# Referring Image Segmentation Using Text Supervision

Fang Liu[1,2,*], Yuhao Liu[2,*], Yuqiu Kong[1,†], Ke Xu[2,†], Lihe Zhang[1] ,
Baocai Yin[1], Gerhard Hancke[2] , Rynson Lau[2,†]
[1] Dalian University of Technology, [2] City University of Hong Kong
{fawnliu2333, yuhaoLiu7456, kkangwing}@gmail.com, {yqkong, zhanglihe, ybc}@dlut.edu.cn,
{gp.hancke, rynson.lau}@cityu.edu.hk

In this Supplemental, we first provide more quantitative comparison and ablation experiments in Sec. 1 and Sec. 2, respectively. We then show more visual results and further comparisons in Sec. 3. The splits of the four benchmark datasets are detailed in Sec. 4.

## 1. Addition Quantitative Comparison

**Performance evaluation of different methods.** Table 1 provides a comprehensive performance evaluation of our proposed Step-1 and Step-2, as well as several methods adapted from other tasks, on four benchmark datasets. In particular, GroupViT† [13] is fine-tuned on the training sets and subsequently validated on the test sets. Regarding Maskclip†[14] and CLIP-ES†[7], we utilize their output as supervision signals to supervise the output of our network in Step-1 via a cross-entropy loss. Other compared methods [9, 2, 11] employ the similar training procedure from their respective repositories.

Our proposed method in Step-1 outperforms the aforementioned methods by large margins, as demonstrated by the significant performance improvements on PointIt metric. Additionally, the segmentation performance is further improved after the second training step, where the pseudo-labels generated by Step-1 are utilized as supervision.

**Comparison of the computational cost.** We compare the parameters, MACs of different methods, and our framework (Steps 1 and 2) in Tab. 2. The computational cost of our proposed method is lower compared with that of AMR. The discrepancy in the computational cost can be primarily attributed to the additional compensation branch utilized by AMR, which is designed to identify and extract more relevant regions. Besides, the MACs of GbS and WWbL are much higher than ours. This is mainly because they employ a heavy segmentation decoder, but ours (Step-1) can directly up-sample and conduct thresholding on the response maps to obtain the initial results. In addition, although the parameters of their models are smaller, our performance is significantly higher than theirs. We also show the inference speeds of different methods. Compared with Step-1 of our framework, Step-2 introduces the segmentation decoder and non-local attention block [12], resulting in the increase of inference time. Although the speed of WWbL is slightly faster than our Step-1 (15ms *vs* 17ms), its performance is much lower than ours.

## 2. Addition Ablation Studies

**Different image encoders in Step-1.** In Tab. 3, we compare the performances of different image encoders with different sources of weights, including ResNet-50 [5] and Swin-B [8] pretrained on ImageNet [3], and ResNet-50/101 pretrained on CLIP [10]. The image encoder with CLIP weights can obtain better performances due to its zero-shot transfer ability. Unless otherwise specified, we adopt ResNet-50 with CLIP weights as our default image encoder in Step-1 so as to reduce the model complexity. Even when we employ the pre-trained weights from ImageNet to initialize our image encoder, our performances still surpass WWbL [11] by large margins (*i.e.*, PointIt: 61.83 *vs* 45.28; IoU: 24.86 *vs* 21.75) on the RefCOCOg (U) val set. These comparisons and results demonstrate that the effectiveness of the proposed framework comes from the unique problem formulation and key designs (*e.g.*, the text-to-image optimization process and the calibration method), rather than the stronger backbones or weights.

**Different values for hidden dimension $C_d$ in Step-1.** When we gradually increase $C_d$ by a factor of 2 starting from 128, the performance grows accordingly and peaks at 1024, as shown in Tab. 4. This shows that our framework does not simply rely on more learning parameters to improve its performance.

