

FMS²: Unified Flow Matching for Segmentation and Synthesis of Thin Structures

Supplementary Material

Babak Asadi (✉), Peiyang Wu, Mani Golparvar-Fard, Viraj Shah, Ramez Hajj

University of Illinois at Urbana-Champaign

{basadi2, peiyang7, mgolpar, vjshah3, rhajj}@illinois.edu

Contents

A Dataset details	1
B SegFlow additional analysis	2
B.1 Run-to-run stability	2
B.2 Ablation of transport paths for SegFlow	2
B.3 Iterative inference fairness and latency	3
B.4 Additional qualitative results for SegFlow	3
B.5 Failure cases of SegFlow	3
C SynFlow additional analysis	5
C.1 Controllable mask generator	5
C.2 Effect of mask injection location	6
C.3 Latent-diffusion baselines	10

A Dataset details

We use three road crack benchmarks and two vascular benchmarks that span diverse imaging modalities and foreground sparsity. Table S1 summarizes the original image resolutions, the number of images used, and the mean foreground proportion (crack or vessel pixels). All images and masks are standardized to 256×256 inputs before being fed to SegFlow and SynFlow, using dataset-specific preprocessing (either direct resizing or cropping).

The three crack datasets cover different road imaging setups: CrackTree260 [7, 9] uses an area-array visible-light camera, CrackLS315 [7] uses a line-scan laser system, and Crack500 [10] consists of handheld phone images with more varied appearance. In all cases cracks occupy well under 3% of pixels. Following the main paper, we extract 500 crack-centered patches of size 256×256 per dataset and split them into 80/10/10 train/validation/test subsets with a fixed random seed.

The two vessel datasets probe transfer beyond roads. For DRIVE [6], we use 20 color retinal fundus photographs (images 21–40). We split them into training (21–32), validation (33–34), and testing (35–40). For each image/mask pair, we first take a centered 512×512 crop and then split it into four non-overlapping 256×256 patches. DRIVE vessels cover about 8.63% of pixels. XCAD [3] provides 1621 unlabeled angiography frames and 126 labeled 512×512 frames; we use only the labeled images, splitting them into 100 train and 26 test images as in the main paper. Coronary vessels cover 5.72% of pixels on average. For XCAD, images and masks are resized to 256×256 for our experiments.

Taken together, these five datasets span extremely sparse cracks (0.25–2.84%) and denser but still thin vascular trees (≈ 5.7 –8.6%), while sharing the requirement of preserving thin-structure topology.

Table S1: Summary of benchmark datasets used in this study. Resolution refers to the original released data. All images are cropped or resized to 256×256 for training and evaluation. Foreground proportion is the mean percentage of crack or vessel pixels computed from the original masks.

Dataset	Modality	Resolution	# Images	Foreground (%)
CrackTree260	Road RGB	$[600, 720] \times [800, 960]$	260	0.46
CrackLS315	Road laser	512×512	315	0.25
Crack500	Road RGB	1500×2000	500	2.84
DRIVE	Retinal fundus	565×584	20	8.63
XCAD	Coronary X-ray angiography	512×512	126	5.72

B SegFlow additional analysis

B.1 Run-to-run stability

Table S2 reports run-to-run mean $\pm$ standard deviation for SegFlow and the two next-best methods across the five benchmarks. SegFlow shows low variation across datasets, supporting the robustness of the reported main-paper trends.

Table S2: Run-to-run mean $\pm$ standard deviation for mIoU / μ^{err} on SegFlow and the two next-best methods.

