DENSE-RAM: EDGE-LEVEL CREDIT ASSIGNMENT FOR REINFORCEMENT LEARNING IN FLOW-BASED UV UNWRAPPING

Anonymous authors
Paper under double-blind review

ABSTRACT

UV unwrapping is a fundamental step in creating textured 3D assets. Recent generative methods learn artist-like seam layouts from data, and reinforcement learning can further refine these layouts using UV-quality feedback. However, scalar feedback assigns a shared reward coefficient to all edges in a sampled layout, overlooking differences in local UV quality. To address this limitation, we introduce Dense-RAM, a reinforcement learning framework for post-training flow-matching seam generators. Built on Reinforce Adjoint Matching (RAM), Dense-RAM computes a candidate-level advantage from chart- and layout-level UV feedback, then distributes it across edges according to local quality. The resulting coefficients adapt the reward correction to each edge while preserving the candidate's mean advantage. Experiments show that Dense-RAM improves chart topology and reduces UV fragmentation. Across RAM, DiffusionNFT, and AWM, edge-level allocation consistently improves final UV quality and reaches comparable quality in fewer optimization steps than scalar feedback.

1 INTRODUCTION

UV unwrapping connects 3D geometry with 2D image representations by mapping a mesh surface into a planar texture domain. This correspondence allows surface appearance to be generated and edited in image space and transferred to the surface for rendering. UV layout quality affects the fidelity of mapped textures and the ease of texture editing, making it an important component of textured 3D asset creation.

Seam placement is central to this process: cutting a mesh along selected edges partitions its surface into charts for planar parameterization. Poorly placed seams can lead to distortion, excessive fragmentation, or visible discontinuities, complicating texture synthesis and editing(Xu et al., 2025). Seam placement therefore requires balancing distortion reduction against fragmentation and visible texture boundaries.

Recent generative approaches learn distributions of seam layouts conditioned on input geometry, using artist-authored UV partitions as ground-truth supervision (Li et al., 2025; Zhao et al., 2026b). Through this data-driven modeling, these methods learn patterns of seam placement and surface segmentation directly from artist examples.

Building on these learned seam distributions, reinforcement learning can further improve generation using UV quality feedback. SeamCrafter uses metric-derived preferences and DPO to improve complete seam sequences, while diffusion and flow RL methods use scalar endpoint rewards, such as DiffusionNFT (Zheng et al., 2026) and AWM (Xue et al., 2026). A scalar multiplier is shared by edge decisions of heterogeneous quality, even when the UV evaluator also returns regional metrics. We refer to the potential interference from this shared feedback as **structural credit interference**. This motivates using local information to shape the update, rather than asserting that scalar methods cannot learn different edge decisions.

We propose Dense-RAM, a structure-aware post-training framework for flow-matching seam generators. Dense-RAM combines chart- and layout-level UV feedback into edge-aligned scores and computes a group-centered, scaled advantage for each candidate. Local score deviations then deter-

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.

mine nonnegative, mean-one edge weights that redistribute this advantage within the RAM regression target. This construction adapts the reward correction to local UV quality while preserving the candidate's mean advantage.

Our evaluation examines UV generation quality, reward utilization, and transfer across RL objectives. Paired scalar/edge comparisons isolate the allocation mechanism, and experiments with DiffusionNFT and AWM show that its benefits extend beyond RAM on the UV task. Our contributions are:

- We propose Dense-RAM, a reinforcement learning method for flow-based UV unwrapping. It uses UV quality feedback to post-train seam generators, improving chart topology and reducing layout fragmentation.
- We introduce an edge-level credit assignment mechanism that can be incorporated into different flow-matching RL methods. It improves RL exploration efficiency by using local UV quality to provide more detailed feedback from each evaluated layout.

2 Related Work

Geometric parameterization. Surface parameterization maps a 3D surface to a planar domain while controlling flattening distortion (Floater & Hormann, 2005; Sheffer et al., 2007). Early approaches use convex-combination constructions (Floater, 1997), intrinsic angle- and area-based energies (Desbrun et al., 2002), and texture-stretch objectives (Sander et al., 2001). LSCM and ABF++ seek angle-preserving maps (Lévy et al., 2002; Sheffer et al., 2005), MIPS penalizes Jacobian-based distortion (Hormann & Greiner, 2000), and ARAP favors locally rigid transformations (Liu et al., 2008). SLIM preserves local injectivity from a locally injective initialization (Rabinovich et al., 2017), while free-boundary bijective parameterization incorporates a boundary-intersection barrier to prevent boundary overlaps (Smith & Schaefer, 2015). OptCuts jointly optimizes cut placement and parameterization to reduce seam length subject to distortion bounds (Li et al., 2018).

Neural parameterization and semantic priors. Neural approaches learn surface representations and mappings. AtlasNet represents 3D shapes as collections of learned parametric patches (Groueix et al., 2018), while Neural Surface Maps optimizes surface-to-surface mappings through compositions of neural maps (Morreale et al., 2021). Related work learns surface-to-texture mappings from multi-view images (Das et al., 2022) or couples local patch selection with differentiable parameterization to identify low-distortion regions (Liu et al., 2023a). FAM learns free-boundary parameterizations through bidirectional mappings and cycle consistency (Zhang et al., 2024), while Nuvo learns neural atlases and FlexPara jointly optimizes chart assignment and surface parameterization (Srinivasan et al., 2025; Zhao et al., 2025). PRISM transfers geodesic-pretrained representations to feedforward surface parameterization (Zhao et al., 2026a). Part-level priors can be obtained from learned feature fields (Liu et al., 2025b) or by lifting image segmentation masks to meshes (Tang et al., 2024). PartUV combines PartField features with geometric decomposition (Wang et al., 2025), while Zamani et al. (2026) introduce semantic partitioning and an ambient-occlusion-based seam visibility objective. Dense-RAM uses UV-quality feedback to refine a pretrained seam generator.

Generative seam modeling. Generative methods learn geometry-conditioned distributions of seam layouts from artist-authored examples. SeamGPT autoregressively predicts quantized seam segments (Li et al., 2025), whereas MeshTailor traces seam paths directly on the mesh graph (Ma et al., 2026). ArtUV combines SeamGPT with learned refinement of an initial UV parameterization (Chen et al., 2026). Flow matching and rectified flow learn continuous transport through velocity-field regression (Lipman et al., 2023; Liu et al., 2023b). SeamFlow applies flow matching to continuous seam scores on mesh edges, providing an edge-aligned representation for subsequent UV evaluation (Zhao et al., 2026b). SeamCrafter refines seam sequences through direct preference optimization (Rafailov et al., 2023), using preferences derived from distortion and island count (Xu et al., 2025). Dense-RAM post-trains SeamFlow with chart- and layout-level feedback, allocating candidate-level advantages across individual mesh edges.

RL post-training for diffusion and flow models. DDPO and Flow-GRPO optimize likelihood-ratio objectives over stochastic sampling trajectories (Black et al., 2024; Liu et al., 2025a). DiffusionNFT, AWM, and Direct Group Preference Optimization (DGPO) incorporate endpoint feedback through regression, matching, or group-preference objectives (Zheng et al., 2026; Xue et al., 2026; Luo et al.,

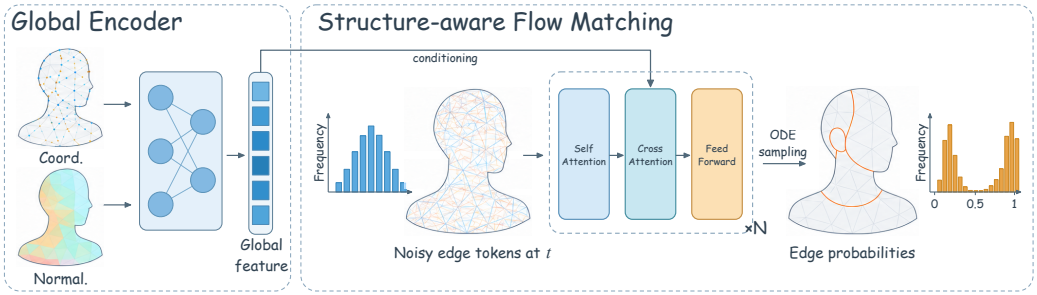


Figure 1: **SeamFlow architecture.** A global encoder extracts shape features from surface coordinates and normals. A flow-matching transformer combines self-attention over noisy edge tokens with cross-attention to the global features, and ODE integration produces continuous seam scores.

2026). Adjoint Matching formulates reward fine-tuning through stochastic optimal control (Domingo i Enrich et al., 2025), and RAM obtains a reward-corrected regression target without differentiating the reward (Bergmeister et al., 2026). Alongside these objective formulations, credit-assignment methods refine how endpoint feedback is distributed across denoising steps. The dense-reward view of preference alignment discounts or reweights reverse steps (Yang et al., 2024), while DenseGRPO estimates reward gains from intermediate clean-image predictions and adapts timestep-dependent exploration (Deng et al., 2026). Dense-RAM distributes each candidate's scalar advantage across mesh edges within RAM's vector-valued regression target. This spatial allocation complements temporal credit assignment and requires no auxiliary allocation network.

