SCSegamba: Lightweight Structure-Aware Vision Mamba for Crack Segmentation in Structures

Supplementary Material

7. Details of SASS and Ablation Experiments

As described in Subsection 3.3, the SASS strategy enhances semantic capture in complex crack regions by scanning texture cues from multiple directions. SASS combines parallel snake and diagonal snake scans, aligning the scanning paths with the actual extension and irregular shapes of cracks, ensuring comprehensive capture of texture information.

To evaluate the necessity of using four scanning paths in SASS, we conducted ablation experiments with different path numbers across various scanning strategies on multi-scenario dataset TUT. As listed in Table 6, all strategies performed significantly better with four paths than with two, likely because four paths allow SAVSS to capture finer crack details and topological cues. Notably, aside from SASS, the diagonal snake-like scan consistently achieved the second-best results, with two-path configurations yielding F1 and mIoU scores 0.48% and 0.45% higher than the diagonal unidirectional scan. This indicates that the diagonal snake-like scan provides more continuous semantic information, enhancing segmentation. Importantly, our proposed SASS achieved the best results with both two-path and four-path setups, demonstrating its effectiveness in capturing diverse crack topologies.

To clarify the implementation of our proposed SASS, we present its execution process in Algorithm 1.

	N	ODS	OIS	P	R	F1	mIoU
Parallel	2	0.8032	0.8126	0.7994	0.8474	0.8231	0.8365
	4	0.8123	0.8184	0.8146	0.8523	0.8330	0.8427
PaSna	2	0.8035	0.8124	0.8062	0.8458	0.8258	0.8369
	4	0.8102	0.8162	0.8219	0.8365	0.8291	0.8408
Diag	2	0.8080	0.8166	0.8058	0.8496	0.8271	0.8408
	4	0.8091	0.8148	0.8225	0.8417	0.8320	0.8410
DigSna	2	0.8094	0.8162	0.8185	0.8470	0.8325	0.8413
	4	0.8153	0.8215	0.8237	0.8497	0.8365	0.8451
SASS	2	0.8130	0.8192	0.8196	0.8478	0.8335	0.8430
	4	**0.8204**	**0.8255**	**0.8241**	**0.8545**	**0.8390**	**0.8479**

Table 6. Ablation study on the number of paths in different scanning strategies. N represents the number of paths. For two-path scans, SASS uses the first parallel snake and diagonal snake scans, while other methods use the first two paths. Best results are in bold, and second-best results are underlined.

$\alpha : \beta$	ODS	OIS	P	R	F1	mIoU
BCE	0.8099	0.8151	0.8207	0.8457	0.8330	0.8414
Dice	0.8022	0.8072	0.8038	0.8430	0.8231	0.8358
5 : 1	0.8125	0.8168	0.8207	0.8432	0.8319	0.8428
4 : 1	0.8144	0.8184	0.8217	0.8442	0.8328	0.8437
3 : 1	0.8180	0.8229	**0.8293**	0.8436	0.8364	0.8463
2 : 1	0.8098	0.8152	0.8204	0.8392	0.8297	0.8408
1 : 1	0.8123	0.8184	0.8141	0.8507	0.8320	0.8423
1 : 2	0.8152	0.8214	0.8210	0.8484	0.8345	0.8443
1 : 3	0.8109	0.8163	0.8226	0.8396	0.8310	0.8418
1 : 4	0.8133	0.8185	0.8163	0.8515	0.8336	0.8433
1 : 5	**0.8204**	**0.8255**	0.8241	**0.8545**	**0.8390**	**0.8479**

Table 7. Sensitivity analysis experiments with different α and β ratios. Best results are in bold, and second-best results are underlined.

8. Details of Objective Function and Analysis

The calculation formulas for BCE [29] loss and Dice [43] loss are as follows:

$$L_{Dice} = 1 - \frac{2\sum_{j=1}^{M} p_j \hat{p}_j + \epsilon}{\sum_{j=1}^{M} p_j + \sum_{j=1}^{M} \hat{p}_j + \epsilon} \tag{20}$$

$$L_{BCE} = -\frac{1}{N}\left[p_j \log(\hat{p}_j) + (1 - p_j)\log(1 - \hat{p}_j)\right] \tag{21}$$

where M denotes the number of samples, p_j is the ground truth label for the j-th sample, $\hat{p}_j$ is the predicted probability for the j-th sample, ϵ is a small constant.

In equation 16, the ratio of α to β is set to 1:5. This is the optimal ratio of α and β selected after experimenting with various hyperparameter settings on multi-scenario dataset. As listed in Table 7, setting the α to β ratio at 1:5 yields the best performance, with improvements of 0.65% in F1 and 0.55% in mIoU over the 1:2 ratio. This suggests that balancing Dice and BCE loss at a 1:5 ratio helps the model better distinguish background pixels from the few crack region pixels, thereby enhancing performance.

