Interactive Attention (IA) module

The Attention Mechanism is proposed in the interactive attention (IA) module (**Fig.1**), which consists of an adapted version of the U-Net and a crop model. U-Net is a commonly used neural network in the biomedical image segmentation community. The Attention Mechanism refers to the input of the U-Net, which contains current X-Ray fluoroscopic image and previous attention images. Unlike the U-Net proposed in [13], which takes current and previous X-Ray fluoroscopic images as input, our IA module utilizes previous attention images, which hints to the network about the types of visual features of the guidewire in previous images. The attention images guide the U-Net to pay attention to a small area inside the current image and generate a segmented image, which was utilized by the crop model to produce a cropped image and cropped segmentation. Such an idea of using previous attention images for initial segmentation stems from the scenario that interventionalists mostly care about the ending part of the guidewire during the surgery. As shown in **Fig.2**, U-Net proposed in [13] segments the guidewire into multiple parts while our IA module only focuses on the correct area of the global image.

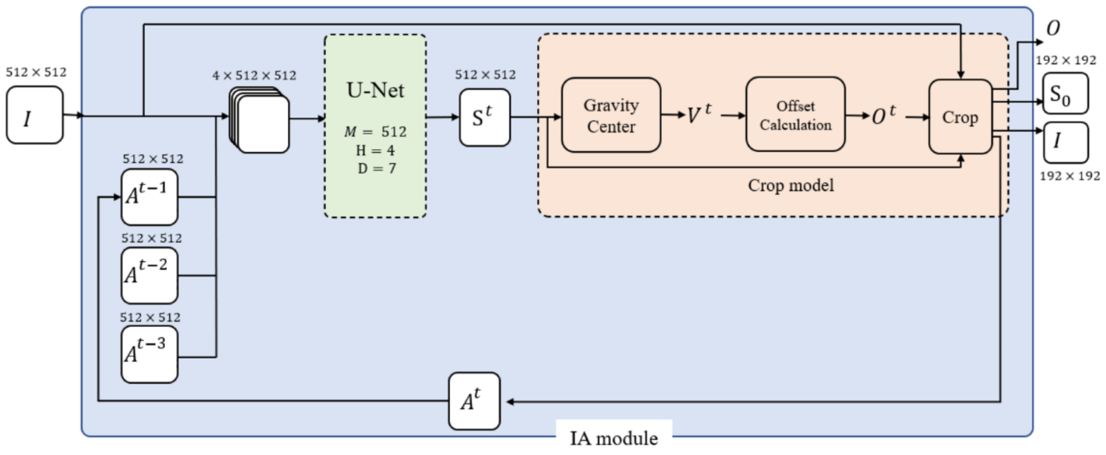

Fig. 1 The composition of Interactive Attention module

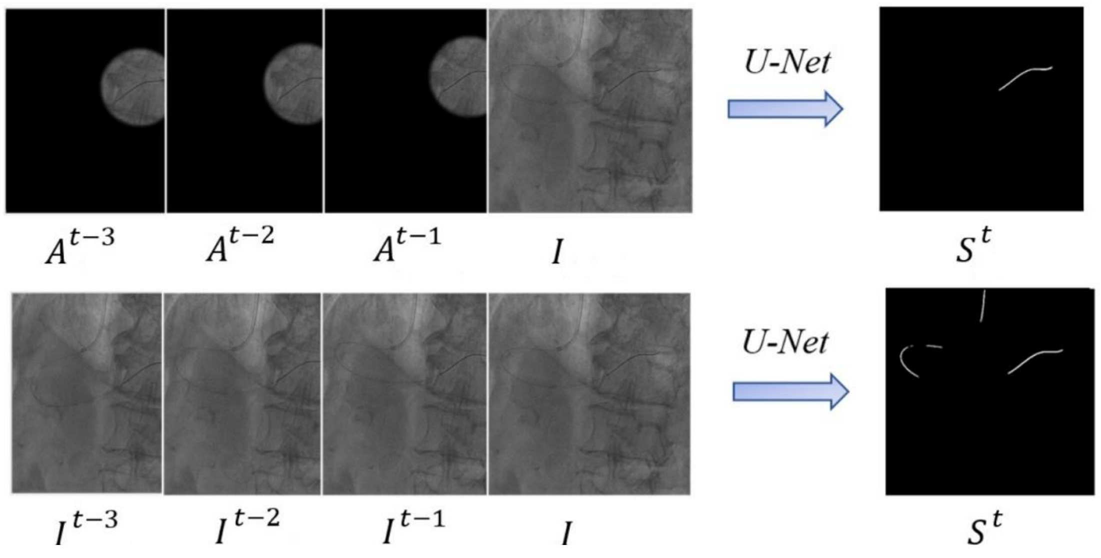

Fig. 2 Comparison between our proposed U-Net in the Interactive Attention module (top) and the U-Net in [13] (bottom)

