Self-Augmented Learning of Differentiable Object Models for Compositional Interpretation of Complex Scenes

Antoni Nowinowski
Poznan University of Technology
antoni.nowinowski@cs.put.poznan.pl

Krzysztof Krawiec
Poznan University of Technology
krzysztof.krawiec@cs.put.poznan.pl

Abstract

This study builds on the architecture of the Disentangler of Visual Priors (DVP), a type of autoencoder that learns to interpret scenes by decomposing the perceived objects into independent visual aspects of shape, size, orientation, and color appearance. These aspects are expressed as latent parameters which control a differentiable renderer that performs image reconstruction, so that the model can be trained end-to-end with gradient using reconstruction loss. In this study, we extend the original DVP so that it can handle multiple objects in a scene and propose effective training strategies. To address the challenges of optimizing in presence of a differentiable renderer, we exploit the interpretability of the latent by using the decoder to generate self-augmented training examples and devising alternative training modes that rely on loss functions defined not only in the image space, but also in the latent space. This significantly facilitates training, which is otherwise challenging due to the presence of extensive plateaus in the image-space reconstruction loss. We compare our approach with two baselines (MONet and LIVE) on a new benchmark which subsumes the previously proposed Multi-dSprites and demonstrate its superiority in terms of reconstruction quality and capacity to decompose overlapping objects. We also analyze the gradients induced by the considered loss functions, explain how they impact the efficacy of training, and discuss the limitations of differentiable rendering in autoencoders and the ways in which they can be addressed.

1 Introduction

Building correct models of reality requires identifying the causes of observed phenomena and disentangling their interactions. In computer vision (CV), the characteristics of a pixel is determined by a multitude of *aspects* (shapes, positions, and orientations of objects, their color characteristics, lighting, etc.), which interact in complex ways governed by the physical laws. The majority of contemporary CV approaches assume that the generic, connectionist substrate of deep learning (DL) is sufficient to learn these causal chains and form the necessary representations. While multiple successful deployments of CV systems seem to support this claim, most of them do not rely on a principled model of image formation, and, as a result, struggle to truly understand scene content. Even if good enough most of the time, such models ultimately fail, often in spectacular ways and in unexpected 'failure modes', showing puzzling susceptibility to confounding factors/variables, spurious correlations, and outliers (see, e.g., [ZBL+18, DJL21]).

Among recent works that attempt to address this deficiency (reviewed in Sec. 4), the DVP proposed in [KN24] is a DL architecture that disentangles the visual aspects of perceived objects and then combines them to reconstruct the scene. It comprises a convolutional encoder, which delineates an

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: NeurIPS 2025 Workshop on Space in Vision, Language, and Embodied AI.

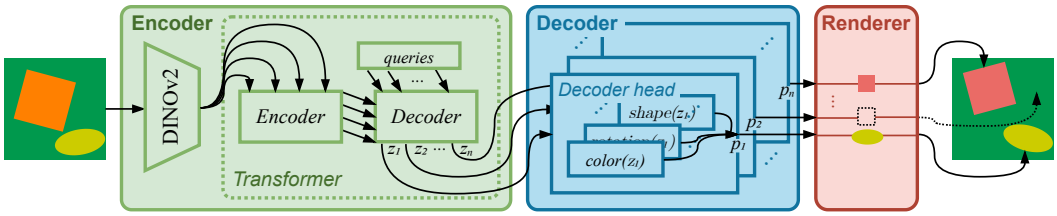

Figure 1: DVP+ reconstructs the scene by structurally modeling each object and rendering it.

object from the scene and captures its characteristics in terms of color, shape, and other aspects, with a decoder that attempts to reconstruct the scene by rendering a possibly similar object. Thanks to differentiable rendering, DVP can be trained end-to-end with a gradient on raw image data.

In this study, we propose an approach based on similar principles and bring in the following contributions: we (i) propose and analyze alternative training modes that use the renderer as scene generator, addressing the main challenge of differentiable rendering, i.e. the abundance of plateaus in the reconstruction loss, which leads to difficulties in training (Sec. 3); (ii) design a significantly generalized architecture, dubbed DVP+, which can handle multiple objects (Fig. 1, Sec. 2); (iii) devise a method for acquiring shape prototypes that does not require training of the entire model and streamlines its preparation (Sec. 3); (iv) demonstrate these advantages with experiments that involve a new benchmark (Sec. 5) and explain the differences between training modes with ablations and analysis of gradients.

2 DVP+: The Disentangler of Visual Priors

DVP+ attempts to reconstruct the observed image in a nontrivial fashion. In conventional autoencoders, this is achieved by forcing the model to encode the image in a low-dimensional latent representation. DVP+ constraints the reconstruction process by explicitly implementing image formation, starting from delineating individual objects, encoding their shapes and colors, and finally rendering them on a raster. By realizing all stages in a differentiable fashion, it sustains the capacity of end-to-end training with gradient-based algorithms. In the following, we detail the core components of DVP+ (Fig. 1): the **encoder** responsible for image analysis and delineation of individual objects, the **decoder head** that predicts and disentangles the visual aspects characterizing each object, and the **renderer** that paints the reconstructed objects on an output raster canvas.

The **encoder** is based on the Detection Transformer (DETR) blueprint [CMS+20], which we choose due to its capacity of parsing visual scenes into separate objects. It comprises a perception module followed by an encoder-decoder transformer (EDT). The former is DINOv2 [ODM+23], a pre-trained vision transformer (ViT) acting as a feature extractor. The ViT is queried on the input image x and produces spatially localized latent vectors, which are then augmented with a positional encoding, flattened into a sequence of tokens and passed to the EDT. In the EDT, the intermediate results are processed through cross-attention layers of the decoder, which uses independent 'object queries' that learn to index the objects in the image. Each object query is a trainable vector that evokes from EDT a latent vector $z_i \in Z$ describing the i-th *object candidate* in the input image; see [CMS+20] for more details on the DETR architecture. As the ordering of z_is is arbitrary, they together form a set $\{z_i\}$ that is subject to subsequent processing.

Next, each object candidate z_i is processed in parallel by the **decoder head**, which comprises seven MLPs, each mapping the z_i to interpretable *object parameters* that control separate aspects of the process of image reconstruction/formation:

- Color: $Z \to \mathbb{R}^3$ (represented as an RGB tuple)
- Translation: $Z \to \mathbb{R}^2$
- Scaling: $Z \to \mathbb{R}$
- Rotation angle: $Z \to \mathbb{R}^2$ (represented as the lengths of opponent side and adjacent side)
- Shape: $Z \to [0, 1]^m$ (where m is the size of the bank of shape prototypes)
- Confidence: $Z \to \mathbb{R}$ (probability of the candidate being a part of the scene)

- Background color: $Z \to \mathbb{R}^3$

The shape predictor uses the softmax activation in its m-dimensional final layer, and the resulting vector is used to softly index a bank of m *shape prototypes*, each stored as a vector of Elliptic Fourier Descriptors (EFD, [KG82]). The elements of the EFD are real and imaginary parts of complex coefficients that form the spectrum which is mapped with inverse Fourier Transform to object's contour represented as complex 'time series'; see details in Sec. B. In principle, rather than learning to choose a shape from a trainable bank of shape prototypes, one could predict shape directly, i.e. map z_i to the vector of EFDs. Indeed, the authors of DVP [KN24] did that, but within our framework it fared much worse than prototype-based method. More importantly, learning shape prototypes allows reasoning about the scene in qualitative terms and categorization of observed entities (cf. Sec. 1), which opens the door to alternative training methods, detailed in the final part of Sec. 3.

The $(m + 10)$-dimensional vector p_i of parameters of the ith object candidate forms the input to the **renderer**. The renderer is a parameter-less (non-trainable) algorithm that 'paints' the candidates on a common canvas, taking into account blending of overlapping fragments of objects (translucence). The background color is determined by the average of z_is predicted for all candidates i.e. $b(\overline{z_i})$. The raster image produced in this way forms the final output of DVP+ and is differentiable w.r.t the scene parameters, enabling the gradient of to be backpropagated through the rendering process. From the numerous differentiable rendering methods proposed in recent years [LCLL19, RRN+20, LLMRK20, LHK+20, CLG+21, JSRV22], we opted for PyTorch3D [RRN+20] for its clean programming interface and computational efficiency. This rendering method implements the traditional computer graphics route of approximating the objects with a mesh[1] of triangles, which are then rasterized.