9. Visualisation Comparisons

To visually demonstrate the advantages of SCSegamba, we present detailed visual results in Figure 7. For the Crack500 [56], DeepCrack [35], and CrackMap [22] datasets, which

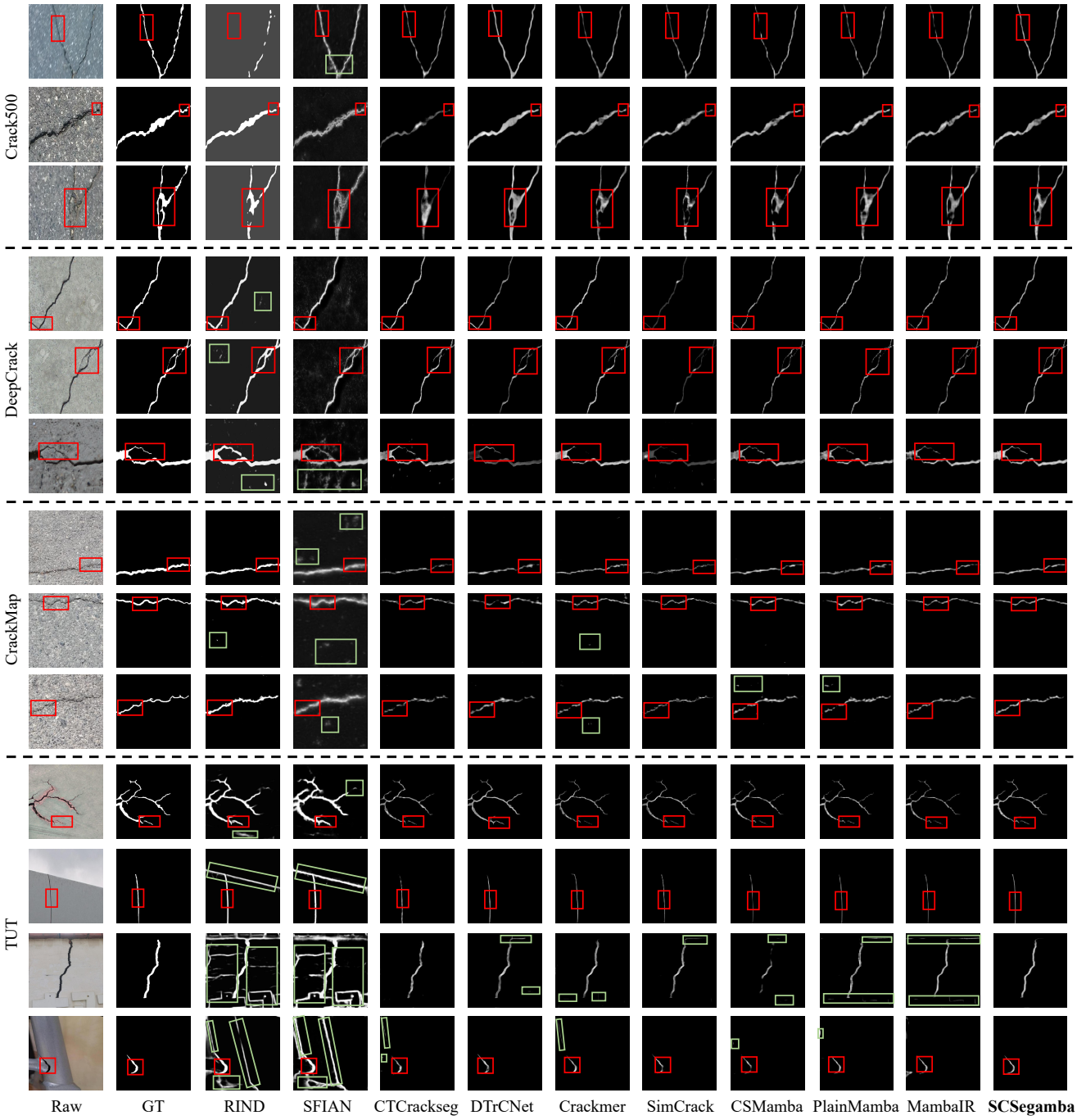

Figure 7. Visual comparison with 9 SOTA methods across four public datasets. Red boxes highlight critical details, and green boxes mark misidentified regions.

primarily include bitumen, concrete, and brick scenarios with minimal background noise and a range of crack thicknesses, our method consistently achieves accurate segmentation, even capturing intricate fine cracks. This is attributed to GBC's strong capability in capturing crack morphology. In contrast, other methods show weaker performance in continuity and fine segmentation, resulting in discontinuities and expanded segmentation areas that do not align with actual crack images.

For the TUT [33] dataset, which includes diverse scenarios and significant background noise, our method excels at suppressing interference. For instance, in images of cracks

Layer Num	ODS	OIS	P	R	F1	mIoU	Params ↓	FLOPs ↓	Model Size ↓
2	0.8102	0.8165	0.8181	0.8420	0.8299	0.8413	**1.56M**	**12.26G**	**20MB**
4	**0.8204**	**0.8255**	**0.8241**	0.8545	**0.8390**	**0.8479**	2.80M	18.16G	37MB
8	0.8174	0.8222	0.8199	**0.8579**	0.8387	0.8461	5.23M	29.27G	68MB
16	0.8126	0.8187	0.8226	0.8475	0.8349	0.8430	10.08M	51.51G	127MB
32	0.5203	0.5365	0.5830	0.5680	0.5754	0.6785	19.79M	95.97G	247MB

Table 8. Experiments with different numbers of SAVSS layers. Best results are in bold, and second-best results are underlined.

