times$ ($+7.38$ dB) and full-frame error $2.09\times$ ($+6.39$ dB). Excess staircase energy relative to the converged image, measured as mean$|\nabla^2|$ on the band minus the reference's own value, falls from 0.759 to 0.152, i.e. $5.0\times$ closer to convergence. Fig. 43 shows two 76^2 px windows on diagonal silhouette edges magnified with nearest-neighbour so no resampling can hide the difference.

The consequence for our numbers is worth stating explicitly: the primary Stage-2 metric is per-vertex direction cosine computed on *geometry*, so it never touched the rasteriser and is unaffected by any of this. The pixel metrics of Appendix S were measured on the improved path, and the reviewers' qualitative concern is addressed by the renderer fix rather than by an argument.

Columns 1–3 of Fig. 43 will look nearly identical to a reader, and we say so rather than implying otherwise: the residual is well under one intensity level per pixel, so no crop of the render can make it dramatic. The evidence is in columns 4–5, where the staircase structure the fix removes is visible without amplification in the intensity profile. Mean absolute error over the three windows falls $1.27 \rightarrow 0.43$ (per window $1.59 \rightarrow 0.53$, $1.15 \rightarrow 0.47$, $1.06 \rightarrow 0.29$), consistent with the $43.49 \rightarrow 50.87$ dB silhouette-band PSNR of Tab. 17. What remains is at the level of quantisation noise against a converged render.

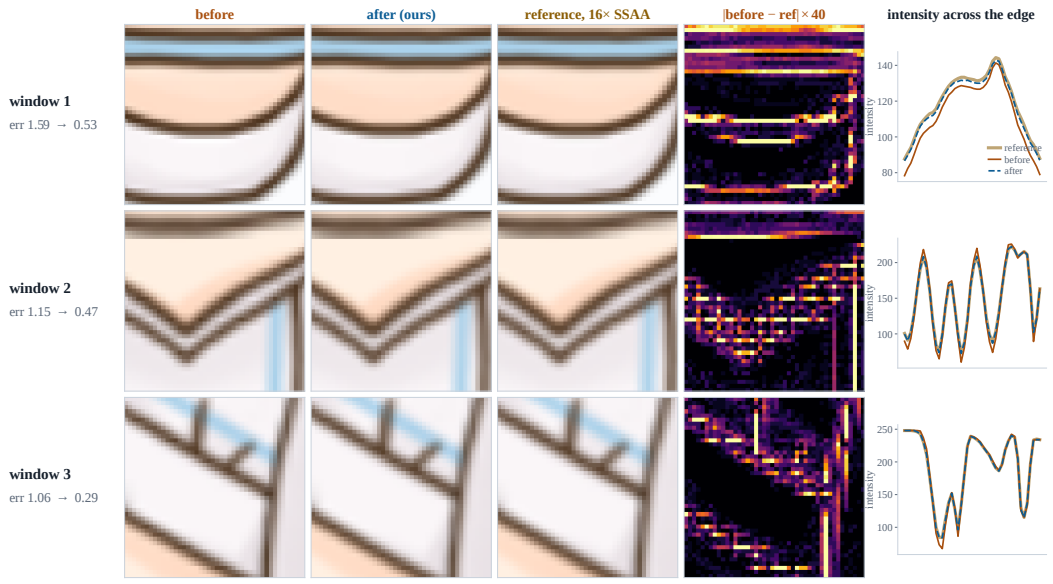

Figure 43: **Silhouette anti-aliasing, before and after the renderer fix.** Three 44×44 px windows chosen automatically at the largest-error locations. Columns 1–3 are the rendered crops (before, after, and a $16\times$-supersampled reference) at nearest-neighbour magnification. Column 4 is $|\text{before} - \text{ref}|$ amplified $40\times$ on a perceptual colour map; column 5 is the intensity profile along the worst scanline of each window.

R Parameter-Vocabulary Extension: Full Per-Parameter Results

This appendix backs §W.5. Table 18 reports every one of the 24 parameters individually on the 46 validation characters under true generation, next to the number of moving-layer records that parameter has in training. Reading the table as a scatter of cosine against support makes the conclusion of §W.5 hard to avoid: the four parameters with fewer than 30 training samples occupy four of the five worst positions, the two with negative cosine are the two rarest in the corpus, and the two best-learned new parameters (gaze) are the two with the most support among the additions. Note also that the middle column, the 8-parameter model evaluated on the same pack and characters, shows the extension is close to free on the parameters both models share, with `BodyAngleX` the only entry where the larger vocabulary is clearly better ($0.576 \rightarrow 0.599$) and `EyeROpen` the only one clearly worse ($0.863 \rightarrow 0.784$).

A second caveat applies to this table specifically and we flag it because it also bounds what the benchmark can measure. A validation character can only be scored on the parameters *its artist actually rigged*, so the effective evaluation pool shrinks for the rarer parameters: 40 of 46 characters carry ground truth for `AngleX`, 25 for gaze, 15 for brow height, and only 8 for `BrowLAngle`. The rarest parameters are therefore doubly disadvantaged, thin in training and thin in evaluation, and their numbers carry correspondingly wide error bars. We state the per-parameter character count in the table so this is visible rather than buried.

Making density-confounded comparisons impossible by construction. The measurement error described in §W.5, where a re-meshing worth 0.063 cosine was briefly mistaken for a model regression worth 0.031, was possible because our evaluation script accepted any (checkpoint, pack) pair and averaged whatever it found. We added three switches that make a controlled comparison expressible and an uncontrolled one obvious: restricting which parameters are scored, mapping parameter names through a checkpoint's own vocabulary so a small-vocabulary model can be scored on a large-vocabulary pack without shifting embedding indices, and restricting which layers are scored while all layers still enter the forward pass as joint context. The last switch is what allows the density control (identical-vertex-count layers only) to be measured without changing the model input, since removing a layer from the forward pass would change every other layer's prediction. We recommend

Table 18: **All** 24 **parameters, individually.** 46 validation characters, true generation, per-vertex direction cosine and magnitude ratio (ideal 1.0). "support" is the number of moving-layer records in training; "chars" is how many of the 46 validation characters have artist ground truth for that parameter at all. $*$ marks a parameter absent from the 8-parameter vocabulary, so its "8p" cell is empty by construction. Colour marks the capability tier of Tab. 27: **good**, marginal, unusable.

Parameter	support	chars	8p	24p cold	24p warm	mag
EyeBallY*	181	23	–	0.999	0.999	1.10
EyeBallX*	215	24	–	0.997	0.997	0.96
BodyAngleZ	1097	13	0.876	0.875	0.874	1.33
AngleZ	2804	36	0.849	0.818	0.845	0.87
BrowRY*	25	12	–	0.840	0.843	0.81
BrowLY*	25	12	–	0.845	0.832	0.64
EyeLOpen	143	29	0.797	0.761	0.789	0.83
EyeROpen	147	29	0.863	0.762	0.784	0.82
Breath*	948	27	–	0.739	0.749	0.68
AngleY	2736	39	0.744	0.742	0.742	1.15
EyeRSmile*	17	4	–	0.372	0.724	0.57
AngleX	2841	40	0.721	0.709	0.720	1.15
EyeLSmile*	16	4	–	0.561	0.710	0.44
HairBack*	91	23	–	0.545	0.644	0.48
BodyAngleY	623	10	–	0.643	0.635	1.51
BodyAngleX	683	14	0.576	0.667	0.599	0.53
HairFront*	233	31	–	0.536	0.574	0.52
HairSide*	244	27	–	0.381	0.427	0.52
BrowLForm*	29	9	–	0.236	0.389	0.67
BrowRForm*	48	10	–	0.145	0.337	0.33
MouthOpenY	126	31	0.239	0.274	0.235	0.38
MouthForm*	123	24	–	0.114	0.176	0.03
BrowRAngle*	18	9	–	−0.020	−0.022	0.04
BrowLAngle*	17	8	–	−0.266	−0.073	0.03

that any future work reporting per-vertex cosine on Live2D-Bench state the pack, the character list and the parameter list alongside the number, and we ship ours with the released models.

S Cross-Decomposer Diagnostic of the Animation Prior

A practical requirement for deployment is that the Stage 2 animation prior accept layer stacks from more than one decomposer. This matters because a user may already have a preferred decomposition tool, and because it isolates whether Stage 2 has learned a transferable animation prior versus merely co-adapting to Stage 1 artifacts.

We test this by feeding the *same* in-the-wild illustration through two independent decomposers, our Stage 1 and the third-party See-through Lin et al. (2026) decomposer, which produces a different layer count and split convention, and animating both stacks with the identical, frozen Stage 2 model (auto-mesh + image-conditioned offset prediction). Fig. 44 shows selected head-roll renders on both decompositions across distinct art styles and body types, with no per-case tuning. These examples show that the frozen model can process either layer source; they do not by themselves establish motion fidelity or disentangle the appearance and geometry cues.

An archived eight-rig rendering diagnostic supplies a larger but less clean quantitative check than the three-character end-to-end control in Table 20. It contains 64 non-rest keyposes across eight characters. The motion-region PSNR (mPSNR) scores only pixels changed by the artist's pose relative to the artist's rest render, using a ground-truth-only difference mask with an 8/255 threshold and a nine-pixel dilation. Four of the eight source names match the final Stage 1 training manifest; the remaining four do not, but earlier training manifests are unavailable. All eight characters also belong to the Stage 2 validation set used for checkpoint selection. Thus none of the rows in Table 19 is an untouched test. The renderer and keypose protocol also differ from Table 20, so the scores must not be pooled. The paired character-level bootstrap interval for the all-eight mPSNR difference is $[0.55, 1.89]$ dB; for the four names absent from the final manifest it is $[-0.07, 2.29]$ dB. More directly, keeping each generated stack fixed at its rest pose improves mPSNR from 12.90 to 13.10 dB for our layers (predicted minus static: -0.20 dB, paired 95% bootstrap interval $[-0.57, 0.16]$, 2/8 characters improved by prediction) and from 11.63 to 12.08 dB for See-through layers. On our layers, static



Figure 44: **Cross-decomposer diagnostic.** Six Stage 2 validation illustrations, each given to *one frozen* Stage 2 checkpoint as three layer stacks: the artist's own (gold), our Stage 1 (blue), and See-through Lin et al. (2026) (orange). Each condition appears twice, as the rendered frame and as the posed mesh at the same keypose, with that stack's own layer count beneath.

rendering also improves full-frame PSNR from 16.33 to 16.90 dB. Thus the cross-decomposer pixel advantage does not demonstrate accurate motion, even with the same layers held fixed. Per-character numbers and split flags are in `eight_rig_render_audit.json`.

We also subtract each candidate stack's own rest render before comparing its pose change with the artist's pose change. Let $R_s(q)$ be the render from stack s at control value q, $R_\star$ the artist stack with

Table 19: Archived eight-rig diagnostic, mean of per-character non-rest keypose scores. Each paired cell is Bunraku / See-through with the same frozen motion model. The footnote gives zero-motion controls on the same decomposed stacks, as well as the artist-layer control. "No name match" refers only to the final Stage 1 manifest.

Sources	n	PSNR ↑	mPSNR ↑	LPIPS ↓
All	8	16.33 / 14.35	12.90 / 11.63	.201 / .268
No name match	4	15.28 / 13.23	11.38 / 10.22	.227 / .311
Name match	4	17.39 / 15.46	14.42 / 13.05	.176 / .225

Same-stack zero-motion mPSNR (Bunraku / See-through): 13.10 / 12.08, 11.66 / 10.21, and 14.54 / 13.94 dB for the three rows. Artist-layer zero-motion mPSNR: 13.59, 12.24, and 14.93 dB.

artist motion, and M the same artist-only motion mask as above. The relative motion-residual error is

$$E_\Delta(s,q) = \left(\frac{\sum_{x \in M} \| [R_s(q) - R_s(0)](x) - [R_\star(q) - R_\star(0)](x) \|_2^2}{\sum_{x \in M} \| [R_\star(q) - R_\star(0)](x) \|_2^2} \right)^{1/2}. \tag{22}$$

A same-stack static candidate scores exactly 1; lower is better. After averaging within characters, predicted motion scores 1.10 on our layers (paired character-bootstrap 95% interval relative to static $[0.03, 0.17]$; only 1/8 characters improves), 1.22 on See-through layers, and 0.99 on artist layers. Our generated stacks are better than See-through stacks in 7/8 paired characters on this metric, yet both automatic stacks are worse than leaving themselves still. This motion-specific negative result is from validation-selected characters and cannot be generalized as an independent test. It suggests that the artist-layer vertex-direction score does not transfer reliably to generated-layer rigs. The exact per-character values are in `eight_rig_render_audit.json`.

The shown poses under all three stacks require no per-case adjustment, but part drift and missing regions remain visible in some cases. The artist-layer column bounds what Stage 2 can achieve when the decomposition is perfect, and where a render shows an artefact the mesh column beside it tells you whether Stage 2 faithfully animated a broken decomposition or predicted the wrong motion.

Quantifying cross-decomposer render fidelity. Beyond the qualitative evidence of Fig. 44, we quantify how the choice of layer-decomposition source affects end-to-end animation fidelity. On the three OOD characters that ship an artist-authored Live2D rig: so a ground-truth *animation* exists as reference; we decompose each character three ways: (i) its artist ground-truth layers, (ii) our Stage 1, and (iii) See-through Lin et al. (2026). Every stack is animated by the identical frozen Stage 2 model and rendered; we report PSNR/SSIM/LPIPS of the rendered composite against the artist's own rendered animation, averaged over all 13 non-rest keyposes per character (Tab. 20). Using the artist's own layers is an upper bound (the reference shares the same textures). Among the two *automatic* decomposers, our Stage 1 outperforms See-through on every metric (PSNR 23.4 vs. 21.1, SSIM 0.883 vs. 0.865, LPIPS 0.088 vs. 0.120), and the advantage holds per character on the two with appreciable motion. This small control primarily measures the effect of decomposition source under a frozen motion model; it does not establish general motion quality on generated rigs.

What the rest-pose control can identify. A rest-pose control measures one component of the two error sources. At rest, no animation, pure decomposition-vs-artist mismatch, our Stage 1 already reaches PSNR 27.2 and See-through 22.6, versus the identical artist layers' ceiling. Adding the predicted animation drops these by only ~ 3.7 and ~ 1.5 dB respectively, so the large gap to the GT-layer condition is consistent with decomposition errors contributing substantially to these absolute pixel differences. This comparison cannot attribute the generated-rig motion failure to decomposition alone: the rest-subtracted motion test in Sec. 4 still favors static layers, and its post hoc artist-layer render control is also inconclusive. Artist-mesh direction cosine and magnitude ratio (Tab. 24) measure a different endpoint.

Caveat: pixel fidelity on GT layers is floor-dominated. When the layer source is the artist's own layers, pixel PSNR is a *weak* proxy for motion quality: a keypose perturbs only a small fraction of pixels and both renders share identical textures, so PSNR saturates near a high floor regardless of motion accuracy. A predict-zero-motion control (render the rest pose against the GT-motion render) scores PSNR 41.6 on the same keyposes, essentially indistinguishable from the animated model's

Table 20: **Cross-decomposer animation render fidelity.** Rendered-composite similarity to the artist's own rendered Live2D animation, mean over 3 OOD characters × 13 non-rest keyposes, with the frozen Stage 2 model animating each decomposition source. All higher-is-better except LPIPS. See the text for how to read the control columns.

	Rest	Animated vs. artist		
Layer source	PSNR	PSNR	SSIM	LPIPS
Artist GT layers	∞	41.8	**0.955**	**0.028**
Ours (Stage 1)	**27.2**	**23.4**	**0.883**	**0.088**
See-through Lin et al. (2026)	22.6	21.1	0.865	0.120

41.8. Pixel metrics on GT layers should therefore be read as a *decomposition-source* comparison, where the between-condition gaps are large and meaningful; LPIPS separates the conditions more cleanly than PSNR (0.028 for GT layers vs. 0.088/0.120 for the automatic decomposers).

