Multi-granularity Interaction Simulation for Unsupervised Interactive Segmentation

Appendix

A. Limitations

In this work, we discard the previous annotation-intensive object-oriented interaction simulation and propose an annotation-free alternation named **M**ulti-granularity **I**nteraction **S**imulation (**MIS**) to train an interactive segmentation model in an unsupervised manner. Although we explore the possibility and promise of unsupervised interactive segmentation and achieve inspiring results, the proposed method still has some limitations: **(1)** Due to the down-sampling, inaccuracy, and domain gap of the pre-trained feature extractor, the region proposals may sometimes contain noise or inconsistent semantics, which hurts the model training and further leads to a performance gap with supervised methods, especially on hard datasets (*e.g.*, SBD and DAVIS). **(2)** The trained segmentation model shares the same limitation as the supervised counterpart. That is, the model may fail in some challenging scenarios such as very thin and elongated objects.

B. Additional Ablation Study

Connectivity. In bottom-up merging (Algorithm 1), we introduce the connectivity constraint to restrict merges to only occur locally. Here we discuss the effectiveness of it by ablation. As shown in Table I, the performance of using or not using connectivity constraint is not much different, but using connectivity constraints greatly improves the speed (3.39 $\times$ improvement) and make the pre-processing more efficient than the previous DINO-based unsupervised segmentation methods (Appendix E). This is because we reduce the search space for the minimum cost from each pair of patches or regions to the adjacent patches or regions based on the prior on images.

Connectivity	Speed (image/s) ↑	GrabCut		Berkeley		SBD		DAVIS	
		NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓
✘	1.10	2.12	2.52	**3.08**	4.69	**6.88**	**9.46**	**5.63**	**7.56**
✓	**3.73** (↑ 3.39 ×)	**1.94** (↓ 0.18)	**2.32** (↓ 0.20)	3.09 (↑ 0.01)	**4.58** (↓ 0.11)	6.91 (↑ 0.03)	9.51 (↑ 1.36)	6.33 (↑ 0.70)	8.44 (↑ 0.88)

Table I. **Ablation on connectivity constraint.**

Feature Extractor. We compare the impact of different pretrained weights, ViT variants, and patch size on the performance of our method and show the results in Table II. Similar to previous findings [19, 11], we find that the model size has little effect on the quality of dense features. On the other hand, our method is robust to different pretrained weights such as MoCov3 [4] and DINOv2 [12].

Method	Model	GrabCut		Berkeley		SBD		DAVIS	
		NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓
DINO [3]	ViT-S/8	1.94	**2.32**	3.09	**4.58**	6.91	9.51	6.33	8.44
	ViT-S/16	2.00	2.52	3.17	5.24	7.02	9.71	6.51	9.10
	ViT-B/8	2.14	2.58	3.52	5.10	7.05	9.70	6.52	8.74
	ViT-B/16	2.02	2.34	**3.01**	4.81	6.88	9.55	**6.27**	8.63
MoCov3 [4]	ViT-S/16	**1.88**	2.34	**3.01**	4.71	6.96	9.60	6.09	**8.26**
DINOv2 [12]	ViT-S/14	2.00	2.80	3.61	5.64	**6.73**	**9.39**	6.76	10.26

Table II. **Ablation on pre-trained feature extractor.**

C. Cross-domain Evaluation

In this section, we verify the cross-domain generalization of our unsupervised method by transferring the model trained on SBD [7] to two segmentation datasets in medical domain (BraTS [2] and OAIZIB [1]). The quantitative results are shown in Figure I and Table III by the IoU (Intersection over Union)-NoC (Number of Clicks) curve, NoC at different IoU thresholds (NoC@80, NoC@85), and IoU at different number of clicks (IoU@5, IoU@10).

Our MIS achieves comparable and in some cases better performance than its supervised counterpart (SimpleClick) without manual annotation. Unlike supervised methods which overfit the instance-level annotations, our model learns interactions at multiple granularities and is better generalizable to different domains. Therefore it can achieve significantly higher IoU in the first few clicks on datasets in medical domain.