Patch Size	ODS	OIS	P	R	F1	mIoU	Params ↓	FLOPs ↓	Model Size ↓
4	0.8053	0.8128	0.8146	0.8443	0.8294	0.8381	**2.61M**	51.81G	**34MB**
8	**0.8204**	**0.8255**	**0.8241**	**0.8545**	**0.8390**	**0.8479**	2.80M	18.16G	37MB
16	0.7910	0.7937	0.8126	0.8141	0.8133	0.8272	3.59M	9.74G	45MB
32	0.7318	0.7364	0.7535	0.7576	0.7555	0.7879	6.74M	**7.64G**	82MB

Table 9. Experiments with different patch sizes. Best results are in bold, and second-best results are underlined.

Methods	ODS	OIS	P	R	F1	mIoU	Params ↓	FLOPs ↓	Model Size ↓
MambaIR [16]	0.7869	0.7956	0.7714	0.8445	0.8071	0.8240	3.57M	19.71G	29MB
CSMamba [37]	0.7140	0.7201	0.6934	0.8171	0.7503	0.7773	12.68M	15.44G	84MB
PlainMamba [55]	0.7787	0.7896	0.7617	0.8531	0.8064	0.8201	**2.20M**	**14.09G**	**18MB**
SCSegamba (**Ours**)	**0.8204**	**0.8255**	**0.8241**	**0.8545**	**0.8390**	**0.8479**	2.80M	18.16G	37MB

Table 10. Comparison experiments of different Mamba-based methods using 4 VSS layers. Best results are in bold, and second-best results are underlined.

on generator blades and steel pipes, it effectively minimizes irrelevant noise and provides precise crack segmentation. This performance is largely attributed to SAVSS's accurate capture of crack topologies. In contrast, CNN-based methods like RIND [38] and SFIAN [5] struggle to distinguish background noise from crack regions, highlighting their limitations in contextual dependency capture. Other Transformer and Mamba-based methods also fall short in segmentation continuity and detail handling compared to our approach.

10. Additional Analysis

To provide a thorough demonstration of the necessity of each component in our proposed SCSegamba, we conducted a more extensive analysis experiment.

Comparison with different numbers of SAVSS layers. In our SCSegamba, we used 4 layers of SAVSS blocks to balance performance and computational requirements. As listed in Table 8, 4 layers achieved optimal results, with F1 and mIoU scores 0.036% and 0.21% higher than with 8 layers, while reducing parameters by 2.43M, computation by 11.11G, and model size by 31MB. Although using only 2 layers minimized resource demands, with 1.56M parameters, performance decreased. Conversely, using 32 layers increased resource use and reduced performance due to redundant features, which impacted generalization. Thus, 4 SAVSS layers strike an effective balance between performance and resource efficiency, making it ideal for practical applications.

Comparison with different Patch Size. In our SAVSS, we set the Patch Size to 8 during Patch Embedding. To verify its effectiveness, we conducted experiments with various Patch Sizes. As listed in Table 9, a Patch Size of 8 yields the best performance, with F1 and mIoU scores 1.16% and 1.17% higher than a Patch Size of 4. Although a smaller Patch Size of 4 reduces parameters and model size, it limits the receptive field and hinders the effective capture of longer textures, impacting segmentation. As shown in Figure 9, as the Patch Size increases, parameter count and model size decrease, but the computational load per patch rises, affecting efficiency. At a Patch Size of 32, performance drops significantly due to reduced fine-grained detail capture and sensitivity to contextual variations. Thus, a Patch Size of 8 balances detail accuracy and generalization while maintaining model efficiency.

Comparison under the same number of VSS layers. In Subsection 4.3, we compare SCSegamba with other SOTA methods, using default VSS layer settings for Mamba-based models like MambaIR [16], CSMamba [37], and PlainMamba [55]. To examine complexity and performance under uniform VSS layer counts, we set all Mamba-based models to 4 VSS layers and conducted comparisons. As listed in Table 2 and 10, although computational requirements for MambaIR, CSMamba, and PlainMamba de-