The IA module is quasi-automatic in clinical usage since it requires a human to manually select attention area of the first H intraoperative images. For latter intraoperative images, the corresponding attention images are generated recursively by the framework automatically without manual selection. This explains why we call the module *'interactive'* since it involves some human effort at the beginning of the surgical procedure.

However, we argue that such an interactive attention module, sacrificing being fully automatic though, substantially increases the segmentation accuracy for subsequent, fine-grained segmentation part. Since angiographic C-arm systems produced by different companies produce X-Ray images that are visually different, it is impossible to collect all of them into one training dataset. The procedure of asking human to manually annotate the interested area, can better guide the network to converge on the guidewire's visual features in that X-Ray image produce by that particular instrument during the intervention surgery. Therefore, it is worthy of being included in our framework as a module, considering the ultra-reliability and accuracy in medical usage.

The Crop Model in IA

The crop model utilizes S^t and I to produce cropped X-Ray image I, initial segmentation S_0, and O - the offset of the cropped area with respect to S^t. Image I and segmentation S_0 are used by subsequent modules performing iterative fine-grained segmentation. Offset O is used to recover final segmentation with cropped size into the segmentation with global size. Note that A^t, the attention image at current time t, which is recursively used by U-Net as input at next time $t+1$, is also generated by the crop model. The algorithm is shown below.

Algorithm 1 Crop Model

The

```
Input The height and width of X-Ray image $M^g, N^g$ $(512 \times 512)$
      The height and width of cropped area $M^c, N^c$ $(192 \times 192)$
      X-Ray image $I$ captured at current time $t$ in sequence $i$
      The corresponding segmented image $S^t$ where guidewire pixels have a value of 1 and the background pixels 0

Output Cropped X-Ray image containing guidewire $I$
       Initial cropped segmentation $S_0$
       Offset of cropped area with respect to $S^t$
       Attention image $A^t$

Step 1 − Calculating Center of Cropped Area $C = (C_x, C_y)$
    $C_x = \frac{\sum_m \sum_n m S_{m,n}}{\sum_m \sum_n S_{m,n}}$    $C_y = \frac{\sum_m \sum_n n S_{m,n}}{\sum_m \sum_n S_{m,n}}$

Step 2 − Calculating Offset of Cropped Area $O = (O_x, O_y)$
    $O_x = int(min\,(max(C_x, M^c/2), M^g - 1 - M^c/2\,) - M^c/2) + 1$
    $O_y = int\big(min\,(max\big(C_y, N^c/2\big), N^g - 1 - N^c/2\,) - N^c/2\big) + 1$

Step 3 − Cropping and Generation
    $I = \{\, I_{m,n} \mid I_{m,n} = I_{m',n'}$ and $0 \le m \le M^c,\ 0 \le n \le N^c \}$
    $S_0 = \{\, S_{m,n} \mid S_{m,n} = S_{m',n'},\ 0 \le m \le M^c,\ 0 \le n \le N^c \}$
            where $m' = m + O_x$ and $n' = n + O_y$
    $A^t = Gaussian\big(A_{m,n}\big)$
            where $A_{m,n} = \begin{cases} I_{m,n} \text{ if } (m - O_x - M^c/2)^2 + \big(n - O_y - N^c/2\big)^2 \le (M^c/2)^2 \\ 0 \text{ otherwise} \end{cases}$

Note $I_{m,n}$ is value of pixel in $m$-th row and $n$-th column of $I$
     $S_{m',n'}$ is value of pixel in $m$-th row and $n$-th column of $S^t$
     $S_{m,n}$ is value of pixel in $m$-th row and $n$-th column of $S_0$
     $int(\cdot), min(\cdot),$ and $max(\cdot)$ are utilized to avoid overflow of the cropped area.
     $Gaussian(\cdot)$ is the low pass 2D filter to smooth the attention image
```

Dilation Model in TE

The dilation model convolutes the output of U-Net in TE, with a kernel ϕ, to compute the maximal pixel value overlapped by ϕ and replace the image pixel in the center of ϕ with that maximal value, thereby causing the expanded area to grow further and finally producing the dilated expansion map.

