BEYOND PROMPT PREDICTION: RESPONSE PREFERENCE LEARNING FOR AUTOMATIC SAM PROMPTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Automatic prompting is essential for extending the Segment Anything Model (SAM) to fully automated segmentation pipelines. Existing approaches typically formulate this problem as direct prompt prediction, where a model learns to generate prompts expected to produce accurate SAM segmentation results. However, SAM responses can vary substantially even for nearby prompts, making it challenging to determine desirable prompts without observing the responses they actually induce. This motivates us to investigate whether prompts can be determined after observing their induced SAM responses. We therefore decouple automatic prompting into candidate generation and response-based prompt determination, and find that even random exploration can discover candidate prompts with strong segmentation potential. This finding indicates that the central challenge of automatic prompting may not be generating a desirable prompt directly, but determining which candidate is preferable based on its realized SAM response. Based on this insight, we propose Response Preference Prompting (RePP), which performs automatic SAM prompting through response preference learning. Given a current prompt, RePP explores a set of candidate prompt updates and learns a preference model to determine which update to retain based on its induced SAM response. To improve exploration efficiency, we further introduce a lightweight multi-point generator that guides a more focused search over candidate updates than random sampling does. Across VOC, COCO, and NWPU, RePP exceeds the strongest compared automatic prompting methods by 4.2–10.7 AP points. Applying its learned response preference model to SAC-Prompt, Matcher, and PPD further improves AP by 15.4, 5.3, and 4.0 points, respectively.

1 INTRODUCTION

The Segment Anything Model (SAM) (Kirillov et al., 2023; Ravi et al., 2025; Carion et al., 2026) has demonstrated remarkable generalization across diverse visual domains through its promptable segmentation paradigm. However, this paradigm requires external prompts to specify target objects, typically in the form of points, boxes, or coarse masks provided by humans. This reliance on manual input limits SAM's scalability in fully automated pipelines and has motivated research on automatic SAM prompting.

Existing approaches to automatic SAM prompting generally follow two directions. One adapts SAM to learn task-specific prompt representations, often requiring retraining of the foundation model (Chen et al., 2024a; He et al., 2024; Xie et al., 2025). To reduce adaptation costs, a growing line of work instead keeps SAM frozen and uses an external generator to produce prompts such as points, boxes, or coarse masks (Liu et al., 2024; Huang et al., 2024; Zhang et al., 2025; Liu et al., 2026). Despite differences in generator design, these methods largely treat automatic prompting as a prompt prediction problem: the generator is trained to predict prompts expected to induce accurate SAM masks. However, SAM responses can be highly sensitive to input prompts and may vary substantially even between nearby prompts, making it difficult to determine an effective prompt before observing its actual SAM response. As shown in Figure 1(a), the predicted prompt yields a mask with an IoU of 0.580, whereas a nearby prompt yields one with a markedly higher IoU of 0.967.

This limitation raises a natural question: can effective prompts be found without directly predicting them? We investigate this through a diagnostic experiment. Starting from an initial point prompt,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

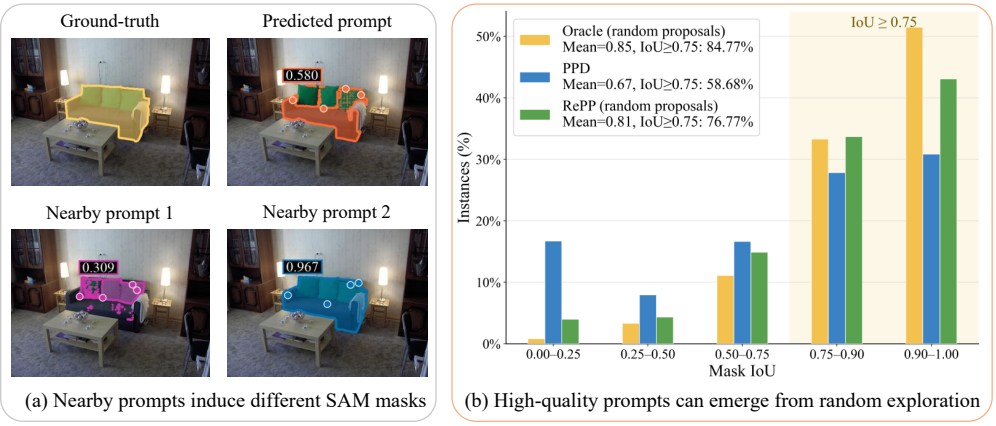

Figure 1: Motivation for response preference learning. (a) Nearby prompt configurations induce markedly different SAM masks; boxed values show mask IoU. (b) Per-instance mask IoU distributions compare the prompt prediction method PPD with random exploration, where the latter retains candidate updates using either ground-truth IoU (oracle) or learned response preference (RePP).

we randomly generate multiple candidate updates at each step by adding a positive or negative point. We obtain a SAM mask for each candidate and retain the update with the highest ground-truth mask IoU. As shown in Figure 1(b), this random exploration with oracle selection produces masks with IoU ≥ 0.75 for 84.8% of instances, compared with 58.7% for PPD (Liu et al., 2026), a prompt prediction method. The distribution shows that even random exploration can yield effective prompts when the induced SAM responses are reliably distinguished. This suggests that the learning objective for automatic SAM prompting could shift from predicting a desirable prompt directly to determining which candidate update is preferable after observing its SAM response.

Based on this insight, we propose Response Preference Prompting (RePP), which formulates automatic SAM prompting as a response preference learning problem. Rather than directly predicting the final prompt, RePP learns to identify which candidate update is preferable according to the SAM response it induces. Given a current prompt, RePP constructs a set of candidate prompt updates and queries the frozen SAM to obtain their corresponding segmentation responses. A learned preference model then evaluates these realized responses and determines which candidate to retain. This replaces the oracle IoU-based decision in our diagnostic experiment with a learned preference criterion. As shown in Figure 1(b), learned preference recovers much of the oracle's high-IoU performance. To further improve candidate exploration efficiency, we introduce a lightweight generator to guide a more focused search. The generator adopts a multi-point proposal scheme, where each candidate update jointly modifies the states of multiple point locations, reducing the number of SAM interactions required by sequential point-wise exploration. RePP thus decouples candidate generation from prompt determination, with the generator guiding where to explore and the preference model determining which candidate to retain.

We evaluate RePP on VOC, COCO, and NWPU, spanning natural images and remote-sensing imagery. Its multi-point variant exceeds the strongest compared automatic prompting method by 10.7, 6.7, and 4.2 AP points, respectively. Applying response preference to SAC-Prompt (Na et al., 2025), Matcher (Liu et al., 2024), and PPD (Liu et al., 2026) also improves AP by 15.4, 5.3, and 4.0 points, respectively, showing that it benefits different prompt generation strategies. Our contributions are summarized as follows:

- We introduce RePP, which reframes automatic SAM prompting as response preference learning. This allows prompt updates to be determined after observing their induced SAM responses, without requiring the final prompt to be predicted in advance.
- We develop a lightweight multi-point generator that guides a more focused search over candidate updates, reducing the number of SAM interactions required by sequential point-wise exploration.
- We show that response preference can be incorporated into existing prompt prediction methods, further improving their segmentation accuracy with modest additional latency.

2 Related Work

SAM (Kirillov et al., 2023; Ravi et al., 2025; Carion et al., 2026) establishes a promptable segmentation paradigm with strong zero-shot generalization, motivating broad adaptation to downstream tasks and domains (Chen et al., 2024b; Tai et al., 2025; Xu et al., 2025; Lu et al., 2026; Mei et al., 2026). However, its reliance on externally provided prompts limits fully automated segmentation and has motivated research on automatic prompt generation. Existing approaches broadly follow two directions. The first adapts SAM to learn task-specific prompt representations (Chen et al., 2024a; He et al., 2024; Na et al., 2025; Xie et al., 2025), typically by introducing trainable components or fine-tuning parts of SAM. For example, SAC (Na et al., 2025) combines an auxiliary prompt generator with task-specific SAM adaptation for nuclei segmentation. Such approaches improve domain-specific performance but require modifying or retraining SAM.