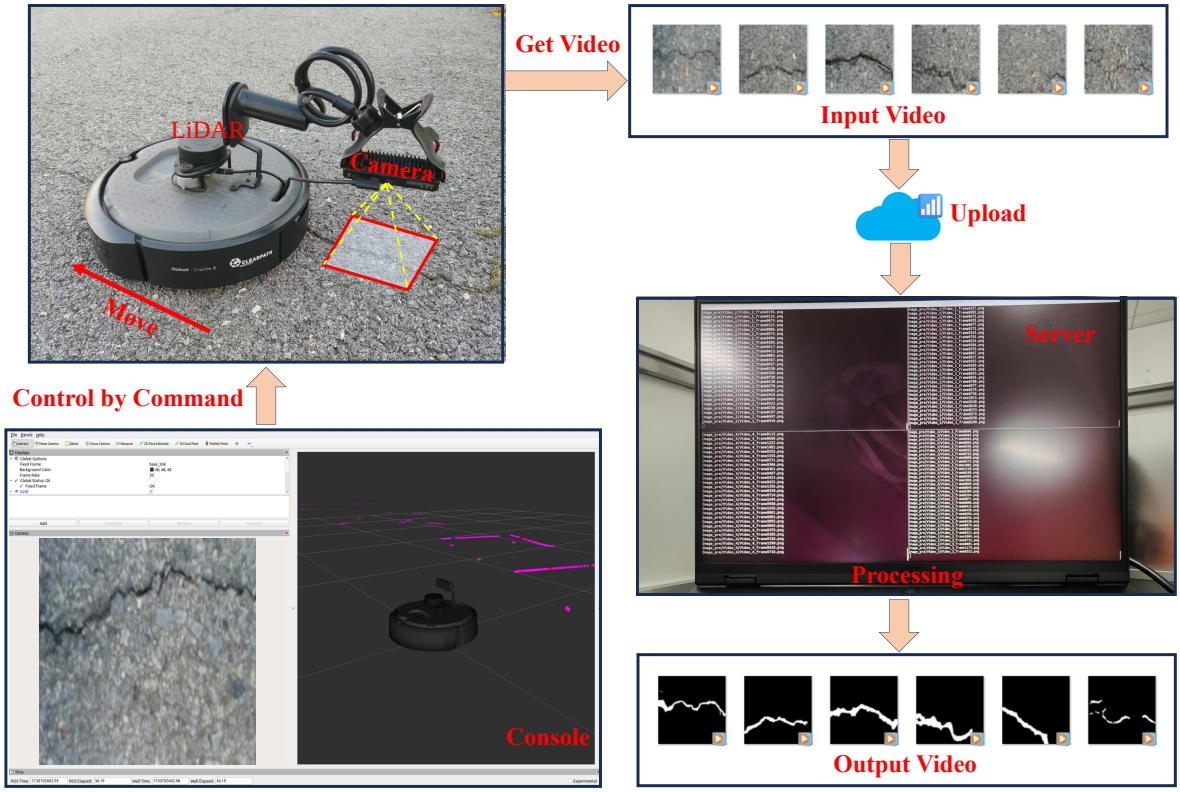

Figure 8. Schematic of real-world deployment. The intelligent vehicle is placed on an outdoor road surface, and we use the server terminal to remotely control it. The vehicle transmits the video data in real-time to the server, where it is processed to obtain the final output.

crease, their performance drops significantly. For example, CSMamba's F1 and mIoU scores drop to 0.7503 and 0.7773. While PlainMamba with 4 layers achieves reductions of 0.60M in parameters, 4.07G in FLOPs, and 19MB in model size, SCSegamba surpasses it by 4.04% in F1 and 3.39% in mIoU. Thus, with 4 SAVSS layers, SCSegamba balances performance and efficiency, capturing crack morphology and texture for high-quality segmentation.

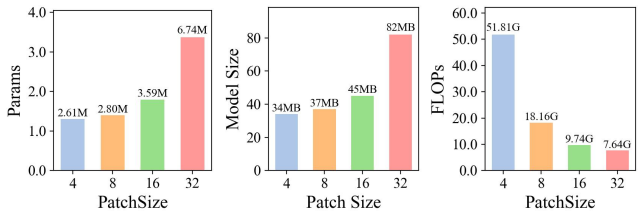

Figure 9. Comparison of computing resources required for different Patch Size

11. Real-world Deployment Applications

To validate the effectiveness of our proposed SCSegamba in real-world applications, we conducted a practical deployment and compared its real-world performance with other SOTA methods. Specifically, our experimental system consists of two main components: the intelligent vehicle and the server. The intelligent vehicle used is a Turtlebot4 Lite driven by a Raspberry Pi 4, equipped with a LiDAR and a camera. The camera model is OAK-D-Pro, fitted with an OV9282 image sensor capable of capturing high-quality crack images. The server is a laptop equipped with a Core i9-13900 CPU running Ubuntu 22.04. The intelligent vehicle and server communicate via the internet. This setup simulates resource-limited equipment to evaluate the performance of our SCSegamba in real-world deployment scenarios.

As shown in Figure 8, in the real-world deployment process, the intelligent vehicle was placed on an outdoor road surface filled with cracks. We remotely controlled the vehicle from the server terminal, directing it to move forward in a straight line at a speed of 0.15 m/s. The camera captured video at a frame rate of 30 frames per second. The vehicle transmitted the recorded video data to the server in real-time via the network. To accelerate data transmission from the vehicle to the server, we set the recording resolution to 512 × 512. Upon receiving the video data, the server first segmented it into frames, then fed each frame into the pre-trained SCSegamba model, which was trained on all datasets, for inference. After segmentation, the server

Methods	Inf Time↓
RIND [38]	0.0909s
SFIAN [5]	0.0286s
CTCrackseg [44]	0.0357s
DTrCNet [48]	0.0213s
Crackmer [46]	0.0323s
SimCrack [20]	0.0345s
CSMamba [37]	0.0625s
PlainMamba [55]	0.1667s
MambaIR [16]	0.0400s
SCSegamba (**Ours**)	0.0313s

Table 11. Comparison of inference time with other SOTA methods on resource-constrained server.

recombined the processed frames into a video, yielding the final output. This setup simulates real-time crack segmentation in an real-world production process.

