Supplementary material for:

Hierarchical VAEs provide a normative account of motion processing in the primate brain

8 Study Limitations & Considerations

We established a synthetic data framework that allows hypothesis generation and testing for neural processing of motion. While our paper opens up many interesting venues for future exploration, it is necessarily simplified, as it focuses on establishing our framework and demonstrating its potential. As such, it currently has several limitations:

First, our simulation generates velocity fields, rather than full spatiotemporal movies, which allows our model to avoid mimicking the complexities of motion extracted by the early visual pathway [120, 121]. This strategy also allows for direct comparison with recorded neural data in MT and MST using random dot kinematograms [100, 101, 103]. However, a more complex model would be necessary to explain responses of neurons earlier in the visual pathway, such as V1, which would require a pixel-computable model as in previous work (e.g., DorsalNet [34]). Likewise, our $> 2\times$ improvement in performance in explaining MT neural data over DorsalNet is likely due in part to their network trained to perform this extraction from spatiotemporal movies, which was not necessarily equivalent to the random dot velocity fields used in the experiments. Thus, it is an open question whether a hierarchical VAE trained and tested on video stimuli would better align to neural data than the model from Mineault et al. [34]. Future work in this space will involve rendering images in simulations and using image-computable models for a fair comparison.

Second, we chose relatively simple environments and simple fixation rules to generate our optic flow fields and avoided the true complexity of 3-D natural environments and their interaction with eye movements and self-motion as has recently been measured [80, 106]. The simplification of our simulation still demonstrates the importance of including such elements in understanding neural representations and provides a framework for incorporating real eye-tracking and scene data [80, 106] into future work with ROFL.

Finally, we only tested neural alignment on one experimental paradigm using neurons in area MT, which leaves the question of whether this is a general principle of brain computation. Addressing this requires testing our approach on more data from other brain areas, such as MST. Based on previous work [103, 107], we expect that hierarchical computation is even more necessary for MST, which is an open question to address in future work.

Interpreting brain-alignment. We measured the alignment between ANN models and MT neurons using both linear predictive power (Fig. 7) and an alternative measure of alignment that is sensitive to the sparsity of latent-to-neuron relationships ("alignment-score", Fig. 8a). Linear regression has been most commonly used to measure similarity, or alignment, between pairs of representations [30, 31, 34, 36], but often results in degenerate [30, 36, 38] and unreliable [122] measures of representational alignment. Consistent with this, our application of linear regression found that it was not effective in differentiating between models: although the cNVAE produced the single best model in terms of neuronal prediction, we found that both hierarchical and non-hierarchical VAEs performed similarly in predicting MT neuron responses (Fig. 7).

In contrast, the alignment score (Fig. 8a) was much more consistent in distinguishing between models (see Fig. 16), and revealed that hierarchical models (both cNVAE and cNAE) had significantly sparser latent-to-neuron relationships. The alignment score measures whether a model has learned a similar representational "form" to the brain, which would enable the sparsity of latent-to-neuron relationships. This concept is closely related to the "*completeness*" score from the DCI framework [22]. The alignment score shown in Fig. 8a was also used in Higgins et al. [16], although they used the magnitude of nonzero coefficients as their feature importance (under lasso regression). However, we note that this alignment score also has limitations, and for example, it is not a proper metric in the mathematical sense [109]. Future work will consider more sophisticated metrics for brain alignment [109–112].

9 Additional Methods

9.1 VAE loss derivation

Suppose some observed data $\boldsymbol{x}$ are sampled from a generative process as follows:

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{z})\, d\boldsymbol{z} = \int p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})\, d\boldsymbol{z}, \tag{1}$$

where $\boldsymbol{z}$ are latent (or unobserved) variables. In this setting, it is interesting to ask which set of latents $\boldsymbol{z}$ are likely, given an observation $\boldsymbol{x}$. In other words, we are interested in posterior inference

$$p(\boldsymbol{z}|\boldsymbol{x}) \propto p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z}). \tag{2}$$

The goal of VAEs is to approximate the (unknown) true posterior $p(\boldsymbol{z}|\boldsymbol{x})$ with a distribution $q(\boldsymbol{z}|\boldsymbol{x};\theta)$, where θ are some free parameters to be learned. This goal is achieved by minimizing the Kullback-Leibler divergence between the approximate posterior q and the true posterior p:

$$\text{Goal:} \quad \text{minimize} \quad \mathcal{D}_{\text{KL}}\Big[q(\boldsymbol{z}|\boldsymbol{x};\theta) \,\Big\|\, p(\boldsymbol{z}|\boldsymbol{x})\Big]. \tag{3}$$

This objective is intractable, but rearranging the terms leads to the following loss that is also the (negative) variational lower bound on $\log p(\boldsymbol{x})$:

$$\mathcal{L}_{\text{VAE}} = -\mathbb{E}_q\Big[\log p(\boldsymbol{x}|\boldsymbol{z};\boldsymbol{\theta}_{dec})\Big] + \mathcal{D}_{\text{KL}}\Big[q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}_{enc}) \,\Big\|\, p(\boldsymbol{z}|\boldsymbol{\theta}_{dec})\Big], \tag{4}$$

where $\boldsymbol{\theta}_{enc}$ and $\boldsymbol{\theta}_{dec}$ are the parameters for the encoder and decoder neural networks (see Fig. 2). The first term in Equation 4 is the reconstruction loss, which we chose to be the Euclidean norm of the differences between predicted and reconstructed velocity vectors (i.e., the endpoint error [81]). We will now focus on the second term in Equation 4, the KL term.

9.1.1 Prior, approximate posterior, and the KL term in vanilla VAE

In standard non-hierarchical VAEs, the prior is not parameterized. Instead, it is chosen to be a simple distribution such as a Gaussian with zero mean and unit covariance. Additionally, the prior does not depend on the inputs $\boldsymbol{x}$, that is:

$$p(\boldsymbol{z}|\boldsymbol{\theta}_{dec}) \;\rightarrow\; p(\boldsymbol{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I}). \tag{5}$$

The approximate posterior is also a Gaussian with mean $\boldsymbol{\mu}(\boldsymbol{x};\boldsymbol{\theta}_{enc})$ and variance $\boldsymbol{\sigma}^2(\boldsymbol{x};\boldsymbol{\theta}_{enc})$:

$$q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}_{enc}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x};\boldsymbol{\theta}_{enc}), \boldsymbol{\sigma}^2(\boldsymbol{x};\boldsymbol{\theta}_{enc})) \tag{6}$$

As a result, we see that the KL term for a vanilla VAE only depends on encoder parameters $\boldsymbol{\theta}_{enc}$.

9.1.2 Prior, approximate posterior, and the KL term in the cNVAE

Similar to the NVAE [52], the cNVAE latent space is organized hierarchically such that latent groups are sampled sequentially, starting from "top" latents all the way down to the "bottom" ones (i.e., $\boldsymbol{z}_1$ to $\boldsymbol{z}_3$ in Fig. 2). In addition to its hierarchical structure, another important difference between cNVAE and vanilla VAE is that the cNVAE prior is learned from data. Note the "mid" and "bottom" latent groups in Fig. 2, indicated as $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$ respectively: the cNVAE is designed in a way that changing the parameters along the decoder pathway will impact the prior distributions on $\boldsymbol{z}_2$ and $\boldsymbol{z}_3$ (but not $\boldsymbol{z}_1$). Note the "h" in Fig. 2, which is also a learnable parameter. In summary, the "top" cNVAE latents (e.g., $\boldsymbol{z}_1$ in Fig. 2) have a fixed prior distribution similar to vanilla VAEs; whereas, the prior distribution for every other cNVAE latent group is parametrized and learned from data.

More formally, the cNVAE latents are partitioned into disjoint groups, $\boldsymbol{z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_L\}$, where L is the number of groups. Then, the prior is represented by:

$$p(\boldsymbol{z}|\boldsymbol{\theta}_{dec}) = \prod_{\ell=1}^{L} p(\boldsymbol{z}_\ell|\boldsymbol{z}_{<\ell};\boldsymbol{\theta}_{dec}) = p(\boldsymbol{z}_1)\cdot\prod_{\ell=2}^{L} p(\boldsymbol{z}_\ell|\boldsymbol{z}_{<\ell};\boldsymbol{\theta}_{dec}), \tag{7}$$

where the conditionals are represented by factorial Normal distributions. For the first latent group we have $p(\boldsymbol{z}_1) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$, similar to vanilla VAEs. In contrast, for every other latent group, we have:

$$p(\boldsymbol{z}_\ell|\boldsymbol{z}_{<\ell};\boldsymbol{\theta}_{dec}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{z}_{<\ell};\boldsymbol{\theta}_{dec}), \boldsymbol{\sigma}^2(\boldsymbol{z}_{<\ell};\boldsymbol{\theta}_{dec})), \tag{8}$$

where $\boldsymbol{\mu}(\boldsymbol{z}_{<\ell};\boldsymbol{\theta}_{dec})$ and $\boldsymbol{\sigma}^2(\boldsymbol{z}_{<\ell};\boldsymbol{\theta}_{dec})$ are outputted from the decoder *sampler* layers. The approximate posterior is represented by:

$$q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}_{enc}) = \prod_{\ell=1}^{L} q(\boldsymbol{z}_\ell|\boldsymbol{z}_{<\ell},\boldsymbol{x};\boldsymbol{\theta}_{enc}). \tag{9}$$