3 Training modes

We propose to train DVP+ in modes that vary in the primary space (used as the starting point for sampling examples) and the loss function.

Primary space. The natural mode of DVP+'s training is the conventional autoassociative learning for image reconstruction, employed also in [KN24]. In this case, the source of data is a training set of images X.

We propose an alternative approach that relies on *self-augmentation* by using the renderer as an *image generator*, in the spirit of numerous studies that used decoders for that purpose, including the seminal variational autoencoders [KW14] and adversarial autoencoders [MSJ+16]. The common challenge in this scenario is that the generator (here: the renderer) requires sampling from a latent distribution, which is in general unknown; in our context, it is the distribution of objects' parameters p, denoted by P in the following. To address this issue, we devise a sampling technique that uses a partially trained *preliminary model* as the source of P. First, an example x is sampled from the training set X. The preliminary model (more specifically, its encoder and decoder) is then queried on x to yield the estimated scene parameters $\{\hat{p}_i\}$. This becomes our sample from P, which can be then passed to the renderer in order to generate an image. The preliminary model can be obtained, for instance, by training the architecture in the conventional way, i.e. for image reconstruction. As we will demonstrate in Sec. 5, this sampling technique proves very effective even if the preliminary model was trained only for a few epochs.

Loss function. We consider two loss functions, both implementing a kind of reconstruction error. The first one is the **image-space loss function** $L_x(x, \hat{x})$ defined as pixel-wise error, used commonly in autoencoders, and also in [KN24] (we use MAE in experiments). As an alternative, we propose a **parameter-space loss function**, which operates on encoder's output. It calculates the reconstruction error of the parameters in $\{\hat{p}_i\}$ predicted by the encoder with respect to the true parameters in $\{p_i\}$ – provided, for instance, by the generator introduced above. As the number of predicted objects, i.e. the size of the $\{\hat{p}_i\}$ set, can be different from the number of actual objects n, and the ordering of objects can vary too, we engage the Hungarian algorithm to efficiently determine the optimal assignment of the predicted candidates in $\{\hat{p}_i\}$ to the actual objects in $\{p_i\}$. The algorithm

[1] While PyTorch3D offers high-level interfaces for several classes of objects, including meshes, point clouds, volumetric representations, and NeRFs, we required greater flexibility. Therefore, we implemented parts of the rendering pipeline ourselves using PyTorch3D's performant low-level functions.

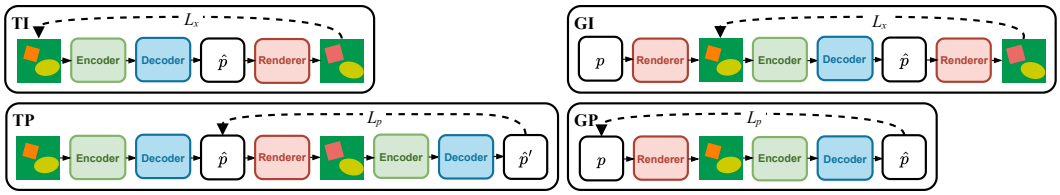

Figure 2: The four training modes of DVP+. Top: image-based loss L_x, bottom: parameter-space loss L_p. Left: images as the source of training data, right: parameters as the source of training data.

determines the optimal assignment that minimizes the per-object matching cost function $C(p_i, \hat{p}_j)$, defined as weighted sum of square differences of selected parameters of object i and candidate j:

$$C(p_i, \hat{p}_j) = \|p_i.t - \hat{p}_j.t\|_2 + 0.1\,\|p_i.c - \hat{p}_j.c\|_2 + 0.01\,\|p_i.conf - \hat{p}_j.conf\|_2 \tag{1}$$

where t, c, and $conf$ denote, respectively, the translation vector, the RGB triple of object's color, and the confidence, i.e. the respective parts of the vector of properties (Sec. 2). Once the optimal assignment has been determined, each of the n matched object-candidate pairs is subject to the actual parameter-space loss function $L_p(p, \hat{p})$, which measures the discrepancy over all parameters of matched objects, using weighted distance functions (tuned empirically to individual parameters in preliminary experiments):

$$\begin{aligned}
L_p(p, \hat{p}) = {} & \|p.b - \hat{p}.b\|_2 && \textit{(background color)} \\
& + \sum\nolimits_{i=1}^{n} \Big(\log(\hat{p}_i.conf) && \textit{(BCE on confidence)} \\
& \qquad + 5\,\|p_i.t - \hat{p}_i.t\|_2 && \textit{(translation)} \\
& \qquad + \|p_i.c - \hat{p}_i.c\|_2 && \textit{(color)} \\
& \qquad + \|p_i.s - \hat{p}_i.s\|_2 && \textit{(scale)} \\
& \qquad + \|p_i.sh - \hat{p}_i.sh\|_2 && \textit{(shape)} \\
& \qquad + 0.05(1 - \cos(p_i.a - \hat{p}_i.a) \cdot sym(p_i.sh))^2 \Big) && \textit{(rotation)} \\
& + \sum\nolimits_{j=n+1}^{m} -\log(1 - \hat{p}_j.conf) && \textit{(BCE on confidence)}
\end{aligned} \tag{2}$$

where b, s, sh and a are, respectively, the properties of background color, scale, shape and rotation angle. sym is the number of symmetries of the shape prototype, calculated from its EFD (see Sec. C for details). The total loss is the sum of L_ps for all n matched pairs.

L_p induces an error surface (over the space of model's parameters) that is different from L_x, which may facilitate training. For instance, when the predicted location of an object in a reconstructed image diverges strongly from its actual location in the input image (e.g. to the extent that they do not even overlap), L_x will respond with high reconstruction error, but its gradient with respect to model parameters will be often close to zero. However, the gradient of L_p can be still high, because it depends, among others, on the actual differences in objects' coordinates. The usefulness of both loss functions will be assessed empirically in Sec. 5.

Training modes. The choice of the above primary space (**T**raining set or **G**enerator) and loss function (**I**mage space vs. **P**arameter space) leads to four *training modes* of DVP+ (Fig. 2). Let f, g, and r denote respectively the encoder, the decoder, and the renderer.

TI $\min_{x \sim X} L_x(x, r(g(f(x))))$: The typical way of training autoencoders: the model is queried on an image x from the training set X, responds with an image $r(g(f(x))$, and is trained by minimizing the image-space reconstruction loss L_x.

GP $\min_{p \sim P} L_p(p, g(f(r(p))))$[2]: This mode is dual with respect to TI: a sample p of object parameters is drawn from a given distribution P (e.g. the image generator described above) and rendered. The rendered image is fed into encoder f and decoder g (i.e., $g \circ f$), resulting in predicted parameters of candidates, $g(f(r(p)))$, which are compared to p with parameter-space loss L_p.

[2]For the sake of brevity we abuse the notation by using L_p to denote both the per-object loss (Eq. 2) as well the loss summed over all object-candidate pairs.

GI $\min_{p \sim P} L_x(r(p), r(g(f(r(p)))))$: Like GP, but both the sampled parameters p and the predicted parameters $g(f(r(p)))$ are passed through r, and the resulting renderings are compared using the image-space loss function L_x.

TP $\min_{x \sim X} L_p(g(f(x)), g(f(r(g(f(x))))))$: Like TI, but both the input image x and the reconstructed one $r(g(f(x)))$ are fed into $g \circ f$, and the resulting predictions are compared using the parameter-space reconstruction error L_p.

While the first two modes (TI and GP) are rather obvious, GI and TP map the arguments of the loss function to the other representation space. As argued earlier, we expect this to provide additional informative guidance for the training process.

Learning of shape prototypes. As signaled earlier, simultaneous optimization of all scene parameters with respect to L_x poses difficulties due to interdependence of their impact on the rendered scene, causing gradients to point to 'misleading' directions, especially when the predicted scene significantly diverges from the target one. As we demonstrate in Sec. 5, this happens to be particularly true for shape, as it is strongly entangled with scale and rotation. Indeed, in preliminary experimenting we found out that inaccurate prediction of rotation hinders the emergence of object shapes, and vice versa: learning how to rotate candidates requires having a reasonably accurate model of object's shape.

