Temporally and Spatially Taming Vision Mamba for Polyp Video Segmentation

Abstract

Automatic segmentation of polyps from colonoscopy videos is of great clinical significance as it can assist clinicians to make more accurate diagnosis and conduct more precise interventions. Video polyp segmentation (VPS) is a very challenging task owing to (1) the ambiguous boundaries between polyps and their surrounding mucosae and (2) the dramatic variations of a polyp between adjacent frames in terms of scale, contrast, and position. In addition, to fulfill clinical requirements, the inference should be made in a real-time manner. In this work, I propose a novel and efficient segmentation network, which is constructed on the recently popular long-range dependency modeling mechanism, Mamba. Specifically, the method of mamba is spatially and temporally employed in the modules proposed by this work, respectively. Based on the modules with mamba, the segmentation network is able to produce accurate predictions for the colonoscopy videos.

Keywords: Video Polyp Segmentation, Mamba.

1 Introduction

In order to better conduct corse research, I started my work based on the perspective of the task of video polyp segmentation (VPS). First, the boundaries between polyps and their surrounding mucosae are ambiguous, as shown in Figure 1 (a-b). Second, there exists variations of close-in targets in terms of scale, contrast, and position when the colonoscopy camera is moving fast, as shown in Figure 1 (c-e). Third, to fulfill clinical requirements, the inference should be made in a real-time manner. To meet these challenges, a segmentation model should be able to exploit spatial-temporal information across multiple frames for continuously locating of foreground polyps. Besides, dynamic representations learning is essential to perceive the polyp changes between adjacent frames, while distinguishing the polyp edges from complex colon structures.

To tackle the issues mentioned above, I propose a novel and efficient segmentation network, which temporally and spatially employs vision mamba for addressing the challenges of VPS. To model temporal correlations among frames, I employ a mamba block with a bidirectional scanning strategy [22]. Such a bidirectional manner can provide more reliable temporal consistency, and it is able to flexibly deal with video sequences with various lengths.

In specific implement, I design a spatial-aligned temporal dependency propagation (STDP) module, where I employ the bidirectional scanning mamba block to model temporal consistency from horizontally

Figure 1. The challenges of VPS: (a)-(b) blurred boundaries, and (c)-(e) dramatic variations of a polyp between adjacent frames in terms of scale, contrast, and position.

concatenated multi-frame features. Although temporal consistency provides temporal cues for locating same polyp target in the colonoscopy video clip, it fails to learn details from the cases with dramatic variations. In this regard, I propose an utmost dynamic feature extraction (UDFE) module to explore the frame-wise detailed dynamic information. UDFE first exploits the global context modeling capability of mamba to efficiently extract structural feature from multi-frame features. Consequently, more detailed information of each frame are provided for perceiving differences among frames. I conduct extensive experiments on two benchmark video polyp datasets: SUN-SEG [9] and CVC-Clinic DB [1]. Experimental results demonstrate the effectiveness of the proposed method, yielding state-of-the-art segmentation performance with real-time inference.

2 Related works

In recent years, deep learning based approaches [18, 21] have achieved better segmentation performance than traditional machine learning methods [4]. However, most of these methods focus on segmenting polyps in single frames, and thus cannot achieve satisfactory performance in VPS without considering temporal correlations among frames.

2.1 Video Polyp Segmentation

For improving the segmentation accuracy in VPS, PNS+ [9] introduces a progressive residual learning strategy in normalized self-attention blocks [8] to extract spatial-temporally aggregated features. SALI [7] jointly uses short-term alignment and long-term interaction modules to extract spatial-temporal features with a memory bank. But the fixed size and frequent accessing of memory bank limits the inference efficiency and flexibility for dealing with inter-frame dramatic changes.

2.2 Mamba for Vision

To tackle the computational inefficiency of transformers, Mamba [6], constructed on selective structured state space models (SSMs), is developed to accelerate long sequence modeling with efficient hardware-aware designs. Vim [22] proposes bidirectional SSMs to enhance global context modeling in visual understanding. Vmamba [12] introduces a cross-scan module (CSM) to extend the global receptive field for 2D vision data. For video understanding, VideoMamba [10] encodes spatial and temporal embeddings into the bidirectional mamba block following the architecture of ViT [2].

Different to above methods, this work not only utilizes mamba for global context modeling, but also employs the bidirectional SSMs to learn spatial-aligned temporal consistency.

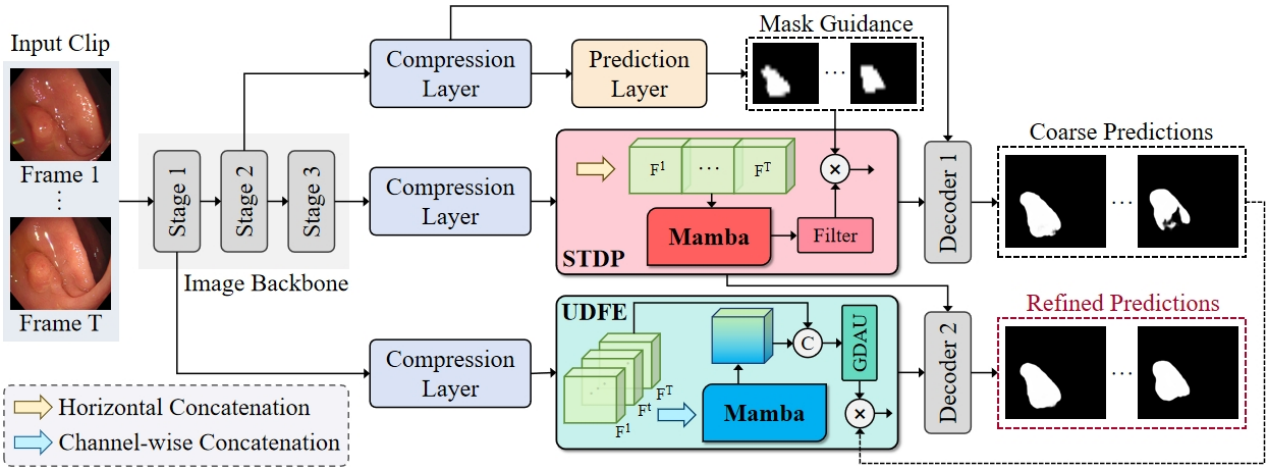

Figure 2. Overview of this research, which consists of a spatial-aligned temporal dependency propagation (STDP) module and an utmost dynamic feature extraction (UDFE) module. Mamba blocks are utilized temporally and spatially in the STDP and the UDFE, respectively.