Additionally, we deployed the weight files of other SOTA methods on the server for comparison. As listed in Table 11, our SCSegamba achieved an inference speed of 0.0313 seconds per frame on the resource-constrained server, outperforming most other methods. This demonstrates that our method has excellent real-time performance, making it suitable for real-time segmentation of cracks in video data.

As shown in Figure 10, compared to other SOTA methods, our SCSegamba better suppresses irrelevant noise in video data and generates continuous crack region segmentation maps. For instance, although SSM-based methods like PlainMamba [55], MambaIR [16], and CSMamba [37] achieve continuous segmentation, they tend to produce false positives in some irrelevant noise spots. Additionally, while CNN and Transformer-based methods achieve high metrics and performance on datasets with faster inference speed, their performance on video data is suboptimal, often showing discontinuous segmentation and poor background suppression. For example, cracks segmented by DTrCNet [48] and CTCrackSeg [44] exhibit significant discontinuities, and Crackmer [46] struggles to distinguish between crack and background regions. Based on the above real-world deployment results, our SCSegamba produces high-quality segmentation results on crack video data with low parameters and computational resources, making it more suitable for deployment on resource-constrained devices and demonstrating its strong performance in practical production scenarios.

Algorithm 1 SASS execution process

Input: Patch matrix dimensions H, W
Output: $O = (o1, o2, o3, o4)$, $O\_inverse = (o1\_inverse, o2\_inverse, o3\_inverse, o4\_inverse)$, $D = (d1, d2, d3, d4)$
Initialize: $L = H \times W$
Initialize $(i, j) \leftarrow (0, 0)$ for $o1$, $(H-1, W-1)$ if H is odd else $(H-1, 0)$ for $o2$
$i_d \leftarrow down$, $j_d \leftarrow left$ if H is odd else $right$
while $j < W$ or $i \geq 0$ **do**
  $idx \leftarrow i \times W + j$, append idx to $o1$, set $o1\_inverse[idx]$
  if $i_d = down$ and $i < H - 1$ **then**
    $i \leftarrow i + 1$, add $down$ to $d1$
  else
    $j \leftarrow j + 1$, $i_d \leftarrow up$ if $i = H - 1$ else $down$, add $right$ to $d1$
  end if
  $idx \leftarrow i \times W + j$, append idx to $o2$, set $o2\_inverse[idx]$
  if $j_d = right$ and $j < W - 1$ **then**
    $j \leftarrow j + 1$, add $right$ to $d2$
  else
    $i \leftarrow i - 1$, $j_d \leftarrow left$ if $j = W - 1$ else $right$, add up to $d2$
  end if
end while
$d1 \leftarrow [d_{start}] + d1[:-1]$, $d2 \leftarrow [d_{start}] + d2[:-1]$
for diag $\leftarrow 0$ **to** $H + W - 2$ **do**
  $direction \leftarrow right$ if diag is even else $down$
  for $k \leftarrow 0$ **to** $\min(\text{diag} + 1, H, W) - 1$ **do**
    $i, j \leftarrow (\text{diag}-k, k)$ if diag is even else $(k, \text{diag}-k)$
    if $j < W$ **then**
      $idx \leftarrow i \times W + j$
      Append idx to $o3$, set $o3\_inverse[idx]$, add $direction$ to $d3$
    end if
    $i, j \leftarrow (\text{diag} - k, W - k - 1)$ if diag is even else $(k, W - \text{diag} + k - 1)$
    if $j < W$ **then**
      $idx \leftarrow i \times W + j$
      Append idx to $o4$, set $o4\_inverse[idx]$, add $direction$ to $d4$
    end if
  end for
end for
$d3 \leftarrow [d_{start}] + d3[:-1]$, $d4 \leftarrow [d_{start}] + d4[:-1]$
Return: $O, O\_inverse, D$