3 Preliminaries

3.1 Flow-Matching Seam Generation

Let $M = (V, E, F)$ be a welded triangular mesh with $m = |E|$ valid model edges. SeamFlow (Zhao et al., 2026b), as illustrated in Figure 1, represents a relaxed seam layout by continuous edge scores $S \in \mathbb{R}^m$, which are thresholded as $y_e = \mathbf{1}[S_e > 0.5]$. Cutting the mesh along the selected edges and parameterizing the resulting charts produces a UV layout. Our post-training objective is to improve the quality of this layout through feedback from the UV evaluator.

The generator predicts a velocity field $v_\theta(X_t, t, M)$ in edge-score space, conditioned on mesh geometry. For a clean sample S and independent Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, conditional flow matching uses

$$X_t = (1-t)\epsilon + tS, \qquad Y = S - \epsilon. \tag{1}$$

Supervised training regresses v_θ against Y, and generation integrates $\mathrm{d}x/\mathrm{d}t = v_\theta(x, t, M)$ from noise at $t = 0$ to a clean endpoint at $t = 1$. The continuous scores are not calibrated seam probabilities. Thresholding and UV parameterization form a black-box evaluation pipeline, so we use regression-based RL to incorporate its quality feedback.

3.2 Scalar Reinforce Adjoint Matching

RAM uses on-policy endpoints, black-box rewards, and analytic noising to construct a stop-gradient regression target (Bergmeister et al., 2026). For a mesh, let $S^{(1)}, \ldots, S^{(G)}$ be independently sampled endpoints and A_g the normalized scalar advantage of candidate g, obtained by comparing its reward with those of the other candidates. Analytic noising of an endpoint follows Eq. (1). With all velocity fields evaluated at the same (X_t, t, M), define

$$H = Y - v_\theta, \quad \Delta = v_\theta - v^{\mathrm{ref}}, \quad \widehat{v}^{\mathrm{s}} = \mathrm{sg}\big(v^{\mathrm{ref}} + A_g H\big). \tag{2}$$

Here v^{ref} is a frozen reference model and sg denotes stop-gradient. Using the same valid-edge averaging as our dense variant, the scalar objective is

$$\mathcal{L}_{\mathrm{scalar}} = \mathbb{E}\left[\frac{1}{m}\,\|\Delta - \mathrm{sg}(A_g H)\|_2^2\right]. \tag{3}$$

The same advantage A_g modulates every edge of a candidate. Although edge residuals and parameter Jacobians can differ, this shared coefficient does not explicitly adapt reward modulation to local UV quality. Dense-RAM introduces edge-specific coefficients while retaining the RAM regression framework.

4 Proposed Method

Dense-RAM converts structured UV feedback into edge-specific learning signals. Post-training retains the SeamFlow architecture. For each mesh, the live policy generates a group of candidates, and the UV evaluator provides quality scores aligned with their edges. These scores determine both a scalar candidate advantage A_g and nonnegative allocation weights w_{ge}.

The weights preserve the candidate's mean advantage while adapting the correction strength across edges. Figure 2 summarizes the method. We first construct structured UV feedback, then allocate credit, and finally incorporate the coefficients into the regression objective.

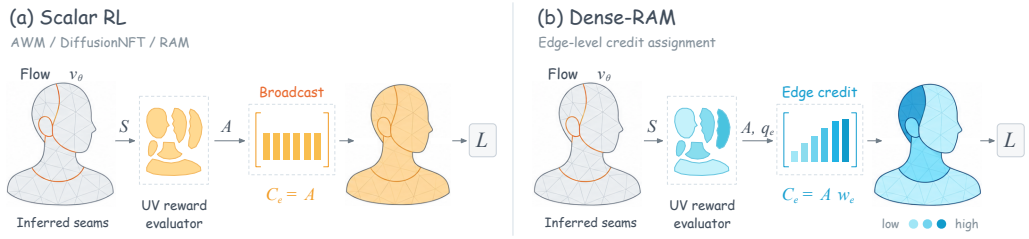


Figure 2: **Scalar versus edge-level credit assignment for flow-based UV unwrapping.** (a) Scalar feedback applies a shared candidate-level coefficient to all edges. (b) Dense-RAM redistributes this coefficient using nonnegative, mean-one weights derived from local UV quality. For RAM, the coefficients are $C_e = A$ and $C_e = Aw_e$, respectively. Colors indicate relative allocation weights.

4.1 Structured UV Quality Rewards

Our reward combines chart-level topology and distortion with layout-level fragmentation and seam visibility. We map these components to the model edges and define the quality score for edge e of candidate g as

$$q_{ge} = \sum_{k \in \mathcal{K}} \lambda_k r^k_{ge}, \qquad \lambda_k \geq 0. \tag{4}$$

The four reward components $\mathcal{K}$ are summarized in Table 1; their weights are fixed during training. All components are oriented so that larger values indicate better quality under the corresponding criterion. AO denotes ambient occlusion.

To align chart-level rewards with model outputs, let $\mathcal{T}^{\text{eval}}_g$ be the evaluated chart set and $\mathcal{N}_g(e)$ the charts incident to edge e. For $k \in \{\text{topo}, \text{dist}\}$,

$$r^k_{ge} = \frac{1}{|\mathcal{N}_g(e)|} \sum_{c \in \mathcal{N}_g(e)} r^k_{gc}. \tag{5}$$

Edge-level reward components are already aligned with the output coordinates. For a layout-level component, we set $r^k_{ge} = r^k_g$ for every valid edge. This construction retains regional quality differences while incorporating whole-layout preferences into each candidate's score.

Table 1: UV reward components.

Level	Component	Criterion
Chart	Topology	$\chi = 1,\ B = 1$
Chart	Distortion	$E_{\text{D}} \leq \tau_{\text{D}}$
Layout	Island count	$n^{\text{reward}}_g \downarrow$
Layout	Seam visibility	$\nu_g \downarrow$

Algorithm 1 Dense-RAM post-training

Require: Mesh dataset $\mathcal{D}$, pretrained parameters θ_0, group size G, endpoint reuse count K
Ensure: Post-trained parameters θ

Initialize $\theta \leftarrow \theta_0$ and freeze $v^{\mathrm{ref}} \leftarrow v_{\theta_0}$
while training budget remains **do**
  Sample a mesh $M \sim \mathcal{D}$
  Generate and detach $\{S^{(g)}\}_{g=1}^{G}$ using the live-policy ODE solver and independent Gaussian noise
  for $g = 1, \ldots, G$ **do**
    Threshold $S^{(g)}$, evaluate the UV layout, and compute q_{ge} using Eq. (4)
  end for
  Compute b_g, A_g using Eq. (6) and set $d_{ge} \leftarrow q_{ge} - b_g$
  Compute w_{ge} using Eq. (7) and set $C_{ge} \leftarrow A_g w_{ge}$
  for all $g \in \{1, \ldots, G\}$ and $k \in \{1, \ldots, K\}$ **do**
    Independently draw (t, ϵ) and construct X_t, Y using Eq. (1)
    Form $\widehat{v}_g^{\mathrm{d}} \leftarrow \mathrm{sg}(v^{\mathrm{ref}} + C_g \odot H)$, with H from Eq. (2)
  end for
  Update θ with one optimizer step using Eq. (9)
end while

Why disk-like topology?

Although the original SeamFlow already achieves a high unwrapping success rate, obtaining UV coordinates does not ensure that the predicted cuts yield disk-like charts. For example, a ring-shaped chart can be parameterized as an annulus, but packing and padding constraints may make its interior hole difficult to fill with other islands, leaving texture space underused. We therefore reward disk topology to favor simply connected charts with a single boundary, aiming to make the resulting UV islands easier to pack and improve texture-space utilization. The distortion and island-count terms balance this preference against deformation and excessive fragmentation.

4.2 Advantage-Preserving Edge Credit Assignment

The allocation separates comparisons between candidates from quality differences within each candidate. First, the valid-edge mean defines a scalar candidate reward b_g; centering over the G candidates of the same mesh gives

$$b_g = \frac{1}{m} \sum_{e} q_{ge}, \qquad \bar{b} = \frac{1}{G} \sum_{h=1}^{G} b_h, \qquad A_g = \beta \frac{b_g - \bar{b}}{\widehat{\sigma}}. \tag{6}$$

The factor $\beta > 0$ controls the magnitude of the reward correction. We estimate the positive scale $\hat{\sigma}$ from pooled reward statistics using an exponential moving average across training steps. Both scalar and dense variants use the same score, centering, and scale estimation.

