A. Additional Dataset Details

A.1. Retinal Blood Vessel and Road Satellite Images

The retinal blood vessel and road satellite image object masks which we use to generate the synthetic images for the experiments in Sec. 4.3.1, are from the Retina Blood Vessel [27] and the Road Extraction Challenge data of DeepGlobe18 [9]. For our retinal vessel mask set, we use the vessel masks from the training set of 80 image/mask pairs, and for our road mask set, we randomly sample the same number of masks from the split-merged full dataset, in order to ensure that these two object types appear equally in the synthetic dataset of Sec. 4.3.1.

A.2. Synthetic Tree-like Object Images

Our algorithm for creating synthetic images of tree-like objects (used in Sec. 4.3.1) is shown as follows.

1. Sample object mask m from either retinal blood vessel or satellite road image datasets (details in App. A.1).
2. Randomly select one of the contiguous components of m, denoted m_c, and enclose it with a tight bounding box.
3. Resize m_c such that its bounding box is 512×512, and randomly place it in a blank 1024×1024 image.
4. Apply two different textures randomly sampled from Colored Brodatz [1] to the foreground ($m_c = 1$) and background ($m_c = 0$), resulting in the final image.

In Fig. 11 we show example images generated with different foreground and background texture combinations for various objects.

A.3. Neural Style Transfer (NST) Generated Images

In Fig. 10 we provide additional example images generated via inpainting and neural style transfer for the textural separability experiments of Sec. 5.2.1.

A.4. Example Images from Plittersdorf and iShape

We show example images from Plittersdorf and iShape in Fig. 12.

B. Hyperparameter/Ablation Studies

B.1. Object Tree-likeness Experiments

B.1.1. Synthetic Data Experiments

Tables 1 and 2 show results on the synthetic tree-like dataset using a wide range of CPR and DoGD hyperparameters (R and a, b, respectively).

	ViT-H		**ViT-B**	
R	τ	ρ	τ	ρ
1	-0.82	-0.96	-0.79	-0.94
3	-0.80	-0.95	-0.76	-0.94
5	-0.77	-0.93	-0.75	-0.93
7	-0.74	-0.92	-0.73	-0.91
9	-0.72	-0.91	-0.70	-0.90
11	-0.71	-0.90	-0.70	-0.89

Table 1. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **CPR** with different hyperparameter values R on the **synthetic dataset**, to supplement Fig. 4.

B.1.2. Real Data Experiments

Tables 3, 4, 5, and 6 show results on DIS and iShape using a wide range of CPR and DoGD hyperparameters (R and a, b, respectively).

B.2. Object Textural Separability Experiments

Table 7 show the textural separability experiment results on real data (iShape and Plittersdorf) using various hyperparameters and models for the classifier g.

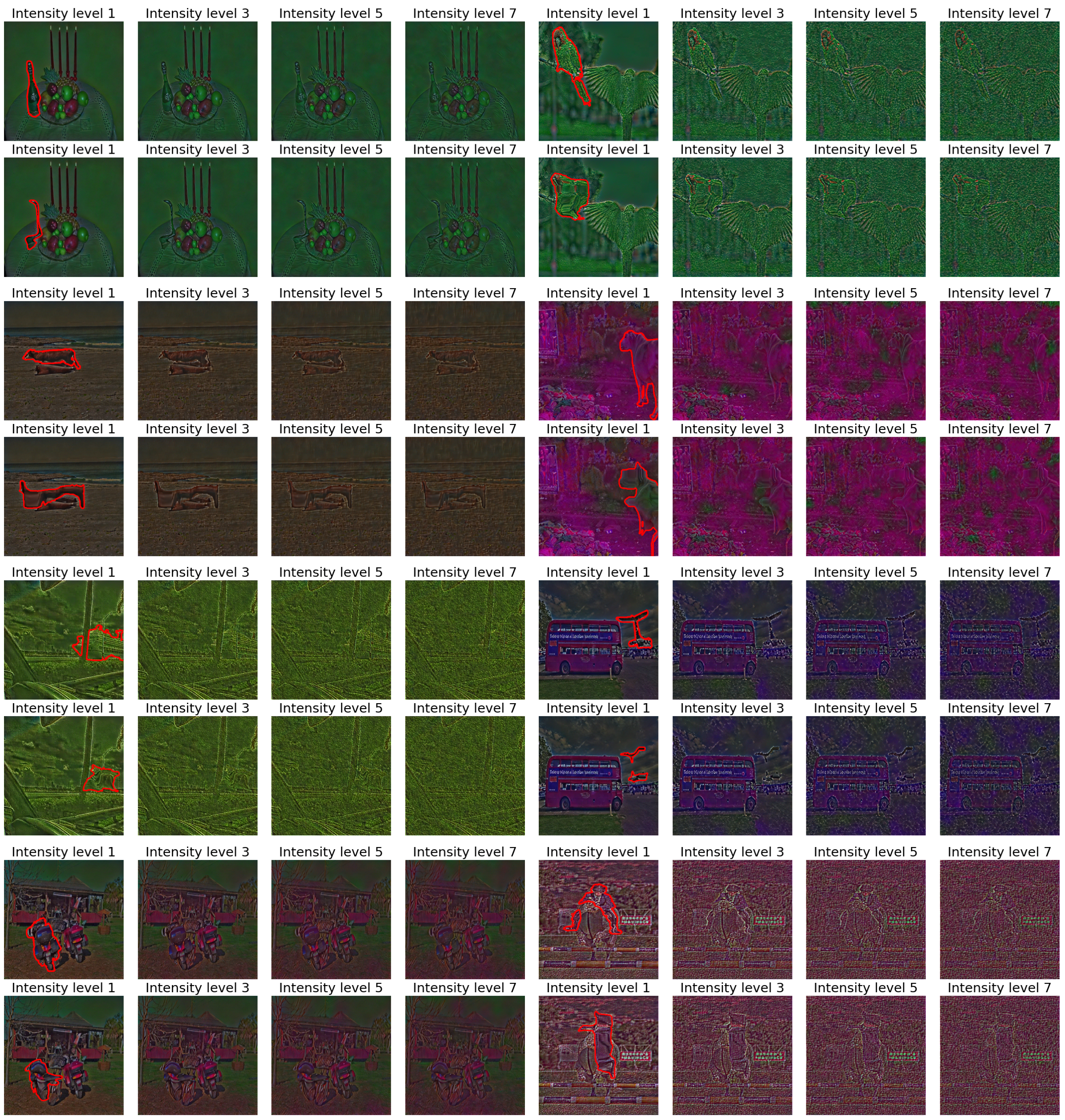

Figure 10. Additional examples of inpainting+neural style transfer-generated images to supplement Fig. **??**.

C. Additional Experimental Details

C.1. SFM Prompting Strategies

In addition to the SFM prompting details presented in Sec. 3, all real and style-transferred (NST) images are prompted with (1) a tight bounding-box and (2) several positive and/or negative prompts randomly sampled from either the foreground or background of the object mask, respectively, with respective counts n_{pos} and n_{neg}. For iShape, we use $n_{pos} = n_{neg} = 5$; for DIS and MOSE, we use $n_{pos} = 5$ and $n_{neg} = 10$ due to the complexity of the objects. For Plittersdorf and NST images, we simply use $n_{pos} = 0$ and $n_{neg} = 2$, due to the objects typically possessing simple shapes.