Rest PSNR in Tab. 20 is a no-motion control (candidate rest pose against artist rest pose) that isolates decomposition-versus-artist mismatch, and the *Animated* columns add the predicted motion. The GT-layer row is an upper bound: it shares the reference textures, so its rest PSNR is unbounded, and its *Animated* PSNR doubles as a predict-nothing floor (41.6), which illustrates that pixel PSNR on GT layers barely responds to motion at all.

Comparing animation predictors. Tab. 21 compares predictors on the clean 46-character benchmark under true generation. The joint implementation scores 0.736 record-averaged direction cosine against 0.693 for the independent per-layer implementation, and their magnitude ratios are 0.99 and 0.86. The implementation changes both layer context and architecture: the independent predictor uses patch cross-attention and a pooled magnitude head, whereas the joint predictor uses pooled appearance and a vertex magnitude head. These numbers therefore do not isolate cross-layer attention. The coordinate-MLP score of 0.582 is an additional architectural baseline. One content-mesh training run with an incorrect parameter vocabulary diverged; it is a debugging record, not evidence against that mesh. The capacity sweep in Tab. 26 tests another question under its specified training schedule. Two further findings. (i) Single-model direction accuracy carries seed variance of the order of the 0.024 floor of §W.3, which is why the joint entry here reads 0.736 while the committed checkpoint reads 0.7397; a 3-seed *output ensemble* with amplitude rescaling exploits that variance to reach cos 0.743 with magnitude ratio 1.00, at 3× inference cost. We quote no tighter number than the 0.024 floor: an earlier version of this appendix reported std ≈ 0.006 here, which came from a per-record rather than a per-character metric and is retracted in §W.3. (ii) The architectural motion priors we tried, a 2.5D depth-parallax turn prior and explicit draw-order / composite-context conditioning, did *not* beat that floor: averaged over its three runs the parallax prior scores 0.7206 against the base pair's 0.7411, i.e. $\Delta\cos -0.021$, which is worse by about the width of the floor rather than better. This local study did not improve the historical artist-mesh direction score. It does not establish a ceiling for the motion predictor or identify which component dominates the generated-rig motion residual; the latter requires the same-stack controls reported in Sec. 4.

T Extended Ablations of the Superseded Autoregressive Stage 2

For completeness we record the hyperparameter sweeps and abandoned variants of the *superseded* autoregressive token model (Appendix G), all measured on a 50-character OOD pool under that design's teacher-forced protocol; none of these numbers is comparable with the main-text true-generation results, and none supports a claim in the main paper. **Hyperparameter sweeps** showed $Q=128$ to be a sweet spot: lower-bin Q=64 retained $\cos=0.995$ but magnitude drifted to 21.5×; higher-bin Q=256 over-diluted the vocabulary, dropping cos to 0.957 while improving the magnitude error to **0.181**. A smaller d=128 (4.4 M) variant reached $\cos=0.993$, within 0.5 pp of the 12 M default, i.e. capacity already saturated for that design too. Removing label smoothing (σ_{LS}=0)

Table 21: **Stage 2 implementation comparison** (46-char clean benchmark, true generation, no teacher forcing). The independent and joint models differ in architecture as well as layer context. The values are from the original record-averaged evaluation, whose two implementations did not score identical record sets; the matched audit above supersedes this pair for inference about performance; ‡the magnitude ratio of the per-vertex coordinate-MLP run was not recorded on this pool; we give the direction cosine, which was, and do not carry over the value measured for it on the earlier 120-example pool, because that protocol supplied the mesh prefix and the two are not comparable.

Variant	dir-cos ↑	mag-ratio $\to 1$
3-seed ensemble + rescale	$\mathbf{0.743}$	$\mathbf{1.00}$
Joint multi-layer (single, ours)	0.736	0.99
Per-layer independent regression	0.693	0.86
Per-vertex coordinate-MLP	0.582	n/a‡
0.8B Qwen-VL regressor	muted motion, $\sim 10\times$ slower	
Content-mesh re-train (format mismatch)	diverged	

sharpened bin predictions but slightly destabilised training ($\cos = 0.992$). **Abandoned variants:** adding an ARAP edge-length regulariser conflicted with the magnitude head and collapsed the cosine to 0.13; a per-axis magnitude (α_x, α_y) caused a $\log 0$ crash in the first epoch.

U Failure Gallery

This gallery is generated from the model the paper ships, at each character's *worst* pose. For every validation character we search all (parameter, keypose) pairs for the lowest per-vertex direction cosine and show that pair; both the artist's frame and ours are read directly out of the released benchmark rig, so no inference is run and the figure cannot drift from the data. An earlier version of this gallery came from the superseded autoregressive design; it has been replaced.

The five rows are five *different* failure modes, and we classify them by which measurable property is wrong rather than describing them impressionistically. A character with one or two layers gives cross-layer attention nothing to condition on, which is a degenerate support rather than a prediction error. Amplitude collapse (magnitude ratio far below 1) and direction error (cosine near or below 0) are separately measurable and appear in different rows. The worst cases reach a cosine of -0.97, i.e. the predicted field points almost exactly opposite to the artist's, which is worth showing plainly: our mean of 0.7676 is an average over characters that includes cases this bad.

The rows of Fig. 45 are different failure modes, not one: a single-layer character gives cross-layer attention nothing to condition on, whereas amplitude collapse and direction error are distinct and separately measurable. The characters shown are those that fit the page at their true aspect ratio.

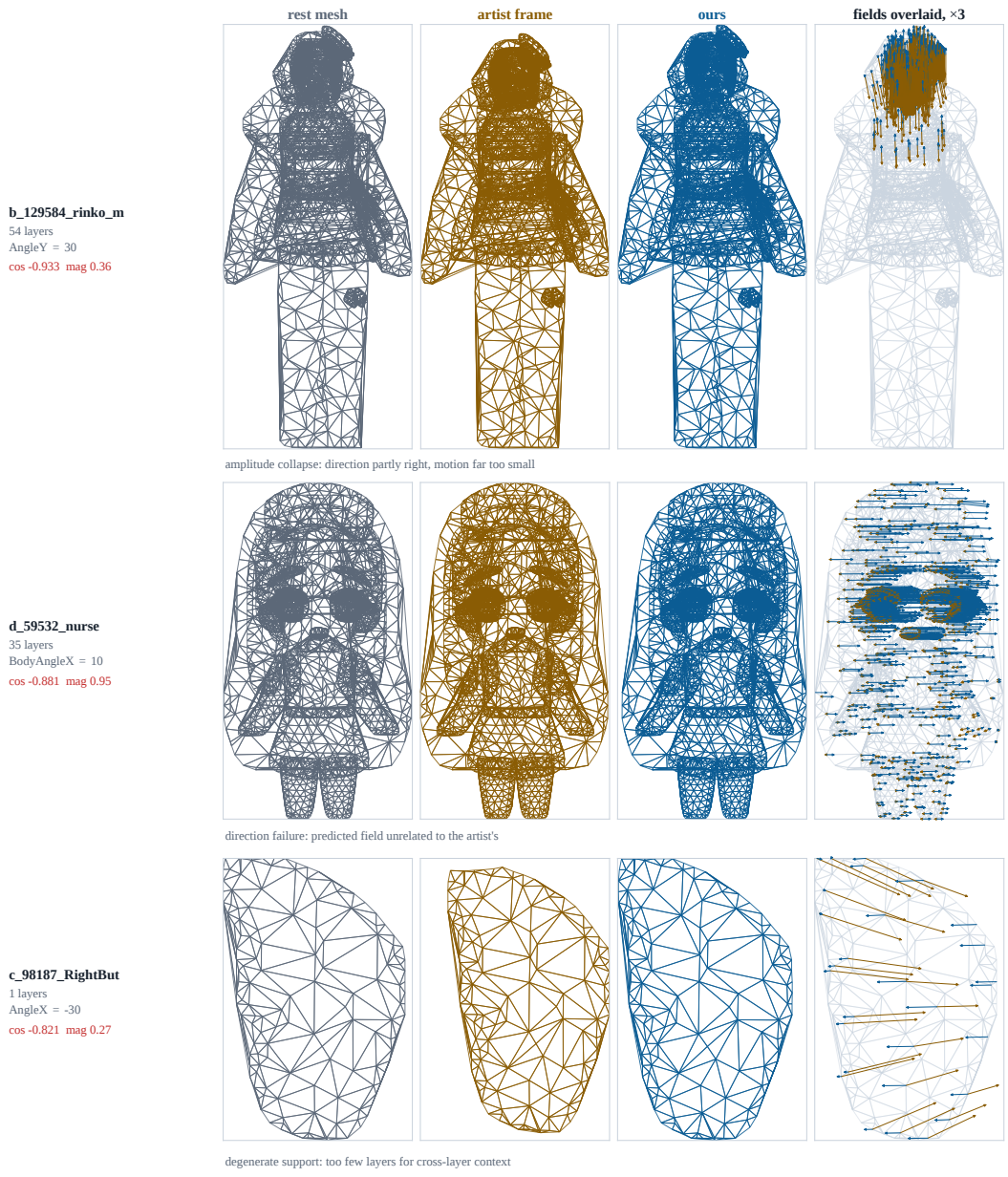

Figure 45: **Where the shipped model fails, at each character's worst pose.** Per row: the mesh at rest, the artist's frame, ours, and both displacement fields overlaid at 3× (gold artist, blue ours). The parameter, keypose, cosine and magnitude ratio beside each row are that character's worst pose, computed from the released rig, and the line under each row is the computed failure classification. Rows are ordered by worst-pose cosine.

V Extended Method Details

The overview in Fig. 2 is drawn from one character's own data rather than illustrated, and three of its annotations are measured quantities worth naming here. The input illustration was itself generated from a text prompt by an image-generation model, so the figure shows the whole chain from text to a drivable rig; nothing about either stage depends on that, and Figs. 48–39 show the same pipeline on artwork found in the wild. The two layers marked in the stack had 33% and 42% of their area hidden by a layer above and therefore had to be completed rather than copied. The token strip's segment widths are the real per-layer vertex counts, the arcs under it are the three strongest measured cross-layer attention pairs, and 92% of that character's attention mass falls outside the within-layer blocks; measured on five characters the diagonal blocks hold between 4.6% and 6.7% of the matrix by area and never more than 12.8% of the mass. The fifth rig cell is a real composite render: the runtime resolves a simultaneous drive by summing the stored per-parameter fields, so three parameters at once needs no new prediction.

V.1 Problem formulation

Given a single illustration $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, we generate a structured Live2D asset

$$\mathcal{A} = \Big(\{\mathcal{L}_i\}_{i=1}^{N},\ \{\mathcal{M}_i\}_{i=1}^{N},\ \{\mathbf{k}_i^{p,0}, \mathbf{k}_i^{p,1}\}_{i \le N,\ p \in \mathcal{P}} \Big), \tag{23}$$

where $\mathcal{L}_i \in \mathbb{R}^{H \times W \times 4}$ is the i-th RGBA layer with hidden regions completed, $\mathcal{M}_i = (V_i, F_i)$ is its 2D triangle deformation mesh, $\mathcal{P}$ is the set of modelled animation parameters, and $(\mathbf{k}_i^{p,0}, \mathbf{k}_i^{p,1})$ are the start / end keypose vertex offsets of layer i under parameter p, so the offset table holds one pair per layer *and* per parameter rather than one pair per layer. Cubism also stores an interpolation type per keyframe; we do not model it and we generate linear tracks throughout, which is what the runtime equation of §V.4 assumes, so it is a constant of our output rather than a component of $\mathcal{A}$. $\mathcal{A}$ contains the fields used by our open WebGL renderer; it is not a native Cubism project and does not encode its full feature set. We decompose into two sequential subtasks handled by two specialised models:

$$\text{Stage 1:}\ \{\mathcal{L}_i\} \sim p_{\theta_1}\big(\cdot \,\big|\, \mathbf{I},\, N \big), \tag{24}$$

$$\text{Stage 2:}\ \mathcal{M}_i = \mathrm{Mesh}(\alpha_i),\quad \big\{\hat{\mathbf{k}}_i^{p,c}\big\}_{i \le N} = F_{\theta_2}\big(\{\mathcal{L}_i, V_i\}_{i \le N},\, p,\, \bar{q}_c\big), \tag{25}$$

A stored keypose offset $\hat{\mathbf{k}}_i^{p,c}$ is the collection of per-vertex offsets $\{\widehat{\Delta v}_{i,j}\}_{j \le n_i}$ emitted for layer i at normalised keypose value $\bar{q}_c$. Here N is the requested layer count and $c \in \{0, 1\}$ indexes the two extremes, $\bar{q}_c \in \{-1, +1\}$. Stage 1 samples a layer stack conditioned on the flattened illustration and requested count. Stage 2 instead uses deterministic meshing and regression. Its mesh $\mathcal{M}_i$ is a deterministic function of the layer's alpha channel α_i alone (§V.4), with no learned component and nothing sampled; its keypose offsets come from a single feed-forward pass of a regressor trained under an L_1 objective, so F_{θ_2} is a function and not a likelihood; the interpolation type is not predicted at all but fixed to LINEAR; and Stage 2 receives no text, its layer conditioning being the frozen image features of $\mathcal{L}_i$ (Eq. 27). One call of F_{θ_2} returns the field of *every* layer at once (Eq. 26).

V.2 Dataset construction and layer semantics

Sources and unified extraction. We collect publicly available Live2D assets from the internet, yielding 8,884 usable Live2D models after deduplication. Since the runtime binary format is encrypted, we render each model through the official Web SDK in a custom HTML renderer and record per-layer RGBA, art-mesh vertices and faces, and keyframed parameter curves. All vertices of a character are expressed in one *shared* canvas frame normalised to $[-1, 1]^2$, so that a layer's position relative to the rest of the character is part of its input rather than being lost to a per-layer crop; coordinates are kept as continuous floating-point values, because Stage 2 (§V.4) regresses displacements directly and therefore needs no quantisation.

V.3 Stage 1: layered diffusion adaptation

We initialise Stage 1 from Qwen-Image-Layered Yin et al. (2025); Wu et al. (2025) and fine-tune its diffusion transformer using pairs of flattened illustrations and artist-ordered RGBA target stacks.

The recorded training command passes the input illustration, target layer images, and requested layer count; it uses the standard flow-matching supervised fine-tuning loss at learning rate 10^{-5}. The target images include occluded painted regions, but the training code does not add an explicit occlusion loss, class-token hierarchy, or per-layer caption condition. More training details and the split-audit limitation are in Appendix B.

V.4 Stage 2: joint multi-layer keypose regression

The Cubism parameter grid. A Live2D rig is driven by named scalar *parameters* p (`ParamAngleX`, `ParamBodyAngleZ`, `ParamMouthOpenY`, ...), each with an artist-declared range $[p_{\min}, p_{\max}]$ and a rest value. For a given layer i and parameter p, the artist authors a small set of *keyposes* (parameter values $q^{(p,1)} < \ldots < q^{(p,K)}$ (typically $K \in \{2,3\}$: one extreme, rest, the other extreme)) and at each keypose displaces the layer's mesh vertices by hand. This layer-and-parameter-indexed table of vertex displacements is the *ParamVertex* track, and it is the entire animation content of the asset: at runtime the slider value is used only to linearly blend between the two bracketing keyposes (Eq. 33). Stage 2 therefore has to predict, for every layer, every parameter, and every keypose, a field of 2D vertex displacements.

Concretely, let $V_i = \{v_{i,1}, \ldots, v_{i,n_i}\} \subset [-1,1]^2$ be layer i's mesh vertices in the shared character canvas frame, and let $\bar{q} = 2(q - p_{\min})/(p_{\max} - p_{\min}) - 1$ be the normalised keypose value. We learn a single function that maps a *whole character* at one (p, q) to all of its per-vertex displacements at once,

$$\{\widehat{\Delta v}_{i,j}\}_{i \le N,\, j \le n_i} = F_{\theta_2}\Big(\{\mathcal{L}_i, V_i\}_{i=1}^{N},\ p,\ \bar{q}\Big), \tag{26}$$

and we predict continuous displacements directly rather than discretised tokens.