To address this challenge in DVP+, we use clustering to discover shape prototypes from the training data. Given a sample of training images X, for each $x \in X$ we start with obtaining an initial estimate p of scene parameters using a preliminary model. This model, obtained from just a few training epochs using a bank of randomized initial shapes (e.g., random hexagons), provides a rough but sufficient starting point for the optimization process to converge[3]. Then, we repeat the following steps. (1) We optimize ps using the image-space loss function, i.e. $\min_p L_x(x, r(p))$, to obtain locally-optimal parameterizations p^*. (2) We gather p^*s obtained in this way for all $x \in X$ and perform k-medoids clustering in the image space[4], in order to capture the typical shapes present in X and discard the outliers. (3) For the medoid x' of each resulting cluster X', we retrieve the corresponding p' and replace with it the ps corresponding to the elements of X'. After repeating these three steps a number of times, we obtain a stable set of shapes that are representative for X, which is then injected into the model as prototype shapes. We found this method to be much more efficient and effective than the one used in DVP, and use it in all experiments reported in this paper.

4 Related work

We focus on reviewing past works related to the central topic of this study, i.e. the challenges originating in the characteristics of the loss function. Methods involving differentiable rendering (DR) frequently face loss function landscapes that simultaneously feature extensive plateaus and are very rugged at places. This issue is present not only when using DR for model training [RGLM21] (especially in early stages of the training) but also in direct optimization, when the goal is to find the combination of scene parameters that minimize the adopted objective function [LLMRK20, MZX+22]. The main reason is that, especially at the beginning of training/optimization, the rendered scene may be completely different from the actual one, with the produced objects failing to overlap with the actual ones. This manifests in low and vanishing gradients and often leads to reaching poor-quality local minima.

In past studies, this has been addressed in several ways. In the Im2Vec approach [RGLM21], the predictions are rendered as image pyramids, with each subsequent level being smaller by a factor 2. The loss is an aggregation of losses at each level, the gradient from the low-resolution levels proving important while processing images with significant dissimilarities. Optimization-based techniques that use explicit primitives, such as DiffVG [LLMRK20] and Gaussian Splatting (GS) [KKLD23], often start with a large number of primitives (object candidates). This increases the likelihood of some of those candidates overlapping with the target object to be captured, providing a better starting point for optimization. The GS authors further refine the initialization strategy by employing a

[3]The quality of the initial shapes is not essential here. If the colors and positions of rendered objects roughly correspond to those in the input image, then the optimization process will most likely converge to a global minimum (see Fig. 4).

[4]We use the MSE between the equally scaled and centered renderings of shapes as the distance metric for clustering. The optimal number of clusters, k, is determined using the silhouette score [Rou87].

Table 1: Reconstruction quality of DVP+ trained with simple training modes. Best results in bold.

Method	MAE ↓	MSE ↓	SSIM ↑	IoU ↑	ARI ↑
DVP+TI	0.1115	0.0404	0.8154	0.0870	0.0
DVP+GP	**0.0637**	**0.0107**	**0.8540**	**0.8452**	**0.8235**
DVP+GI	0.4964	0.3287	0.0202	0.0	0.0
DVP+TP	0.4895	0.3216	0.4224	0.0653	0.0838

conventional algorithm (Structure from Motion) to derive the initial parameters of the object candidates. To address the related problem where multiple object candidates might redundantly reconstruct a single object, the LIVE method [MZX+22] introduces new object candidates sequentially. This also leads to reconstructing the target with fewer primitives, making the reconstruction more interpretable. In contrast, volumetric approaches such as Neural Radiance Fields (NeRF) [MST+20] model scenes with continuous volumetric scene functions. This formulation, grounded in volume rendering physics, inherently promotes a smoother optimization landscape compared to discrete primitive-based methods, which simplify the optimization process but complicate further processing of the reconstructed scene.

In contrast, we cope with challenging loss functions by engaging the renderer-decoder as scene generator and learning in 'dual' modes presented in Sec.3. In a broader context, DVP+ and some of the approaches mentioned above, by explicitly modeling visual aspects of objects and rendering them for reconstruction, subscribe to the seminal paradigm of *vision as inverse graphics* [BTHR78].

5 Results

We focus on comparing the training modes of DVP+ presented in Sec. 3, i.e. TI, GP, GI, and TP. All configurations of DVP+ use DINOv2 [ODM+23] as the pretrained perception model, which remains frozen during training.

While many of similar approaches have been in the past evaluated on the Multi-dSprites dataset [BMW+19], we found it lacking in low image dimensions (64 x 64). This inclined us to devise a higher-resolution benchmark, dubbed MDS-HR. The other important difference is that MDS-HR does not feature object occlusion. Using the procedures described there, we generated a training and testing set with similar to Multi-dSprites characteristic, comprising, respectively, $55\,000$ and 5000 examples, each being a 128 x 128 image featuring from 1 to 4 objects randomly distributed in the image, representing 3 unique shapes (Fig. 3). Details on MDS-HR and access to the dataset are provided in Sec. A; Sec. F presents our software implementation and provides open access to it.

Simple training regimes. We start with the 'monolithic' training regimes that optimize from scratch *all* trainable parameters with the loss function defined in a given mode. Each training process comprised 100 epochs of the Adam optimizer [KB15] with the learning rate 0.0001 (see Sec.D for more details). Training of a single DVP+ model lasted approximately 10 hours on a workstation equipped with RTX 3090, Intel Core i7-11700KF CPU and 32 GB of RAM (see Sec. G for details on the hardware architecture).

The generator introduced at the beginning of Sec. 3 and used in GP and GI offers an unfair advantage for DVP+, as it exposes the learner to examples from outside the training set – possibly including those from the test set. Therefore, for GP and GI, we sample only the ps that correspond to actual examples from the training set, i.e. $p \sim \{p : r(p) \in X\}$.

Table 1 reports the test-set reconstruction accuracy in terms of Mean Square Error (MSE), Structural Similarity Measure (SSIM, [WBSS04]), Intersection over Union (IoU), and Adjusted Rand Index (ARI, [HA85]). MSE and SSIM are calculated directly from pixels' RGB values (scaled to the $[0, 1]$ interval). IoU and ARI are more structural and require the rendered pixels to be unambiguously assigned to an object or to the background (for IoU) or to each distinct object (for ARI). Therefore, we force the models to render scenes as binary masks with white objects on a black background.

GP clearly fares best, which corroborates that training by optimizing parameter-space loss L_p is advantageous compared to optimizing the conventional image-space reconstruction loss L_x (TI). While TI still achieves adequate results on pixel-based metrics, its performance on the structural

Table 2: Reconstruction accuracy for particular training regimes and the baseline methods.

Method	MAE ↓	MSE ↓	SSIM ↑	IoU ↑	ARI ↑
DVP+GP	0.0637	0.0107	0.8540	0.8452	0.8235
DVP+TI+	0.0444	0.0050	0.9148	0.9341	0.2275
DVP+TI-TP	0.0482	0.0068	0.8878	0.8884	0.8693
DVP+TI-TP-OptP	0.0023	0.0002	0.9947	0.9831	0.9780
DVP+TI-TP-OptZ	0.0015	**0.0001**	**0.9961**	**0.9927**	**0.9820**
MONet	0.0126	0.0013	0.9676	0.9312	0.9137
LIVE	0.0098	0.0010	0.9850	0.6767	0.6740
Opt-Iter	**0.0012**	0.0002	0.9941	0.9625	0.9414

metrics is substantially worse. In fact, the TI model has learned to reconstruct only the background color, ignoring the objects. A key issue arises early in the training: the initial object candidates produced by the model exhibit little or no resemblance to the ground truth objects. The image-space loss L_x between such dissimilar images does not provide any relevant information (as discussed in Sec. 3), causing the model to end up in a local minimum achievable via reconstructing only the background (see Sec. E for visualizations). The more sophisticated training modes (GI and TP) fail severely, producing almost constant outputs, regardless of the input. This occurs because their loss functions lack a component enforcing reconstruction of the original input data.