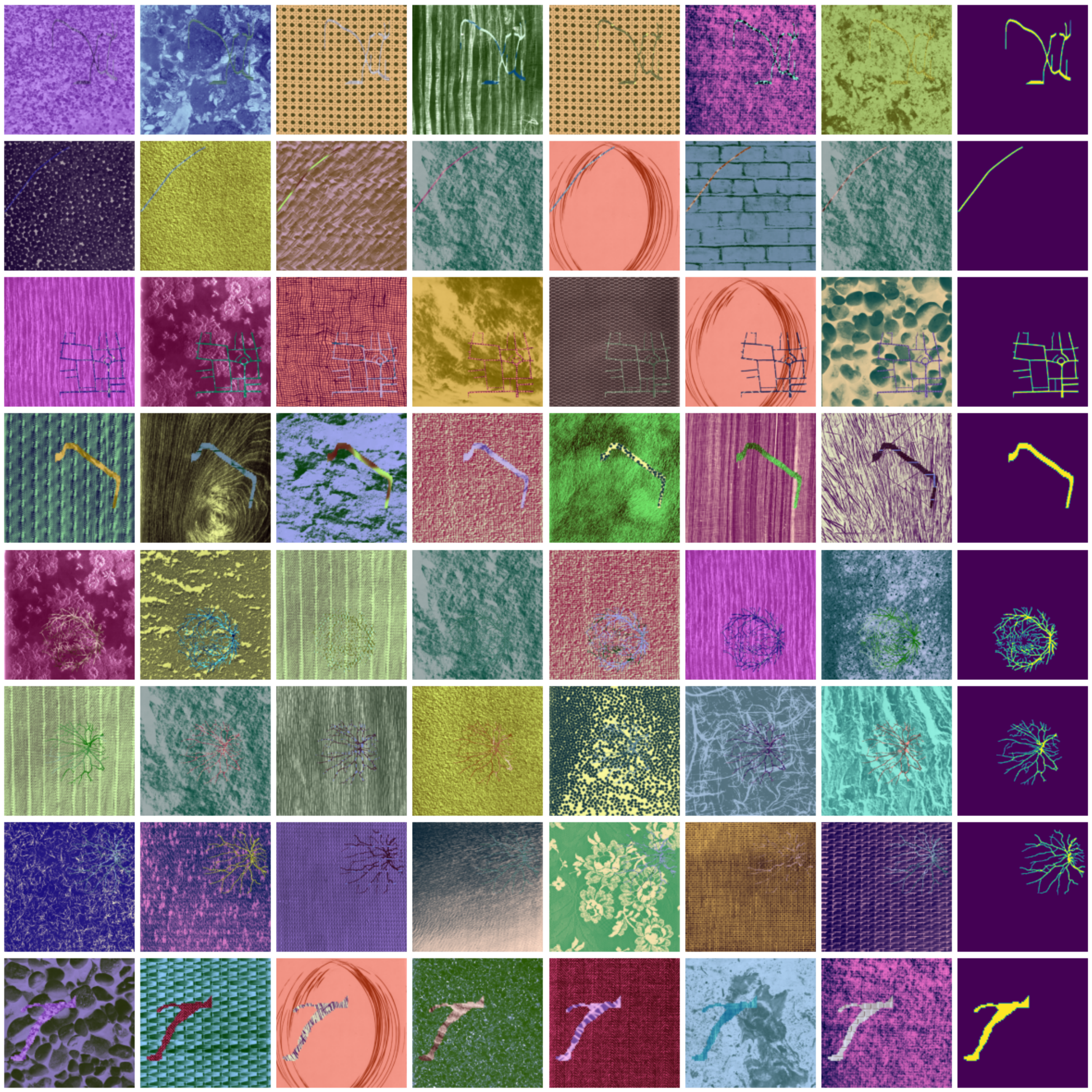

Figure 11. Examples of synthetic tree-like object images. Object masks shown in the right-most column, with images created using randomly chosen different foreground and background texture combinations in the left columns (before the majority-voting procedure described in Sec. 4.3.1).

C.2. Neural Style Transfer Experiments

Models. For the NST experiments (Sec. 5.2.1), the inpainting model used is Runway's stable diffusion inpainting pipeline, `fp16` variant. The NST model itself is an implementation of [17] based on `https://github.com/pytorch/tutorials/blob/main/advanced_source/neural_style_tutorial.py`, using typical settings of a standard VGG19 CNN with content layer of `conv_4`, style layers of `conv_i` for `i = 1, 2, 3, 4, 5`, and ImageNet normalization.

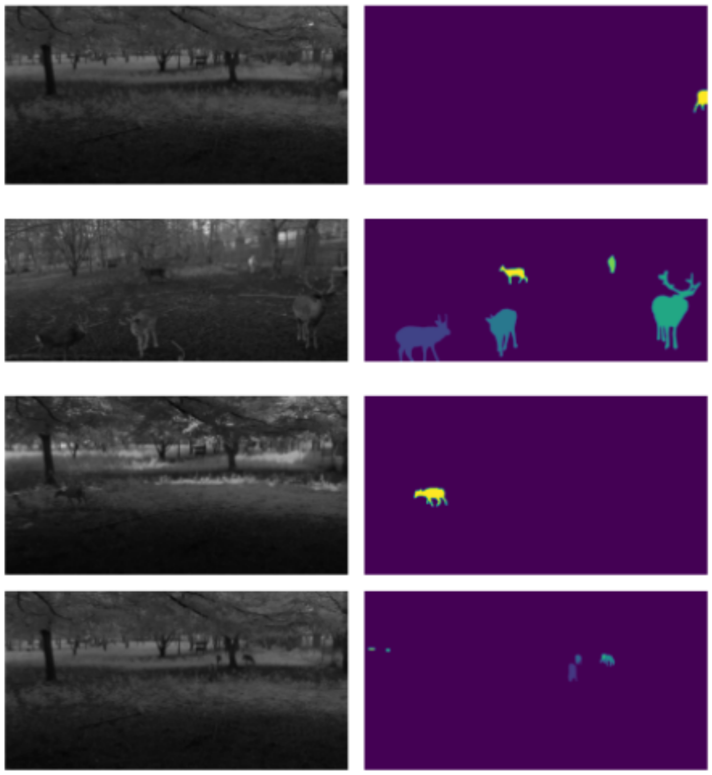
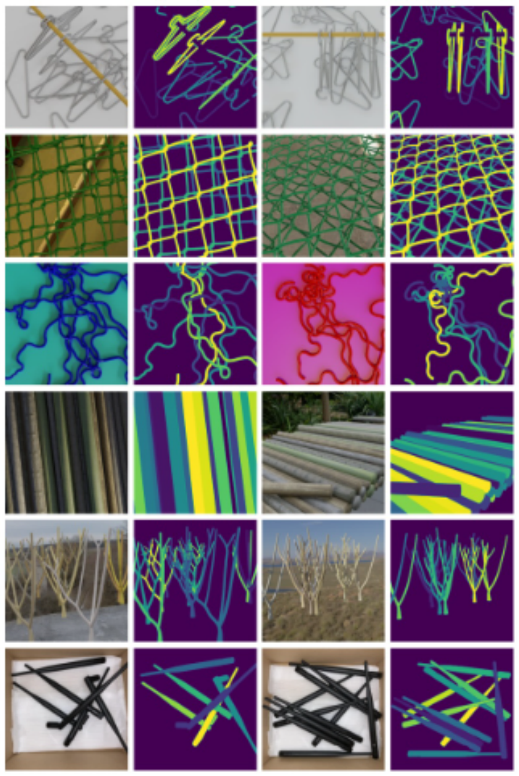
Figure 12. Example images and object masks from Plittersdorf (left group) and iShape (right group).