Content-conforming per-layer mesh. Each Stage 2 layer needs a base mesh. On rigged training characters we keep the artist's own vertices, which are the native support of the ground-truth displacement field. On a novel illustration there is no artist mesh, so we synthesise one from the layer's alpha channel alone: take every pixel with $\alpha > 4$ (out of 255) as content, dilate that silhouette by 3 px, sample *all* external contours (not only the largest, so a hair layer split into a lock plus a detached tail stays meshed) with a per-contour budget proportional to arc length, add a jittered interior lattice restricted to content pixels, and run Delaunay triangulation Barber et al. (1996). Roughly 55% of the vertex budget goes to the boundary. Two details of this recipe are load-bearing, and both were established by measurement rather than by design.

First, the threshold is pinned between two failures that pull in opposite directions, and it has to be low but *not* zero. Pushing it up fails because Stage 1 emits soft, anti-aliased layer borders: over the in-the-wild layers we measure, pixels with $0 < \alpha \le 16/255$ account for a mean of 10% of all visible pixels and up to 99.8% on a wispy layer, so a nominal cut such as $\alpha > 16/255$ meshes only the opaque core and leaves most of the drawn pixels outside the support; on our worst layer, coverage collapses to 0.18. Pushing it to zero fails for the opposite reason: Stage 1's diffusion decoder leaves an *imperceptible* noise floor of 1 to $4/255$ (under 1.6% opacity) across most of a layer's bounding box, so at $\alpha > 0$ the external contour degenerates to the bounding rectangle and the "content-conforming" mesh is a box. On real Stage 1 output, $\alpha > 0$ marks 78% of the bounding box as content while $\alpha > 4$ marks 52%. We therefore use $\alpha > 4$, which sits above the noise floor and below the perceptible rim ($\alpha \ge 8$): measured over 24 real Stage 1 layers, thresholds 0 and 4 give the *same* coverage of perceptible pixels (0.9995 against 0.9996) while the ratio of mesh area to bounding-box area falls from 0.998 to 0.788, i.e. only at $\alpha > 4$ does the mesh actually hug the silhouette. An earlier version of this paper argued for $\alpha > 0$ here; that was wrong for the reason just given, and the figures in this version are built at $\alpha > 4$.

Second, we retain *all* Delaunay faces. Throughout this paper *coverage* means one thing: the fraction of a layer's *perceptible* pixels ($\alpha \ge 8$, i.e. at least 3% opacity) that fall inside some triangle. Under that definition, discarding faces whose centroid falls off the dilated mask costs little on average, 0.9990 against 0.9996, but it is the tail that matters: the fifth percentile falls from 0.9986 to 0.9945, the worst layer from 0.9863 to 0.9682, and the number of layers below 0.99 goes from 1 to 4 of 151. A single uncovered rim on one layer is a visible white seam, so we pay the $\sim$21 extra faces per layer and keep them all. We note that an earlier version of this paper reported a far larger cost for the discard; that measurement used $\alpha > 0$ as the mask, which as explained above degenerates to the bounding box,

and we retract it. The discard also buys nothing visually, because a face lying over transparent texture rasterises to nothing. Retaining faces is therefore free while the gaps that discarding opens are not.

On rigged characters the artist's vertex set alone is likewise not a sufficient support: artists mesh inside the drawn silhouette, so a plain triangulation of their vertices covers a mean of 0.918 of visible pixels (minimum 0.576). We therefore *extend* that mesh for rendering (keeping every artist vertex, adding boundary vertices sampled from the dilated contour, and re-triangulating the union) which raises coverage to 0.9998 (median 1.000) on that condition. Two threshold details belong here so the numbers are comparable. The artist-mesh extension masks at $\alpha > 0$ while the generated mesh masks at $\alpha > 4$, and this is deliberate: a Stage-1 decoder leaves an imperceptible noise floor covering 26% of a layer's bounding box, whereas artist textures leave only 2.2%, so $\alpha > 0$ degenerates to the bounding box on the former but still follows the silhouette on the latter. Each added vertex inherits its displacement from the nearest artist vertex, and the model is never shown the added vertices, so artist-rig metrics remain defined on the artist's own vertex set. At deployment on a new illustration, the model instead receives vertices generated from Stage 1 alpha; this changes the vertex distribution. The artist-vertex test therefore does not directly measure the deployed mesh condition.

The 3 px dilation is not cosmetic: without it the mesh boundary sits exactly on the alpha edge, the centroid test deletes the rim triangles, and the uncovered opaque pixels render as thin white gaps along every layer border. Measured over 151 in-the-wild decomposed layers under the definition above, coverage rises from 0.9956 (median 0.9970, 14 layers below 0.99) without dilation to 0.9996 (median 1.000, 1 layer below 0.99) with it, which is what closes the gaps. Fig. 46 shows the effect directly: the generated mesh boundary lies just outside the drawn colour edge on every silhouette, and a layer that is split into several disconnected islands receives one mesh component per island. The alternative support is a uniform quad grid over the layer's bounding box, which is what our own earlier training packs used. We want to be precise about what the contour-conforming mesh does and does not buy over it, because one intuitive argument for it does not survive measurement. Over 46 in-the-wild layers the two supports are indistinguishable on placement quality: the fraction of vertices landing on content is 0.376 for the grid versus 0.368 for ours, the fraction of mesh area lying over transparent pixels is 0.574 for both, and the IoU between mesh footprint and silhouette is 0.426 for both. The reason is precisely the soft-alpha property above: scored at $\alpha > 0$, as that comparison was, a Stage 1 layer's content region nearly fills its bounding box, so a grid over that box is not obviously wasteful. What the contour-conforming mesh does buy is *efficiency at equal coverage*: it reaches the same coverage with 83 vertices and 112 faces against the grid's 112 and 182, i.e. 26% fewer tokens. Since every vertex is a token and attention is quadratic in sequence length, that is a direct saving in the dominant cost (§I). Tab. 25 additionally shows the contour-conforming vertex distribution is somewhat easier to learn a displacement field on, but that comparison is vertex-level only, each condition is evaluated on its own vertex set, and we draw neither a causal nor a rendering conclusion from it.

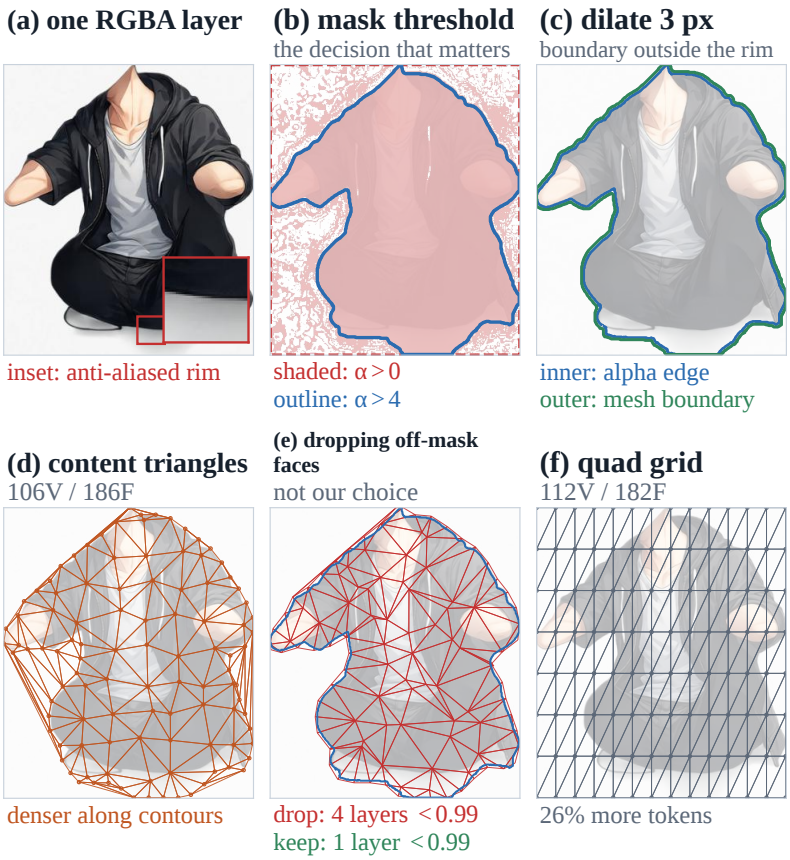

Figure 46: **Per-layer mesh construction, and what each choice avoids.** Each panel is one real Stage-1 layer, drawn by the code that builds the shipped meshes: (a) the RGBA layer, inset on its anti-aliased rim; (b) the $\alpha > 0$ mask (shaded) against $\alpha > 4$ (outline); (c) the 3 px dilation; (d) the triangulation; (e) keeping every Delaunay face against dropping off-mask faces; (f) a uniform quad grid at equal coverage. The numbers behind each choice are in the text.

Per-vertex conditioning. Every vertex of every layer becomes one token. Writing $\gamma_m(x) = \big[x,\ \{\sin(2^\ell \pi x), \cos(2^\ell \pi x)\}_{\ell=0}^{m-1}\big]$ for a Fourier positional encoding, the input embedding of vertex j of layer i is

$$h_{i,j}^{(0)} = W_v \Big[\gamma_6(v_{i,j});\ \mathbf{e}_p^{\text{par}};\ \gamma_3(\bar{q});\ \mathbf{e}_i^{\text{lay}}\Big] + W_a\, \phi(\mathcal{L}_i), \tag{27}$$

where $\mathbf{e}_p^{\text{par}} \in \mathbb{R}^{64}$ is a learned embedding of *which* Live2D parameter is being driven, $\mathbf{e}_i^{\text{lay}} \in \mathbb{R}^{64}$ is a learned layer-identity embedding indexed by the layer's position in the stack, $\phi(\mathcal{L}_i) \in \mathbb{R}^{384}$ is the frozen DINOv2-small Oquab et al. (2023) appearance descriptor of layer i (256 patch tokens mean-pooled), and W_v, W_a are linear maps into the model width $d = 256$.

How image evidence enters, and how it deliberately does not. The appearance term is the only path by which pixels reach Stage 2, so it is worth stating precisely what it is. Each layer is cropped to its alpha bounding box, aspect-padded into a 128×128 tile, bilinearly resized to the 224×224 the backbone expects, and passed once through a *frozen* DINOv2-small encoder; the resulting 16×16 grid of 256 patch tokens (ViT-S/14) is mean-pooled into a single 384-d vector, which is projected and **added to every token of that layer**. So a layer contributes one global appearance summary shared by all of its vertices, and a vertex is told what kind of part it sits on (long twin-tails rather than a rigid shoulder pad) but not what the texture looks like at its own position. There is deliberately *no* cross-attention between vertex tokens and image patch tokens, and no encoding of the composited

character image: the geometry stream and the appearance stream meet only through this additive summary.

That is a design decision we tested rather than assumed. We also implemented the richer alternative, bilinearly sampling the DINOv2 patch grid at each vertex's own position inside its layer tile, so every vertex receives a *local* appearance feature, plus a per-vertex alpha value and an area-weighted composite descriptor of the whole character as a global context token. On identical data and validation characters this hurts: cos 0.7269 against 0.7676, and magnitude calibration degrades badly (median ratio 1.735 against 1.234). Our reading is that per-vertex image features let the network key displacement on local texture detail, which does not transfer across characters, whereas one pooled descriptor per layer forces it to rely on part identity plus the geometric configuration; which does. We therefore keep the pooled form, and we report the negative result because it bounds how much of the remaining error can be blamed on weak image conditioning.

Joint multi-layer self-attention. The central design choice is that *all* layers of a character are concatenated into one sequence of $T = \sum_i n_i$ tokens and processed by a single Transformer, so self-attention spans layer boundaries:

$$H = \mathrm{Transformer}_{\theta_2}\Big(\big[h^{(0)}_{1,1}, \ldots, h^{(0)}_{1,n_1},\ \ldots,\ h^{(0)}_{N,1}, \ldots, h^{(0)}_{N,n_N} \big] \Big), \tag{28}$$

with 6 pre-norm blocks, 4 heads, and no causal mask (5.1 M trainable parameters in total). Each block is $x \leftarrow x + \mathrm{MHA}(\mathrm{LN}(x))$ followed by $x \leftarrow x + \mathrm{MLP}(\mathrm{LN}(x))$ with a $4\times$ expansion; padding is handled by a key-padding mask so batched characters of different sizes never attend across characters.

It is worth being concrete about what "spans layer boundaries" means, because it is the paper's central mechanism. The sequence is *vertices*, not layers: a 40-layer character with $\sim$80 vertices per layer yields $T \approx 3200$ tokens, and every one of the T^2 attention pairs is permitted. Grouping the tokens by layer partitions that attention matrix into an $N \times N$ arrangement of blocks. The N diagonal blocks are within-layer attention; and are *all* a per-layer model can ever use, since it processes each layer in isolation. The off-diagonal blocks are exactly what the joint formulation adds: they let an iris vertex read the position of the eye-white vertices it must stay inside, and a fringe vertex read the forehead it must stay attached to. Because there is no causal mask, the coupling is bidirectional, the eye white is equally free to condition on the iris. The layer-identity embedding $\mathbf{e}^{\mathrm{lay}}_i$ is what makes these blocks addressable: it is indexed by depth position in the stack, so attention can be modulated by *which* layer a token belongs to and by how far apart in depth two layers are, rather than treating the sequence as an unordered bag of vertices. Nothing in the mechanism is layer-count-specific, so a character with 8 layers and one with 125 use the same weights.

Predicting each layer independently gives that model no direct access to other layersḿotion context. Joint attention permits such information flow, which could help an iris track its eye white or a fringe track the forehead; it cannot guarantee seam consistency. The saved independent and joint implementations differ in other design choices, so their matched evaluation in Tab. 3 does not isolate attention scope. The cost is that attention is quadratic in T: measured on one GPU, going from 1052 to 5476 tokens ($5.2\times$) raises peak activation memory $24.7\times$, and the joint forwards are the only phase of Stage 2 whose cost grows appreciably with character complexity (§I). This is also why the token saving from the contour-conforming mesh matters. Fig. 3 shows the token construction that makes this possible; the block structure of the attention matrix itself is quantified above.

The attention operator, and what per-layer prediction is in these terms. It is worth writing the operator out, because it makes the ablation exact rather than rhetorical. Let $H \in \mathbb{R}^{T \times d}$ stack the current token states, $d = 256$, and let $S_i \subset \{1, \ldots, T\}$ be the index set of layer i's vertices, so $\{S_i\}_{i=1}^{N}$ partitions the sequence and $|S_i| = n_i$. Each of the 4 heads projects $Q^{(k)} = HW^{(k)}_Q$, $K^{(k)} = HW^{(k)}_K$, $V^{(k)} = HW^{(k)}_V$ with $W^{(k)}_\bullet \in \mathbb{R}^{d \times d_h}$, $d_h = 64$, and forms

$$\begin{aligned} A^{(k)} &= \mathrm{softmax}\Big(\tfrac{Q^{(k)} K^{(k)\top}}{\sqrt{d_h}} + M \Big) \in \mathbb{R}^{T \times T}, \\ M_{ab} &= \begin{cases} 0 & b \text{ real} \\ -\infty & b \text{ padding}, \end{cases} \end{aligned} \tag{29}$$

the only mask being the padding mask that keeps batched characters from attending to one another. Reading $A^{(k)}$ through the partition gives N^2 sub-blocks $A^{(k)}[S_i, S_j]$: the N diagonal blocks

$A^{(k)}[S_i, S_i]$ are within-layer attention, and the $N^2 - N$ off-diagonal blocks $A^{(k)}[S_i, S_j]$, $i \neq j$, are cross-layer attention. Per-layer prediction prevents these off-diagonal interactions. Our implemented per-layer baseline, however, also differs in its appearance encoding and magnitude head, so the measured comparison is not a mask-only ablation. Concretely, on a 40-layer character with $\sim$80 vertices per layer ($T \approx 3200$), the diagonal blocks account for only $\sim 2.5\%$ of the T^2 entries, so the joint formulation makes roughly 97.5% of the attention budget available to inter-layer reasoning that a per-layer model structurally cannot access.