Dataset	SegFlow (Ours)	FlowSDF	SCSegamba
CRACK500	.612 $\pm$.0045 / 24.9 $\pm$ 1.1	.531 $\pm$.0097 / 36.6 $\pm$ 1.3	.549 $\pm$.0110 / 24.8 $\pm$ 0.5
CrackTree260	.645 $\pm$.0042 / 76.1 $\pm$ 2.5	.577 $\pm$.0175 / 157.8 $\pm$ 6.7	.437 $\pm$.0132 / 202.8 $\pm$ 3.7
CrackLS315	.442 $\pm$.0047 / 96.3 $\pm$ 0.7	.221 $\pm$.0147 / 137.2 $\pm$ 3.3	.386 $\pm$.0040 / 94.1 $\pm$ 1.3
DRIVE	.653 $\pm$.0040 / 49.8 $\pm$ 1.7	.635 $\pm$.0013 / 130.3 $\pm$ 2.9	.605 $\pm$.0148 / 65.6 $\pm$ 9.0
XCAD	.642 $\pm$.0111 / 10.6 $\pm$ 0.7	.592 $\pm$.0240 / 18.1 $\pm$ 2.5	.564 $\pm$.0196 / 23.4 $\pm$ 2.0

B.2 Ablation of transport paths for SegFlow

We tested nonlinear transport alternatives on CRACK500 (Table S3). The default linear image $\rightarrow$ mask path performs best overall, while variance-preserving CFM [1], rectified flow [2], and asymmetric-noise paths reduce performance. This suggests that the simple image $\rightarrow$ mask path is already effective for thin structures.

Table S3: Ablation of transport paths for SegFlow on CRACK500. Higher is better for mIoU, F1, and cIDice; lower is better for μ^{err} .

Path	mIoU $\uparrow$	F1 $\uparrow$	cIDice $\uparrow$	$\mu^{\text{err}}\downarrow$
Linear	.612	.753	.832	27.868
VP-CFM [1]	.592	.736	.822	27.093
Rectified [2]	.580	.725	.821	27.953
Asym.-noise	.487	.636	.739	40.221

B.3 Iterative inference fairness and latency

SegFlow’s multi-step ODE is native flow-matching inference, not post-hoc refinement. Table S4 shows that SegFlow is far lighter than iterative generative baselines, while non-iterative models do not match its accuracy.

Table S4: Computational comparison of SegFlow and representative baselines. Parameters and GFLOPs are reported per model; “Steps” denotes the number of inference steps for iterative generative methods.

Method	Params (M)	GFLOPs	Steps
SegFlow (Ours)	2.96	325.8	5–10
FlowSDF	160.6	9082.0	20
SemFlow	943.2	4289.8	25
Mask2Former	215.5	151.6	N/A

B.4 Additional qualitative results for SegFlow

Figure S1 shows additional qualitative segmentation results obtained with SegFlow-Base on all five datasets.

B.5 Failure cases of SegFlow

Although SegFlow generally preserves thin connected structures, Figure S2 illustrates representative failure modes. Errors mainly occur when structures are extremely thin, low contrast, partially occluded, or visually confused with strong background patterns such as road markings, which can lead to false negatives and local discontinuities.