Combined training regimes. In the next step, we attempt to improve on the models identified in Table 1. While DVP+GP is clearly superior, it has the disadvantage of requiring ground truth scene parameter (sampling of p). To mitigate this dependency, we appoint DVP+TI as the basis for further investigations. To address the absence of informative gradient in TI, we initialize the weights (and the weights of all models used in the following) with those of DVP+GP from an early (4th) training epoch, so that the scene renderings are good enough for L_x to provide meaningful guidance. We then train this **TI+** configuration with L_x, improving on MAE, MSE and IoU (Table 2). However, ARI is notably low, primarily due to the model reconstructing single objects with multiple candidates. ARI is the only metric here that is sensitive to this undesirable behavior, which is not explicitly penalized in training, and thus models may tend to exercise it, as reconstructing an object with multiple candidates/shapes is often easier than with a single candidate.

To tackle this challenge, we combine TI and TP into **TI-TP**, where training consists in optimizing $L_x + L_p$, i.e. optimizing all parameters of the model with the gradient $\nabla L_x(x, r(g(f(x)))) + \nabla L_p(g(f(x)), g(f(r(g(f(x))))))$ for $x \in X$.[5] While TI-TP turns out to deteriorate most metrics compared to TI+, it substantially outperforms DVP+GP on ARI – a valuable outcome, as ARI reflects how well one-to-one relation between target objects and object candidates is captured.

Given the superiority of TI-TP on ARI, we adopt it as the basis for the -Opt configurations, in which the predictions made by the encoder f and decoder g of TI-TP serve as starting points for a gradient-based optimization procedure, which tunes the scene parameters individually for each scene DVP+ is queried on. We consider two variants that vary in the variables that are subject to optimization: in **TI-TP-OptP**, we optimize p with $\nabla_p r(p)$, while in **TI-TP-OptZ**, we optimize z with $\nabla_z r(g(z))$. In both cases, we allow 100 iterations of the Adam optimizer [KB15]. These hybrids of DL and optimization achieve best results across all metrics. Interestingly, following the gradient in the latent space Z proves slightly more effective than optimization in the interpretable space of objects' parameters.

To put these results in the context of prior work, Table 2 presents also the performance of MONet [BMW+19], LIVE [MZX+22] (Sec. 4), and Opt-Iter, a purely optimization-based algorithm that hybridizes LIVE with our shape representation.[6] Opt-Iter iteratively identifies the image region that is most divergent from the current scene approximation, initializes a new object candidate at that location (sampling its color from the region), and then optimizes all candidates introduced so far to minimize L_x. The cycle repeats until a predetermined number of objects is reached. See Sec. I for

[5] We verified that these functions vary in similar ranges, so their expected impact on training is comparable.

[6] We do not include DVP [KN24] in the comparison, as it is by design incapable of handling multiple objects.

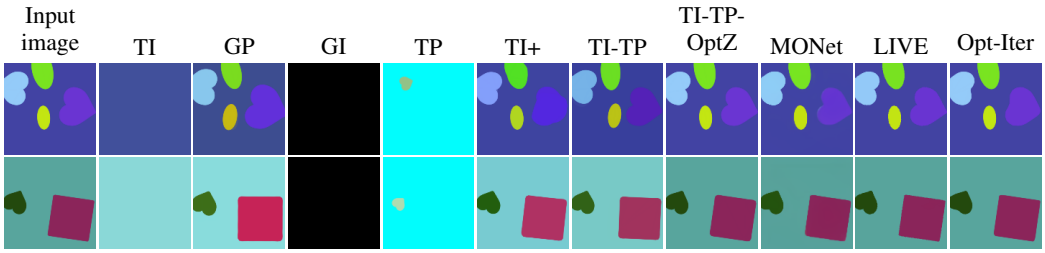

Figure 3: Reconstruction results for selected images from the testing set.

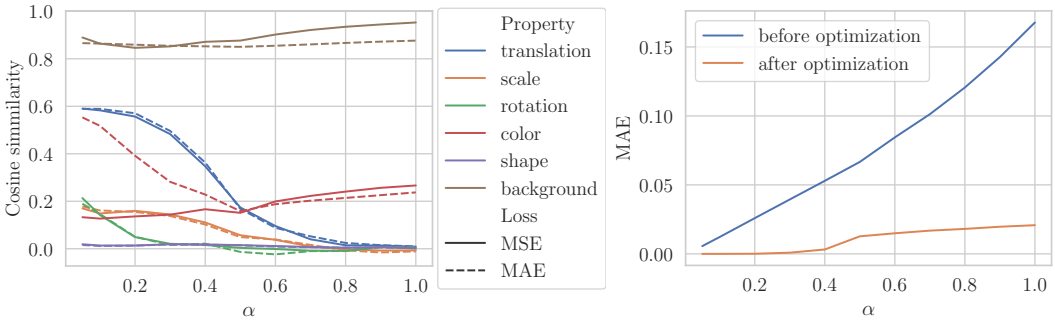

Figure 4: Left: The cosine similarity between gradients calculated for the image-space loss and parameter-space loss for perturbed renderings (α: perturbation strength). Right: The value of the loss function (MAE) before and after optimization.

more details on MONet and Opt-Iter and their parameterization. While the baseline methods fare overall well, they yield to the best configurations of DVP+, particularly on the structural metrics.

Figure 3 presents examples of reconstructions obtained with DVP+ and baseline methods. As all combined variants achieve relatively high metrics, the differences between the renderings they provide are subtle, yet noticeable (e.g., the angles of rotation; the sharpness of corners, especially for the heart shape). More examples of renderings can be found in Sec. E.

Analysis of gradients. To provide a better insight into the differences of the learning guidance provided by the image-space loss function L_x and the parameter-space loss function L_p, we analyze the renderer r in the following way. We sample a pair of random examples from the training set, retrieve their scene parameter vectors p_1 and p_2 and combine them linearly $p' = \alpha p_1 + (1-\alpha)p_2$. Then, we query the renderer on p', calculate the gradient of L_x function on the resulting scene, $\nabla_{p'} L_x(r(p), r(p'))$, and confront it with the analogous gradient exerted by L_p, that is $\nabla_{p'} L_p(p, p')$ using the cosine similarity function. Figure 4a presents the similarity as a function of α varying in $[0.1, 1]$, averaged over a sample of 2048 pairs of examples, for p factored into parameters responsible for particular scene aspects, separately for MSE- and MAE-based L_x. The curves show thus how the direction of the gradient exerted by L_x diverges from that of the 'native' gradient L_p when the rendered scene $r(p')$ is perturbed with respect to p (which is meant to represent the true, optimal interpretation of the scene). Except for color properties, the similarity between parameter vectors decreases quickly with the perturbation strength α, which explains why training DVP+ with L_x is more challenging. Color aspects are exempt from this trend, because the RGB components of rendered pixels depend trivially on their representation in p; this is particularly visible for the background color. The counterintuitive effect of MSE-based L_x on color arises because MSE heavily focus on small mismatched areas with high color differences, prioritizing them over larger areas of good overlap where color differences are minimal. Note that the shape parameters exhibit very low similarity, corroborating the difficulty of learning this particular visual aspect. Figure 4b presents the average of $L_p(p, p')$ and how it can be reduced in 100 steps of optimization with Adam [KB15]. The loss can be reduced to near-zero as long as the gradient directions of ∇L_x and ∇L_p align.

Ablations. We ablate the best configurations reported in Table 2 by eliminating the encoder and/or decoder, in order to assess their usefulness as providers of initial parameter estimates, i.e. starting points for the optimization process. The first configuration, **Rand-OptP**, is an ablation of TI-TP-

Table 3: Test-set evaluation of the ablated variants of the DVP+TI-TP-Opt models.