**The coefficient $\lambda$ of classification loss $\mathcal{L}_{cls}$ in Step-1.** The final loss that we use in Step-1 includes two items, *i.e.*, classification loss $\mathcal{L}_{cls}$ and calibration loss $\mathcal{L}_{cal}$. To avoid over-tuning these hyper-parameters, the coefficient for $\mathcal{L}_{cal}$ is set as 1, and we only adjust the coefficient for $\mathcal{L}_{cls}$, *i.e.*, $\lambda$. All

| Metric | Method | Backbone | ReferIt | RefCOCO | | | RefCOCO+ | | | RefCOCOg | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | test | val | testA | testB | val | testA | testB | val (G) | val (U) | test (U) |
| PointIt | AMR† [9] | ResNet-50 | 25.78 | 22.99 | 12.94 | 36.27 | 24.19 | 14.62 | 38.15 | 37.25 | 35.95 | 36.99 |
| | MaskCLIP† [14] | ResNet-50 | 28.89 | 20.54 | 24.66 | 18.67 | 28.54 | 32.94 | 22.62 | 23.60 | 22.55 | 22.72 |
| | GroupViT† [13] | GroupViT | 40.09 | 33.17 | 34.15 | 32.74 | 33.99 | 33.48 | 34.20 | 40.17 | 40.79 | 40.39 |
| | CLIP-ES† [7] | ViT-Base | 60.50 | 48.27 | 58.34 | 36.96 | 53.22 | 62.98 | 40.52 | 59.40 | 55.82 | 54.93 |
| | GbS† [2] | VGG16 | 48.12 | 35.31 | 33.77 | 36.82 | 33.87 | 33.00 | 36.75 | 36.63 | 38.57 | 39.16 |
| | WWbL† [11] | VGG16 | 57.40 | 38.43 | 38.77 | 37.45 | 44.09 | 43.33 | 44.36 | 42.29 | 45.28 | 43.14 |
| | Ours (Step-1) | ResNet-50 | 72.56 | 60.95 | 71.12 | 49.64 | 48.50 | 49.42 | 47.25 | 65.80 | 65.03 | 66.08 |
| | Ours (Step-2) | ResNet-50 | **74.94** | **60.86** | **70.39** | **49.72** | **60.63** | **68.72** | **50.93** | **67.61** | **68.55** | **68.40** |

Table 1. Quantitative comparison of different methods using text description labels on four RIS benchmarks. (G) and (U) denote the Google and UMD dataset partitions of RefCOCOg. † indicates the methods adapted from other tasks.

| Model | Params. | MACs | Speed | IoU | PointIt | PointM |
|---|---|---|---|---|---|---|
| AMR† [9] | 156.87Mb | 27.88G | 72ms | 18.98 | 25.78 | 7.12 |
| GbS† [2] | 48.41Mb | 34.99G | 55ms | 14.21 | 48.12 | 30.30 |
| WWbL† [11] | 81.30Mb | 67.20G | 15ms | 28.01 | 56.09 | 42.84 |
| Ours (Step-1) | 115.66Mb | 12.40G | 17ms | 33.33 | 72.56 | 61.70 |
| Ours (Step-2) | 142.09Mb | 21.26G | 22ms | 44.57 | 74.94 | 67.00 |

Table 2. Comparison of computation costs of the methods adapted from other tasks with our framework (Step-1 and Step-2) on the ReferIt *test* set. The MACs are tested on 320×320 resolution. The inference speeds are based on one GTX1080Ti with a batch size of 1. † indicates methods re-trained by us.

| Weights | Encoders | Train | | Val | |
|---|---|---|---|---|---|
| | | IoU | PointIt | IoU | PointIt |
| ImageNet [3] | ResNet-50 | 25.72 | 62.05 | 24.86 | 61.83 |
| | Swin-B | 24.04 | 65.21 | 24.32 | 64.44 |
| CLIP [10] | ResNet-50 | 27.81 | 66.98 | 26.62 | 65.07 |
| | ResNet-101 | 27.62 | 68.02 | 27.47 | 67.28 |

Table 3. Comparison of different image encoders on the RefCOCOg (U) *training* and *val* sets in Step-1. Swin-B denotes the *base* version of the Swin Transformer. The weights refer to the different sources of the pre-trained model parameters.

experiments are conducted with the same experimental settings as the final framework. As shown in Tab. 5, we test different values of $\lambda$ (from 1 to 10), and the best performance is achieved when $\lambda = 5$. Meanwhile, the variances of the IoU and PointIt performances in these experiments are small (only 0.1 and 0.7, respectively), which also indicates that the proposed framework is stable and insensitive to the loss ratio.