The second direction keeps SAM frozen and generates explicit visual prompts through external prompting mechanisms. Early approaches commonly employ object detectors or lightweight segmentation models to generate box prompts (Luo et al., 2024; Qiao et al., 2025) or coarse mask prompts (Zhang et al., 2025; Wang et al., 2025). More recently, point prompting has attracted increasing attention due to its flexibility and low interaction cost. Unlike boxes or masks, however, point prompts admit multiple effective configurations for the same object, making automatic point generation inherently ambiguous. Existing methods address this ambiguity through different prompt generation mechanisms. Some directly derive point prompts from semantic correspondence or learned localization cues, such as feature matching (Liu et al., 2024; Bo et al., 2026) and prompt confidence maps (Na et al., 2025; Chen et al., 2025). Others formulate point prompting as an iterative prediction process, in which a model progressively updates the current prompt over multiple steps. At each step, the model predicts a prompt edit action, such as adding a positive or negative point, and carries the updated prompt forward for subsequent refinement. To learn such actions, existing methods impose foreground–background constraints (Huang et al., 2024), optimize feature-space relationships (Liu et al., 2025), or further exploit changes in SAM mask quality as optimization signals (Shi et al., 2026; Liu et al., 2026). Despite their different designs, existing methods largely treat automatic prompting as direct prompt prediction, requiring desirable prompts to be determined before observing their actual SAM responses. Yet SAM responses can vary substantially even across nearby prompts, making such prediction inherently difficult. In contrast, RePP determines prompt updates from their realized SAM responses through response preference learning.

3 Methodology

3.1 RePP Formulation

Our diagnostic experiment suggests that effective prompting does not necessarily require directly predicting the desired prompt. Instead, candidate updates can be explored first and then determined from their actual SAM responses. Based on this observation, we formulate automatic SAM prompting as a response preference learning problem and propose RePP. As illustrated in Figure 2, RePP performs iterative prompt refinement by decoupling candidate generation from prompt determination. At each step, candidate updates are generated from the current prompt state through either random exploration or a learned multi-point generator. The frozen SAM produces their corresponding segmentation responses, which are compared by the response preference model to determine the update retained for the next step. The retained prompt and response then form the state for subsequent refinement.

Formally, let P_t denote the current prompt at iteration t and u_t the update applied to it. The iterative refinement process can be generally written as

$$P_{t+1} = P_t \oplus u_t, \tag{1}$$

where $\oplus$ denotes prompt modification. The key difference between direct prompt prediction and RePP lies in how u_t is determined.

Under the conventional prompt prediction formulation, a learned predictor $\mathcal{F}_\phi$ directly produces the update from the current prompting state s_t: $u_t = \mathcal{F}_\phi(s_t)$. The update is thus determined before its resulting SAM response is observed, requiring the predictor to implicitly anticipate how SAM will respond to the predicted prompt.

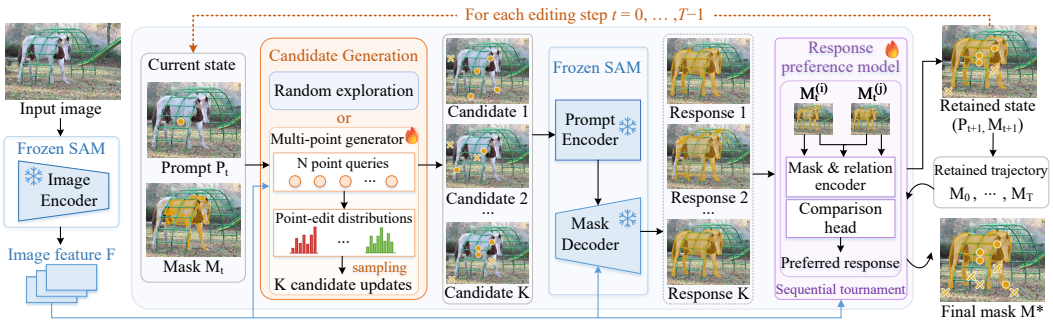

Figure 2: Overview of RePP. Circles and crosses denote positive and negative point prompts.

RePP instead separates candidate generation from the determination of the retained update:

$$u_t = \mathcal{D}_\theta(\mathcal{U}_t, \mathcal{M}_t), \qquad u_t \in \mathcal{U}_t,$$
$$\mathcal{U}_t = \mathcal{G}(s_t) = \{u_t^{(k)}\}_{k=1}^K, \qquad \mathcal{M}_t = \{\mathcal{S}_I(P_t \oplus u_t^{(k)})\}_{k=1}^K. \tag{2}$$

Here, $\mathcal{G}$ denotes candidate generation, $\mathcal{D}_\theta$ response-based prompt determination, and $\mathcal{S}_I$ the frozen SAM conditioned on image I. While $\mathcal{G}$ defines which prompt updates are explored, $\mathcal{D}_\theta$ determines which candidate in $\mathcal{U}_t$ to retain from the corresponding SAM responses $\mathcal{M}_t$. This separation allows candidate generation to vary without changing the determination process: $\mathcal{G}$ can use random exploration, as in our diagnostic experiment, or a learned generator for a more focused search. In either case, the retained update is determined from realized SAM responses rather than predicted in advance.

We next describe how $\mathcal{D}_\theta$ learns response preferences from SAM responses, followed by the multi-point generator $\mathcal{G}$ for efficient candidate exploration.

3.2 Response Preference Learning

The response-based determination in Eq. 2 is implemented by a pairwise response preference model, which learns to identify the preferable prompt update from two SAM responses. We next describe its architecture, preference learning objective, and use for candidate determination.

Response preference model. Given two candidate responses $M_t^{(i)}$ and $M_t^{(j)}$, together with the frozen SAM feature F, the preference model predicts their relative segmentation quality. The two responses are first encoded by a shared mask encoder, while a separate relation branch captures their spatial difference and overlap, denoted by Δ_{ij}. These representations, together with F, are projected into spatial tokens and jointly modeled by a lightweight Transformer, followed by a prediction head that produces an ordered preference score:

$$\widetilde{C}^{ij} = \mathcal{H}_\theta\left(F, M_t^{(i)}, M_t^{(j)}, \Delta_{ij}\right), \tag{3}$$

where $\mathcal{H}_\theta$ denotes the response preference network. To ensure that reversing the candidate order also reverses the preference, we define

$$C^{ij} = \frac{1}{2}\left(\widetilde{C}^{ij} - \widetilde{C}^{ji}\right). \tag{4}$$

A positive C^{ij} indicates that $M_t^{(i)}$ is preferred over $M_t^{(j)}$. Further architectural details are provided in Appendix A.

Preference learning. We generate preference supervision through random prompt exploration with the frozen SAM. At each prompt state, legal updates are randomly sampled and independently applied to obtain their SAM responses. Each response is evaluated by its IoU with the matched ground-truth mask, and the best candidate advances to the next state. Repeating this process yields response pairs across different refinement stages for preference learning.

For two responses $M_t^{(i)}$ and $M_t^{(j)}$ from the same prompt state, we use their IoUs $q_t^{(i)}$ and $q_t^{(j)}$ as supervision. Their IoU difference is converted into a soft preference target, and the preference model is optimized with a soft binary cross-entropy objective:

$$\mathcal{L}_{\text{pref}} = \text{BCE}\left(\sigma(C^{ij}), \sigma\left(\frac{q_t^{(i)} - q_t^{(j)}}{\tau}\right)\right), \tag{5}$$

where C^{ij} denotes the predicted preference logit, $\sigma(\cdot)$ is the sigmoid function, and τ controls the smoothness of the IoU-derived target. This objective encourages predicted preferences to reflect the relative quality of response pairs while preserving the preference degree implied by their IoU difference.