Ablated configuration	MAE ↓	MSE ↓	SSIM ↑	IoU ↑	ARI ↑
Rand-OptP	0.0409	0.0178	0.8976	0.2399	0.3046
Rand-OptZ	0.0502	0.0217	0.8610	0.2532	0.1983

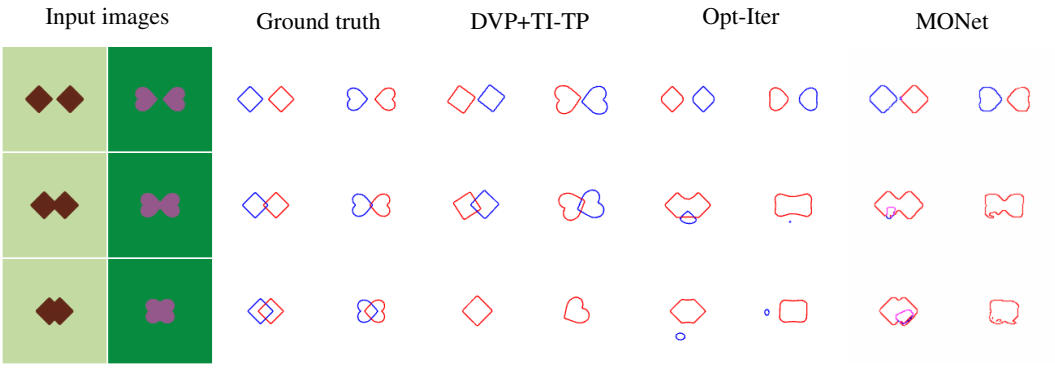

Figure 5: Impact of object overlap on shape recovery.

OptP and consists in optimization *in the parameter space* P. We start with randomly initializing the parameters p for n object slots, where n is the number of objects in the target scene. Then we query the renderer r on it, and minimize image loss until convergence by performing descent along the gradient $\nabla_p L_x(x, r(p))$. Note that this configuration does not rely on shape prototypes. The second configuration, **Rand-OptZ**, is an ablation of TI-TP-OptZ, consists in *optimization in the latent space* Z. The starting point of the optimization z is randomly drawn from the space of latents Z, and the optimization follows $\nabla_z L_x(x, r(g(z)))$.

The metrics achieved by both ablated variants of DVP+, presented in Table 3, are much worse than those for the non-ablated TI-TP-OptP and -OptZ models in Table 2, clearly indicating that the initial estimates predicted by the model are essential for high-quality scene reconstruction.

Overlapping objects. Figure 5 juxtaposes DVP+TI-TP and the baselines in terms of their capacity to recover the shapes of partially overlapping objects. Our model is capable of delineating the objects for moderate degree of overlap (middle row), which is prohibitively difficult for the baselines. When the overlap is high (bottom row), all methods fail. For DVP+TI-TP, this is due to the DETR component segmenting the percept as a single object.

6 Limitations and future work

The above observations are based on a single, synthetic benchmark. However, MDS-HR features a diversified repertoire of shapes, colors, and other visual aspects, providing therefore a representative sample of visual stimuli, similarly to the popular Multi-dSprites [BMW+19]. In an ongoing follow-up work, we are applying DVP+ to medical imaging (histopathology and cytopathology), attempting to model the visual characteristics of individual cells in human tissues.

DVP+ is limited to interpretation of 2D scenes and makes strong assumptions about object appearance (uniform coloring, no textures) and their interactions (simple blending mode when object overlap). However, the main point of this study was to devise (i) a general blueprint for gradient-trainable scene understanding models that involve elements of physical plausibility and (ii) ways in which such models can be effectively trained. We hypothesize (and plan to investigate in future work) that most of those limitations can be addressed by 'lifting' the corresponding elements of our approach. For instance, DVP+ could be relatively easily generalized to 3D scenes by replacing the current shape model with 3D models, updating the definitions of visual aspects to 3D, and equipping it with a 3D renderer.

Acknowledgments and Disclosure of Funding

This research was supported by the statutory funds of Poznan University of Technology and the Polish Ministry of Science and Higher Education, grant nos. 0311/SBAD/0770 and 0311/SBAD/0762.

References

[BMW+19] Christopher P. Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. (arXiv:1901.11390), January 2019.

[BTHR78] Harry Barrow, J Tenenbaum, A Hanson, and E Riseman. Recovering intrinsic scene characteristics. *Comput. vis. syst*, 2(3-26):2, 1978.

[CLG+21] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khamis, Or Litany, and Sanja Fidler. Dib-r++: Learning to predict lighting and material with a hybrid differentiable renderer. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 22834–22848. Curran Associates, Inc., 2021.

[CMS+20] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, page 213–229, Berlin, Heidelberg, 2020. Springer-Verlag.

[DJL21] Alex J. DeGrave, Joseph D. Janizek, and Su-In Lee. AI for radiographic COVID-19 detection selects shortcuts over signal. *Nature Machine Intelligence*, 3(77):610–619, July 2021.

[HA85] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, December 1985.

[JSRV22] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.Jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022.

[KB15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[KG82] Frank P Kuhl and Charles R Giardina. Elliptic fourier features of a closed contour. *Computer Graphics and Image Processing*, 18(3):236–258, 1982.

[KKLD23] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), July 2023.

[KN24] Krzysztof Krawiec and Antoni Nowinowski. Disentangling visual priors: Unsupervised learning of scene interpretations with compositional autoencoder. In Tarek R. Besold, Artur d'Avila Garcez, Ernesto Jimenez-Ruiz, Roberto Confalonieri, Pranava Madhyastha, and Benedikt Wagner, editors, *Neural-Symbolic Learning and Reasoning*, pages 240–256, Cham, 2024. Springer Nature Switzerland.

[KW14] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[LCLL19] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7707–7716, 2019.

[LHK+20] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020.

[LLMRK20] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020.

[MSJ+16] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders, 2016.

[MST+20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[MZX+22] Xu Ma, Yuqian Zhou, Xingqian Xu, Bin Sun, Valerii Filev, Nikita Orlov, Yun Fu, and Humphrey Shi. Towards layer-wise image vectorization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2022.

[ODM+23] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[RGLM21] Pradyumna Reddy, Michaël Gharbi, Michal Lukáč, and Niloy J. Mitra. Im2Vec: Synthesizing vector graphics without vector supervision. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2124–2133, 2021.

[Rou87] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[RRN+20] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with PyTorch3D. (arXiv:2007.08501), 2020.

[WBSS04] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[ZBL+18] John R. Zech, Marcus A. Badgeley, Manway Liu, Anthony B. Costa, Joseph J. Titano, and Eric K. Oermann. Confounding variables can degrade generalization performance of radiological deep learning models. *PLOS Medicine*, 15(11):e1002683, November 2018. arXiv:1807.00431 [cs].

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We propose a novel scene interpretation architecture (Sec. 2), a few modes of its training (Sec. 3) and experimentally demonstrate their superiority to reference algorithms (Sec. 5) on a proposed benchmark (Sec. A) that subsumes prior work (MultidSprites) and is more challenging. In the experimental part, we attempt to explain the observed differences between methods and configurations.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the proposed architecture and training approach are made clear throughout the paper, and have been summarized in Sec. 6, along with suggestions of how they could be addressed.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: Our study does not contain theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All relevant information necessary for reproducing the presented results has been included in the paper, including details on experimental configurations in Sections 5 and H, on our software implementation in Sec. F, and the proposed benchmark (Sec. A).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The pointers to the source code of the implementation of the proposed method and the instructions how to reproduce the experiments are provided in Sec. F. The proposed benchmark is also made publicly available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All details required to understand the results and justification of experimental configuration are provided in Sec. 5 and in the Appendices, especially in Sections H, F and A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

Justification: Due to the relatively high computational costs of model training, we could not afford repeating training runs multiple times. However, in the text we were cautious not to draw definitive and premature conclusions from observed differences on metrics, except for the cases where they were large. We use multiple (5) metrics to provide multi-faceted assessment of compared architectures and arrive at more confident conclusions. Also, while experimenting, we did not find our architectures to be particularly susceptible to parameter initialization, so the figures reported in the tables can be considered quite robust estimates of their true performance.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The information about the computer resources and computational costs have been provided in Sections 5 and G.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: Both authors reviewed the NeurIPS Code of Ethics and declare alignment with it. There are no other special considerations nor regulations that need to be taken into account.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This work does not have direct societal impact, and in particular no negative societal impact. To the contrary, one could claim that the proposed method, by being more transparent and explainable than typical DL models, can ultimately lead to solutions that are more robust and trustworthy for humans.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This study does not incur tangible risks of misuse, and as such does not require safeguards.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All the assets used in this study have been properly credited with citations and references in the source code.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The assets (i.e. the implementation of the proposed method and the MDR-HR benchmark) are documented in the appendices (Secs. F and A) and in the repositories indicated there. They have been properly anonymized.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our submission does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our submission does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The proposed method and methodology do not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

A The MDS-HR dataset

The benchmark proposed and used in this paper, dubbed MDS-HR, is inspired by the popular Multi-dSprites benchmark [BMW+19], but diverges from it in several respects. The primary motivation for its development was the low resolution of Multi-dSprites (64x64), making the disentanglement task more challenging, because the pre-trained models serving as feature extractors in DVP+ (DINOv2 [ODM+23] in our case) struggle to provide useful features for robust characterization of objects comprising just a handful of pixels; see examples in the top row of Fig. 6. To more accurately represent object boundaries in MDS-HR, we incorporated anti-aliasing, replacing the hard, jagged contours of Multi-dSprites. This better reflects the continuous nature of objects contours, which in real-world images contain pixels that have intermediate values between the foreground and the background, rather than just having one of those colors. The other important difference is that MDS-HR does not feature object occlusion. Instead, the colors of overlapping objects are blended. Moreover, in MDS-HR, objects are on average larger and can also extend beyond the viewport. Images shown in in Figure 6 exemplify the differences between the datasets.