		ViT-H		**ViT-B**	
a	b	τ	ρ	τ	ρ
63	3	0.42	0.60	0.48	0.66
63	7	0.45	0.63	0.49	0.68
63	15	0.44	0.64	0.48	0.68
63	31	0.45	0.66	0.49	0.71
127	3	0.61	0.80	0.66	0.84
127	7	0.63	0.81	0.67	0.84
127	15	0.62	0.81	0.66	0.84
127	31	0.60	0.80	0.64	0.83
255	3	0.66	0.84	0.68	0.85
255	7	0.66	0.84	0.68	0.86
255	15	0.65	0.84	0.67	0.85
255	31	0.59	0.79	0.60	0.80

Table 2. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **DoGD** with different hyperparameter values a, b on the **synthetic dataset**, to supplement Fig. 4.

Style transfer intensity. As mentioned in Sec. 5.2.1, we perform style transfer experiments for a monotonically-increasing range of eight degrees of style transfer intensity. This is created by defining content and style weights λ_c and λ_s, respectively for the NST algorithm according to

$$\lambda_c = \frac{1}{1+\alpha} \quad \text{and} \quad \lambda_s = \frac{\alpha}{1+\alpha}, \tag{6}$$

	ViT-H		ViT-B	
R	τ	ρ	τ	ρ
1	-0.56	-0.73	-0.61	-0.79
3	-0.55	-0.72	-0.61	-0.79
5	-0.59	-0.76	-0.63	-0.81
7	-0.53	-0.7	-0.59	-0.77
9	-0.53	-0.71	-0.57	-0.75
11	-0.51	-0.69	-0.56	-0.74

Table 3. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **CPR** with different hyperparameter values R on **DIS**, to supplement Fig. 5.

		antenna		branch		fence		hanger		log		wire	
R	**SAM Enc.**	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ
1	ViT-H	-0.18	-0.25	-0.63	-0.83	-0.68	-0.88	-0.30	-0.43	-0.60	-0.78	-0.08	-0.12
	ViT-B	-0.19	-0.29	-0.64	-0.83	-0.68	-0.88	-0.47	-0.66	-0.61	-0.79	-0.12	-0.18
3	ViT-H	-0.20	-0.30	-0.63	-0.83	-0.65	-0.85	-0.28	-0.40	-0.61	-0.79	-0.07	-0.10
	ViT-B	-0.17	-0.26	-0.64	-0.83	-0.65	-0.85	-0.48	-0.66	-0.61	-0.79	-0.13	-0.19
5	ViT-H	-0.13	-0.21	-0.61	-0.81	-0.61	-0.81	-0.25	-0.37	-0.60	-0.79	-0.06	-0.10
	ViT-B	-0.18	-0.27	-0.63	-0.82	-0.63	-0.83	-0.44	-0.62	-0.62	-0.80	-0.12	-0.17
7	ViT-H	-0.16	-0.23	-0.57	-0.77	-0.54	-0.74	-0.20	-0.29	-0.60	-0.79	-0.06	-0.10
	ViT-B	-0.15	-0.24	-0.59	-0.77	-0.55	-0.75	-0.39	-0.55	-0.62	-0.81	-0.10	-0.15
9	ViT-H	-0.16	-0.23	-0.52	-0.72	-0.47	-0.65	-0.13	-0.19	-0.59	-0.78	-0.05	-0.08
	ViT-B	-0.16	-0.22	-0.57	-0.76	-0.49	-0.67	-0.29	-0.42	-0.62	-0.81	-0.10	-0.15
11	ViT-H	-0.08	-0.13	-0.49	-0.67	-0.51	-0.70	-0.02	-0.03	-0.60	-0.78	-0.03	-0.04
	ViT-B	-0.12	-0.16	-0.54	-0.73	-0.55	-0.75	-0.21	-0.28	-0.64	-0.82	-0.07	-0.11

Table 4. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **CPR** with different hyperparameter values R on **iShape**, to supplement Fig. 5.

		ViT-H		ViT-B	
a	b	τ	ρ	τ	ρ
63	3	0.63	0.83	0.57	0.78
63	7	0.59	0.79	0.57	0.77
63	15	0.54	0.72	0.50	0.67
63	31	0.45	0.62	0.39	0.56
127	3	0.58	0.77	0.45	0.64
127	7	0.58	0.77	0.49	0.66
127	15	0.51	0.67	0.42	0.57
127	31	0.36	0.50	0.26	0.39
255	3	0.53	0.71	0.42	0.59
255	7	0.53	0.71	0.42	0.59
255	15	0.47	0.66	0.36	0.52
255	31	0.29	0.43	0.19	0.29

Table 5. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **DoGD** with different hyperparameter values a, b on **DIS**, to supplement Fig. 5.

where α ranges linearly from $\alpha = 1$ to $\alpha = 4,000$ with eight equally-spaced values, representing the degree of style transfer intensity.

			antenna		branch		fence		hanger		log		wire	
a	b	**SAM Enc.**	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ
63	3	ViT-H	0.39	0.55	0.62	0.82	0.65	0.85	−0.15	−0.23	0.23	0.35	0.35	0.50
		ViT-B	0.25	0.35	0.62	0.81	0.60	0.81	−0.07	−0.11	0.17	0.25	0.36	0.51
63	7	ViT-H	0.28	0.38	0.61	0.81	0.62	0.83	−0.13	−0.21	0.22	0.33	0.34	0.49
		ViT-B	0.13	0.18	0.61	0.81	0.60	0.80	−0.07	−0.11	0.14	0.22	0.33	0.48
63	15	ViT-H	0.10	0.12	0.59	0.78	0.59	0.80	−0.05	−0.07	0.19	0.30	0.29	0.43
		ViT-B	0.04	0.03	0.59	0.77	0.56	0.77	−0.01	−0.01	0.09	0.16	0.25	0.36
63	31	ViT-H	0.06	0.04	0.55	0.74	0.46	0.65	0.07	0.10	0.14	0.26	0.09	0.13
		ViT-B	−0.11	−0.17	0.56	0.74	0.45	0.64	0.07	0.11	0.02	0.08	0.02	0.03
127	3	ViT-H	0.23	0.30	0.52	0.70	0.46	0.65	0.52	0.72	0.30	0.46	0.08	0.11
		ViT-B	0.12	0.14	0.51	0.69	0.45	0.64	0.50	0.70	0.13	0.22	0.06	0.08
127	7	ViT-H	0.23	0.26	0.48	0.65	0.41	0.59	0.51	0.70	0.29	0.46	−0.01	−0.01
		ViT-B	0.10	0.09	0.51	0.68	0.41	0.59	0.49	0.68	0.13	0.22	−0.03	−0.05
127	15	ViT-H	0.14	0.17	0.43	0.59	0.38	0.55	0.50	0.69	0.28	0.43	−0.09	−0.12
		ViT-B	0.07	0.03	0.45	0.62	0.36	0.52	0.48	0.67	0.12	0.20	−0.14	−0.20
127	31	ViT-H	0.05	0.05	0.36	0.50	0.08	0.12	0.48	0.67	0.30	0.45	−0.15	−0.22
		ViT-B	0.04	0.00	0.38	0.52	0.12	0.18	0.46	0.65	0.12	0.20	−0.21	−0.31
255	3	ViT-H	0.04	0.06	0.33	0.46	−0.20	−0.30	0.44	0.61	0.52	0.72	−0.17	−0.25
		ViT-B	0.21	0.31	0.30	0.42	−0.13	−0.20	0.42	0.59	0.42	0.60	−0.23	−0.34
255	7	ViT-H	0.02	0.02	0.29	0.41	−0.25	−0.35	0.43	0.60	0.51	0.71	−0.20	−0.29
		ViT-B	0.19	0.27	0.26	0.36	−0.18	−0.26	0.42	0.58	0.41	0.59	−0.25	−0.36
255	15	ViT-H	−0.04	−0.06	0.24	0.34	−0.31	−0.45	0.43	0.61	0.50	0.70	−0.20	−0.30
		ViT-B	0.21	0.29	0.20	0.28	−0.26	−0.39	0.40	0.56	0.40	0.58	−0.28	−0.40
255	31	ViT-H	−0.11	−0.14	0.11	0.16	−0.47	−0.65	0.43	0.59	0.46	0.66	−0.23	−0.33
		ViT-B	0.13	0.20	0.07	0.09	−0.42	−0.60	0.39	0.55	0.38	0.55	−0.30	−0.44

Table 6. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object tree-likeness as measured by **DoGD** with different hyperparameter values R on **iShape**, to supplement Fig. 5.