Similarly, we adopt a Gaussian parameterization for each conditional in the approximate posterior:

$$q(\boldsymbol{z}_\ell|\boldsymbol{z}_{<\ell},\boldsymbol{x};\boldsymbol{\theta}_{enc}) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{z}_{<\ell},x;\boldsymbol{\theta}_{enc}), \boldsymbol{\sigma}^2(\boldsymbol{z}_{<\ell},\boldsymbol{x};\boldsymbol{\theta}_{enc})), \tag{10}$$

where $\boldsymbol{\mu}(\boldsymbol{z}_{<\ell},x;\boldsymbol{\theta}_{enc})$ and $\boldsymbol{\sigma}^2(\boldsymbol{z}_{<\ell},\boldsymbol{x};\boldsymbol{\theta}_{enc})$ are outputted from the encoder *sampler* layers. We are now in a position to explicitly write down the KL term from Equation 4 for the cNVAE:

$$\text{KL term } = \mathcal{D}_{\text{KL}}\Big[q(\boldsymbol{z}_1|x,\boldsymbol{\theta}_{enc}) \,\Big\|\, p(\boldsymbol{z}_1)\Big] + \sum_{\ell=2}^{L} \mathbb{E}_{q(\boldsymbol{z}_{<\ell}|\boldsymbol{x},\boldsymbol{\theta}_{enc})}\Big[\text{KL}_\ell(\boldsymbol{\theta}_{enc},\boldsymbol{\theta}_{dec})\Big], \tag{11}$$

where KL_ℓ refers to the local KL term for group ℓ and is given by:

$$\text{KL}_\ell(\boldsymbol{\theta}_{enc},\boldsymbol{\theta}_{dec}) \coloneqq \mathcal{D}_{\text{KL}}\Big[q\left(\boldsymbol{z}_\ell|\boldsymbol{z}_{<\ell},x;\boldsymbol{\theta}_{enc}\right) \Big\|\, p\left(\boldsymbol{z}_\ell|\boldsymbol{z}_{<\ell},\boldsymbol{\theta}_{dec}\right)\Big], \tag{12}$$

and the approximate posterior up to the $(\ell-1)^{th}$ group is defined as:

$$q(\boldsymbol{z}_{<\ell}|\boldsymbol{x};\boldsymbol{\theta}_{enc}) \coloneqq \prod_{i=1}^{\ell-1} q(\boldsymbol{z}_i|\boldsymbol{z}_i,\boldsymbol{x};\boldsymbol{\theta}_{enc}). \tag{13}$$

This concludes our derivation of the KL terms shown in Table 2.

9.2 Key architectural differences between NVAE and cNVAE

Our main contribution to architecture design lies in the modification of the NVAE "*sampler*" layer and the introduction of the "*expand*" layer, which are simple deconvolutions. In the NVAE, the purpose of a sampler layer is to map encoder features to mean and variance vectors $\boldsymbol{\mu}(\boldsymbol{x})$ and $\boldsymbol{\sigma}^2(\boldsymbol{x})$, which are then used to construct approximate posterior distributions $q\left(\boldsymbol{z}|\boldsymbol{x}\right) = \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\sigma}^2(\boldsymbol{x}))$. Note that during inference, downsampling operations will progressively reduce the spatial scale of representations along the encoder pathway, such that processed stimuli will have different spatial dimensions at different stages of inference. In the NVAE, all sampler layers are convolutional with a fixed kernel size of 3×3 and padding of 1. Thus, the application of a sampler layer does not alter the spatial dimension of hidden encoder features, resulting in a convolutional latent space.

For example, at level ℓ of the hierarchy, $\boldsymbol{\mu}_\ell\left(\boldsymbol{x}\right)$ and $\boldsymbol{\sigma}_\ell^2\left(\boldsymbol{x}\right)$ will have shapes like $d \times s_\ell \times s_\ell$, where d is number of latent variables per latent group, and s_ℓ is the spatial scale of the processed encoder features at level ℓ. As a result, NVAE latents themselves end up being convolutional in nature, with shapes like $d \times s_\ell \times s_\ell$. This design choice leads to the massive expansion of NVAE latent dimensionality, which is often orders of magnitude larger than the input dimensionality.

To address this, we modified NVAE sampler layers in a manner that would integrate out spatial information before the sampling step, achieving a compressed, non-convolutional, and more abstract latent space. Specifically, we designed the cNVAE sampler layers such that their kernel sizes are selected adaptively to match the spatial scale of their input feature. For example, if latent group ℓ operates at the scale of s_ℓ, then the sampler layer will have a kernel size of $s_\ell \times s_\ell$, effectively integrating over space before the sampling step. This results in latents with shape $d \times 1 \times 1$.

Finally, combining the latents with decoder features requires projecting them back into convolutional space. We achieve this by introducing "*expand*" modules which are simple deconvolution layers with a kernel size of $s_\ell \times s_\ell$. Thus, processing a $d \times 1 \times 1$ latent by an expand layer results in a $d \times s_\ell \times s_\ell$ output, allowing them to be concatenated with the decoder features (blue "+" in Fig. 2).

cNVAE pros and cons. Our approach resulted in a much smaller latent space compared to NVAE, which allowed us to examine the function of individual latents, decoupled from their spatial influence. However, one drawback of our approach is that it makes the architecture input-size-dependent. Scaling up the input increases the parameter count of cNVAE. This did not pose serious problems in the present work, as we generated ROFL datasets in a relatively small spatial scale of 17×17. However, this dependence on input dimensionality could cause problems when larger datasets are considered. To mitigate this, one possibility is to use depthwise separable convolutions for sampler and expand layers.

Other details. We used the same regular residual cells for both the encoder and the decoder ("r" in Fig. 2). We did not experience memory issues because of the relatively small scale of our experiments. However, rendering the input data at a larger spatial dimension would lead to larger memory usage, which might require using depthwise separable convolutions as in Figure 3b in Vahdat and Kautz [52]. Similar to NVAE, we found that the inclusion of squeeze & excitation [123] helped. Contrary to the suggestion of NVAE [52] and others [69], batch norm [124] destabilized training in our case. Instead, we found that weight normalization [125] and swish activation [126, 127] were both instrumental in training. We used weight normalization without data-dependent initialization. Furthermore, we could obtain good results without the need for spectral regularization, perhaps because our dataset is synthetic and has a much smaller dimensionality compared to real-world datasets such as faces. Finally, we also found that residual Normal parameterization of the approximate posterior was helpful.

Model hyperparameters. The latent space of cNVAE contains a hierarchy of latent groups that are sampled sequentially: "top" latents are sampled first, all the way down to "bottom" latents that are closest to the stimulus space (Fig. 2). Child [68] demonstrated that model depth, in this statistical sense, was responsible for the success of hierarchical VAEs. Here, we increased the depth of cNVAE architecture while ensuring that training remained stable. Thus, we ended up with a model that had 21 hierarchical latent groups: 12 groups operating at the scale of 8×8, 6 groups at the scale of 4×4, and 3 groups at the scale of 2×2 (see Fig. 2 and Fig. 3). Each latent group has 20 latents, which yields a $21 \times 20 = 420$ dimensional latent space. See below for more details. We observed that various aspects of model performance such as reconstruction loss and DCI scores increased with depth, corroborating previous work [68]. We set the initial number of channels to 32 in the first layer of the encoder and doubled the channel size every time we downsample the features spatially. This resulted in a width of 256 in the final layer. Please refer to our code for more information.

9.3 cNVAE vs. NVAE latent dimensionality

As an illustrative example, here we count the number of latent variables in cNVAE and compare them to an otherwise identical but non-compressed NVAE. The total number of latent variables for a hierarchical VAE is given by:

$$D = \sum_{\ell=1}^{L} d_\ell, \tag{14}$$

where d_ℓ is the number of latents in level ℓ, and L is the total number of hierarchical levels. As described above, $d_\ell = d \cdot s_\ell^2$ for NVAE, where d is the number of latents per level and s_ℓ is the spatial scale of level ℓ. In contrast, this number is a constant for the cNVAE: $d_\ell = d$. In this paper, we set $d = 20$ and had the following configuration for the cNVAE:

Table 4: The hierarchical models (cNVAE, cNAE) process information at various spatial scales. These models have a total of 21 latent groups or layers, each containing $d = 20$ latent variables. For hierarchical models, we concatenate the latent representations across all levels of the hierarchy resulting in a single latent vector with dimensionality $D = 420$. In contrast, non-hierarchical models (VAE, AE) do not possess such a multi-scale organization as their latent space contains a single latent group, operating at a single spatial scale. This single latent group contains the entire set of $d = D = 420$ latent variables. See Table 2 and Fig. 2.

Model	Number of latent groups				Number of latent variables per group	Latent space dimensionality
	2×2	4×4	8×8	total		
cNVAE	3	6	12	21	$d = 20$	$D = 21 \times 20 = 420$
VAE	1	–	–	1	$d = 420$	$D = 1 \times 420 = 420$
cNAE	3	6	12	21	$d = 20$	$D = 21 \times 20 = 420$
AE	1	–	–	1	$d = 420$	$D = 1 \times 420 = 420$

- "top" latents: 3 groups operating at the spatial scale of 2×2
- "mid" latents: 6 groups operating at the spatial scale of 4×4
- "bottom" latents: 12 groups operating at the spatial scale of 8×8

This resulted in an overall latent dimensionality of $D = (3 + 6 + 12) \times d = 21 \times 20 = 420$ for the cNVAE. In contrast, an NVAE with a similar number of hierarchical levels would have a dimensionality of $D = (3 \cdot 2^2 + 6 \cdot 4^2 + 12 \cdot 8^2) \times d = 876 \times 20 = 17,520$.

Note that in Fig. 4, we necessarily had to reduce the NVAE depth to roughly match its latent dimensionality to other models. Specifically, the NVAE in Fig. 4 had $2/3/6$ latent groups at top/mid/bottom with $d = 1$. This configuration resulted in latent dimensionality of $D = (2 \cdot 2^2 + 3 \cdot 4^2 + 6 \cdot 8^2) \times 1 = 440$, which is roughly comparable to other models at $D = 420$. See Table 4 for a comparison between cNVAE and the alternative models used in this study.

9.4 Model training details

We generated $750,000$ samples for each ROFL category, with a $600,000/75,000/75,000$ split for train/validation/test datasets. Model performance (reconstruction loss, NELBO) was evaluated using the validation set (Fig. 14 and 15). Similarly, we used the validation set to estimate mutual information between latents, $\boldsymbol{z}$, and ground truth factors, $\boldsymbol{g}$, as shown in Fig. 3. To evaluate the latent code, we used the validation set to train linear regressors to predict $\boldsymbol{g}$ from $\boldsymbol{z}$. We then test the performance using the test set. This includes results presented in Fig. 4 and Fig. 5.

