Analytic Rotor-Based Canonicalisation in VAEs: A Controlled Study of Geometric Inductive Biases

Anonymous authors
Paper under double-blind review

Abstract

This paper asks, whether in pose-aware variational autoencoders where the pose of each training view is already known, it is better to apply that pose through an exact geometric transformation or to give pose coordinates to a decoder and ask the decoder to learn the rendering rule from data. We study this question in a deliberately simple two-dimensional silhouette setting. The encoder extracts object content from two rotated views, the decoder predicts one canonical silhouette, and the proposed model renders each posed view by an analytic rotor-induced image warp. The matched baseline uses the same encoder, latent size, and decoder width, but concatenates the pose code to the decoder input. Across three random seeds in a compressed-capacity setting, the rotor pathway improves validation canonical binary cross-entropy from 0.0919 ± 0.0022 to 0.0889 ± 0.0051, improves thresholded canonical Dice from 0.8339 ± 0.0042 to 0.8407 ± 0.0035, improves thresholded view Dice from 0.7981 ± 0.0119 to 0.8352 ± 0.0030, and reduces relative-pose composition error from 0.0319 ± 0.0022 to 0.0092 ± 0.0004. The baseline obtains lower probabilistic view cross-entropy, which we interpret cautiously because its smoother predictions can reduce cross-entropy while giving worse thresholded shape agreement. These results support the value of explicit analytic warping for this known-pose canonicalisation problem. They do not establish that planar Geometric Algebra is superior to all analytic matrix or spatial-transformer warps. Finally, we outline how the same design principle could be carried to three-dimensional rotors and Conformal Geometric Algebra motors, but we leave that extension as future empirical work.

1 Introduction

Learning systems for images and shapes often need to handle pose. A common solution is to provide pose as numerical coordinates, such as an angle, a pair $[\cos\theta, \sin\theta]$, a quaternion, or the entries of a rotation matrix, where $\theta \in \mathbb{R}$ denotes a planar angle. Those representations are not interchangeable with unstructured coordinates because rotation matrices, quaternions, and rotors all have composition rules. However, when such quantities are merely concatenated to a neural decoder, the decoder must still learn from data how the pose code should act on the output signal.

Geometric Algebra[1] offers one way to keep the action separate from the learned decoder. In Geometric Algebra, a rotor is a normalised even multivector. Its multiplication law represents composition, and its sandwich action represents rotation. In two dimensions this is algebraically equivalent to the usual complex phase or special orthogonal matrix representation, so the planar rotor is not introduced here as a more expressive parameterisation. Rather, it gives a compact setting in which we can test an architectural principle, i.e. decode canonical content once and apply known pose through a fixed analytic geometric action instead of a learned coordinate-conditioned rendering map.

[1]For a complete treatment of Geometric Algebra, we refer the reader to Hestenes & Sobczyk (1984); Doran & Lasenby (2003).

The main experimental difficulty is isolation, because a fully unsupervised pose-aware model would normally mix pose inference, content inference, decoder capacity, training regularisation, and the chosen pose representation. Consequently, it would be difficult to decide whether any improvement came from the geometric action itself. The specific gap addressed here is therefore the absence of a controlled known-pose VAE comparison between two generative design choices, analytic pose application and coordinate-conditioned learned rendering.

To fill this gap, we build a paired-view variational autoencoder for binary silhouettes. The synthetic generator supplies two viewing angles, denoted θ_a and θ_b, for two observed views of the same object. The encoder extracts content from the views, the decoder predicts a canonical logit field, and the proposed model renders the posed views by an exact rotor-induced warp. This design removes pose estimation as a confounder and focuses the comparison on the generative pose pathway.

The contribution is therefore deliberately modest and precisely scoped. We do not claim to solve unsupervised pose learning, and we do not claim that a two-dimensional rotor warp is empirically superior to an equivalent analytic matrix-based spatial transformer. Instead, we show that, in this known-pose silhouette task, replacing coordinate-conditioned learned rendering with an explicit analytic action improves thresholded shape fidelity and strongly improves the relative-pose consistency that the model is designed to preserve.

The two-dimensional setting is also useful as a mathematically transparent stepping stone. In three dimensions, machine learning practice has several useful rotation parameterisations, including quaternions and continuous five- or six-dimensional representations (Zhou et al., 2019). Geometric Algebra is not the only viable language for three-dimensional rotations. Its advantage for the future direction considered here is that spatial rotors and Conformal Geometric Algebra motors represent rotations and rigid motions with a single compositional sandwich action. The present paper does not test that three-dimensional claim; it only motivates a natural next experiment.

Finally, the complete Python source code for all models and geometric verifications is provided in the supplementary material to support reproducibility.

2 Related Work

Variational autoencoders provide the probabilistic modelling framework used here (Kingma & Welling, 2014). A long line of work studies how latent variable models separate content from nuisance transformations such as translation, rotation, and scale. For example, β-VAE encourages statistical separation through regularised latent channels, although such unsupervised pressure alone may not isolate geometric factors reliably (Higgins et al., 2017). Spatial-VAE introduces explicit structure for planar translation and rotation (Bepler et al., 2019), TARGET-VAE adds equivariant inference architectures for semantic representations under large pose variation (Nasiri & Bepler, 2022) and EQ-VAE regularises latent spaces so that semantic transformations act more predictably (Kouzelis et al., 2025). Together, these works show that the geometry of the latent representation matters, but they do not isolate the known-pose rendering pathway studied in this paper.

The closest historical example is the family of transformational autoencoders, beginning with Transforming Auto-Encoders which learn from transformed pairs and encode transformation-aware structure (Hinton et al., 2011). Spatial Transformer Networks introduced differentiable image warping as a learnable module (Jaderberg et al., 2015). VITAE then combined variational autoencoders with spatial transformers to disentangle appearance and perspective in generative models (Detlefsen & Hauberg, 2019). Our work is adjacent to these papers, but it focuses on a narrower known-pose comparison in which the proposed decoder does not infer an unconstrained transformation, because the supplied angle is routed through a fixed analytic rotor warp.

Group-equivariant generative models address a broader goal. Holographic-VAE, for instance, formulates rotationally equivariant neural fields for three-dimensional shape generation under SO(3) (Visani et al., 2024). By contrast, our study does not aim to recover pose from data or build a fully equivariant generator. It asks whether, once pose is known, generation is better structured when pose is applied as an exact geometric action rather than consumed as ordinary decoder input.

The work also relates to Geometric Algebra neural networks. Geometric Clifford Algebra Networks introduce multivector group-action layers for geometry-guided optimisation (Ruhe et al., 2023b) and Clifford Group Equivariant Neural Networks develop equivariant architectures whose actions extend from vectors to Clifford algebras (Ruhe et al., 2023a). Our model is simpler as it is not a fully multivector-valued neural network. It embeds one specific Geometric Algebra object, the planar rotor, into the generative pathway of a VAE and tests the resulting inductive bias against a matched coordinate-conditioned decoder.

Finally, there is substantial related work on three-dimensional generative modelling with explicit geometry. For example, NeRF-VAE combines a VAE with neural radiance fields and differentiable volume rendering for geometry-aware scene generation (Kosiorek et al., 2021), while EG3D conditions three-dimensional generative rendering on camera information (Chan et al., 2022). These papers support the broader idea that explicit geometric rendering can be valuable, but they do not test the two-dimensional rotor-versus-coordinate pathway isolated here.

Although related ideas therefore appear in transformational autoencoders, spatial-transformer-based generative models, equivariant VAEs, and three-dimensional neural rendering, the present evidence should be read as supporting explicit analytic warping over coordinate-conditioned decoding in this controlled known-pose setting. It should not be read as establishing superiority specific to Geometric Algebra over analytic matrix or spatial-transformer schemes outside Geometric Algebra.

3 Geometric Setup and Motivation

This section defines the planar Geometric Algebra used by the model. All multiplication rules in this section are fixed mathematical operations, not quantities discovered by the neural network. In the implementation, these rules appear as a deterministic warp layer placed after the canonical decoder.

3.1 Planar Geometric Algebra and the Rotor Action

Let $\mathrm{Cl}(2,0)$ denote the real Clifford algebra generated by orthonormal basis vectors e_1 and e_2. These basis vectors satisfy:

$$e_1^2 = 1 \qquad e_2^2 = 1 \qquad e_1 e_2 = -e_2 e_1 \tag{1}$$

Define the unit bivector $\mathbf{I}$ by:

$$\mathbf{I} \coloneqq e_1 e_2 \tag{2}$$

Appendix A shows step by step that:

$$\mathbf{I}^2 = -1 \tag{3}$$

For an angle $\phi \in \mathbb{R}$, define the planar rotor by:

$$r(\phi) \coloneqq \cos\left(\frac{\phi}{2}\right) - \sin\left(\frac{\phi}{2}\right)\mathbf{I} \tag{4}$$

We write the reverse of a multivector A as $A^\dagger$ rather than using a long tilde. For the planar rotor, the reverse is:

$$r(\phi)^\dagger = \cos\left(\frac{\phi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\mathbf{I} \tag{5}$$

The rotor has unit norm in the sense that:

$$r(\phi)\; r(\phi)^\dagger = 1 \tag{6}$$

Let $u \in \mathbb{R}^2$ be the vector:

$$u = u_1 e_1 + u_2 e_2 \tag{7}$$

where $u_1, u_2 \in \mathbb{R}$ are scalar coordinates. The active rotor action is the sandwich product:

$$u' = r(\phi)\; u\, r(\phi)^\dagger \tag{8}$$

Appendix A expands every multiplication. The result is:

$$u' = (\cos\phi\, u_1 - \sin\phi\, u_2)e_1 + (\sin\phi\, u_1 + \cos\phi\, u_2)e_2 \tag{9}$$

Equation (9) is the usual active planar rotation, obtained here from the rotor product rather than inserted as an external matrix formula.