Object/image altering procedure. The procedure that creates the "altered" version of objects via non-affine transformations is detailed as follows.

1. Apply a non-affine transformation created using the `albumentations` Python library [5] to both the image and the object/mask, defined shortly.
2. Clean up unexpected obsolete regions.
3. Properly align the location of the distorted object.
4. Remove small isolated regions of the object.

In Python code, this is implemented as follows, given an input image NumPy array `raw_img_arr` and corresponding binary object mask `raw_msk_arr`:

```python
import torch
import numpy as np
from PIL import Image
import albumentations as A
import torchvision.transforms as transforms
from skimage.morphology import dilation,label, \
disk, remove_small_holes, remove_small_objects, \
opening, closing

shape_transformation = A.Compose([
    A.ElasticTransform(
        alpha=2500, sigma=24, alpha_affine=0,
        interpolation=1, border_mode=0, value=0,
        mask_value=0, always_apply=True,
        approximate=False, same_dxdy=False, p=1.0),
    A.GridDistortion(
        num_steps=20, distort_limit=.99,
        interpolation=1, border_mode=0, value=0,
        mask_value=0, normalized=False,
        always_apply=False, p=1.0)]
```

		iShape													
		antenna		branch		fence		hanger		log		wire		**Plittersdorf**	
Weak Classifier Model	**SAM Enc.**	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ	τ	ρ
Logistic,	ViT-H	0.44	0.59	0.50	0.68	0.65	0.85	0.63	0.83	0.33	0.48	0.49	0.67	0.26	0.38
$C = 2$	ViT-B	0.46	0.61	0.56	0.75	0.67	0.86	0.65	0.84	0.36	0.52	0.55	0.73	0.36	0.50
Logistic,	ViT-H	0.42	0.58	0.49	0.67	0.64	0.84	0.62	0.81	0.31	0.46	0.49	0.67	0.34	0.49
$C = 1$	ViT-B	0.44	0.60	0.55	0.75	0.66	0.85	0.63	0.82	0.35	0.50	0.55	0.72	0.39	0.55
Logistic,	ViT-H	0.39	0.53	0.49	0.67	0.65	0.85	0.60	0.80	0.30	0.43	0.48	0.65	0.33	0.47
$C = 0.5$	ViT-B	0.48	0.65	0.57	0.76	0.66	0.86	0.62	0.81	0.33	0.48	0.52	0.69	0.41	0.56
Random Forest,	ViT-H	0.48	0.66	0.45	0.63	0.46	0.64	0.45	0.64	0.19	0.27	0.25	0.36	0.35	0.50
$n_{\text{estimators}} = 2$	ViT-B	0.50	0.67	0.49	0.68	0.47	0.65	0.50	0.69	0.24	0.34	0.31	0.45	0.41	0.56
Random Forest,	ViT-H	0.51	0.69	0.42	0.60	0.54	0.73	0.48	0.67	0.21	0.30	0.30	0.43	0.22	0.31
$n_{\text{estimators}} = 4$	ViT-B	0.51	0.70	0.48	0.66	0.53	0.72	0.51	0.70	0.22	0.32	0.37	0.52	0.32	0.45
Random Forest,	ViT-H	0.48	0.65	0.46	0.63	0.56	0.75	0.50	0.69	0.19	0.28	0.35	0.50	0.22	0.31
$n_{\text{estimators}} = 8$	ViT-B	0.49	0.67	0.49	0.68	0.55	0.74	0.54	0.73	0.21	0.31	0.40	0.57	0.25	0.35
Random Forest,	ViT-H	0.45	0.62	0.48	0.66	0.57	0.77	0.52	0.72	0.20	0.29	0.38	0.53	0.08	0.12
$n_{\text{estimators}} = 16$	ViT-B	0.58	0.77	0.52	0.70	0.54	0.74	0.54	0.74	0.23	0.33	0.41	0.57	0.15	0.23

Table 7. Rank correlation (Kendall τ and Spearman ρ) between SAM prediction IoU and object textural separability (Algorithm 3), on real datasets (iShape and Plittersdorf), using different weak classifier models g and hyperparameters to measure textural separability (to supplement Fig. 8).

```
)

imsize = 512
loader = transforms.Compose([
    transforms.Resize(imsize),  # scale imported image
    transforms.CenterCrop(imsize),
    transforms.ToTensor()])

unloader = transforms.ToPILImage()

def shift_array(array, shift_spec):
    assert len(shift_spec) == 2 and \
        isinstance(array,np.ndarray)

    shift_corrected_arr = array.copy()
    if shift_spec[0] != 0:
        shift = shift_spec[0]
        shift_corrected_arr = np.concatenate(
            [shift_corrected_arr[shift:],
            shift_corrected_arr[:shift]],
            axis=0)

    if shift_spec[1] != 0:
        shift = shift_spec[1]
        shift_corrected_arr = np.concatenate(
            [shift_corrected_arr[:,shift:],
            shift_corrected_arr[:,:shift]],
            axis=1)

    return shift_corrected_arr

def center_correction(raw_msk, distorted_msk):
    assert raw_msk.ndim == 4 and \
        distorted_msk.ndim == 3
    pos_idx = torch.argwhere(raw_msk[0,0])
    _raw_msk_center = (pos_idx.min(dim=0).values + \
        pos_idx.max(dim=0).values)//2
    pos_idx = torch.argwhere(distorted_msk[0])
    _distorted_msk_center = (pos_idx.min(dim=0).values + \
        pos_idx.max(dim=0).values)//2

    position_correction = _distorted_msk_center -_raw_msk_center
    corrected_distorted_msk = distorted_msk.squeeze()
    print(corrected_distorted_msk.shape)
    assert position_correction.ndim == 1 and len(position_correction) == 2

    if position_correction[0] !=0:
        shift = position_correction[0]

        corrected_distorted_msk = torch.cat(
            [corrected_distorted_msk[shift:],
            corrected_distorted_msk[:shift]],
            dim=0)
    if position_correction[1] !=0:
        shift = position_correction[1]
        corrected_distorted_msk = torch.cat(
            [corrected_distorted_msk[:,shift:],
            corrected_distorted_msk[:,:shift]],
            dim=1)
    return corrected_distorted_msk

def get_distorted_ImageAndMask(trans, img, *msks):

    # format check
    assert all([msk.ndim == 2 for msk in msks])

    # apply transformation to both image and msk
    transformed = trans(image=img, masks = msks)
    distorted_img_shifted = transformed['image']
    distorted_msks_shifted = transformed['masks']

    #clean up the unexpected obsolete regions
    ## get area excluding border
    positive_region_map = label(distorted_msks_shifted[1])
    label_ids, label_cnts = np.unique(positive_region_map,return_counts=True)
    max_id = label_ids[1:][np.argmax(label_cnts[1:])]
    eligible_region_map = np.where(positive_region_map==max_id, 1, 0)

    distorted_msk_shifted = distorted_msks_shifted[0]*eligible_region_map
    distorted_msk_shifted = remove_small_objects(remove_small_holes(
        closing(opening(distorted_msk_shifted>0, disk(2)),disk(2))
        ))

    #find center shift as a results of transformation
    pos_idx = np.argwhere(msks[0]).squeeze()
    _raw_msk_center = (pos_idx.min(0)+pos_idx.max(0))//2
    pos_idx = np.argwhere(distorted_msk_shifted).squeeze()
    _distorted_msk_center = (pos_idx.min(0)+pos_idx.max(0))//2
    shift_spec = _distorted_msk_center -_raw_msk_center

    #shift the arrays so that the obj location aligns
    distorted_img = shift_array(distorted_img_shifted,shift_spec)
    distorted_msk = shift_array(distorted_msk_shifted,shift_spec)


    # remove relatively small regions
    distorted_msk_labeled = label(distorted_msk)
    unique_ids, id_cnts = np.unique(
        distorted_msk_labeled, return_counts = True)
    area_distribution = id_cnts[1:].astype('float')/id_cnts[1:].sum()

    for area_id, area_rate in zip(unique_ids[1:], area_distribution):
        if area_rate < 0.125:
            distorted_msk_labeled[distorted_msk_labeled==area_id] = 0

    distorted_msk_arr = distorted_msk_labeled>0

    return distorted_img, distorted_msk_arr

distorted_img_arr, distorted_msk_arr = get_distorted_ImageAndMask(
    shape_transformation,raw_img_arr,raw_msk_arr,raw_msk_arr.copy())
```