KL annealing and balancing. We annealed the KL term during the first half of the training, which is known to be an effective trick in training VAEs [52, 69, 128, 129]. Additionally, during the annealing period, we employed a KL balancing mechanism which ensures an equal amount of information is encoded in each hierarchical latent group [52, 130, 131].

Gradient norm clipping. During training, we occasionally encountered updates with large gradient norms. Similar to vdvae [68], we used gradient clipping, with an empirically determined clip value of 250. However, we did not skip the updates with large gradients, as we found that the model usually gets stuck and does not recover after skipping an update.

Training hyperparameters. We used batch size of 600, learning rate of 0.002, and trained models for 160 epochs (equivalent to $160k$ steps). Each training session took roughly 24 hours to conclude on a single Quadro RTX 5000 GPU. Similar to NVAE, we used the AdaMax optimizer [132] with a cosine learning rate schedule [133] but without warm restarts.

9.5 Choosing β values for different figures

It is known that different choices of the β value in the VAE loss (Table 2) will result in different model properties [82]. Therefore, while we scanned across many β values spanning two orders of magnitude, we displayed results for certain betas to best demonstrate our points.

To ensure a fair model comparison in the results shown in Figs. 3, 4, 11 and 12, we selected β values that maximized the overall untangling score for each architecture (cNVAE, $\beta = 0.15$; VAE, $\beta = 1.5$). This can be seen in Fig. 5, which presents DCI scores over the entire β spectrum. The top row of Fig. 5 displays informativeness (= untangling) scores, revealing an inverted U shape, consistent with previous work [82]. Different architectures peak at different β values.

For the neuron shown in Fig. 8b, we found that different β values lead to qualitatively similar outcomes, that is, extreme sparsity in cNVAE feature importances, in sharp contrast to that of VAE. We deliberately chose to show a larger $\beta = 5$ for VAE because previous work by Higgins et al. [16] suggested that increasing β values increases neuronal alignment, which we did not observe here. In contrast, even a very small $\beta = 0.01$ for the cNVAE results in a large alignment. This result (paired with other observations) suggests that brain alignment, as we measured it here, emerges due to the architecture rather than from large β values alone—although we do observe some dependence on β, so ultimately a combination of both architecture and loss is important (but mostly architecture).

9.6 Details on latent code evaluation

Each ROFL sample $\boldsymbol{x} \in \mathbb{R}^{2\times N\times N}$ is rendered from a low dimensional ground truth vector $\boldsymbol{g} \in \mathbb{R}^{K}$, where N is the number of pixels (stimulus size) and K is the true underlying dimensionality of the data. In other words, the data lies on a curved K-dimensional manifold embedded in a $2N^2$-dimensional space.

In the present work, we worked with relatively small stimuli with $N = 17$. This choice allowed us to train more models and explore a larger combination of hyperparameters. We chose an odd value for N so that there was a pixel at the exact center, which served as the center of gaze or the point. The true dimensionality (number of generative variables) of different ROFL categories is given in Table 1. For example, it is $K = 11$ for `fixate-1`.

We trained unsupervised models by feeding them raw samples $\boldsymbol{x}$ and asking them to produce a reconstruction $\hat{\boldsymbol{x}}$. The models accomplish this by mapping their inputs to latent codes $\boldsymbol{z}$ during inference (red pathways in Fig. 2), and mapping $\boldsymbol{z}$ to $\hat{\boldsymbol{x}}$ during generation (blue pathways in Fig. 2). Note that for all models we have $\boldsymbol{z} \in \mathbb{R}^{420}$ (see Table 4).

In sum, our approach consists of four steps:

1. Data generation: $\boldsymbol{g} \rightarrow \boxed{\text{ROFL}} \rightarrow \boldsymbol{x}$
2. Unsupervised training: $\boldsymbol{x} \rightarrow \boxed{\text{Model}} \rightarrow \hat{\boldsymbol{x}}$
3. Inference: $\boldsymbol{x} \rightarrow \boxed{\text{Model}} \rightarrow \boldsymbol{z}$
4. Evaluation: $\boldsymbol{z} \leftrightarrow \boldsymbol{g}$?

Within this framework, we investigated the relationships between data generative factors $\boldsymbol{g}$ and model-learned latent codes $\boldsymbol{z}$ by using the DCI metrics, which we discuss in detail below.

9.7 Metrics & Baselines

We trained a variety of unsupervised models (Table 2) and evaluated them based on (1) their ability to represent the generative variables of the ROFL stimuli; and (2) the neural alignment of their representation. This resulted in a total of five different metrics used to evaluate model performance. Below, we described the different metrics used for each.

#1: untangling & disentangling of data generative factors. To evaluate the latent codes, we employed the DCI framework [22] and made use of the fact that ROFL provides ground truth factors for each data sample. DCI is comprised of three different metrics, *Disentanglement*, *Completeness*, and *Informativeness*, which we will discuss in detail below.

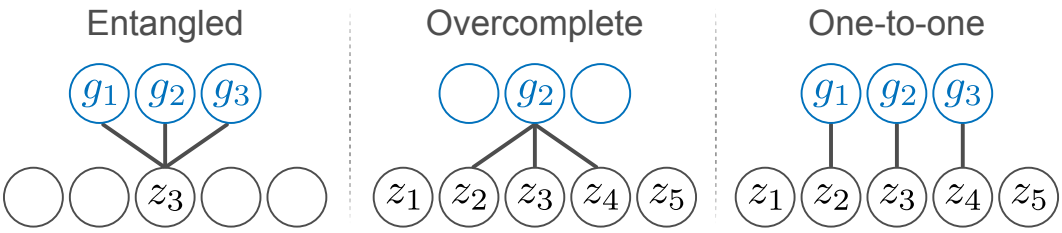

Figure 9: Suppose we train a regression model to predict ground truth factors g_j (blue) from latents z_i (grey). Entanglement (left) refers to a situation in which one latent variable contains information about many generative factors. An overcomplete code (middle) is one in which a single generative factor is predicted from many latent variables. A one-to-one relationship (right) is the ideal outcome.

#2: explaining variability in biological neurons & brain alignment. We used ridge regression to predict macaque MT neuron responses from model latents. To measure the alignment between different latent codes and biological neurons, we used two standard metrics: linear predictive power and brain-alignment score defined in Fig. 8a.

9.7.1 The DCI framework

In this section, we provide more details about the DCI framework [22], and how it was used to evaluate the disentangling performance of our models. The core motivation behind DCI is that a good latent code ideally disentangles as many factors of variation as possible while discarding as little information as possible. In other words, we consider a latent code to be "*good*" as long as:

1. It can predict the variance of data generative factors, and
2. It exhibits a one-to-one relationship with the generative factors (Fig. 9).

To make this idea concrete, the starting point is to train a regressor f_j (e.g., lasso) that predicts a given data generative factor g_j from a latent code $\boldsymbol{z}$. The performance of $\boldsymbol{f}$ in predicting $\boldsymbol{g}$ (measured using e.g., R^2) is referred to as the *Informativeness* score. The notion of *Informativeness*, if measured using a linear regressor, is identical to *Untangling* [20].

As the next step, the weights of $\boldsymbol{f}$ are used to construct a matrix of relative importance R, where R_{ij} denotes the relative importance of z_i in predicting g_j. For example, R can be the magnitude of lasso coefficients. The matrix of relative importance is then used to define two pseudo-probability distributions:

$$P_{ij} = \frac{R_{ij}}{\sum_k R_{ik}} = \text{probability of } z_i \text{ being important for predicting } g_j \tag{15}$$

$$\tilde{P}_{ij} = \frac{R_{ij}}{\sum_k R_{kj}} = \text{probability of } g_j \text{ being predicted by } z_i \tag{16}$$

These distributions are then used to define the following metrics, which complete the DCI trinity:

$$\textit{Disentanglement}: \quad D_i = 1 - H_K(P_{i\cdot}); \quad \text{where} \quad H_K(P_{i\cdot}) = -\sum_j P_{ij} \log_K P_{ij} \tag{17}$$

$$\textit{Completeness}: \quad C_j = 1 - H_D(\tilde{P}_{\cdot j}); \quad \text{where} \quad H_D(\tilde{P}_{\cdot j}) = -\sum_i \tilde{P}_{ij} \log_D \tilde{P}_{ij} \tag{18}$$

Here, K is the number of data generative factors (e.g., $K = 11$ for `fixate-1`, Table 1), and D is the number of latent variables ($D = 420$ for the models considered in our paper, Table 4).

Intuitively, if latent variable z_i contributes to predicting a single ground truth factor, then $D_i = 1$. Conversely, $D_i = 0$ when z_i equally contributes to the prediction of all ground truth factors. If the ground truth factor g_j is being predicted from a single latent variable, then $C_j = 1$ (a complete code).

On the other hand, if all latents contribute to predicting g_j, then $C_j = 0$ (maximally overcomplete). Finally, the overall disentanglement or completeness scores are obtained by averaging across all latents or generative factors.

In the DCI paper, Eastwood and Williams [22] used lasso or random forest as their choice of the regressor $\boldsymbol{f}$. Initially, we experimented with lasso but found that the lasso coefficient had a strong impact on the resulting DCI scores. To mitigate this, we used linear regression instead and estimated the matrix of relative importance using scikit-learns's permutation importance score (sklearn.inspection.permutation_importance), which measures how much performance drops based on the shuffling of a given feature.

9.7.2 Brain alignment score

As depicted in Fig. 8a, the alignment score emphasizes the degree of correspondence between latent variables and neurons. A sparse relationship between latents and neurons suggests a higher alignment, indicating that the model's representation closely mirrors the neuron's representational "form" [16]. This notion of alignment is intimately tied to the "completeness" score discussed above. See Fig. 10.