A second important fact is composition. If $\phi_1, \phi_2 \in \mathbb{R}$, then:

$$r(\phi_2)\; r(\phi_1) = r(\phi_1 + \phi_2) \tag{10}$$

This additive form is special to planar rotations, or more generally to rotations in a common plane generated by commuting bivectors. In three dimensions, rotor multiplication still gives exact composition, but the composed rotor is not generally obtained by simply adding scalar rotation angles unless the rotations are coplanar.

3.2 From Euclidean Rotation to Image-space Warping

The decoder outputs a canonical image on a raster grid, not a coordinate vector. Let $D = [-1, 1]^2$ denote the normalised image domain, and let $\ell_{\text{can}} : D \to \mathbb{R}$ denote a canonical logit field. Because inverse sampling can query points outside D, we use the extended field:

$$\ell_{\text{can}}^{\text{ext}}(u) = \begin{cases} \ell_{\text{can}}(u), & u \in D \\ \ell_{\text{pad}}, & u \notin D \end{cases} \tag{11}$$

where $\ell_{\text{pad}} \in \mathbb{R}$ is the padding logit assigned to source coordinates outside the normalised square domain D.

Let $R(\phi) \in \mathrm{SO}(2)$ denote the standard Euclidean rotation matrix. The continuous inverse-sampling warp is:

$$\big(\mathcal{G}_\phi \ell_{\text{can}}\big)(u) \coloneqq \ell_{\text{can}}^{\text{ext}}(R(-\phi)u) \qquad u \in D \tag{12}$$

The minus sign appears because each output coordinate asks which source coordinate should be sampled. Raster images also use a downward-pointing vertical axis, whereas the algebra above uses an upward-pointing Euclidean vertical axis. Appendix B derives the sign change and shows that a positive image-space rotation by θ is implemented by the Euclidean rotor associated with $-\theta$. Therefore, the image-space rotor is:

$$r_{\text{img}}(\theta) = r(-\theta) \tag{13}$$

On the discrete grid, Equation (12) is implemented by bilinear sampling of logits. The continuous transformation is exactly compositional, and the discrete implementation inherits the same structure up to interpolation and padding effects.

4 Proposed Model

The previous section introduced the fixed geometry. We now place that geometry inside a VAE-style latent variable model.

4.1 Controlled Problem Statement

Let H and W denote the image height and width. Let $c \in \{0, 1\}^{H \times W}$ denote a canonical binary silhouette of one object. Let $\theta_a, \theta_b \in [-\pi, \pi)$ denote two known viewing angles. The observed binary images are generated as:

$$x_a = \mathbf{1}[\mathcal{G}_{\theta_a} c > 0.5] \qquad x_b = \mathbf{1}[\mathcal{G}_{\theta_b} c > 0.5] \tag{14}$$

where the threshold is applied pixelwise after bilinear raster rotation during dataset generation. Thus, the stored targets x_a and x_b are binary, while the differentiable model warp used during training samples logits bilinearly. Each training example contains:

$$(c, x_a, x_b, \theta_a, \theta_b) \tag{15}$$

The learning problem is not to infer the angles from scratch. Those quantities are supplied by the synthetic generator. The learning problem is to infer object content from the two posed views, decode a canonical silhouette, and reconstruct each posed observation using the supplied pose.

4.2 Generative Model and Approximate Posterior

To make the VAE component explicit, we first describe the idealised conditional generative model. Let $z \in \mathbb{R}^d$ be a latent content vector with prior:

$$p(z) = \mathcal{N}(0, I_d) \tag{16}$$

where d is the latent dimension and I_d is the $d \times d$ identity matrix. Given z, the canonical decoder D_ψ with parameters ψ produces a canonical logit field:

$$\ell_{\text{can}} = D_\psi(z) \tag{17}$$

The corresponding posed logits are:

$$\ell_a = \mathcal{G}_{\theta_a} \ell_{\text{can}} \qquad \ell_b = \mathcal{G}_{\theta_b} \ell_{\text{can}} \tag{18}$$

Let Ω be the set of pixel indices and let $\text{sigm}(t) = (1 + e^{-t})^{-1}$ denote the sigmoid function. To keep canonical supervision outside the formal generative likelihood, the Bernoulli likelihood is defined only for the two observed views and factorises over pixels as:

$$\begin{aligned} p_\psi(x_a, x_b \mid z, \theta_a, \theta_b) = & \prod_{u \in \Omega} \text{Bernoulli}(x_a(u); \text{sigm}(\ell_a(u))) \\ & \times \prod_{u \in \Omega} \text{Bernoulli}(x_b(u); \text{sigm}(\ell_b(u))) \end{aligned} \tag{19}$$

Equivalently, the conditional joint model is:

$$p_\psi(z, x_a, x_b \mid \theta_a, \theta_b) = p(z)\, p_\psi(x_a, x_b \mid z, \theta_a, \theta_b) \tag{20}$$

The canonical target c is therefore treated as auxiliary supervised information, not as a random variable in the formal likelihood above.

The implemented inference model uses the same encoder E_ω, with parameters ω, on each view. For view x_v, where $v \in \{a, b\}$, the encoder predicts:

$$(\mu_v, \log \sigma_v^2) = E_\omega(x_v) \tag{21}$$

where $\mu_v \in \mathbb{R}^d$, $\sigma_v^2 \in \mathbb{R}_+^d$, and the logarithm is applied componentwise. This defines the diagonal Gaussian:

$$q_\omega(z \mid x_v) = \mathcal{N}\big(\mu_v, \text{diag}(\sigma_{v,1}^2, \ldots, \sigma_{v,d}^2)\big) \tag{22}$$

The code samples independent reparameterised latent variables:

$$z_a = \mu_a + \sigma_a \odot \varepsilon_a \qquad z_b = \mu_b + \sigma_b \odot \varepsilon_b \qquad \varepsilon_a, \varepsilon_b \sim \mathcal{N}(0, I_d) \tag{23}$$

where $\odot$ denotes elementwise multiplication. The fused content sample and fused mean are:

$$z_s = \frac{1}{2}(z_a + z_b) \qquad \mu_s = \frac{1}{2}(\mu_a + \mu_b) \tag{24}$$

Because z_a and z_b are independent Gaussian samples, the induced fused posterior is also Gaussian:

$$q_s(z \mid x_a, x_b) = \mathcal{N}(\mu_s, \Sigma_s) \qquad \Sigma_s = \frac{1}{4}(\Sigma_a + \Sigma_b) \tag{25}$$

where $\Sigma_v = \text{diag}(\sigma_{v,1}^2, \ldots, \sigma_{v,d}^2)$ for $v \in \{a, b\}$. Appendix D derives Equation (25) line by line.

The objective used below is therefore a supervised VAE-style objective rather than an evidence lower bound derived from a single fused posterior $q_s(z \mid x_a, x_b)$ with a corresponding $\text{KL}(q_s(z \mid x_a, x_b) \| p(z))$ term, where the observed-view Bernoulli reconstruction losses and per-view Kullback-Leibler penalties provide the VAE-style component, whereas the canonical BCE and Dice losses are deliberately kept outside the likelihood in Equation (19) as supervised canonicalisation losses and the invariance and confidence penalties act as additional regularisers that make the controlled canonicalisation problem well-posed and improve binary shape fidelity.

4.3 Canonical Decoder and Analytic Rendering

The decoder maps the fused content code to two versions of the canonical logit field:

$$\ell_{\text{can}} = D_\psi(z_s) \qquad \bar{\ell}_{\text{can}} = D_\psi(\mu_s) \tag{26}$$

The sampled field ℓ_{can} is used in the stochastic reconstruction path. The mean-field version $\bar{\ell}_{\text{can}}$ is used for canonical supervision and stable evaluation metrics.

The posed reconstructions are obtained analytically as:

$$\hat{\ell}_a = \mathcal{G}_{\theta_a}\, \ell_{\text{can}} \qquad \hat{\ell}_b = \mathcal{G}_{\theta_b}\, \ell_{\text{can}} \tag{27}$$

and the deterministic mean-field posed logits are:

$$\bar{\hat{\ell}}_a = \mathcal{G}_{\theta_a}\, \bar{\ell}_{\text{can}} \qquad \bar{\hat{\ell}}_b = \mathcal{G}_{\theta_b}\, \bar{\ell}_{\text{can}} \tag{28}$$

Equations (27) and (28) mean that the decoder does not learn a separate function of content and pose. It learns only a canonical field. Pose enters afterward through a non-learned differentiable sampling layer whose affine grid is computed from the rotor. Thus, the phrase 'the network does not learn how pose acts' means that the action of pose on image coordinates is fixed by the algebra and by the inverse-sampling convention; only the canonical silhouette is learned.

