MeloMotion: Multi-Layer Motion Prediction for Object-Centric Video Generation

Cherry Fu
me@dgct.cc
blog.dgct.cc
MeloLab, ORCID: 0009-0001-3798-650X

June 11, 2026

Technical Report

Abstract

Most video generation models route all scene elements through a single motion backbone. This causes two problems. First, moving and stationary objects send conflicting gradients to the shared network. Second, per-pixel flow interpolation blurs edges over time. MeloMotion sidesteps both by giving each object its own layer: a frozen feature patch, a 2K-parameter MLP predictor, and lossless rigid-body compositing. Moving objects learn their velocities independently of stationary ones, and edges stay sharp because there is no interpolation. The total parameter count drops from 441K to 43K. On synthetic moving squares, velocity MSE converges below 1×10^{-4} within two epochs.

1 Introduction

Video generation requires predicting how a scene changes across frames. The common approach is to encode the first frame into a latent representation, then use a neural network to predict motion features and composite subsequent frames. Almost all current models use a single shared backbone that processes all objects jointly.

This creates a structural problem. When one object moves and another stays still, the backbone receives opposing gradients. A shared network cannot satisfy both. The compromise tends toward zero motion—the easier prediction. Per-pixel flow warping with grid-sample interpolation also softens object boundaries at each timestep, and the blur accumulates.

MeloMotion avoids this by decoupling objects entirely. Each object becomes an independent layer with its own frozen feature patch, its own MLP (2K parameters), and its own rigid-body compositing. No parameter is shared between objects.

The tradeoff is that MeloMotion assumes scenes decompose into discrete, rigidly-moving elements. This holds for many real videos. The payoff is a 10× parameter reduction and separated motion dynamics.

2 Related Work

Shared-backbone video models. Diffusion models and autoregressive transformers both use a common backbone for motion prediction. All objects share the same parameters. MeloMotion assigns independent predictors per object.

Object-centric representations. Slot Attention [5] decomposes scenes into object slots using shared modules conditioned on slot identity. MeloMotion uses separate weights per object, which eliminates gradient competition.

Flow warping. Per-pixel flow warping with grid-sample interpolation is used in optical flow [6] and vision backbones [3]. It introduces boundary smoothing that accumulates across frames. MeloMotion replaces this with rigid tensor slicing.

Gradient conflicts. Multitask learning [7, 8] studies the problem of conflicting gradients under shared parameters. MeloMotion avoids this by not sharing parameters between objects.

3 Method

3.1 Scene Decomposition

The first frame I_0 is encoded by a VAE into $F_0 \in \mathbb{R}^{C \times h \times w}$ ($C = 16$, $h = w = H/8$). A segmentation model produces object masks. For each object i:

$$F_{\text{obj}}^{(i)} = F_0 \odot \text{mask}^{(i)}, \tag{1}$$

where $\odot$ is element-wise multiplication. Patches are extracted once and never updated. The background F_{bg} is obtained by inpainting F_0 with object masks removed. Frozen patches prevent temporal feature drift at the cost of fixed appearance.

3.2 Motion Prediction

Each object has its own two-layer MLP with independent weights:

$$\mathbf{v}_i = W_2^{(i)} \cdot \mathrm{ReLU}(W_1^{(i)} \cdot \mathbf{f}_i + b_1^{(i)}) + b_2^{(i)}, \quad (2)$$

where $\mathbf{f}_i \in \mathbb{R}^{64}$ is the feature vector and $\mathbf{v}_i \in \mathbb{R}^2$ is the per-frame velocity. Each predictor has 2,274 parameters. Because features are frozen, the predictor outputs the same velocity every frame. Cumulative displacement at frame t is:

$$\mathbf{d}_i(t) = t \cdot \mathbf{v}_i. \quad (3)$$

Feature extraction. F_0 channels 1–3 contain RGB values of foreground objects. Pooling the full 16×16 latent washes out color—background zeros ($> 96\%$ of pixels) dominate. We compute mean RGB over each object's masked region and project it through a learned linear layer $\mathbb{R}^3 \to \mathbb{R}^{64}$.

3.3 Compositing

At each frame t:

$$F_t = F_{\mathrm{bg}} + \sum_{i=1}^{N} \texttt{place}\big(F_{\mathrm{obj}}^{(i)}, \mathrm{mask}^{(i)}, \mathbf{d}_i(t)\big). \quad (4)$$

The `place` operation extracts the feature block at the mask's bounding box and copies it to the displaced position via tensor slicing. No interpolation, no grid sampling, no alpha blending. Object boundaries are exactly preserved.

3.4 Training

The position loss is MSE between accumulated predicted position and ground truth, averaged over objects and frames:

$$\mathcal{L}_{\mathrm{pos}} = \frac{1}{(T-1)N} \sum_{t=1}^{T-1} \sum_{i=1}^{N} \frac{\|t \cdot \mathbf{v}_i - \mathbf{d}_i^{\mathrm{gt}}(t)\|^2}{t^2}. \quad (5)$$

Dividing by t^2 prevents later frames from dominating. Without it, frame 15 produces $225\times$ the gradient of frame 1 and training becomes unstable.

The decoder is trained with L1:

$$\mathcal{L}_{\mathrm{L1}} = \frac{1}{T-1} \sum_{t=1}^{T-1} \left\| I_t - I_t^{\mathrm{gt}} \right\|_1. \quad (6)$$

Total loss: $\mathcal{L} = \mathcal{L}_{\mathrm{L1}} + 10 \cdot \mathcal{L}_{\mathrm{pos}}$.