C.3. Fine-Tuning Experiment Details

Here we provide all details for the SAM fine-tuning experiments of Section 6. The loss function that we use is the sum of the Dice and cross-entropy losses (as implemented by MONAI [6]), optimized via AdamW [36] with a learning rate of 10^{-5} and a weight decay strength of 0.1. We use a batch size of 16 and fine-tune for 40 epochs on the k (image, mask) examples which were sampled from DIS-train using the given selection strategy (see Section 6). All parameters of SAM's mask decoder are fine-tuned, while the image encoder is frozen.

D. Additional Results

D.1. SAM Self-Attention Maps

In Fig. 13 we show examples of SAM's attention patterns for images with objects of high tree-likeness. These were computed via ground truth mask-guided attention rollout [2].

D.2. Mechanistic Relationships between Metrics and Model Architectural Choices

D.2.1. The Relationship of Vision Transformer Effective Patch Size with CPR

As was described in Sec. 4.1, in theory, when the patch size of an SFM's image encoder is small relative to the typical branch width of a tree-like object, patches can still contain internal pixels, but as patch size grows, CPR implies that boundary pixels dominate every patch, leaving no truly "interior" representation. Thus, CPR should directly interact with patch size: larger patches exacerbate the imbalance between boundary and interior content, *i.e.*, the typical CPR of objects will be higher. In this section, we will verify this behavior experimentally.

As the patch sizes of the SFMs (SAM, HQ-SAM and SAM 2) are fixed *a priori*, we propose to instead modify the *effective patch size* by altering the size of the objects themselves. We do this by zooming a given image in (cropping) our out, centered on the object mask, such that the length of the longer edge of the object's tight bounding box is of size p pixels (and similarly modifying the object mask accordingly). In the case of zooming out, we zero-pad the image to set it to the proper size for SFM input, 1024×1024. For example, if p is doubled (the object is made twice as large), the object will be covered by roughly four times as many patches, making the *effective* patch size halved, because more patches are required to cover the same object. In summary, higher p corresponds to smaller effective patch size.

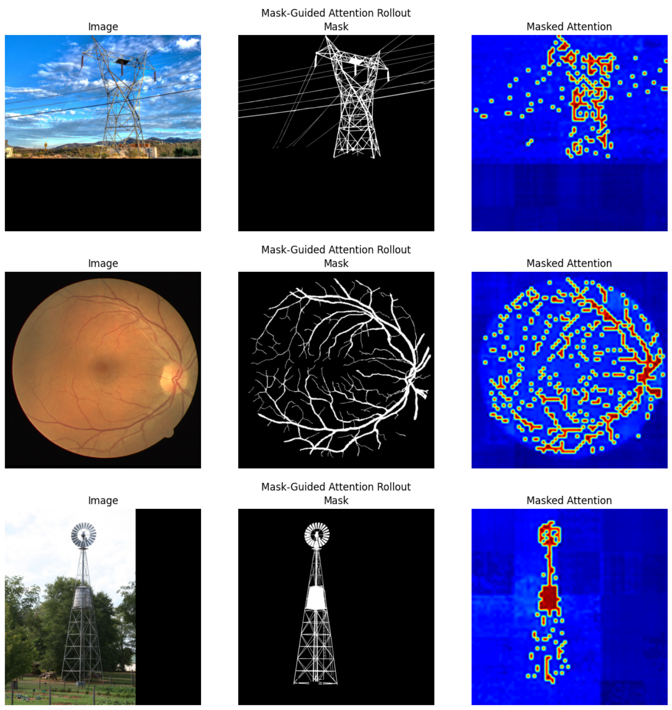
Image
Mask-Guided Attention Rollout
Mask
Masked Attention
Image
Mask-Guided Attention Rollout
Mask
Masked Attention
Image
Mask-Guided Attention Rollout
Mask
Masked Attention

Figure 13. Attention patterns (right column) for example images (left column) with objects of high tree-likeness (shown via ground truth masks in middle column).

In Fig. 14, we show how the performance (IoU) vs. object CPR curve on DIS is affected by different choices of $p \in [32, 64, 96, 160, 224, 352, 480, 736]$, for both variants of SAM. We note that we discard any datapoint where the zooming-in operation made the object grow outside of the image boundaries. We see that as was predicted, larger patch sizes (smaller p) results in objects typically having noticeably higher CPR (higher average CPR μ for a given p, as shown in the plot legends). Moreover, the higher typical CPR results in the model performance on a given object being typically worse.

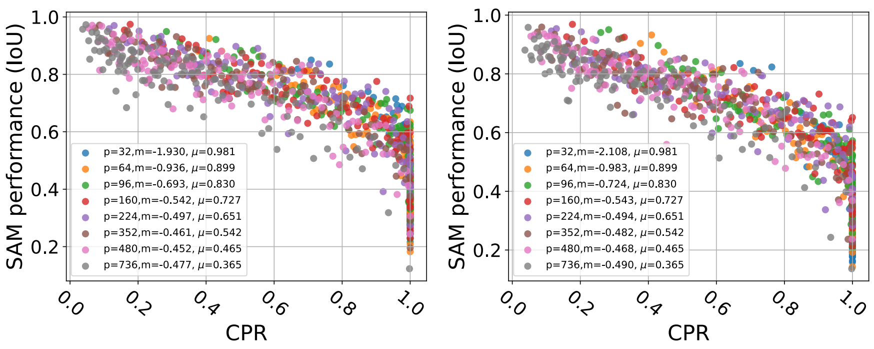

Figure 14. The effect on the relationship between segmentation performance (IoU) vs. CPR (object tree-likeness) of changing the model's effective patch size, increased with decreasing p, computed on DIS for the SAM ViT-H (left) and ViT-B (right) variants. Also shown are each plot's legend are average CPR values μ and estimated linear slope of IoU vs. CPR, for each p.

A potential mechanistic solution to this problem could be to use vision transformer models with smaller patch sizes to perform better on tree-like objects, yet this carries a high computational burden, because the size of the self-attention matrices grows quadratically with the patch size becoming smaller.

D.2.2. The Relationship of Vision Transformer Attention Map Fragmentation with CPR

As we mentioned in Section 4.1 and showed via a few examples in App. D.1, attention maps on tree-like objects typically appear to be fragmented and locally-focused, due to responding to local edges independently instead of forming clear global object representations. Here, we will test this empirically by determining if the correlation of the fragmentation of an input image's attention map with the tree-likeness of the object that the image contains. Here, the attention map for a given input image is defined via the attention rollout technique used in App. D.1.