Proposition 1 (Exact relative pose composition at the continuous level)**.** *Let $\theta_a, \theta_b \in \mathbb{R}$, and let $\ell : D \to \mathbb{R}$ be a canonical logit field with the extension and warp conventions in Equations* (11)-(12)*. Ignoring boundary effects from padding, the continuous warp satisfies:*

$$\mathcal{G}_{\theta_b - \theta_a}\big(\mathcal{G}_{\theta_a}\ell\big) = \mathcal{G}_{\theta_b}\ell \tag{29}$$

Proof. For any $u \in D$, use the definition of the inverse-sampling warp:

$$\begin{aligned}
\big(\mathcal{G}_{\theta_b-\theta_a}(\mathcal{G}_{\theta_a}\ell)\big)(u) &= (\mathcal{G}_{\theta_a}\ell)\big(R(-(\theta_b-\theta_a))u\big) && (30)\\
&= (\mathcal{G}_{\theta_a}\ell)\big(R(-\theta_b+\theta_a)u\big) && (31)\\
&= \ell\big(R(-\theta_a)R(-\theta_b+\theta_a)u\big) && (32)\\
&= \ell\big(R(-\theta_a-\theta_b+\theta_a)u\big) && (33)\\
&= \ell\big(R(-\theta_b)u\big) && (34)\\
&= (\mathcal{G}_{\theta_b}\ell)(u) && (35)
\end{aligned}$$

The fourth line uses the planar rotation identity $R(\alpha)R(\beta) = R(\alpha+\beta)$. On a raster grid the same identity is perturbed by interpolation and padding, but the continuous composition law remains the intended structure. □

Proposition 1 motivates the composition metric used in the experiments. For the rotor model this metric partly validates a property built into the pathway, so it should not be interpreted as a fully independent benchmark score. It is instead a direct check that the intended inductive bias survives discretisation and optimisation.

4.4 Loss Function and Baseline Objective

This subsection defines the training objective. Let $\text{BCE}(\ell, y)$ denote the mean binary cross-entropy with logits between a logit image ℓ and a binary target image y. It is defined in Appendix D. The observed-view binary cross-entropy is:

$$\mathcal{L}_{\text{view-BCE}} = \text{BCE}(\hat{\ell}_a, x_a) + \text{BCE}(\hat{\ell}_b, x_b) \tag{36}$$

The observed-view soft Dice loss is:

$$\mathcal{L}_{\text{view-Dice}} = \big(1 - \text{Dice}_{\text{soft}}(\hat{\ell}_a, x_a)\big) + \big(1 - \text{Dice}_{\text{soft}}(\hat{\ell}_b, x_b)\big) \tag{37}$$

For logits ℓ and binary target y, the soft Dice coefficient is:

$$\mathrm{Dice}_{\mathrm{soft}}(\ell, y) = \frac{2\sum_{u\in\Omega} \mathrm{sigm}(\ell(u))y(u) + \eta}{\sum_{u\in\Omega} \mathrm{sigm}(\ell(u)) + \sum_{u\in\Omega} y(u) + \eta} \tag{38}$$

where $\eta > 0$ is a small numerical constant that prevents division by zero.

The canonical supervision terms are:

$$\mathcal{L}_{\text{can-BCE}} = \mathrm{BCE}(\bar{\ell}_{\mathrm{can}}, c) \qquad \mathcal{L}_{\text{can-Dice}} = 1 - \mathrm{Dice}_{\mathrm{soft}}(\bar{\ell}_{\mathrm{can}}, c) \tag{39}$$

The latent invariance penalty is:

$$\mathcal{L}_{\mathrm{inv}} = \|\mu_a - \mu_b\|_2^2 \tag{40}$$

The VAE regulariser uses the Kullback-Leibler divergence, denoted $\mathrm{KL}(q\|p)$, from an approximate posterior q to a prior p:

$$\mathcal{L}_{\mathrm{KL}} = \mathrm{KL}(q_\omega(z \mid x_a) \,\|\, \mathcal{N}(0, I_d)) + \mathrm{KL}(q_\omega(z \mid x_b) \,\|\, \mathcal{N}(0, I_d)) \tag{41}$$

Appendix D expands this quantity component by component for diagonal Gaussian posteriors.

Finally, the confidence penalty discourages uncertain Bernoulli probabilities. For any logit image ℓ, define:

$$\mathcal{C}(\ell) = \frac{1}{|\Omega|} \sum_{u\in\Omega} \mathrm{sigm}(\ell(u))\big(1 - \mathrm{sigm}(\ell(u))\big) \tag{42}$$

The confidence loss is:

$$\mathcal{L}_{\mathrm{conf}} = \mathcal{C}(\bar{\ell}_{\mathrm{can}}) + \mathcal{C}(\bar{\hat{\ell}}_a) + \mathcal{C}(\bar{\hat{\ell}}_b) \tag{43}$$

This term does not create geometric structure by itself. It sharpens occupancy predictions after the reconstruction losses have determined where the object should lie.

The full objective is:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{\text{view-BCE}} + \lambda_{\text{view-Dice}}\,\mathcal{L}_{\text{view-Dice}} + \lambda_{\text{can-BCE}}\,\mathcal{L}_{\text{can-BCE}} \\ + \lambda_{\text{can-Dice}}\,\mathcal{L}_{\text{can-Dice}} + \lambda_{\mathrm{inv}}\,\mathcal{L}_{\mathrm{inv}} + \beta_z\,\mathcal{L}_{\mathrm{KL}} + \lambda_{\mathrm{conf}}\,\mathcal{L}_{\mathrm{conf}}\end{aligned} \tag{44}$$

The quantitative experiments use:

$$\begin{aligned}\lambda_{\text{can-BCE}} = 5.0 \quad \lambda_{\text{can-Dice}} = 2.0 \qquad \lambda_{\text{view-Dice}} = 1.0 \\ \lambda_{\mathrm{inv}} = 5.0 \qquad \beta_z = 10^{-4} \qquad \lambda_{\mathrm{conf}} = 0.02\end{aligned} \tag{45}$$

These weights were chosen empirically in preliminary pilot runs and then held fixed for all reported comparisons. We place larger weights on canonical reconstruction and latent invariance because the experiment is designed around canonicalisation and content agreement. The KL and confidence terms are auxiliary regularisers, so they are kept small to avoid dominating the reconstruction objective.

The coordinate-conditioned baseline uses the same objective in Equation (44). Its decoder input is the concatenation of the fused content code and a pose code $p(\theta) = [\cos\theta, \sin\theta]$. For $v \in \{a, b\}$, its posed logits are defined explicitly as:

$$\hat{\ell}_v^{\mathrm{base}} = D_\psi([z_s, p(\theta_v)]) \qquad v \in \{a, b\} \tag{46}$$

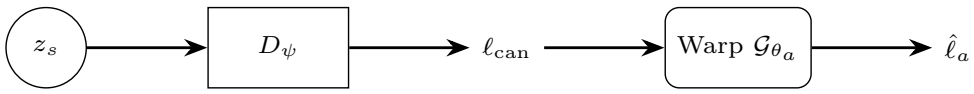
and its canonical mean-field logits are:

$$\bar{\ell}_{\mathrm{can}}^{\mathrm{base}} = D_\psi([\mu_s, p(0)]) \qquad p(0) = [1, 0] \tag{47}$$

The baseline loss terms are evaluated by replacing $\hat{\ell}_a$, $\hat{\ell}_b$ and $\bar{\ell}_{\mathrm{can}}$ in the definitions above with $\hat{\ell}_a^{\mathrm{base}}$, $\hat{\ell}_b^{\mathrm{base}}$ and $\bar{\ell}_{\mathrm{can}}^{\mathrm{base}}$. Therefore, canonical supervision is applied to both models during training, not only during evaluation.

Two modelling choices are structural. First, canonical losses are necessary because observed views alone do not identify an absolute canonical frame. A common angular offset applied to the canonical prediction and subtracted from all view angles leaves the posed reconstructions unchanged. Second, asymmetric silhouettes are necessary because symmetric shapes do not have a unique absolute orientation. Appendix C proves both statements.

A) Proposed Geometric Algebra VAE



B) Coordinate-conditioned learned rendering baseline

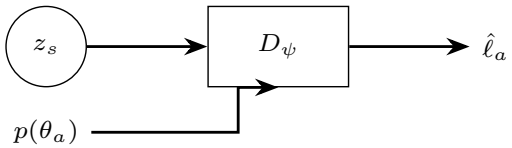


Figure 1: Comparison of the generative pathways. The encoders are identical and omitted for clarity. **(A)** The proposed model decodes the fused content code z_s into a canonical logit field ℓ_{can} and then rotates it with the analytic warp $\mathcal{G}_{\theta_a}$. **(B)** The baseline concatenates z_s with the coordinate pose vector $p(\theta_a) = [\cos\theta_a, \sin\theta_a]$ and learns the posed rendering map from data.

4.5 Coordinate-conditioned Learned Rendering Baseline

The baseline is designed to isolate the architectural difference in the generative pathway. Both models use the same encoder architecture, latent dimension, and transposed-convolution decoder width. The proposed model decodes a canonical shape and applies pose analytically. The baseline receives the coordinate pose code:

$$p(\theta) = [\cos\theta, \sin\theta] \in \mathbb{R}^2 \tag{48}$$

concatenated to the fused content code at the decoder input. It must therefore learn the posed rendering map from data. Because this concatenation gives the baseline slightly more trainable parameters, any advantage of the proposed model cannot be attributed to greater decoder capacity.

This comparison intentionally does not test a rotor warp against an equivalent matrix-based spatial transformer. In two dimensions those continuous warps are mathematically identical. The comparison instead tests explicit analytic warping against coordinate-conditioned learned rendering.

5 Experimental Protocol

The theory above motivates the model. This section describes the synthetic setup used to test it. All experiments were implemented in Python using PyTorch. The geometric sanity checks were independently verified with the Python `clifford` package for Geometric Algebra (Hadfield et al.). Full software details appear in Appendix F.