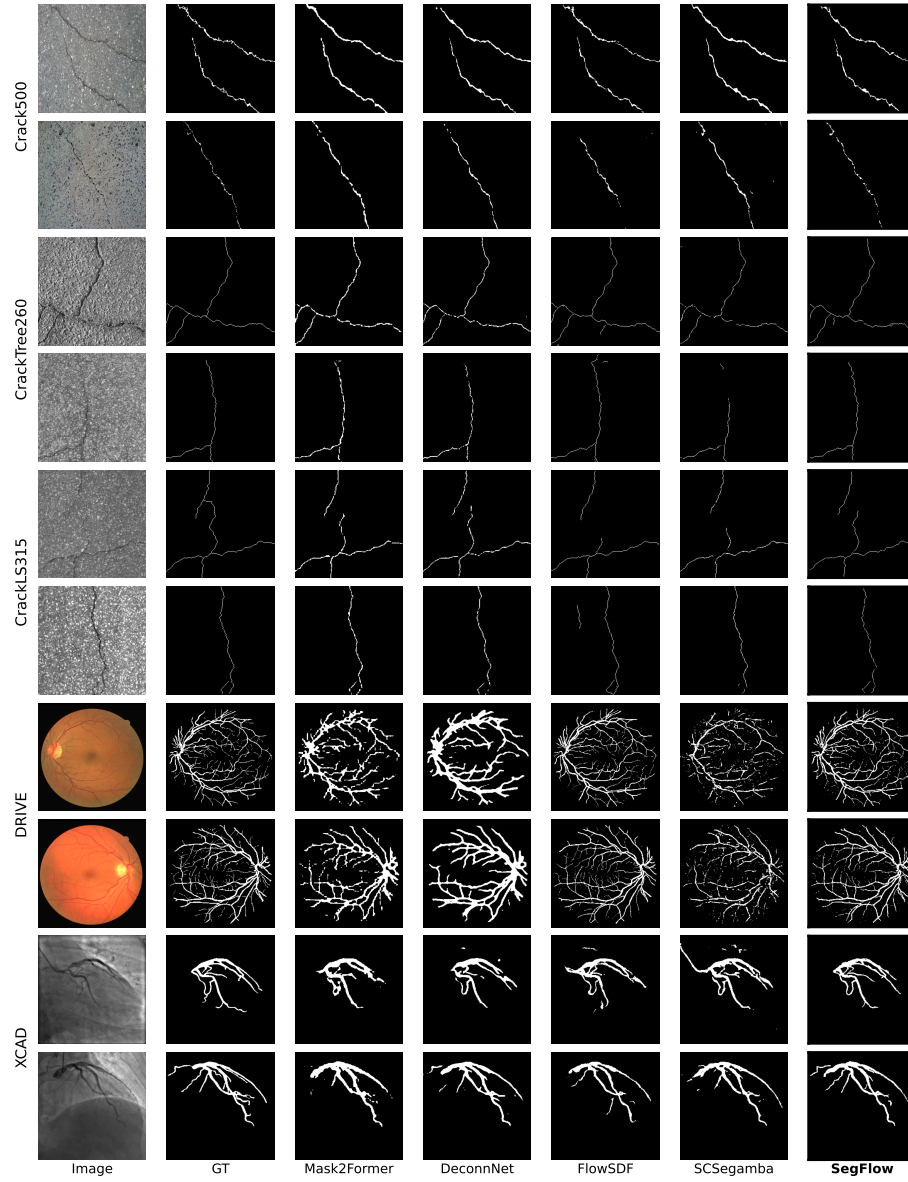


Fig. S1: Qualitative comparison of segmentation results on five datasets (Crack500, CrackTree260, CrackLS315, DRIVE, and XCAD). For each dataset, two test examples are shown where SegFlow produces sharper, more topology-consistent masks than state-of-the-art thin-structure segmentation models developed under different design philosophies, complementing the quantitative gains reported in Table 1 of the main paper.