3 Method

3.1 Overview

The framework of this work is illustrated in Figure 2. Firstly, the compressed high-level features [3,9] are fed into the STDP to model temporal consistency among frames. Secondly, mamba used in the UDFE extracts structural features from the multi-frame features. Then, a gated dynamic activation unit (GDAU) explores utmost dynamic information from the low-level and structural features in a frame-wise manner. Based on the enhanced spatiotemporal features and the utmost dynamic features, the proposed segmentation network can generate refined predictions. In the practical implement, the prediction layer is composed of a 3×3 convolution layer, and the decoders are adopted from [9], where low-level and high-level features are fused with convolution layers.

3.2 Spatial-aligned Temporal Dependency Propagation

Modeling inter-frame temporal consistency is significant to continuously locate polyps from the input colonoscopy frames. Traditional convolution fails to model the inter-frame relationship in time dimension. Besides, successively using cross-attention on contiguous frames may cause enormous computational burden when the clip sequence grows longer. Instead, vision mamba (Vim) [22] sheds light on modeling long-range dependency within an image based on the structured state space model (SSM), avoiding complex attention calculations. Inspired by the bidirectional scanning strategy of Vim, I introduce the mamba block in the STDP to learn spatial-aligned temporal dependency from the high-level semantic features. The detailed framework of STDP is shown in Figure 3 and the schema of bidirectional scanning mamba is shown in Figure 4.

In the mamba block of STDP, input features are divided into patches and flattened into the sequence x_t. Then, x_t is projected to $x'_t = \mathrm{SiLU}(\mathrm{Conv1d}(\mathrm{P}(x_t)))$ and $z_t = \mathrm{SiLU}(\mathrm{P}(x_t))$, where P denotes the linear projection layer and SiLU denotes the SiLU activation, as shown in Figure 4. Under the SSM, system

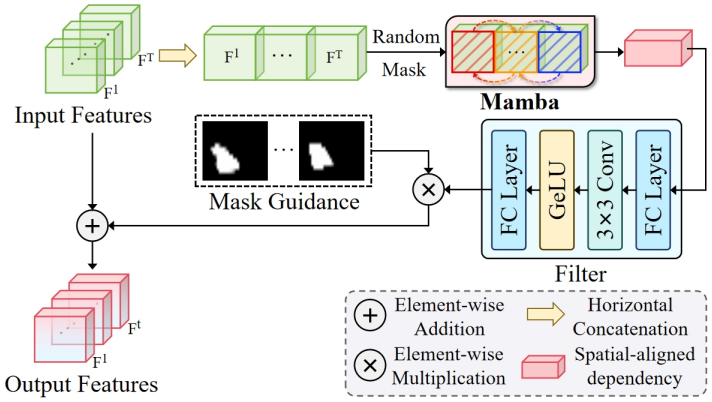

Figure 3. Illustration of STDP. The horizontally concatenated input features are fed into a bidirectional scanning mamba block to model spatiotemporal dependency.

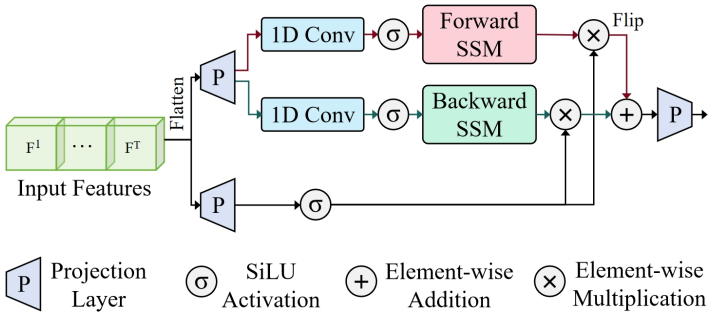

Figure 4. The simplified structure of the bidirectional mamba used in the proposed STDP module.

matrix $\mathbf{A}_t \in \mathbb{R}^{N\times N}$ and projection matrices $\mathbf{B}_t \in \mathbb{R}^{N\times 1}$, $\mathbf{C}_t \in \mathbb{R}^{1\times N}$ are used to model the input sequence. To discretize the continuous system for mamba, a timescale parameter Δ_t is introduced to obtain the discrete counterparts $\bar{\mathbf{A}}_t$, $\bar{\mathbf{B}}_t$ with a zero-order hold (ZOH) method. The formulation on the discretized state variable $h_t \in \mathbb{R}^N$ with N hidden states (see [6, 22] for more details) is generalized as:

$$\bar{\mathbf{A}}_t, \bar{\mathbf{B}}_t = \mathrm{ZOH}(\mathbf{A}, \mathbf{B}_t, \Delta_t) \tag{1}$$

$$h_t = \bar{\mathbf{A}}_t h_{t-1} + \bar{\mathbf{B}}_t x'_t, \quad y_t = \mathbf{C}_t h_t \tag{2}$$

where $\odot$ denotes element-wise multiplication. Thus, the output of the mamba block used in STDP can be defined as:

$$O_{Mamba}(x_t) = \mathrm{P}(y_t \odot z_t + y'_t \odot z_t) \tag{3}$$

where P denotes the linear projection layer. y_t and y'_t are the outputs of forward and backward SSM, respectively. Benefited from efficient hardware perception design of mamba, long-range dependency can be effectively extracted. In addition, as the dependency is computed patch-to-patch horizontally, the obtained inter-frame temporal dependency is of spatial position alignment. In order to propagate the dependency availably related to foreground targets, a filter constructed on multi-layer perceptron (MLP) is introduced under the mask guidance from last stage. Thus, the final output of STDP is given as follows:

$$O_f(F'_3) = \mathrm{MLP}\big(O_{Mamba}(F'_3)\big) \tag{4}$$

$$O_{STDP} = O_f(F'_3) \odot \mathrm{PL}(F_2) + F_3 \tag{5}$$

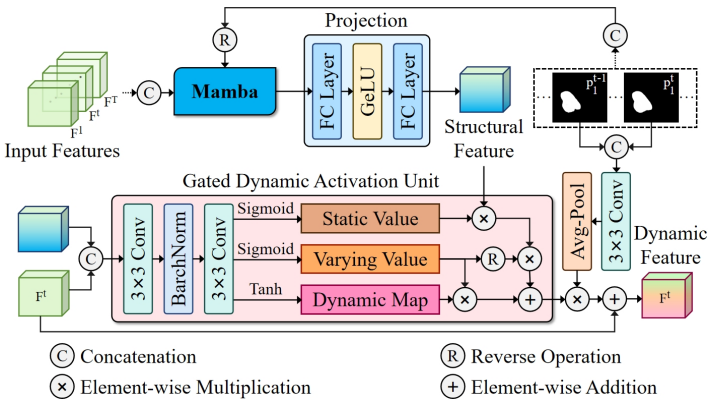

Figure 5. The detailed structure of UDFE. Firstly, structural feature of the input features is extracted by a mamba block. Then, gated dynamic activation unit is employed to explore the dynamic information in a frame-wise manner.

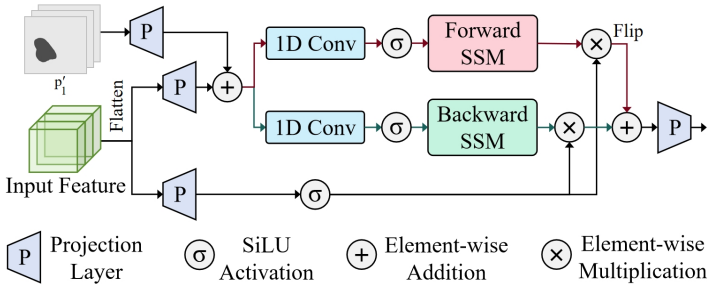

Figure 6. The simplified structure of the bidirectional mamba used in the proposed UDFE.

where $O_f(\cdot)$ stands for the output of the filter. F_i is the compressed features and F_3' is the concatenated features from the third compressed layer. PL denotes the prediction layer with single 3×3 convolution layer. I then use the first decoder d_1 to predict the coarse prediction p_1, which can be defined as follows:

$$p_1 = d_1(O_{STDP}, F_2) \tag{6}$$

3.3 Utmost Dynamic Feature Extraction

Due to the movement of colonoscopy camera, polyps in adjacent frames may have large variations in terms of size, contrast and position. Although spatiotemporal dependency can provide the consistent position information of the same polyp target, it fails to perceive the detailed changes between adjacent frames. In addition, the boundaries between polyps and their surrounding mucosa are usually blurred. Learning frame-wise dynamic features from structural feature is capable to identify detailed differences and distinguish the polyp edge regions from complex colon structures. To this end, I propose a UDFE module to explore dynamic information from low-level features in a frame-wise manner.

The architecture of UDFE shown in Figure 5 has a two-stage procedure. Firstly, the input features are concatenated in the dimension of channels, so that the multiple frame features are regarded as an overall feature. Through modeling global context by a mamba block and a MLP, structural feature is extracted from the concatenated feature. Another procedure is dynamic information excavation with a gated dynamic activation unit (GDAU). In this step, static value, varying value and dynamic map are calculated from the structural feature and each frame feature. Besides, dynamic regions of each frame

are activated to the utmost under the activation ratio calculated from the masks of current and previous frame, which can also alleviate the impact of dramatic scene changes.

As SSM is calculated under a causal mode [15,20], using absolute embeddings may limit the receptive field to some patches, and thus significant textural details contained in long-term patches are ignored. Therefore, I utilize the linear projected features from the coarse prediction masks to complement relative position information, as shown in Figure 6. Notably, in order to strengthen the background position information of colon structures, I apply a reverse operation, $p_1' = 1 - p_1$, on the channel-wise concatenated masks. Thus, the projection of input sequence for x_t' in the mamba block of UDFE is rewritten as $x_t' = \mathrm{SiLU}(\mathrm{Conv1d}(\mathrm{P}(x_t) + \mathrm{P}(p_1')))$. The structural feature extracted by the mamba block can be formulated as:

$$S = \mathrm{MLP}\big(O'_{Mamba}(F_1', p_1')\big) \tag{7}$$

where $F_1' \in \mathbb{R}^{\frac{H}{s_1} \times \frac{W}{s_1} \times T \cdot c}$ represents the channel-wise concatenated features. Subsequently, GDAU is employed to search and explore frame-wise dynamic information. In particular, static value from the similarity and varying value from the discrepancy between each frame feature and the structural feature are computed, respectively. Moreover, a dynamic map is calculated for recording the dynamic regions. According to these values and the dynamic map, the static and dynamic regions can be distinguished effectively. I then activate the corresponding dynamic feature to the utmost under the activation ratio calculated from the masks of current frame and previous frame. Finally, another decoder d_2 is applied to aggregate the enhanced features of STDP and UDFE, obtaining refined segmentation results, which can be defined as:

$$p_2 = d_2(O_{STDP}, O_{UDFE}) \tag{8}$$

3.4 Loss

To supervise the learning of the proposed network, I adopt the Dice loss [14] ($\mathcal{L}_{Dice}$) to facilitate the convergence. Furthermore, given the structural feature learning is an important component for the proposed modules, I introduce a weighted IoU loss ($\mathcal{L}_{WIoU}$) and a weighted BCE loss ($\mathcal{L}_{Wbce}$) [19], attending to optimize the global structure of predictions and pay more attentions to the pixels which are different from surroundings. I employ these three loss functions simultaneously on the outputs of the prediction layer, the first decoder and the second decoder. Thus, I can formulate the total loss function as follows:

$$\mathcal{L}_{seg}(m) = \mathcal{L}_{Dice}(m, y) + \mathcal{L}_{WIoU}(m, y) + \mathcal{L}_{Wbce}(m, y) \tag{9}$$

$$\mathcal{L}_{total} = \mathcal{L}_{seg}\big(\mathrm{PL}(F_2)\big) + \mathcal{L}_{seg}(p_1) + \mathcal{L}_{seg}(p_2) \tag{10}$$

where m represents the prediction masks and y represents the corresponding ground truth pictures.

4 Implementation details

4.1 Comparing with the released source codes

In this work, the proposed modules are constructed on the vision mamba [22] with relative codes shown as Figure 9 (a). Specifically, I remove the original position embedding and class token in the

initialization of vision mamba, because the original vision mamba is proposed for image classification, which is different to the segmentation task of this work. Instead, I add the self designed embedding in its mamba block, as shown in Figure 9 (b) and (c).

4.2 Experimental environment setup

I implement the proposed segmentation network using PyTorch framework on a NVIDIA GeForce RTX 3090 with 24 GB memory. The decoder of STUDNet is adopted from [9], while the pretrained encoder can be adopted from Res2Net-50 [5] and PVTv2-B2 [17]. The model is trained for 30 epochs and employ AdamW as the optimizer with an initial learning rate of $2e-4$. Besides, CosineAnnealingLR [13] is introduced as the scheduler with a maximum of 20 iterations and a minimum learning rate of $5e-5$. The depth of mamba block used in both STDP and UDFE is set to 2. In addition, the partition ratio of the training, validation and test sets is set to $6:2:2$ for CVC-Clinic DB. The training batch size is set to 4. For training on SUN-SEG, each input clip is made up of 6 frames for a better performance-efficiency balance. As the coherence of video frames is weak in CVC-Clinic DB, the input clip for training is composed of 2 frames. All frames are resized into a resolution of 352×352. Following [9], random crop, random horizontal flip and random rotation are applied for data augmentations.

4.3 Main contributions

In this work, I mainly utilize vision mamba in the dimension of time and space, respectively. In the STDP, compared to directly use mamba in a static image, I horizontally concatenate consecutive frame features and thus vision mamba can scan the concatenated feature to model the temporal consistency. To simulate the scenes with dramatic changes, I use random masking to enhance the robustness for the temporal learning. Besides, in order to propagate the extraceted temporal dependency, I use a filter under the mask guidance. More details can be seen in the section 3.2. The codes of STDP's module definition and executive process are as shown in Figure 10.

On the other hand, I design the UDFE module to explore dynamic features of each frame. In this module, the vision mamba is employed spatially to extract structural feature from the consecutive frames. After the structural feature is obtained, I use a gated recurrent unit based on convolutions to compute static value, varying value and dynamic map. These values can be further used to calculate the dynamic regions from the structural feature and frame feature. More details can be seen in the section 3.3. The codes of UDFE's module definition is shown as Figure 11. The relative executive process is shown as Figure 12.

5 Results and analysis

For the section of experiments, the proposed segmentation network is named as STUDNet.

SDPM	UDFE	SUN-SEG-Easy (%)				SUN-SEG-Hard (%)			
		$\mathcal{S}_\alpha$	E_ϕ	F_β^w	Dice	$\mathcal{S}_\alpha$	E_ϕ	F_β^w	Dice
–	–	87.42	91.58	82.04	82.73	86.07	90.86	79.86	80.29
✓	–	88.97	92.38	83.81	84.73	88.21	91.90	82.23	83.48
–	✓	89.03	92.22	83.55	84.54	88.71	92.11	82.92	83.79
✓	✓	89.84	93.43	85.08	85.86	88.75	92.66	83.32	84.59

Table 1. Statistical comparisons of ablation studies on the SUN-SEG test set for proposed components.

5.1 Ablation Study

Table 1 shows the ablation results of the proposed STDP and UDFE. Compared to the baseline, methods equipped with single STDP or single UDFE achieve segmentation improvements in all metrics. The results indicates the effectiveness of learning spatial-aligned temporal consistency for segmenting polyps in colonoscopy videos. In addition, the significant improvement of $\mathcal{S}_\alpha$ and E_ϕ demonstrates that the extracted frame-wise utmost dynamic features provide more detailed information to further improve the segmentation performance.

5.2 Comparisons

Method	Year	Task Type	Backbone	SUN-SEG-Easy (%)				SUN-SEG-Hard (%)				CVC-Clinic DB (%)			
				$\mathcal{S}_\alpha$	E_ϕ	F_β^w	Dice	$\mathcal{S}_\alpha$	E_ϕ	F_β^w	Dice	$\mathcal{S}_\alpha$	E_ϕ	F_β^w	Dice
Vim	2024	NIS	Vim-S	84.33	87.64	78.02	77.55	81.27	85.99	73.06	72.74	93.61	96.43	91.19	91.44
Vmamba	2024	NIS	Vmamba-S	85.86	89.57	78.59	79.37	83.18	88.15	74.68	75.62	93.98	96.84	91.90	91.81
Polyp-SAM	2024	IPS	Sam-ViT-B	85.74	89.21	77.89	78.71	84.42	88.40	75.98	77.16	95.48	97.39	93.49	93.76
G-CASCADE	2024	IPS	PVTv2-B2	87.81	92.28	83.34	83.56	86.32	91.49	80.41	81.12	96.33	98.20	94.85	95.30
VideoMamba	2024	NVS	VideoM-M	88.34	92.15	82.58	83.67	87.10	91.52	80.97	81.22	95.96	97.73	94.72	94.55
PNS+	2022	VPS	Res2Net-50	86.74	87.66	79.17	79.43	86.14	88.16	78.45	78.99	95.52	97.52	94.02	94.17
SALI	2024	VPS	PVTv2-B5	89.12	92.51	83.26	84.36	87.88	92.18	81.28	82.89	94.94	97.91	92.88	93.70
STUDNet	–	VPS	Res2Net-50	89.74	94.04	84.16	85.70	88.49	92.86	81.61	83.45	96.63	98.73	96.01	95.81
STUDNet	–	VPS	PVTv2-B2	89.84	93.43	85.08	85.86	88.75	92.66	83.32	84.59	96.74	98.86	96.39	96.52