Next, deviations from the candidate mean determine nonnegative allocation weights:

$$d_{ge} = q_{ge} - b_g, \quad \widetilde{w}_{ge} = \left[1 + \alpha \, \mathrm{sign}(A_g) \frac{d_{ge}}{\sigma_d}\right]_+, \quad w_{ge} = \frac{\widetilde{w}_{ge}}{m^{-1} \sum_j \widetilde{w}_{gj}}. \tag{7}$$

Here $[z]_+ = \max(z, 0)$, $\alpha \geq 0$ controls allocation strength, and $\sigma_d > 0$ is a local-deviation scale shared across the candidate's valid edges. Every mean and normalization excludes padding. The resulting edge-level advantage, used as the RAM correction coefficient, is

$$C_{ge} = A_g w_{ge}. \tag{8}$$

For positive advantages, weights are nondecreasing in local quality; for negative advantages, the ordering is reversed. Clipping or uniform allocation can produce ties. All nonzero coefficients retain the candidate advantage's sign, while clipping may set some coefficients to zero. The rule thus adjusts correction magnitudes according to local quality without a learned allocation model.

4.3 DENSE-RAM OPTIMIZATION

We replace $A_g H$ in the scalar RAM target with $C_g \odot H$, yielding the detached target $\widehat{v}_g^{\mathrm{d}} = \mathrm{sg}(v^{\mathrm{ref}} + C_g \odot H)$. The coefficients modify the reward correction within the regression target, rather than weighting the squared loss. For each evaluated endpoint, we construct K independent analytic noising pairs (t, ϵ) and optimize

$$\mathcal{L}_{\text{Dense-RAM}} = \mathbb{E}\left[\frac{1}{G}\sum_{g=1}^{G}\frac{1}{K}\sum_{k=1}^{K}\frac{1}{m}\left\|\Delta^{(g,k)} - \mathrm{sg}\big(C_g \odot H^{(g,k)}\big)\right\|_2^2\right]. \tag{9}$$

We stop gradients through the sampled endpoints, evaluator outputs, group statistics, and regression target, so only the policy prediction receives gradients. The policy and frozen reference share the same supervised initialization. Each candidate group yields one optimizer step, reusing its UV feedback across K analytic noising draws. The scalar control uses the same rewards, normalization, sampling, and endpoint reuse with $\alpha = 0$, isolating the effect of edge allocation.

Allocation properties. For finite raw scores and positive finite normalization scales, the normalized weights satisfy

$$w_{ge} \geq 0, \qquad \frac{1}{m}\sum_{e} w_{ge} = 1, \qquad \frac{1}{m}\sum_{e} C_{ge} = A_g. \tag{10}$$

Normalization preserves each candidate's scalar advantage as the mean correction coefficient across its valid edges. Setting $\alpha = 0$ recovers scalar RAM under otherwise identical training settings, enabling a controlled comparison of scalar and edge-level allocation. Appendix A establishes these coefficient properties and analyzes the resulting change in the RAM update.

Chart-to-edge aggregation uses sparse incidence reductions, and allocation is linear in the number of valid candidate edges. No trainable allocation parameters are added. The noising distribution, solver settings, and execution configuration are given in Appendix B.

5 EXPERIMENTS

We evaluate Dense-RAM through comparisons with existing UV-generation methods, paired Scalar / Dense experiments under RAM, DiffusionNFT, and AWM, and reward-component ablations. These experiments examine UV quality, learning progress under matched training settings, and the contribution of each reward component.

5.1 EXPERIMENTAL SETUP

Datasets and splits. We initialize Dense-RAM from the SeamFlow checkpoint pretrained on 350,000 curated Objaverse meshes with UV layouts and seam annotations, following the original pretraining protocol(Zhao et al., 2026b). For RL post-training, we select 5,500 meshes from Objaverse(Deitke et al., 2023b;a). All 5,500 meshes are disjoint from the SeamFlow pretraining set. We partition them into 5,000 meshes for RL training and 500 meshes for testing, so the test meshes are held out from both supervised pretraining and RL training. Tables 2,3,4 and Figure 5 report results on the 500 test meshes.

Models and training. Dense-RAM and scalar RAM start from the same SeamFlow checkpoint and share the UV evaluator and reward components. 10 meshes are used in training for every step. Both sample G = 24 candidates per mesh, reuse each endpoint for K = 8 analytic noising draws, and train for 500 optimizer steps; Appendix C describes endpoint sampling and feedback reuse. Comparisons across RAM, DiffusionNFT, and AWM use the same seam-generation backbone and follow the paired evaluation protocol in Appendix C.

Evaluation metrics. We assess topology, fragmentation, seam placement, and texture-space occupancy using four complementary metrics. *All-disk rate* (↑) is the fraction of candidate layouts whose seam-induced charts all pass the disk-topology test before parameterization, measuring topology at the layout level. *UV islands* (↓) counts connected islands in the final UV atlas, including any additional cuts introduced during parameterization, and measures layout fragmentation. *Visibility* (↓) is the seam-length-weighted mean of $1 - \mathrm{ao}_e$ over predicted seam edges, where $\mathrm{ao}_e = 1$ denotes full

Table 2: Quantitative comparison with existing UV-generation methods on the test set. We report angular and area distortion, final UV-island count, seam visibility, and end-to-end inference time.

Method	Venue	Angle ↓	Area ↓	Islands ↓	Visibility ↓	Time (s) ↓
Blender Smart UV	—	1.69	1.50	230.7	0.82	0.4
xatlas	—	0.15	0.08	202.1	0.84	3.8
OptCuts	TOG'18	0.20	0.13	22.1	0.76	5.8
Nuvo	ECCV'24	1.32	1.44	8.0	0.74	537
FlexPara	TPAMI'25	0.42	0.51	6.1	0.68	338
PartUV	SIGGRAPH Asia'25	0.14	0.17	76.4	0.69	11.2
SeamFlow	SIGGRAPH Asia'26	0.13	0.16	66.4	0.67	5.4
Dense-RAM (ours)	—	0.11	0.12	37.2	0.63	5.4

ambient occlusion; lower values indicate less exposed seam locations. *UV utilization* (↑) is the fraction of the texture-atlas area covered by the union of packed UV triangles, measuring texture-space occupancy.

5.2 UV Generation Quality

We compare Dense-RAM with Blender Smart UV Project, xatlas, PartUV, Nuvo, FlexPara, and OptCuts, following the baseline suite of SeamFlow(Zhao et al., 2026b; Wang et al., 2025; Srinivasan et al., 2025; Zhao et al., 2025; Li et al., 2018). These methods span conventional UV tools, geometric optimization, and neural parameterization. We include the original SeamFlow model to measure the gains from RL post-training.

Quantitative quality. Table 2 compares distortion, fragmentation, seam visibility, and end-to-end inference time. Relative to SeamFlow, Dense-RAM reduces the mean UV-island count from 66.4 to 37.2 and seam visibility from 0.67 to 0.63, while lowering angular distortion from 0.13 to 0.11 and area distortion from 0.16 to 0.12. These results show that post-training produces less fragmented layouts with less exposed seams while preserving low parameterization distortion. Although Nuvo and FlexPara produce fewer islands, their angular and area distortion are higher, highlighting the need to assess fragmentation together with deformation. Dense-RAM achieves these gains at the same reported end-to-end inference time as SeamFlow (5.4 s), without an additional inference module.

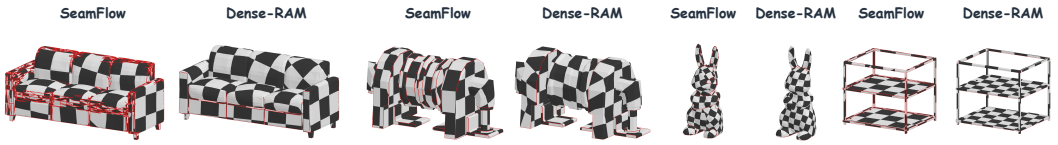


Figure 3: **Checkerboard texture comparisons between SeamFlow and Dense-RAM.** Red lines indicate UV seams. Dense-RAM produces less fragmented seam layouts and more coherent texture patterns across the selected examples. 🔍 *Zoom in for details.*

Qualitative results. As shown in Figure 3, Dense-RAM produces less fragmented seam layouts and more coherent texture patterns than SeamFlow on the selected meshes. Figure 4 compares SeamFlow with the 500-step Dense-RAM checkpoint on twelve selected meshes, showing each 3D surface alongside its UV atlas. Across these examples, Dense-RAM reduces fragmentation on both regular objects and articulated shapes. For the sofa and mechanical housing, UV-island counts fall from 1,436 to 46 and from 189 to 27, respectively, replacing many small fragments with larger regions. The rabbit's island count decreases from 20 to 10, while the stylized skeleton decreases from 646 to 172 despite its more intricate structure. Fewer UV regions can simplify texture editing by reducing the number of island boundaries to manage. These examples illustrate the overall effect of post-training; the paired comparisons in Section 5.3 isolate the contribution of edge-level credit assignment.

