Supplementary

In the supplementary, we additionally provide implementation details, training scheme descriptions, complete visualization comparisons, additional ablation studies, and further discussion regarding the limitations and future work.

1. MLLM-based Part Count Estimation

In most surgical video datasets, instrument annotations are provided in a part-wise manner, i.e., each part of the instrument is labeled as a different category. Consequently, the number of parts for each instrument class can be directly obtained as a prior (all four datasets used in our paper follow this convention). However, in real-world deployments, part-wise annotations may be unavailable, making it unclear how many parts an instrument consists of. To address this, we design a strategy that leverages an MLLM to automatically infer the number of parts when such annotations are missing (see Fig. 3). Concretely, we rely only on class-wise labels. Taking the Vessel Sealer as an example, we first perform a cutout based on the class label to isolate the instrument from all frames containing it, and feed these cropped instrument images into the MLLM with the prompt: "How many parts does this instrument consist of?" We then collect the answers derived by MLLM across samples. Finally, we apply majority voting to the responses and use the most frequent value as the estimated number of parts for each instrument. We validated the proposed strategy on both the EndoVis2017 [1] and EndoVis2018 [2] datasets. As shown in Table 1 and Table 2, the results demonstrate that our method successfully assigns the correct part count to all instruments across both datasets.

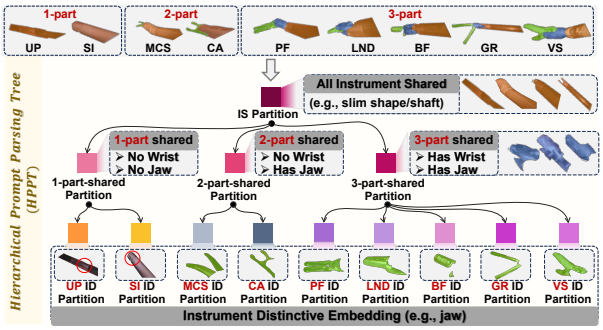
2. Implementation details

We follow the preprocessing procedure described in [6] to preprocess the dataset. Based on the supplementary materials, along with the provided code and the open-source implementations of [6] and [28], our project can be fully reproduced. When using SAM, we adopt the SAM ViT-H [15] version as our encoder θ_{Enc}. In our practical implementation, for the prompt partition $\mathrm{P} \in \mathbb{R}^{u \times b}$, the number of tokens u is treated as a tunable hyperparameter. Based on our grid search results, the best performance is achieved when u is set to 12. We employ the Adam optimizer with a learning rate of $1e^{-4}$ for the optimization. We use the PyTorch framework, and the model is trained with an NVIDIA A6000 GPU.

3. Metrics for Class-Incremental Learning

To provide a deeper understanding of model performance in the class-incremental learning (CIL) setting, we adopt two widely-used evaluation metrics: Backward Transfer (BWT)

Figure 1. Illustration of HPPT construction.



and Forward Transfer (FWT) [3, 8, 21]. These metrics quantify how knowledge is transferred across tasks during the incremental learning process. Due to the fundamental differences between class-incremental learning and other continual learning scenarios, we adopt specific BWT and FWT metrics to align with this setting.

Backward Transfer (BWT) BWT evaluates the influence of learning new data on the performance of previously learned (old and regular) classes. Specifically, for training episode t, we define BWT as the degradation (or improvement) in performance on old and regular classes compared to the previous episode $(t-1)$:

$$\mathrm{BWT} = \frac{1}{|\mathcal{C}_t^{\text{old}} \cup \mathcal{C}_t^{\text{reg}}|} \sum_{c \in \mathcal{C}_t^{\text{old}} \cup \mathcal{C}_t^{\text{reg}}} \left(\mathrm{IoU}_{t(c)} - \mathrm{IoU}_{t-1(c)}\right), \tag{9}$$

where $\mathrm{IoU}_{t(c)}$ denotes the intersection over union (IoU) score for class c after training at episode t, and $\mathrm{IoU}_{t-1(c)}$ is the corresponding IoU score from the previous episode.

Interpretation:

- BWT < 0: indicates *forgetting*, meaning that the performance on previously learned (old and regular) classes has decreased after learning the current episode.
- BWT > 0: indicates *positive backward transfer*, suggesting that learning new classes leads to improved performance on past classes. This may happen when the new data provides beneficial features that reinforce earlier knowledge.
- BWT $= 0$: indicates no change in past performance.