We measure the fragmentation of a self-attention map $A \in [0,1]^{K\times K}$ via Moran's spatial autocorrelation index I [39]. Moran's I is lower for more fragmented/less clustered arrays, so we would expect theoretically that the CPR/tree-likeness of an object's mask is negatively correlated with the Moran's I of the model's computed attention map for the image containing the object.

To empirically measure the relationship between an SFM attention map's fragmentation and the tree-likeness of an object to be segmented, we measure the correlation of the Moran's I of an input image's self attention map with the CPR of the input image's associated mask label, computed for both SAM-H and SAM-B on DIS and all classes of iShape, using the same aggregation procedure as in the main experiments (Sec. 4.3.2). The results are shown in Table 8, with accompanying plots in Fig. 15. Overall, we indeed see a strong negative correlation between object CPR and self-attention map Moran's I: objects of higher tree-likeness correspond to more fragmented self-attention maps, as was indicated by our qualitative results in App. D.1.

SFM Model	**DIS**		**iShape (all classes)**	
	τ	ρ	τ	ρ
SAM-H	-0.72	-0.89	-0.76	-0.91
SAM-B	-0.59	-0.78	-0.72	-0.88

Table 8. Nonlinear (Spearman ρ, Kendall τ) correlations of self-attention map fragmentation (Moran's autocorrelation I) with object tree-likeness (CPR).

D.3. Correlation Between Metrics

In Fig. D.3, we show the relationship between our proposed tree-likeness metrics CPR and DoGD on all three datasets (with default hyperparameters and ViT-H SAM), quantified by correlations shown in Table 9.

			iShape							
	Synth.	**DIS**	antenna	branch	fence	hanger	log	wire	*all*	**Avg.**
Pearson r	-0.94	-0.67	-0.92	-0.93	-0.82	-0.81	-0.90	-0.66	-0.79	-0.83
Spearman ρ	-0.82	-0.70	-0.91	-0.93	-0.86	-0.76	-0.94	-0.79	-0.76	-0.83
Kendall τ	-0.62	-0.51	-0.75	-0.78	-0.68	-0.58	-0.79	-0.61	-0.57	-0.65

Table 9. Linear (Pearson r) and nonlinear (Spearman ρ, Kendall τ) correlations between CPR and DoGD on all evaluated datasets.

D.4. Effect of Object Line Thickness on Performance

In Section 4.1, we hypothesized that the noticeable SAM segmentation performance difference between retinal vessel images and satellite road images, two types of objects with branching structures, is due to the former objects typically having more dense and irregularly-spaced branching structures than the latter. However, it is important to consider the possibility that this is simply due to retinal vessels having thinner lines than road objects, which would help evaluate if the proposed tree-likeness metrics (CPR and DoGD) are actually characterizing the true underlying features that drive performance.

To test this, we performed the same synthetic data experiments as in Section 4.3.1, but applied transformations which thickened object lines, to see how this affected the overall relationship between measured object tree-likeness and segmentation performance. To do so, between steps 2 and 3 of the procedure used to generate these images (App. A.2), we:

1. Converted the object mask m_c to its skeleton, effectively resulting in a mask consisting of lines with 1-pixel widths.
2. Apply a dilation transformation with a radius of 4 pixels to this skeleton, resulting in each path/line structure in the mask being strictly 9 pixels wide.

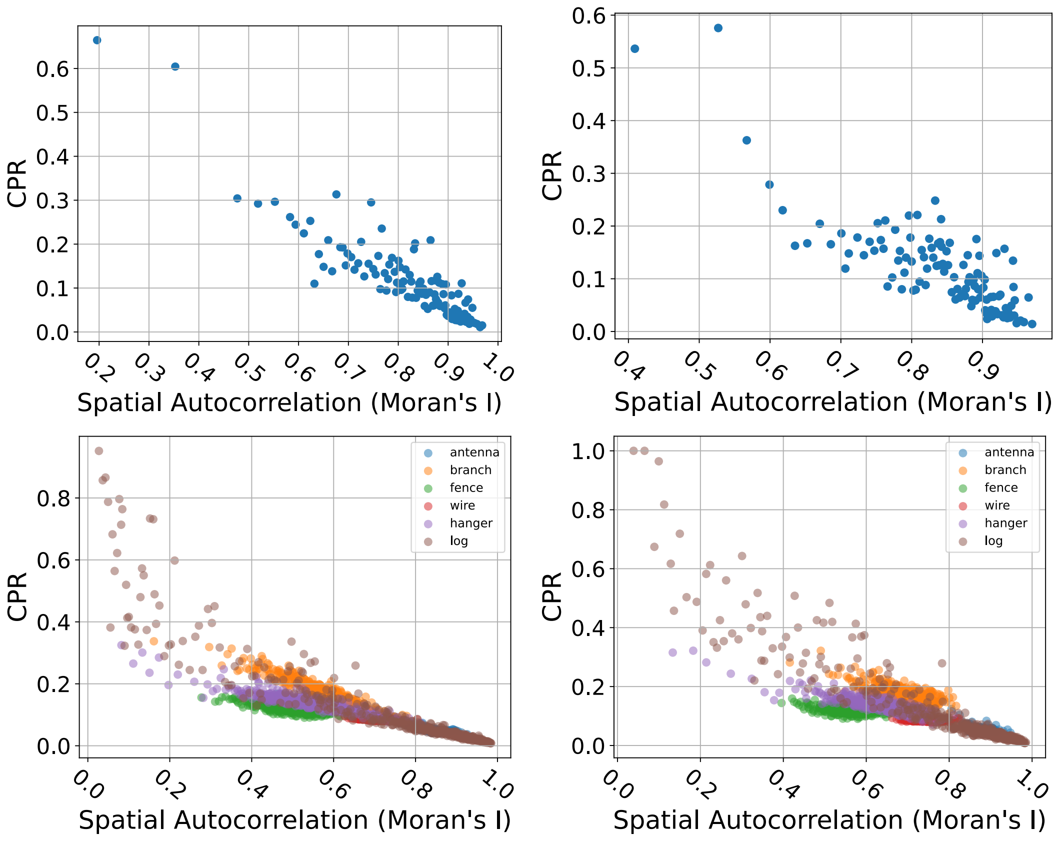

Figure 15. Object tree-likeness (CPR) vs. self-attention map fragmentation (Moran's autocorrelation I), on DIS (top row) and iShape (bottom row) for both ViT-H (left column) and ViT-B (right column) variants of SAM.

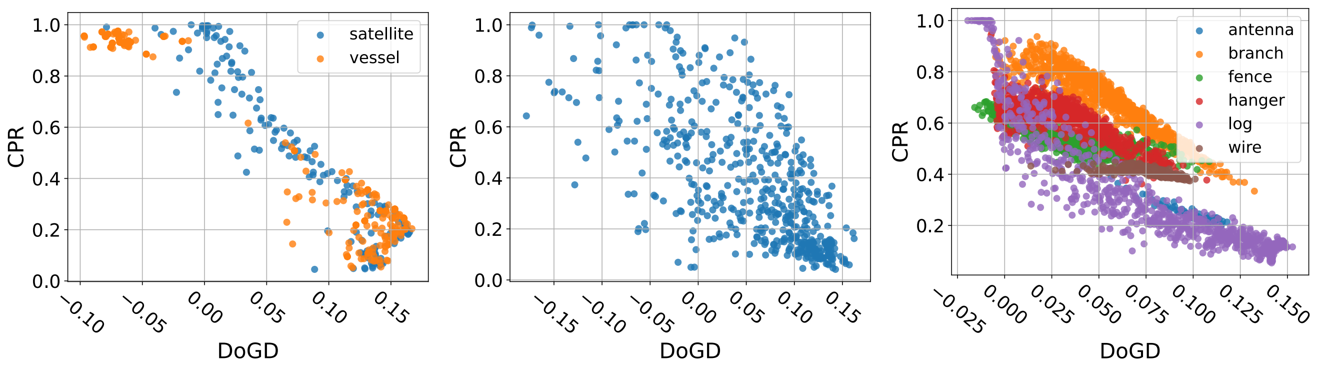

Figure 16. Relationship between tree-likeness metrics CPR and DoGD. From left to right, on synthetic data (Fig. 4), DIS, and iShape (Fig. 5).