Table 2. Statistical comparison with different state-of-the-art methods on the test sets of SUN-SEG and CVC-Clinic DB.

To further demonstrate the advantages of the proposed method, I compare STUDNet with other seven state-of-the-arts methods, which can be classified as natural image segmentation (NIS), image polyp segmentation (IPS), natural video segmentation (NVS) and video polyp segmentation (VPS). The competitors include Vim [22], Vmamba [12], Polyp-SAM [11], G-CASCADE [16], VideoMamba [10], PNS+ [9] and SALI [7]. The statistical comparison results on both SUN-SEG and CVC-ClinicDB are presented in Table 2. Benefited from the spatial-aligned temporal consistency and utmost dynamic information, STUDNet outperforms other state-of-the-art methods in all metrics. While improving the segmentation performance, the proposed method also reaches the demand for real-time inference, as shown in Table 3 and Figure 7.

Method	Backbone	SUN-SEG-Hard		Param. (M)	GFLOPs	FPS
		$\mathcal{S}_\alpha$(%)	Dice(%)			
VideoMamba	VideoM-M	87.10	81.22	75.66	10.90	18
PNS+	Res2Net-50	86.14	78.99	9.80	68.58	64
SALI	PVTv2-B5	87.88	82.89	82.73	86.58	10
STUDNet	Res2Net-50	88.49	83.45	38.16	75.17	45
STUDNet	PVTv2-B2	88.75	84.59	29.67	64.18	36

Table 3. Performance-efficiency comparisons of different methods on SUN-SEG-Hard test set.

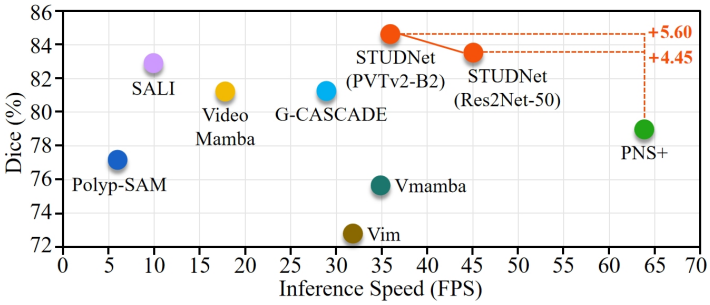

Figure 7. Performance-efficiency comparisons with other state-of-the-art methods on SUN-SEG-Hard test set.

I also perform qualitative visualization comparisons with other state-of-the-art methods. For consecutive changing frames shown in Figure 8, STUDNet provides more stable and refined predictions. It also reflects the effectiveness of the proposed STDP for modeling reliable temporal consistency between adjacent frames and UDFE for exploring more foreground details.

6 Conclusion and future work

In this paper, I propose a novel and efficient segmentation network based on a spatial-aligned temporal modeling strategy and an utmost dynamic representation learning mechanism. Specifically, I propose a STDP module to model reliable temporal consistency from the input video sequence with a bidirectional scanning mamba block. In addition, I propose a UDFE module to excavate frame-wise utmost dynamic information from the structural feature generated by the mamba block. The learned temporal consistency can be used to locate polyps from continuous frames, while the extracted dynamic features can effectively deal with the dramatic changes among frames. These proposed modules make STUDNet adapt to the challenges of VPS availably, achieving state-of-the-art results with a real-time speed on both SUN-SEG dataset and CVC-Clinic DB. Future investigations include assessing the proposed method in more domains and exploring hierarchical recurrent but lightweight spatiotemproal dependency modeling mechanism.