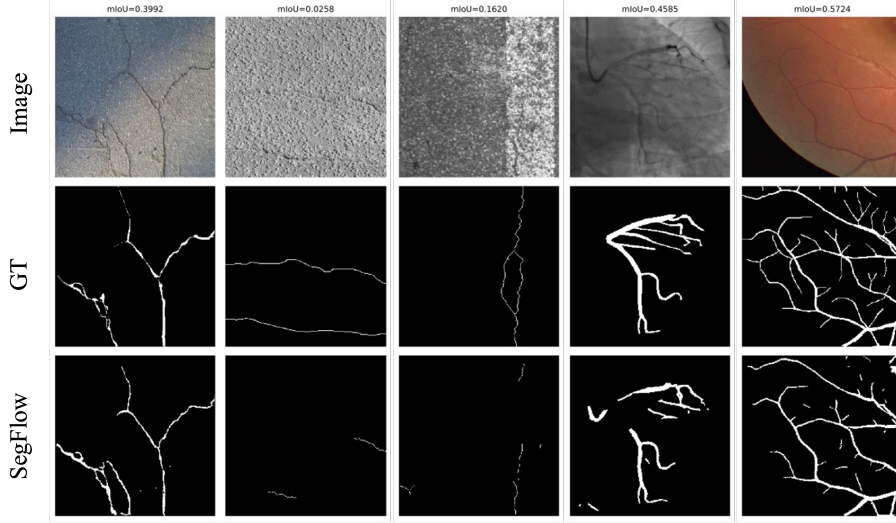


Fig. S2: Representative failure cases in crack/vessel segmentation. Top: input; middle: ground truth; bottom: prediction (mIoU shown above each example). False negatives occur primarily for extremely thin/low-contrast structures or when the target is interrupted/occluded by strong background patterns (e.g., road paint/markings), resulting in missing or disconnected segments.

C SynFlow additional analysis

C.1 Controllable mask generator

SynFlow can be conditioned either on ground-truth masks from the training set or on synthetic masks sampled from a small conditional flow-matching model attached to the SynFlow branch. This *controllable mask generator* broadens the mask distribution seen by SynFlow while exposing an explicit control over the fraction of crack (or vessel) pixels.

Given a binary mask $M \in \{0, 1\}^{H \times W}$, we compute its foreground ratio

$$\rho(M) = \frac{1}{HW} \sum_{i,j} M_{ij}, \quad (1)$$

and assign a discrete control label y via a binning function

$$y = \mathcal{B}(\rho(M)) \in \{0, \dots, C - 1\}, \quad (2)$$

where $\mathcal{B}$ partitions ρ into C sparsity bins. At inference, selecting a target label y^* specifies a desired foreground proportion; sampling with y^* yields masks in the corresponding sparsity bin, and passing them through SynFlow produces paired image-mask samples at that controlled crack (or vessel) proportion.

The mask generator is a U-Net that operates on single-channel masks (1-channel input and output) and predicts a time-indexed velocity field $v_\theta(x, t, y)$

over masks, conditioned on time t and the sparsity label y . Conditioning on y is implemented by augmenting the standard sinusoidal time embedding. Let $\psi(t) \in \mathbb{R}^d$ denote the time embedding and let $E : \{0, \dots, C - 1\} \rightarrow \mathbb{R}^d$ be a learnable embedding table. We form a joint conditioning vector

$$z = \phi(\psi(t) + E(y)) \in \mathbb{R}^d, \quad (3)$$

where ϕ is a small MLP. In each residual block, z is linearly projected and added to the hidden activations, so the control label modulates features throughout the U-Net and steers the generator toward masks with the requested crack proportion.

Consistent with SynFlow, the mask generator uses classifier-free guidance. During training, the label y is randomly replaced by a null token $\emptyset$ for a subset of steps. At sampling time, we combine the conditional and unconditional velocity fields as

$$\hat{v}_\theta(x, t, y) = v_\theta(x, t, \emptyset) + \omega(v_\theta(x, t, y) - v_\theta(x, t, \emptyset)), \quad (4)$$

with guidance weight ω . The guided field $\hat{v}_\theta$ is then used in the ODE integrator. Increasing ω yields masks that more tightly match the target sparsity bin while retaining diversity across noise seeds.

Figure S3 shows representative examples of real training masks and samples produced by the controllable mask generator at different sparsity bins.

It should also be noted that, for crack datasets, we further mimic the evolution of damage by applying a second-stage geometric augmentation on top of the generated masks. Starting from each synthetic mask, we perform controlled region growing (propagation) and skeleton-guided branch extension, followed by radius-based thinning and thickening. This yields families of crack geometries with varying width and length on the same background, better reflecting how crack extent and width influence infrastructure condition assessment (see Figure S4).

C.2 Effect of mask injection location

We also study two SynFlow variants that differ only in how the conditioning mask is injected into the U-Net backbone: (i) injecting the mask features only in the decoder, and (ii) injecting the mask features in both encoder and decoder layers. All other settings follow the main-paper SynFlow configuration.