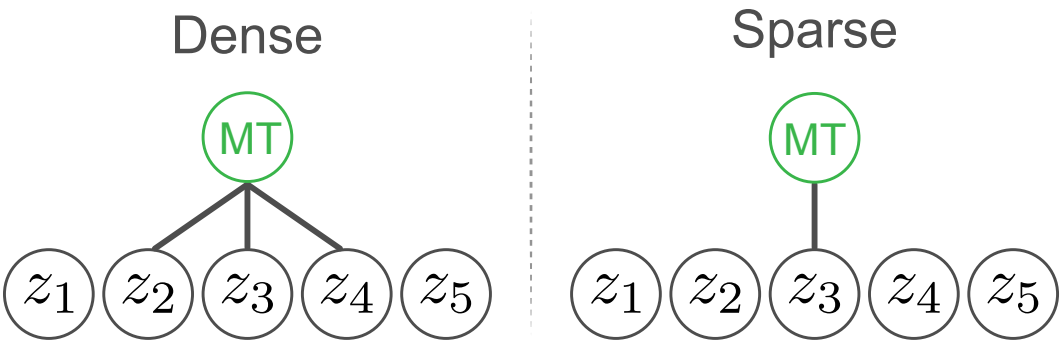

Figure 10: Consider training a regressor to predict MT neuron responses (green) from latent variables z_i (grey). Beyond accurate predictions, we assess the alignment between $\boldsymbol{z}$ and MT based on the sparsity of the resulting latent-to-neuron relationships. One possibility is that many latents contribute to predicting the neuron in a dense fashion. This would suggest that the neuron's function is diffusely represented within the latents, hinting that the representational form of the latents is ultimately different form that of MT, perhaps through rotation or affine transformations. Conversely, when few latents effectively predict MT, it suggests a similar representational form, laying the foundation for our rationale in defining our alignment score (Fig. 8a).

9.7.3 Baselines

In Fig. 4 we compared the cNVAE with alternative models introduced in Table 2. We also included an NVAE with matched number of latents, but necessarily fewer hierarchical levels (see section 9.3 for more details). To demonstrate the difficulty of our untangling task, we included two simple baselines: Raw data and a linear code based on the first 420 dimensions of data principal components (PCA). Both Raw and PCA baselines performed poorly, only weakly untangling self-motion velocity ($\sim 55\%$) and object velocity in x and y directions (47%). This result demonstrates the importance of nonlinear computation in extracting good representations.

10 Additional Results

In this section, we present additional results demonstrating the advantages of hierarchical VAEs versus non-hierarchical ones, which support the results in the main text. All models explored in this section were trained on `fixate-1`. For both cNVAE and VAE, we chose β values that maximized their informativeness (we discuss this choice in section 9.5). Specifically, we set $\beta = 0.15$ for cNVAE and $\beta = 1.5$ for VAE. These same models trained using the respective β values were also used for results presented in Fig. 3 and Fig. 4 in the main text.

10.1 Individual cNVAE latents exhibit greater linear correlation with ground truth factors

In the main text (Fig. 4) we demonstrated that the cNVAE latent representations $\boldsymbol{z}$ contained explicit information about ground truth factors $\boldsymbol{g}$. We achieved this by training linear regression models that map the 420-dimensional latent space to each ground truth factor. Here, we demonstrate that the degree that *individual* latents z_i are (Pearson) correlated with individual factors g_j (Fig. 11). For some ground truth factors, we find that cNVAE learns latents that are almost perfectly correlated with them. For instance, $r = -0.91$ for F_x and $r = 0.87$ for F_y, compared to modest correlation values of $r = 0.34$ and $r = 0.38$ for VAE. The higher linear correlation is consistently observed for cNVAE over VAE. Especially, we note that the VAE does not capture object-related variables at all, consistent with other results in the main paper (Fig. 3 and Fig. 4).

For the cNVAE, the indices for the selected highly correlated latents are as follows: $[296, 248, 393, 284, 368, 92, 206, 105, 338, 60, 63]$, ordered left-to-right in the same manner as in Fig. 11. Here, an index of 0 corresponds to the very top latent operating at the spatial scale of 2×2, and an index of 419 corresponds to the very bottom one at scale 8×8 (see Fig. 2). By considering the hierarchical grouping of these latents (see dashed lines in Fig. 3) we see that the majority of the highly correlated latents operate at the spatial scale of 8, with the exception of object-related ones corresponding to X_{obj}, Z_{obj}, $V_{obj,y}$, and $V_{obj,z}$ which operate at 4×4.

In conclusion, Fig. 11 revealed that cNVAE learns individual latents that are highly correlated with individual ground truth factors, which provides complementary insights as to why cNVAE is so successful in untangling (Fig. 4). However, we note that a high correlation with individual latents is not the best way to capture the overall functional organization, which is better reflected for example in the mutual information heatmap of Fig. 3.

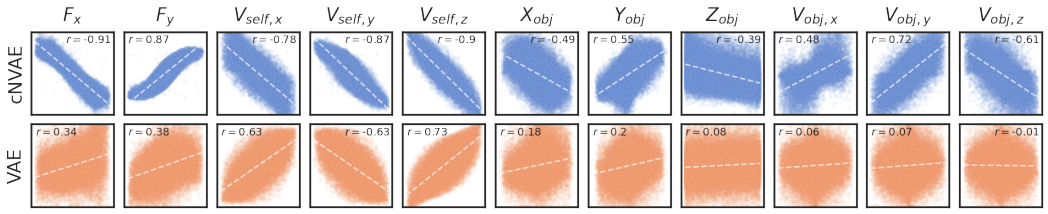

Figure 11: For each ground truth factor and model, we identified a single latent variable that displayed the highest Pearson correlation with that variable. The ground truth factors exhibited greater linearity within the latent space of cNVAE compared to VAE. Scatter points correspond to data samples from the test set. x-axis, the value of ground truth factors; y-axis, the value of latent variables.

10.2 Generated samples: non-hierarchical VAE completely neglects objects

If a generative model has truly learned the underlying structure of a dataset, then it should be able to generate fake samples that look like its training data. Here, we examine generated samples from both the cNVAE and VAE models, as illustrated in Fig. 12. For comparison, we also include five samples randomly drawn from the `fixate-1` dataset that these models were trained on (Fig. 12, first row). The cNVAE samples appear indistinguishable from true samples; however, VAE fails to capture and generate objects, corroborating our previous results (see Fig. 3, Fig. 4, and Fig. 11). This striking disparity between cNVAE and VAE indicates that a hierarchically organized latent space is necessary for effectively modeling datasets that contain multiple interacting spatial scales.

10.3 Latent traversal reveals multi-scale understanding in cNVAE

One desired property of VAE latent representations is that continuous interpolation along a latent dimension results in interpretable effects on the corresponding reconstruction. Here, we provide visual examples of latent traversal in cNVAE, focusing on a latent dimension that effectively captures Y_{obj} (Figure 13). These examples demonstrate intriguing interactions between object velocity patterns and self-motion within the scene. Notably, in some examples such as rows 0 and 6, relocating the object from the bottom to the top of the scene induces a significant change in its linear velocity along the y-axis. This phenomenon arises due to the influence of self-motion, as seen in the background.

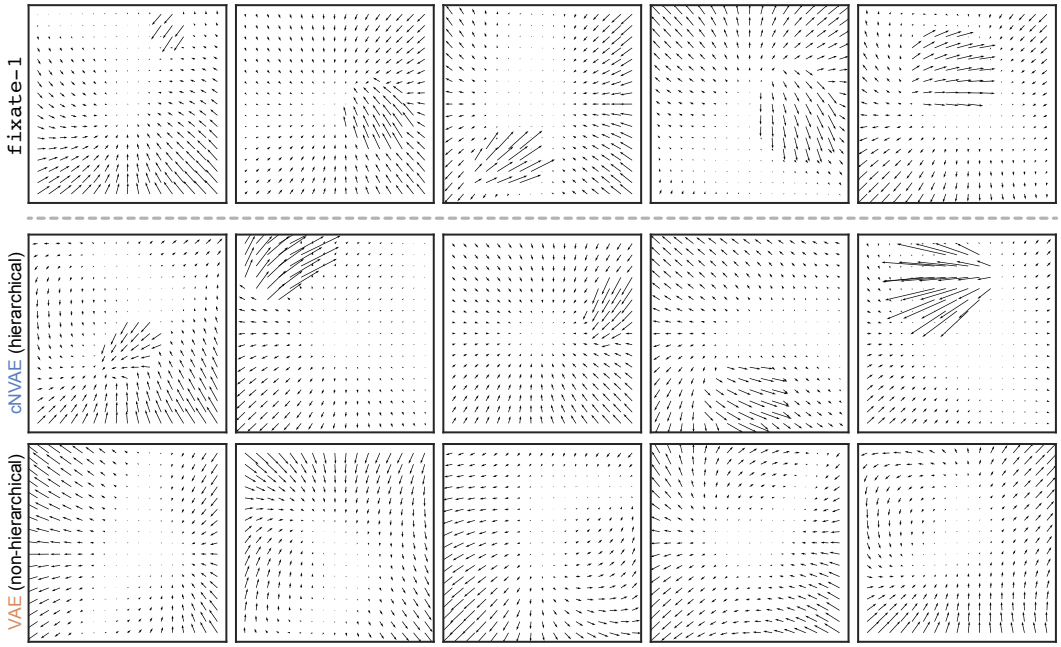

Figure 12: The first row shows random samples drawn from ROFL `fixate-1` category. The second and third rows are samples generated from cNVAE and VAE models that were trained on `fixate-1`. Generated samples from cNVAE closely resemble real data; whereas, VAE ignores objects. Samples were generated at a temperature of $t = 1$ without cherry-picking.

These illustrations collectively showcase that flow vectors in the visual scene are shaped both by local variables, such as the object's physical location, and their interactions with global-influencing variables like self-motion. By comprehending data across scales, cNVAE is able to capture such hierarchical and multi-scale properties inherent in naturalistic optic flow patterns. In contrast, non-hierarchical VAEs exhibit a complete failure in this regard.

10.4 All models achieve good reconstruction performance

All models considered in this paper are trained based (in part or fully) on reconstruction loss, corresponding to how well they could reproduce a presented stimulus. Here we report the models' reconstruction performance, to ensure that they are all able to achieve a reasonable performance. Figure 14 illustrates how the reconstruction loss (EPEPD, endpoint error per dim) and negative evidence lower bound (NELBO) depend on different values of β and model types. Both hierarchical and non-hierarchical VAE models perform well, with cNVAE exhibiting slightly superior performance for $\beta < 0.8$.