5.3 Effectiveness of Edge-Level Credit Assignment

We compare Scalar and Dense allocation under DiffusionNFT, AWM, and RAM on the same UV task (Zheng et al., 2026; Xue et al., 2026). Within each objective, the two variants share the backbone,

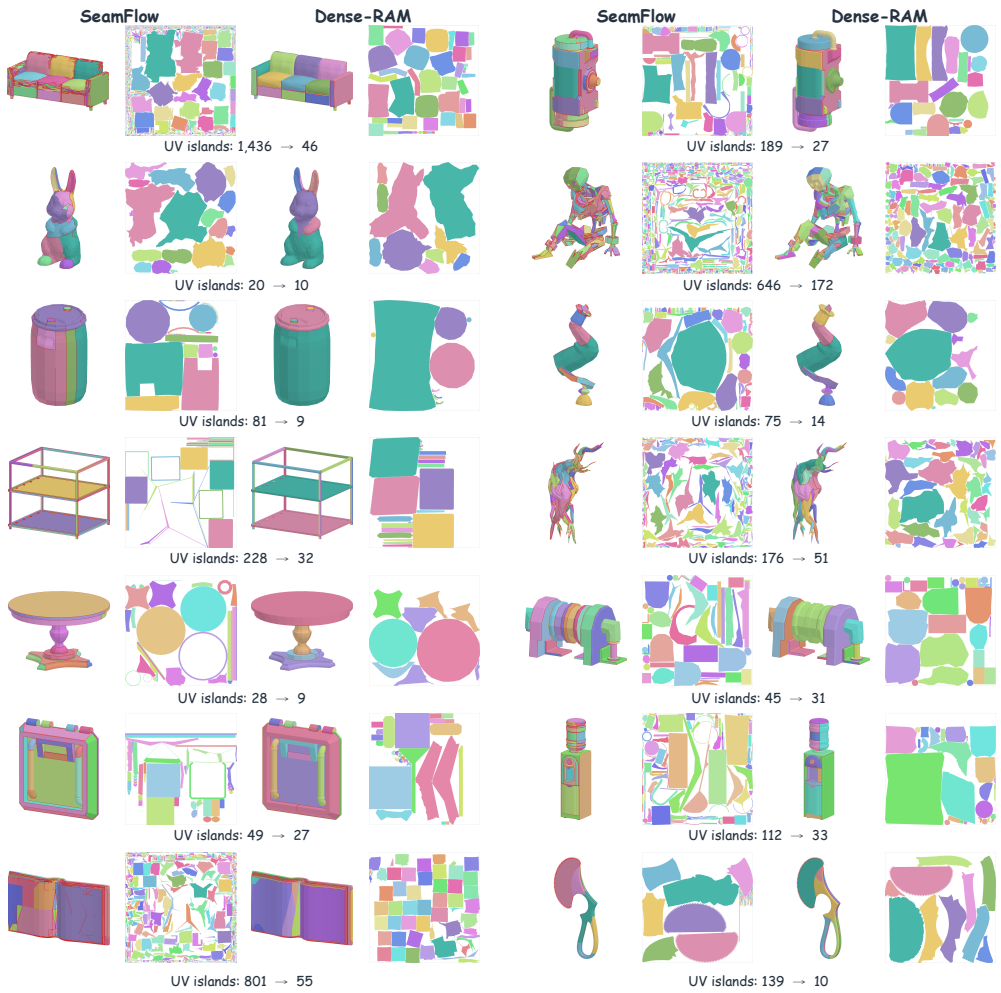


Figure 4: **Qualitative effects of Dense-RAM post-training (500 steps).** Red lines mark seams; colors link UV islands to their 3D regions within each output.Arrows show final UV-island count changes.

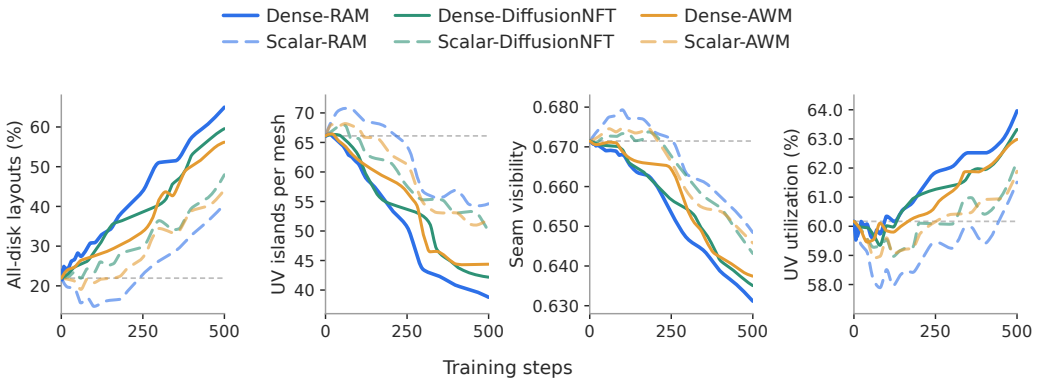


Figure 5: **Evaluation curves for UV quality.** All-disk rate (↑), UV islands (↓), visibility (↓), and UV utilization (↑), from left to right. Colors identify RL methods; solid and dashed curves denote Dense and Scalar variants, respectively. Gray dashed lines indicate the original SeamFlow baseline.
initial checkpoint, reward configuration, and training settings; only the allocation of reward feedback across edges changes.

As shown in Table 3, after 500 optimizer steps, Dense allocation improves all four reported metrics under each objective. For RAM, it raises the all-disk rate from 40.50% to 64.98% and UV utilization

from 61.52% to 63.96%, while reducing UV-island count from 54.62 to 37.24 and seam visibility from 0.6484 to 0.6311. The same direction of improvement under DiffusionNFT and AWM supports the benefit of dense allocation beyond RAM on this UV task.

Figure 5 shows how UV quality evolves during training. Dense variants generally reach comparable metric values in fewer optimizer steps than their Scalar counterparts under the paired training settings. Together with the 500-step results in Table 3, these curves show that edge-level allocation improves both final UV quality and learning progress over a fixed training schedule.

5.4 Reward Ablations

Table 4 evaluates the contribution of topology, distortion, island count, and seam visibility to the reward. Each ablation sets the corresponding weight λ_k to zero while keeping the other reward weights, initial checkpoint, training settings, and reward-evaluation budget fixed.

Removing the topology reward lowers the all-disk rate from 65.0% to 49.2%. Removing the distortion reward increases angular and area distortion from 0.11 and 0.12 to 0.17 and 0.21, respectively, while removing the visibility reward raises seam visibility from 0.631 to 0.676. These changes show that the reward components contribute to complementary aspects of UV quality.

The strongest trade-off appears when the island-count reward is removed: the all-disk rate rises from 65.0% to 92.1%, and angular and area distortion fall from 0.11 and 0.12 to 0.08 and 0.09, but the mean UV-island count increases from 37.2 to 312.4 and UV utilization drops from 64.0% to 56.0%. These results show that favorable topology and distortion scores can accompany excessive fragmentation. The island-count reward counterbalances this tendency, favoring less fragmented layouts with higher texture-space occupancy.

Table 3: Credit-assignment comparisons across RL objectives at a matched reward-evaluation budget. Each objective is evaluated through a Scalar / Dense pair.

Objective	Allocation	All-disk(%) ↑	UV islands ↓	Visibility ↓	UV utilization(%) ↑
SeamFlow	—	21.94	66.42	0.6701	60.17
DiffusionNFT	Scalar	47.98	49.78	0.6431	62.26
DiffusionNFT	Dense	59.58	42.18	0.6350	63.32
AWM	Scalar	44.24	52.20	0.6458	61.89
AWM	Dense	56.18	44.38	0.6374	62.98
RAM	Scalar	40.50	54.62	0.6484	61.52
RAM	Dense	**64.98**	**37.24**	**0.6311**	**63.96**

Table 4: Ablation results for different reward configurations.

Variant	All-disk (%) ↑	Angle dist. ↓	Area dist. ↓	UV islands ↓	Visibility ↓	UV util. (%) ↑
SeamFlow	21.9	0.13	0.16	66.4	0.670	60.2
w/o topology	49.2	0.12	0.13	35.2	0.628	62.8
w/o distortion	64.7	0.17	0.21	36.4	0.627	63.2
w/o island count	92.1	0.08	0.09	312.4	0.650	56.0
w/o visibility	65.6	0.11	0.12	39.2	0.676	63.5
Full reward	65.0	0.11	0.12	37.2	0.631	64.0

6 Conclusion and Discussion

We present Dense-RAM, a reinforcement learning framework for post-training flow-matching seam generators. Dense-RAM allocates candidate-level advantages according to local UV quality, introducing edge-level credit assignment into the RAM regression target while preserving each candidate's mean advantage. Across RAM, DiffusionNFT, and AWM, dense allocation improves the all-disk rate and UV utilization and reduces fragmentation and seam visibility relative to scalar allocation after 500 optimizer steps. Under the paired training settings, dense allocation also generally reaches comparable metric values earlier in training.

Limitations. Dense-RAM relies on hand-designed UV rewards, and reduced fragmentation may coincide with higher chart distortion. Future work will explore adaptive rewards and broader validation across other tasks.