Two properties of this design are worth making explicit. First, the operator is *permutation-equivariant* over tokens: no sequence-position encoding is used anywhere, so nothing depends on the order in which vertices are concatenated. The grouping into blocks above is a way of *reading* the attention matrix, not a constraint imposed on it. A token's identity is carried entirely by its input embedding (Eq. 27) (its canvas position, the driven parameter, the keypose value, its layer-identity code) which is what lets the same weights serve characters with 8 and with 125 layers, and what lets the layer-identity embedding modulate attention by depth relationship rather than by arbitrary index. Second, because a vertex's canvas coordinate is its own positional signal and all layers share one canvas frame, geometric adjacency across layers is directly available: an iris vertex and the eye-white vertices behind it are nearby in the same coordinate system, so attention can key on spatial proximity across a layer boundary without any explicit correspondence being supplied. This is the mechanism behind the coordination effect; the model is never told which layers are related, only that they inhabit one frame.

Two things this design deliberately does *not* do, for completeness. It does not attend over image patches (§V.4, "How image evidence enters"), and it does not attend over parameters or keyposes: each $(p, \bar{q})$ is an independent forward pass, so the model cannot reason jointly about, say, a head turn and a body lean. Composite poses are instead formed at runtime by summing the per-parameter displacement fields, which is the same superposition the Cubism runtime performs and is therefore the correct behaviour for our target format, but it does mean genuinely coupled multi-parameter deformation is outside the current formulation.

Decoupled direction and magnitude. Live2D displacement magnitudes are extremely heavy-tailed: most vertices move by well under 1% of the canvas while a large head turn moves silhouette vertices by tens of percent (Appendix H). Regressing raw displacements therefore lets a handful of large-motion vertices dominate the loss. We instead factor each displacement into a bounded shape term and a log-scale term. With $a_i = \max_j \|\Delta v_{i,j}\|_\infty$ the layer's peak displacement at the current $(p, \bar{q})$ and $\bar{u}_{i,j} = \mathrm{clip}(\Delta v_{i,j}/a_i, -1, 1)$ the scale-normalised target, two small MLP heads read every token,

$$u_{i,j} = \tanh\big(f_{\text{dir}}(H_{i,j})\big) \in [-1, 1]^2, \qquad s_{i,j} = f_{\text{mag}}(H_{i,j}) \in \mathbb{R}, \tag{30}$$

and the predicted offset is recomposed multiplicatively,

$$\widehat{\Delta v}_{i,j} = u_{i,j} \cdot \exp\big(s_{i,j}\big). \tag{31}$$

The $\tanh$ makes the shape term exactly as bounded as its target, and putting the scale in $\log$ space makes a $2\times$ magnitude error cost the same whether the true motion is 0.002 or 0.2.

Training objective. Both terms are trained with L_1 losses, masked to real (non-padding) vertices:

$$\mathcal{L}_{\text{Stage 2}} = \mathbb{E}_{i,j}\Big[w_i \,\big\|u_{i,j} - \bar{u}_{i,j}\big\|_1\Big] + \lambda_a \,\mathbb{E}_{i,j}\Big[w_i \,\big|\, s_{i,j} - \log a_i \,\big|\Big], \tag{32}$$

with $\lambda_a = 1$ and per-layer weights $w_i \propto a_i^\rho$ normalised to unit mean. The default model uses $\rho = 0$ (uniform weighting); $\rho > 0$ up-weights large-amplitude layers and is ablated in Tab. 26. We train 30 epochs of AdamW at lr $2{\cdot}10^{-4}$ on one H800. For a bounded sequence length, training subsamples at most 28 vertices per layer and 48 layers per character (cap $T \leq 1200$); at inference the model runs on the full vertex set of every layer, which the attention handles without retraining because Eq. 27 contains no absolute token index.

Inference and runtime interpolation. Inference is one forward pass per $(p, \bar{q})$ over the whole character, no autoregressive rollout, hence no exposure-bias gap between training and test, and every number we report in §W is measured in this true-generation regime. Sweeping the parameters we model over their keyposes yields the complete $\{\mathbf{k}_i^{p,0}, \mathbf{k}_i^{p,1}\}$ table of Eq. 23; the released models cover

the 8 parameters of §W.3 and the 24-parameter extension of Appendix R. Writing $\hat{V}_i^{(p,k)} = V_i + \hat{\mathbf{k}}_i^{p,k}$ for the deformed vertex positions that a stored keypose offset produces, the Live2D Cubism viewer (or our WebGL renderer) drives per-vertex linear interpolation between the two keyposes bracketing the slider value q,

$$V_i(q) = (1-t)\,\hat{V}_i^{(p,k)} + t\,\hat{V}_i^{(p,k+1)}, \;\; t = (q - q^{(p,k)})/(q^{(p,k+1)} - q^{(p,k)}), \tag{33}$$

followed by per-triangle rasterisation with premultiplied alpha. This is the LINEAR case, which is the only one we generate; a STEP track replaces the blend by $V_i(q) = \hat{V}_i^{(p,k)}$ for $q < q^{(p,k+1)}$.

A property the representation gives us for free: texture editing. Because a layer's mesh UVs are derived from its alpha bounding box, any edit that preserves a layer's alpha channel and pixel dimensions leaves the mesh (and hence the already-predicted animation) exactly valid. This is a property of the output representation rather than a component we train: it means an off-the-shelf instruction-guided image editor Qwen Team (2025) can repaint one clothing layer's RGB, and once the original alpha and size are restored the rig is re-exported with the new texture and *no* re-animation. We treat this as an application of the generated asset and report it in §W.6; nothing in Stage 1 or Stage 2 is modified for it.

W EARLIER EXPERIMENTS AND RESULTS

W.1 SETUP

Dataset. Our corpus contains a diverse collection of Live2D models (§V.2), categorized into human / humanoid and non-human classes by Qwen3.6-Plus tags. Model-action replacement and color / texture replacement significantly expand the Stage 1 supervision and the animation subset before per-layer / per-parameter expansion. All specific numerical statistics regarding the dataset scale are summarized in Table 22. To mitigate computational overhead from extreme layer counts, we systematically compress the training set. Specifically, we merge layers based on inherent Live2D semantic groupings while strictly preserving the original PSD depth order. This reduces structural complexity without compromising vital occlusion relationships. For comprehensive details regarding our data augmentation strategies and the statistical distribution of the dataset, please refer to Appendix D.

Dynamic resolution and layer grouping. To efficiently balance computational cost and generation fidelity, we employ a dynamic, layer-count-aware resolution scaling strategy. Instead of rigid resolution buckets, we allocate a target pixel budget based on the effective output layer count L, preserving the original aspect ratio.

The target pixel budget P_{target} is inversely proportional to L, clamped by empirical bounds:

$$P_{\text{target}} = \max\left(196608, \min\left(1048576, \frac{6400000}{L+2}\right)\right)$$

To prevent upsampling artifacts, the scaling factor s is bounded by the original pixel count $P_{\text{orig}} = W_{\text{orig}} \times H_{\text{orig}}$ of the input illustration:

$$s = \sqrt{\frac{\min(P_{\text{target}}, P_{\text{orig}})}{P_{\text{orig}}}}$$

Target dimensions are then uniformly scaled and rounded to the nearest integer: $W_{\text{target}} = \text{round}(W_{\text{orig}} \cdot s)$ and $H_{\text{target}} = \text{round}(H_{\text{orig}} \cdot s)$.

Live2D-Bench. Existing benchmarks stop at either image-to-multi-layer decomposition or raster / video animation; none evaluates whether a method generates a complete Live2D asset. We therefore build *Live2D-Bench*, a unified benchmark that combines image-to-multi-layer evaluation for semantic RGBA decomposition with per-layer mesh-animation evaluation for topology, keypose offsets, and rendered motion. The fixed pool has 120 examples: 100 human / humanoid and 20 non-human cases, stratified by layer count into 10 to 20, 20 to 35, and 35+ bins in a 1:2:1 ratio. For more details, please refer to Appendix E.

Table 22: Live2D corpus and benchmark composition.

Quantity	Count
Raw usable Live2D models	8,884
Human / humanoid models	7,773
Non-human models	1,111
Augmented layer-decomposition examples	$\sim$ 50,000
Animation examples before per-layer expansion	$\sim$ 35,000
Live2D-Bench examples	120
Human / humanoid benchmark examples	100
Non-human benchmark examples	20
Layer-count bins (10 to 20)/(20 to 35)/(35+)	30/60/30

Baselines. Stage 1 baselines are depth-based segmentation using Marigold-depth Ke et al. (2024), SAM segmentation without inpainting Kirillov et al. (2023); Carion et al. (2025), zero-shot Qwen-Image-Layered Yin et al. (2025), and See-through Lin et al. (2026). Specifically, for Marigold-depth,

we employ the original model and partition the estimated depth into discrete layers according to the ground-truth layer count. Our SAM baseline utilizes the fine-tuned model introduced in See-through, which performs a 19-class segmentation based on predefined semantic labels. For See-through itself, we adopt its latest v3 release for evaluation. Stage 2 baselines cover oracle classical fitters (rigid, FFD Sederberg & Parry (1986), ARAP Sorkine et al. (2007)), image-blind retrieval, DINOv2-conditioned regressors, and image-to-video systems Xing et al. (2024); Wan et al. (2025); Yang et al. (2025b); Meng et al. (2025).

Metrics. We report three metric families: full-image / layer-level scores, mesh-level cosine / magnitude / RMSE / PCK, and animation-level warped-frame L_1 / PSNR / SSIM / LPIPS over 24-frame Body and Face Angle X loops. Crucially, to evaluate predicted outputs with variable layer counts against the ground truth, our layer-level evaluation employs the Hungarian algorithm (balancing α-IoU and RGB L_1 distance) to establish optimal bipartite layer matching. This design allows us to simultaneously assess layer ordering correctness and compute per-layer fidelity metrics. Furthermore, we introduce a novel *per-pixel Cov-MAE* metric to explicitly quantify the granularity of the image decomposition. The Live2D-Bench HTML inspector complements the automatic metrics. Detailed formulations for all evaluation metrics are provided in Appendix E.

Implementation. Stage 1 uses a full-parameter checkpoint labeled step 6000 from a resumed Qwen-Image-Layered fine-tuning run; the checkpoint filename does not establish the cumulative update count across resumes. Stage 2 is the 5.1 M-parameter joint Transformer of §V.4 ($d = 256$, 6 blocks, 4 heads), trained from scratch for 30 epochs of AdamW at lr $2{\cdot}10^{-4}$ on one H800 over $1{,}443$ characters and $357{,}381$ (character, parameter, keypose, layer) records, with 50 characters held out. Stage 2 inference is a single forward pass per (parameter, keypose) pair over the entire character (all layers at once, no autoregressive rollout) and the generated bundle plays in our WebGL renderer at 30 fps. Stage 2 end-to-end cost, timed phase by phase with `cuda.synchronize()` around each phase and the median of nine repetitions reported, is 0.71 s for a small character (10 layers, $T = 1052$ vertex tokens) and 2.84 s for a large one (63 layers, $T = 5476$), with a real Stage-1 output (19 layers, $T = 1873$) at 1.39 s. Peak activation memory is 189 MiB and 1982 MiB respectively on top of 103.9 MiB of resident weights (19.4 for the joint model, 84.5 for the frozen DINOv2), so the whole of Stage 2 fits comfortably on a consumer card. The $5.2\times$ token growth costs $24.7\times$ activation memory, which is the expected $O(T^2)$ attention scaling and the one place where very large characters will eventually need windowing. Of the large character's 2.84 s, the joint forward passes are 37%, DINOv2 feature extraction 22%, rig serialisation 20%, image I/O 19% and *mesh construction only* 2.7%: the triangulation whose design occupies §V.4 is computationally free. A complete rig is 13 forward passes (8 parameters, three keyposes for the five angle parameters and two for the three $[0, 1]$ parameters, with the rest pose needing none), each pass covering *all* layers of the character at once. These were measured on a shared A800 with other tenants occupying 58.5 GB and the contention is visible in the spread (the small case's forward phase ranges 0.16 to 0.35 s), so the minima are the better estimate of an uncontended run; full per-phase tables are in Appendix O.

W.2 Stage 1: layer decomposition

Live2D-Bench scores layer stacks with full-image, Hungarian-matched per-layer metrics. Tab. 23 reports a representative subset.

Three points stand out. First, full-image metrics are misleading on their own: depth segmentation obtains strong PSNR / SSIM by producing a few globally similar layers whose composite matches the input, but its penalised αmIoU is only 0.081. Second, unmatched layers matter; See-through's matched αmIoU drops once its missing layers are penalised. Third, on this fixed diagnostic pool, Stage 1 has the best reported layer-aware metrics: penalised αmIoU 0.298, penalised cost 0.491, order 0.674 ($+56\%$ over the best baseline), and the best Group III scores. Matched-layer LPIPS 0.055 is 14% better than See-through and 42% better than SAM segmentation.

W.3 Stage 2: mesh and animation

Protocol and metric. Stage 2 is evaluated in the regime in which it is deployed: *true generation*. For each character excluded from gradient training, we feed the model the artist's layers and mesh vertices together with a (parameter, keypose) pair, take one forward pass, and compare the predicted



Figure 47: Stage 1 qualitative results on Live2D-Bench. Each example shows how the input illustration is decomposed into an ordered Live2D layer stack with editable RGBA textures and completed occluded regions, rather than a single flat segmentation mask.

Table 23: Historical Stage 1 training-source diagnostic on Live2D-Bench: all 120 recovered source names match the final Stage 1 training manifest. Group I is full-image composite; Group II is per-layer Hungarian matching with unmatched-layer penalties; Group III is global set-theoretic. These rankings do not establish unseen-source generalization.

Metric	Depth seg.	SAM seg.	Qwen-Image-Layered	See-through	Ours
Group I: full-image (composite vs. input)					
PSNR ↑	**28.24**	25.58	16.58	19.88	24.28
SSIM ↑	**0.977**	0.953	0.767	0.890	0.813
LPIPS ↓	**0.061**	0.082	0.241	0.198	0.149
α-IoU ↑	**0.957**	0.851	0.916	0.873	0.950
RGB-L1 ↓	**0.0161**	0.0520	0.2206	0.1419	0.0627
Group II: per-layer					
matched-layer LPIPS ↓	0.210	0.066	0.143	0.064	**0.055**
penalised α-IoU ↑	0.081	0.137	0.175	0.265	**0.298**
penalised cost ↓	0.546	0.616	0.573	0.640	**0.491**
Order (pair-weighted) ↑	0.391	0.302	0.431	0.266	**0.674**
Cov-MAE per-pixel ↓	0.166	0.197	0.445	0.167	**0.090**
Mask Dice (penalised) ↓	0.867	0.799	0.725	0.653	**0.514**

Table 24: Stage 2 mesh-animation results under **true generation** on the 50 validation characters, which were used for checkpoint selection. Metrics per §W.3. The FFD rows are expressiveness oracles, not competing methods; see the text.

	dir-cos ↑		mag	chars
Method	mean	med.	med. → 1	≥ 0.8
Ours (joint, 5.1 M)	**0.7676**	**0.8278**	1.234	**31**/50
Expressiveness oracles on the same pool and metric[†]				
FFD 2×2 lattice oracle	0.9909	0.9955	0.989	46/46
FFD 3×3 lattice oracle	0.9982	0.9992	0.998	46/46
FFD 4×4 lattice oracle	0.9993	0.9997	1.000	46/46

displacement field against the artist's. No ground-truth displacement is fed back and no teacher forcing is used. These are artist-layer validation results, however, because the same 50 characters were used for checkpoint and architecture selection; generated meshes and layers are evaluated separately. The primary metric is the *per-vertex direction cosine* between predicted and ground-truth displacement, restricted to vertices the artist actually moves ($\|\Delta v\| > 0.005$ of canvas extent, since the direction of a numerically zero displacement is undefined), averaged within a (layer, parameter, keypose) triple, then within a character, and finally reported as a mean and median over characters. The companion metric is the *magnitude ratio*, the mean predicted displacement norm over the mean ground-truth norm on the same vertices, whose ideal value is 1.0: below 1 means the motion is too timid, above 1 too violent. For distributional context we also count characters above 0.80 cosine; this threshold has not been validated as an artist-usability criterion.