10.5 Untangling and reconstruction loss are strongly anti-correlated for cNVAE but not VAE

In this work, we demonstrated that a virtue of hierarchical VAEs is that they make information about ground truth factors easily (linearly) accessible in their latent representations. We were able to quantify this relationship because we used a simulation (ROFL) where ground truth factors were available. However, such a condition is not necessarily the case in real-world datasets that typically lack such direct knowledge [22].

To address this, we examined the relationship between informativeness and reconstruction loss, as well as NELBO, depicted in Fig. 15, to test whether there is a relationship between them. Strikingly, we observe a significant and strong negative linear relationship between reconstruction loss and informativeness in cNVAE ($r = -0.91, p = 8.9 \times 10^{-7}$, $t-$test). Conversely, no such correlation is observed for VAE ($r = 0.10, p = 0.72$, $t-$test). Notably, this relationship is absent for both models when comparing informativeness with NELBO (Fig. 15). These findings suggest that reconstruction

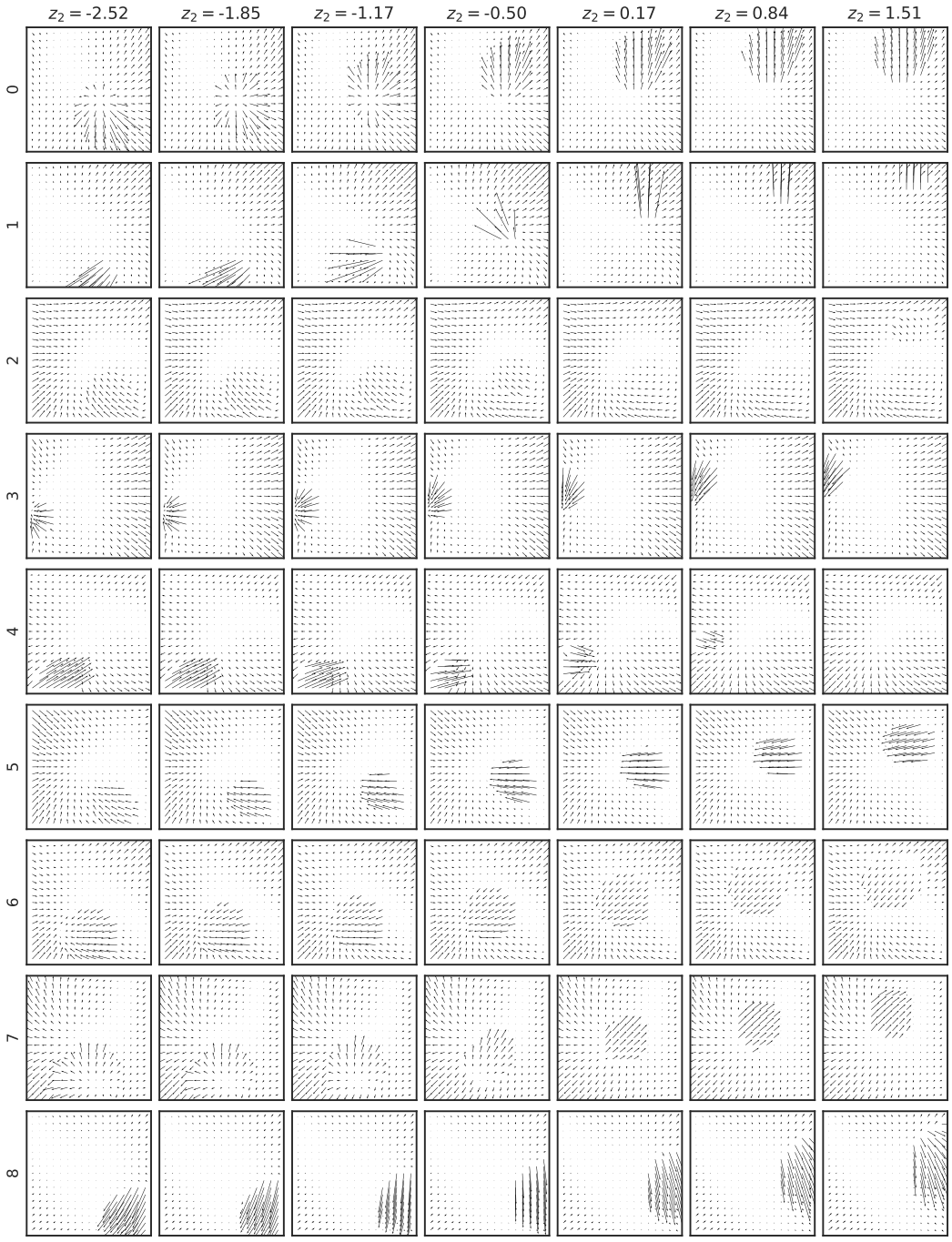

Figure 13: Latent traversal performed using a latent variable that effectively captures Y_{obj}. This variable belongs to a latent group operating at the spatial scale of 2×2 (see Figs. 2 and 3; table 4).

loss—exclusively in the case of cNVAE—can serve as a viable proxy for informativeness, even in the absence of ground truth factors in real-world datasets.

10.6 Brain alignment requires more than just linear regression performance

In the field of neuroscience, linear regression R^2 scores have commonly been employed to measure alignment between biological and artificial neural networks [30, 31, 34, 36]. However, we found

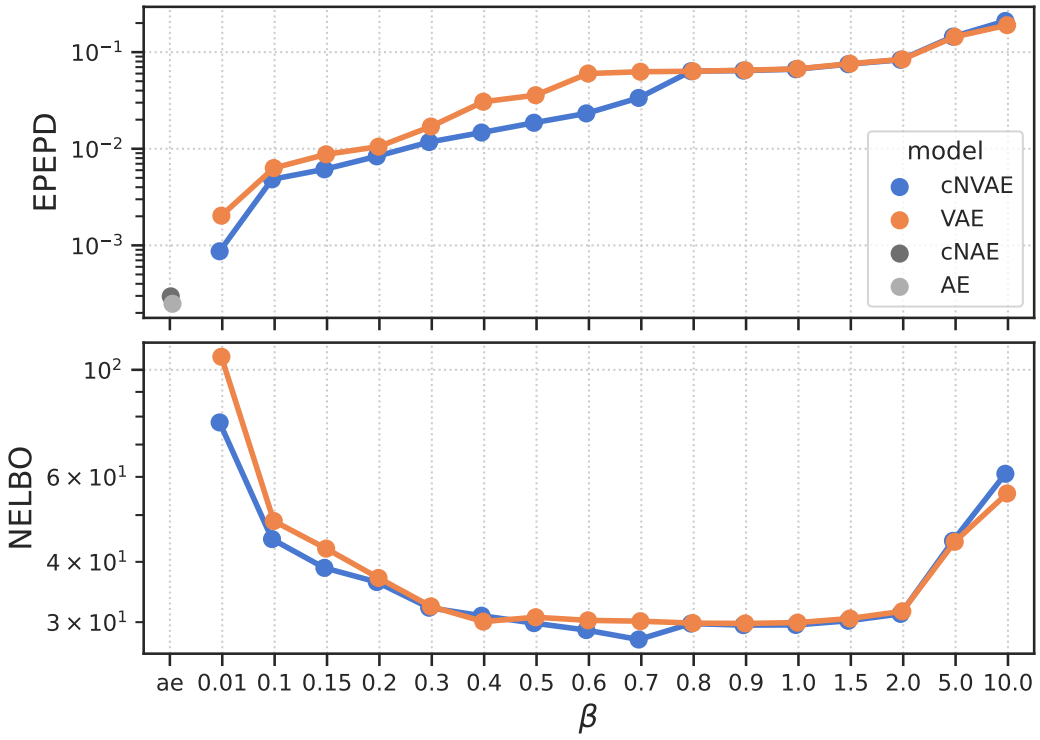

Figure 14: Both models demonstrate robust reconstruction performance, with cNVAE exhibiting a slight improvement. EPEPD, endpoint error per dim (lower is better). NELBO, negative evidence lower bound (lower is better).

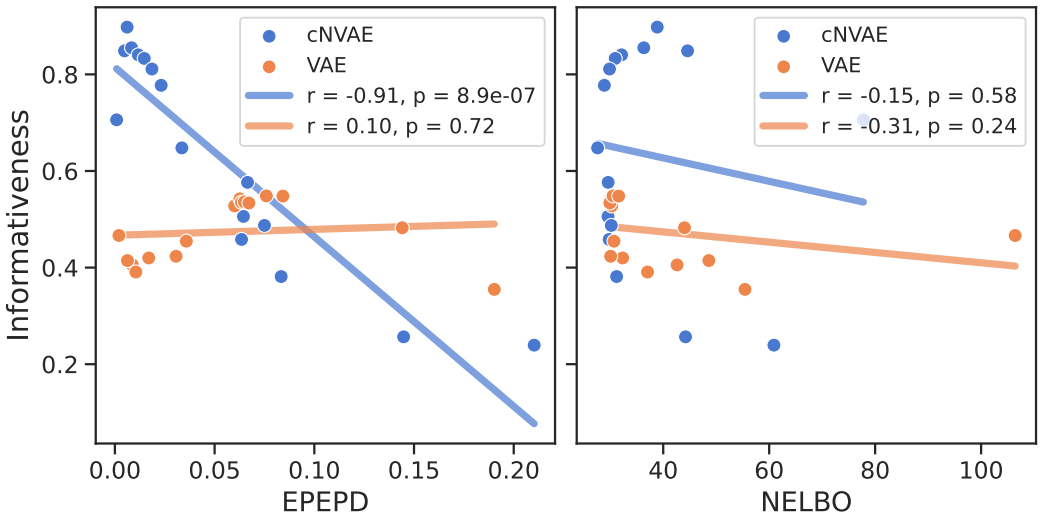

Figure 15: Strong anti-correlation between reconstruction loss and informativeness (i.e., untangling) observed only for cNVAE. Each scatter point corresponds to a model trained with a different β value. EPEPD, endpoint error per dim (lower is better). NELBO, negative evidence lower bound (lower is better).

that relying solely on regression performance provided a weak signal for model selection, which is consistent with recent work [36–38, 109, 122]. In contrast, our alignment score defined in Fig. 8a provided a much stronger signal for differentiating between models (compare Fig. 7 and Fig. 8c).