Preference-based determination. Given the response set $\mathcal{M}_t$ at iteration t, we use the learned preference model to determine which candidate update in $\mathcal{U}_t$ should be retained. Specifically, the candidate responses are compared through a sequential tournament: a current winner is successively compared with the remaining candidates and replaced whenever another response is preferred. The final winner determines the update u_t in Eq. 2.

3.3 Efficient Candidate Generation

Random exploration provides a simple realization of the candidate generator $\mathcal{G}$ in Eq. 2, but unguided sampling may spend many SAM interactions on less useful regions. We therefore learn a lightweight multi-point generator that guides a more focused search over candidate updates.

Prompt-state representation. We discretize the target region into a regular grid $\mathcal{C} = \{\mathbf{c}_i\}_{i=1}^{L}$, where each grid location $\mathbf{c}_i$ is assigned a state $z_i^t \in \{+, -, \emptyset\}$, indicating a positive point, negative point, or inactive location. The current SAM prompt P_t is thus defined by:

$$P_t = \left(P_t^+, P_t^-\right), \qquad P_t^+ = \{\mathbf{c}_i \mid z_i^t = +\}, \quad P_t^- = \{\mathbf{c}_i \mid z_i^t = -\}. \tag{6}$$

Changing a grid state therefore corresponds to adding, relabeling, or removing a point prompt.

To generate subsequent prompt updates, the generator represents each grid location using four cues: its local frozen-SAM feature $\mathbf{f}_i$, current prompt state z_i^t, relative coordinate $\mathbf{u}_i$ within the target region, and the aligned value m_i^t from the current SAM mask. The generator input is therefore written as

$$\mathcal{O}_t = \left\{\left(\mathbf{f}_i, z_i^t, \mathbf{u}_i, m_i^t\right)\right\}_{i=1}^{L}. \tag{7}$$

These grid-wise observations $\mathcal{O}_t$ are encoded into a shared representation $\mathbf{F}_t$ that captures local spatial context and interactions across candidate locations.

Multi-point candidate generation. Given the shared representation $\mathbf{F}_t$, the generator employs N learnable point queries to jointly form each candidate update. Each query attends to $\mathbf{F}_t$ while interacting with the other queries, producing a distribution over valid prompt edits. Each edit corresponds to assigning a new prompt state to a selected grid location. Accordingly, the k-th candidate update $u_t^{(k)} \in \mathcal{U}_t$ in Eq. 2 is represented as

$$u_t^{(k)} = \left\{a_{t,1}^{(k)}, \ldots, a_{t,N}^{(k)}\right\}, \qquad a_{t,n}^{(k)} = (\mathbf{c}_i, \tilde{z}_i), \tag{8}$$

where each query contributes an edit $a_{t,n}^{(k)}$ that assigns an updated prompt state $\tilde{z}_i$ to a selected grid location $\mathbf{c}_i$.

Multiple candidates are obtained by sampling different edit combinations from the query-specific distributions, forming the candidate set $\mathcal{U}_t$. All edits within one candidate are applied jointly before querying SAM, so an N-point update requires only one SAM decoding step. This allows multiple prompt modifications to be explored with fewer sequential SAM interactions. Architectural details of the multi-point generator are provided in Appendix A.

Generator learning. The generator is trained using the same random exploration data constructed for response preference learning. Under the multi-point formulation, each candidate update in the shared data is represented as the N-edit set in Eq. 8, together with its resulting SAM response and corresponding IoU. These evaluated updates provide supervision for learning which multi-point edits are more likely to improve the current response.

Since the N edits in each candidate update form an unordered set, we use Hungarian matching (Kuhn, 1955) to align them with the N point queries during training. Let $p_{t,n}^{(k)}$ denote the probability assigned by the n-th query to its matched edit. We optimize the generator with

$$\mathcal{L}_{\text{gen}} = -\frac{1}{N}\sum_{n=1}^{N} |r_t^{(k)}| \left[\mathbb{I}\Big(r_t^{(k)} \geq 0\Big) \log p_{t,n}^{(k)} + \mathbb{I}\Big(r_t^{(k)} < 0\Big) \log\Big(1 - p_{t,n}^{(k)}\Big)\right], \tag{9}$$

where $r_t^{(k)}$ is the IoU change $q_t^{(k)} - q_t$, normalized by $1 - q_t$ for improvements and by q_t for degradations. The objective promotes beneficial edits and suppresses detrimental ones, with larger response changes receiving stronger supervision.

3.4 Iterative Inference

At inference, RePP independently refines the prompt for each instance over T iterations, starting from the initial prompt P_0 and its corresponding SAM response M_0. At each iteration $t = 0, \ldots, T-1$, the generator produces a candidate set $\mathcal{U}_t$ from the current observation $\mathcal{O}_t$. Each candidate update is applied to the current prompt P_t and evaluated by the frozen SAM, yielding the candidate response set $\mathcal{M}_t$. The response preference model then compares these responses and selects the update u_t to retain. Applying the selected update yields the next prompt $P_{t+1} = P_t \oplus u_t$ and its corresponding response M_{t+1}, which are carried forward to the subsequent iteration.

Since successive refinements do not necessarily improve the response monotonically, we retain all prompt–response states encountered during refinement, $\mathcal{P}_{0:T} = \{P_0, \ldots, P_T\}$ and $\mathcal{M}_{0:T} = \{M_0, \ldots, M_T\}$. The final prompt is then selected using the learned determination module $\mathcal{D}_\theta$:

$$P^* = \mathcal{D}_\theta\left(\mathcal{P}_{0:T}, \mathcal{M}_{0:T}\right), \qquad P^* \in \mathcal{P}_{0:T}, \tag{10}$$

with the response M^* corresponding to P^* taken as the final segmentation output.

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate RePP on three instance segmentation benchmarks spanning natural and remote-sensing imagery. VOC (Everingham et al., 2010) contains diverse object categories in natural scenes. We train on VOC2012 train and evaluate on VOC2012 val. COCO (Lin et al., 2014) provides large-scale natural images with more complex scenes and object distributions. We train on a fixed random subset of 2,000 train2017 images and evaluate on the full val2017 split. NWPU (Cheng et al., 2014) contains remote-sensing images with distinct object scales and visual characteristics. Following common practice (Chen et al., 2024a), we use an 80%/20% train/test split.

Evaluation metrics. Following standard instance segmentation protocols, we report mask AP averaged over IoU thresholds from 0.50 to 0.95, together with AP_{50} and AP_{75}. For the response preference model, we additionally report preference accuracy, measuring whether the predicted ordering of response pairs agrees with that induced by their ground-truth IoUs.

Implementation details. All experiments use a frozen SAM (Kirillov et al., 2023). Instance-level bounding boxes are obtained using RT-DETR (Zhao et al., 2024), and their center points are used to initialize the positive point prompts. The response preference model and multi-point generator are trained with Adam for 20,000 iterations using an initial learning rate of 3×10^{-4}, decayed by 0.5 every 5,000 iterations. The temperature τ is set to 0.03. During inference, random exploration uses $K = 6$ candidates, while the multi-point generator uses $K = 3$. Sensitivity analyses for these hyperparameters are presented in Appendix B. All ablation experiments are performed on VOC, which offers a compact yet diverse benchmark for evaluating individual design choices.

Table 1: Ablation of response determination and preference formulations.

Determination	Formulation	Pref. Acc.	AP	AP_{50}	AP_{75}
Oracle	GT IoU	100.0	65.6	84.9	71.4
Random	–	50.1	12.3	22.8	11.7
SAM Score	Decoder Score	55.6	23.7	36.1	24.6
Learned Preference	IoU Regression	76.8	55.4	77.8	59.9
	Listwise Ranking	79.2	58.5	80.1	63.3
	Bradley–Terry	79.6	58.8	79.8	63.9
	Pairwise Hinge	80.1	59.0	79.8	64.2
	Pairwise BCE	80.0	58.7	80.5	64.2
	Pairwise Soft BCE (Ours)	**80.7**	**59.8**	**82.0**	**65.3**

4.2 Ablation Studies

We conduct ablation studies on VOC to examine three key aspects of RePP: how response preference is learned, how candidate updates are organized for efficient refinement, and how preference contributes at different stages of inference.