**Different image encoders in Step-2.** As shown in Tab. 6, using Swin-B as the image encoder yields better results than using ResNet-50, but at the costs of higher complexity and computations (*e.g.*, FLOPs: 38.22G *vs* 21.63G; speed [1]: 30ms *vs* 22ms). Even though we can achieve a better RIS performance by using a stronger visual encoder, we adopt ResNet-50 as our default visual encoder in Step-2 to balance the trade-off between the model's complexity and accuracy.

[1]The inference time for one pair of image and query.

| | 128 | 256 | 512 | 1024 | 2048 |
|---|---|---|---|---|---|
| IoU | 26.42 | 26.87 | 27.35 | **27.81** | 27.65 |
| PointIt | 63.40 | 64.77 | 65.24 | **66.98** | 66.60 |
| PointM | 49.55 | 51.31 | 52.13 | **53.69** | 53.45 |

Table 4. Comparison of using different values for hidden dimension $C_d$, on RefCOCOg (U) *train* set.

**The influence of refinement in Step-2.** In Tab. 7, we also present the comparison of IoU performances with and without the refinement operation [1] on the training sets of four datasets, as we use it to generate better pseudo-labels for the training process of Step-2. We can clearly see that the refinement operation (see 2rd-row in Tab. 7) can indeed enhance the quality of the responses. Besides, the higher the quality of the response maps (*i.e.*, the results of not having this refinement as shown in row-1), the higher the performance gains of the refinement (*i.e.*, row-2).

## 3. More Visualization Results

**Qualitative Results**. More qualitative results of our framework are shown in Fig. 1. Our framework can properly localize and segment different kinds of sample targets, including long and complex sentences (*e.g.*, Images (a), (c) and (g)), appearance descriptions (*e.g.*, "green", "blue" and "red" in Images (b) and (j)), spatial positions (*e.g.*, "top", "left" and "center" in Images (n), (u) and (v)) and mutual relations (*e.g.*, "sitting on" Image (h)).

**Comparison with WSGs**. WWbL usually misidentifies other regions as target regions, and it is difficult to reduce the influence of noisy regions. As shown in Table 1, we provide more comparisons of IoU and PointIt performances with GbS and WWbL on different benchmark datasets. In

| $\lambda$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| IoU | 26.92 | 27.56 | 27.80 | 27.70 | **27.81** | 27.40 | 27.43 | 27.27 | 27.06 | 26.97 |
| PointIt | 63.93 | 65.89 | 66.29 | 66.72 | **66.98** | 66.18 | 66.84 | 66.83 | 66.56 | 66.53 |
| PointM | 51.74 | 53.18 | 53.55 | 53.58 | **53.69** | 52.87 | 53.17 | 53.06 | 52.76 | 52.52 |

Table 5. Effect of using different values for $\lambda$ in the classification loss ($\mathcal{L}_{cls}$). All experiments are performed on the RefCOCOg (U) *training* set using Step-1 of our framework. We set $\lambda$=5, by default.

| Encoders | Train | | | Val | | |
|---|---|---|---|---|---|---|
| | IoU | PointIt | PointM | IoU | PointIt | PointM |
| ResNet-50 | 36.25 | 66.82 | 57.59 | 36.19 | 68.53 | 58.84 |
| Swin-B | 37.01 | 67.80 | 59.41 | 36.80 | 67.53 | 58.70 |

Table 6. Comparison of different image encoders on the RefCOCOg (U) *training* and *val* sets in Step-2. Swin-B and ResNet-50 use the pre-trained weights from ImageNet [3] and CLIP [10], respectively.