TE module Training

To train the U-Net in the TE module such that it can learn to expand the endpoints of S_k in a topology-aware way, a new dataset is generated *offline* by modifying the training dataset shared in this paper. For each intraoperative fluoroscopic image I and the corresponding original (radiologist-annotated) guidewire segmentation label L, L is randomly cropped (**Fig.3**). Each type of cropped result, called **Cropped (Type *i*)**, is treated as one training data, which mimics the under-segmentation input for the TE module potentially encountered during the test. Considering multiple possibilities of guidewire morphology during intervention, it is impossible to recover the full structure of the cropped guidewire without knowing the ground truth. Therefore, the corresponding label to train U-Net in the TE module is not the original label L but the images, called **Expanded (Type *i*)** (containing sparse yellow areas only), that are generated by slight expansion at the endpoints of the training data **Cropped (Type *i*)**. The ground truth label L is utilized to ensure that the sparse areas lie exactly inside L (Quantitive description of the crop and slight expansion procedure can be found in Supplementary Material). Such training data and labels correspond to our expectation that the U-Net in the TE module can learn to expand at the endpoints of its input in a topology-aware way and to generate output that theoretically overlaps with the ground truth.

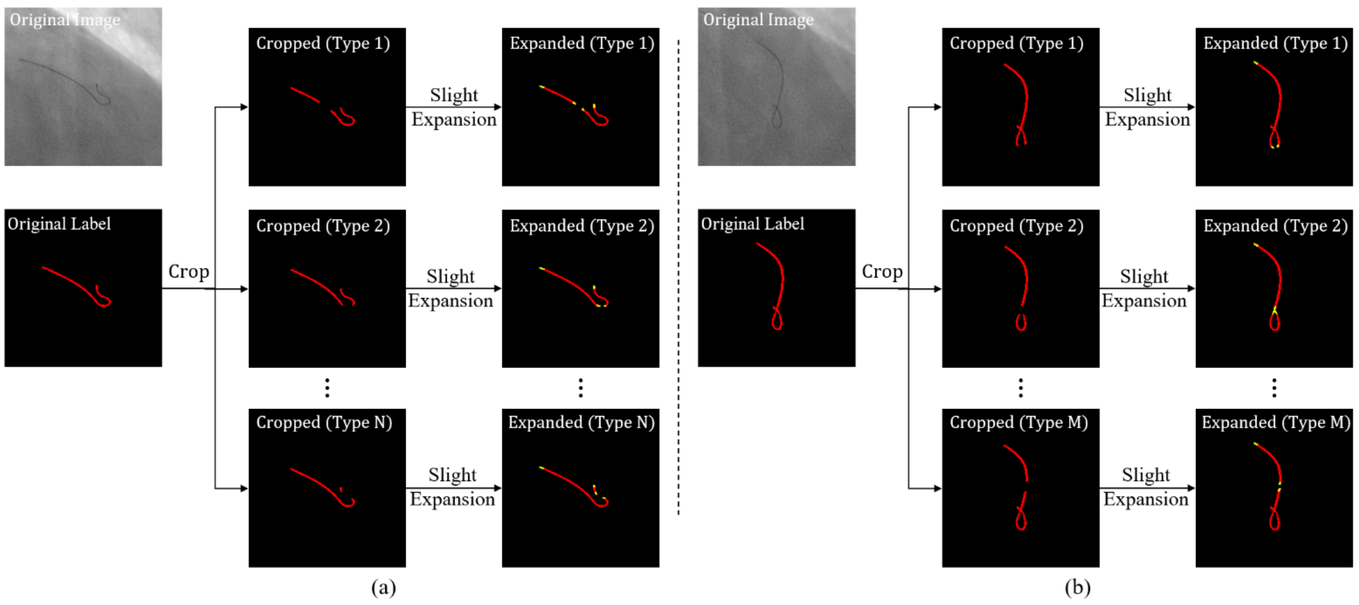

Fig. 3 Exemplary training data (Cropped) and label (Expanded) for TE and FS module. Both data and label are generated *offline* by modifying our shared dataset in this paper. (Note that Expanded contain yellow areas only)

Considering the clinical importance of the guidewire loop structure, the number of cropping choices for fluoroscopic images containing looped guidewires, $M (M = 25)$, is substantially higher than $N (N = 8)$, the number for images containing unlooped guidewires.