5.1 Dataset Generation

Each example begins with a canonical binary silhouette of size 32×32. The canonical silhouettes are drawn from six asymmetric prototype polygons. Each polygon is rasterised at $8\times$ higher resolution, downsampled bilinearly, and thresholded back to binary occupancy. The global scale factor is $0.22 \times H \times 8$, where $H = 32$ is the image size. The exact vertex coordinates of all six templates are listed in Appendix E. Two angles θ_a and θ_b are drawn independently and uniformly from $[-\pi, \pi)$, and the corresponding posed views are obtained by the same image-space warp convention used by the model.

The dataset is intentionally simple. By removing texture, clutter, and pose-estimation uncertainty, it makes the effect of the generative pose pathway directly measurable.

5.2 Architectures and Optimisation

We report two settings. The first is a compressed multi-seed setting designed to expose inductive-bias differences under limited capacity. The second is a larger single-seed rotor-only setting used to show that the proposed pathway can produce sharp binary silhouettes when capacity is increased.

In the compressed setting, both the rotor model and the coordinate-conditioned baseline use four stride-2 convolutional encoder layers, latent dimension 16, and decoder base width 8. The training set contains 256 paired examples, the validation set contains 128 paired examples, the optimiser is Adam with learning rate 2×10^{-3}, the batch size is 32, and training runs for 12 epochs. We report the mean and standard deviation over seeds $\{7, 11, 17\}$.

In the larger illustrative reconstruction setting, only the rotor model is widened, to latent dimension 64 and base width 32. By 'illustrative reconstruction setting', we mean a single-seed higher-capacity run used only to visualise thresholded reconstructions. It is not part of the matched ablation claim. The validation set contains 64 examples and training runs for 16 epochs with the same learning rate and batch size. The exact layer-by-layer architecture for both settings appears in Appendix E.

5.3 Metrics

We evaluate canonical reconstruction with validation canonical binary cross-entropy and validation thresholded canonical Dice. The thresholded Dice uses the binary prediction:

$$\mathbf{1}[\mathrm{sigm}(\ell(u)) > 0.5] \tag{49}$$

for each pixel $u \in \Omega$. We also report thresholded view Dice on the observed posed views.

The structural metric is composition consistency. For a validation example with views x_a and x_b, let $\bar{\hat{\ell}}_a$ and $\bar{\hat{\ell}}_b$ denote the deterministic predicted logits from Equation (28). Let $\Delta = \theta_b - \theta_a$ be the relative angle. We define:

$$\mathcal{L}_{\mathrm{comp}} = \mathrm{MSE}\Big(\mathrm{sigm}\big(\mathcal{G}_\Delta \bar{\hat{\ell}}_a\big),\ \mathrm{sigm}\big(\bar{\hat{\ell}}_b\big)\Big) \tag{50}$$

Equation (50) asks whether the predicted view at angle θ_a, when analytically rotated by the known relative angle, agrees with the predicted view at angle θ_b. For the rotor model, the continuous version is expected to be small by Proposition 1. Non-zero values reflect raster interpolation, padding, and optimisation error. For the baseline, no exact group law is enforced.

6 Results

This section presents the empirical evidence in the same order as the modelling claims.

6.1 Geometric Validation and Illustrative Reconstructions

Figure 2 visualises the analytic action of the image-space rotor warp on a canonical silhouette. No neural network is involved. The figure is a geometric sanity check confirming that the implemented warp follows the expected raster sign convention before it is applied to decoder outputs.

Figure 3 shows the rotor model in the larger illustrative reconstruction configuration. The top block compares true canonical silhouettes with thresholded canonical predictions, while the lower rows compare ground-truth posed views with thresholded reconstructions. The silhouettes are sharp because the model predicts Bernoulli occupancies and the confidence penalty discourages logits near the ambiguous probability 0.5. In this configuration, the validation canonical binary cross-entropy is 0.0380 and the validation canonical Dice is 0.9080. These values describe a single illustrative configuration; the quantitative claim comes from the compressed multi-seed ablation below.

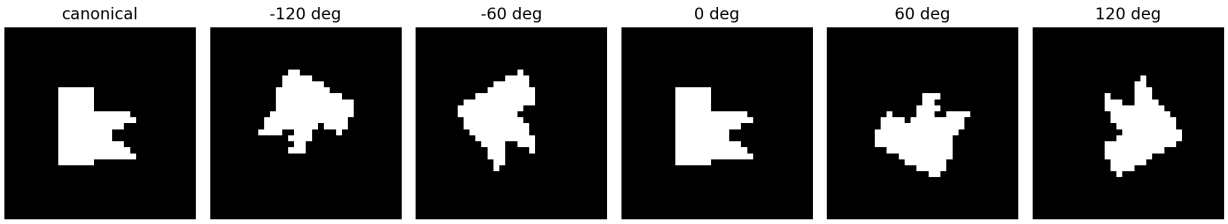


Figure 2: Analytic rotor action on a canonical silhouette. The figure is produced by the exact image-space warp induced by the planar rotor with raster sign convention $r_{\mathrm{img}}(\theta) = r(-\theta)$.

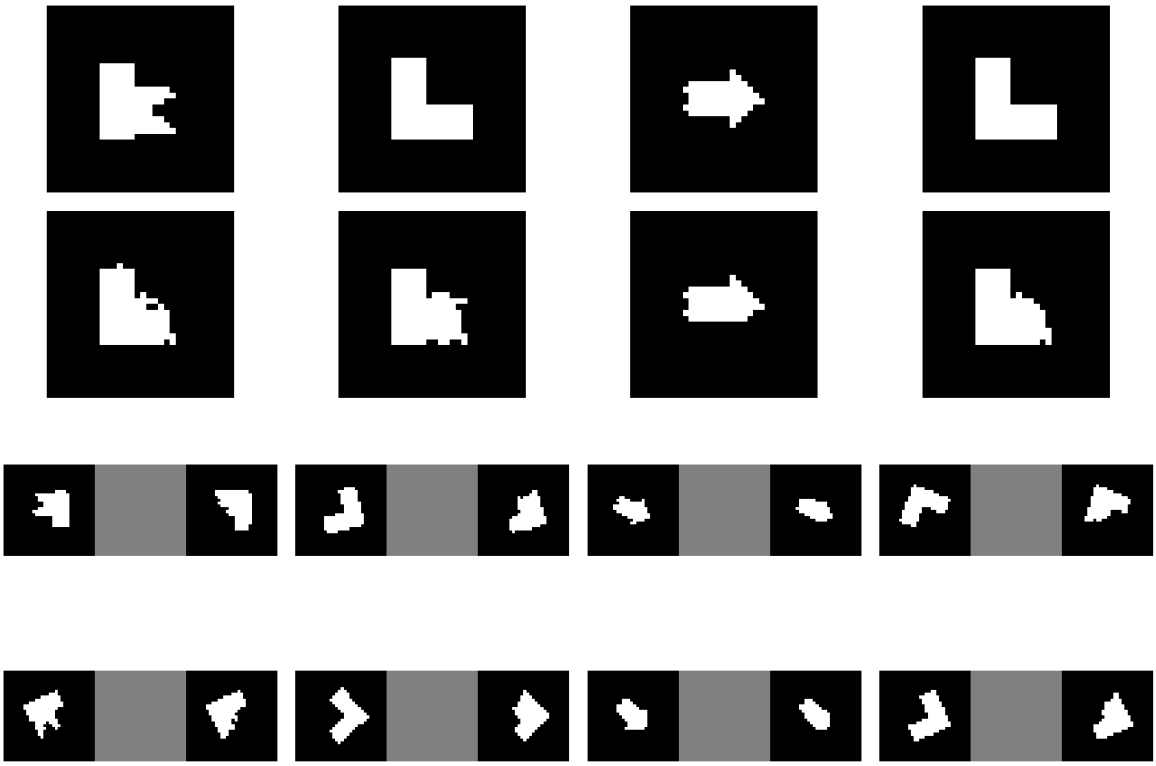


Figure 3: Thresholded illustrative reconstructions from the larger rotor-only configuration. The upper block compares ground-truth canonical silhouettes with predicted canonical silhouettes. The lower blocks compare observed views with thresholded reconstructions at two poses for each object.

6.2 Ablation Against the Matched Coordinate-Conditioned Baseline

Table 1 and Figure 4 report the compressed multi-seed experiment. The two models see the same data and use the same encoder and decoder widths. The only deliberate difference is whether pose is applied by the analytic rotor warp or supplied as coordinates to a learned decoder.

Figure 5 gives a compact view of the same final means reported in Table 1. Each metric is oriented so that a positive percentage means the rotor model is better and a negative percentage means the coordinate-conditioned baseline is better. The figure is therefore an interpretive summary of the reported ablation, not a separate experiment.

Metric	GA rotor	Coord.-conditioned learned rendering baseline
Validation canonical BCE ↓	$\mathbf{0.0889 \pm 0.0051}$	0.0919 ± 0.0022
Thresholded canonical Dice ↑	$\mathbf{0.8407 \pm 0.0035}$	0.8339 ± 0.0042
Thresholded view Dice ↑	$\mathbf{0.8352 \pm 0.0030}$	0.7981 ± 0.0119
Validation view BCE ↓	0.3221 ± 0.0099	$\mathbf{0.2744 \pm 0.0094}$
Composition consistency MSE ↓	$\mathbf{0.0092 \pm 0.0004}$	0.0319 ± 0.0022

Table 1: Compressed multi-seed ablation over seeds $\{7, 11, 17\}$. Means and standard deviations are reported. Bold marks the better value for each metric. The rotor model improves canonical fidelity, thresholded view Dice, and relative-pose consistency, while the coordinate-conditioned baseline obtains lower probabilistic view BCE.