Analysis of response preference learning. We first examine whether realized SAM responses can effectively guide candidate determination. As shown in Table 1, random determination performs poorly, while the SAM decoder score provides only limited improvement. Learning response preference substantially improves segmentation performance and narrows the gap to the oracle based on ground-truth IoU, showing the benefit of explicitly modeling quality differences between SAM responses. We further compare different preference learning formulations. IoU regression performs worse, as estimating the absolute quality of each response is more difficult than determining their relative quality. Bradley–Terry (Bradley & Terry, 1952) and listwise ranking improve upon regression by learning relative ordering. However, both still derive preferences from separately predicted response scores, leaving the relation between paired responses implicit. In contrast, direct pairwise comparison explicitly models their relative quality and achieves stronger performance. Among them, soft BCE, defined in Eq. 5, achieves the highest preference accuracy and segmentation AP, suggesting that preserving the preference degree implied by IoU differences provides richer supervision than binary ordering alone.

Analysis of candidate generation. We next examine how candidate generation affects refinement accuracy and efficiency. Latency is measured on a single NVIDIA RTX 4090 GPU. As shown in Table 2, random exploration achieves strong performance, but its unguided nature requires a larger candidate set ($K = 6$) for sufficient exploration, leading to substantially higher inference cost. In comparison, the learned multi-point generator uses fewer candidates ($K = 3$) and performs multiple point updates within each refinement step, substantially reducing inference cost while retaining competitive segmentation accuracy. We further vary the number of point updates N and refinement iterations T. Increasing N generally shortens the refinement process and reduces latency, but an overly large N degrades accuracy because fewer intermediate SAM responses are available to guide subsequent updates. We therefore adopt $N = 4$ and $T = 3$ as the default setting, which preserves reasonably strong segmentation performance while keeping the inference cost low.

Role and generality of response preference. We further analyze how response preference contributes during refinement. In Table 3, each setting (a, b) denotes the determination strategy used for per-step candidate updates and the final output, respectively. Applying preference at either stage clearly improves over $(\text{Greedy}, \text{Last})$, while combining both gives the strongest non-oracle result. This confirms that per-step and final preference play complementary roles in guiding refinement and determining the final response. We also examine whether these benefits transfer to existing automatic prompting methods. Incorporating response preference consistently improves SAC-Prompt, Matcher, and PPD, with AP gains of 15.4, 5.3, and 4.0 points, respectively, while introducing only modest additional latency. These consistent improvements show that response preference can complement different candidate generation strategies. Visual comparisons of these methods with and without response preference are provided in Appendix C.

Table 2: Ablation of candidate generation and multi-point refinement.

Candidate Source	N	T	K	Edits	AP	AP_{50}	AP_{75}	Latency (s)
Initial Prompt	–	–	–	–	21.7	33.4	22.5	0.234
Random	1	12	6	12	59.8	82.0	65.3	1.000 (+0.766)
Multi-point	1	12	3	12	59.6	81.7	64.3	0.734 (+0.500)
	2	6	3	12	59.5	81.9	65.0	0.518 (+0.284)
	3	4	3	12	58.1	81.0	63.2	0.443 (+0.209)
	4	4	3	16	58.4	81.8	63.5	0.460 (+0.226)
	4	3	3	12	58.1	80.7	63.1	0.409 (+0.175)
	6	2	3	12	56.5	80.2	60.8	0.373 (+0.139)
	12	1	3	12	53.8	78.0	58.4	0.339 (+0.105)

Table 3: Analysis of the role and generality of response preference.

Analysis	Setting	AP	AP_{50}	AP_{75}	Latency (s)
RePP (Multi-point)	(Greedy, Last)	43.6	69.0	44.4	0.279
	(Greedy, Preference)	53.4 (+9.8)	76.8	57.4	0.314 (+0.035)
	(Preference, Last)	55.5 (+11.9)	79.3	60.1	0.394 (+0.115)
	(Preference, Preference)	**58.1** (+14.5)	**80.7**	**63.1**	0.409 (+0.130)
	(Oracle, Oracle)	61.1	82.7	66.7	–
Plug-in	SAC-Prompt	37.5	62.4	38.5	0.309
	SAC-Prompt + Preference	**52.9** (+15.4)	**77.4**	**57.3**	0.425 (+0.116)
	Matcher	43.3	69.8	45.0	5.062
	Matcher + Preference	**48.6** (+5.3)	**73.3**	**51.6**	5.105 (+0.043)
	PPD	47.4	67.9	51.2	0.681
	PPD + Preference	**51.4** (+4.0)	**72.1**	**55.1**	0.807 (+0.126)

4.3 Comparison with Existing Automatic Prompting Methods

We compare RePP with representative automatic prompting methods, including SAC-Prompt (Na et al., 2025), Matcher (Liu et al., 2024), AlignSAM (Huang et al., 2024), FM-PPO (Liu et al., 2025), PPD (Liu et al., 2026), and PromptPilot (Shi et al., 2026). For SAC-Prompt, we follow the prompting strategy introduced in Segment Any Cell (Na et al., 2025) and adapt it to our setting. Since these methods were originally designed for whole-image prompting, we adapt them to the same instance-level setting as RePP for fair comparison, using RT-DETR predicted boxes to identify individual instances. We also include SAM (Point), using predicted box centers as point prompts, and SAM (Box), using predicted boxes as box prompts.

VOC dataset. Table 4 shows clear gains for RePP on VOC. The random variant reaches 59.8 AP and 65.3 AP_{75}, clearly improving over the existing automatic prompting methods. The multi-point variant retains strong performance with 58.1 AP while substantially reducing the inference cost, as analyzed in Table 2.

COCO dataset. The advantage of RePP also extends to COCO. The multi-point variant achieves 45.2 AP and 48.8 AP_{75}, with the random variant reaching a comparable 44.6 AP. Both variants outperform the compared automatic prompting methods across all three metrics and achieve higher AP and AP_{75} than the SAM box-prompt reference.

NWPU dataset. On the remote-sensing NWPU dataset, RePP maintains consistent gains across both variants, achieving 62.6 AP with random generation and 60.1 AP with multi-point generation. The improvements over existing automatic prompting methods, together with the results on VOC and COCO, demonstrate the effectiveness of response preference across different image domains. Qualitative comparisons with existing methods are presented in Appendix D.2.

Table 4: Results on the VOC, COCO, and NWPU datasets.