We generate the MDS-HR scenes by sampling several object parameters. The number of objects per scene is sampled uniformly from the interval $[1, 4]$. Continuous parameters: color, rotation angle, translation, and scale, are sampled from uniform distributions over the ranges $[0, 1]$, $[0, 2\pi]$, $[0.05, 0.95]$, and $[0.1, 0.3]$[7], respectively. To prevent excessive overlap, object positions are determined by sampling multiple sets of translation vectors for the objects (eight sets per object) and selecting the one that maximizes the minimum pairwise distance between the objects. Finally, object shapes are chosen uniformly from a predefined set: ellipse, heart, and square.

The source code used to generate MDS-HR is available as a part of the code based described in Section F, Running the scripts available there with the above default parameterization produces MDS-HR as used in this paper, comprising the training and testing sets of, respectively $55\,000$ and

[7] A centered square with a scale of 0.5 would completely cover the entire scene.

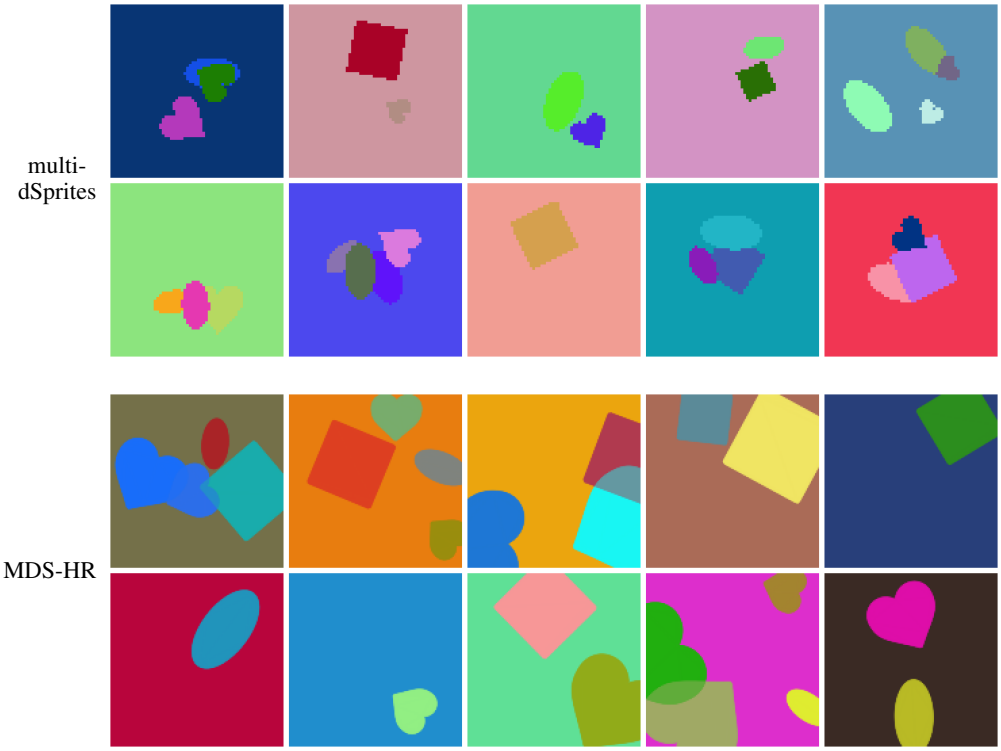

Figure 6: The first ten examples from the training sets of Multi-dSprites (top) and MDS-HR (bottom). Best viewed when zoomed in.

5000 examples. Obviously, it is possible to generate auxiliary sets by changing the above parameters and/or the seed of the random number generator.

B Representing shapes with Elliptic Fourier Descriptors (EFD)

All spectral representations of shape start with the observation that a closed contour of a 2D object, represented as a sequence of K points (x_p, y_p) such that $x_1 = x_K$ and $y_1 = y_K$, can be alternatively viewed as a 1D cyclic complex time series $x_p + jy_p$, where j is the imaginary unit. Such time series lend themselves naturally to discrete Fourier transform, with the coefficients of the resulting spectrum conveniently capturing the coarse aspect of the shape in the low frequencies and the details in the higher frequencies. Another advantage of this class of shape representations is the existence of the inverse transform, which facilitates shape processing and synthesis.

The elliptic Fourier Transform (EFD) [KG82] is a particular method from this family. In contrast to the more basic representation of shape with simple Fourier descriptors, which relies on polar coordinates and assumes the ray length to be a *function* of the angle, EFD can model arbitrary shapes, including such that require the contour to intersect itself. The EFD coefficients order N are defined as:

$$\begin{aligned}
A_n &= \frac{T}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\Delta x_p}{\Delta t_p}\left(\cos\frac{2n\pi t_p}{T} - \cos\frac{2n\pi t_{p-1}}{T}\right)\\
B_n &= \frac{T}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\Delta x_p}{\Delta t_p}\left(\sin\frac{2n\pi t_p}{T} - \sin\frac{2n\pi t_{p-1}}{T}\right)\\
C_n &= \frac{T}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\Delta y_p}{\Delta t_p}\left(\cos\frac{2n\pi t_p}{T} - \cos\frac{2n\pi t_{p-1}}{T}\right)\\
D_n &= \frac{T}{2n^2\pi^2}\sum_{p=1}^{K}\frac{\Delta y_p}{\Delta t_p}\left(\sin\frac{2n\pi t_p}{T} - \sin\frac{2n\pi t_{p-1}}{T}\right)
\end{aligned} \quad \text{for } n = 1, 2, ..., N \tag{3}$$

where $\Delta x_p \equiv x_p - x_{p-1}$, $\Delta y_p \equiv y_p - y_{p-1}$, $\Delta t_p = \sqrt{\Delta x_p^2 + \Delta y_p^2}$ and $T = \sum_{p=1}^{K} \Delta t_p$. The coefficients A_n, B_n, C_n, and D_n form together the EFD. They are translation invariant by design and can be further normalized to be invariant w.r.t. rotation and scale.

The original contour can be reconstructed using the inverse transform given by the following equations:

$$\begin{aligned}
x_p &= \sum_{n=1}^{N}\left(A_n\cos\frac{2n\pi t_p}{T} + B_n\sin\frac{2n\pi t_p}{T}\right)\\
y_p &= \sum_{n=1}^{N}\left(C_n\cos\frac{2n\pi t_p}{T} + D_n\sin\frac{2n\pi t_p}{T}\right)
\end{aligned} \quad \text{for } p = 1, 2, ..., K \tag{4}$$

In DVP+, the coefficients of EFD form the shape representation. We used $N = 16$ and $K = 64$. The $shape()$ component in the decoder head (Fig. 1) maintains a bank of $k = 3$ EFDs defined in this way, each represented as a vector, and producing a sequence of (x_p, y_p)s via the above inverse transform. k is set to 3 by the clustering algorithm (see the final paragraphs of Sec. 3), which successfully discovers that MDS-HR features three classes of shape. When queried on specific latent z, $shape(z)$ performs k-ary soft-indexing of these coordinates, and returns a weighted sum of them, to be passed to the renderer. As the inverse transform is differentiable, the gradient back-propagated from the renderer can effectively update the parameters of $shape()$ component, which learns how to pick the appropriate shape using soft indexing. The k vectors representing the EFDs are not trainable themselves and remain fixed during training, as they have been estimated using the clustering-based algorithm described at the end of Sec. 3.