We show the results of this procedure (how it affects the relationship between segmentation IoU and tree-likeness) in Fig. 17. We see that the result of IoU and object tree-likeness being correlated is mostly invariant to the object line thickening transformation. However, while the DoGD results are generally maintained because DoGD quantifies the deviation of filling-rate at different scales, we see that CPR experiences some degradation in its correlation with IoU for certain retinal vessel components. This is possibly due to the highly dense tree structures, originally thin with varying widths, forced to be similarly thickened and then possibly merged with each other, thereby reducing the overall CPR of the object. This may be a limitaiton of CPR in quantifying the tree or net-like geometric structure. Fortunately, such special cases where all object pixels fall around a fixed range of object skeletons are rare in realistic scenarios.

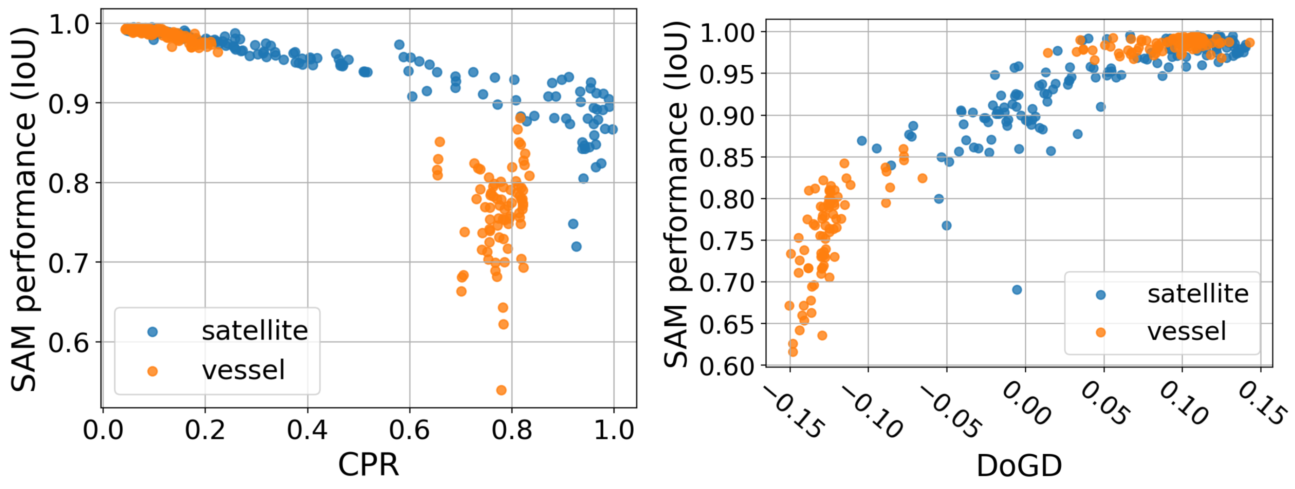

Figure 17. SAM segmentation IoU vs. object tree-likeness (CPR, left; DoGD, right) on the synthetic dataset with dilated object lines.

D.5. Performance vs. NST intensity for all SFMs

To supplement Fig. 7 right, we show the same results of segmentation IoU vs. NST intensity for all groups (controlled, altered, and mixed) for the additional SFMs (SAM ViT-B, SAM2 ViT-B+ and ViT-L, HQ-SAM ViT-H and ViT-B) in Fig. 18.

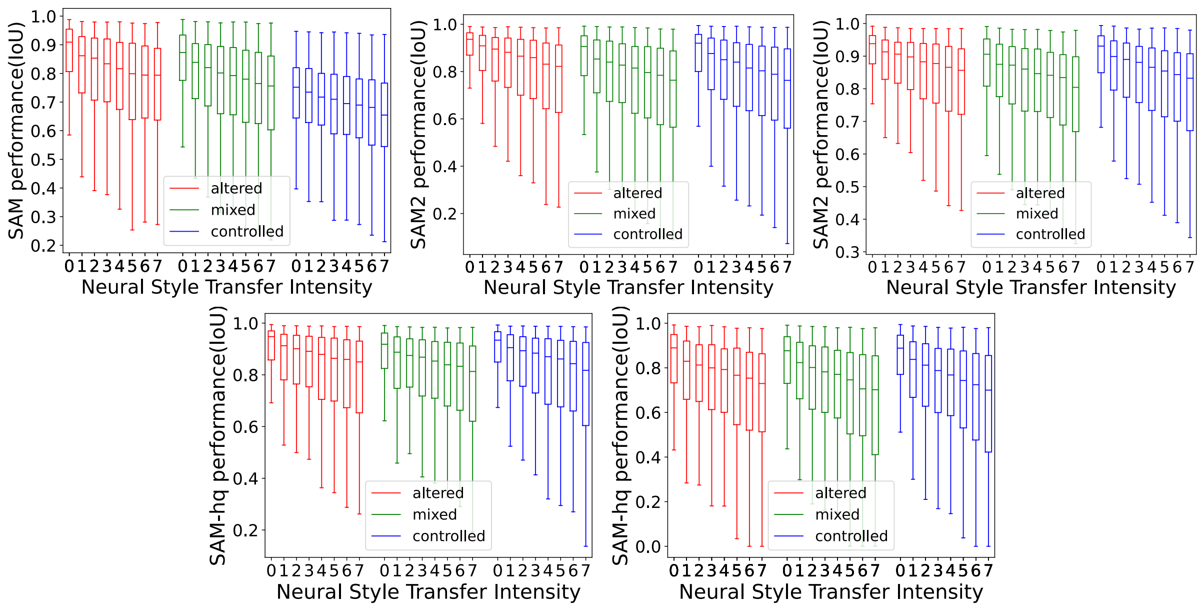

Figure 18. From left-to-right and top-to-bottom: segmentation IoU vs. NST intensity for all groups (controlled, altered, and mixed) for SAM ViT-B, SAM 2 ViT-B+, SAM 2 ViT-L, HQ-SAM ViT-H, and HQ-SAM ViT-B.

D.6. Comparison to Segmentation Models Trained From Scratch

We additionally evaluate training models from scratch on each dataset to see if the discovered relationships of performance with tree-likeness (CPR and DoGD) and textural separability metrics is indeed a weakness unique to segmentation foundation models (SFMs), or if such cases are simply universally challenging from a general standpoint, and cannot be solved by training a model from scratch for the specific dataset.

For DIS, we use the pretrained IS-Net model available directly from the original paper [42], obtained from `https://github.com/xuebinqin/DIS`. For iShape, we use the iShapeInst models from Wang et al. [54], obtained from `https://github.com/youngnuaa/iShapeInst`. We defer to those works for all architectural and training details. For Plittersdorf, no existing pretrained models could be found, so we trained a Mask R-CNN [22] on the Plittersdorf training set, fine-tuning from COCO weights (we could not train *completely* from scratch due to the relatively small size of this dataset), and tested on the test set. The Plittersdorf model utilized a ResNet-50-FPN backbone, with the `COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml` configuration of the Detectron2 [55] model zoo. For training, we used a batch size of 2, a learning rate of 2.5×10^{-4}, $30,000$ total training iterations, 300 warmup iterations, and a learning rate schedule that reduced the learning rate by factor of 0.1 at 60% and 80% of total training time.