Table S5 reports FID and KID for image synthesis on five datasets for these two variants. Injecting masks in both encoder and decoder consistently yields lower FID and KID across most datasets, so this is the configuration used for all SynFlow results in the main paper.

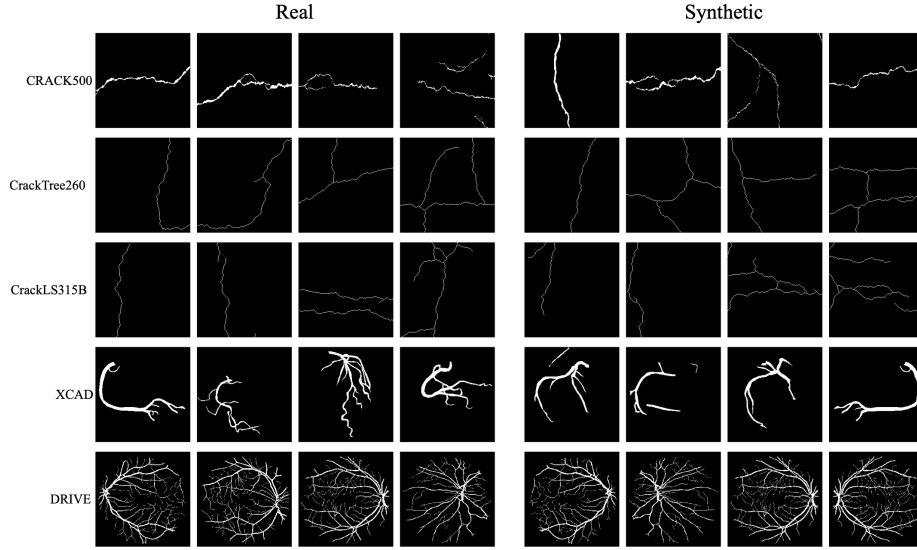


Fig. S3: Examples from the controllable mask generator. For each dataset, we show real training masks (left) and masks sampled from different sparsity bins (right).

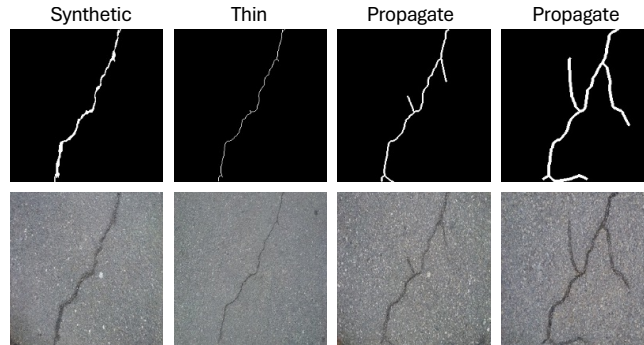


Fig. S4: Controllable mask generator with SynFlow rendering. **Top:** base mask plus thin, propagated, and multi-branch variants. **Bottom:** SynFlow outputs preserving topology and texture.

Table S5: Effect of mask injection location in SynFlow. “Decoder-only” applies mask conditioning only in decoder layers, whereas “Encoder+Decoder” injects masks in both encoder and decoder feature maps. We report FID and KID ($\times 10^3$); lower is better. Encoder+Decoder is adopted for all SynFlow results.

Dataset	Injection	FID↓	KID $\times 10^3$ ↓
CRACK500	Decoder-only	30.18	13.75
CRACK500	Encoder+Decoder	18.36	0.53
CrackTree260	Decoder-only	23.25	6.07
CrackTree260	Encoder+Decoder	20.14	3.36
CrackLS315	Decoder-only	23.53	7.98
CrackLS315	Encoder+Decoder	19.81	4.53
DRIVE	Decoder-only	8.79	-7.50
DRIVE	Encoder+Decoder	3.04	-9.41
XCAD	Decoder-only	64.97	7.24
XCAD	Encoder+Decoder	40.13	-0.07

Figures S5–S9 provide qualitative comparisons for each dataset. Each panel shows, from top to bottom: (1) input mask, (2) real image, (3) SynFlow image with encoder+decoder mask injection, and (4) SynFlow image with decoder-only injection. Encoder+decoder injection produces sharper crack/vessel traces and fewer topology breaks, consistent with the quantitative trends in Table S5.