| | ReferIt | RefCOCO | RefCOCO+ | RefCOCOg | |
|---|---|---|---|---|---|
| | | | | Google | UMD |
| PRMS | 33.30 | 26.63 | 26.05 | 27.17 | 27.92 |
| PRMS +*R* | 43.26 | 31.33 | 30.86 | 34.99 | 35.66 |

Table 7. Quantitative comparison without (*i.e.*, PRMS) and with (*i.e.*, PRMS + *R*) the refinement operation [1], performed using different *training* sets in Step-2 on the IoU metric. The refinement operation is built on PRMS, and the better the result of PRMS, the better the result after the refinement.

Fig. 2, we also show predicted results of WWbL [11] and our framework. Both the quantitative and qualitative results reveal that WWbL is less effectiveness nor applicable in handling the RIS task. For the example shown in the 1st row, the query is "*a man in grey*". It locates the woman (noises) as one of the target regions. In contrast, our framework optimizes a text-to-image response process, which can continuously adjust the locations and regions of the initial response map, instead of being fixed as in WWbL. Besides, our calibration method also utilizes negative samples (such as "*woman on the right*", "*a woman holding flowers*", and "*elephant in a fenced area behind a woman*") to suppress these noisy regions that are unrelated to the referring expression.

## 4. Dataset Details

**RefCOCO**. It is collected in an interactive game interface [6], and divided into training, val, testA, and testB sets, including 120,624, 10,834, 5,657, and 5,095 samples, respectively. Each image in this dataset has multiple objects of the same category. **RefCOCO+**. It is also split into training, val, testA, and testB datasets, which have 120,191, 10,758, 5,726, and 4,889 samples, respectively. **RefCOCOg**. It has two partitions: Google and UMD partitions. The former includes 85,474 training samples and 9,536 val samples. The latter contains 80,512 training samples, 4,896 val samples and 9,602 test samples. The average length of expressions in this dataset is 8.4 words. **ReferIt**. It is collected from IAPR TC-12 [4], and divided into training (59,976 samples) and test (60,105 samples) sets.

## References


[1] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. In *CVPR*, 2019. 2, 3

[2] Assaf Arbelle, Sivan Doveh, Amit Alfassy, Joseph Shtok, Guy Lev, Eli Schwartz, Hilde Kuehne, Hila Barak Levi, Prasanna Sattigeri, Rameswar Panda, et al. Detector-free weakly supervised grounding by separation. In *ICCV*, 2021. 1, 2

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2, 3

[4] Hugo Jair Escalante, Carlos A Hernández, Jesus A Gonzalez, Aurelio López-López, Manuel Montes, Eduardo F Morales, L Enrique Sucar, Luis Villasenor, and Michael Grubinger. The segmented and annotated iapr tc-12 benchmark. *CVIU*, 2010. 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[6] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 3

[7] Yuqi Lin, Minghao Chen, Wenxiao Wang, Boxi Wu, Ke Li, Binbin Lin, Haifeng Liu, and Xiaofei He. Clip is also an efficient segmenter: A text-driven approach for weakly supervised semantic segmentation. *arXiv preprint arXiv:2212.09506*, 2022. 1, 2

[8] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 1

[9] Jie Qin, Jie Wu, Xuefeng Xiao, Lujun Li, and Xingang Wang. Activation modulation and recalibration scheme for weakly supervised semantic segmentation. In *AAAI*, 2022. 1, 2