We show all results in Fig. 19 for DIS, Fig. 20 and Table 10 for iShape, and Fig. 21 for Plittersdorf. There are several important takeaways when we compare these results of models trained from scratch to the behavior of the segmentation foundation models in the main text. First, in all cases, we see that the models perform noticeably better (in average IoU) than the SFMs, which makes sense given that they were trained specifically and only on these datasets, which is unsurprising. In the case of object tree-likeness (on DIS and iShape), we see that the sharpness (approximate slope) of the effect of CPR and DoGD on performance is far lower than compared to the SFM results in the main text (Sec. 4.3.2). In other words, these models typically perform much better on objects of high tree-likeness (*i.e.*, low CPR/high DoGD) than the SFMs, indicating that they do not share the same "failure mode" behavior. The correlations between IoU and tree-likeness are lower as well for these models trained from scratch (comparing Fig. 5 for the SFMs to Figs. 19 and 20 and Table 10 here). We see similar, if not stronger results for textural separability: there is, on average, little to no correlation of performance with separability on both iShape (Fig. 20 and Table 10) and Plittersdorf (Fig. 21), compared to the noticeably stronger effect of an object's textural separability on SFM performance (Fig. 8).

We hypothesize that any remaining correlation of performance with these object characteristics (tree-likeness and textural separability) is simply because as shown in the figures, objects with higher tree-likeness or lower textural separability are simply less common in the dataset, so models trained from scratch will perform slightly worse on such cases compared to on cases which have more examples to learn from. Overall, these results support that SFMs are uniquely susceptible to objects of high tree-likeness or low textural separability compared to models trained from scratch on datasets that have these types of uncommon objects.

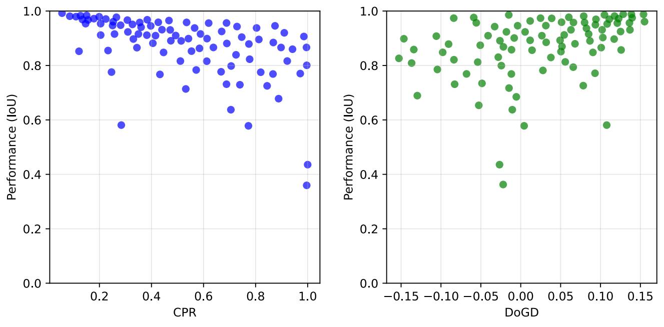

Figure 19. For IS-Net model trained from scratch on DIS: relationship of performance (IoU on the DIS validation set) with tree-likeness metrics: CPR, left and DoGD, right. For CPR, Spearman correlation $\rho = -0.61$ and Kendall $\tau = -0.45$. For DoGD, $\rho = 0.46$ and $\tau = 0.32$.

D.7. All results on MOSE

In Fig. 22 we show the corresponding plots for the SFM performance (IoU) vs. tree-likeness (table in Fig. 5) and textural separability (table in Fig. 8 relationship results on the MOSE dataset. These are for the same SFM architectures as shown in the plots of Figs. 5) and 8).

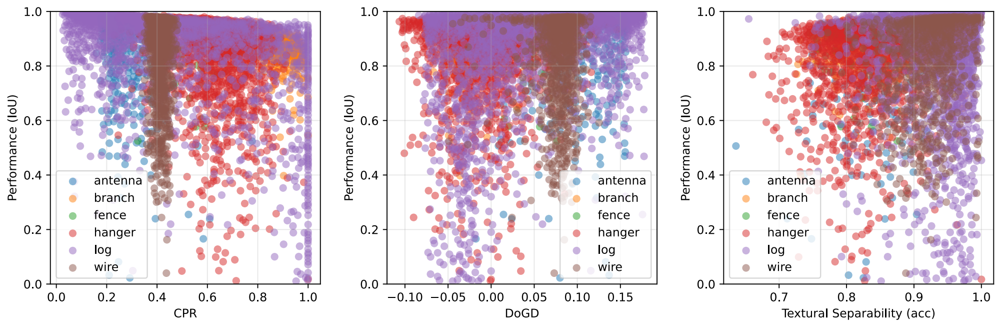

Figure 20. For iShapeInst models trained from scratch on iShape: relationship of performance (IoU) with tree-likeness metrics of CPR (left) and DoGD (center), and with textural separability (right). See Table 10 for numerical correlation results.

		antenna	branch	fence	hanger	log	wire	**Avg.**
CPR	Spearman ρ	-0.31	-0.78	-0.61	-0.36	-0.53	-0.08	**-0.45**
	Kendall τ	-0.21	-0.59	-0.45	-0.25	-0.38	-0.05	**-0.32**
DoGD	Spearman ρ	0.08	0.55	0.39	0.16	0.36	-0.01	**0.26**
	Kendall τ	0.06	0.39	0.26	0.11	0.25	-0.01	**0.18**
Textural Separability	Spearman ρ	0.34	0.40	0.51	0.43	0.10	0.22	**0.33**
	Kendall τ	0.23	0.27	0.35	0.30	0.07	0.15	**0.23**

Table 10. For iShape, nonlinear (Spearman ρ, Kendall τ) correlations between performance of models trained from scratch (IoU) and both tree-likeness metrics (CPR and DoGD) as well as textural separability.

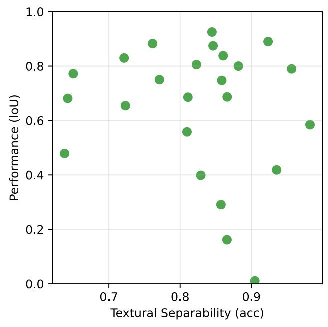

Figure 21. For Mask R-CNN model trained on Plittersdorf: relationship of performance (IoU) with textural separability (Spearman correlation $\rho = -0.07$ and Kendall $\tau = -0.05$.)

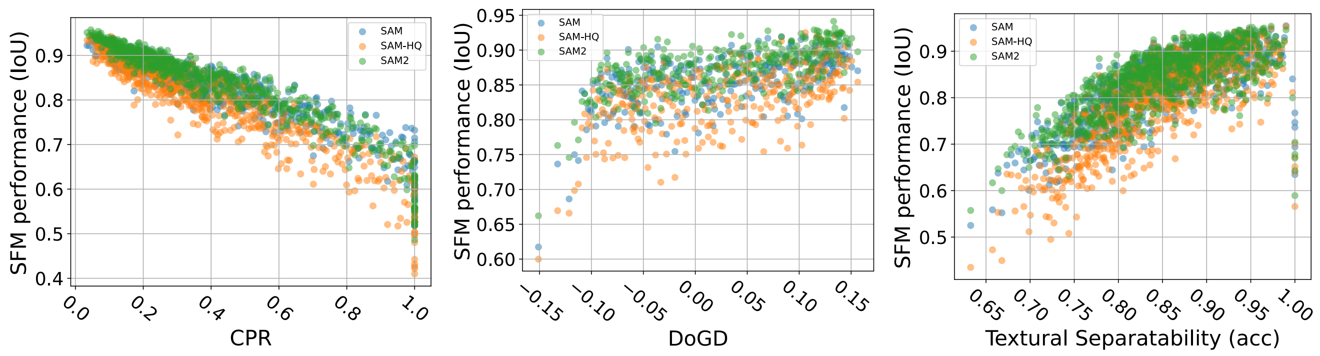

Figure 22. Segmentation IoU vs. object tree-likeness (CPR, left, and DoGD, center) and textural separability (right) for the MOSE dataset, for all three evaluated SFMs.