AI USE STATEMENT

In this work, we used generative AI tools for polishing writing and LaTeX editing. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Andreas Bergmeister, Stefanie Jegelka, Nikolas Nüsken, Carles Domingo-Enrich, and Jakiw Pidstrigach. Reinforce adjoint matching: Scaling rl post-training of diffusion and flow-matching models. *ArXiv*, abs/2605.10759, 2026.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 4965–4987, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/14f75513f0f1ca01de1e826b52e6b840-Paper-Conference.pdf.

Yuguang Chen, Xinhai Liu, Yang Li, Victor Cheung, Zhuo Chen, Dongyu Zhang, and Chunchao Guo. ArtUV: Artist-style UV unwrapping. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LN7VQ3ed1t.

Sagnik Das, Ke Ma, Zhixin Shu, and Dimitris Samaras. Learning an isometric surface parameterization for texture unwrapping. In *European Conference on Computer Vision*, pp. 580–597. Springer, 2022.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-XL: A universe of 10m+ 3D objects. In *Advances in Neural Information Processing Systems*, volume 36, pp. 35799–35813. Curran Associates, Inc., 2023a. doi: 10.52202/075280-1554. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/70364304877b5e767de4e9a2a511be0c-Paper-Datasets_and_Benchmarks.pdf.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3D objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13142–13153, June 2023b. doi: 10.1109/CVPR52729.2023.01263.

Haoyou Deng, Keyu Yan, Chaojie Mao, Xiang Wang, Yu Liu, Changxin Gao, and Nong Sang. Densegrpo: From sparse to dense reward for flow matching model alignment. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 76096–76109, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/7b2e844c52349134268e819a9b56b9e8-Paper-Conference.pdf.

Mathieu Desbrun, Mark Meyer, and Pierre Alliez. Intrinsic parameterizations of surface meshes. In *Computer graphics forum*, volume 21, pp. 209–218. Wiley Online Library, 2002.

Carles Domingo i Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 53791–53846, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/852f50969a9e523ec41d26f2f68bd456-Paper-Conference.pdf.

Michael S Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

Michael S Floater and Kai Hormann. Surface parameterization: a tutorial and survey. *Advances in Multiresolution for Geometric Modelling*, pp. 157–186, 2005.

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proc. CVPR*, pp. 216–224, 2018.

Kai Hormann and Günther Greiner. Mips: An efficient global parametrization method. *Curve and Surface Design: Saint-Malo 1999*, 2000.

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, 21(3):362–371, 2002.

Minchen Li, Danny M Kaufman, Vladimir G Kim, Justin Solomon, and Alla Sheffer. Optcuts: Joint optimization of surface cuts and parameterization. *ACM Transactions on Graphics*, 37(6):1–13, 2018.

Yang Li, Victor Cheung, Xinhai Liu, Yuguang Chen, Zhongjin Luo, Biwen Lei, Haohan Weng, Zibo Zhao, Jingwei Huang, Zhuo Chen, et al. Auto-regressive surface cutting. *arXiv preprint arXiv:2506.18017*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di ZHANG, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 40783–40818. Curran Associates, Inc., 2025a. doi: 10.52202/085713-1362. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3a10c46572628d58cb44fb705f25cbbf-Paper-Conference.pdf.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J Gortler. A local/global approach to mesh parameterization. In *Computer Graphics Forum*, volume 27, pp. 1495–1504, 2008.

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9704–9715, 2025b.

Richard Liu, Noam Aigerman, Vladimir G Kim, and Rana Hanocka. Da wand: Distortion-aware selection using neural mesh parameterization. In *Proc. CVPR*, pp. 16739–16749, 2023a.

Xingchao Liu, Chengyue Gong, et al. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations*, 2023b.

Yihong Luo, Tianyang Hu, and Jing Tang. Reinforcing diffusion models by direct group preference optimization. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 21438–21454, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/244da015b91e64f2d9362703fa2a902b-Paper-Conference.pdf.

Xueqi Ma, Xingguang Yan, Congyue Zhang, and Hui Huang. Meshtailor: Cutting seams via generative mesh traversal, 2026. URL https://arxiv.org/abs/2603.27309.

Luca Morreale, Noam Aigerman, Vladimir G Kim, and Niloy J Mitra. Neural surface maps. In *Proc. CVPR*, pp. 4639–4648, 2021.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. Scalable locally injective mappings. *ACM Transactions on Graphics*, 36(4):1, 2017.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Proc. NeurIPS*, 2023.

Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pp. 409–416, 2001.

Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. Abf++: fast and robust angle based flattening. *ACM Transactions on Graphics*, 24(2):311–330, 2005.

Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, 02 2007. ISSN 1572-2740. doi: 10.1561/0600000011. URL https://doi.org/10.1561/0600000011.

Jason Smith and Scott Schaefer. Bijective parameterization with free boundaries. *ACM Transactions on Graphics*, 34(4):1–9, 2015.

Pratul P Srinivasan, Stephan J Garbin, Dor Verbin, Jonathan T Barron, and Ben Mildenhall. Nuvo: Neural uv mapping for unruly 3d representations. In *Proc. ECCV*, pp. 18–34, 2025.

George Tang, William Zhao, Logan Ford, David Benhaim, and Paul Zhang. Segment any mesh. *arXiv preprint arXiv:2408.13679*, 2024.

Zhaoning Wang, Xinyue Wei, Ruoxi Shi, Xiaoshuai Zhang, Hao Su, and Minghua Liu. Partuv: Part-based uv unwrapping of 3d meshes. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–12, 2025.

Duoteng Xu, Yuguang Chen, Jing Li, Xinhai Liu, Xueqi Ma, Zhuo Chen, Dongyu Zhang, and Chunchao Guo. Seamcrafter: Enhancing mesh seam generation for artist uv unwrapping via reinforcement learning. *arXiv preprint arXiv:2509.20725*, 2025.

Shuchen Xue, Chongjian GE, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning RL with pretraining in diffusion models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=nLY2pOYBrJ.

Shentao Yang, Tianqi Chen, and Mingyuan Zhou. A dense reward view on aligning text-to-image diffusion with preference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 55998–56032. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/yang24e.html.

Amirhossein Zamani, Bruno Roy, and Arianna Rampini. Unsupervised representation learning for 3d mesh parameterization with semantic and visibility objectives. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 139261–139290, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e17afd5784c442d178744533c16d3c96-Paper-Conference.pdf.

Qijian Zhang, Junhui Hou, Wenping Wang, and Ying He. Flatten anything: Unsupervised neural surface parameterization. In *Proc. NeurIPS*, 2024.

Yuming Zhao, Qijian Zhang, Junhui Hou, Jiazhi Xia, Wenping Wang, and Ying He. Flexpara: Flexible neural surface parameterization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Yuming Zhao, Junhui Hou, Qijian Zhang, Jia Qin, and Ying He. From extrinsic to intrinsic: Geodesic-guided representation learning for 3d geometric data. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=zupxQlD4cw.

Yuming Zhao, Zangyueyang Xian, Qijian Zhang, Rendong Liang, Qin Jia, Ying He, and Junhui Hou. Seamflow: Structure-aware flow matching on edge probabilities for artist-like uv unwrapping. *ArXiv*, abs/2609.04751, 2026b.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. Diffusionnft: Online diffusion reinforcement with forward process. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 134129–134150, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/d8e68ddfe22520b45b8fb8d5cbde5a21-Paper-Conference.pdf.

A Algebra of Advantage-Preserving Allocation

This appendix characterizes the allocation rule of Eqs. (7)–(8) and its effect on the RAM update. Section A.1 collects the coefficient properties stated in Eq. (10) and identifies the regime in which clipping is active. Section A.2 shows that the Dense-RAM update equals the scalar RAM update plus a within-candidate contrast term, and gives that term in closed form. Throughout, we fix a mesh, an evaluated group, and one candidate, and suppress the candidate index g. We assume $m \geq 1$ valid edges, finite scores, $\alpha \geq 0$, and a positive finite scale σ_d shared across the candidate's valid edges; all sums and means run over valid edges only, excluding padding. When all edges of a candidate share the same score, $d \equiv 0$ and the rule returns $w_e = 1$ for any positive σ_d, so a lower floor on σ_d does not change the allocation. For scalar $a \in \mathbb{R}^m$ and scalar- or vector-valued b indexed by edges, we write $\mathrm{Cov}_e[a, b] = m^{-1} \sum_e (a_e - \bar{a})(b_e - \bar{b})$ for the empirical covariance across the m edges (componentwise when b is vector-valued) and $\mathrm{Var}_e[a] = \mathrm{Cov}_e[a, a]$, and we adopt the convention $\mathrm{sign}(0) = 0$.

A.1 Coefficient Identities

Recall $b = m^{-1} \sum_e q_e$, $d_e = q_e - b$, $\tilde{w}_e = \left[1 + \alpha\ \mathrm{sign}(A)\, d_e/\sigma_d\right]_+$, $z = m^{-1} \sum_e \tilde{w}_e$, $w_e = \tilde{w}_e / z$, and $C_e = A\, w_e$. When no edge is clipped, the rule has the closed form

$$C_e \;=\; A \;+\; \frac{\alpha\,|A|}{\sigma_d}\, d_e, \tag{11}$$

that is, the scalar coefficient plus a zero-mean local contrast whose strength is proportional to $|A|$. Eq. (11) is the object the rest of this appendix is built on; the two propositions below state when it holds and which of its properties survive clipping.