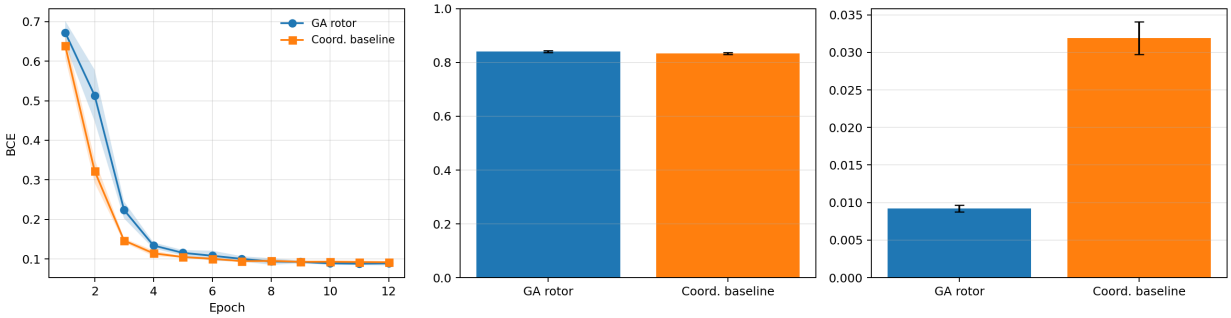


Figure 4: Quantitative ablation in the compressed setting. Left: validation canonical BCE over epochs averaged over three seeds. Middle: final thresholded canonical Dice. Right: final composition consistency MSE from Equation (50).

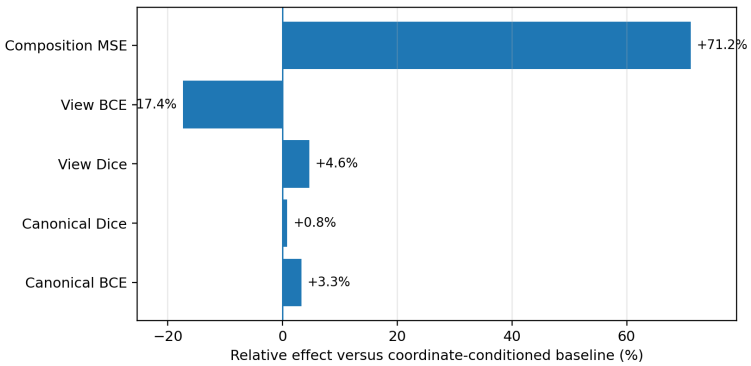


Figure 5: Relative effect summary for the compressed multi-seed ablation. The rotor model has the largest advantage on composition consistency and also improves thresholded shape agreement, while the coordinate-conditioned baseline retains the lower probabilistic view BCE. Positive values favour the rotor model after orienting each metric according to whether lower or higher is better.

The results support a restrained claim. The rotor pathway improves canonical reconstruction modestly and improves thresholded view Dice more clearly. The largest gap appears in composition consistency, where the error is reduced by more than a factor of three. Because this metric is closely aligned with the built-in analytic action, it should be interpreted as validation of the intended inductive bias rather than as an independent proof of broad performance superiority.

The view binary cross-entropy requires context. The coordinate-conditioned baseline obtains a lower probabilistic view BCE. This does not contradict the thresholded Dice result, because a learned decoder can

reduce cross-entropy by producing smoother boundary probabilities, while a rigidly warped sharp canonical shape can be penalised heavily for small raster misalignments. Thus, Table 1 and Figure 5 show a trade-off. The baseline is better on probabilistic view BCE, whereas the rotor model is better on thresholded shape agreement and relative-pose structure.

7 Limitations

This study supplies ground-truth angles, so it does not solve pose estimation. A practical end-to-end system would need a pose-estimation module or an equivariant inference mechanism that infers pose and content jointly. The domain is also intentionally simple: binary silhouettes, planar rotation, and canonical supervision. These choices make the mathematical comparison clean, but they prevent broad claims about natural images or general three-dimensional scenes.

The experiment also does not isolate Geometric Algebra as a parameterisation against all other analytic pose mechanisms. In two dimensions, the rotor warp is equivalent to an SO(2) matrix warp and to an appropriately restricted spatial transformer. The empirical claim is therefore architectural, namely that applying a known transformation analytically improves this controlled model relative to coordinate-conditioned learned rendering. It is not a claim that planar rotors are empirically superior to all matrix-based analytic warps.

The larger illustrative reconstruction result is single-seed rather than multi-seed averaged. The primary empirical evidence is the compressed multi-seed ablation. Finally, the three-dimensional discussion is a forward-looking sketch. It identifies a natural extension through spatial rotors and Conformal Geometric Algebra motors, but it does not provide three-dimensional experiments (Appendix G).

8 Conclusion

This paper presented a known-pose two-dimensional VAE in which canonical content is decoded once and pose is applied through an explicit rotor-induced image warp. The study compared this analytic pathway with a matched coordinate-conditioned decoder that receives the same pose information but must learn the rendering law. In the controlled silhouette setting, the rotor pathway gives modestly better canonical reconstruction, better thresholded shape agreement, and substantially lower relative-pose composition error, while the coordinate-conditioned baseline obtains lower probabilistic view BCE.

The main lesson is not that planar rotors uniquely solve pose representation. Rather, the evidence suggests that latent generative models can benefit when known transformations are routed through exact analytic actions instead of being absorbed into a learned decoder. A natural next step is to test the same design principle in three dimensions, using spatial rotors for pure rotation or motors in Conformal Geometric Algebra for rigid motion, while retaining the separation between canonical decoding and analytic transformation.

References

Tristan Bepler, Ellen Zhong, Kotaro Kelley, Edward Brignole, and Bonnie Berger. Explicitly disentangling image content from translation and rotation with spatial-VAE. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J. Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16123–16133, 2022.

Nicki Skafte Detlefsen and Søren Hauberg. Explicit disentanglement of appearance and perspective in generative models. In *Advances in Neural Information Processing Systems 32*, 2019.

Chris Doran and Anthony Lasenby. *Geometric algebra for physicists*. Cambridge University Press, 2003.

Hugo Hadfield, Eric Wieser, Alex Arsenovic, Robert Kern, and The Pygae Team. pygae/clifford. URL https://doi.org/10.5281/zenodo.1453978.

David Hestenes and Garret Sobczyk. *Clifford algebra to geometric calculus: a unified language for mathematics and physics*. Springer Science & Business Media, 1984.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International conference on artificial neural networks*, pp. 44–51. Springer, 2011.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014.

Adam R. Kosiorek, Heiko Strathmann, Daniel Zoran, Pol Moreno, Rosalia Schneider, Sona Mokra, and Danilo Jimenez Rezende. NeRF-VAE: A Geometry-Aware 3D Scene Generative Model. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5742–5752. PMLR, 2021.

Theodoros Kouzelis, Ioannis Kakogeorgiou, Spyros Gidaris, and Nikos Komodakis. Eq-vae: equivariance regularized latent space for improved generative image modeling (2025). *URL https://arxiv. org/abs/2502.09509*, 2025.

Alireza Nasiri and Tristan Bepler. Unsupervised object representation learning using translation and rotation group equivariant vae. *Advances in Neural Information Processing Systems*, 35:15255–15267, 2022.

David Ruhe, Johannes Brandstetter, and Patrick Forré. Clifford group equivariant neural networks. *Advances in neural information processing systems*, 36:62922–62990, 2023a.

David Ruhe, Jayesh K Gupta, Steven De Keninck, Max Welling, and Johannes Brandstetter. Geometric clifford algebra networks. In *International conference on machine learning*, pp. 29306–29337. PMLR, 2023b.

Gian Marco Visani, Michael N Pun, Arman Angaji, and Armita Nourmohammad. Holographic-(v) ae: An end-to-end so (3)-equivariant (variational) autoencoder in fourier space. *Physical review research*, 6(2): 023006, 2024.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5745–5753, 2019.

A Step-by-step Planar Geometric Algebra Calculations

The main text uses the planar rotor formulas in concise form. This appendix writes out the intermediate algebraic steps explicitly. The reverse operation is denoted by the dagger superscript $A^\dagger$.

A.1 Computing $\mathbf{I}^2$

Recall that $\mathbf{I} = e_1 e_2$. Then:

$$\begin{aligned}
\mathbf{I}^2 &= (e_1 e_2)(e_1 e_2) && (51)\\
&= e_1 e_2 e_1 e_2 && (52)\\
&= e_1 (e_2 e_1) e_2 && (53)\\
&= e_1 (-e_1 e_2) e_2 && (54)\\
&= -e_1 e_1 e_2 e_2 && (55)\\
&= -(e_1^2)(e_2^2) && (56)\\
&= -(1)(1) && (57)\\
&= -1 && (58)
\end{aligned}$$

A.2 Rotor Reverse and Unit Norm

Let:

$$c_\phi \coloneqq \cos\left(\frac{\phi}{2}\right) \qquad s_\phi \coloneqq \sin\left(\frac{\phi}{2}\right) \tag{59}$$

The rotor and its reverse are:

$$r(\phi) = c_\phi - s_\phi \mathbf{I} \qquad r(\phi)^\dagger = c_\phi + s_\phi \mathbf{I} \tag{60}$$

Their product is:

$$\begin{aligned}
r(\phi)\, r(\phi)^\dagger &= (c_\phi - s_\phi \mathbf{I})(c_\phi + s_\phi \mathbf{I}) && (61)\\
&= c_\phi c_\phi + c_\phi s_\phi \mathbf{I} - s_\phi \mathbf{I} c_\phi - s_\phi \mathbf{I} s_\phi \mathbf{I} && (62)\\
&= c_\phi^2 + c_\phi s_\phi \mathbf{I} - s_\phi c_\phi \mathbf{I} - s_\phi^2 \mathbf{I}^2 && (63)\\
&= c_\phi^2 + (c_\phi s_\phi - s_\phi c_\phi)\mathbf{I} - s_\phi^2 (-1) && (64)\\
&= c_\phi^2 + 0\mathbf{I} + s_\phi^2 && (65)\\
&= c_\phi^2 + s_\phi^2 && (66)\\
&= 1 && (67)
\end{aligned}$$