Method	VOC			COCO			NWPU		
	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}	AP	AP_{50}	AP_{75}
SAM (Point)	21.7	33.4	22.5	21.9	37.3	22.5	31.4	51.4	34.3
SAM (Box)	55.0	77.2	59.8	44.3	68.5	47.8	49.0	79.3	51.8
SAC-Prompt	37.5	62.4	38.5	36.6	61.2	37.9	55.9	85.0	59.4
Matcher	43.3	69.8	45.0	37.7	63.6	38.5	46.0	78.8	50.4
AlignSAM	38.9	64.7	38.9	35.7	60.8	35.1	42.2	69.3	45.2
FM-PPO	37.5	63.1	37.4	32.2	56.5	30.8	42.4	68.0	45.7
PPD	47.4	67.9	51.2	38.5	63.0	40.3	47.7	78.2	54.3
PromptPilot	43.6	63.2	45.7	37.1	61.5	38.5	45.5	76.4	50.8
RePP (Random)	**59.8**	**82.0**	**65.3**	44.6	68.3	48.4	**62.6**	**89.8**	**66.8**
RePP (Multi-point)	58.1	80.7	63.1	**45.2**	**68.8**	**48.8**	60.1	87.0	64.6

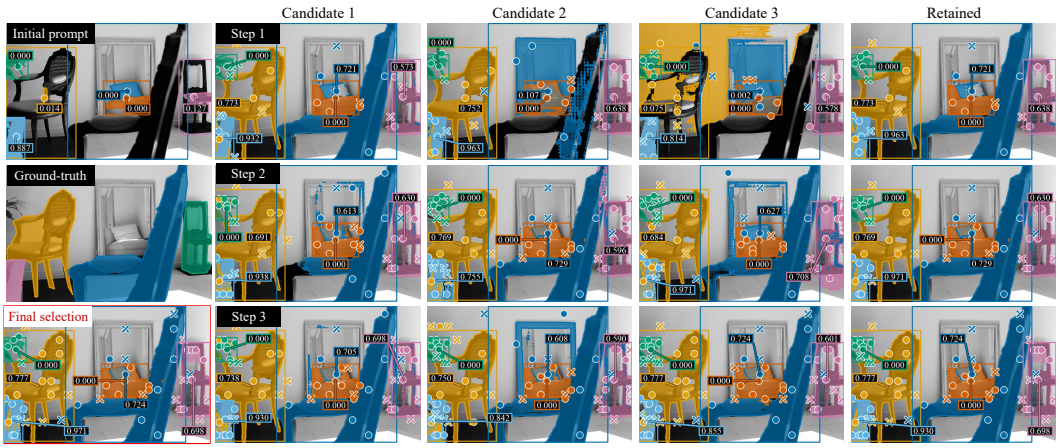

Figure 3: Visualization of RePP refinement with the multi-point generator. Circles and crosses denote positive and negative point prompts, respectively, while boxed values report the IoU of each predicted instance mask against its ground truth. IoU values are shown for reference only.

4.4 Visualization

Figure 3 visualizes the iterative refinement of RePP with the multi-point generator. Starting from the initial prompt, the generator produces three candidate updates at each iteration, which can induce noticeably different SAM responses across instances. The preference model determines the preferred candidate for each instance, and the retained update is applied to that instance's prompt for the next iteration. After refinement, final selection determines the output from the refinement trajectory. The visualization shows how response preference allows RePP to adapt prompt decisions to the realized SAM behavior of individual instances. Additional examples are provided in Appendix D.3.

5 Conclusion

This paper presents RePP, a response preference learning framework for automatic SAM prompting. RePP decouples candidate generation from prompt determination, allowing prompt updates to be determined after observing their induced SAM responses rather than directly predicting a desirable prompt in advance. Corresponding to these two decoupled components, RePP introduces a lightweight multi-point generator to improve candidate exploration efficiency and a learned preference model to support prompt determination from realized SAM responses. Experiments across diverse datasets show that RePP consistently outperforms existing automatic prompting methods on both natural and remote-sensing images. Further gains obtained by incorporating response preference into existing prompt prediction methods demonstrate its broader effectiveness and generality.

AI USE STATEMENT

In this work, we did not use generative AI tools for any tasks requiring disclosure. Additionally, we used generative AI tools to assist with editing software code, analyzing existing literature, searching for information, editing the manuscript to improve readability, identifying relevant literature, formatting references, and proposing the title or keywords. We reviewed all AI-assisted work and verified the resulting code, literature-related information, references, and manuscript edits for correctness and consistency with the original sources and our research. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed information to support the reproducibility of RePP throughout the main paper, appendix, and supplementary materials. The overall formulation and refinement procedure are described in Section 3.1, while the architecture, supervision, and training objective of the response preference model are presented in Section 3.2. The prompt-state representation, multi-point candidate generation, and generator learning objective are described in Section 3.3, and the complete test-time refinement and final response determination procedure is specified in Section 3.4. Experimental settings, including dataset splits, evaluation metrics, model initialization, training settings, and inference configurations, are provided in Section 4.1. Additional implementation details are given in Appendix A.1 for input preparation and shared candidate construction, Appendix A.2 for the response preference model, and Appendix A.3 for the multi-point generator. Hyperparameter sensitivity is further reported in Appendix B, while Appendix D.1 details the unified instance-level protocol used for comparison with existing prompting methods. Anonymous source code, including implementations for model training, inference, and evaluation, is available at `https://anonymous.4open.science/r/repp-iclr27-80D0/`.

REFERENCES

Yuntian Bo, Yazhou Zhu, Piotr Koniusz, and Haofeng Zhang. Focus on background: Exploring SAM's potential in few-shot medical image segmentation with background-centric prompting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30032–30041, 2026.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. The method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollar, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. *International Conference on Learning Representations*, pp. 138846–138923, 2026.

Keyan Chen, Chenyang Liu, Hao Chen, Haotian Zhang, Wenyuan Li, Zhengxia Zou, and Zhenwei Shi. RSPrompter: Learning to prompt for remote sensing instance segmentation based on visual foundation model. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–17, 2024a.

Yi Chen, Muyoung Son, Chuanbo Hua, and Joo-Young Kim. AoP-SAM: Automation of prompts for efficient segmentation. In *AAAI Conference on Artificial Intelligence*, volume 39, pp. 2284–2292, 2025.

Zhiwen Chen, Zhiyu Zhu, Yifan Zhang, Junhui Hou, Guangming Shi, and Jinjian Wu. Segment any event streams via weighted adaptation of pivotal tokens. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3890–3900, 2024b.

Gong Cheng, Junwei Han, Peicheng Zhou, and Lei Guo. Multi-class geospatial object detection and geographic image classification based on collection of part detectors. *ISPRS Journal of Photogrammetry and Remote Sensing*, 98:119–132, 2014.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88 (2):303–338, 2010.

Weizhao He, Yang Zhang, Wei Zhuo, Linlin Shen, Jiaqi Yang, Songhe Deng, and Liang Sun. APSeg: Auto-prompt network for cross-domain few-shot semantic segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23762–23772, 2024.

Duojun Huang, Xinyu Xiong, Jie Ma, Jichang Li, Zequn Jie, Lin Ma, and Guanbin Li. AlignSAM: Aligning segment anything model to open context via reinforcement learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3205–3215, 2024.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pp. 740–755, 2014.

Xueyu Liu, Rui Wang, Yexin Lai, Guangze Shi, Feixue Shao, Fang Hao, Jianan Zhang, Jia Shen, Yongfei Wu, and Wen Zheng. Plug-and-play PPO: An adaptive point prompt optimizer making SAM greater. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4332–4342, 2025.

Xueyu Liu, Xiaoyi Zhang, Meilin Liu, Guangze Shi, Jia Shen, Yujie Wang, Cai Zhao, Ziyuan He, Yongfei Wu, Mingqiang Wei, and Yongle Chen. Attack for defense: Adversarial agents for point prompt optimization empowering segment anything model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6591–6600, 2026.

Yang Liu, Muzhi Zhu, Hengtao Li, Hao Chen, Xinlong Wang, and Chunhua Shen. Matcher: Segment anything with one shot using all-purpose feature matching. In *International Conference on Learning Representations*, 2024.

Yujie Lu, Jingwen Li, Sibo Ju, Yanzhou Su, He Yao, Yisong Liu, Min Zhu, and Junlong Cheng. SegMoTE: Token-level mixture of experts for medical image segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36332–36342, 2026.

Muying Luo, Tao Zhang, Shiqing Wei, and Shunping Ji. SAM-RSIS: Progressively adapting SAM with box prompting to remote sensing image instance segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–14, 2024.

Haiyang Mei, Qiming Huang, Hai Ci, and Mike Zheng Shou. RobotSeg: A model and dataset for segmenting robots in image and video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14346–14356, 2026.

Saiyang Na, Yuzhi Guo, Feng Jiang, Hehuan Ma, Jean Gao, and Junzhou Huang. Segment any cell: A SAM-based auto-prompting fine-tuning framework for nuclei segmentation. *IEEE Transactions on Neural Networks and Learning Systems*, 36(12):19986–19995, 2025.