Why we do not rank image-to-video baselines against the current rig model. An earlier autoregressive Stage 2 was evaluated against ToonCrafter, Wan2.2-I2V, CogVideoX-I2V, and AniDoc on a 120-example image-to-video protocol. The current joint Stage 2 has instead been rendered on three artist-rigged characters, with a different crop normalization and different keyposes. The old pixel scores therefore cannot be used as a baseline ranking for the current model. On the three-character protocol, even a zero-motion control reaches 41.6 dB against 41.8 dB for predicted motion on artist layers, making pixel PSNR especially insensitive to animation. A matched, motion-sensitive comparison with video systems remains future work; we assess the current motion model on vertex displacements and report the small end-to-end render test separately.

Tab. 24 reports the per-vertex direction cosine (mean and median over characters), the magnitude ratio (ideal 1.0) and the number of characters whose cosine reaches 0.80, on the 50 validation characters, which have zero overlap with gradient training but were used for model selection.

The FFD rows need reading carefully, because they are not competing methods. Each is handed the artist's own displacement field and fits a free-form-deformation lattice to it by least squares, so it measures what a classical lattice rig can *express*, not what anything can predict from an image; they are measured on the identical pool through the identical metric code path as the row above. The result is worth stating plainly: even a 2×2 lattice reproduces the artist's per-layer field to a direction cosine of 0.99, so the representation is essentially never the limiting factor and the whole gap to 1.0 in the first row is a *prediction* gap. Four of the 50 validation characters carry no records in this pack and are absent from every FFD row. A per-vertex *independent* regression baseline is reported separately in Tab. 21, trained on the same data as our model rather than borrowed from an earlier run.

With only (layer stack, mesh vertices, parameter id, keypose value) as input and a single forward pass, Stage 2 reaches a per-vertex direction cosine of 0.7676 on average and 0.8278 at the median across the 50 validation characters, and 31 of the 50 (62%) clear 0.80. The median-versus-mean gap indicates a left tail of low-cosine characters; the chosen 0.80 cutoff has not been calibrated to artist usability. Amplitude is a weaker axis (the median magnitude ratio is 1.234), and this aggregate hides a systematic effect discussed below: small motions are over-shot while large turns are under-shot. We stress that 0.7676 is a *true-generation* number. Our own earlier autoregressive token model reported cosines above 0.99 under teacher forcing, where the ground-truth prefix is fed back at every step; that setting measures next-token accuracy, not the quality of a generated rig, and is not used for the current claim. Renderer aliasing came from texture minification without a mip chain rather than from missing multisampling; adding mipmapping reduces edge-band error against a $16\times$-supersampled reference by $2.34\times$ (Appendix Q). The primary geometry metric is independent of that rasterizer. Earlier video-baseline results use a different Stage 2 and cannot rank the current model; current pixel results also require the zero-motion controls in Appendix S. Qualitative comparison is in Fig. 55.

Mesh representation: content-conforming triangles are the easier support to learn on. The mesh is the support on which the whole animation lives, so we compare the two natural choices under otherwise identical conditions, the same 50 validation characters, the same 5.1 M architecture, the same objective and schedule, with only the mesh representation on which the model is trained and evaluated changed (Tab. 25). Content-conforming triangulation is ahead on direction (0.7676 vs. 0.7542 mean, 0.8278 vs. 0.8049 median), but the 0.013 mean gap is *inside* the 0.024 seed spread established above, so this pair of single runs does not establish an ordering on direction alone. The count above the chosen 0.80 threshold is descriptive, not a validated usability measure: 31 vs. 26 of 50 clear it. The quad grid is better on amplitude calibration (1.116 vs. 1.234, closer to the ideal 1.0). Taken together, these data show a modest vertex-level difference and a clearer token-cost advantage for triangles, not a demonstrated usability advantage. We measured both meshes on 46 validation layers and the fraction of vertices landing on opaque content is similar (0.376 ± 0.152 for the grid against 0.368 ± 0.142 for ours), as are mesh area over transparent region (0.574 both) and intersection over union with the silhouette (0.426 both); a quad grid does *not* waste vertices on emptiness, because a layer's alpha bounding box is mostly filled by that layer. At equal visible coverage, content-conforming triangulation uses 83 vertices and 112 faces per layer where the grid needs 112 and 182, i.e. 26% fewer tokens for the same support, and places its budget densely along contours and sparsely in flat interiors.

Two caveats bound what this experiment establishes, and we state both plainly. First, the per-vertex metric is computed on each condition's *own* vertex set (artist vertices vs. resampled grid vertices), so this is not a vertex-identical comparison; it is the same characters, task, objective and architecture, and the artist mesh is the native support of the ground truth while the grid is our synthetic resampling of it, but the two rows are not two readings of one ruler. Second, and importantly, **this is a vertex-level result only and we make no rendering claim from it**: the artist-mesh data pack stores the artist's vertices and displacements but no per-layer UV / texture correspondence, so artist-mesh rigs cannot be rasterised faithfully; we tried, and the renders show visible edge truncation. Every rendered figure and every qualitative comparison in this paper therefore comes either from the grid-mesh rigs, whose UVs are exact by construction, or from the in-the-wild pipeline, where the mesh vertices are generated *from* the layer's own alpha channel and coverage of perceptible pixels is 0.9996 on average with a median of 1.000 (§V.4). The mesh ablation should be read as evidence about which vertex distribution is easier to learn a displacement field on, not as evidence about final image quality.

The under-segmented stacks in Fig. 4 fail in the ways a rigger would predict: `wild2_chibi` (10 layers) emits the paper sheet behind the character as one opaque grey layer that then animates as if it

Table 25: **Ablation A, mesh representation.** Identical 50 validation characters, identical 5.1 M architecture, identical objective and schedule; only the mesh changes. True generation. Two caveats: each row is measured on its own vertex set, so the comparison is same-character and same-task but not same-vertex; and this is a *vertex-level* result only, the artist-mesh pack carries no per-layer UV correspondence, so it supports no claim about rendered image quality.

	dir-cos ↑		mag	chars
Mesh	mean	med.	med. → 1	≥ 0.8
Content-conforming triangles (ours)	**0.7676**	**0.8278**	1.234	**31**/50
Quad grid re-mesh	0.7542	0.8049	**1.116**	26/50

were the body; `wild2_kimono` (13) loses the kimono torso and legs; `w3_hoodieboy` (12) loses both legs and detaches a shoe; `webB_chibi` (11) loses the boots and hair ornament; `webC_chibi2` (9) loses the fur cuff; `w3_witch` loses the hat; `w3_goth` loses the head-bow and exposes a source watermark our stack keeps in a back layer. Our stacks stay complete on all 13. See-through is used as published with no tuning by us and was not designed for rigging, so this is evidence that decomposition quality dominates the end-to-end result, not a claim about that method's own task.

Stage 1 error propagation on in-the-wild illustrations. The end-to-end question a user cares about is whether the pipeline works on art that was never a Live2D model. We collected 13 real web illustrations with no associated rig, decomposed each one *twice* (once with our Stage 1 and once with the third-party See-through decomposer Lin et al. (2026)) and animated all 26 resulting layer stacks (446 layers) with the identical frozen Stage 2 model, adding 5 further illustrations processed by our Stage 1 alone for a total of 31 rigs and 541 layers. Because the animation model, its weights, the mesh procedure and the parameter sweep are held fixed, any difference between the two conditions is attributable to the decomposer alone, which is exactly the Stage-1-to-Stage-2 error-propagation channel: Stage 2 can only move what Stage 1 gave it, so a layer that was cut through the middle of a sleeve, or whose occluded region was left hollow, tears or reveals background no matter how good the predicted displacement field is. Fig. 4 shows paired examples at one identical parameter value, and the failures are exactly of this kind: an under-segmented stack loses a limb, a garment, or the character/background separation, and the animation model can only deform the parts it was handed. The quantitative version of this experiment, on the three held-out characters that ship an artist rig so that a ground-truth animation exists, is reported in Appendix S: with layer source as the only variable, our Stage 1 reaches rendered PSNR 23.4 / SSIM 0.883 / LPIPS 0.088 against See-through's 21.1 / 0.865 / 0.120, and a rest-pose control shows that much of the rendered difference is already present before motion. A separate eight-rig validation control finds that generated-layer motion is less accurate than keeping the same stack still under a rest-subtracted metric, so these three-character pixel scores do not establish motion fidelity. One protocol detail matters for fairness: See-through emits a 1280^2 padded canvas whereas our Stage 1 emits a tight crop, so we normalise every rendered canvas to the character's own content bounding box. Without that normalisation a side-by-side would simply shrink whichever method pads more, which would flatter our method for the wrong reason. Additional raw renders, generated meshes, and failure cases are shown in Appendices M–S; the displayed examples were not individually repaired.

W.4 Ablations

Joint versus independent implementations. On the clean 46-character benchmark under true generation, the joint implementation scores 0.736 direction cosine against 0.693 for the independent per-layer implementation and improves the magnitude ratio from 0.86 to 0.99 (Tab. 21). These figures came from a record-averaged evaluation that requires both predicted and artist movement above 0.001 of canvas extent. They do not use the character-averaged 0.005 ground-truth-only gate of the separate 50-character headline test. The independent implementation also uses DINO patch cross-attention and a pooled magnitude head, whereas the joint implementation uses pooled layer appearance and a vertex magnitude head. Consequently this comparison supports the complete joint design, but it does not isolate cross-layer attention as the sole cause of the gain. A matched-record, character-level reanalysis is described in Appendix E.

Table 26: **Ablations B and C** on the clean 46-character benchmark (zero training overlap), true generation, same data / objective / 30 epochs throughout. Numbers are comparable *within* this table only: this pool and mesh pack differ from Tab. 24, which is why the 5.1 M row reads 0.7397 here and 0.7676 there.

	dir-cos ↑		mag	chars
Variant	mean	med.	med. → 1	≥ 0.8
B: model capacity (identical data and objective)				
5.1 M (*d* 256, 6 blk; ours)	**0.7397**	**0.7944**	1.130	**22**/46
38.7 M (*d* 512, 12 blk)	0.6784	0.7605	0.975	20/46
115.0 M (*d* 768, 16 blk)	0.7333	0.7748	1.063	**22**/46
204.1 M	0.7097	0.7807	1.171	20/46
571.6 M	0.6853	0.7565	**1.008**	19/46
1.0 B	diverged (val cos −0.085)			
C: amplitude reweighting $w_i \propto a_i^\rho$ (Eq. 32)				
$\rho = 0$ (default)	0.7397	0.7944	1.130	22/46
$\rho = 0.25$	0.7392	n/a	1.150	n/a
$\rho = 0.5$	0.7301	n/a	1.247	n/a

How large a difference is meaningful here. Before reading any ablation it is worth establishing the noise floor, because several of the differences below are small. Two runs of the committed configuration that differ *only* in random seed score 0.7289 and 0.7533 on the clean 46-character benchmark, a range of 0.024. We therefore treat a mean-cosine difference below roughly 0.02 as uninformative on its own, and we say so at each point where a comparison falls inside that band rather than reporting the ordering as if it were established. Two caveats bound this estimate itself. It comes from two replicates, so it is a crude estimate of a standard deviation rather than a confidence interval, and a third run of a related configuration scores 0.6918 but has no surviving training log, so we exclude it rather than use an unverifiable run to widen the floor. An earlier version of this paper quoted ≈ 0.006 for this quantity; that figure came from a per-record metric rather than the per-character metric we report, and we retract it.

What the attention actually does. The formalisation above says per-layer prediction is this model plus a block-diagonal constraint, but it does not say whether the model uses the freedom that removing the constraint gives it. That is measurable, so we measured it: we captured every block's attention weights on a real character (15 layers, $T = 1435$ vertex tokens) and computed how much attention *mass* falls inside the diagonal blocks against how much of the matrix they occupy. The diagonal blocks are 6.7% of the matrix by area, and they receive 4.4% of the mass in the first block, rising monotonically to 12.2% in the sixth: on average 91.9% of the attention mass is cross-layer. Two readings are worth separating. The first is that attention is *diffuse rather than diagonal*, so the model is not quietly recovering the per-layer baseline inside a joint architecture. The second is a depth trend we did not anticipate: early blocks attend broadly across the whole character and later blocks progressively localise, which is the pattern one would expect if the early layers establish a global frame and the later ones refine within a part. We are careful not to over-read the first number: a uniform attention would also put mass in proportion to area, so this measurement shows the cross-layer capacity is used, not that using it helps. The implementation comparison, 0.693 against 0.736, is suggestive but also changes the appearance and magnitude heads; a mask-only training control is still needed to attribute its gain to attention scope. Reading the attention matrix directly on a validation character (15 layers, $T = 1435$), the within-layer diagonal blocks occupy 6.7% of the matrix by area and receive only 4.4% of the attention mass in the first transformer block, rising to 12.2% in the last: attention is diffuse rather than diagonal, and it localises with depth rather than starting localised. Some layers are also attended to by all the others, appearing as bright columns in the matrix.

Capacity under the tested schedule. Ablation B scales the same architecture, data and objective across six points from 5.1 M to 571.6 M parameters (a 112× range) and direction accuracy never improves: 0.7397 at 5.1 M, 0.7333 at 115.0 M, 0.7097 at 204.1 M, 0.6853 at 571.6 M. Individual adjacent gaps here are comparable to the 0.024 seed floor, so we do not claim that any one pair is separated; the finding is the *absence of improvement across two orders of magnitude* together with outright divergence at the top, which we cannot attribute with confidence from the available seed count. The smallest model is the best on both mean and median cosine and ties for the most characters

above 0.80, and the trend beyond 115 M is degradation; interestingly the largest converged models are the best-calibrated on amplitude (1.008 at 571.6 M, 0.975 at 38.7 M), so scale buys magnitude at the cost of direction. At 1.0 B the run diverged outright (validation cosine -0.085), so the extra capacity destabilised optimisation rather than helping it. Training loss keeps falling as models grow while validation cosine does not, which is consistent with a generalisation gap under this optimization schedule. Three plausible causes of the observed ceiling are task ambiguity, decomposition quality, and limited rig diversity; the sweep does not isolate their contributions. First and most fundamentally, the task is *ambiguous*: one still image plus one parameter value is consistent with many plausible artist motions, and a pointwise regression loss can only recover a central tendency of that conditional distribution, the median for the L_1 objective we use, the mean for an L_2 one. Averaging over plausible motions is exactly what produces amplitude compression on the large turns where artists disagree most, and a larger model predicts that same central tendency more precisely rather than escaping it. Second, Stage 1 decomposition quality caps what Stage 2 can express. Third, rig-data diversity is limited: $1{,}443$ characters is large for this domain but small in absolute terms, and the 8 parameters we model are the common subset rather than the full Cubism vocabulary. Promising next tests include distributional motion predictions, improved layers, and more diverse rigs; the present sweep does not rule out better optimization of a larger network.