A.3 Useful Bivector-Vector Products

Let $u = u_1 e_1 + u_2 e_2$. First compute $\mathbf{I}u$:

$$\begin{aligned}
\mathbf{I}u &= (e_1 e_2)(u_1 e_1 + u_2 e_2) && (68)\\
&= (e_1 e_2)(u_1 e_1) + (e_1 e_2)(u_2 e_2) && (69)\\
&= u_1 e_1 e_2 e_1 + u_2 e_1 e_2 e_2 && (70)\\
&= u_1 e_1 (e_2 e_1) + u_2 e_1 (e_2 e_2) && (71)\\
&= u_1 e_1 (-e_1 e_2) + u_2 e_1 (1) && (72)\\
&= -u_1 e_1 e_1 e_2 + u_2 e_1 && (73)\\
&= -u_1 e_2 + u_2 e_1 && (74)
\end{aligned}$$

Next compute $u\mathbf{I}$:

$$\begin{aligned} u\mathbf{I} &= (u_1e_1 + u_2e_2)(e_1e_2) && (75)\\ &= (u_1e_1)(e_1e_2) + (u_2e_2)(e_1e_2) && (76)\\ &= u_1e_1e_1e_2 + u_2e_2e_1e_2 && (77)\\ &= u_1(e_1e_1)e_2 + u_2(e_2e_1)e_2 && (78)\\ &= u_1e_2 + u_2(-e_1e_2)e_2 && (79)\\ &= u_1e_2 - u_2e_1e_2e_2 && (80)\\ &= u_1e_2 - u_2e_1 && (81) \end{aligned}$$

Finally compute $\mathbf{I}u\mathbf{I}$:

$$\begin{aligned} \mathbf{I}u\mathbf{I} &= (-u_1e_2 + u_2e_1)\mathbf{I} && (82)\\ &= (-u_1e_2 + u_2e_1)(e_1e_2) && (83)\\ &= -u_1e_2e_1e_2 + u_2e_1e_1e_2 && (84)\\ &= -u_1(e_2e_1)e_2 + u_2(e_1e_1)e_2 && (85)\\ &= -u_1(-e_1e_2)e_2 + u_2e_2 && (86)\\ &= u_1e_1e_2e_2 + u_2e_2 && (87)\\ &= u_1e_1 + u_2e_2 && (88)\\ &= u && (89) \end{aligned}$$

A.4 Deriving the Sandwich Action

Let $x = x_1e_1 + x_2e_2$. Start with the sandwich product:

$$\begin{aligned} r(\phi)\, xr(\phi)^\dagger &= (c_\phi - s_\phi\mathbf{I})x(c_\phi + s_\phi\mathbf{I}) && (90)\\ &= (c_\phi x - s_\phi\mathbf{I}x)(c_\phi + s_\phi\mathbf{I}) && (91)\\ &= (c_\phi x - s_\phi\mathbf{I}x)c_\phi + (c_\phi x - s_\phi\mathbf{I}x)s_\phi\mathbf{I} && (92)\\ &= c_\phi xc_\phi - s_\phi\mathbf{I}xc_\phi + c_\phi xs_\phi\mathbf{I} - s_\phi\mathbf{I}xs_\phi\mathbf{I} && (93)\\ &= c_\phi^2x - c_\phi s_\phi\mathbf{I}x + c_\phi s_\phi x\mathbf{I} - s_\phi^2\mathbf{I}x\mathbf{I} && (94) \end{aligned}$$

Substitute the identities from the previous subsection:

$$\begin{aligned} r(\phi)\, xr(\phi)^\dagger &= c_\phi^2x - c_\phi s_\phi(-x_1e_2 + x_2e_1) + c_\phi s_\phi(x_1e_2 - x_2e_1) - s_\phi^2x && (95)\\ &= c_\phi^2x - s_\phi^2x + c_\phi s_\phi(x_1e_2 - x_2e_1) + c_\phi s_\phi(x_1e_2 - x_2e_1) && (96)\\ &= (c_\phi^2 - s_\phi^2)x + 2c_\phi s_\phi(x_1e_2 - x_2e_1) && (97) \end{aligned}$$

Now expand $x = x_1e_1 + x_2e_2$ and collect the e_1 and e_2 coefficients:

$$\begin{aligned} r(\phi)\, xr(\phi)^\dagger &= (c_\phi^2 - s_\phi^2)(x_1e_1 + x_2e_2) + 2c_\phi s_\phi(x_1e_2 - x_2e_1) && (98)\\ &= (c_\phi^2 - s_\phi^2)x_1e_1 + (c_\phi^2 - s_\phi^2)x_2e_2 + 2c_\phi s_\phi x_1e_2 - 2c_\phi s_\phi x_2e_1 && (99)\\ &= \big((c_\phi^2 - s_\phi^2)x_1 - 2c_\phi s_\phi x_2\big)e_1 + \big((c_\phi^2 - s_\phi^2)x_2 + 2c_\phi s_\phi x_1\big)e_2 && (100) \end{aligned}$$

Use the double-angle identities:

$$c_\phi^2 - s_\phi^2 = \cos\phi \qquad 2c_\phi s_\phi = \sin\phi \tag{101}$$

The final expression is:

$$r(\phi)\, xr(\phi)^\dagger = (\cos\phi\, x_1 - \sin\phi\, x_2)e_1 + (\sin\phi\, x_1 + \cos\phi\, x_2)e_2 \tag{102}$$

A.5 Deriving Rotor Composition

Let:

$$c_{\phi_i} = \cos\left(\frac{\phi_i}{2}\right) \qquad s_{\phi_i} = \sin\left(\frac{\phi_i}{2}\right) \qquad i \in \{1, 2\} \tag{103}$$

Then:

$$\begin{aligned} r(\phi_2)\, r(\phi_1) &= (c_{\phi_2} - s_{\phi_2}\mathbf{I})(c_{\phi_1} - s_{\phi_1}\mathbf{I}) && (104)\\ &= c_{\phi_2}c_{\phi_1} - c_{\phi_2}s_{\phi_1}\mathbf{I} - s_{\phi_2}\mathbf{I}c_{\phi_1} + s_{\phi_2}\mathbf{I}s_{\phi_1}\mathbf{I} && (105)\\ &= c_{\phi_2}c_{\phi_1} - c_{\phi_2}s_{\phi_1}\mathbf{I} - s_{\phi_2}c_{\phi_1}\mathbf{I} + s_{\phi_2}s_{\phi_1}\mathbf{I}^2 && (106)\\ &= c_{\phi_2}c_{\phi_1} - c_{\phi_2}s_{\phi_1}\mathbf{I} - s_{\phi_2}c_{\phi_1}\mathbf{I} - s_{\phi_2}s_{\phi_1} && (107)\\ &= (c_{\phi_2}c_{\phi_1} - s_{\phi_2}s_{\phi_1}) - (c_{\phi_2}s_{\phi_1} + s_{\phi_2}c_{\phi_1})\mathbf{I} && (108) \end{aligned}$$

Use the angle-sum identities:

$$\begin{aligned} c_{\phi_2}c_{\phi_1} - s_{\phi_2}s_{\phi_1} &= \cos\left(\frac{\phi_2}{2}\right)\cos\left(\frac{\phi_1}{2}\right) - \sin\left(\frac{\phi_2}{2}\right)\sin\left(\frac{\phi_1}{2}\right) && (109)\\ &= \cos\left(\frac{\phi_1 + \phi_2}{2}\right) && (110)\\ c_{\phi_2}s_{\phi_1} + s_{\phi_2}c_{\phi_1} &= \cos\left(\frac{\phi_2}{2}\right)\sin\left(\frac{\phi_1}{2}\right) + \sin\left(\frac{\phi_2}{2}\right)\cos\left(\frac{\phi_1}{2}\right) && (111)\\ &= \sin\left(\frac{\phi_1 + \phi_2}{2}\right) && (112) \end{aligned}$$

Substitute these identities into the product:

$$\begin{aligned} r(\phi_2)\, r(\phi_1) &= \cos\left(\frac{\phi_1 + \phi_2}{2}\right) - \sin\left(\frac{\phi_1 + \phi_2}{2}\right)\mathbf{I} && (113)\\ &= r(\phi_1 + \phi_2) && (114) \end{aligned}$$

B Raster Sign Convention in Detail

The rotor action above lives in Euclidean coordinates with an upward-pointing vertical axis. Raster images use a downward-pointing vertical axis. This appendix derives the sign change.