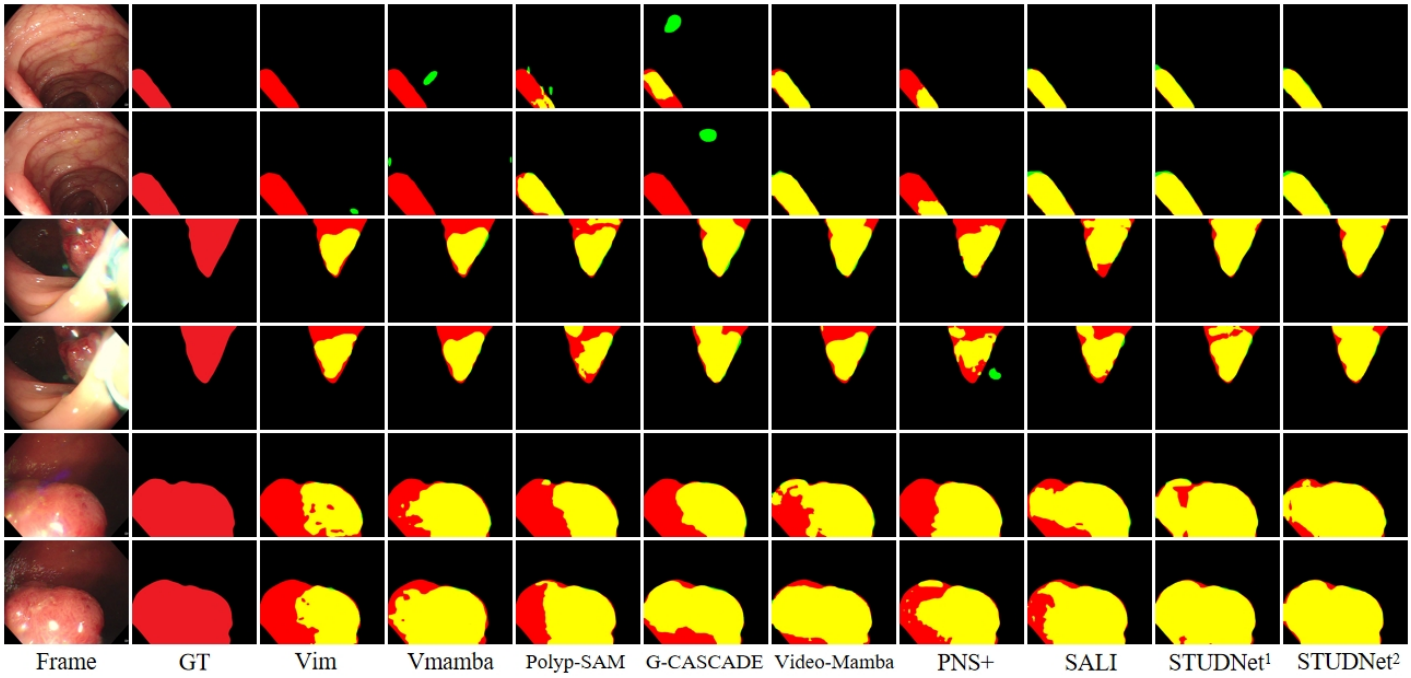

Figure 8. Visual comparisons with different state-of-the-art methods on the SUN-SEG test sets. Red, green and yellow areas represent the ground truth, prediction and their overlapping regions, respectively. Note that STUDNet[1] uses Res2Net-50 as backbone, and STUDNet[2] uses PVTv2-B2 as backbone.

```python
class VisionMamba(nn.Module):
    def __init__(self,
                 img_size=224,
                 patch_size=16,
                 stride=16,
                 depth=24,
                 embed_dim=192,
                 dt_rank="auto",
                 d_state=16,
                 d_conv=4,
                 conv_bias=True,
                 mask_guide=False,
                 f_num=1,
                 channels=3,
                 num_classes=1000,
                 ssm_cfg=None,
                 drop_rate=0.,
                 drop_path_rate=0.1,
                 norm_epsilon: float = 1e-5,
                 rms_norm: bool = False,
                 initializer_cfg=None,
                 fused_add_norm=False,
                 residual_in_fp32=False,
                 device=None,
                 dtype=None,
                 ft_seq_len=None,
                 pt_hw_seq_len=14,
                 if_bidirectional=False,
                 final_pool_type='none',
                 if_abs_pos_embed=False,
                 if_rope=False,
                 if_rope_residual=False,
                 flip_img_sequences_ratio=-1.,
                 if_bimamba=False,
                 bimamba_type="none",
                 if_cls_token=False,
                 if_devide_out=False,
                 init_layer_scale=None,
                 use_double_cls_token=False,
                 use_middle_cls_token=False,
                 **kwargs):
        factory_kwargs = {"device": device, "dtype": dtype}
        # add factory_kwargs into kwargs
        kwargs.update(factory_kwargs)
        super().__init__()
```

(a)

```python
### Mask Embedding with Linear ###
if self.mask_guide:
    self.x_mask_embeds = nn.Linear(f_num, self.d_inner, bias=False, **factory_kwargs)
    self.z_mask_embeds = nn.Linear(f_num, self.d_inner, bias=False, **factory_kwargs)
```

(b)

```python
# We do matmul and transpose BLH -> HBL at the same time
xz = rearrange(
    self.in_proj.weight @ rearrange(hidden_states, "b l d -> d (b l)"),
    "d (b l) -> b d l",
    l=seqlen,
)
if self.in_proj.bias is not None:
    xz = xz + rearrange(self.in_proj.bias.to(dtype=xz.dtype), "d -> d 1")

if self.mask_guide and masks is not None:
    x, z = xz.chunk(2, dim=1)

    x_mask = masks[0].flatten(-2).transpose(1, 2)
    x_m_embeds = self.x_mask_embeds(x_mask).transpose(1, 2)
    # x_mask = masks[0].flatten(-2).expand(-1, x.shape[1], -1)

    # z_mask = masks[1].flatten(-2).transpose(1, 2)
    # z_m_embeds = self.z_mask_embeds(z_mask).transpose(1, 2)
    # z_mask = masks[1].flatten(-2).expand(-1, z.shape[1], -1)

    x = x + x_m_embeds
    # x = x_mask.mul(x)

    # z = z + z_m_embeds
    # z = z_mask.mul(z)

    xz = torch.cat((x, z), dim=1)
```

(c)