Conditioning signals that did not help. Beyond the ablations above we tried four further conditioning signals, three of them suggested by reviewers of an earlier version, and none exceeded the seed noise floor: explicit per-vertex draw-order conditioning (0.7148), a zero-initialised 2.5D parallax residual (two seeds, 0.7225 and 0.7170), the two combined at 40 epochs (0.7153), and per-vertex image-feature sampling in place of one pooled vector per layer (0.6918). Two baseline seeds give 0.7289 and 0.7533 on the same benchmark, so the noise floor is 0.024 and none of these moves the mean beyond it. The parallax variants are the interesting near-miss: parallax with draw order reaches the best median (0.8255), the most characters above 0.80 ($25/46$) and by far the best amplitude (1.054 against 1.238), i.e. it improves the typical character and the amplitude while losing badly on a few, which is the one negative result that points at our amplitude-compression failure mode. Full table, per-variant reasoning, and a measurement error we made and corrected are in Appendix P.

Amplitude reweighting is a wash. Since amplitude is the weaker axis, we tried the obvious fix of up-weighting large-displacement layers in the loss (ablation C). It does not help: $\rho = 0.25$ leaves direction unchanged (0.7392 vs. 0.7397) and pushes the magnitude ratio further above target (1.150 vs. 1.130), and $\rho = 0.5$ degrades direction to 0.7301 while overshooting badly (1.247). The reason is visible in the baseline number itself: aggregate amplitude is already slightly *above* 1.0, so a global up-weighting overshoots. The deficit is specific to large turns, and an aggregate objective gives no credit for fixing only those. A correct fix has to be conditional on motion scale rather than a global reweighting; we leave it open.

Two things in Fig. 48 are worth looking for. In the crowded cases the coordination holds: under tilt Z the hood, hair, face and held plush of `test5` rotate as one object rather than shearing against each other, and under turn X the fringe stays attached to the forehead, which is precisely the failure mode per-layer prediction produces (§W.4). The remaining visible weakness is amplitude: large turns come out smaller than an artist would draw them (§X).

W.5 Scaling the parameter vocabulary: $8 \to 24$

Everything above models the 8 Cubism parameters that almost every rig in the corpus declares. That subset covers head turn, nod and tilt, body sway and eye blink, but it is not a usable idle animation: a character that never moves its gaze, brows, hair or chest reads as frozen. We therefore ask whether the same architecture absorbs a three-times larger parameter vocabulary, and what breaks when it does.

Data. We re-extracted the corpus keeping every parameter with at least 200 layer records, which yields 24 parameters over $1{,}306$ characters and $449{,}157$ layer records: the original 8, plus gaze (`EyeBallX/Y`), eye smile, mouth form, four brow controls per side, three hair-sway groups and breathing. `ParamCheek` had only 28 records corpus-wide and was dropped rather than shipped broken. The validation character list is copied verbatim from the 8-parameter split, and we verified that the re-extraction is faithful: over the $1{,}211$ characters present in both packs the base vertex positions agree to a median $\max|\Delta|$ of 1.0×10^{-6}.

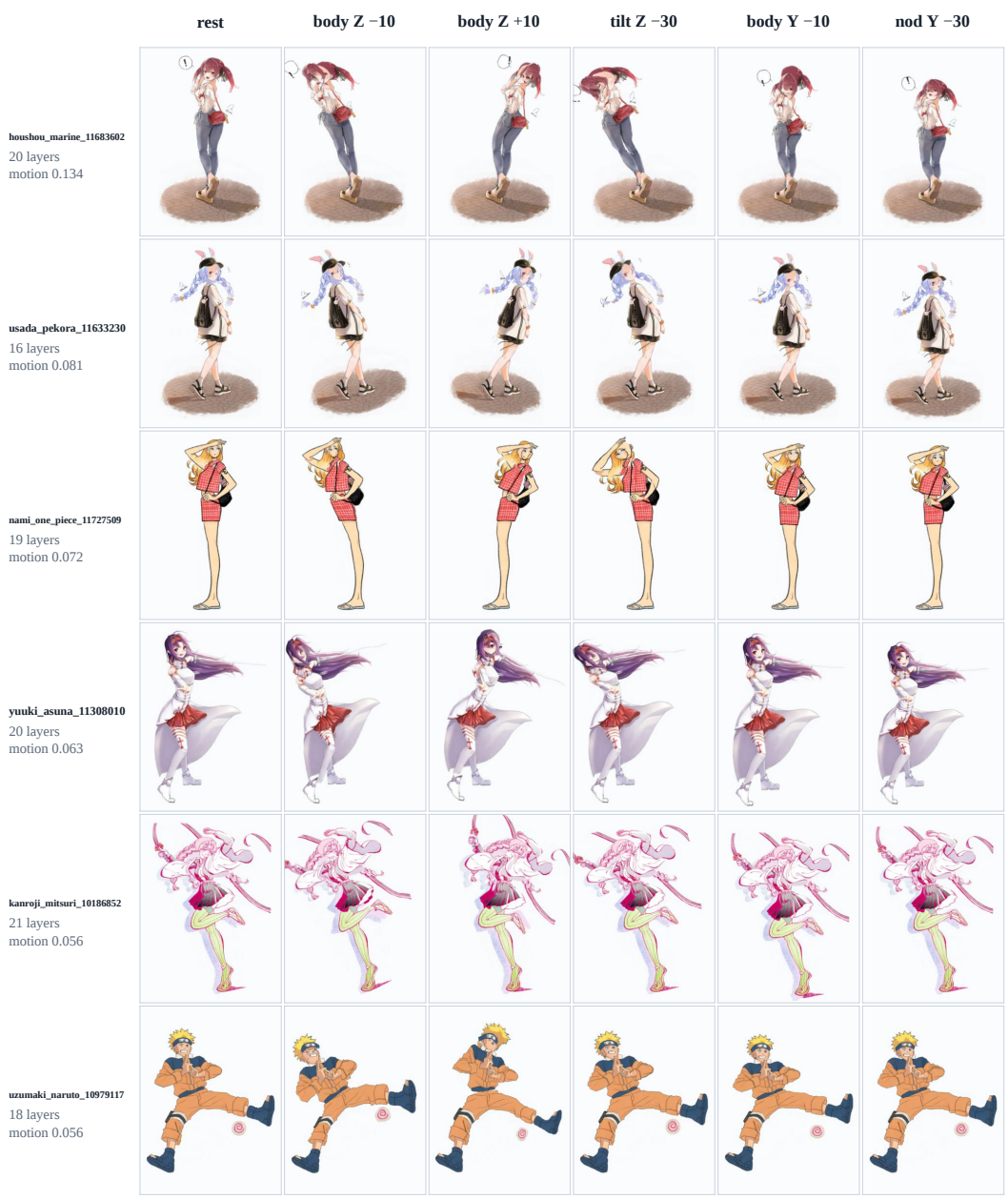

Figure 48: **In-the-wild rigs, raw output.** Web illustrations that were never Live2D models, each decomposed by Stage 1 and animated by Stage 2 at six *absolute* parameter values, so every cell is reproducible from the released rig. Rows are ordered by measured motion magnitude (printed beside each row). One frozen checkpoint, one triangulation procedure and one parameter sweep produced every cell. The remaining characters are in Appendix M.

The vocabulary triples but the capability does not. Table 27 groups the 24 parameters by the quality they reach, and the grouping is stark: 13 parameters land above a direction cosine of 0.70, 5 sit between 0.40 and 0.70, and 6 fall below 0.40. The winners include the two we most wanted, gaze at 0.999 and 0.997, together with breathing at 0.749 and brow height at 0.84, so the extension does deliver a complete idle loop. The weakest controls are brow *form* and brow *angle* and mouth form; side-hair sway remains marginal. Table 18 in Appendix R lists all 24 with their training support, and the ordering is explained almost entirely by that one column. Brow angle has 17 to 18 moving-layer samples against head turn's 2,841; the two parameters with negative cosine are the two rarest in the

Table 27: **Parameter-vocabulary extension, capability tiers.** The 24-parameter model on 46 validation characters, true generation. Tiers are by per-parameter direction cosine; $*$ marks a parameter the 8-parameter model does not have. The full per-parameter table with training support is Tab. 18.

Tier	Parameters	count
good, ≥ 0.70	`EyeBallY*` `EyeBallX*` `BodyAngleZ` `AngleZ` `BrowRY*` `BrowLY*` `EyeROpen` `EyeLOpen` `Breath*` `AngleY` `AngleX` `EyeRSmile*` `EyeLSmile*`	13
marginal, 0.40–0.70	`HairBack*` `BodyAngleY*` `HairSide*` `BodyAngleX` `HairFront*`	5
unusable, < 0.40	`BrowLForm*` `BrowRForm*` `MouthOpenY` `MouthForm*` `BrowRAngle*` `BrowLAngle*`	6

Table 28: **Warm-start ablation on the 24-parameter vocabulary.** Per-parameter direction cosine averaged within groups, 46 validation characters, true generation. "Cold" trains all 24 from scratch; "warm" copies the 8 shared parameter embeddings from the 8-parameter checkpoint by name and trains identically thereafter. The split is exactly the two released vocabularies: the 8 rows that exist in the 8-parameter checkpoint against the 16 that do not and are therefore randomly initialised in both runs.

Parameter group	# params	cold	warm
warm-started (8 shared)	8	**0.7009**	0.6983
newly added	16	0.4792	**0.5588**
all	24	0.5531	**0.6053**

corpus. This is a long-tail data problem, not a capacity problem, which is consistent with ablation B: the same architecture scaled $112\times$ does not improve the parameters it already has, so it will not rescue the ones it barely sees. We report the failures rather than pruning the vocabulary to the 18 that pass 0.40, because a reader deciding whether to model 24 parameters needs to know which ones the data can currently support.

One weakness is *not* caused by the extension. `MouthOpenY` scores 0.239 in the 8-parameter model and 0.235 in the 24-parameter one: mouth opening was always our worst common parameter. Its motion is dominated by a few strongly deforming layers whose displacement is close to a pure anisotropic scale, which the peak-normalised direction target of Eq. 30 represents poorly, and no amount of extra parameter coverage changes that.

Warm-starting from the small vocabulary recovers most of the loss. Trained from scratch on 24 parameters the model spends capacity on the tail and gives ground on the original 8. Copying the 8 shared parameter-embedding rows from the 8-parameter checkpoint *by parameter name*, leaving the 16 new rows at random initialisation, and then training on all 24 is strictly better on the axis that matters (Tab. 28): the eight warm-started parameters hold level (0.6983 against 0.7009 cold, inside seed noise) while the 16 new ones rise from 0.4792 to 0.5588, a 17% relative gain. The gain concentrates exactly where support is thinnest, `EyeRSmile` $0.372 \rightarrow 0.724$ on 17 samples and `EyeLSmile` $0.561 \rightarrow 0.710$ on 16, which is what a transfer explanation predicts: a rare parameter cannot learn a general displacement field from 17 examples, but it can learn to *re-index* a field the shared trunk already represents. Warm-starting is what we release.

A metric caveat we had to resolve: per-vertex cosine is not invariant to mesh density. The 8-parameter model reports 0.7676 and the 24-parameter model 0.689, and the tempting reading, that tripling the vocabulary costs 0.079, is wrong. Three things changed between those two evaluations, and only one of them is the model. Table 29 locks them one at a time. Four validation characters are absent from the re-extracted pack, worth 0.013. The re-extraction also re-meshed 24% of layers more densely, mean vertex count per layer rising from 47.5 to 71.9 and never falling, worth a further 0.063, which is twice the size of the effect being studied. Only the last row isolates the model under

Table 29: **Why** 0.7676 **and** 0.689 **are not comparable.** Each row changes exactly one thing from the row above, on shared validation characters and the shared 8 parameters, true generation throughout. Mesh density accounts for twice as much as the model change under study.

Setting changed	model	chars	dir-cos	Δ
as reported in Tab. 24	8p	50	0.7676	–
− 4 chars absent from the 24p pack	8p	46	0.7543	−0.013
+ denser re-meshing (24% of layers)	8p	46	0.6913	−0.063
only the model differs	24p cold	46	**0.6608**	**−0.031**
same, warm-started (released)	24p warm	46	0.6866	+0.026

this matched validation pack and training schedule: the measured cost of the extra 16 parameters is 0.031; magnitude-ratio deviation from one falls from 0.065 to 0.047.

We verified the density term rather than inferring it. Scoring the same model on only the $1{,}273$ layers whose vertex count is *identical* in both packs gives 0.7161 on one pack and 0.7150 on the other, equal to within 0.001, while the 409 densified layers score 0.6822. Everything else was excluded by direct comparison: the two packs share one physical DINOv2 token file with $57{,}972/57{,}972$ identical row indices (sampled per-layer feature agreement $1.2{\times}10^{-7}$), parameter ranges are identical, layers per pose differ by a median of 0, base vertices by 3.1×10^{-6} and ground-truth offsets by 4.8×10^{-8}. The mechanism is simple in hindsight: a denser mesh adds interior vertices whose true displacement is small and whose *direction* is therefore near-degenerate, yet they still clear the $\|\Delta v\| > 0.005$ gate and enter the average. **Any per-vertex cosine comparison across meshes of different density is confounded**, which affects how our own mesh ablation must be read (§W.3) and, we expect, any future work that adopts this metric. Checkpoints should be compared only on one fixed pack, character set and parameter set; we added the three switches that make this enforceable to our evaluation script and report them in Appendix R.

Released models. We release both versions, because they are not a replacement pair. The 8-parameter model is the more faithful one on the motions it covers (0.6913 against 0.6866 on identical data and parameters) and is what the headline numbers of this paper refer to; the 24-parameter warm-started model is the one that produces a complete idle animation. Each release carries its weights, its parameter vocabulary (whose list order defines the embedding index), an architecture record, and a model card stating per-parameter quality including the 8 parameters that do not work. The reference inference script is verified numerically identical to the pipeline these metrics were computed with: on a 17-layer in-the-wild character all $1{,}634$ base vertices and all 66 (parameter, keypose) frames match to 0.0000 px.

W.6 Editability and outfit swap

Our output remains editable per layer, mesh edge, and keyframe; replacing a layer, changing vertex offsets, or swapping a motion template does not require re-running either stage.

Because a layer's mesh UVs derive from its alpha bounding box, an edit that restores the original alpha and pixel size cannot invalidate the mesh and therefore cannot invalidate the animation. We check that mechanically rather than by eye: per row, the vertex positions, the triangle index lists and *every* predicted displacement frame are byte-identical between the source and the edited rig, which over the 16 edits of Figs. 49 and 50 amounts to 323 layers, $31{,}508$ vertices, $50{,}762$ triangles and $21{,}318$ displacement frames compared, with zero differences. The categories deliberately go beyond clothing: hair, hair with animal ears, a hair ornament, garment, footwear with legwear, headwear, armour, an accessory, a held prop, and the eye irises, the last being the hardest case because those layers are only 30×20 and 44×15 pixels.