The quantitive description of Slight Expansion is that:

Expand at the endpoints of Cropped (Type i) according to the label L to generate slight expansion Expanded (Type i) such that

DiceLoss(Expanded, Label) / DiceLoss(Cropped, Label) = 5 %

Backbone Filtering algorithm

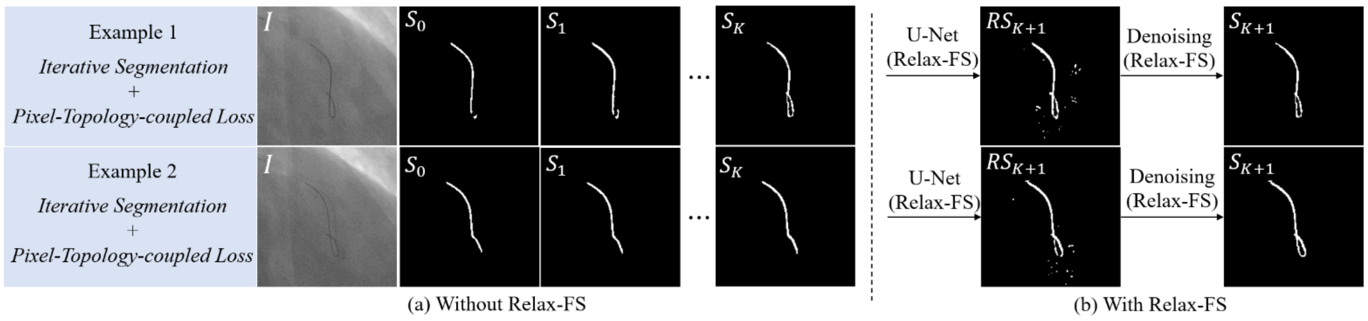

Fig. 4 Successful and failed segmentation cases by our proposed integration of TE and FS with/without Relax-FS

However, for the exceptional cases shown in **Fig. 4(a) Example 2**, our framework does not work because the TE module incrementally expands the segmentation (from last iteration) toward the correct topology only if small fragments exist in the segmentation. When FS cannot produce any small segmented connectivity, TE cannot work regardless of how many times the framework iterates. To solve this challenge, we retrain U-Net of FS at different iterations to share the same network architecture but different parameters[1]. For FS in the previous K iterations, the corresponding U-Net is well trained such that the network can suitably recognize the guidewire from the background. For U-Net in FS^{K+1}, the FS module in the last iteration, the number of training epochs is substantially decreased to terminate the training procedure early such that U-Net in FS^{K}, renamed as *Relax-FS*, cannot receive adequate training and is unable to suitably distinguish between the guidewire and background. In this way, U-Net of *Relax-FS* evaluates any area as the guidewire as long as the area is visually similar to the guidewire, thereby taking S_K as input and producing (relaxed) segmentation result RS_{K+1} containing substantial noise that is irrelevant to the guidewire. The RS_{K+1} may sometimes be beneficial in producing more fragments of the guidewire (**Fig. 4(b) Example 2**) than S_K but sometimes even worse than S_K by adding purely noise (**Fig. 4(b) Example 1**). Since the ground truth is always unknown during the testing stage, one cannot evaluate whether RS_{K+1} is worse than S_K, and thus such a relaxation step is always performed for S_K. The segmentation result S_K generated *before* U-Net in Relax-FS is appropriately segmented (or under-segmented), whereas the (relaxed) segmentation result RS_{K+1} acquired *after* U-Net in Relax-FS is over-segmented and contains irrelevant noise. To minimize the potential negative impact brought by such a relaxation step, an additional heuristic-based denoising step, which utilizes S_K as the backbone image, eliminates noise in RS_{K+1} by iteratively merging that image with the small fragments inside RS_{K+1} that have similar smoothness or direction to the backbone. The denoised output S_{K+1} is the final output of the proposed algorithm.