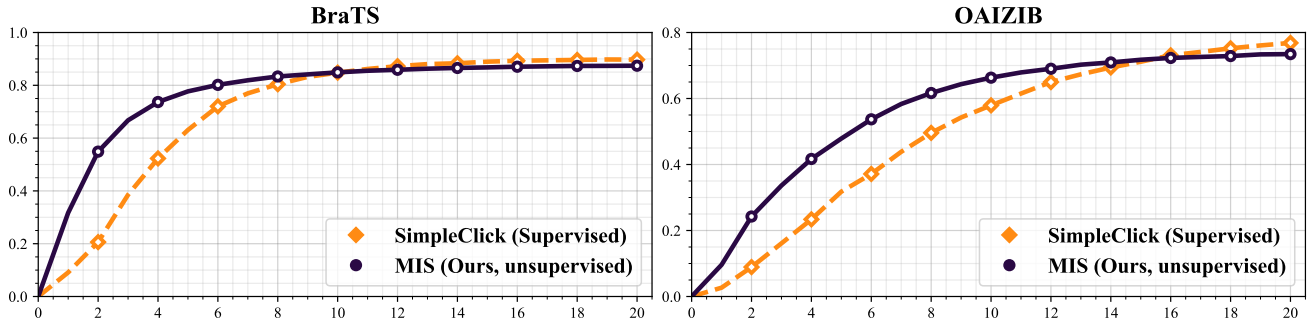

Figure I. **Comparison of IoU-NoC curve on BraTS and OAIZIB dataset.** Our unsupervised method generalizes better compared to the supervised counterpart, which is reflected in relying on fewer clicks to achieve a higher IoU.

Method	Backbone	BraTS				OAIZIB			
		NoC@80 ↓	NoC@85 ↓	IoU@5 ↑	IoU@10 ↑	NoC@80 ↓	NoC@85 ↓	IoU@5 ↑	IoU@10 ↑
SimpleClick [9]	ViT-B	7.75	9.63	63.08	84.79	**15.82**	**18.29**	31.78	57.92
MIS (*Ours*)	ViT-B	**6.55**	**9.11**	**77.70**	**84.90**	15.90	18.48	**47.87**	**66.31**

Table III. **Quantitative results on BraTS and OAIZIB dataset.**

D. Further Description of Ward's Method

In MIS, we use Ward's method [20] to implement the bottom-up merging process. As described in Section 3.1, Ward's method measures the cost of merging by the increment in the total within-cluster sum of squared error (SSE). During the merging process, the center $\boldsymbol{\mu}$ and the size s of each region are recorded. Initially, the size is set to 1 for each patch and the center is its feature. The detailed calculation after each merging is as follows:

(1) Computing the center of the newly generated region

$$\boldsymbol{\mu}_{A\cup B} = \frac{s_A\boldsymbol{\mu}_A + s_B\boldsymbol{\mu}_B}{s_A + s_B},$$

where s_A and s_B is the number of patches within region A and B.

(2) Computing the size of the newly generated region

$$s_{A\cup B} = s_A + s_B.$$

(3) Computing the cost of merging newly generated regions with other regions. Since the other regions are not changed, the

cost between them remains the same. The cost of merging is computed as

$$
\begin{aligned}
\mathrm{Cost}(A,B) &= e_{A\cup B} - (e_A + e_B) \\
&= \sum_{i\in A\cup B} ||\boldsymbol{F}_i - \boldsymbol{\mu}_{A\cup B}||^2 - \sum_{i\in A} ||\boldsymbol{F}_i - \boldsymbol{\mu}_A||^2 - \sum_{i\in B} ||\boldsymbol{F}_i - \boldsymbol{\mu}_B||^2 \\
&= \sum_{i\in A}(||\boldsymbol{F}_i - \boldsymbol{\mu}_{A\cup B}||^2 - ||\boldsymbol{F}_i - \boldsymbol{\mu}_A||^2) + \sum_{i\in B}(||\boldsymbol{F}_i - \boldsymbol{\mu}_{A\cup B}||^2 - ||\boldsymbol{F}_i - \boldsymbol{\mu}_B||^2) \\
&= \sum_{i\in A}\sum_j (2\cdot \boldsymbol{F}_i^j - \boldsymbol{\mu}_{A\cup B}^j - \boldsymbol{\mu}_A^j)(\boldsymbol{\mu}_A^j - \boldsymbol{\mu}_{A\cup B}^j) + \sum_{i\in B}\sum_j (2\cdot \boldsymbol{F}_i^j - \boldsymbol{\mu}_{A\cup B}^j - \boldsymbol{\mu}_B^j)(\boldsymbol{\mu}_B^j - \boldsymbol{\mu}_{A\cup B}^j) \\
&= \sum_j \sum_{i\in A} (2\cdot \boldsymbol{F}_i^j - \boldsymbol{\mu}_{A\cup B}^j - \boldsymbol{\mu}_A^j)(\boldsymbol{\mu}_A^j - \boldsymbol{\mu}_{A\cup B}^j) + \sum_j \sum_{i\in B} (2\cdot \boldsymbol{F}_i^j - \boldsymbol{\mu}_{A\cup B}^j - \boldsymbol{\mu}_B^j)(\boldsymbol{\mu}_B^j - \boldsymbol{\mu}_{A\cup B}^j) \\
&\text{(Notice that } \boldsymbol{\mu}_A^j = \frac{1}{s_A}\sum_{i\in A}\boldsymbol{F}_i^j \text{ and } \boldsymbol{\mu}_B^j = \frac{1}{s_B}\sum_{i\in B}\boldsymbol{F}_i^j\text{)} \\
&= \sum_j (2\cdot s_A\boldsymbol{\mu}_A^j - s_A\boldsymbol{\mu}_{A\cup B}^j - s_A\boldsymbol{\mu}_A^j)(\boldsymbol{\mu}_A^j - \boldsymbol{\mu}_{A\cup B}^j) + \sum_j (2\cdot s_B\boldsymbol{\mu}_B^j - s_B\boldsymbol{\mu}_{A\cup B}^j - s_B\boldsymbol{\mu}_B^j)(\boldsymbol{\mu}_B^j - \boldsymbol{\mu}_{A\cup B}^j) \\
&= \sum_j (s_A(\boldsymbol{\mu}_A^j - \boldsymbol{\mu}_{A\cup B}^j)^2 + s_B(\boldsymbol{\mu}_B^j - \boldsymbol{\mu}_{A\cup B}^j)^2) \;\; (\text{Notice that } \boldsymbol{\mu}_{A\cup B} = \frac{s_A\boldsymbol{\mu}_A + s_B\boldsymbol{\mu}_B}{s_A + s_B}) \\
&= \sum_j \frac{s_A s_B^2 + s_A^2 s_B}{(s_A + s_B)^2}(\boldsymbol{\mu}_A^j - \boldsymbol{\mu}_B^j)^2 \\
&= \frac{s_A\cdot s_B}{s_A + s_B}||\boldsymbol{\mu}_A - \boldsymbol{\mu}_B||^2.
\end{aligned}
$$