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry,

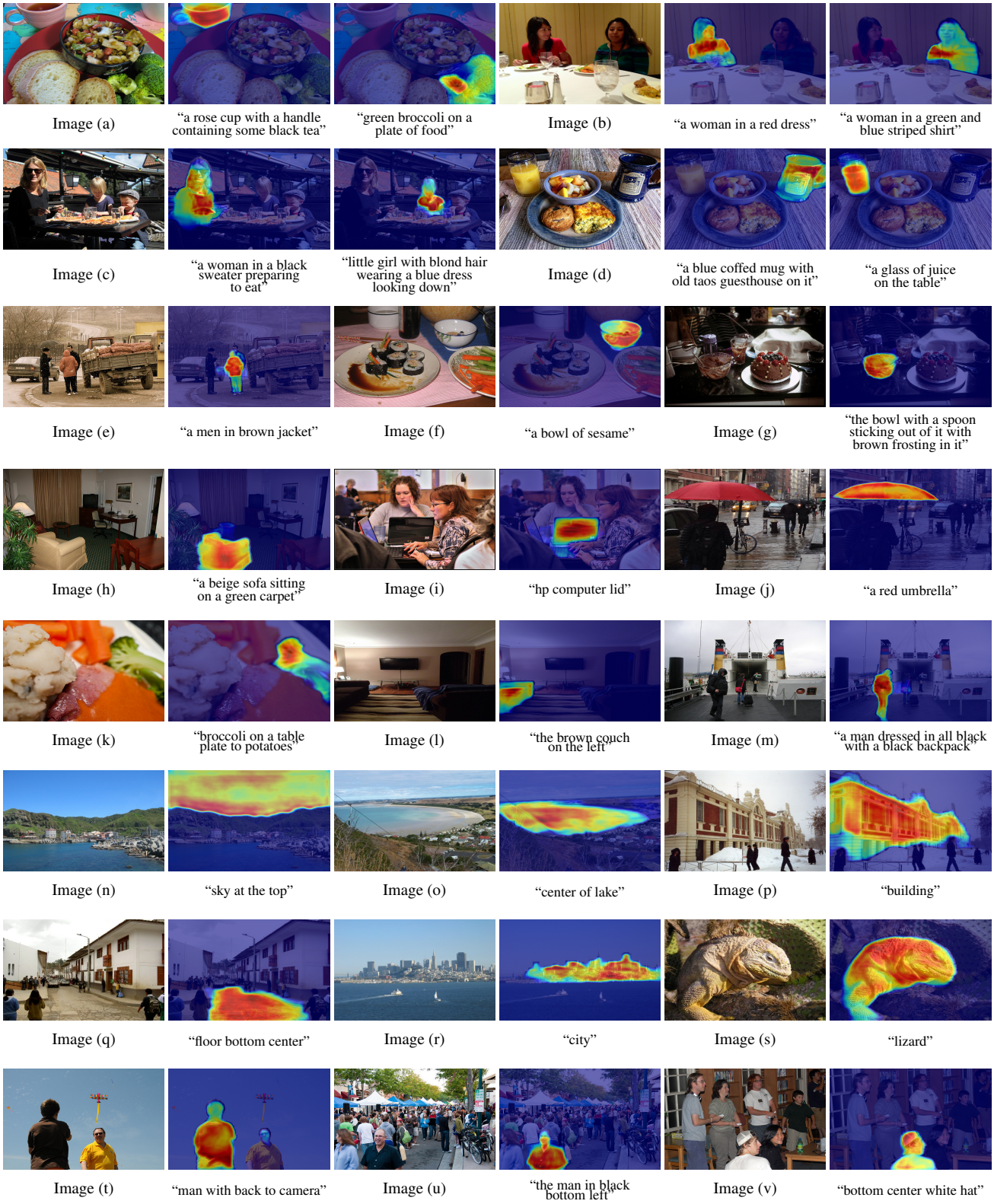


Figure 1. Qualitative results of referring image segmentation obtained by our framework.