Proposition 1 (Well-defined weights and mean conservation)**.** *$z \geq 1$, with equality if and only if no edge is clipped. Consequently the weights are well defined and satisfy*

$$w_e \geq 0, \qquad \frac{1}{m} \sum_e w_e = 1, \qquad \frac{1}{m} \sum_e C_e = A. \tag{12}$$

Every nonzero C_e has the sign of A. If $\alpha = 0$, then $w_e = 1$ and $C_e = A$ for all e. If $A = 0$, then $C_e = 0$ for all e. If no edge is clipped, Eq. (11) holds.

Proof. Since $\sum_e d_e = 0$ and $[x]_+ \geq x$,

$$z \;\geq\; \frac{1}{m} \sum_e \Big(1 + \alpha\ \mathrm{sign}(A)\, \frac{d_e}{\sigma_d}\Big) \;=\; 1,$$

with equality iff $[x]_+ = x$ for every edge, i.e. iff no edge is clipped. Nonnegativity of w_e, the unit mean, and $m^{-1} \sum_e C_e = A \cdot m^{-1} \sum_e w_e = A$ follow directly; $C_e = A w_e$ with $w_e \geq 0$ gives sign preservation. For $\alpha = 0$, $\tilde{w}_e = 1$ for all e. For $A = 0$, $\mathrm{sign}(A) = 0$ gives $w_e = 1$ and $C_e = 0$. Without clipping, $z = 1$ and $C_e = A\big(1 + \alpha\ \mathrm{sign}(A)\, d_e/\sigma_d\big) = A + \alpha |A|\, d_e/\sigma_d$ because $A\ \mathrm{sign}(A) = |A|$. □

Proposition 2 (Coefficient magnitudes)**.** *With or without clipping,*

$$\frac{1}{m} \sum_e |C_e| \;=\; |A|, \qquad \frac{1}{m} \sum_e C_e^2 \;=\; A^2 \big(1 + \mathrm{Var}_e[w]\big) \;\geq\; A^2. \tag{13}$$

Without clipping, $\mathrm{Var}_e[w] = \alpha^2\ \mathrm{Var}_e[d]/\sigma_d^2$; in particular, if σ_d is the within-candidate standard deviation of d, then $m^{-1} \sum_e C_e^2 = (1 + \alpha^2)\, A^2$.

Proof. $|C_e| = |A|\, w_e$ and Eq. (12) give the first identity. For the second, expand $w_e^2 = (w_e - 1)^2 + 2(w_e - 1) + 1$ and use $\sum_e (w_e - 1) = 0$, so that $m^{-1} \sum_e w_e^2 = 1 + \mathrm{Var}_e[w]$. Without clipping, $w_e - 1 = \alpha\ \mathrm{sign}(A)\, d_e/\sigma_d$, whose variance across edges is $\alpha^2\ \mathrm{Var}_e[d]/\sigma_d^2$. □

Allocation therefore preserves the mean and the mean absolute coefficient but raises the second moment: the correction is concentrated on edges whose local score deviates from the candidate mean. For a positive advantage the weights are nondecreasing in d_e; for a negative advantage the order is reversed; clipping can create ties at zero. Because the chart-level rewards in Appendix B are binary, the local score of a candidate often takes only two values on its edges, and in that case the clipping regime can be read off explicitly.

Remark 1 (When clipping is active). Suppose σ_d is the within-candidate standard deviation of d and the local score takes exactly two values on the candidate's valid edges, the smaller group holding a fraction $p \le \frac{1}{2}$ of the edges. Then $|d_e|/\sigma_d = \sqrt{(1-p)/p}$ on the smaller group and $\sqrt{p/(1-p)} \le 1$ on the larger group, and an edge is clipped iff $\operatorname{sign}(A)\, d_e/\sigma_d < -1/\alpha$. Hence, for $\alpha \le 1$, the larger group is never clipped, and the smaller group is clipped iff it lies on the side opposite to $\operatorname{sign}(A)$ and

$$p \;<\; \frac{\alpha^2}{1+\alpha^2}.$$

For $\alpha = 0.5$ this threshold is 0.2: clipping is active exactly when fewer than 20% of a candidate's edges are locally worse than the rest of a positive-advantage candidate, or locally better than the rest of a negative-advantage candidate. Those edges then receive coefficient 0 and, by mean-one normalization, every other edge receives coefficient $A/(1-p)$.

A.2 Decomposition of the Dense-RAM Update

Fix a training row (X_t, t, M) and let $J_\theta = \nabla_\theta v_\theta(X_t, t, M) \in \mathbb{R}^{m\times p}$, with $J_e \in \mathbb{R}^{1\times p}$ its e-th row. The endpoint, the noising input, the advantages, and the regression targets are detached, so the per-row gradients of the scalar and dense objectives (Eqs. (3) and (9)) are

$$\hat{g}_{\mathrm{S}} \;=\; \frac{2}{m}\, J_\theta^\top\big(\Delta - A\,H\big), \qquad \hat{g}_{\mathrm{D}} \;=\; \frac{2}{m}\, J_\theta^\top\big(\Delta - C \odot H\big), \tag{14}$$

with $\Delta = v_\theta - v^{\mathrm{ref}}$ and $H = Y - v_\theta$ as in Eq. (2). Both are sums of per-edge contributions. Writing

$$u_e \;=\; H_e\, J_e^\top \;\in\; \mathbb{R}^p, \qquad \bar{u} \;=\; \frac{1}{m}\sum_e u_e, \tag{15}$$

we have $J_\theta^\top(c \odot H) = \sum_e c_e\, u_e$ for any coefficient vector $c \in \mathbb{R}^m$. The vector u_e is the parameter direction that moves the edge-e prediction toward its regression target: the directional derivative of $v_{\theta,e}$ along u_e is $J_e u_e = H_e \|J_e\|^2$, which has the sign of $H_e = Y_e - v_{\theta,e}$. We call u_e the *pull* of edge e and $\bar{u}$ the mean pull. In this notation the two updates read

$$\hat{g}_{\mathrm{S}} \;=\; \frac{2}{m} J_\theta^\top \Delta \;-\; 2A\,\bar{u}, \qquad \hat{g}_{\mathrm{D}} \;=\; \frac{2}{m} J_\theta^\top \Delta \;-\; 2A\big(\bar{u} + \mathrm{Cov}_e[w, u]\big), \tag{16}$$

which is the content of the following proposition.

Proposition 3 (Covariance form of the update difference)**.** *For any allocation with $w_e \ge 0$ and $m^{-1}\sum_e w_e = 1$,*

$$\hat{g}_{\mathrm{D}} - \hat{g}_{\mathrm{S}} \;=\; -\,2A\;\mathrm{Cov}_e\big[w,\, u\big]. \tag{17}$$

When no edge is clipped,

$$\hat{g}_{\mathrm{D}} - \hat{g}_{\mathrm{S}} \;=\; -\,\frac{2\alpha\,|A|}{\sigma_d}\;\mathrm{Cov}_e\big[d,\, u\big] \;=\; \frac{2}{m}\, J_\theta^\top\Big(-\frac{\alpha\,|A|}{\sigma_d}\; d \odot H\Big). \tag{18}$$

In particular, the two updates coincide whenever $A = 0$, whenever $\alpha = 0$, or whenever the allocation weights are uncorrelated with the pulls across edges. Moreover,

$$\big\|\hat{g}_{\mathrm{D}} - \hat{g}_{\mathrm{S}}\big\| \;\le\; 2\,|A|\,\sqrt{\mathrm{Var}_e[w]}\,\Big(\frac{1}{m}\sum_e \|u_e - \bar{u}\|^2\Big)^{1/2}. \tag{19}$$

Proof. From Eq. (14) and $C_e = A w_e$,

$$\hat{g}_{\mathrm{D}} - \hat{g}_{\mathrm{S}} \;=\; \frac{2}{m}\, J_\theta^\top\big((A\mathbf{1} - C) \odot H\big) \;=\; -\,\frac{2A}{m}\sum_e (w_e - 1)\, u_e.$$

Because $\sum_e (w_e - 1) = 0$, subtracting the constant $\bar{u}$ inside the sum changes nothing: $m^{-1} \sum_e (w_e - 1)\, u_e = m^{-1} \sum_e (w_e - 1)(u_e - \bar{u}) = \mathrm{Cov}_e[w, u]$, since w has mean one. This proves Eq. (17), and Eq. (16) follows by writing $m^{-1} \sum_e w_e u_e = \bar{u} + \mathrm{Cov}_e[w, u]$. Without clipping, Proposition 1 gives $w_e - 1 = \alpha\, \mathrm{sign}(A)\, d_e / \sigma_d$, and $A\, \mathrm{sign}(A) = |A|$ yields the first equality in Eq. (18); the second follows from $\sum_e d_e = 0$ by the same centering argument read backwards. The cases $A = 0$ and $\alpha = 0$ are immediate from $w_e = 1$. Eq. (19) is the Cauchy–Schwarz inequality applied to $m^{-1} \sum_e (w_e - 1)(u_e - \bar{u})$, using $\sum_e (w_e - 1) = 0$. □

All identities above are linear in the row, so they hold verbatim for the K-draw and G-candidate averages in Eq. (9), with u_e replaced by its average over the analytic noising draws of a candidate and $A\ \mathrm{Cov}_e[w, u]$ by its average over the candidates of a group.