Yang Qiao, Bo Zhong, Bailin Du, He Cai, Jinxiong Jiang, Qinhuo Liu, Aixia Yang, Junjun Wu, and Xiaoya Wang. SAM enhanced semantic segmentation for remote sensing imagery without additional training. *IEEE Transactions on Geoscience and Remote Sensing*, 63:1–16, 2025.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollar, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations*, pp. 28085–28128, 2025.

Guangze Shi, Yingjie Mi, Jia Shen, Feixue Shao, Jiarui Cao, Yexin Lai, Xueyu Liu, Rui Wang, Yongfei Wu, and Mingqiang Wei. PromptPilot: Game-theoretic multi-agent prompt optimization for segment anything. In *International Conference on Machine Learning*, 2026.

Wei-En Tai, Yu-Lin Shih, Cheng Sun, Yu-Chiang Frank Wang, and Hwann-Tzong Chen. Segment anything, even occluded. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 29385–29394, 2025.

Chenhao Wang, Zhihao Xi, Diyou Liu, Yuman Feng, Yupeng Deng, Kai Li, Jingbo Chen, Jiansheng Chen, and Yu Meng. PCP: A prompt-based cartographic-level polygonal vector extraction framework for remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 63: 1–15, 2025.

Bin Xie, Hao Tang, Bin Duan, Dawen Cai, Yan Yan, and Gady Agam. MaskSAM: Auto-prompt SAM with mask classification for volumetric medical image segmentation. In *IEEE/CVF International Conference on Computer Vision*, pp. 24423–24433, 2025.

Jianyun Xu, Song Wang, Ziqian Ni, Chunyong Hu, Sheng Yang, Jianke Zhu, and Qiang Li. SAM4D: Segment anything in camera and LiDAR streams. In *IEEE/CVF International Conference on Computer Vision*, pp. 28535–28545, 2025.

Quan Zhang, Yuxin Qi, Xi Tang, Jinwei Fang, Xi Lin, Ke Zhang, and Chun Yuan. IMDPrompter: Adapting SAM to image manipulation detection by cross-view automated prompt learning. In *International Conference on Learning Representations*, pp. 1–16, 2025.

Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. DETRs beat YOLOs on real-time object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16965–16974, 2024.

APPENDIX

This appendix provides implementation details and additional experimental evidence for RePP. We first describe the shared candidate cache, the response preference model, the multi-point candidate generator, and the inference procedure. We then examine sensitivity to the number of candidates K and the preference temperature τ. Finally, we visualize the effect of applying preference to existing prompting methods, specify the instance-level comparison protocol, and present further qualitative comparisons and RePP refinement examples.

A ADDITIONAL METHOD AND IMPLEMENTATION DETAILS

A.1 INPUT PREPARATION AND SHARED CANDIDATE CACHE

RePP processes each instance in an image independently. An RT-DETR-predicted box identifies the target instance; we enlarge the box by a factor of 1.2 about its center to define the region of interest (ROI). Within each ROI, a regular 9×9 grid provides $L = 81$ possible point locations. We initialize P_0 with a positive point at the predicted box center. Each grid location has a positive, negative, or inactive state z_i^t, as defined in Eq. 6.

As illustrated in Figure 2, RePP uses image features F extracted by the frozen SAM image encoder throughout refinement. These features comprise three FPN levels. For each grid location $\mathbf{c}_i$, we bilinearly sample a 3×3 neighborhood from each level and concatenate the sampled features to obtain its visual descriptor $\mathbf{f}_i$. The response preference model and multi-point generator use the frozen features in their respective network inputs, as detailed below.

Both learned modules are trained from a shared candidate cache. Starting from a prompt state P_t, we sample K legal candidate updates $\{u_t^{(k)}\}_{k=1}^{K}$. Each update is an unordered set of N point edits, with no two edits modifying the same grid location. We apply all edits in an update before querying the frozen SAM once, obtaining its response $M_t^{(k)}$ and ground-truth IoU $q_t^{(k)}$. The cache records the current state and, for each candidate, its edits, updated prompt, SAM response, and IoU. The candidate with the highest IoU advances the cached trajectory to the next prompt state. Repeating this process yields candidate pools at different steps of refinement.

The two modules use these pools for different learning objectives. The response preference model compares masks from the same pool, so each pair shares the image, ROI, and current prompt state. The multi-point generator learns from the recorded edit sets and the IoU changes of their induced responses relative to the current response. Ground-truth IoU is used only to construct the cache and training targets; at inference, the learned response preference model determines which candidate update to retain and which response to output.

A.2 RESPONSE PREFERENCE MODEL

The response preference model compares two SAM responses $M_t^{(i)}$ and $M_t^{(j)}$ drawn from the same candidate pool. Their masks are binarized and aligned to 64×64, and the shared frozen-SAM features F are sampled over the corresponding ROI to provide a dense feature map at the same resolution. Thus, the comparison is conditioned on both the image content and the two realized masks.

The model first encodes these inputs through three branches. A feature branch projects the ROI feature map to 64 channels using a 1×1 convolution, group normalization, GELU, and a residual convolution block. A shared mask branch applies a 3×3 convolution and a residual block to each candidate mask. A separate relation branch encodes four maps that describe how the masks differ and overlap:

$$\Delta_{ij} = \left[M_t^{(i)} - M_t^{(j)},\ |M_t^{(i)} - M_t^{(j)}|, M_t^{(i)} M_t^{(j)},\ M_t^{(i)} + M_t^{(j)} - M_t^{(i)} M_t^{(j)}\right]. \tag{11}$$

These maps represent the signed difference, absolute difference, intersection, and union of the two masks, respectively. The relation branch uses the same convolutional structure as the mask branch but has independent weights.

Table 5: Effect of candidate number K on VOC.

Candidate source	N	T	K	AP	AP_{50}	AP_{75}	Latency (s)
Random	1	12	2	55.0	78.4	59.1	0.513
	1	12	4	58.6	81.2	63.2	0.752
	1	12	6	59.8	**82.0**	65.3	1.000
	1	12	8	**60.2**	**82.0**	**65.7**	1.246
	1	12	10	60.1	81.8	65.6	1.497
Multi-point	4	3	2	56.5	79.4	61.8	0.363
	4	3	3	58.1	80.7	63.1	0.409
	4	3	4	58.9	81.1	64.5	0.459
	4	3	5	**59.1**	**81.3**	**64.8**	0.506
	4	3	6	58.9	**81.3**	63.8	0.552

The encoded visual feature, two mask features, and pair-relation feature are first pooled to 16×16. We concatenate the visual feature with each of the other three features, respectively, and pass the results through token stems. The two mask inputs share a stem, while the relation input uses a separate stem. This produces three groups of $16 \times 16 = 256$ tokens, each with a feature dimension of 128. Tokens at the same spatial location share a positional embedding across groups, and a group embedding distinguishes the first mask, the second mask, and their relation. We prepend a learned classification token and process the resulting $1 + 3 \times 256$ tokens with two pre-norm Transformer blocks. From the Transformer outputs, we form a comparison representation by concatenating four 128-dimensional vectors: the classification token, the difference between the mean tokens of the two masks, and the mean and maximum tokens of the relation group. A prediction head maps this 512-dimensional representation to the ordered logit $\widetilde{C}^{ij}$. We then exchange the two mask inputs to obtain $\widetilde{C}^{ji}$ through the same network. The final preference logit is $C^{ij} = (\widetilde{C}^{ij} - \widetilde{C}^{ji})/2$, as defined in Eq. 4; $C^{ij} > 0$ indicates a preference for $M_t^{(i)}$.

For training, we sample response pairs with different IoU gaps from the shared candidate cache described above. Both responses in a pair originate from the same prompt state, so the target measures their relative mask quality under the same image and prompt context. We randomly exchange their input order to prevent the model from associating preference with a fixed input position.