E. Speed Comparison

Besides accuracy and diversity, the efficiency of the pre-processing in our method is also worth mentioning. We compare the speed of generating region proposals for some other DINO-based methods and our method (*i.e.*, bottom-up merging) in Table IV. The speed is represented by the number of images processed per second and does not include the time of pre-processing and feature extraction. 8 × speed and 16 × speed denote processing with 8 × and 16 × downsampled features, respectively. All results are measured using the same 100 images on the same machine with Intel Core i5-11600K CPU. Compared to previous methods, our method is significantly faster. In addition, the computational overhead of top-down sampling is negligible (about 10^{-3} seconds per image) and can be omitted by parallel processing with the data loader.

Method	TokenCut [19] $_{\text{CVPR22}}$	DSM [11] $_{\text{CVPR22}}$	MIS (*Ours*)
8× Speed (image/s) ↑	0.31	1.73	**7.37**
16× Speed (image/s) ↑	7.29	3.53	**36.95**

Table IV. **Speed comparison.**

In addition, since there are few cost items that need to be updated after each merge, the computational bottleneck of our method does not lie in the matrix operation which will occupy more CPU cores, so the processing speed can be further improved by processing different images in parallel. Furthermore, the the merging algorithm is executed in CPU and the feature extraction is executed in GPU, thus the speed of pre-processing can be further optimized through multi-processing. With the above optimizations, our pre-processing takes only about 2 minutes for 8498 images on a server with 4 NVIDIA RTX 3090 GPUs and an AMD EPYC 7502 32-Core Processor, demonstrating the efficiency and potential for application to large-scale data.

F. Additional Implementation Details

F.1. Patch Feature Extraction

To ensure the fully unsupervised setting, we employ a self-supervisedly pre-trained model as the feature extractor. Specifically, a Vision Transformer (ViT) [5] trained with DINO [3] is adopted since the property of representing patch-level semantics [3, 6, 15]. In order to prevent the extracted features from ignoring small objects or parts, we use ViT-Small [15] with

a patch size of 8. Before feeding an image into the ViT, we resize the image to make its height and width divisible by the patch size (*i.e.*, the target height is $(h + p - h \mod p)$ when the original height h is not divisible by the patch size p), and the position embedding of the ViT is interpolated to fit the image size using bilinear interpolation. We finally use all output tokens of the last block except the *cls* token as the patch features.

F.2. Hardware

The model is trained with two NVIDIA RTX 3090 GPUs using a batch size of 32.

F.3. Baselines

We use the official code to implement TokenCut [19][1], FreeMask [17][2], and DSM [11][3]. The hyperparameters are set as suggested in the original papers. The detailed settings are as follows.

TokenCut. We follow the salient detection setting and take the salient masks as region proposals to train the interactive segmentation model. Specifically, we set the threshold τ for constructing the graph to 0.2 and set the weight ϵ for the negative edge to 10^{-5}. The *key* features of its last layer are used as the input features.