Let (x_E, y_E) denote Euclidean coordinates and (x_I, y_I) denote image coordinates. The coordinate conversion is:

$$x_E = x_I \qquad y_E = -y_I \tag{115}$$

Define the reflection matrix:

$$S = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{116}$$

Then:

$$\begin{bmatrix} x_E \\ y_E \end{bmatrix} = S \begin{bmatrix} x_I \\ y_I \end{bmatrix} \tag{117}$$

Let the active Euclidean rotation by angle θ be:

$$R_E(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{118}$$

To express the same geometric transformation in image coordinates, compute:

$$\begin{bmatrix} x'_I \\ y'_I \end{bmatrix} = S^{-1} R_E(\theta) S \begin{bmatrix} x_I \\ y_I \end{bmatrix} \tag{119}$$

$$= S R_E(\theta) S \begin{bmatrix} x_I \\ y_I \end{bmatrix} \tag{120}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_I \\ y_I \end{bmatrix} \tag{121}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ \sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} x_I \\ y_I \end{bmatrix} \tag{122}$$

$$= \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_I \\ y_I \end{bmatrix} \tag{123}$$

$$= R_E(-\theta) \begin{bmatrix} x_I \\ y_I \end{bmatrix} \tag{124}$$

Therefore, a positive image-space rotation by θ corresponds to the Euclidean matrix $R_E(-\theta)$ and hence to the Euclidean rotor:

$$r_{\text{img}}(\theta) = r(-\theta) \tag{125}$$

C Composition, Frame Identifiability, and Asymmetry

This appendix proves the structural claims used in the main text.

C.1 Composition in Rotor Form

Because $r_{\text{img}}(\theta) = r(-\theta)$ and planar rotor multiplication adds angles, one has:

$$r_{\text{img}}(\theta_2) r_{\text{img}}(\theta_1) = r(-\theta_2)\, r(-\theta_1) \tag{126}$$

$$= r(-\theta_1 - \theta_2) \tag{127}$$

$$= r_{\text{img}}(\theta_1 + \theta_2) \tag{128}$$

This is the planar composition law used by the continuous warp. In three dimensions the corresponding statement is that the product of two rotors gives the composed rotation. The simple scalar angle-addition formula above is valid only for rotations in the same plane, or equivalently for commuting bivector generators.

C.2 Why Canonical Supervision is Necessary

Suppose a model predicts a canonical field ℓ_{can} and posed reconstructions $\mathcal{G}_{\theta_i} \ell_{\text{can}}$ for angles $\{\theta_i\}_{i=1}^n$. Let $\alpha \in \mathbb{R}$ be arbitrary and define:

$$\ell'_{\text{can}} = \mathcal{G}_\alpha \ell_{\text{can}} \qquad \theta'_i = \theta_i - \alpha \tag{129}$$

Then, by composition:

$$\mathcal{G}_{\theta'_i} \ell'_{\text{can}} = \mathcal{G}_{\theta_i - \alpha}(\mathcal{G}_\alpha \ell_{\text{can}}) \tag{130}$$

$$= \mathcal{G}_{\theta_i - \alpha + \alpha} \ell_{\text{can}} \tag{131}$$

$$= \mathcal{G}_{\theta_i} \ell_{\text{can}} \tag{132}$$

Therefore the observed reconstructions alone cannot determine the absolute canonical frame. Any global offset α produces an equivalent posed reconstruction, and canonical supervision removes this ambiguity by anchoring the decoded canonical silhouette to the known canonical target.

C.3 Why the Silhouettes Must be Asymmetric

Let c be a canonical silhouette. Its stabiliser under rotation is the set of angles that leave the silhouette unchanged:

$$\mathrm{Stab}(c) = \{\theta \in [0, 2\pi) : \mathcal{G}_\theta c = c\} \tag{133}$$

If $\mathrm{Stab}(c)$ contains an angle $\theta_0 \neq 0$, then the pose of c is identifiable only up to that symmetry. Specifically:

$$\mathcal{G}_{\theta+\theta_0} c = \mathcal{G}_\theta(\mathcal{G}_{\theta_0} c) \tag{134}$$

$$= \mathcal{G}_\theta c \tag{135}$$

Thus, a learning problem that aims to recover an absolute orientation must either treat orientations that differ by such a symmetry as equivalent or choose objects with no non-trivial rotational symmetry. Our synthetic prototypes are hand-designed to be asymmetric, so the stabiliser is trivial in practice.

D Loss Details and Auxiliary Calculations

This appendix expands the loss terms used in the main text.

D.1 Binary Cross-Entropy with Logits

Let $y \in \{0, 1\}$ be a binary target and let $\ell \in \mathbb{R}$ be a logit. The Bernoulli probability is:

$$p = \mathrm{sigm}(\ell) = \frac{1}{1 + e^{-\ell}} \tag{136}$$

The Bernoulli likelihood is:

$$P(y \mid \ell) = p^y (1 - p)^{1-y} \tag{137}$$

The negative log-likelihood is:

$$-\log P(y \mid \ell) = -\log \left(p^y (1 - p)^{1-y}\right) \tag{138}$$

$$= -\left(y \log p + (1 - y) \log(1 - p)\right) \tag{139}$$

$$= -y \log p - (1 - y) \log(1 - p) \tag{140}$$

Therefore, for an image with pixel set Ω, the mean binary cross-entropy with logits is:

$$\mathrm{BCE}(\ell, y) = \frac{1}{|\Omega|} \sum_{u \in \Omega} \left[-y(u) \log \mathrm{sigm}(\ell(u)) - (1 - y(u)) \log\big(1 - \mathrm{sigm}(\ell(u))\big)\right] \tag{141}$$

D.2 Induced Fused Posterior from Averaging Two Gaussian Samples

Let:

$$z_a \sim \mathcal{N}(\mu_a, \Sigma_a) \qquad z_b \sim \mathcal{N}(\mu_b, \Sigma_b) \tag{142}$$

with z_a and z_b independent. Define:

$$z_s = \frac{1}{2}(z_a + z_b) \tag{143}$$

The mean of z_s is:

$$\mathbb{E}[z_s] = \mathbb{E}\left[\frac{1}{2}(z_a + z_b)\right] \tag{144}$$

$$= \frac{1}{2}\mathbb{E}[z_a + z_b] \tag{145}$$

$$= \frac{1}{2}\left(\mathbb{E}[z_a] + \mathbb{E}[z_b]\right) \tag{146}$$

$$= \frac{1}{2}(\mu_a + \mu_b) \tag{147}$$

$$= \mu_s \tag{148}$$

The covariance of z_s is:

$$\begin{aligned}
\mathrm{Cov}(z_s) &= \mathrm{Cov}\left(\frac{1}{2}(z_a + z_b)\right) && (149)\\
&= \frac{1}{4}\,\mathrm{Cov}(z_a + z_b) && (150)\\
&= \frac{1}{4}\left(\mathrm{Cov}(z_a) + \mathrm{Cov}(z_b) + \mathrm{Cov}(z_a, z_b) + \mathrm{Cov}(z_b, z_a)\right) && (151)\\
&= \frac{1}{4}\left(\Sigma_a + \Sigma_b + 0 + 0\right) && (152)\\
&= \frac{1}{4}(\Sigma_a + \Sigma_b) && (153)\\
&= \Sigma_s && (154)
\end{aligned}$$

The zero cross-covariance terms appear because z_a and z_b are sampled independently. Therefore:

$$q_s(z \mid x_a, x_b) = \mathcal{N}(\mu_s, \Sigma_s) \tag{155}$$

D.3 Diagonal Gaussian Kullback-Leibler Divergence

Let:

$$q(z \mid x) = \mathcal{N}\big(\mu, \mathrm{diag}(\sigma_1^2, \ldots, \sigma_d^2)\big) \qquad p(z) = \mathcal{N}(0, I_d) \tag{156}$$

The Kullback-Leibler divergence is:

$$\mathrm{KL}(q\|p) = \mathbb{E}_q[\log q(z \mid x) - \log p(z)] \tag{157}$$

For the diagonal Gaussian posterior, the log density is:

$$\log q(z \mid x) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{d}\log\sigma_j^2 - \frac{1}{2}\sum_{j=1}^{d}\frac{(z_j - \mu_j)^2}{\sigma_j^2} \tag{158}$$

For the standard normal prior, the log density is:

$$\log p(z) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{d} z_j^2 \tag{159}$$

Subtract the two log densities:

$$\log q(z \mid x) - \log p(z) = -\frac{1}{2}\sum_{j=1}^{d}\log\sigma_j^2 - \frac{1}{2}\sum_{j=1}^{d}\frac{(z_j - \mu_j)^2}{\sigma_j^2} + \frac{1}{2}\sum_{j=1}^{d} z_j^2 \tag{160}$$

Take expectation under q:

$$\mathrm{KL}(q\|p) = -\frac{1}{2}\sum_{j=1}^{d}\log\sigma_j^2 - \frac{1}{2}\sum_{j=1}^{d}\mathbb{E}_q\left[\frac{(z_j - \mu_j)^2}{\sigma_j^2}\right] + \frac{1}{2}\sum_{j=1}^{d}\mathbb{E}_q[z_j^2] \tag{161}$$

Compute the first expectation term:

$$\begin{aligned}
\mathbb{E}_q\left[\frac{(z_j - \mu_j)^2}{\sigma_j^2}\right] &= \frac{1}{\sigma_j^2}\mathbb{E}_q[(z_j - \mu_j)^2] && (162)\\
&= \frac{1}{\sigma_j^2}\sigma_j^2 && (163)\\
&= 1 && (164)
\end{aligned}$$

Compute the second expectation term:

$$\mathbb{E}_q[z_j^2] = \mathrm{Var}_q(z_j) + \big(\mathbb{E}_q[z_j]\big)^2 \tag{165}$$
$$= \sigma_j^2 + \mu_j^2 \tag{166}$$

Substitute both expectations into the KL expression:

$$\mathrm{KL}(q\|p) = -\frac{1}{2}\sum_{j=1}^{d}\log\sigma_j^2 - \frac{1}{2}\sum_{j=1}^{d}1 + \frac{1}{2}\sum_{j=1}^{d}(\sigma_j^2 + \mu_j^2) \tag{167}$$

$$= \frac{1}{2}\sum_{j=1}^{d}\left(\sigma_j^2 + \mu_j^2 - 1 - \log\sigma_j^2\right) \tag{168}$$