A.3 Multi-point Candidate Generator

The multi-point generator uses the grid-wise observations defined in Eq. 7. At each grid location, it combines the visual descriptor $\mathbf{f}_i$ described above with three cues from the current prompt state: an embedding of z_i^t, a Fourier encoding of the ROI-relative coordinate, and the aligned current-mask value m_i^t. Each cue is mapped to a 256-dimensional vector, and the vectors are added to form a grid token. Three residual convolution blocks first capture local relationships on the 9×9 grid. Four Transformer encoder layers with eight attention heads then allow each grid location to incorporate information from the full ROI.

To construct a candidate update, the generator uses N learnable point queries and a two-layer decoder. In each layer, the queries attend to the encoded grid tokens through cross-attention, interact with one another through self-attention, and pass through a feed-forward block. The resulting query representations are combined with the grid tokens, and a shared action head scores the three possible updated point states for each query–location pair, producing $N \times (3L)$ action logits. We decode these logits into a set of N legal edits: at each decoding step, we choose the highest-scoring query–action pair, fix its query and grid location, and update the legal actions for the remaining queries. This prevents an edit from leaving a point state unchanged or two edits in the same candidate from modifying one location. To obtain K distinct candidates, we perturb the action logits independently with Gumbel noise and repeat the decoding procedure. All N edits in a candidate are applied to the current prompt together, so each candidate requires only one frozen-SAM decoding call.

Table 6: Sensitivity to the soft preference temperature τ on VOC.

τ	AP	AP_{50}	AP_{75}
0.01	59.2	81.5	64.7
0.02	59.4	81.8	64.8
0.03	**59.8**	82.0	**65.3**
0.05	59.7	82.2	64.4
0.10	**59.8**	**82.5**	64.6

B Hyperparameter Sensitivity

Number of candidate updates. Table 5 varies the number of candidate updates evaluated in each refinement iteration. Random generation uses one edit over 12 iterations, while multi-point generation uses four edits over three iterations; both settings have a nominal budget of 12 edits. Each candidate is evaluated by the frozen SAM, followed by preference determination within the iteration and over the resulting trajectory.

Increasing K substantially benefits random exploration when the candidate pool is small: AP rises from 55.0 at $K = 2$ to 59.8 at $K = 6$. Further increasing K to 8 yields only 0.4 additional AP while raising latency from 1.000 to 1.246 s; at $K = 10$, AP remains near the same level despite another increase in latency. Multi-point generation shows a similar pattern at a lower inference cost. Moving from $K = 2$ to $K = 3$ improves AP by 1.6 points for an additional 0.047 s, whereas moving from $K = 3$ to $K = 5$ gains 1.0 AP for an additional 0.096 s. We therefore use $K = 6$ for random generation and $K = 3$ for multi-point generation.

Temperature of the soft preference target. Table 6 varies τ in the IoU-derived soft preference target while keeping random generation at $K = 6$ and $T = 12$. A smaller τ makes the target more sensitive to IoU differences, whereas a larger τ produces softer preferences between responses.

AP varies by only 0.6 points across the tested temperatures. The selected value $\tau = 0.03$ ties for the highest AP and achieves the highest AP_{75}, while $\tau = 0.10$ gives the highest AP_{50} but a lower AP_{75}. These results indicate that the preference model remains effective across the tested range, with $\tau = 0.03$ providing the strongest result at the stricter overlap threshold.

C Applying Preference to Existing Prompting Methods

We apply the learned response preference model to SAC-Prompt, Matcher, and PPD, whose different prompting strategies provide distinct candidate SAM responses. Figure 4 compares each method before and after adding preference. SAC-Prompt derives points from predicted foreground and background probabilities, but points in plausible regions do not necessarily induce an accurate SAM response. In the first row, the horse mask in column (b) extends into the background and has an IoU of 0.000; with preference, column (c) retains a response that follows the horse, reaching 0.813 IoU. The third row shows the same distinction for the dogs, whose IoUs increase from 0.003 and 0.511 in column (b) to 0.938 and 0.901 in column (c). These changes indicate that assessing the resulting masks can resolve errors that point probabilities alone do not reveal.

Matcher generates candidates through reference-feature correspondence, then scores and fuses retained masks. A strong feature match, however, does not ensure that SAM delineates the target accurately, and fusing candidates can preserve errors from individual masks. In the third row, preference improves one dog from 0.536 in column (d) to 0.933 in column (e), showing the value of comparing realized responses after candidate generation. PPD instead learns to remove points judged harmful from its feature-matched prompts. The first row illustrates a remaining difficulty: column (f) nearly misses the person (0.001 IoU), whereas preference in column (g) identifies a response with 0.652 IoU. The fourth row shows a similarly large improvement for a cyclist, from 0.227 to 0.883.

Across these examples, the original methods generate or filter prompts using different criteria, but those criteria do not always reflect the quality of the masks SAM actually produces. RePP's response preference model compares the induced masks when determining which candidate to retain. The

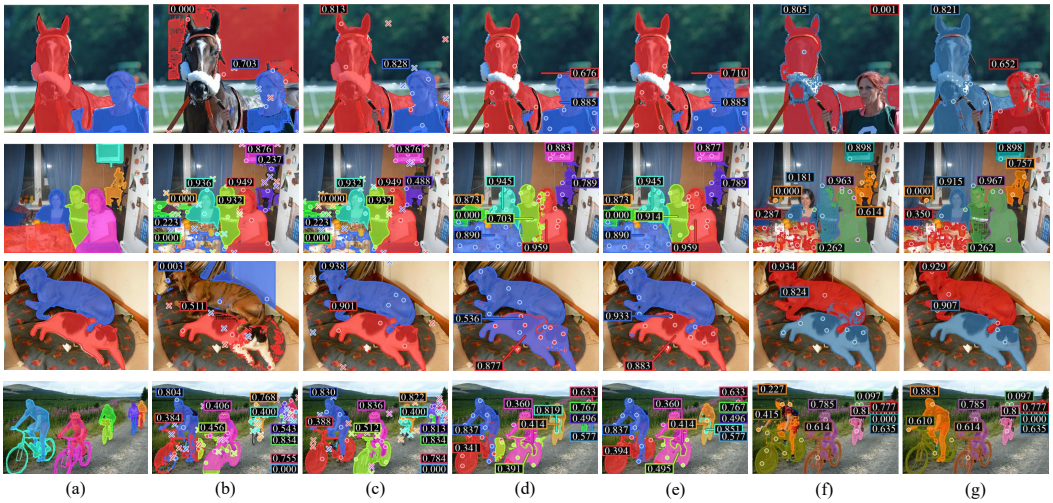

Figure 4: Visualization of existing prompting methods with response preference. (a) Ground truth; (b,c) SAC-Prompt; (d,e) Matcher; (f,g) PPD. The second column of each method pair uses preference. Circles and crosses denote positive and negative point prompts, respectively; boxed values report per-instance mask IoU.

improvements illustrated in Figure 4 are reflected in Table 3: adding preference raises AP by 15.4, 5.3, and 4.0 points for SAC-Prompt, Matcher, and PPD, respectively, with modest latency increases of 0.043–0.126 s per image.

D Additional Comparisons

D.1 Comparison Protocol

The main paper compares RePP with SAC-Prompt, Matcher, AlignSAM, FM-PPO, PPD, and PromptPilot. These methods were originally formulated for whole-image processing, where prompts or training targets can correspond to multiple objects. A resulting mask may therefore be difficult to associate with a particular target, especially when nearby objects share the same category. To ensure a fair comparison with RePP, we evaluate all methods under the same instance-level protocol. They use the same RT-DETR-predicted boxes to identify target instances and the same frozen SAM backbone for segmentation. For each instance, we enlarge its predicted box by a factor of 1.2 to define the ROI and associate prompts and training targets with the matched instance. When a method uses Dice supervision, we compute it against that instance's mask rather than the foreground of its entire class, so neighboring instances remain distinct during optimization.