Under gradient descent $\theta \leftarrow \theta - \eta\, \hat{g}$, the scalar step moves every edge toward the endpoint of a positive-advantage candidate, and away from that of a negative-advantage one, with the common coefficient A; by Eq. (18), Dense-RAM adds a *within-candidate contrast* that, relative to the mean pull, moves further toward the candidate at edges above its own mean score and away from it at edges below (with clipping, w_e is nondecreasing in $\mathrm{sign}(A)\, d_e$, so Eq. (17) reads the same way). The direction of this contrast does not depend on the sign of A, only its strength does: a good candidate is imitated more strongly at its locally good edges and a bad candidate is rejected more strongly at its locally bad edges, which is the same direction in parameter space. Because the contrast is scaled by $|A|$, a candidate at the group mean contributes no local signal however uneven its charts are; this is the price of preserving the candidate advantage, and it is what separates Dense-RAM from independent per-edge advantages, which keep local signal at $A = 0$ at the cost of independent reinforcement signs within a candidate. Since the row gradient is affine in the coefficient vector, Eq. (18) also identifies Dense-RAM as scalar RAM with an additional centered per-edge coefficient inside the regression target, not a reweighting of the squared loss.

Whether the added term acts as a larger step or as a reallocation is not fixed by the identities. For $\bar{u} \neq 0$, decomposing the contrast along and orthogonal to the scalar direction,

$$-2A\ \mathrm{Cov}_e[w, u] \;=\; \gamma\,(-2A\,\bar{u}) + r, \qquad \gamma = \frac{\langle \mathrm{Cov}_e[w, u],\ \bar{u} \rangle}{\|\bar{u}\|^2}, \qquad r \perp \bar{u}, \tag{20}$$

the dense update is a scalar update with effective advantage $(1 + \gamma)A$ plus an orthogonal reallocation r; γ and $\|r\|$ are the quantities to measure before attributing a difference in training progress to allocation rather than to step size. Finally, Eq. (17) is exact for every row, so at a fixed checkpoint the dense and scalar updates have the same mean if and only if $\mathbb{E}[A\ \mathrm{Cov}_e[w, u]] = 0$. The effects of this update are empirically evaluated by Section5.3 and Figure 5 under a matching step budget.

B SeamFlow Pretraining and Dense-RAM Method

This appendix describes the supervised seam generator that initializes Dense-RAM, the construction of UV-quality rewards, and the post-training procedure.

B.1 Pretrained SeamFlow Generator

SeamFlow learns an artist-derived seam prior through supervised conditional flow matching on curated meshes and UV layouts (Zhao et al., 2026b). Each mesh edge is embedded from its endpoint coordinates, while a global encoder extracts shape features from surface positions and normals. The velocity network combines edge self-attention with cross-attention to these shape features, predicting all edge velocities jointly.

Let $P \in [0, 1]^m$ denote the relaxed seam target in the original pretraining formulation. For Gaussian noise $\epsilon \sim \mathcal{N}(0, I)$, time $t \sim \mathcal{U}(0, 1)$, and geometry conditioning H_M, pretraining minimizes

$$\begin{aligned} Z_t &= (1 - t)\epsilon + tP, \\ \mathcal{L}_{\mathrm{pre}}(\phi) &= \mathbb{E}_{M,P,t,\epsilon}\left[\|\mathcal{V}_\phi(Z_t, t, H_M) - (P - \epsilon)\|_2^2 \right]. \end{aligned} \tag{21}$$

Here (M, P) follows the training data, and $\mathcal{V}_\phi$ is the pretrained velocity predictor. Generation integrates this field from Gaussian noise to produce a seam layout conditioned on the mesh.

Dense-RAM retains the generator architecture and initializes both the trainable policy and frozen reference from the same supervised checkpoint. Post-training refines the seam prior through evaluated UV quality and edge-level target modulation, without adding a learned allocation network.

B.2 UV Evaluation and Reward Components

For candidate g, we threshold the continuous edge scores as $y_{ge} = \mathbf{1}[S_e^{(g)} > 0.5]$, cut the mesh along the predicted seams, and parameterize the resulting charts. The seam-induced charts $\mathcal{T}_g^{\mathrm{cut}}$ and final UV islands $\mathcal{I}_g^{\mathrm{UV}}$ are distinguished because parameterization may introduce additional cuts. Chart-level feedback is mapped to valid model edges by averaging over their nonempty incident-chart sets $\mathcal{N}_g(e)$, as defined in Eq. (5). Layout-level feedback is shared across all edges of a candidate.

The four reward components instantiate Eq. (4) as

$$q_{ge} = r_{ge}^{\mathrm{topo}} + r_{ge}^{\mathrm{dist}} + r_g^{\mathrm{island}} + 0.5 r_g^{\mathrm{ao}}. \tag{22}$$

Chart-level topology and distortion. Topology rewards disk-like charts, while distortion rewards charts whose Dirichlet energy remains below a threshold:

$$\begin{aligned} r_{gc}^{\mathrm{topo}} &= 2\mathbf{1}[\chi(c) = 1 \text{ and } B(c) = 1] - 1, \\ r_{gc}^{\mathrm{dist}} &= 2\mathbf{1}[\text{energy evaluation succeeds and } E_{gc}^{\mathrm{D}} \le \tau_{\mathrm{D}}] - 1. \end{aligned} \tag{23}$$

Here $\chi(c)$ is the Euler characteristic and $B(c)$ the number of boundary loops. Each test yields $+1$ when satisfied and -1 otherwise. A failed Dirichlet energy evaluation receives a distortion reward of -1. Averaging these rewards over incident charts preserves regional quality differences in the edge scores. The binary distortion reward complements the continuous distortion measures used for evaluation in Appendix C.

Layout-level fragmentation and visibility. The island and ambient-occlusion terms are

$$\begin{aligned} r_g^{\mathrm{island}} &= -\tanh\left(\frac{n_g^{\mathrm{reward}} - 26}{30}\right), \\ r_g^{\mathrm{ao}} &= \mathrm{clip}(1 - 2\nu_g, -1, 1), \qquad \nu_g = \frac{\sum_e y_{ge}\ell_e(1 - \mathrm{ao}_e)}{\sum_e y_{ge}\ell_e}. \end{aligned} \tag{24}$$

Here n_g^{reward} is the island count used for reward computation, kept distinct from the final UV-island count reported after parameterization. The reference count 26 is the median chart count of the original model, and scale 30 controls the smooth transition of the fragmentation penalty. In the visibility term, ℓ_e denotes edge length and $\mathrm{ao}_e = 1$ denotes full occlusion; the ratio is defined when the selected seams have positive total length. The island term discourages fragmentation, while the visibility term favors seams in occluded regions.

These layout-level terms influence the candidate advantage but cancel in the within-candidate deviations $d_{ge} = q_{ge} - b_g$, because their values are shared across edges. The topology and distortion terms provide the local differences that guide allocation. Table 4 evaluates the contribution of each reward component.

B.3 Advantage Normalization and Allocation

For each mesh, we evaluate G sampled candidates with the same reward function. The candidate score b_g averages the raw scores over valid edges, and group-centering and scaling these scores gives the scalar advantages in Eq. (6). The reward scale $\widehat{\sigma}$ is estimated from pooled reward statistics with an exponential moving average across training steps. The local-deviation scale σ_d is shared across the valid edges of a candidate; both scales are assumed positive and finite.

Local deviations modulate the candidate advantage through Eq. (7). Clipping makes the allocation weights nonnegative, and normalization gives them a mean of one across valid edges. Consequently, $C_{ge} = A_g w_{ge}$ preserves the mean candidate advantage while adjusting the correction strength at each edge. Setting $\alpha = 0$ recovers uniform scalar feedback. Advantages and allocation coefficients are detached when constructing the regression target.

B.4 Endpoint Sampling and Reward Reuse

The current policy generates seam-score endpoints by integrating its velocity field from independent Gaussian initial states. Sampling is deterministic conditional on the initial noise and policy parameters, while the endpoint distribution changes as the policy is updated.

Each evaluated endpoint supplies K independent analytic noising pairs (t, ϵ) for the regression objective in Eq. (9). The two variants share the post-training time distribution on $[0, 1]$ and use independent noise $\epsilon \sim \mathcal{N}(0, I)$. These pairs reuse the same UV feedback, allowing multiple regression targets to be constructed without additional UV evaluations. The frozen reference anchors the targets, and edge-level coefficients modulate their reward correction. Scalar and Dense variants follow the same procedure, differing only in the allocation of feedback across edges.