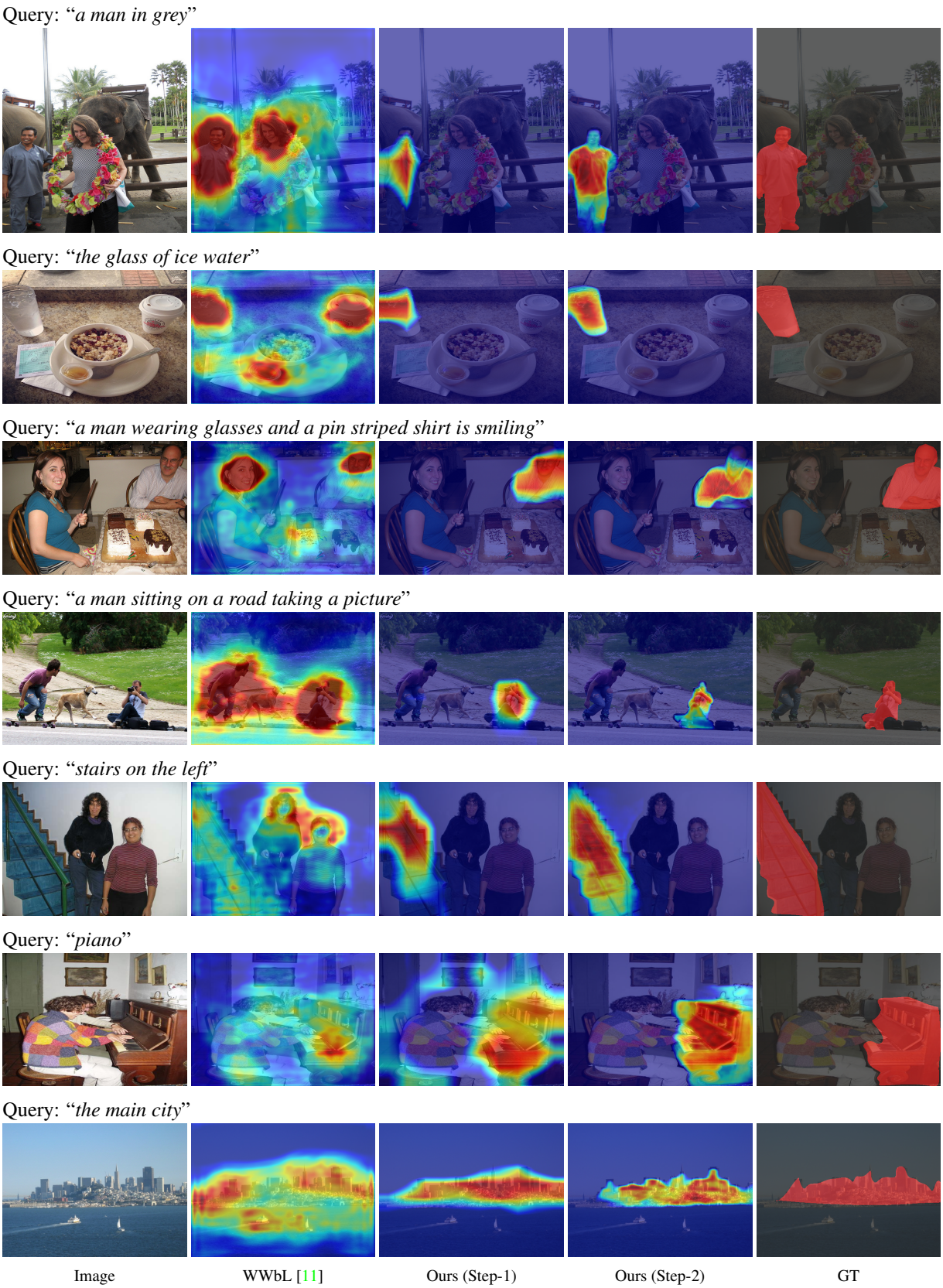


Figure 2. More visual comparison of WWbL and our framework (Step-1 and 2) for WRIS.

Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 2, 3

[11] Tal Shaharabany, Yoad Tewel, and Lior Wolf. What is where by looking: Weakly-supervised open-world phrase-grounding without text inputs. In *NeurIPS*, 2022. 1, 2, 3, 5

[12] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 1

[13] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *CVPR*, 2022. 1, 2

[14] Chong Zhou, Chen Change Loy, and Bo Dai. Extract free dense labels from clip. In *ECCV*, 2022. 1, 2