Figure 9. Part of codes for the realization of the proposed STDP module.

```python
self.spacetime_extractor = VisionMamba(
    patch_size=self.patch_size, img_size=self.img_size, channels=self.fea_channels, d_state=1, d_conv=3, conv_bias=False,
    stride=self.patch_size, embed_dim=self.embed_dim, depth=2, drop_path_rate=0., rms_norm=False,
    residual_in_fp32=True, fused_add_norm=True, final_pool_type='all',
    if_abs_pos_embed=False, bimamba_type="v2", if_cls_token=False,
    if_devide_out=True, use_middle_cls_token=False, **kwargs
)
self.spacetime_extractor.default_cfg = _cfg()
self.spacetime_outlayer = Mlp(in_features=self.embed_dim, hidden_features=int(self.embed_dim * mlp_ratio), out_features=self.fea_channels)

self.mask_extract = nn.Conv2d(self.fea_channels, 1, kernel_size=3, stride=1, padding=1)
```

```python
B, C, H, W = x2.shape
if sdpm_on:
    x2 = x2.reshape(origin_shape[1], origin_shape[0], C, H, W)
    x2_frames = []

    # Add random mask
    mask_flag = False
    if mask_on:
        if random.random() > 0.5:
            mask_flag = True
            mask_idx = random.randint(1, origin_shape[1]-1)
    for i in range(origin_shape[1]):
        if mask_flag and i == mask_idx:
            mask_tensor = torch.zeros(x2[i].shape, dtype=torch.float32).to(x2[i].device)
            x2_frames.append(mask_tensor)
        else:
            x2_frames.append(x2[i])         # x2[i]: [B, fea_channels, 16, 32]
    x2_cat = torch.cat(x2_frames, dim=3)    #x2_cat: [B, fea_channels, 16, 192]
    x2_cat = self.spacetime_extractor(x2_cat)
    x2_cat = x2_cat.reshape(B, x2_cat.shape[2], self.img_size[0] // self.patch_size, self.img_size[1] // self.patch_size // self.f_num)
    x2_cat = F.interpolate(x2_cat, size=(x2.shape[-2], x2.shape[-1]), mode='nearest')
    x2_cat = self.spacetime_outlayer(x2_cat.reshape(B, H*W, x2_cat.shape[1]), H, W)

    ### Add Mask Guidance ###
    mask_guide = F.interpolate(self.mask_extract(x1), size=(x2.shape[-2], x2.shape[-1]), mode="bilinear", align_corners=False)
    x2_cat = (1+torch.sigmoid(mask_guide)).expand(-1, C, -1, -1).mul(x2_cat.reshape(B, C, H, W)) + x2.reshape(B, C, H, W)
```

Figure 10. Part of codes for the realization of the proposed STDP module.

```
patch_size = 2
self.downsample_ratio_1 = 1 / patch_size
self.patch_size_1 = patch_size
# middle_embeds = embed_dim // 2
self.mamba_1 = VisionMamba(
    patch_size=patch_size, img_size=img_size, channels=in_channels, d_state=1, d_conv=3,
    conv_bias=False, mask_guide=True, f_num=f_num,
    stride=patch_size, embed_dim=in_channels, depth=1, drop_path_rate=0., rms_norm=False,
    residual_in_fp32=True, fused_add_norm=True, final_pool_type='all',
    if_abs_pos_embed=False, bimamba_type="v2", if_cls_token=False,
    if_devide_out=True, use_middle_cls_token=False, **kwargs
)
middle_size = tuple(elem // patch_size for elem in img_size)
self.patch_size_2 = 2
self.downsample_ratio_2 = 1 / self.patch_size_2
self.mamba_2 = VisionMamba(
    patch_size=self.patch_size_2, img_size=middle_size, channels=in_channels, d_state=1, d_conv=3,
    conv_bias=False, mask_guide=True, f_num=f_num,
    stride=self.patch_size_2, embed_dim=embed_dim, depth=1, drop_path_rate=0., rms_norm=False,
    residual_in_fp32=True, fused_add_norm=True, final_pool_type='all',
    if_abs_pos_embed=False, bimamba_type="v2", if_cls_token=False,
    if_devide_out=True, use_middle_cls_token=False, **kwargs
)
self.mamba_1.default_cfg = _cfg()
self.mamba_2.default_cfg = _cfg()

self.filter = Mlp(in_features=embed_dim, hidden_features=channels*2, out_features=channels)
self.channels = channels

self.s_fea_mem = None

### GRU ###
self.gru = nn.Sequential(
    nn.Conv2d(channels*2, channels*3, kernel_size=3, stride=1, padding=1),
    nn.BatchNorm2d(channels*3),
    # nn.PReLU(),
    nn.Conv2d(channels*3, channels*3, kernel_size=3, stride=1, padding=1),
)

self.lf_temp = None
self.temp = nn.Sequential(
    nn.Conv2d(2, 32, kernel_size=3, stride=1, padding=1),
    nn.BatchNorm2d(32),
    nn.Conv2d(32, 1, kernel_size=3, stride=1, padding=1),
    nn.AvgPool2d(1, (44, 44))
)
```