The parameter-space loss function L_p (Eq. 2 and Sec. C), used in the GP and GI training modes (Sec. 3) contains a term that measures the L_2 discrepancy in the contour space between the predicted EFD, $\hat{p}_i.sh$, with the target EFD, $p_i.sh$.

C The parameter-space loss function L_p

The parameter space loss function introduced in Sec. 3 comprises an object matching phase and a loss calculation phase.

1. In the matching phase, the first two terms of the per-object matching cost function, $C(p_i, \hat{p}_j)$ (Eq. 1), aim to select the optimal match between object candidates and ground truth objects on translation and color. The third term enforces matching stability, as we observed that instability (i.e., the matching process selecting an object with low *confidence* – not preferred by the model) hindered the training process. In training, examples are grouped by the number of ground truth objects to form chunks inside the batches, enabling vectorized computation on these chunks. We utilized the Hungarian method implementation available in the SciPy[8] library due to the lack of an efficient GPU-based counterpart.

2. During loss calculation, determining whether a candidate is a part of the scene is formulated as a binary classification task. Matched candidates serve as the positive class, unmatched candidates as the negative class, and Binary Cross-Entropy (BCE) is used for this loss (Eq. 2, lines 2 and 8). A significantly larger weight on the translation term (Eq. 2, line 3) is a form of *curriculum learning* – as we found out in preliminary experiments, DVP+ models struggle to predict any of the object parameters until they have learned to localize objects.

   The function sym present in the term focused on rotation computes the number of symmetries for shapes represented by Elliptical Fourier Descriptors (EFDs). It first identifies *significant* EFD harmonics by comparing their magnitudes ($L2$ norms of coefficients) against a threshold. The core of the method lies in calculating the Greatest Common Divisor (GCD) of the indices of these significant harmonics. This GCD value is returned as the order of rotational symmetry, defaulting to 1 if no harmonics are sufficiently significant. This handling of the prediction angle rewards the model with low loss values if it predicts the rotation angle correctly *modulo the number of object's symmetries*, or, more precisely, modulo $2\pi/sym()$ (e.g., modulo 90 angular degrees for squares).

D Training of models

All models were trained using the Adam optimizer [KB15] with a learning rate of 0.0001 and a budget of 100 epochs. For training, all DVP+ configurations used a batch size of 128 (the largest power of two size fitting into the GPU memory), while MONet used a batch size of 64 (due to the same constraint). For testing, models were selected based on their L_x scores in the validation set. The validation set contained 5000 examples and has been sampled randomly from the training set, effectively reducing the size of the latter from 55 000 to 50 000. The validation set contains the same examples in all configurations examined in experiments.

E Additional results

Figure 7 presents more examples of scene reconstructions produced by the methods compared in Sec. 5, presented similarly to Fig. 3. Most of the basic configurations of DVP+ (TI, GP, GI and TP) trained in the simple training regimes, i.e. with one of the considered loss functions (in the image space or in the parameter space) fail almost completely, for the reasons discussed in Sec. 5. The only exception is GP, the variant trained with the parameter-space loss function L_p. In contrast, all combined variants of DVP+ and the baselines reproduce images quite well, and in some cases almost perfectly. Recall, however, that for MONet it is a given, as it represents objects with raster masks, so it does not inherently model the notion of shape.

Bar the trivial omissions of objects by the basic variants of DVP+, the visual aspect that seems to be most problematic for the method is rotation. Translation, scale and colors (both background and object) do not pose significant challenges. This corroborates the conclusions of our discussion on the particular strong entanglement of this aspect with object shape, which motivated us to detach the learning of shapes from the rest of the training process (see the final paragraphs of Sec. 3).

[8] https://scipy.org/

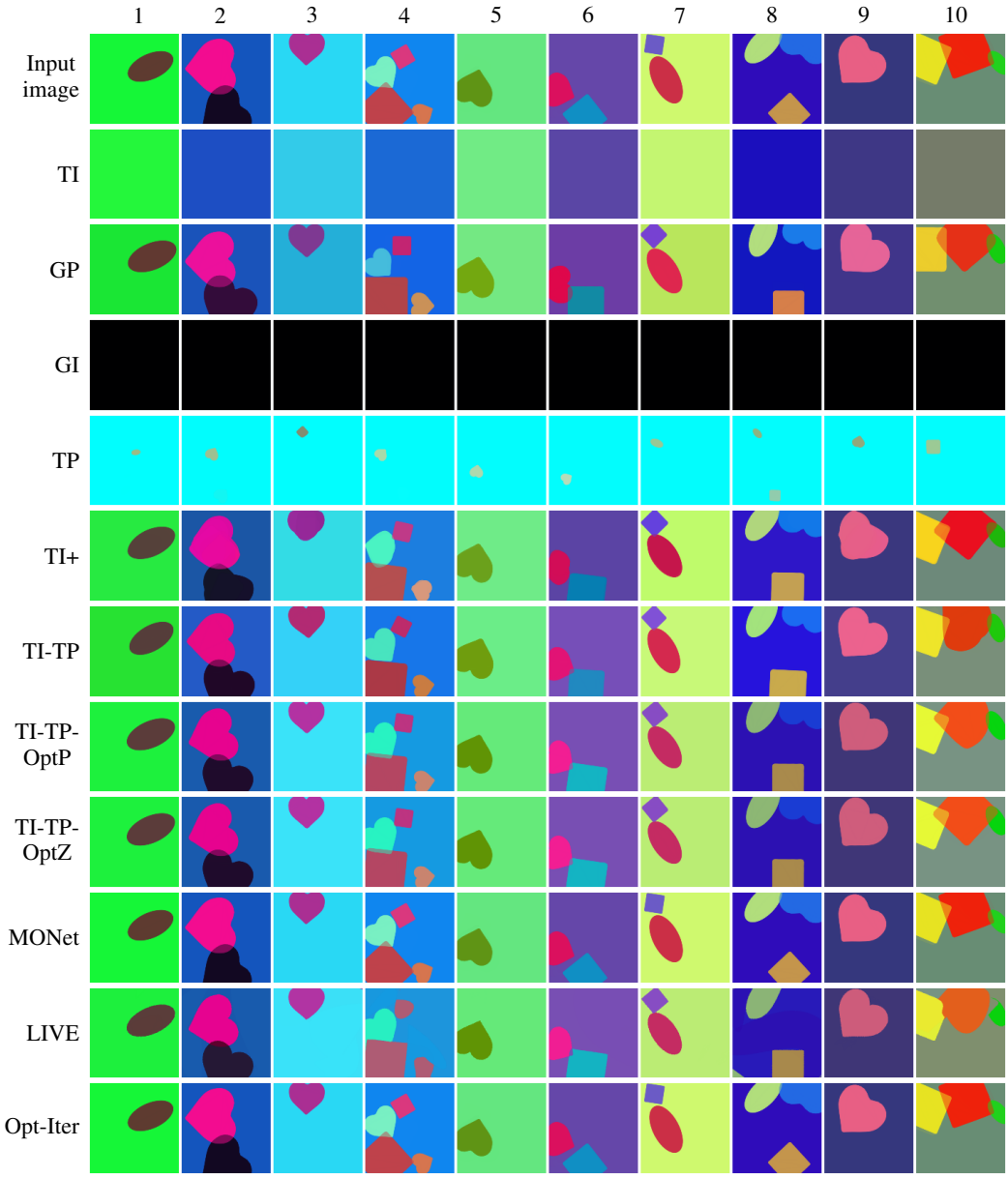

Figure 7: More reconstruction results for selected images from the testing set (cf. Fig. 3).