To investigate this further, we conducted statistical tests, comparing pairs of models based on either their performance in explaining the variance of MT neurons or their brain-alignment scores. The results are presented in Table 5. We observed that ridge regression performance was not particularly effective in distinguishing between models. We measured the effect sizes between distributions of $N = 141$ neurons, measuring how well ridge regression performance (Fig. 7) or alignment scores (Fig. 8c) discriminated between models. We found that the effect sizes were substantially larger for the alignment scores (Fig. 16). These findings suggest that while model performance is necessary, it is not sufficient for accurately measuring alignment between artificial and biological representations. In conclusion, refined and more comprehensive approaches are required to properly capture brain alignment [109–112].

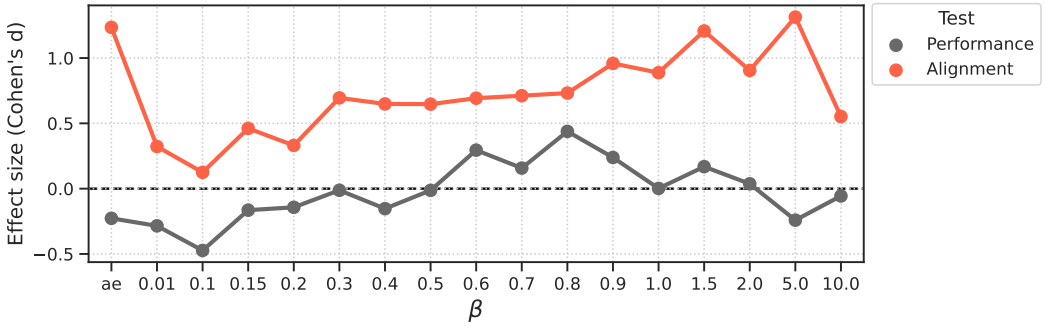

Figure 16: Effect sizes are shown for the two different approaches in model comparisons: using ridge regression *Performance* (related to Fig. 7), versus, *Alignment* scores (related to Fig. 8c). Alignment demonstrates a substantially larger effect size, leading to a more pronounced differentiation among the models. Positive values indicate a larger value for hierarchical models (i.e., cNVAE > VAE or cNAE > AE). For different β values, model *Performance* difference fluctuates between positive and negative values. In contrast, the difference in model *Alignment* scores is consistently positive, indicating a greater brain alignment for hierarchical models.

10.7 Dissecting cross-validated model performance on neural data

One measure of neural alignment that we consider here used unsupervised latents to predict experimentally recorded neural responses (via linear regression). The neurons in this dataset had long presentations of optic flow stimuli, but only a fraction of neurons (105 out of 141 neurons) had multiple repeats of a short stimulus sequence. Because these repeats facilitated a much clearer measure of model performance, we used these where available, following previous work [34]. However, due to the short stimulus sequence (5 sec; see Fig. 6) used to estimate performance in the neurons with stimulus repeats, here we perform additional tests to verify that our conclusions hold using a more rigorous cross-validation procedure.

Specifically, here we used the long stimulus sequence to both train model parameters (80% of data) and determine hyperparameters (including regularization and optimal latencies) using a separate validation set (20%). We then predicted model performance on the repeats. This is in contrast to our approach in the main text, which used the repeat data itself to optimize hyperparameters, but as a result risked overfitting on the short stimulus sequence. Indeed, we found that independent determination of hyperparameters (from the validation set) resulted in decreased model performance on the test set (Table 6) for both cNVAE and VAE models, although with comparable effect sizes. As a result, our conclusions of overlapping performance between cNVAE and VAE with a slightly better performance of the cNVAE were robust.

β	Performance (Figure 7)			Alignment scores (Figure 8c)		
	effect size	$p-$value	significant?	effect size	$p-$value	significant?
ae	-0.23	0.01	✓	1.2	8e-29	✓
0.01	-0.28	0.002	✓	0.32	0.0004	✓
0.1	-0.47	2e-07	✓	0.12	0.2	✗
0.15	-0.16	0.07	✗	0.46	4e-07	✓
0.2	-0.14	0.1	✗	0.33	0.0003	✓
0.3	-0.012	0.9	✗	0.69	4e-13	✓
0.4	-0.15	0.09	✗	0.65	7e-12	✓
0.5	-0.013	0.9	✗	0.65	7e-12	✓
0.6	0.29	0.001	✓	0.69	4e-13	✓
0.7	0.16	0.08	✗	0.71	1e-13	✓
0.8	0.44	1e-06	✓	0.73	4e-14	✓
0.9	0.24	0.008	✓	0.96	1e-20	✓
1.0	0.0016	1	✗	0.89	1e-18	✓
1.5	0.17	0.07	✗	1.2	3e-28	✓
2.0	0.037	0.7	✗	0.91	3e-19	✓
5.0	-0.24	0.008	✓	1.3	7e-31	✓
10.0	-0.056	0.6	✗	0.55	3e-09	✓

Table 5: Hierarchical architectures are significantly more aligned with MT neurons compared to non-hierarchical ones. Paired $t-$tests were performed, and $p-$values were adjusted for multiple comparisons using the Benjamini-Hochberg correction method [134]. Effect sizes were estimated using Cohen's d [135], where positive values indicate cNVAE > VAE (or cNAE > AE, first row).

Table 6: Model performance on the subset of neurons for which we had repeat data ($N = 105$). Here, we used the repeat data as a held-out test set. We found that model performances remain statistically indistinguishable, providing additional support for our main results.

Model	Performance, R $(\mu \pm se;\ N = 105)$			
	$\beta = 0.5$	$\beta = 0.8$	$\beta = 1$	$\beta = 5$
cNVAE	$.452 \pm .025$	$.479 \pm .025$	$.486 \pm .024$	$.462 \pm .025$
VAE	$.461 \pm .027$	$.401 \pm .029$	$.466 \pm .028$	$.461 \pm .027$
cNAE	$.460 \pm .026$			
AE	$.412 \pm .031$			

11 Additional Discussion

11.1 Visual-only cues and unsupervised learning are sufficient for untangling optic flow causes

It is well known that an observer's self-motion through an environment can be estimated using optic flow patterns alone [77], although it has been proposed that the brain also uses a combination of visual and non-visual cues [136], such as efference copies of motor commands [137], vestibular inputs

[138], and proprioception [139]. Indeed, using optic flow alone can be complicated by rotational eye movements and independently moving objects [48, 140, 141]. How can the brain estimate the motion of an object when self-motion is non-trivially affecting the entire flow field [142, 143], including that of the object itself? This is a critical motion processing problem [144], manifested in diverse ecological scenarios such as prey-predator dynamics, crossing the street, or playing soccer. Are retinal visual-only cues sufficient to extract information about different self-motion components, as well as object motion [145]?

Unlike the approach of Mineault et al. [34], our work is based on the premise that retinal-only signals contain sufficient information about their ground truth causes and that these factors can be extracted using unsupervised learning alone. In contrast, Mineault et al. [34] trained their DorsalNet by providing dense supervision on self-motion components, arguing that such supervised training is biologically plausible since these signals are approximately available to the observer from vestibular inputs and other such extra-retinal sources.

Our results (Fig. 4) provide proof-of-concept that it is possible to untangle ground truth causes of retinal optic flow in a completely unsupervised manner, using the raw optic flow patterns alone. Comparing the performance of alternative models revealed two essential ingredients for the success of our approach: (1) an appropriate loss function—inference [1]; and, (2) a properly designed architecture—hierarchical, like the sensory cortex [3].

11.2 Comparing cNVAE to NVAE, its non-compressed counterpart

In Fig. 4, we investigated the performance of the original non-compressed NVAE with an approximately matched latent dimensionality (440, slightly larger than 420 for cNVAE: more details are provided in section 9.3). We found that although NVAE did better than non-hierarchical VAE, it still underperformed cNVAE. This is because trying to match the latent dimensionality of NVAE with cNVAE necessitates reducing the number of hierarchical latent groups in the NVAE since most of its convolutional latents capture spatial dimensions rather than abstract features like cNVAE. From previous work [68], we know that the "stochastic depth" of hierarchical VAEs is the key reason behind their effectiveness; therefore, it was expected that reduced depth would hurt an NVAE with a matched number of latents.

It is worth noting that cNVAE and NVAE had a balanced performance across all ground truth factors (Fig. 4), in that they captured both global (self-motion-related) and local (object-related) aspects well. In contrast, non-hierarchical VAE completely ignored object-related variables and focused solely on the global scale determined by fixation point and self-motion. This suggests that hierarchical architecture is crucial for a more comprehensive understanding of multi-scale datasets such as ROFL.

11.3 Disentanglement is in the eyes of the beholder

The conventional definition of disentanglement in machine learning relies on relating the latent representation to predefined generative factors, a choice that is often not uniquely determined, and is often guided by heuristics [18]. For example, in the present work we chose to represent velocities in Cartesian coordinates, rather than polar coordinates, and judged the degree of disentanglement based on the relationship of the latents to these factors. Notably, these coordinate systems are related by a nonlinear transform, and latents linearly related to one coordinate system will then be less linearly related to the other.

Throughout our experiments, we observed one or more cNVAE latent variables that exhibited equally high mutual information with all components of self-motion velocity in Cartesian coordinates (e.g., latent dimensions 154 and 122 in Fig. 3). When we translated the generative factors into polar coordinates and looked at their relationship with these latents, we found that these latents were solely encoding the magnitude of self-motion, irrespective of its direction (Fig. 17). Thus, these seemingly highly entangled latent variables would have achieved almost perfect disentanglement scores had we employed polar coordinates instead of Cartesian coordinates to represent the ground truth velocities ($d_i \approx 0.38 \rightarrow d_i \approx 0.81$).