[1] This explains why the full title of the FS module is not FS but FS^{k}.

Algorithm 2 Backbone Filtering algorithm

```
Input: The under-segmented result $S_K$ generated via $TE - FS$ reciprocal procedure at iteration $K$
            (before the Relaxation-FS module)
       The over-segmented result $RS_{K+1}$ generated via $TE - FS$ reciprocal procedure at final iteration $K+1$
            (after the Relaxation-FS module)

Output: Final segmentation result $S_{K+1}$(with cropped size)
Step 0 – Filter small connectivity (size <= $\epsilon$) in $S_K$
Step 1 – Skeletonization $S_K = Ske(S_K)$ and $RS_{K+1} = Ske(RS_{K+1})$
Step 2 – Get the image $\Theta = RS_{K+1} - S_K$ containing the extra part
Step 3 – Get the largest connectivity $\Omega$ of backbone image $S_K$ as backbone connectivity
            $\Omega = FindAllConnectivity(S_K)$
Step 4 – Get curved sequence $C$ and loop $L$ (if exists) of the backbone connectivity
            $C, L = FindSeq(S_K)$
Step 5 – For ending part $v$ in $\{\tilde{C}_{\varepsilon:1}, \tilde{C}_{-\varepsilon:-1}\}$ (start and end) do
Step 6 –     Initialize the empty compatibility score set $\psi = \{\}$
Step 7 –     For each connectivity $\mu$ in $FindAllConnectivity(\Theta)$ do
Step 8 –         Initialize the compatibility score $y = 0$
Step 9 –         Get the curved sequence $\tau = FindSeq(\mu)[0]$
Step 10 –        If $Distance(\mu_{-1}, \tau_1) > D_{max}$ do
Step 11 –            Continue
Step 12 –        Possible connection sequence $\xi = Interp(v, \tau) + \tau$
Step 13 –        Add smoothness score $y += smoothness(\xi)$
Step 14 –        Direction $dir_\xi = direction(\xi)$
Step 15 –        If $dir_\xi == direction(v)$ do
Step 16 –            $y += INF$
Step 17 –        Append the tuple $(y, \xi)$ consists of score and the connection sequence into $\psi$
Step 18 –        Find the connection sequence $\xi_{max}$ with maximum score inside $\psi$
Step 19 –        If $\psi$ is empty do
Step 20 –            Break
Step 21 –        Else do
Step 22 –        Incrementally expand backbone sequence $C = \begin{cases} \xi + C \text{ when } (v = \tilde{C}_{\varepsilon:1}\tilde{C}_{\varepsilon:1}) \\ C + \xi \text{ when } v = \tilde{C}_{-\varepsilon:-1} \end{cases}$
Step 23 –        Create a new backbone image $RS_{K+1}$ via $C$: $RS_{K+1} = SeqToImg(\{C, L\})$
Step 24 – If $\psi$ is empty for both start and end do
Step 25 –        return $RS_{K+1}$ as final segmentation $S_{K+1}$(with cropped size)
Step 26 – Go back to step 2

Note $Ske(\cdot)$ skeletonize the input segmentation image into image with one pixel-width without changing the topology
     $FindAllConnectivity(\cdot)$ returns a list containing all connectivities of the input segmentation image
     $FindSeq(\cdot)$ return the curved sequence and the loop sequence of the input connectivity
```

Essential parameter configurations in Our Framework

Functionality Modules	Parameters	Description
IA module	$H = 3$	number of previous attention images used in IA module
	$(2,2)$	Convolution Kernel Per Layer
	7	Convolution Layers
	6	Deconvolution Layers
TE-FS iterative segmentation	$K = 4$	number of iterations for the fine-grained segmentation
	$(2,2)$	Convolution Kernel Per Layer
	6	Convolution Layers
	5	Deconvolution Layers
	3×3	Dilation Kernel ϕ
	$1000\ (k = 1)$ $300\ (k = 2)$ $100\ (k = 3)$ $30\ (k = 4)$	Number of data need to select at $k - th$ iteration for re-training FS module
	$\alpha = 1000$	Score to evaluate loop in segmentation result in $Algo$ 2
	$\beta = 600$	Score to evaluate broken topology in segmentation result in $Algo$ 2
	$200\ (k = 1,2,3)$ $30\ (k = 2)$	Number of epochs to train FS^k at $k - th$ iteration
BF module	$D_{max} = 12$	Maximum tolerable distance between the backbone and connectivity to be added
	10	Minimum size of the connectivity to be added into backbone
	$\varepsilon = 3$	Size of the subsequence used for calculating smoothness score