Notice that most methods and configurations struggle with objects cropped by the boundaries of the viewport (except for MONet, for the reasons indicated above). This is particularly true for rectangles (e.g. the red one in #4, the blue one in #6, the orange one in #8), probably due to this shape having the highest number of symmetries among the shapes present in MDS-HR (cf. the role of the number of symmetries in the discussion on the parameter-space loss function L_p in Sec. C). This happens less often for the heart shape and mostly for the more basic variants of the approach (e.g. #6 for GP and TI).

F Software implementation

Concerning the **overall architecture** of DVP+, we implemented it in PyTorch[9], with particular support of the following libraries:

- Torchvision[10]
- TorchMetrics[11]
- TensorDict[12]
- Kornia[13]
- scikit-learn[14]
- torchtyping[15]
- typeguard[16]

The constituent models and components came from the following sources:

- DINO implementation and weights from the PyTorch Hub[17]
- Conditional DETR[18] (a DETR-derived architecture) implementation from HuggingFace's Transformers library[19]

The source code of the implementation is available as an Git repository under the following location: `https://github.com/Antollo/dvp_plus` We made it available under the permissive MIT license. The repository contains also the functions and scripts used to populate the MDS-HR benchmark (Sec. A) and the actual benchmark as well. The README.md file in the root folder of the repository contains the instructions on preparation of the Python environment and installation, and points to Jypyter notebooks for training and querying of particular variants of DVP+ as well as the baseline methods.

For the **differentiable rendering** we used the implementation available in the PyTorch3D[20] [RRN+20]. PyTorch3D's renderer is based on the SoftRas method [LCLL19]. While PyTorch3D offers various utilities for handling many types of shape representations, integrating it with our model required implementing additional steps. The resulting rendering pipeline comprises the following steps:

1. Our model predicts a shape represented as a vector of Elliptical Fourier Descriptors (EFDs; Sec. B),
2. The EFD is transformed into a shape contour – a sequence of (x_p, y_p),
3. The sequence is subject to the geometric transformations (translation, rotation, scaling) determined by the predictors in the decoder head (Fig. 1) of the remaining visual aspects (see Sec. 2 for the list of aspects).

[9] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. *In NIPS-W*, 2017.

[10] `https://github.com/pytorch/vision`

[11] `https://github.com/Lightning-AI/torchmetrics`

[12] `https://github.com/pytorch/tensordict`

[13] `https://github.com/kornia/kornia`

[14] `https://scikit-learn.org/`

[15] `https://github.com/patrick-kidger/torchtyping`

[16] `https://github.com/agronholm/typeguard`

[17] `https://pytorch.org/hub/`

[19] `https://github.com/huggingface/transformers`

[19] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional DETR for fast training convergence. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[20] `https://pytorch3d.org/`

4. The transformed contour is triangularized. This step generates a list of indices that point to the contour's vertices to form a mesh of triangles defining the shape (conceptually similar to the index buffer used in indexed drawing in computer graphics). For this task, we implemented PyTorch bindings for the earcut.hpp[21] library.
5. Finally, the array of indices and the array of vertices, containing position, color, and confidence (as predicted by the encoder head; Fig. 1), assigned to each vertex, are passed to the renderer.

Concerning the actual rendering algorithm, inspired by the SoftRas method [LCLL19], we compute a signed distance function $d(i, j)$ for each pixel i and triangle j. This rasterization step is accelerated using CUDA kernels in PyTorch3D. A sigmoid function then transforms these distances into a soft mask D_j^i representing the influence of each triangle on each pixel:

$$D_j^i = \text{sigmoid}\left(\frac{d(i,j)}{\sigma}\right) \tag{5}$$

These masks, combined with an object-specific confidence score f_j (for the object containing triangle j), predicted in the decoder head, determine the final per-pixel, per-triangle weights w_j^i.

$$w_j^i = \frac{D_j^i \exp\left(f_j/\gamma\right)}{\sum_k D_k^i \exp\left(f_k/\gamma\right)} \tag{6}$$

Finally, the resulting image I is calculated as a weighted sum of colors (where C_j is the color of jth triangle and C_b is the background color):

$$\begin{aligned} \alpha &= 1 - \prod_j \left(1 - w_j^i C_j\right) \\ I^i &= \alpha \sum_j w_j^i C_j + (1-\alpha) C_b \end{aligned} \tag{7}$$

G Hardware specification

The configuration of the workstation used in the experimental part of the study was quite moderate:

- CPU: Intel Core i7-11700KF (3.6 GHz base, up to 5.0 GHz boost, 8 cores, 16 threads)
- GPU: NVIDIA GeForce RTX 3090 (24 GB VRAM)
- RAM: 32 GB DDR4 (3200 MT/s)
- Storage: 2 TB Kingston KC3000 NVMe
- Operating system: Ubuntu 22.04.5 (via WSL 2 on Windows 11)

Training of a single DVP+ model on this hardware does not exceed 10 hours, with the actual times that we observed in our experiments ranging from a minimum of 6 hours 49 minutes for DVP+TI+ to a maximum of 9 hours 21 minutes for DVP+TI-TP.

H Experimental configurations

The DVP+ architecture, as described in Sec. 2, has the following configuration:

Encoder

1. Pre-trained vision transformer: DINOv2 (`dinov2_vitb14_reg` variant) was used as the feature extractor. Its weights were kept frozen during all DVP+ training.
2. Encoder-Decoder Transformer (EDT): We used a standard Conditional DETR[22] transformer (6 encoder layers, 6 decoder layers, 8 attention heads, 256 hidden dimensions) for the EDT component. We had to disable the dropout, as it hindered training.

[21] https://github.com/mapbox/earcut.hpp

3. Object queries: 8 learnable object queries were used to probe the EDT.

Decoder

The decoder head consists of seven independent Multi-Layer Perceptrons (MLPs). Each MLP has 3 hidden layers of 256 units each, and ReLU activations between them. The MLPs vary only in the size of the output layer (determined by the dimensionality of the predicted vector, e.g. 3 for color) and in the activation function:

1. Color, translation, scaling, confidence, shape, and background color MLPs use the sigmoid as the final activation.
2. Rotation MLP used tanh as the final activation, followed by a normalization to unit length.

For the DVP+TI-TP-OptP and DVP+TI-TP-OptZ, 100 iterations of the Adam optimizer with the learning rate of 0.01 were used for the per-scene optimization of, respectively, parameters p or latents z. The learning rate was controlled by PyTorch's `ReduceLROnPlateau` learning rate scheduler configured with `patience` of 10 iterations, `cooldown` of 10 iterations, and `factor` of 0.5.

I Baseline methods

LIVE. As signaled in the main text, LIVE [MZX+22] operates similarly to our Opt-Iter baseline (because our Opt-Iter was inspired and based on LIVE). LIVE iteratively identifies the image region that is most divergent from the current scene approximation, initializes a new object candidate at that location (sampling its color from the region), and then optimizes all candidates introduced so far to minimize L_x. To represent object shape, LIVE uses closed Bézier paths (in contrast to Opt-Iter, which relies on EFDs – as a matter of fact, this is the main difference between both methods). The cycle repeats until a predetermined number of objects is reached. By adding consecutive object candidates sequentially, LIVE attempts to address the risk of single objects being reconstructed with multiple object candidates. This also leads to reconstructing the target with fewer primitives, making the reconstruction more interpretable.

We used the official LIVE implementation[23]. The results reported in Sec. 5 employed the predefined `experiment_5x1` configuration.

MONet. The Multi-Object Network (MONet) [BMW+19] is a composite architecture that combines image segmentation based on an attention mechanism (to delineate image components) with a variational autoencoder (VAE), for rendering individual components in the scene. Similarly to our approach, it attempts to reconstruct the input image/scene and can be trained end-to-end with gradient. However, MONet does not involve geometric aspects of image formation, in particular it does not perform geometric rendering of objects: the subimages of individual components are generated with the VAE and 'inpainted' into the scene using raster masks.

Our implementation of MONet is based on Object-Centric Library[24] and uses the following parameter setting: 5 slots and 6 U-Net blocks.

[22] Depu Meng, Xiaokang Chen, Zejia Fan, Gang Zeng, Houqiang Li, Yuhui Yuan, Lei Sun, and Jingdong Wang. Conditional DETR for fast training convergence. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2021.

[23] https://github.com/Picsart-AI-Research/LIVE-Layerwise-Image-Vectorization

[24] https://github.com/addtt/object-centric-library