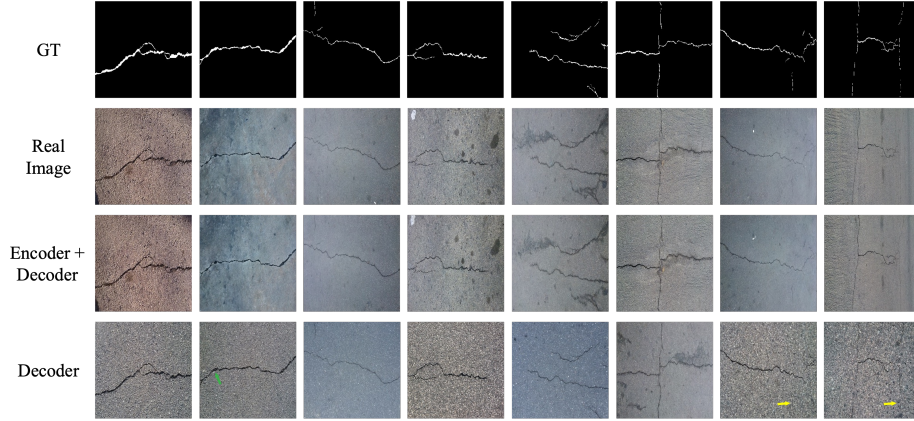


Fig. S5: SynFlow image synthesis on **Crack500**. Encoder+decoder mask injection yields sharper and more faithful crack rendering than decoder-only conditioning.

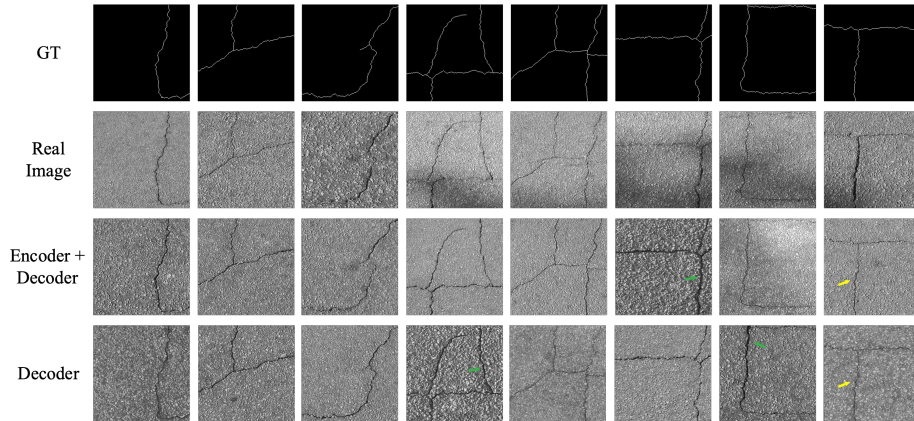


Fig. S6: SynFlow image synthesis on **CrackTree260**. Encoder+decoder mask injection yields sharper and more faithful crack rendering than decoder-only conditioning.