The hyper-parameters used in our framework, shown in Table 1, are set after empirical fine-tuning on our established validation dataset $\mathcal{T}$. All U-Net in IA, TE and FS module are trained with same parameters: The loss function of the model is optimized using stochastic gradient descent with a learning rate of 0.01, a decay of 0.0015 and a momentum of 0.99. Data augmentation is also utilize: every image in training dataset is augmented with a 50% probability via a horizontal and vertical flip, a random rotation (around the image center) ranging from -15 to $+15$ degrees, a horizontal and vertical translation in a range of $\pm18\%$ of the image size, a random intensity shift ranging from -0.09 to $+0.09$ and a Gaussian noise with variance 0.02.

During generation of the dataset to train U-Net in TE module, we randomly cut $C_{i,t}^{c}$ and $L_{i,t}^{c}$ into multiple curves. The smallest size of each curve is at least 15% of the original sequence ($C_{i,t}^{c}$ or $L_{i,t}^{c}$). The size of the part that was cut off ranges from 10% to 30% of the size of original sequence. The procedure is repeated for $Z_c = 8$ times for X-Ray image containing curve structure only and $z_L = 24$ times for guidewire with loop structure.

Iterative Segmentation Termination Condition

The TE-FS cycle would terminate until both pixel-level and topology-level difference between S_k (segmentation of iteration k) and S_{k+1} (segmentation of iteration $k+1$) are sufficiently smaller (lower than a threshold) or the maximum number of iterations is reached. Otherwise the TE-FS cycle would keep iterating. The thresholding mechanism is defined as

$$\text{DiceLoss}(S_{k+1} - S_k, S_{k+1}) \leq 0.08 * \text{DiceLoss}(S_k, S_{k+1})$$

AND

$$\text{TopologicalIntegrity}(S_k, S_{k+1}) = 1$$

$\mathrm{TopologyIntegrity}(\mathrm{A}, \mathrm{B})$ function consists of a skeletonization step and depth-first search (DFS). The skeletonization step, a morphological operation commonly used for image processing, skeletonizes the input (predicted or ground truth) guidewire segmentation result into one-pixel-wide segmentation. DFS calculates the number of connectivities in this segmentation. When only one connection exists, DFS is also utilized to evaluate whether a loop exists in the segmentation.

Specifically, when guidewire loop is found in B, $\mathrm{TopologyIntegrity}\ (\mathrm{A}, \mathrm{B})$ is equal to 1 if and only if the guidewire segmentation result B features integrated topology (only one connectivity exists) and contains a loop structure. When no loop structure of the guidewire is found in A, $\mathrm{TopologyIntegrity}\ (\mathrm{A}, \mathrm{B})$ is equal to 1 as long as the predicted result features integrated topology. Otherwise $\mathrm{TopologyIntegrity}\ (\mathrm{A}, \mathrm{B})$ would return 0.

Execute time of our framework (millisecond)

Ours	Single Step	IA module	TS-FS	BF	BF (Parallel)
		50	90	40 ~ 180	25 ~ 52
	Integration	Single WorkStation		Multi-Workstation	
		180 ~ 320		165 ~ 192	

Table above shows execute time of our framework (and each module) on Nvidia Tesla K80. The time of BF module varies greatly since it depends on the number of noises in $\tilde{S}_{i,t}^{c,K}$. The longest time of our method to segment one X-Ray image (on single workstation) is 320 ms. However, BF module can be expedited via running on multiple workstations (ALIENWARE i7) simultaneously. After parallel processing on three workstations, it costs at most 195 ms, which makes our method real time.

Our algorithm is speed up by multiple workstations mainly on noise-filtering (BF) module. The noise image would firstly be copied by FPGA and sent into multiple workstations in a parallel way. Since different fragments can be simultaneously judged by multiple workstations, in a parallel way, as potential candidates for merging with Backbone image or not, this speeds up the algorithm.