Our findings shed light on the fact that disentanglement cannot be considered an inherent property of objective reality; rather, it is significantly influenced by our a priori determination of independent factors of variation within the data. Given the inherent complexity and richness of natural datasets, it

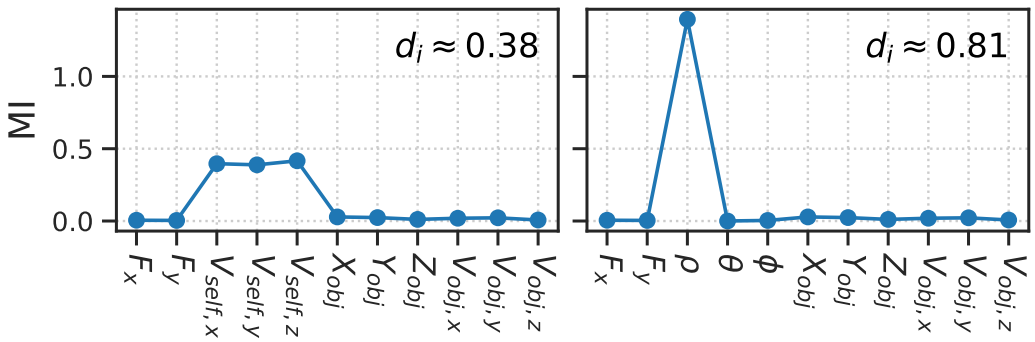

Figure 17: This latent variable cares about the magnitude of self-motion only. Its disentanglement score depends strongly on what we choose a priori as our independent factors of variation in data.

becomes challenging to unequivocally determine the true factors of variation a priori, pointing to the need for a more systematic approach.

When it comes to interpreting brain functions, perhaps considering behavioral relevance of generative factors could guide our choices. For example, an animal directing its own movements does not pick grid coordinates, but rather chooses direction and speed, possibly making a polar representation more "behaviorally relevant" to representations within its brain.

Another promising approach lies in focusing on the symmetry transformations inherent in the world [18, 146]. By seeking representations that respect the underlying geometrical and group-theoretical structures of the world [147, 148], and comparing them in principled ways [109–111], we may uncover new perspectives that provide valuable insights beyond the conventional and subjective notions of disentanglement.

12 Additional Background & Related Work

The deep connections between VAEs and neuroscience is explored in a review by Marino [17], but work looking at the synergy between neuroscience and VAEs has thus far been limited to the following recent papers.

Storrs et al. [15] trained PixelVAE [149] models on synthesized images of glossy surfaces and found that the model spontaneously learned to cluster images according to underlying physical factors like material reflectance and illumination, despite receiving no explicit supervision. Most notably, PixelVAE mimicked human perceptual judgment of gloss in that it made errors that were similar to those made by humans.

Higgins et al. [16] trained β-VAE [82] models on face images and evaluated the models on predicting responses of individual inferotemporal neurons from Macaque face patch. They found that β-VAE discovers individual latents that are aligned with IT neurons and that this alignment linearly increases with increased beta values.

Csikor et al. [65] investigated the properties of representations and inference in a biologically inspired hierarchical VAE called Topdown VAE that is capable of reproducing some key properties of representations emerging in V1 and V2 of the visual cortex. The authors showed that architectures that feature top-down influences in their recognition model give rise to a richer representation, such that specific information that is not present in mean activations becomes linearly decodable from posterior correlations.

Finally, Miliotou et al. [66] learned mapping from the latent space of a hierarchical VAE to fMRI voxel space. This allowed the use of the trained decoder to reconstruct images directly from brain responses. Importantly, they perform ablation experiments and find hierarchy is an essential component of their models.

13 Appendix: ROFL derivations and detailed description

ROFL aims to generate synthetic optical flow patterns that closely resemble those arising from ecologically relevant types of motion. This entails the consideration of both self-motion and object-motion components. Specifically, the self-motion component encompasses various subtypes such as translational and rotational motions induced by eye or head movements. Furthermore, it is crucial to have control over the relative prominence of these two types of motion. Depending on the specific configuration, the resulting flow pattern can exhibit an *object-dominated* characteristic when an abundance of objects is incorporated, or conversely, a *self-motion-dominated* nature if fewer or no objects are present. By carefully manipulating these factors, ROFL enables the generation of optical flow patterns that faithfully capture the dynamics of real-world scenarios involving an interplay between self-motion and object motion.

13.1 Setup

Self-motion is meaningful if the observer is moving relative to a static background (i.e., the environment). We consider a solid background wall at a distance $Z = 1$ from the observer which extends to infinity in the $X - Y$ plane. Define the following coordinate frames

1. **Fixed coordinates**, represented with capital letters: (X, Y, Z).
2. **Observer's coordinates**, represented with lowercase letters: (x, y, z).

Both coordinate systems are centered around the observer. The difference is that $\hat{Z}$ is always perpendicular to the background plane, while $\hat{z}$ is defined such that it is parallel to the fixation point.

All points in the space are initially expressed in the fixed coordinates. For example, any given point like $\vec{P} = (X, Y, Z)$ has its coordinates defined in the fixed system. The same is true for fixation point, $\vec{F} = (X_0, Y_0, Z)$, and the observer's position $\vec{O} = (0, 0, 0)$, and so on. See Fig. 18.

13.1.1 Relating the two coordinate systems

How do we go from the fixed coordinates to the observer's coordinates? We need to find a rotation matrix that maps the $\hat{Z}$ direction to the direction defined by the fixation point. By definition, this will determine $\hat{z}$ and all the other directions in the observer's coordinates. First, let us express $\vec{F}$ using its polar coordinates:

$$\vec{F} = (X_0, Y_0, Z) \equiv \left(\|\vec{F}\|, \Theta_0, \Phi_0 \right), \tag{19}$$

where $\|\vec{F}\| = \sqrt{X_0^2 + Y_0^2 + Z^2}$, $\cos\Theta_0 = Z/\|\vec{F}\|$, and $\tan\Phi_0 = Y_0/X_0$. It turns out what we are looking for is a rotation by an amount Θ_0 about a unit vector $\hat{u}$ that depends only on Φ_0. That is,

$$\hat{u}(\Phi_0) = \begin{pmatrix} -\sin\Phi_0 \\ \cos\Phi_0 \\ 0 \end{pmatrix}, \tag{20}$$

and the rotation matrix is given by $R_{\hat{u}}(\Theta_0) = I + (\sin\Theta_0)\, L(\hat{u}) + (1 - \cos\Theta_0)\, L(\hat{u})^2$.

Here, $L(\vec{\omega})$ is the skew-symmetric matrix representation of a vector $\vec{\omega}$, that is

$$L(\vec{\omega}) \coloneqq \vec{\omega} \cdot \vec{L} = \omega_X L_X + \omega_Y L_Y + \omega_Z L_Z, \tag{21}$$

where L_i are the familiar basis matrices for $\mathfrak{so}(3)$, the Lie algebra of $SO(3)$:

$$L_X = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix}, \quad L_Y = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix}, \quad L_Z = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{22}$$

The rotation matrix in its full form is expressed as

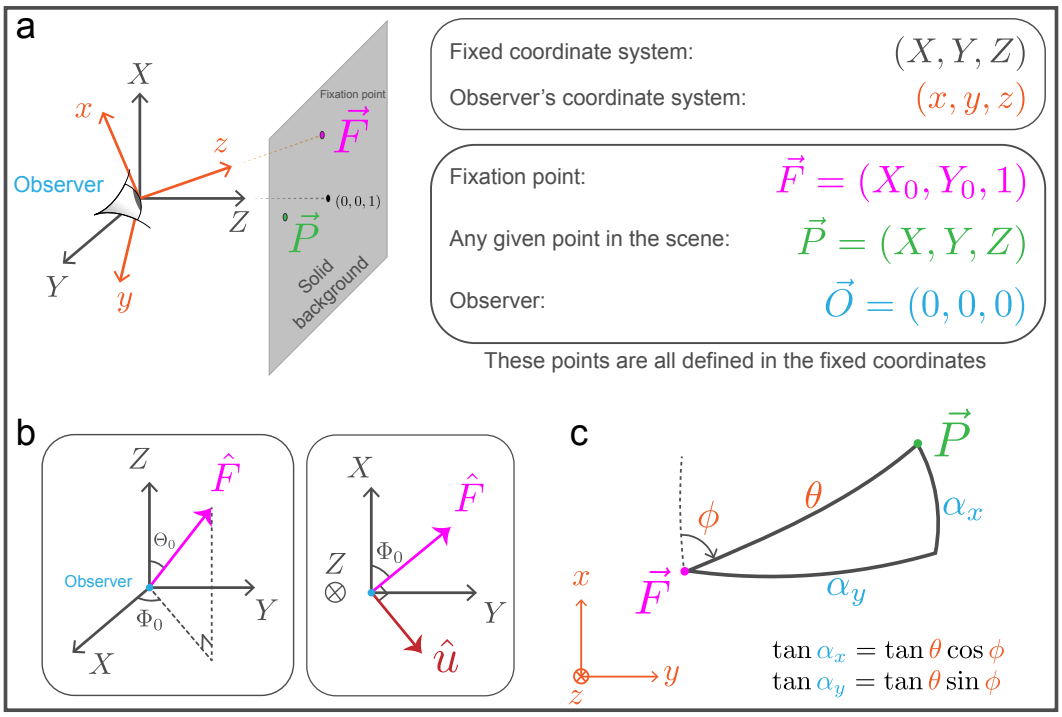

Figure 18: Setup. **(a)** Background and coordinate systems. **(b)** The rotation of the fixed coordinates $\left(\hat{X}, \hat{Y}, \hat{Z}\right)$ onto the observer-centric coordinates $(\hat{x}, \hat{y}, \hat{z})$ can be accomplished by applying a rotation of angle Θ_0 around the unit vector $\hat{u}$ as defined in Equation 20. Note that by definition, $\hat{z}$ is always parallel to the fixation point. See the text for more details. **(c)** Perspective from the observer's point of view. The retinotopic coordinates of point $\vec{P}$ are denoted by α_x and α_y, measured in radians.

$$R \coloneqq R_{\hat{u}}(\Theta_0) = \begin{pmatrix} s^2 + c^2 \cos\Theta_0 & -cs\,[1 - \cos\Theta_0] & c \sin\Theta_0 \\ -cs\,[1 - \cos\Theta_0] & c^2 + s^2 \cos\Theta_0 & s \sin\Theta_0 \\ -c \sin\Theta_0 & -s \sin\Theta_0 & \cos\Theta_0 \end{pmatrix}, \tag{23}$$

where we have defined $c \coloneqq \cos\Phi_0$ and $s \coloneqq \sin\Phi_0$.