Within this shared protocol, we retain each method's prompt construction. AlignSAM and RePP begin with a positive point at the predicted-box center. FM-PPO, PPD, and PromptPilot require dense initial point prompts; we follow their original feature-matching initialization while restricting the selected points to each instance's enlarged predicted-box ROI. Matcher likewise selects reference-matched point prompts within this ROI and retains its candidate evaluation and mask fusion procedure. For SAC-Prompt, we train its prompt generator to predict a foreground probability map for each instance rather than for all foreground objects in the image, while keeping SAM frozen. These adjustments make each method operate at the instance level while retaining its original prompting procedure.

D.2 Qualitative Comparison

Figure 5 presents each method's point prompts alongside the SAM masks they induce. SAC-Prompt selects positive and negative points from regions with high foreground and background probabilities, respectively, but these points do not necessarily form an effective prompt configuration. In the first



Figure 5: Qualitative comparison with existing automatic prompting methods. (a) Ground truth; (b) SAC-Prompt; (c) Matcher; (d) AlignSAM; (e) PPD; (f) PromptPilot; (g) RePP (Multi-point). Circles and crosses denote positive and negative point prompts, respectively; boxed values report per-instance mask IoU.

row, its points cover both the horse and rider, yet the rider mask in column (b) reaches only 0.245 IoU. Matcher finds points through reference-feature correspondence, then scores and fuses candidate masks. However, errors in the retained masks can accumulate during fusion, introducing background regions into the output, as illustrated in rows 1, 2, and 4 of column (c). AlignSAM learns successive edits that favor points in the target foreground; satisfying this spatial constraint, however, does not ensure that SAM covers the full instance. In the third row, the points in column (d) lie within the person, while the induced mask covers only part of the target. PPD and PromptPilot both seek useful subsets of dense feature-matched point candidates, though they construct those subsets differently. Despite placing candidate points near the rider, both PPD and PromptPilot nearly miss it in columns (e) and (f) of the first row. In contrast, RePP uses the masks induced by candidate updates to decide which prompt to retain. Across the crowded natural scenes and the smaller objects in the remote-sensing scenes, column (g) shows more complete and consistent instance masks.

These comparisons reveal a common difficulty across different prompting strategies: cues that make a point configuration plausible do not reliably predict the quality of its SAM response. RePP separates these two decisions, allowing candidates to be proposed using the current prompt state and then judged by the masks they actually induce. This response-based determination helps retain updates that better delineate the target, consistent with the gains of 10.7, 6.7, and 4.2 AP on VOC, COCO, and NWPU over the strongest compared automatic prompting methods in Table 4.

D.3 Additional RePP Refinement Examples

Figure 6 follows RePP through three refinement steps on examples from VOC, COCO, and NWPU. Within each step, the three candidate columns start from the same prompt state but produce visibly different masks. This variation is especially apparent in the crowded COCO scene, where an update that improves one person can leave a neighboring instance less accurately segmented. The retained column shows how preference is applied to each instance's realized responses before its prompt is

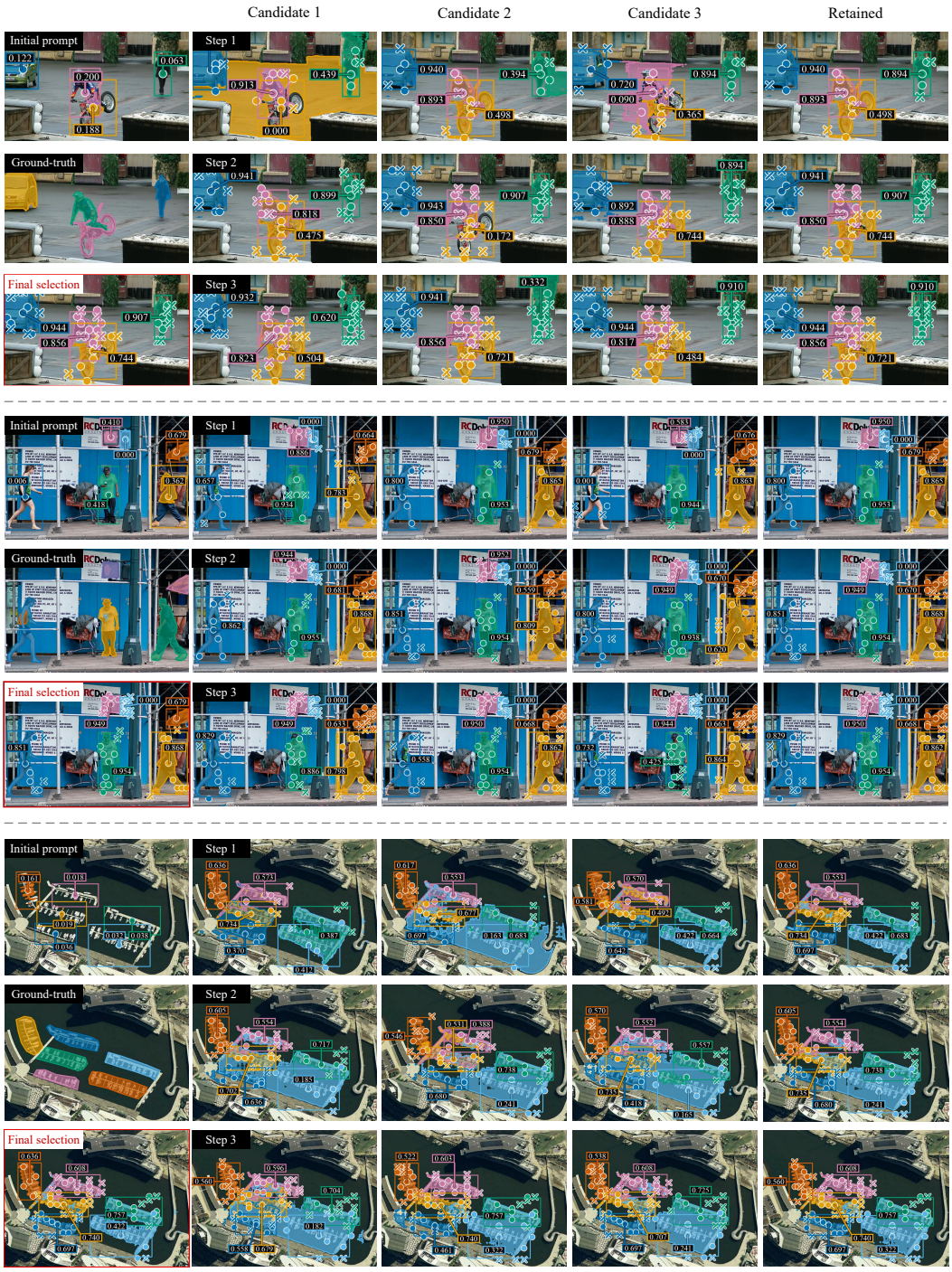

Figure 6: Additional RePP refinement examples on VOC, COCO, and NWPU. For each example, the left column shows the initial prompt, ground truth, and final output. The middle columns show three candidate SAM responses at each step, and the right column shows the retained response. Circles and crosses denote positive and negative point prompts, respectively; boxed values report per-instance mask IoU.

carried forward. In the NWPU example, candidate edits also change the extent and boundaries of nearby objects, showing that this decision remains consequential when targets are small and closely spaced.

The final-output panels illustrate a second decision after refinement: RePP determines the output from the prompt–response states visited along the trajectory, rather than automatically using its last state. This allows a response found earlier to be retained when later edits do not improve that instance. The examples therefore show the distinct roles of preference within each step and across the trajectory. Consistent with this interpretation, Table 3 shows that using preference at both stages improves VOC AP from 43.6 to 58.1 over the setting without either preference decision.