Figure 11. Codes for the module definition of the proposed UDFE module.

```python
def forward(self, x, masks):
    B, T, C, H, W = x.shape
    if self.group:
        f = x[:, 0]
        for ti in range(1, T):
            f = torch.cat([f, x[:, ti]], dim=1)
        m = F.interpolate(1-masks.reshape(B, T, H, W), scale_factor=self.downsample_ratio_1, mode='bilinear', align_corners=False)
        structure_f = self.mamba_1(f, (m, m))
        m1 = F.interpolate(m, scale_factor=self.downsample_ratio_2, mode='bilinear', align_corners=False)
        sf_H = H // self.patch_size_1
        sf_W = W // self.patch_size_1
        structure_f = self.mamba_2(structure_f.reshape(B, structure_f.shape[-1], sf_H, sf_W), (m1, m1))
        sf_H = sf_H // self.patch_size_2
        sf_W = sf_W // self.patch_size_2
        sf_C = structure_f.shape[-1]
        # structure_f = F.interpolate(structure_f.reshape(B, structure_f.shape[-1], sf_H, sf_W), size=(H, W), mode='nearest')
        structure_f = F.interpolate(structure_f.reshape(B, structure_f.shape[-1], sf_H, sf_W), size=(H, W), mode='bilinear', align_corners=False)
        structure_f = self.filter(structure_f.reshape(B, H*W, sf_C), H, W).reshape(B, self.channels, H, W)
        self.s_fea_mem = structure_f
    else:
        f0 = x[:, 0]
        self.s_fea_mem = f0
        for ti in range(1, T):
            f1 = self.s_fea_mem
            f2 = x[:, ti]
            # channel-wise concat
            f = torch.cat([f1, f2], dim=1)
            # mask-embeddings
            m1 = F.interpolate(1-masks[:, ti], scale_factor=self.downsample_ratio_1, mode='bilinear', align_corners=False)
            structure_f = self.mamba_1(f, (m1, m1))
            m1 = F.interpolate(m1, scale_factor=self.downsample_ratio_2, mode='bilinear', align_corners=False)
            sf_H = H // self.patch_size_1
            sf_W = W // self.patch_size_1
            structure_f = self.mamba_2(structure_f.reshape(B, structure_f.shape[-1], sf_H, sf_W), (m1, m1))
            sf_H = sf_H // self.patch_size_2
            sf_W = sf_W // self.patch_size_2
            sf_C = structure_f.shape[-1]
            # structure_f = F.interpolate(structure_f.reshape(B, structure_f.shape[-1], sf_H, sf_W), size=(H, W), mode='nearest')
            structure_f = F.interpolate(structure_f.reshape(B, structure_f.shape[-1], sf_H, sf_W), size=(H, W), mode='bilinear', align_corners=False)
            structure_f = self.filter(structure_f.reshape(B, H*W, sf_C), H, W).reshape(B, self.channels, H, W)
            self.s_fea_mem = structure_f


    self.lf_temp = masks[:, 0]
    for ti in range(0, T):
        d_f = torch.cat([self.s_fea_mem, x[:, ti]], dim=1)
        values = self.gru(d_f)
        static_v = torch.sigmoid(values[:, :self.channels])
        varying_v = torch.sigmoid(values[:, self.channels:self.channels*2])
        dynamic_v = torch.tanh(values[:, self.channels*2:])
        dynamic_f = static_v * self.s_fea_mem * (1-varying_v) + varying_v * dynamic_v

        ratio = self.temp(torch.cat([self.lf_temp, masks[:, ti]], dim=1))
        x[:, ti] = x[:, ti] + dynamic_f * ratio
        self.lf_temp = masks[:, ti]
        # x[:, ti] = (x[:, ti] + dynamic_f) / 2
    return x
```

Figure 12. Codes for the executive process of the proposed UDFE module.

References

[1] Jorge Bernal, F Javier Sánchez, Gloria Fernández-Esparrach, Debora Gil, Cristina Rodríguez, and Fernando Vilariño. Wm-dova maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians. Computerized Medical Imaging and Graphics, 43:99–111, 2015.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[3] Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen, and Ling Shao. Pranet: Parallel reverse attention network for polyp segmentation. In International Conference on Medical Image Computing and Computer-assisted Intervention, pages 263–273. Springer, 2020.

[4] Melanie Ganz, Xiaoyun Yang, and Greg Slabaugh. Automatic segmentation of polyps in colonoscopic narrow-band imaging data. IEEE Transactions on Biomedical Engineering, 59(8):2144–2151, 2012.

[5] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(2):652–662, 2019.

[6] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. arXiv preprint arXiv:2312.00752, 2023.

[7] Qiang Hu, Zhenyu Yi, Ying Zhou, Fang Peng, Mei Liu, Qiang Li, and Zhiwei Wang. Sali: Short-term alignment and long-term interaction network for colonoscopy video polyp segmentation. arXiv preprint arXiv:2406.13532, 2024.

[8] Ge-Peng Ji, Yu-Cheng Chou, Deng-Ping Fan, Geng Chen, Huazhu Fu, Debesh Jha, and Ling Shao. Progressively normalized self-attention network for video polyp segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention, pages 142–152. Springer, 2021.

[9] Ge-Peng Ji, Guobao Xiao, Yu-Cheng Chou, Deng-Ping Fan, Kai Zhao, Geng Chen, and Luc Van Gool. Video polyp segmentation: A deep learning perspective. Machine Intelligence Research, 19(6):531–549, 2022.

[10] Kunchang Li, Xinhao Li, Yi Wang, Yinan He, Yali Wang, Limin Wang, and Yu Qiao. Videomamba: State space model for efficient video understanding. In European Conference on Computer Vision, pages 237–255. Springer, 2025.

[11] Yuheng Li, Mingzhe Hu, and Xiaofeng Yang. Polyp-sam: Transfer sam for polyp segmentation. In Medical Imaging 2024: Computer-Aided Diagnosis, volume 12927, pages 759–765. SPIE, 2024.

[12] Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, and Yunfan Liu. Vmamba: Visual state space model, 2024.

[13] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983, 2016.

[14] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In 2016 Fourth International Conference on 3D vision (3DV), pages 565–571. Ieee, 2016.

[15] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research, 21(140):1–67, 2020.

[16] Md Mostafijur Rahman and Radu Marculescu. G-cascade: Efficient cascaded graph convolutional decoding for 2d medical image segmentation. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 7728–7737, 2024.

[17] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvt v2: Improved baselines with pyramid vision transformer. Computational Visual Media, 8(3):415–424, 2022.

[18] Jun Wei, Yiwen Hu, Ruimao Zhang, Zhen Li, S Kevin Zhou, and Shuguang Cui. Shallow attention network for polyp segmentation. In Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part I 24, pages 699–708. Springer, 2021.

[19] Jun Wei, Shuhui Wang, and Qingming Huang. F^3net: Fusion, feedback and focus for salient object detection. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 12321–12328, 2020.

[20] Weihao Yu and Xinchao Wang. Mambaout: Do we really need mamba for vision? arXiv preprint arXiv:2405.07992, 2024.

[21] Yundong Zhang, Huiye Liu, and Qiang Hu. Transfuse: Fusing transformers and cnns for medical image segmentation. In Medical image computing and computer assisted intervention–MICCAI 2021: 24th international conference, Strasbourg, France, September 27–October 1, 2021, proceedings, Part I 24, pages 14–24. Springer, 2021.

[22] 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 Forty-first International Conference on Machine Learning, 2024.