Any point $\vec{P} = (X, Y, Z)$ defined in the fixed coordinate system can be mapped to a point $\vec{p} = R^T \vec{P}$ with components $\vec{p} = (x, y, z)$ given in the observer's system. For example, it can be shown that $R^T \vec{F}$ is parallel to $\hat{z}$, which is how we defined R in the first place. See Fig. 18.

In conclusion, we have a rotation matrix R that is fully determined as a function of the fixation point $\vec{F}$. Applying R rotates the fixed coordinate system onto the observer's system

$$\begin{aligned} \hat{x} &\coloneqq R\hat{X}, \\ \hat{y} &\coloneqq R\hat{Y}, \\ \hat{z} &\coloneqq R\hat{Z}. \end{aligned} \tag{24}$$

This way, the observer's coordinates (x, y, z) are obtained ensuring that the fixation point is always parallel to $\hat{z}$.

13.1.2 Adding motion

Consider a point $\vec{P} = (X, Y, Z)$. In the previous section, we showed that the rotation matrix in Equation 23 can be used to relate $\vec{P}$ to its observer-centric coordinates

$$\vec{P} = R\vec{p}. \tag{25}$$

Differentiate both sides to get

$$\vec{V} \coloneqq \frac{d\vec{P}}{dt} = \frac{dR}{dt}\vec{p} + R\frac{d\vec{p}}{dt}, \tag{26}$$

where $\vec{V}$ is the velocity of the point relative to the observer, with components measured in the fixed coordinate system.

13.1.3 Derivative of the rotation matrix

Here we will derive a closed-form expression for dR/dt in terms of the observer's angular velocity. This is important because it essentially captures the effects of eye movement.

Let us use $\vec{\omega} = (\omega_X, \omega_Y, \omega_Z)$ to denote the angular velocity of observer's coordinates, with components expressed in fixed coordinates. The time derivative of R is given as follows

$$\frac{dR}{dt} = L(\vec{\omega})R, \tag{27}$$

where $L(\vec{\omega})$ is given in Equation 21. Taken together, equations 3, 4, 8, and 9 yield

$$\boxed{\begin{aligned}\frac{d\vec{p}}{dt} &= R^T\vec{V} - \left[R^T L(\vec{\omega})R\right]\vec{p} \\ &= R^T\vec{V} - \left[R^T \begin{pmatrix} 0 & -\omega_Z & \omega_Y \\ \omega_Z & 0 & -\omega_X \\ \omega_Y & \omega_X & 0 \end{pmatrix} R\right]\vec{p}\end{aligned}} \tag{28}$$

This equation allows relating the velocity of a moving point expressed in fixed coordinates, $\vec{V}$, to its observer-centric equivalent, $\vec{v} \coloneqq d\vec{p}/dt$. Importantly, Equation 28 is a general relationship valid for any combination of velocities and rotations.

13.1.4 Motion perceived by the observer

Before discussing specific applications, we need to solve one remaining piece of the puzzle: what is $\vec{v}$, and how is it related to the motion perceived by the observer? To address this, we need to map the retinotopic coordinates of the observer (measured in radians), to points in the environment (measured in the observer-centric coordinates).

Denote the retinotopic coordinates using (α_x, α_y). The goal is to relate each (α_x, α_y) pair to a point in the world $\vec{p}$ with coordinates (x, y, z). It can be shown that

$$\begin{aligned}\tan\alpha_x &= x/z = \tan\theta\cos\phi, \\ \tan\alpha_y &= y/z = \tan\theta\sin\phi,\end{aligned} \tag{29}$$

where θ and ϕ are the polar coordinates of $\vec{p}$. See Fig. 18c for an illustration.

Differentiate both sides of Equation 29 and rearrange terms to obtain the time derivative of (α_x, α_y) as a function of $\vec{v} = (\dot{x}, \dot{y}, \dot{z})$ and $\vec{p}$. That is,

$$\boxed{\begin{aligned}\dot{\alpha_x} &= \frac{\dot{x}z - \dot{z}x}{z^2 + x^2} \\ \dot{\alpha_y} &= \frac{\dot{y}z - \dot{z}y}{z^2 + y^2}\end{aligned}} \tag{30}$$

Thus, we obtained a relationship between retinotopic velocity $(\dot{\alpha_x}, \dot{\alpha_y})$ (measured in $radians/s$) and velocity of objects in the world (measured in m/s). In conclusion, equations 28 and 30 enable us to compute the observer's perceived retinotopic velocity for any type of motion.

13.2 Example: maintaining fixation

We are now ready to apply the formalism developed in the preceding section to create optical flow patterns in realistic settings. In this section, we will consider a specific example: self-motion while maintaining fixation on a background point. This is an important and ecologically relevant type of motion, as we often tend to maintain fixation while navigating the environment—think about this next time you are moving around. In fact, one can argue that fixation occurs more naturally during behaviors such as locomotion, as opposed to typical vision experiments where the animals are stationary and passively viewing a screen.

What does it mean to maintain fixation while moving? It means the eyes rotate to cancel out the velocity of the fixation point. Assume the observer is moving with velocity $\vec{V}_{self}$. Equivalently, we can imagine the observer is stationary while the world is moving with velocity $\vec{V} = -\vec{V}_{self}$ (this is the same $\vec{V}$ that appears in Equation 28).

Any eye movement can be represented mathematically as a rotation of the observer's coordinates with respect to the fixed coordinates. Use $\vec{\omega}$ to denote the angular velocity of the rotation. We can determine $\vec{\omega}$ by requiring that the rotation cancels out the velocity of the fixation point. To this aim, we first compute the normal component of $\vec{V}$ with respect to the fixation point $\vec{F}$ as follows

$$\vec{V}_{\perp} \coloneqq \vec{V} - \frac{\vec{V} \cdot \vec{F}}{\|\vec{F}\|^2}\vec{F}. \tag{31}$$

Use the above equation to compute the angular velocity

$$\vec{\omega} = \frac{\vec{F} \times \vec{V}_{\perp}}{\|\vec{F}\|^2}. \tag{32}$$

Plug Equation 32 in Equation 28 to find $\vec{v}$, and use Equation 30 to find $(\dot{\alpha_x}, \dot{\alpha_y})$. This way, we just calculated the motion perceived by the observer during self-motion while maintaining fixation.

13.3 The algorithm

Start by choosing a fixation point $\vec{F} = (X_0, Y_0, Z)$, field-of-view extent (fov), and retinal resolution (res). For example, $fov = 45°$ and $res = 5.625°$. Use this to create a mesh grid for (α_x, α_y). The grid covers $(-fov, +fov)$ degrees in each dimension and has shape $2 * fov/res + 1$, or 17×17 in our example. Use Equation 29 to compute (θ, ϕ) pair for each point on the (α_x, α_y) grid.

Because the simulation is scale-invariant, we can always fix the distance between the observer and the background wall. Assume $Z = cte. = 1$, and use equations 23 and 25 to find $\vec{p} = (x, y, z)$ per grid point. Note that we assume infinite resolution for the points in the environment.

So far, we partitioned the retina into a grid and found the observer-centric coordinates of a point in the real world that falls on each grid point. Now we can sample a random self-motion velocity $\vec{V}_{self}$ and use the results described in the previous sections to find $(\dot{\alpha_x}, \dot{\alpha_y})$. This provides us with the $2D$ retinal velocity vector on the grid and concludes the algorithm for the simple case of `fixate-0` (i.e., no objects).

Please refer to our code for additional details, including instructions on how to incorporate independently moving objects into the scene.

13.3.1 Valid fixation points

The observer-wall distance is constrained to be a positive value. That is, the observer is always situated on the left side of the wall ($Z > 0$ for all points; Fig. 18). This constraint, combined with a given fov value and Equation 23 results in a theoretical bound on which fixation points are valid. To derive this bound, we start by considering the $\hat{Z}$ component of Equation 25

$$(R\vec{p})_3 = R_{31}x + R_{32}y + R_{33}z = Z > 0. \tag{33}$$

Divide both sides by z and use Equation 29 to get

$$R_{31}\tan\alpha + R_{32}\tan\beta + R_{33} > 0. \tag{34}$$

Plug the matrix entries from Equation 23 and rearrange to get

$$\begin{aligned} &\sin\Theta_0(\cos\Phi_0\tan\alpha + \sin\Phi_0\tan\beta) < \cos\Theta_0, \\ \Rightarrow \quad &X_0\tan\alpha + Y_0\tan\beta < Z. \end{aligned} \tag{35}$$

For a given fov value, we have $-\tan(fov) \leq \tan\alpha, \tan\beta \leq \tan(fov)$. Assuming $Z = 1$, we find that the (X_0, Y_0) coordinates of a valid fixation point should satisfy the following inequality

$$|X_0| + |Y_0| < \frac{1}{\tan(fov)}. \tag{36}$$

In other words, the L_1 norm of (X_0, Y_0) should be less than $1/\tan(fov)$. For example, when $fov = 45°$, the upper bound is 1. This result can be used to choose an appropriate fov: smaller values allow a larger phase space to explore for fixation points.

13.4 Conclusions

We introduced Retinal Optic Flow Learning (ROFL), a novel synthetic data framework to simulate realistic optical flow patterns. The main result is given in Equation 28, which shows the simulation has the potential to generalize to various other types of motion. For instance, the observer might engage in smooth pursuit of a moving object instead of maintaining fixation, a scenario that can be modeled simply by adjusting the choice of $\vec{\omega}$ in Equation 28. This flexibility enables us to generate a diverse repertoire of realistic optical flow patterns, which can be used in training and evaluating unsupervised models, among other applications.

We believe that our framework will prove valuable not only to neuroscientists studying motion perception but also to machine learning researchers interested in disentangled representation learning. By bridging these domains, we aim to foster interdisciplinary collaboration and advance our understanding of both biological and artificial information processing systems.