Thank you very much for your recognition of our research direction and methodology, as well as your positive feedback on our writing and expression. Below we provide a detailed explanation of your comments and questions. If there are any aspects that have not been clarified or if you have new questions, please feel free to let us know, and we will promptly address them.

> **[Q1, Q3, Q4]** What does M represent? ... How to construct a prompt? ... Why construct a learnable word embedding vector to replace v_m? What is the role of the learnable vector?

I apologize for the lack of clarity in this matter. The Eq. (13-14) is the prompt construction method in the traditional CLIP-based TSU method. Here, we design a prompt with M words in Eq. (13), where M is the number of prompt words. Next, we use a word embedding model to convert the M words as M word embedding vectors (see Eq.(14)). In this method, the prompt is designed through a hand-crafted method. For different prompts, the length M can be different. For example, if the prompt is "a photo of a [CLASS] road", then M is 5. If the prompt is "a photo of a road whose surface is [CLASS]", then M is 8. It can be observed that these prompts are fixed and cannot be learned.

In this hand-crafted prompt method, the performance of the model depends heavily on the experience of the person designing the prompt [1]. To overcome this shortage, our model **did not use** the prompt given in Eq. (13-14). The prompt construction in our model is given in Eq. (15-16). Here, given an image taken at the trajectory sample e_i, for its p-th aspect scene understanding, we assume we have a prompt with M words. For its m-th world, its embedding vector is calculated as

$$\tilde{\boldsymbol{v}}_{m,e_i}^{(p)} = \boldsymbol{w}_m^{(p)} + \boldsymbol{r}_{e_i},$$

where $\boldsymbol{w}_m^{(p)}$ is a learnable parameter. Using these learnable word embedding vectors, we construct a prompt embedding matrix as

$$\tilde{\boldsymbol{\mathcal{V}}}_{e_i}^{(p)} = \left(\tilde{\boldsymbol{v}}_{1,e_i}^{(p)}, \ldots, \tilde{\boldsymbol{v}}_{m,e_i}^{(p)}, \ldots, \tilde{\boldsymbol{v}}_{M,e_i}^{(p)}\right).$$

Because this prompt embedding matrix is constructed by our model, we can directly set its length M. Therefore, **the prompt length M in our model is same for different aspects.**

In Eq. (15), we construct the prompt embedding using a learnable parameter $\boldsymbol{w}_m^{(p)}$ and the ST context representation vector $\boldsymbol{r}_{e_i}$. The learnable parameter ensures the prompt can be trained to adapt to the scene understanding tasks, which avoid the complexity and instability of hand-crafted prompt engineering. Meanwhile, the ST context representation vector incorporate spatio-temporal information into the prompt.

> **[Q2]** what does Out Degree(OD) represent?

The Out Degree (OD) property of a given road segment A refers to the number of road segments connected to A in the out direction. We will add this explanation in the revision.

> **[Q5]** Whether it will take a lot of memory and time for attention mechanism. It would be better to analyze the time complexity.

- In terms of **memory usage** for ST-CLIP, since the encoders of CLIP are frozen, the number of updatable parameters is limited. Therefore, it is sufficient to run on a NVIDIA Tesla P40 GPU with 8GB of VRAM, as described in **Appendix B**.
- In terms of **time complexity** for ST-CLIP, the introduction to the computational complexity of the model is provided in **Appendix C**, following Algorithm 1. Here, we briefly summarize it.

  Since the image and text encoders of ST-CLIP are kept frozen, the primary sources of the model's time complexity lie in two aspects: the **spatio-temporal context representation learning** and **ST-aware multi-aspect prompts learning**. The time complexity of the former one can be approximated as $\mathcal{O}(D^2)$, where D denotes the dimension of the ST-context representation vector. The time complexity of the latter one can be roughly given as $\mathcal{O}((M \times P)^2 D)$, where M represents the length of learnable prompt and P is number of aspects. In conclusion, the overall time complexity of ST-CLIP is $\mathcal{O}(D^2 + (M \times P)^2 D)$.

  To validate the efficiency of our model, we also supplemented relevant experiments on both training and inference speed. We conducted experiments on the Beijing dataset with 16-shot setting to test the training and inference time of ST-CLIP and other baseline models using two visual backbone networks: ResNet-50 and ViT-B/32. For specific experimental details, please refer to **Appendix B**. We compared using the official code of baseline models and the results are shown in the following table:

Model	Training Time (ResNet-50)	Inference Speed (ResNet-50)	Training Time (ViT-B/32)	Inference Speed (ViT-B/32)
CLIP$_{\mathrm{ZS}}$	\	387.8 item/s	\	387.8 item/s
CoOp	10min16s	19.2 item/s	10min40s	19.0 item/s
CoCoOp	12min38s	14.4 item/s	13min11s	14.2 item/s
CLIP-Adapter	6min21s	19.0 item/s	7min14s	19.4 item/s
Tip-Adapter	6min24s	375.4 item/s	6min40s	375.1 item/s
Tip-Adapter-F	21min46s	373.2 item/s	22min4s	372.5 item/s
ST-CLIP	11min5s	35.3 item/s	11min33s	34.4 item/s

  - The training time of ST-CLIP is moderate. Compared to the optimal baseline model Tip-Adapter-F, it significantly reduces training time while achieving better experimental results.
  - Regarding inference speed, ST-CLIP constructs unique text features for each input image, requiring the use of the text encoder each time. In contrast, for Tip-Adapter-F, the text features for all images are the same and fixed, so they can be pre-computed and reused, resulting in faster inference speed. Compared to other baseline models, ST-CLIP demonstrates faster inference speed due to its ability to simultaneously address all aspects without the need for one-by-one processing.

References

[1] Zhou K, Yang J, Loy C C, et al. Learning to prompt for vision-language models[J]. International Journal of Computer Vision, 2022, 130(9): 2337-2348.