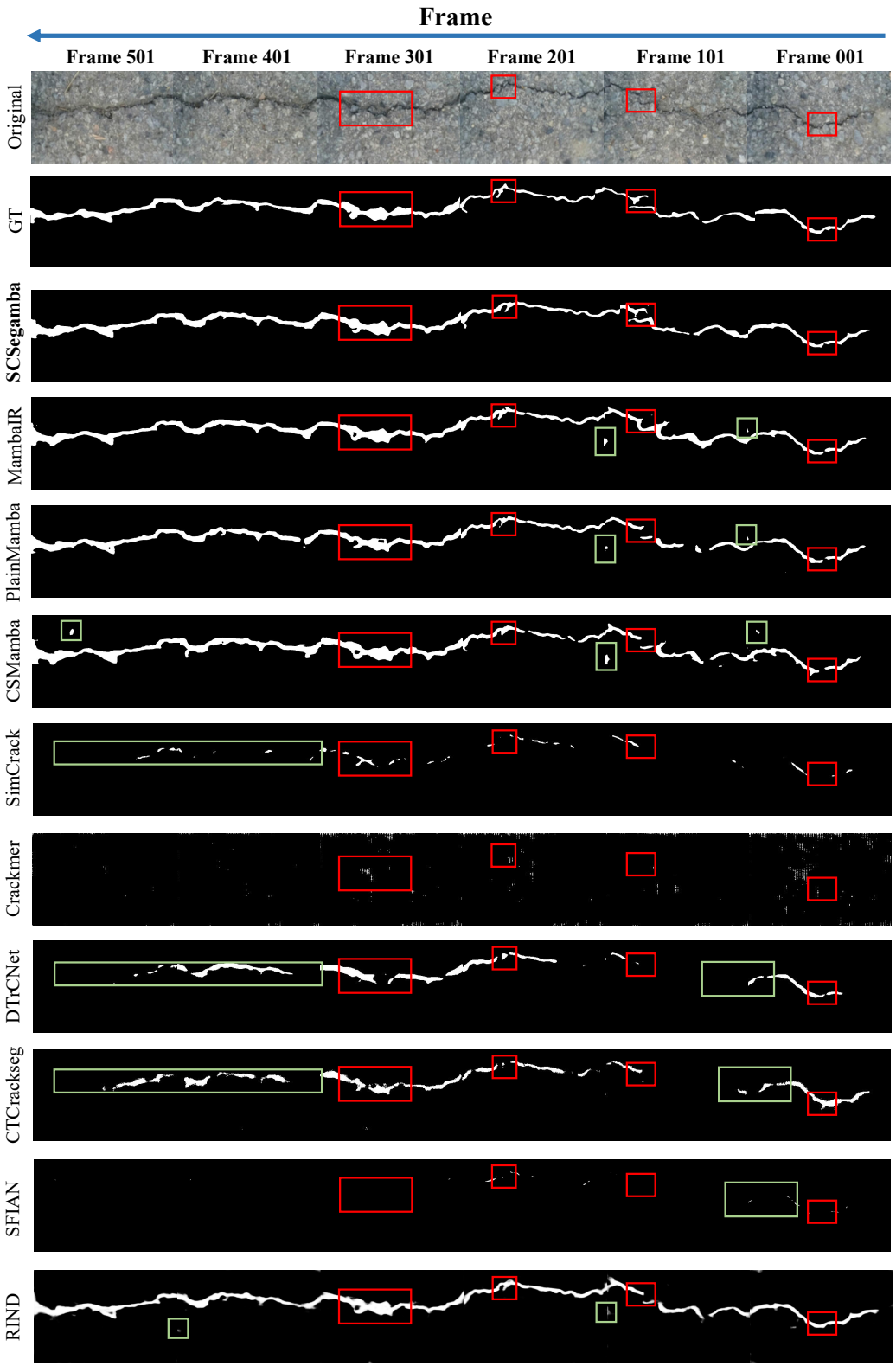

Figure 10. Visualisation comparison on video data keyframes. The interval between keyframes is 100 frames in order to ensure continuity of observation. Red boxes highlight critical details, and green boxes mark misidentified regions.

References

[1] Zaid Al-Huda, Bo Peng, Riyadh Nazar Ali Algburi, Mugahed A Al-antari, AL-Jarazi Rabea, Omar Al-maqtari, and Donghai Zhai. Asymmetric dual-decoder-u-net for pavement crack semantic segmentation. *Automation in Construction*, 156:105138, 2023. 2

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision*, pages 801–818, 2018. 1

[3] Zhuangzhuang Chen, Jin Zhang, Zhuonan Lai, Jie Chen, Zun Liu, and Jianqiang Li. Geometry-aware guided loss for deep crack recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4703–4712, 2022. 1

[4] Zhuangzhuang Chen, Zhuonan Lai, Jie Chen, and Jianqiang Li. Mind marginal non-crack regions: Clustering-inspired representation learning for crack segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12698–12708, 2024. 1

[5] Xu Cheng, Tian He, Fan Shi, Meng Zhao, Xiufeng Liu, and Shengyong Chen. Selective feature fusion and irregular-aware network for pavement crack detection. *IEEE Transactions on Intelligent Transportation Systems*, 2023. 1, 2, 6, 7, 3, 5

[6] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019. 2

[7] Wooram Choi and Young-Jin Cha. Sddnet: Real-time crack segmentation. *IEEE Transactions on Industrial Electronics*, 67(9):8016–8025, 2019. 2

[8] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017. 4

[9] Jiaxiu Dong, Niannian Wang, Hongyuan Fang, Wentong Guo, Bin Li, and Kejie Zhai. Mfafnet: An innovative crack intelligent segmentation method based on multi-layer feature association fusion network. *Advanced Engineering Informatics*, 62:102584, 2024. 1, 2

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *The International Conference on Learning Representations*, 2021. 1, 3

[11] Zhengyang Geng, Meng-Hao Guo, Hongxu Chen, Xia Li, Ke Wei, and Zhouchen Lin. Is attention better than matrix decomposition? In *International Conference on Learning Representations*, 2021. 8

[12] Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models. In *International Conference on Machine Learning*, pages 7616–7633. PMLR, 2022. 3

[13] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in Neural Information Processing Systems*, 35:35971–35983, 2022. 3