Prompt-driven outfit swap (optional Stage 3). Editability is easy to claim and hard to demonstrate, so we instantiate it as a concrete capability. Because a layer's mesh UVs are derived from that layer's alpha bounding box, an edit that leaves the alpha channel and the pixel dimensions untouched cannot invalidate the mesh, and therefore cannot invalidate the animation either. We use this to change a character's outfit: one clothing layer's RGB is repainted by an instruction-guided image editor Qwen

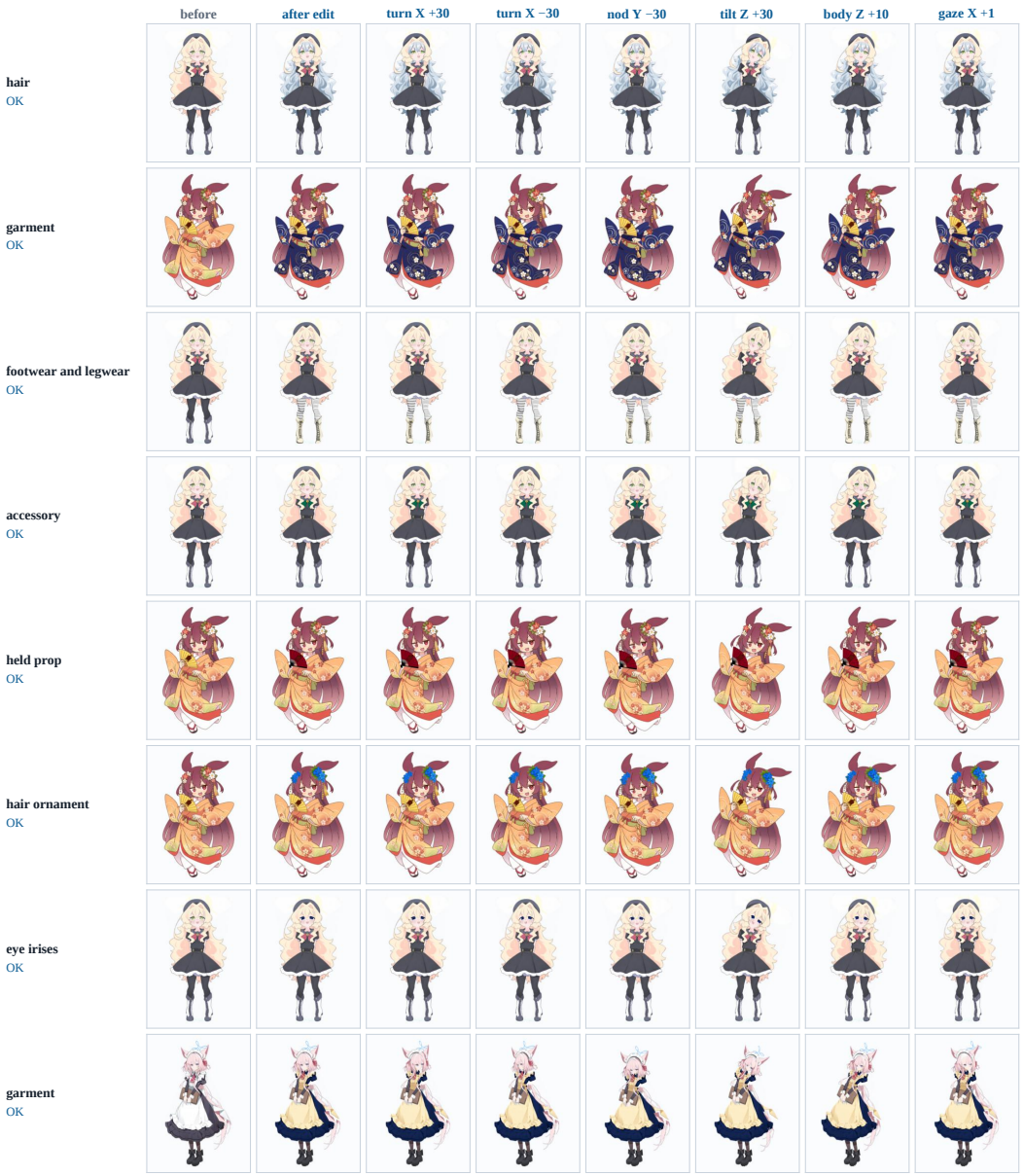

Figure 49: **Prompt-driven texture editing with zero re-animation.** Each row is one edit: the rig before, after repainting one to five layers' RGB from a natural-language instruction, and the *same* rig driven to six absolute parameter values. Every pose column replays the stored displacement frames on the new texture rather than re-generating them. Sheet 2 is Fig. 50; the verification and the category list are in the text.

Team (2025) from a natural-language instruction, the original alpha and size are restored, and the rig is re-exported with the new texture and *no* re-animation and *no* re-decomposition. Figs. 49 and 50 show all 16 edits we ran, on 5 characters and spanning 10 layer categories, repainting 31 layers in total. We checked the invariance mechanically rather than by eye: comparing each edited rig against its source, the vertex positions, the triangle index lists and every predicted displacement frame are byte-identical, which across the 16 edits is 323 layers, 31,508 vertices, 50,762 triangles and 21,318 frames compared with zero differences, and every edited texture retains its exact pixel dimensions and an identical alpha channel; the only field that changes anywhere in the file is the texture path.

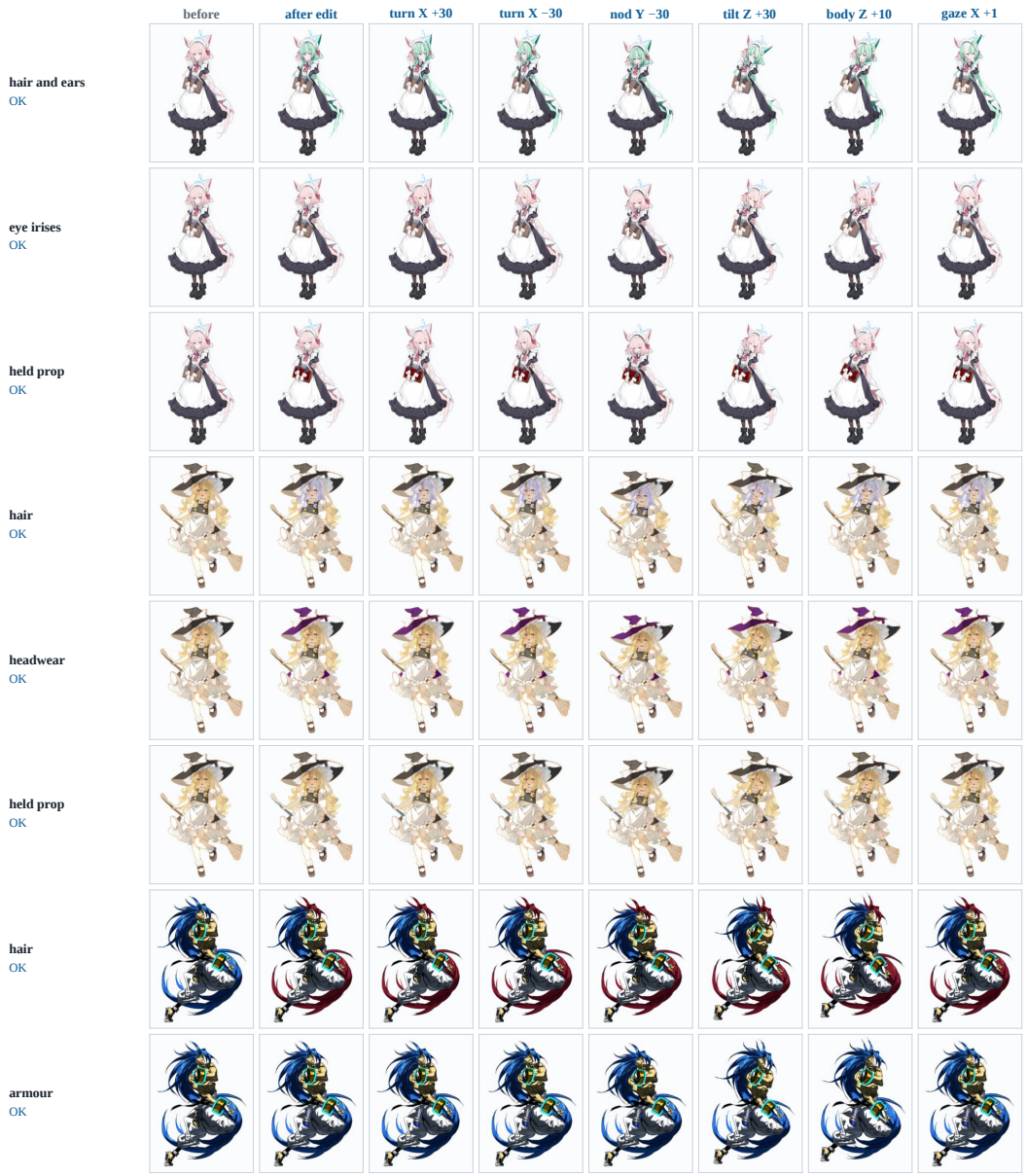

Figure 50: **Prompt-driven texture editing, sheet 2.** Format and verification as Fig. 49. These eight edits are on the three characters added to test whether the invariance is a property of the representation or of two lucky rigs: a maid outfit (garment, hair with animal ears over five layers, irises, held prop), a witch (hair, headwear, broom), and a heavily armoured fighter (hair over four layers, gauntlet).

This is what an editable structured asset buys that a video cannot: a downstream appearance change costs one image edit rather than a re-generation.

Two honest limitations. When a garment layer shares pixels with skin (a short sleeve whose layer also contains the bare arm) the editor can desaturate the skin along with the cloth unless the instruction explicitly names and protects the skin colour; our first attempt on one character did exactly that (Fig. 51), and a skin-protecting instruction fixed it, which means the failure is a prompting failure rather than an architectural one but still a failure a user can hit. And because alpha is deliberately held fixed, only edits that preserve the garment's silhouette are in scope: turning a skirt into trousers changes the alpha support, invalidates the mesh, and requires re-running the pipeline.

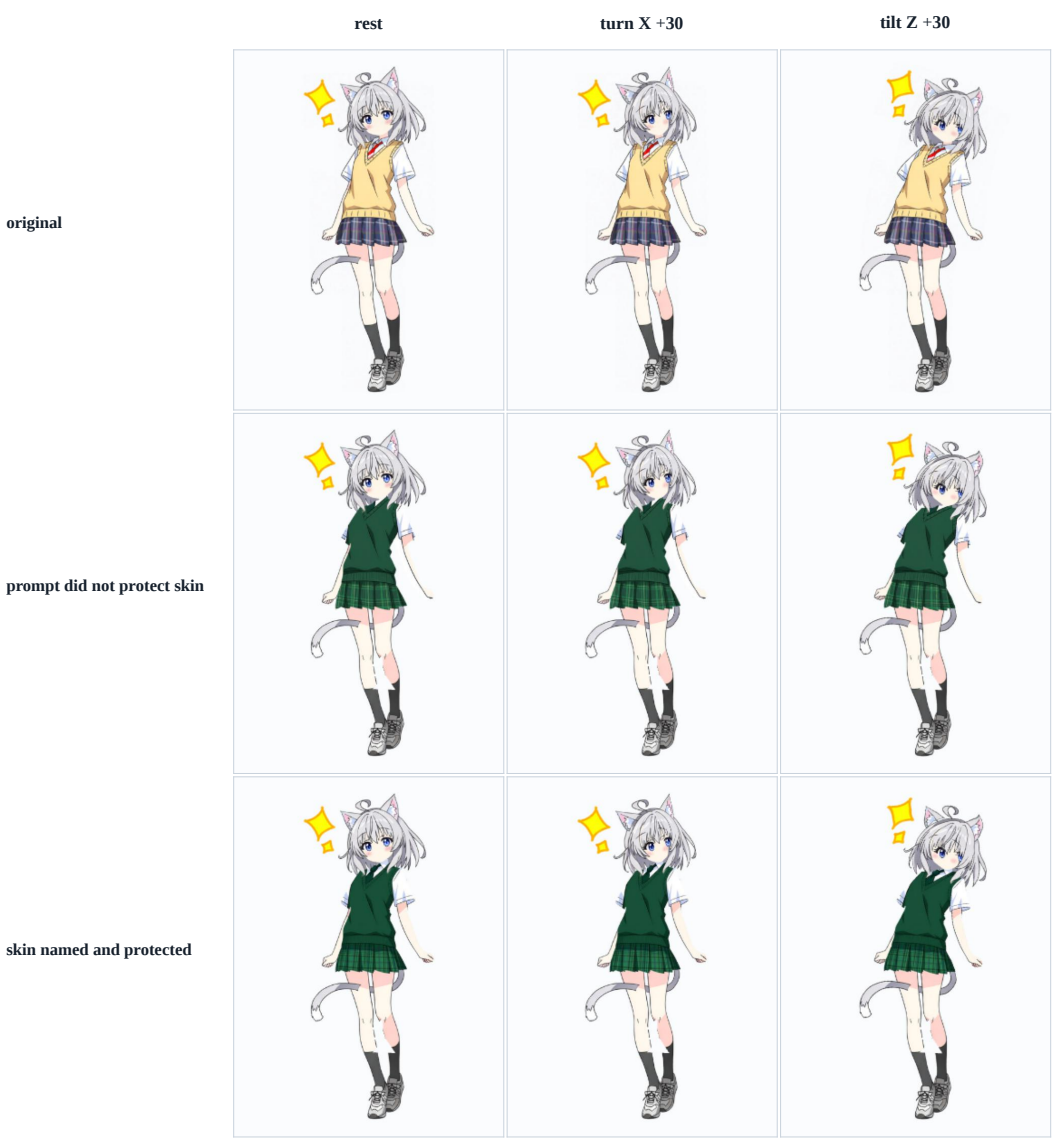

Figure 51: **Honest re-skin failure and its fix.** On `q_webA_catgirl` a single Stage-1 layer holds both the school uniform and the bare arm. Asking only for a green uniform (centre) desaturated the arm and hand to near-white; naming and protecting the skin colour in the instruction (right) preserved them. An edit can only be as local as the layer it lives in, so this failure is inherited from decomposition granularity rather than from the editing step. The right-hand version is the one used in Fig. 49.

What Live2D-Bench measures. The benchmark is deliberately stricter than evaluating a rendered frame. A plausible composite can still be unusable if it merges front hair with the face, omits occluded sleeves, emits no mesh, or bakes motion into pixels rather than editable keyposes; conversely, a mesh predictor can look accurate under oracle layers while failing once the layer alpha support changes. Live2D-Bench therefore exposes the full asset interface: ordered RGBA layers, per-layer artist meshes, parameter ids, keypose values, vertex-offset fields, and rendered Body / Face Angle loops. Its 10 to 20, 20 to 35, and 35+ layer bins test semantic separation, the common production regime, and ordering / occlusion pressure, while the non-human split checks tails, ears, wings, plush bodies, and accessories. This protocol distinguishes a Live2D asset from a movie of a Live2D-like character; a redistributable, source-disjoint benchmark remains to be assembled.

X ADDITIONAL LIMITATIONS

We present a prototype that turns one illustration into a structured, editable, animatable open Live2D-style asset. Stage 1 predicts ordered RGBA layers with hidden-region completion; Stage 2 meshes them and predicts vertex keyposes for all layers jointly. On 50 artist-layer validation characters excluded from gradient training but used for checkpoint selection, direction cosine is 0.7676 (median 0.8278), with 31 of 50 above 0.80. This does not establish performance on an untouched test or on automatically generated meshes. A texture-editing demonstration reuses stored geometry and motion. The full end-to-end claim is supported only by a small three-character rigged control and qualitative examples.

Limitations. Several limitations are worth stating precisely, because each points at a different missing piece rather than at a tuning deficiency.

Amplitude compression on large motion. Aggregate amplitude is well calibrated (the median magnitude ratio is close to 1) but the error is not uniform across motion scale: small displacements are slightly over-shot and large head or body turns are under-shot. One possible explanation is pointwise regression on an ambiguous task, since one still image and one parameter value can admit several plausible artist motions. The present experiments do not isolate that cause. Scaling the model $40\times$ does not fix it (Tab. 26) and neither does globally up-weighting amplitude in the loss; a distributional output (sampling, or a flow-matching / diffusion head over displacement fields) is the principled remedy and is our main planned direction.

Animated draw order is not modelled, and conditioning on static order does not help. We tested the obvious remedy of feeding each vertex its layer's normalised draw-order position explicitly; it scored below both baseline seeds (Appendix P), which suggests the ordering is already recoverable from layer identity plus the shared canvas. We predict a static layer ordering and deform within it. Real rigs sometimes keyframe the draw order itself, so that a hair strand passes in front of a face partway through a turn, or an arm crosses the torso. Nothing in our formulation expresses an order that changes with a parameter value, so those transitions are simply unavailable; adding a per-layer, per-keypose depth channel is a natural extension.