The implementation parameterises $\mathrm{logvar}_j = \log\sigma_j^2$. Therefore:

$$\sigma_j^2 = e^{\mathrm{logvar}_j} \tag{169}$$

$$\mathrm{KL}(q\|p) = \frac{1}{2}\sum_{j=1}^{d}\left(e^{\mathrm{logvar}_j} + \mu_j^2 - 1 - \mathrm{logvar}_j\right) \tag{170}$$

D.4 Why the Confidence Penalty Sharpens Binary Occupancy

For a Bernoulli random variable $Y \sim \mathrm{Bernoulli}(p)$, first compute the mean:

$$\mathbb{E}[Y] = 0 \cdot P(Y=0) + 1 \cdot P(Y=1) \tag{171}$$
$$= 0 \cdot (1-p) + 1 \cdot p \tag{172}$$
$$= p \tag{173}$$

Next compute the second moment:

$$\mathbb{E}[Y^2] = 0^2 \cdot P(Y=0) + 1^2 \cdot P(Y=1) \tag{174}$$
$$= 0^2 \cdot (1-p) + 1^2 \cdot p \tag{175}$$
$$= p \tag{176}$$

Then the variance is:

$$\mathrm{Var}(Y) = \mathbb{E}[Y^2] - \mathbb{E}[Y]^2 \tag{177}$$
$$= p - p^2 \tag{178}$$
$$= p(1-p) \tag{179}$$

The derivative of $p(1-p)$ with respect to p is:

$$\frac{d}{dp}\left(p(1-p)\right) = \frac{d}{dp}(p - p^2) \tag{180}$$
$$= 1 - 2p \tag{181}$$

The derivative vanishes when:

$$1 - 2p = 0 \tag{182}$$
$$2p = 1 \tag{183}$$
$$p = \frac{1}{2} \tag{184}$$

The second derivative is:

$$\frac{d^2}{dp^2}\left(p(1-p)\right) = -2 \tag{185}$$

Since the second derivative is negative, $p = 1/2$ is the maximum. The minimum on the interval $[0, 1]$ occurs at the endpoints $p = 0$ and $p = 1$. Therefore penalising $p(1-p)$ discourages ambiguous probabilities near $1/2$ and encourages crisp binary occupancies.

E Full Reproducibility Details

This appendix records the exact architecture and data generation details used by the Python implementation.

E.1 Network Architecture

Table 2 lists the exact layers used in the compressed setting. The illustrative reconstruction setting changes only the base width and latent dimension.

Component	Layers
Encoder E_ω	Conv2d$(1, C, 4, 2, 1)$ + SiLU; Conv2d$(C, 2C, 4, 2, 1)$ + SiLU; Conv2d$(2C, 4C, 4, 2, 1)$ + SiLU; Conv2d$(4C, 8C, 4, 2, 1)$ + SiLU; Flatten; Linear$(8C \cdot (H/16)^2, 256)$ + SiLU; Linear$(256, d)$ for μ; Linear$(256, d)$ for $\log \sigma^2$.
Rotor decoder D_ψ	Linear$(d, 256)$ + SiLU; Linear$(256, 8C \cdot (H/16)^2)$ + SiLU; ConvTranspose2d$(8C, 4C, 4, 2, 1)$ + SiLU; ConvTranspose2d$(4C, 2C, 4, 2, 1)$ + SiLU; ConvTranspose2d$(2C, C, 4, 2, 1)$ + SiLU; ConvTranspose2d$(C, 1, 4, 2, 1)$; exact rotor warp applied after decoding.
Coord.-conditioned learned rendering baseline decoder	Linear$(d + 2, 256)$ + SiLU; Linear$(256, 8C \cdot (H/16)^2)$ + SiLU; ConvTranspose2d$(8C, 4C, 4, 2, 1)$ + SiLU; ConvTranspose2d$(4C, 2C, 4, 2, 1)$ + SiLU; ConvTranspose2d$(2C, C, 4, 2, 1)$ + SiLU; ConvTranspose2d$(C, 1, 4, 2, 1)$.

Table 2: Exact layer definitions. In the compressed setting $C = 8$ and $d = 16$. In the larger illustrative reconstruction setting $C = 32$ and $d = 64$. The image size is $H = W = 32$.

E.2 Optimisation Hyperparameters

The compressed multi-seed setting uses Adam with learning rate 2×10^{-3}, batch size 32, 12 epochs, seeds $\{7, 11, 17\}$, 256 training pairs, and 128 validation pairs. The larger illustrative reconstruction setting uses the same optimiser and learning rate, batch size 32, 16 epochs, latent dimension 64, base width 32, seed 7, 256 training pairs, and 64 validation pairs. The hyperparameters and loss weights were selected empirically in pilot runs and then fixed before the reported comparisons.

The exact loss weights are:

$$\begin{aligned} \beta_z = 10^{-4} \quad \lambda_{\text{can-BCE}} = 5.0 \quad \lambda_{\text{can-Dice}} = 2.0 \\ \lambda_{\text{view-Dice}} = 1.0 \qquad \lambda_{\text{inv}} = 5.0 \qquad \lambda_{\text{conf}} = 0.02 \end{aligned} \tag{186}$$

E.3 Prototype Polygons

The six canonical prototype polygons are specified by the following ordered vertex lists in local coordinates before scaling:

$$T_1 = [(-1.1, -0.35), (0.1, -0.35), (0.1, -0.75), (1.0, 0.0), (0.1, 0.75), (0.1, 0.35), (-1.1, 0.35), (-0.75, 0.0)] \tag{187}$$

$$T_2 = [(-1.0, -1.0), (-0.2, -1.0), (-0.2, 0.2), (1.0, 0.2), (1.0, 1.0), (-1.0, 1.0)] \tag{188}$$

$$T_3 = [(-1.0, -0.8), (0.6, -0.8), (1.0, -0.2), (1.0, 0.9), (-0.7, 0.9), (-1.0, 0.2)] \tag{189}$$

$$T_4 = [(0.0, -1.2), (0.9, -0.2), (0.4, 1.0), (-0.7, 0.5), (-1.0, -0.3)] \tag{190}$$

$$T_5 = [(-1.0, -0.9), (-0.2, -0.9), (-0.2, -0.2), (1.0, -0.2), (0.2, 0.3), (1.0, 0.8), (-0.2, 0.8), (-0.2, 1.0), (-1.0, 1.0)] \tag{191}$$

$$T_6 = [(-1.1, -0.6), (0.2, -0.9), (1.0, -0.2), (0.6, 0.9), (-0.7, 1.0), (-1.0, 0.2)] \tag{192}$$

For each sample, one template is chosen uniformly at random. Let $\alpha = 8$ denote the anti-aliasing factor and $H = 32$ the final image size. The high-resolution canvas has side length $H\alpha$ and the global scale is:

$$s = 0.22\, H\alpha \tag{193}$$

If (x_k, y_k) is a local vertex of a template and (c_x, c_y) is the centre of the high-resolution canvas, then the rasterised vertex is:

$$(c_x + sx_k,\ c_y + sy_k) \tag{194}$$

The filled polygon is downsampled bilinearly to 32×32 and then thresholded at 0.5 to return a binary silhouette.

F Software, Implementation, and `clifford` Verification

All experiments were implemented in Python using PyTorch (Python 3.13.5 and torch 2.10.0) on CPU. The implementation additionally uses NumPy, Pillow and Matplotlib for data generation and visualisation.

The analytic rotor formulas were independently checked using the Python `clifford` library (Hadfield et al.). The verification script evaluates the sandwich action and rotor composition numerically. The recorded maximum absolute errors were:

$$\text{max rotor-vector error} = 4.44 \times 10^{-16} \qquad \text{max composition error} = 3.33 \times 10^{-16} \tag{195}$$

These values are consistent with floating-point roundoff and confirm that the analytic formulas used in the paper match the independent library implementation.

G Natural Extension to Three-dimensional Rotors and Conformal Geometric Algebra Motors

This appendix sketches a possible extension and it is not an empirical claim of the present paper. In Euclidean Geometric Algebra $\mathrm{Cl}(3,0)$, a pure rotation is represented by a spatial rotor R. A vector $x \in \mathbb{R}^3$ is rotated by the sandwich product:

$$x' = RxR^{\dagger} \tag{196}$$

The product of two spatial rotors gives the composed rotation:

$$R_{21} = R_2 R_1 \tag{197}$$

Unlike the planar case, this composition is generally not equivalent to adding two scalar angles unless the rotations share a plane of rotation.

For full rigid motion, one can use Conformal Geometric Algebra (CGA). In CGA, a motor M combines rotation and translation in one compositional object and acts on an embedded conformal point X by:

$$X' = MXM^{\dagger} \tag{198}$$

A direct three-dimensional analogue of the present model would preserve the same separation of responsibilities. The encoder would infer pose-invariant object content from posed observations, a canonical decoder would predict a canonical three-dimensional representation, and the known pose would be applied analytically at the output rather than learned implicitly by the decoder. One possible choice is a canonical occupancy logit field:

$$\ell_{\mathrm{can}} : [-1, 1]^3 \to \mathbb{R} \tag{199}$$

Posed volumes or rendered views could then be obtained by evaluating this field at inverse-transformed query points induced by a rotor or motor. On a voxel grid, this would use trilinear sampling, just as the present two-dimensional model uses bilinear sampling.

The present paper does not claim that the two-dimensional experiment proves the three-dimensional motor case. It identifies a reusable architectural principle in which canonical content is decoded once and known geometric transformations are routed through an exact analytic object rather than through generic pose coordinates.