Forward Transfer (FWT) FWT measures how prior learning benefits the model in learning new classes. For episode t, we define FWT as the difference between the IoU score of the model on newly introduced classes and the average IoU score obtained when training those classes in isolation (i.e., individual training):

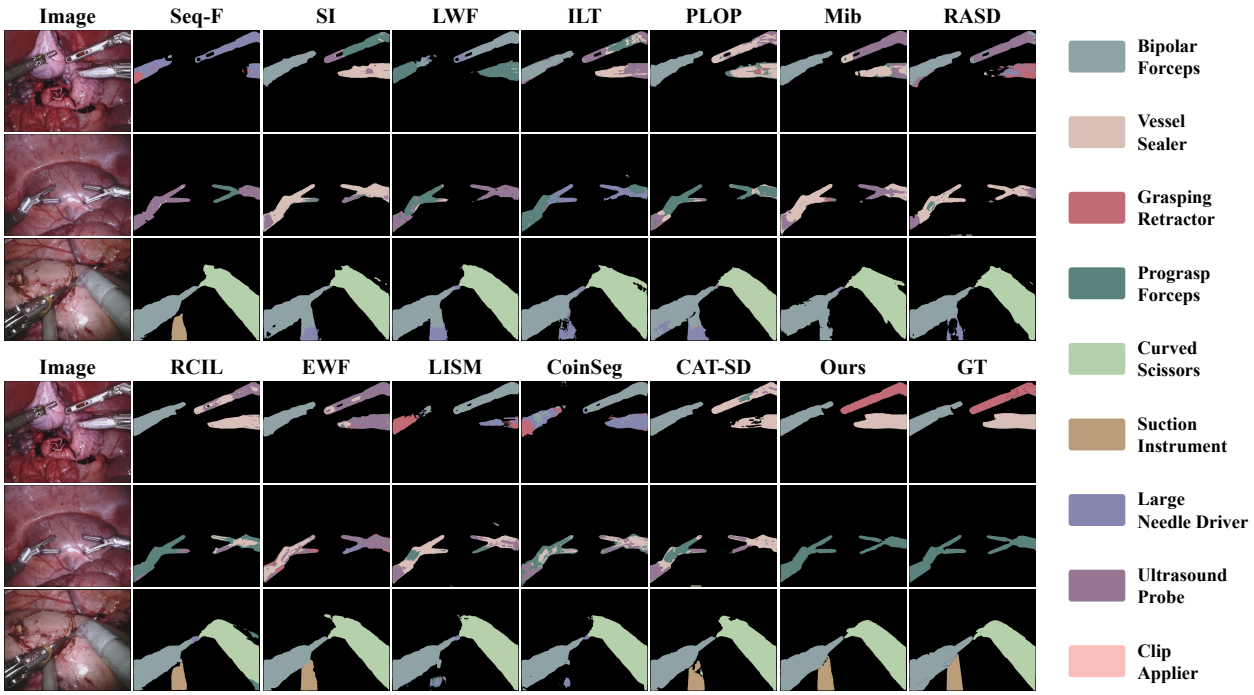


Figure 2. Visualization comparison of segmentation results across all methods. *Rows 1–2, 4-5*: previous datasets. *Row 3, 6*: current dataset.

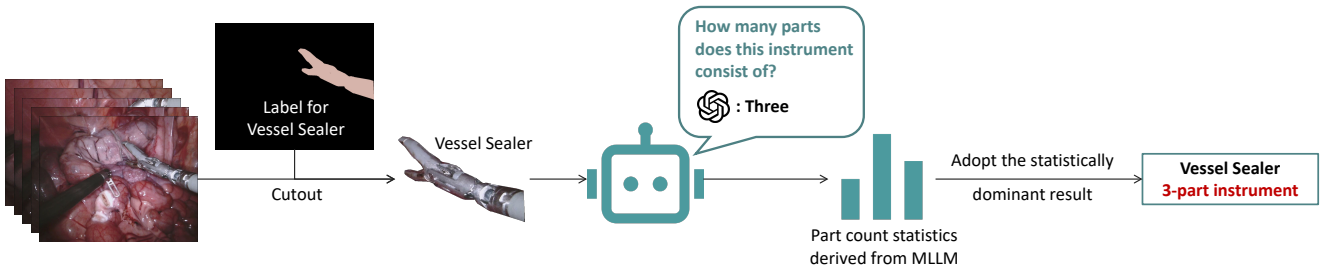


Figure 3. Automatic estimation of instrument part count.

$$\text{FWT} = \frac{1}{|\mathcal{C}_t^{\text{new}}|} \sum_{c \in \mathcal{C}_t^{\text{new}}} \left(\text{IoU}_{t(c)} - \text{IndIoU}_{(c)} \right) \tag{10}$$

where $\text{IoU}_t(c)$ denotes the intersection over union score (IoU) for class c after training at episode t, and $\text{IndIoU}(c)$ refers to the IoU score obtained when class c is trained by individual training.