[14] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations*, 2022. 2, 3

[15] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The International Conference on Learning Representations*, 2022. 2

[16] Hang Guo, Jinmin Li, Tao Dai, Zhihao Ouyang, Xudong Ren, and Shu-Tao Xia. Mambair: A simple baseline for image restoration with state-space model. In *European Conference on Computer Vision*, pages 222–241. Springer, 2024. 2, 6, 7, 3, 5

[17] Jing-Ming Guo, Herleeyandi Markoni, and Jiann-Der Lee. Barnet: Boundary aware refinement network for crack detection. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):7343–7358, 2021. 2

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[19] Yung-An Hsieh and Yichang James Tsai. Machine learning for crack detection: Review and model performance comparison. *Journal of Computing in Civil Engineering*, 34(5): 04020038, 2020. 1

[20] Achref Jaziri, Martin Mundt, Andres Fernandez, and Visvanathan Ramesh. Designing a hybrid neural system to learn real-world crack segmentation from fractal-based simulation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 8636–8646, 2024. 6, 7, 5

[21] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020. 2

[22] Iason Katsamenis, Eftychios Protopapadakis, Nikolaos Bakalos, Andreas Varvarigos, Anastasios Doulamis, Nikolaos Doulamis, and Athanasios Voulodimos. A few-shot attention recurrent residual u-net for crack segmentation. In *International Symposium on Visual Computing*, pages 199–209. Springer, 2023. 5, 6, 1

[23] Narges Kheradmandi and Vida Mehranfar. A critical review and comparative study on image segmentation-based techniques for pavement crack detection. *Construction and Building Materials*, 321:126162, 2022. 1

[24] Hong Lang, Ye Yuan, Jiang Chen, Shuo Ding, Jian John Lu, and Yong Zhang. Augmented concrete crack segmentation: Learning complete representation to defend background interference in concrete pavements. *IEEE Transactions on Instrumentation and Measurement*, 2024. 1

[25] David Lattanzi and Gregory R Miller. Robust automated concrete damage detection algorithms for field applications.

Journal of Computing in Civil Engineering, 28(2):253–262, 2014. 2

[26] Qin Lei, Jiang Zhong, and Chen Wang. Joint optimization of crack segmentation with an adaptive dynamic threshold module. *IEEE Transactions on Intelligent Transportation Systems*, 2024. 2

[27] Huaiyuan Li, Hui Li, Chuang Li, Baohai Wu, and Jinghuai Gao. Hybrid swin transformer-cnn model for pore-crack structure identification. *IEEE Transactions on Geoscience and Remote Sensing*, 2024. 2

[28] Jialin Li, Qiang Nie, Weifu Fu, Yuhuan Lin, Guangpin Tao, Yong Liu, and Chengjie Wang. Lors: Low-rank residual structure for parameter-efficient network stacking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15866–15876, 2024. 4

[29] Qiufu Li, Xi Jia, Jiancan Zhou, Linlin Shen, and Jinming Duan. Rediscovering bce loss for uniform classification. *arXiv preprint arXiv:2403.07289*, 2024. 5, 1

[30] Jianghai Liao, Yuanhao Yue, Dejin Zhang, Wei Tu, Rui Cao, Qin Zou, and Qingquan Li. Automatic tunnel crack inspection using an efficient mobile imaging module and a lightweight cnn. *IEEE Transactions on Intelligent Transportation Systems*, 23(9):15190–15203, 2022. 1

[31] Huajun Liu, Xiangyu Miao, Christoph Mertz, Chengzhong Xu, and Hui Kong. Crackformer: Transformer network for fine-grained crack detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3783–3792, 2021. 2

[32] Huajun Liu, Jing Yang, Xiangyu Miao, Christoph Mertz, and Hui Kong. Crackformer network for pavement crack segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 24(9):9240–9252, 2023. 2

[33] Hui Liu, Chen Jia, Fan Shi, Xu Cheng, Mianzhao Wang, and Shengyong Chen. Staircase cascaded fusion of lightweight local pattern recognition and long-range dependencies for structural crack segmentation. *arXiv preprint arXiv:2408.12815*, 2024. 1, 5, 7, 2

[34] Wenze Liu, Hao Lu, Hongtao Fu, and Zhiguo Cao. Learning to upsample by learning to sample. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6027–6037, 2023. 5

[35] Yahui Liu, Jian Yao, Xiaohu Lu, Renping Xie, and Li Li. Deepcrack: A deep hierarchical feature learning architecture for crack segmentation. *Neurocomputing*, 338:139–153, 2019. 2, 5, 6, 1

[36] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model. *arXiv preprint arXiv:2401.10166*, 2024. 2, 3, 4

[37] Liu Mushui, Jun Dan, Ziqian Lu, Yunlong Yu, Yingming Li, and Xi Li. Cm-unet: Hybrid cnn-mamba unet for remote sensing image semantic segmentation. *arXiv preprint arXiv:2405.10530*, 2024. 2, 6, 7, 3, 5

[38] Mengyang Pu, Yaping Huang, Qingji Guan, and Haibin Ling. Rindnet: Edge detection for discontinuity in reflectance, illumination, normal and depth. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6879–6888, 2021. 6, 7, 3, 5