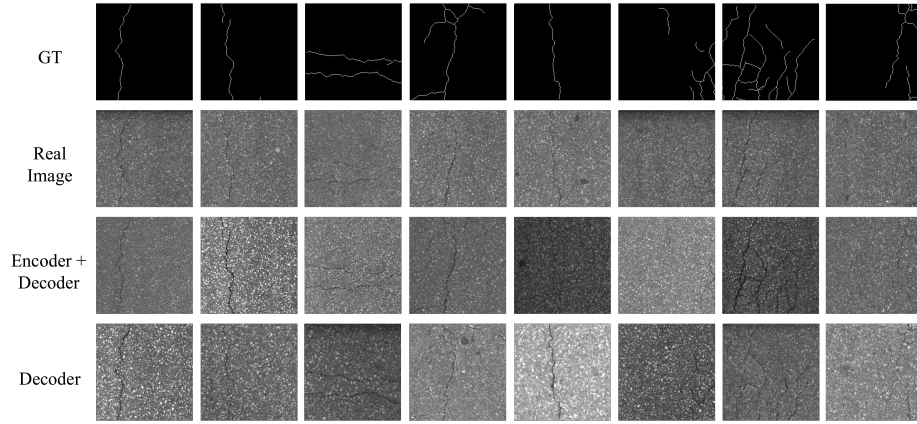


Fig. S7: SynFlow image synthesis on **CrackLS315**. Encoder+decoder mask injection better preserves continuous, topology-fragile crack traces than decoder-only conditioning.

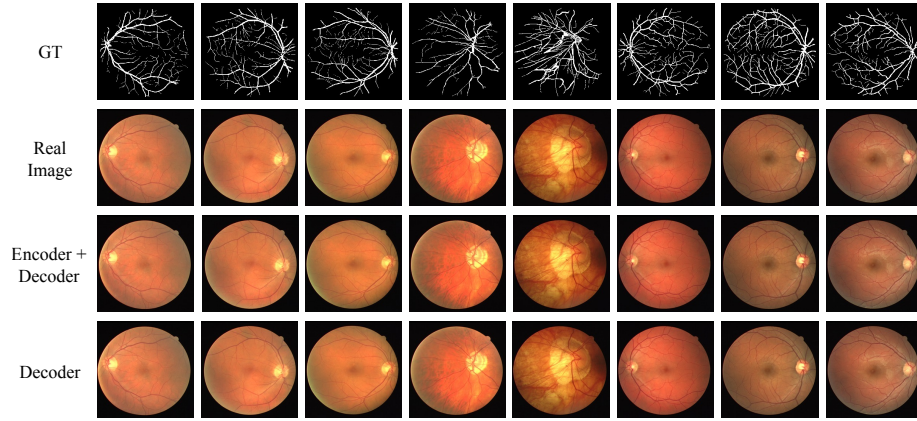


Fig. S8: SynFlow image synthesis on **DRIVE**. Encoder+decoder mask injection produces crisper and more continuous vessel trees than decoder-only conditioning.

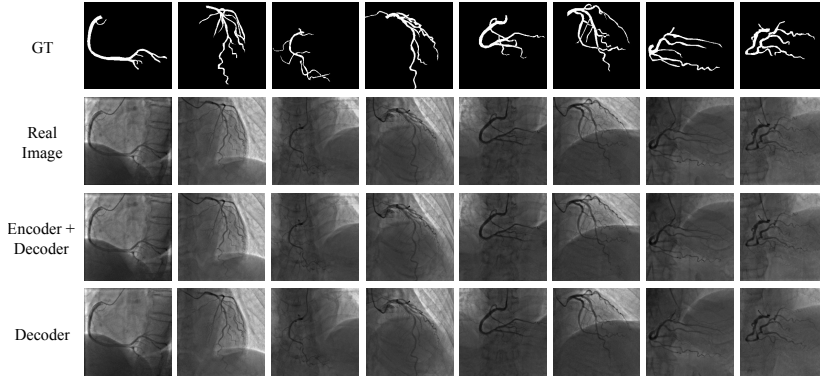


Fig. S9: SynFlow image synthesis on **XCAD**. Encoder+decoder mask injection better preserves thin coronary branches and reduces spurious background responses.