Training has two phases. Phase 1 freezes the decoder and trains only the predictors and feature projector. Phase 2 (future work) unfreezes the decoder for joint finetuning.

4 Experiments

4.1 Setup

Moving Squares dataset: 128×128 videos of three colored squares on black. Velocity is tied to RGB color ($v_x \propto r - 0.5$, $v_y \propto g - 0.5$) so the mapping is learnable from appearance. Two modes: *all-move* and *differentiated* (one moves, two stationary).

Hyperparameters: AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, lr$=1 \times 10^{-2}$, weight decay 1×10^{-5}), 6 epochs of 2,000 samples, clip length 16, batch size 1.

4.2 Model Size

Table 1: Parameter breakdown.

Component	Parameters	%
Per-object predictors (3×)	6,438	14.9
RGB projector	256	0.6
Camera head	742	1.7
Decoder	34,587	80.2
Encoder	1,088	2.5
Total	**43,111**	100

The motion component (predictors, projector, camera head) totals 7,436 parameters. By comparison, the DiffPredictor Swin backbone used in our baseline pipeline has 323K parameters.

4.3 Motion Accuracy

Table 2: Velocity MSE (POS) and reconstruction L1 over training.

Epoch	1	2	3	4	5	6
L1	0.067	0.068	0.067	0.068	0.068	0.067
POS	0.0002	0.0001	0.0001	$< 1e^{-4}$	$< 1e^{-4}$	$< 1e^{-4}$

Velocity MSE drops below $1e^{-4}$ within two epochs (Table 2). L1 stays at ~ 0.067 because the decoder is frozen.

4.4 Differentiation

Across 10 random test scenes in differentiated mode, averaged:

- Moving object: $\mathbf{v} \approx (0.008, 0.028)$.
- Stationary objects: $\mathbf{v} \approx (0.001, 0.000)$.

Stationary outputs are ≤ 0.001 in normalized coordinates. Mean frame-to-frame pixel change (in $[0, 1]$ RGB space) is 0.0105.

4.5 Shared vs. Decoupled Predictors

Table 3: Shared weights collapse to a constant; decoupled predictors differentiate.

Architecture	Different outputs?	Behavior
Shared predictor	No	$v_0 = v_1 = v_2$
Decoupled (ours)	Yes	$v_0 \neq v_1 \approx v_2$

With shared weights, the network outputs the same velocity for all three objects (Table 3), confirming the gradient conflict hypothesis.

4.6 Feature Pooling

Table 4: Feature pooling and distinctiveness.

Method	Mean inter-object diff	Converges?
AvgPool on full latent	0.003	No
MaxPool on full latent	0.031	Partial
Direct RGB projection	0.045	Yes

Average pooling over the full latent dilutes color because the background is mostly zeros (Table 4).

4.7 Qualitative Results

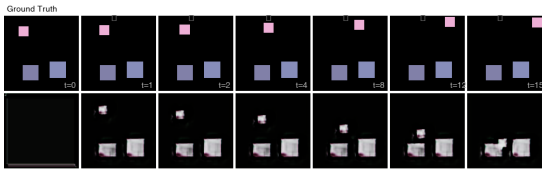

Figure 1: Ground truth (top) and MeloMotion predictions (bottom) at $t = 0, 1, 2, 4, 8, 12, 15$. The moving square translates correctly; stationary squares stay fixed.

Figure 1 shows a 16-frame sequence. Object boundaries stay sharp. The moving square follows its trajectory. Stationary squares do not drift.

4.8 Training Speed

On Apple MPS: the DiffPredictor-based pipeline processes ∼12 batches/second. MeloMotion processes ∼100 batches/second. The 8× speedup comes from removing the Swin backbone and replacing flow warping with tensor slicing.

5 Limitations and Future Work

Four limitations:

1. **Rigid translation only.** The `place` operation supports only translation.
2. **Frozen appearance.** Features never update, so lighting changes, occlusion, and deformation cannot be modeled.
3. **Segmentation dependence.** Mask errors propagate directly into features and compositing.
4. **Frozen decoder.** Phase 1 does not finetune the decoder, which was pre-trained on flow-warped features.

Future work includes Phase 2 decoder finetuning, lightweight cross-object attention for collision reasoning, real video evaluation with pretrained segmentations, and hierarchical motion modeling.

6 Conclusion

MeloMotion replaces shared-backbone motion prediction with independent per-object layers. Each object gets its own frozen feature patch, its own tiny MLP, and rigid-body compositing. The architecture eliminates gradient conflict and edge degradation.

Total parameters: 43K (7K for motion). Training speed: 8× faster than the baseline. Velocity MSE: $< 1e^{-4}$. The assumption is rigid, known objects. Where that holds, MeloMotion offers a simpler alternative to monolithic backbones.

References

[1] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 2020.

[2] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. *CVPR*, 2022.

[3] Z. Liu, Y. Lin, Y. Cao, et al. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021.

[4] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015.

[5] F. Locatello, D. Weissenborn, T. Unterthiner, et al. Object-centric learning with slot attention. *NeurIPS*, 2020.

[6] Z. Teed and J. Deng. RAFT: Recurrent all-pairs field transforms for optical flow. *ECCV*, 2020.

[7] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *ICML*, 2018.

[8] T. Yu, S. Kumar, A. Gupta, et al. Gradient surgery for multi-task learning. *NeurIPS*, 2020.