Texture edits cannot change silhouette. The outfit-swap step preserves the animation precisely because it holds a layer's alpha channel and pixel dimensions fixed. That is also its boundary: edits that change the garment's outline invalidate the mesh support and require re-running decomposition and animation. A related failure is that when a garment layer shares pixels with skin, the editor may alter the skin unless the instruction protects it explicitly.

The eye-open channel is unreliable on auto-decomposed layers. Driving `ParamEyeLOpen` $=$ `ParamEyeROpen` $= 0$ on an in-the-wild rig displaces the whole head group instead of only the eyelid layers (Fig. 52); the same rigs are correct under the head-angle, body-sway and mouth parameters, so this is specific to that channel. Two causes compound: our Stage 1 does not reliably emit the eyelid as a separate layer, so there is no layer for the motion to attach to, and in training the two eye parameters contribute roughly 6k deformation records each against roughly 91k for each head angle, with the smallest displacements of any parameter, so the channel is also the least supervised. On artist-authored stacks, where the eyelid *is* its own layer, the parameter behaves. Isolating eyelid layers in Stage 1 and up-sampling eye records in training are both straightforward next steps, and we exclude blink from the in-the-wild pose sweeps in the supplementary material rather than showing it silently.

Background props are decomposed and animated like body parts. Stage 1 has no notion of what belongs to the character, so a sheet of paper behind a chibi, a held plush toy, or a floating motif becomes a layer, receives a mesh, and is deformed by head-turn parameters that should not touch it. A character/background gate before Stage 1 would address this and is not part of the pipeline evaluated here.

Dependence on Stage 1. Stage 2 can only animate the layers it is given. A layer cut through a sleeve, or one whose occluded region was completed poorly, produces artefacts no matter how accurate the displacement field is, and our cross-decomposer experiment shows the end-to-end gap on in-the-wild art is dominated by decomposition rather than by motion error (Appendix S).

The long tail of the parameter vocabulary is not learnable from present data. Extending the model from 8 to 24 Cubism parameters (§W.5) produces 13 parameters above 0.70 and 6 below 0.40, and the split is explained by training support rather than by anything about the motion: brow angle has 17 to 18 moving-layer samples in the whole corpus against head turn's $2{,}841$, and the two parameters with negative direction cosine are the two rarest. Warm-starting from the small-vocabulary model recovers a substantial part of the gap ($0.4792 \rightarrow 0.5588$ on the 16 new parameters) precisely because a rare parameter can re-index a displacement field the shared trunk already represents, but it cannot manufacture the missing supervision. Brow expression and mouth form therefore remain out of reach until the corpus grows in those channels, and we release the failures visibly rather than pruning the vocabulary to what happens to work.

Our own primary metric has a confound we had to discover. Per-vertex direction cosine is not invariant to mesh density: a denser mesh contributes interior vertices whose true displacement is small and whose direction is consequently near-degenerate, yet which still clear the movement threshold and enter the average. We measured this at 0.063 cosine for a 24% re-meshing that raised mean vertex count from 47.5 to 71.9, which is twice the size of the model effect we were studying at the time (§W.5). Any per-vertex comparison across different meshes is therefore confounded, including our own mesh ablation, and Live2D-Bench numbers should be quoted with their pack, character list and parameter list attached. A density-normalised or area-weighted variant of the metric would be a genuine contribution and we do not have one.

Scope. Physics dampers, expression blending, and viseme parameters are runtime extensions we do not learn; we ship an open asset bundle plus our own renderer rather than a native `.moc3`; and our training distribution skews toward Japanese anime aesthetics, leaving photorealistic, 3D-shaded, and multi-character inputs out of scope. The released motion code and renderer provide a basis for addressing these; a rights-cleared benchmark and Stage 1 release are still needed.

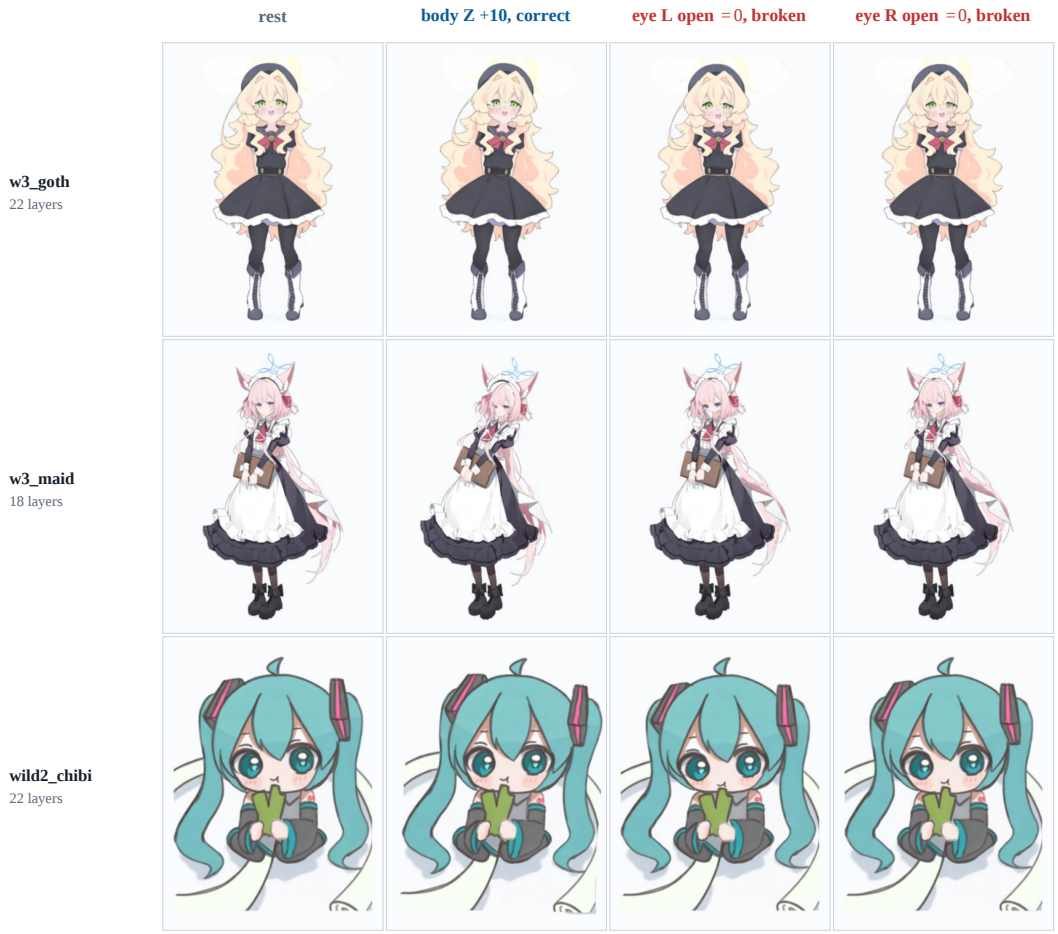

Figure 52: **Failure mode: the eye-open parameter on auto-decomposed layers.** Three in-the-wild rigs at rest (left), under `ParamBodyAngleZ` at its maximum (centre, correct), and with `ParamEyeLOpen` = `ParamEyeROpen` = 0 (right, broken). The blink displaces the whole head group rather than the eyelids: the face slides out from under the hair, the skin layer translates off the head, or the head shrinks and rotates. Because the centre column of each row is the same rig under a different parameter and is correct, the mesh and the rig are sound and the fault is specific to the eye channel. Raw renders, no per-example fix.

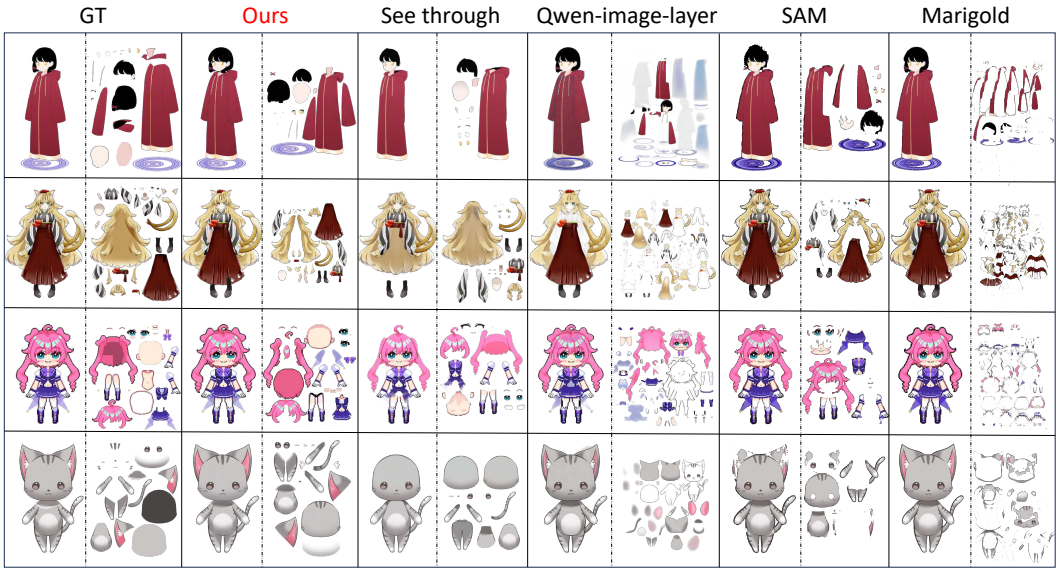

Figure 53: Qualitative comparison against segmentation baselines on the same input illustration.All methods take the same GT illustration as input. From left to right (GT, ours, See through, Qwen-image-layer, SAM, and Marigold), each column pairs the reconstructed character with its decomposed layer textures.

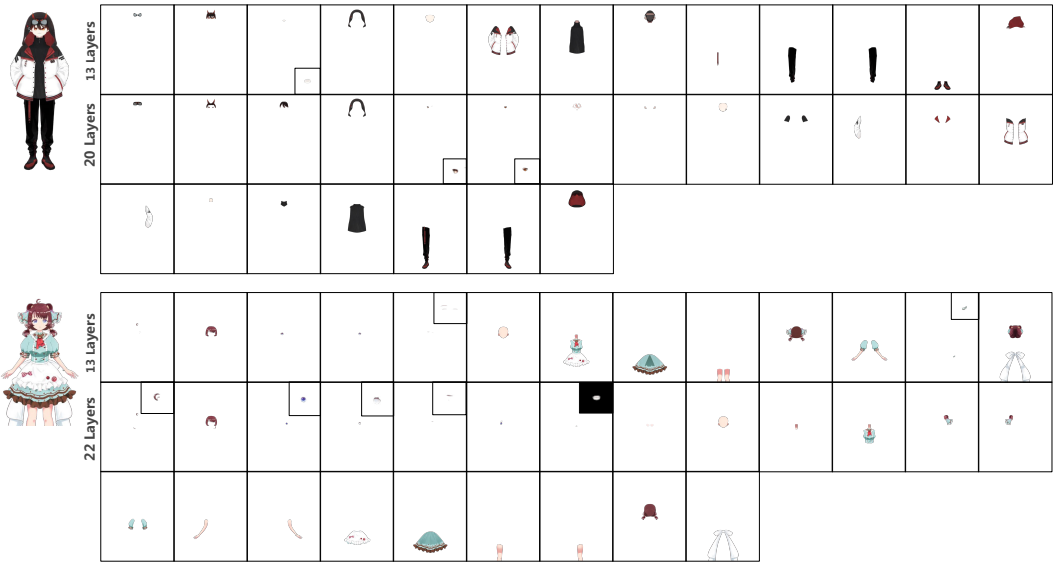

Figure 54: The reconstructed characters (left) are shown alongside their decomposed layer textures (right) under different specified layer counts. Our approach naturally adapts to free layer numbers, supporting both coarse and fine-grained separation.

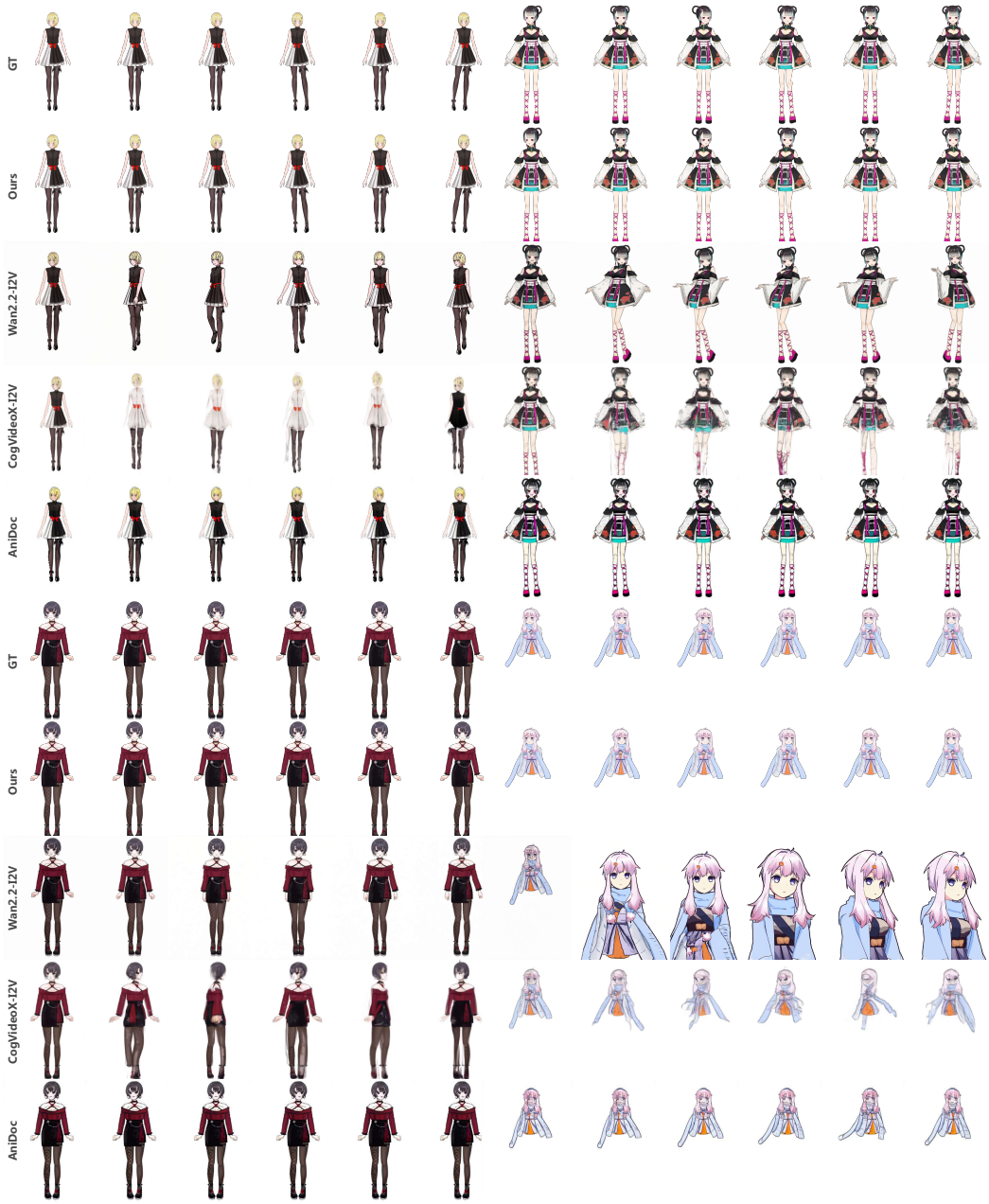

Figure 55: Qualitative comparison against image-to-video baselines on the same input illustration and animation prompt. Each row shows six frames evenly sampled from the generated clip; from top to bottom: GT, Wan2.2-I2V, CogVideoX-I2V, AniDoc, and ours.