C.3 Latent-diffusion baselines

Table S6 and Figure S10 compare SynFlow with latent-diffusion baselines including LDM [5], ControlNet [8], and T2I-Adapter [4]. SynFlow better preserves thin-structure geometry, while latent-diffusion baselines often show blurred filaments, weak junction connectivity, and mask-image drift. For the quantitative results, generated images are manually annotated and compared against the original conditioning masks.

Table S6: Comparison of SynFlow with latent-diffusion mask-conditioned synthesis baselines. Generated images are manually annotated and compared against the conditioning masks. Higher is better for mIoU, F1, and cIDice; lower is better for μ^{err} .

Method	Venue	mIoU $\uparrow$	F1 $\uparrow$	cIDice $\uparrow$	$\mu^{\text{err}}\downarrow$
LDM	CVPR 2022	.329	.493	.575	6.833
ControlNet	ICCV 2023	<u>.467</u>	<u>.632</u>	<u>.738</u>	8.667
T2I-Adapter	AAAI 2024	.413	.574	.676	<u>5.917</u>
SynFlow	Ours	.561	.717	.852	4.583

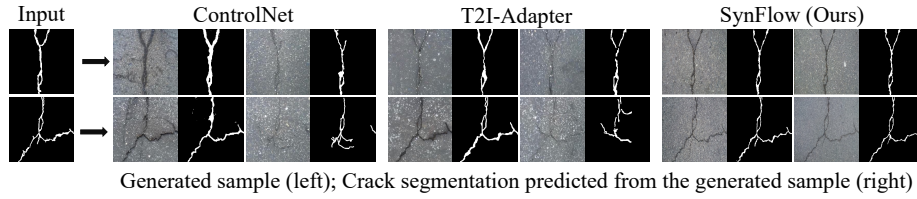


Fig. S10: Qualitative comparison of SynFlow with latent-diffusion baselines. SynFlow better follows the conditioning mask and preserves thin connected structures, whereas diffusion baselines can blur filaments or drift from the target geometry.

References

1. Albergo, M., Boffi, N.M., Vanden-Eijnden, E.: Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research* **26**(209), 1–80 (2025)
2. Liu, X., Gong, C., Liu, Q.: Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003* (2022)
3. Ma, Y., Hua, Y., Deng, H., Song, T., Wang, H., Xue, Z., Cao, H., Ma, R., Guan, H.: Self-supervised vessel segmentation via adversarial learning. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 7536–7545 (2021)
4. Mou, C., Wang, X., Xie, L., Wu, Y., Zhang, J., Qi, Z., Shan, Y.: T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. vol. 38, pp. 4296–4304 (2024)
5. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-Resolution Image Synthesis with Latent Diffusion Models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 10684–10695 (2022). <https://doi.org/10.1109/CVPR52688.2022.01046>
6. Staal, J., Abramoff, M.D., Niemeijer, M., Viergever, M.A., van Ginneken, B.: Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging* **23**(4), 501–509 (2004). <https://doi.org/10.1109/TMI.2004.825627>
7. Zhang, L., Yang, F., Zhang, Y.D., Zhu, Y.J.: Road crack detection using deep convolutional neural network. In: *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. pp. 3708–3712 (2016)
8. Zhang, L., Rao, A., Agrawala, M.: Adding Conditional Control to Text-to-Image Diffusion Models. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. pp. 3836–3847 (2023). <https://doi.org/10.1109/ICCV.2023.03885>
9. Zou, Q., Cao, Y., Li, Q., Mao, Q., Wang, S.: Cracktree: Automatic crack detection from pavement images. *Pattern Recognition Letters* **33**(3), 227–238 (2012). <https://doi.org/10.1016/j.patrec.2011.11.004>
10. Zou, Q., Zhang, Z., Li, Q., Qi, X., Wang, Q., Wang, S.: Deepcrack: Learning hierarchical convolutional features for crack detection. *IEEE Transactions on Image Processing* **28**(3), 1498–1512 (2019). <https://doi.org/10.1109/TIP.2018.2878966>