B.5 Dense-AWM and Dense-DiffusionNFT Implementation

Shared reward processing. Both adapters retain the sampling, rollout, and analytic noising procedure of their scalar counterparts. The `scalar_dense` reward mode uses the group-centered, EMA-scaled advantage A_g and the mean-one edge weights from Eq. (7) to form $C_{ge} = A_g w_{ge}$; invalid edges receive zero. For N endpoints and a padded edge dimension $S_{\max}$, this replaces a scalar reward tensor of shape $[N]$ with $[N, S_{\max}]$. Each endpoint's reward row is repeated across its K noising draws, producing $[NK, S_{\max}]$. All rewards and weights are detached, and all edge means below exclude padding. We suppress the noising-draw index and write u for the flow target Y in Eq. (1).

Dense-AWM. Our AWM adapter uses the signed advantage-weighted matching loss with a reference penalty (Xue et al., 2026):

$$\mathcal{L}^{\mathrm{D}}_{\mathrm{AWM}} = \mathbb{E}\left[\frac{1}{m}\sum_{e} C_{ge}(v_{\theta,ge} - u_{ge})^2 + \beta_{\mathrm{KL}}\frac{1}{m}\sum_{e}(v_{\theta,ge} - v^{\mathrm{ref}}_{ge})^2\right]. \tag{25}$$

The dense advantage multiplies the elementwise squared matching error before averaging over valid edges; the reference penalty remains uniformly weighted. The reference is the frozen supervised model, and the final loss averages over noising rows. Scalar AWM broadcasts A_g over edges, so $\alpha = 0$ recovers exactly the same loss. Mean advantage is preserved, but the weighted loss and gradient magnitude can change because matching residuals differ across edges.

Dense-DiffusionNFT. The NFT adapter maps each edge coefficient to a bounded positive-branch weight and retains the implicit positive and negative policies (Zheng et al., 2026):

$$\begin{aligned} p_{ge} &= \mathrm{clip}\left(\frac{1}{2} + \frac{C_{ge}}{2\beta s_{\mathrm{pos}}}, 0, 1\right), \\ v^{+}_{ge} &= v^{\mathrm{old}}_{ge} + b_{\mathrm{NFT}}(v_{\theta,ge} - v^{\mathrm{old}}_{ge}), \\ v^{-}_{ge} &= v^{\mathrm{old}}_{ge} - b_{\mathrm{NFT}}(v_{\theta,ge} - v^{\mathrm{old}}_{ge}), \\ \mathcal{L}^{\mathrm{D}}_{\mathrm{NFT}} &= \mathbb{E}\left[\frac{1}{m}\sum_{e}\left\{p_{ge}(v^{+}_{ge} - u_{ge})^2 + (1 - p_{ge})(v^{-}_{ge} - u_{ge})^2\right\}\right]. \end{aligned} \tag{26}$$

Here β is the reward coefficient in Eq. (6), $s_{\mathrm{pos}} > 0$ is `nft_pos_scale`, and $b_{\mathrm{NFT}} > 0$ controls the implicit branches. Dividing by β removes the reward scaling before the probability mapping. Dense feedback enters only through p_{ge}, which mixes the two squared errors separately for each edge before reduction. By default, $v^{\mathrm{old}} = \mathrm{sg}(v_\theta)$ (`nft_old=policy`), with no EMA update or additional KL term; the `ref` option instead uses the frozen supervised model. Under the default setting, $p_{ge} = 1/2$ gives zero gradient at that edge. Setting $\alpha = 0$ recovers scalar NFT exactly. Clipping can change the mean signed coefficient $m^{-1}\sum_e(2p_{ge} - 1)$; a larger s_{pos} or smaller allocation strength reduces saturation. Scalar and Dense controls use the same s_{pos}, branch parameter, and old-policy setting.

C Evaluation Protocols and Cross-Objective Comparisons

This appendix describes the UV evaluation metrics, paired allocation comparisons, and qualitative analysis. Table 3 measures final quality after 500 training steps, while Figure 5 tracks progress toward improved seam layouts during training.

C.1 UV Evaluation Protocol

Task-level comparisons follow the baseline coverage of SeamFlow and include learned seam-generation references. Methods are compared on the same declared mesh split with a shared metric implementation. Complete UV systems retain their native parameterization stage; the SeamFlow post-training variants share the same parameterization pipeline. Consequently, task-level comparisons evaluate complete UV systems, while Scalar / Dense comparisons hold the downstream UV pipeline fixed. Published numbers obtained on another dataset or backend are identified separately from common-protocol measurements.

The evaluation distinguishes seam-induced topology from the geometry of the final UV atlas. All-disk rate tests the predicted charts before parameterization, while angular and area distortion measure deformation in the resulting UV maps. Final UV-island count, seam visibility, UV utilization, and inference time describe fragmentation, seam placement, texture-space occupancy, and deployment cost. The count used by the island reward is distinguished from final UV-island count, since parameterization may introduce further cuts.

Metrics are averaged over candidates within each mesh before aggregation across meshes. Failure counts use all attempted meshes, whereas each metric average states its valid subset.

Angular and area distortion. For task-level UV comparison, we follow the area-weighted log-distortion measures in SeamFlow (Zhao et al., 2026b). For each successfully parameterized triangular face f, let $J_f \in \mathbb{R}^{3\times 2}$ be the Jacobian of the affine map from UV coordinates to the 3D surface, with positive singular values $\sigma_{1f} \geq \sigma_{2f} > 0$. Let a_f^{3D} and a_f^{UV} denote its surface and UV areas, respectively, and define the global area scale

$$\bar{\rho} = \frac{\sum_f a_f^{\mathrm{3D}}}{\sum_f a_f^{\mathrm{UV}}}. \tag{27}$$

The two distortion measures are

$$\begin{aligned} D_{\mathrm{angle}} &= \left[\frac{\sum_f a_f^{\mathrm{3D}} \left(\ln \frac{\sigma_{1f}}{\sigma_{2f}} \right)^2}{\sum_f a_f^{\mathrm{3D}}} \right]^{1/2}, \\ D_{\mathrm{area}} &= \left[\frac{\sum_f a_f^{\mathrm{3D}} \left(\ln \frac{\sigma_{1f}\sigma_{2f}}{\bar{\rho}} \right)^2}{\sum_f a_f^{\mathrm{3D}}} \right]^{1/2}. \end{aligned} \tag{28}$$

The natural logarithm treats reciprocal stretch ratios symmetrically, and normalization by $\bar{\rho}$ removes global scale from the area measure. Face-level contributions are aggregated into one value per candidate before the candidate-to-mesh and mesh-to-dataset averages. These continuous evaluation metrics complement the binary reward based on a Dirichlet energy threshold in Appendix B.

Topology, fragmentation, and visibility. All-disk rate is the fraction of candidate layouts for which every predicted chart satisfies the disk test in Eq. (23). Final UV islands are connected face components across edges that are continuous in both 3D and UV coordinates. Seam visibility is the seam-length-weighted exposed fraction defined by ν_g in Eq. (24); lower values indicate less exposed seams. UV utilization measures the occupied fraction of the atlas, excluding padding. For non-overlapping packed charts, this fraction equals the summed UV-triangle area divided by atlas area; the occupancy definition counts overlapping regions only once. Inference time covers seam generation and the subsequent parameterization and packing stages. A failed parameterization and a failed disk test describe different outcomes: a parameterizer may introduce additional cuts even when the predicted charts fail the disk criterion.

C.2 Cross-Objective Comparison Design

We evaluate edge-level allocation through paired Scalar / Dense comparisons under RAM, Diffusion-NFT, and AWM. Within each objective, the pair shares the backbone, initial checkpoint, mesh split, reward components, candidate protocol, and training settings. Scalar applies candidate-level feedback uniformly across edges, whereas Dense allocates feedback according to local UV quality. Both variants are evaluated after 500 optimizer steps and throughout the corresponding training schedule.

For RAM, the dense correction is $C_{ge} = A_g w_{ge}$, with the nonnegative, mean-one weights defined in Eq. (7). The coefficient enters the regression target in Eq. (9). Setting $\alpha = 0$ recovers scalar RAM, while the dense configuration uses $\alpha = 0.5$. The coefficient identities in Appendix A concern RAM; the cross-objective conclusions rest on the paired empirical comparisons.

At 500 steps, each dense variant achieves higher all-disk rate and UV utilization, fewer UV islands, and lower seam visibility than its scalar counterpart (Table 3). The learning curves show earlier attainment of comparable quality (Figure 5), supporting improved reward-guided exploration under the paired training schedules. The interpretation of these learning trajectories is discussed in Appendix C.3.

C.3 Evaluating Exploration Efficiency

The 500-step comparison evaluates final quality at a fixed optimization horizon, while the learning curves assess how quickly each variant reaches comparable quality under the paired training schedules. Together, they characterize reward-guided exploration in terms of progress over optimizer steps. In RAM, multiple analytic noising draws reuse the UV feedback from each evaluated endpoint. Optimizer steps and reward evaluations therefore represent distinct budget units; the learning curves do not by themselves quantify a reduction in reward calls.