[39] Haochen Qi, Xiangwei Kong, Zhibo Jin, Jiqiang Zhang, and Zinan Wang. A vision-transformer-based convex variational network for bridge pavement defect segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 2024. 2

[40] Zhong Qu, Wen Chen, Shi-Yan Wang, Tu-Ming Yi, and Ling Liu. A crack detection algorithm for concrete pavement based on attention mechanism and multi-features fusion. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):11710–11719, 2021. 2

[41] Jianing Quan, Baozhen Ge, and Min Wang. Crackvit: a unified cnn-transformer model for pixel-level crack extraction. *Neural Computing and Applications*, 35(15):10957–10973, 2023. 2

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 8

[43] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: Third International Workshop, DLMIA 2017, and 7th International Workshop, ML-CDS 2017, Held in Conjunction with MICCAI 2017*, pages 240–248. Springer, 2017. 5, 1

[44] Huaqi Tao, Bingxi Liu, Jinqiang Cui, and Hong Zhang. A convolutional-transformer network for crack segmentation with boundary awareness. In *2023 IEEE International Conference on Image Processing*, pages 86–90. IEEE, 2023. 2, 6, 7, 5

[45] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017. 1, 2

[46] Jin Wang, Zhigao Zeng, Pradip Kumar Sharma, Osama Alfarraj, Amr Tolba, Jianming Zhang, and Lei Wang. Dual-path network combining cnn and transformer for pavement crack segmentation. *Automation in Construction*, 158: 105217, 2024. 1, 6, 7, 5

[47] Chunlong Xia, Xinliang Wang, Feng Lv, Xin Hao, and Yifeng Shi. Vit-comer: Vision transformer with convolutional multi-scale feature interaction for dense predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5493–5502, 2024. 1

[48] Chao Xiang, Jingjing Guo, Ran Cao, and Lu Deng. A crack-segmentation algorithm fusing transformers and convolutional neural networks for complex detection scenarios. *Automation in Construction*, 152:104894, 2023. 1, 2, 6, 7, 5

[49] Xiao Xiao, Shen Lian, Zhiming Luo, and Shaozi Li. Weighted res-unet for high-quality retina vessel segmentation. In *2018 9th international conference on information technology in medicine and education*, pages 327–331. IEEE, 2018. 2

[50] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34: 12077–12090, 2021. 8

[51] Xinyu Xie, Yawen Cui, Tao Tan, Xubin Zheng, and Zitong Yu. Fusionmamba: Dynamic feature enhancement for multimodal image fusion with mamba. *Visual Intelligence*, 2(1): 37, 2024. 2

[52] Zhaohu Xing, Tian Ye, Yijun Yang, Guang Liu, and Lei Zhu. Segmamba: Long-range sequential modeling mamba for 3d medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 578–588. Springer, 2024. 2

[53] Kangmin Xu, Liang Liao, Jing Xiao, Chaofeng Chen, Haoning Wu, Qiong Yan, and Weisi Lin. Boosting image quality assessment through efficient transformer adaptation with local feature enhancement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2662–2672, 2024. 2

[54] Tomoyuki Yamaguchi, Shingo Nakamura, Ryo Saegusa, and Shuji Hashimoto. Image-based crack detection for real concrete surfaces. *IEEJ Transactions on Electrical and Electronic Engineering*, 3(1):128–135, 2008. 2

[55] Chenhongyi Yang, Zehui Chen, Miguel Espinosa, Linus Ericsson, Zhenyu Wang, Jiaming Liu, and Elliot J Crowley. Plainmamba: Improving non-hierarchical mamba in visual recognition. *arXiv preprint arXiv:2403.17695*, 2024. 2, 3, 4, 6, 7, 5

[56] Fan Yang, Lei Zhang, Sijia Yu, Danil Prokhorov, Xue Mei, and Haibin Ling. Feature pyramid and hierarchical boosting network for pavement crack detection. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1525–1535, 2019. 2, 5, 6, 1

[57] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4471–4480, 2019. 4

[58] Hang Zhang, Allen A Zhang, Zishuo Dong, Anzheng He, Yang Liu, You Zhan, and Kelvin CP Wang. Robust semantic segmentation for automatic crack detection within pavement images using multi-mixing of global context and local image features. *IEEE Transactions on Intelligent Transportation Systems*, 2024. 1

[59] Tianjie Zhang, Donglei Wang, and Yang Lu. Ecsnet: An accelerated real-time image segmentation cnn architecture for pavement crack detection. *IEEE Transactions on Intelligent Transportation Systems*, 2023. 1

[60] Xiaohu Zhang and Haifeng Huang. Distilling knowledge from a transformer-based crack segmentation model to a light-weighted symmetry model with mixed loss function for portable crack detection equipment. *Symmetry*, 16(5):520, 2024. 3

[61] Jian Zhou, Peisen S Huang, and Fu-Pen Chiang. Wavelet-based pavement distress detection and evaluation. *Optical Engineering*, 45(2):027007–027007, 2006. 2

[62] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. In *International Conference on Machine Learning*, 2024. 2, 3, 4