Interpretation:

- FWT < 0: indicates *negative forward transfer*, suggesting that prior knowledge interferes with the learning of new classes.
- FWT > 0: indicates *positive forward transfer*, meaning that knowledge from previous episodes facilitates learning of new classes.
- FWT $= 0$: indicates no forward transfer.

4. Training and Inference Scheme

In the first episode, we freeze the encoder of SAM and jointly train the decoder, segmentation head, adapter, and the prompt parsing tree. Since each image in the dataset typically contains more than one class of surgical instrument, we activate all instrument-aware prompts during the forward propagation, allowing each pixel to receive prediction scores for all current classes. As a result, the model generates a set of binary segmentation masks, one for each class. To fuse these predictions $\{\hat{y}_c\}_{c \in \mathcal{C}}$ into a final segmentation result $\hat{y} \in \mathcal{C}^{H \times W}$, we apply an argmax-based fusion. For each pixel location (i, j), the final predicted class is determined as:

$$\hat{y}(i,j) = \begin{cases} \arg\max\limits_{c \in \mathbf{C}} \hat{y}_c(i,j), & \text{if } \max\limits_{c \in \mathbf{C}} \hat{y}_c(i,j) > \tau \\ \text{background}, & \text{otherwise}, \end{cases} \tag{11}$$

Table 1. Part Count Inference Results on EndoVis2017 Dataset

Instrument	Total Cutout	Part Count Inferred by MLLM				GT
		1	2	3	Result	
Large Needle Driver	1197	156	213	828	3	3
Prograsp Forceps	1051	55	232	764	3	3
Bipolar Forceps	657	189	132	336	3	3
Ultrasound Probe	449	294	103	52	1	1
Curved Scissors	351	54	205	92	2	2
Vessel Sealer	386	122	53	211	3	3
Grasping Retractor	193	46	20	127	3	3

Table 2. Part Count Inference Results on EndoVis2018 Dataset

Instrument	Total Cutout	Part Count Inferred by MLLM				GT
		1	2	3	Result	
Large Needle Driver	278	58	96	124	3	3
Prograsp Forceps	972	167	232	573	3	3
Bipolar Forceps	1757	395	132	1230	3	3
Ultrasound Probe	156	103	34	52	1	1
Curved Scissors	1554	205	1257	92	2	2
Suction Instrument	272	159	31	82	1	1
Clip Applier	44	6	26	12	2	2

where τ denotes a confidence threshold used to ignore low-confidence predictions, and it is set to 0.5 based on our grid search. We then apply the following objective function for optimization:

$$\min_{\substack{\mathcal{T}^t,\theta_a,\\ \{\theta^c_{\text{seg}}\}_{c\in\mathbf{C}},\\ \{\theta^n_{\text{Dec}}\}_{n=1}^N}} \sum_{c\in\mathbf{C}} \mathcal{L}_{\text{ce}}\left(\theta^c_{\text{seg}}\left(\theta_{\text{Dec}}\left(f_{\text{Enc}}\left(x,\theta_a\right)\right),\mathcal{T}^t\right),y^c\right). \tag{12}$$

This design ensures that, during backpropagation, the loss computed from the prediction and ground truth of a specific class is only propagated to the corresponding components. As a result, each module learns knowledge that is specific and relevant to its associated class. In the subsequent training process, each episode consists of two stages. First, we enable the positive forward transfer. During this stage, the attention layers in the decoder, the segmentation heads for previously learned and regular classes, the adapter, and the prompt parsing tree are all frozen. Only the prompt partition corresponding to the new class is updated. Once the new class has been trained, its prompt partition is inserted into the current prompt parsing tree. Next, we conduct iterative self-reflection, where all modules—including the decoder, segmentation heads, adapter, and prompt parsing tree are frozen. In this stage, we train the f_{dign} as introduced in the manuscript. The output from the optimized f_{dign} is then used as the updated prompt parsing tree for the next episode. During inference, all instrument-aware prompts are retrieved and fused according to the aforementioned strategy to produce the final segmentation mask.

5. Discussion

Although prompt-based approaches and foundation models have recently emerged in surgical instrument segmentation, none have addressed the Class-Incremental Learning (CIL) setting. To the best of our knowledge, this work is the first to integrate prompt tuning with the Segment Anything Model (SAM) specifically for surgical instrument class incremental segmentation, thereby bridging a significant gap in the field. The complexity of this task stems from the fine-grained nature of surgical scenes. Unlike natural domains that typically involve coarse-grained class shifts (e.g., distinguishing a bird from a tree), surgical CIL requires discriminating between instruments with highly similar metallic textures and slim structures, such as differentiating an L-hook from a J-hook. Our method has been extensively validated on two prevalent surgeries, Nephrectomy (EndoVis 2017/2018) and Cholecystectomy (CholecSeg8k/M2CAI-Seg). In future work, we will extend our framework to additional procedures, such as prostatectomy.