FreeMask. We follow the Free Mask setting and take the object masks as region proposals to train the interactive segmentation model. The shorter side size of the input image is set to 800. DenseCL [18] self-supervised pre-trained ResNet-101 [8] is employed as the feature extractor. The features of the last stage are used as the key, which is then interpolated with scale factors of (1.0, 0.5, 0.25) to form the query. The dot product of query and key is used as the soft mask, and the threshold for binarizing the mask is set to 0.5. MatrixNMS [16] is employed to filter a large number of overlapping masks. After mask NMS, we further filter out low-quality masks with maskness score less than 0.7.

DSM. We follow the multi-region segmentation setting and take the non-background regions as region proposals to train the interactive segmentation model. When constructing the graph, the weight of each edge is calculated by the weighted sum of feature similarity and color similarity, where the weight for feature similarity and color similarity is 1.0 and 10.0, respectively. The color similarity is only added in k-NN nodes. We evaluate the results of extracting 5, 10, and 15 regions per image and find that the result with 5 regions per image is best. ViT-S/16 is employed as the feature extractor and the *key* features of its last layer are used as the input features. The supplementary experimental result of ViT-S/8 is shown in Table V.

Method	Model	GrabCut		Berkeley		SBD		DAVIS	
		NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓	NoC@85 ↓	NoC@90 ↓
DSM [11]	ViT-S/8	4.42	5.16	6.07	7.87	8.62	11.66	8.17	10.35
	ViT-S/16	3.64	4.64	5.49	7.75	8.59	11.57	7.08	10.11

Table V. **Supplementary experimental results for DSM [11].**

G. Additional Qualitative Results

Merging Process. Figure II shows some examples of the merging process, where the two highlighted regions of the white border are involved in the merge. Semantically meaningful regions at multiple granularities are gradually produced during the merge process.

Multi-granularity Region Proposal. Figure III shows some example of regions discovered by our method in multiple granularities. The results are obtained by select the regions represented by the shallowest n nodes in the merging tree. The proposals generated by our MIS contains diverse possible segments including common objects, parts of objects, and combination of objects. The patches within each region are semantic consistent in some granularity, thus providing meaningful interactions for training the model.

Interactive Image Segmentation. We show some examples of interaction and segmentation of our model on GradCut [14], Berkeley [10], SBD [7], and DAVIS [13] in Figure IV and Figure V. It can be found that the trained model can respond correctly to user interactions. For some simple objects, our model can produce satisfactory predictions within the first few clicks. It can also handle complex scenes such as partially occluded objects and adjacent instances.

[1] https://github.com/YangtaoWANG95/TokenCut
[2] https://github.com/NVlabs/FreeSOLO
[3] https://github.com/lukemelas/deep-spectral-segmentation

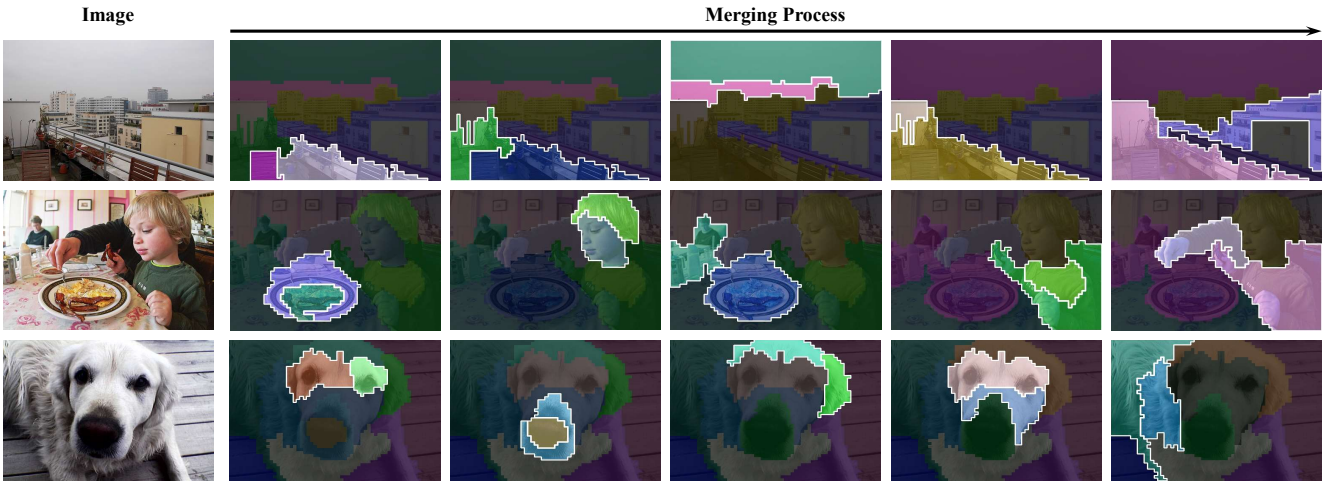

Figure II. **Visualization of the merging process.**

Figure III. **Visualization of the multi-granularity region proposal.**

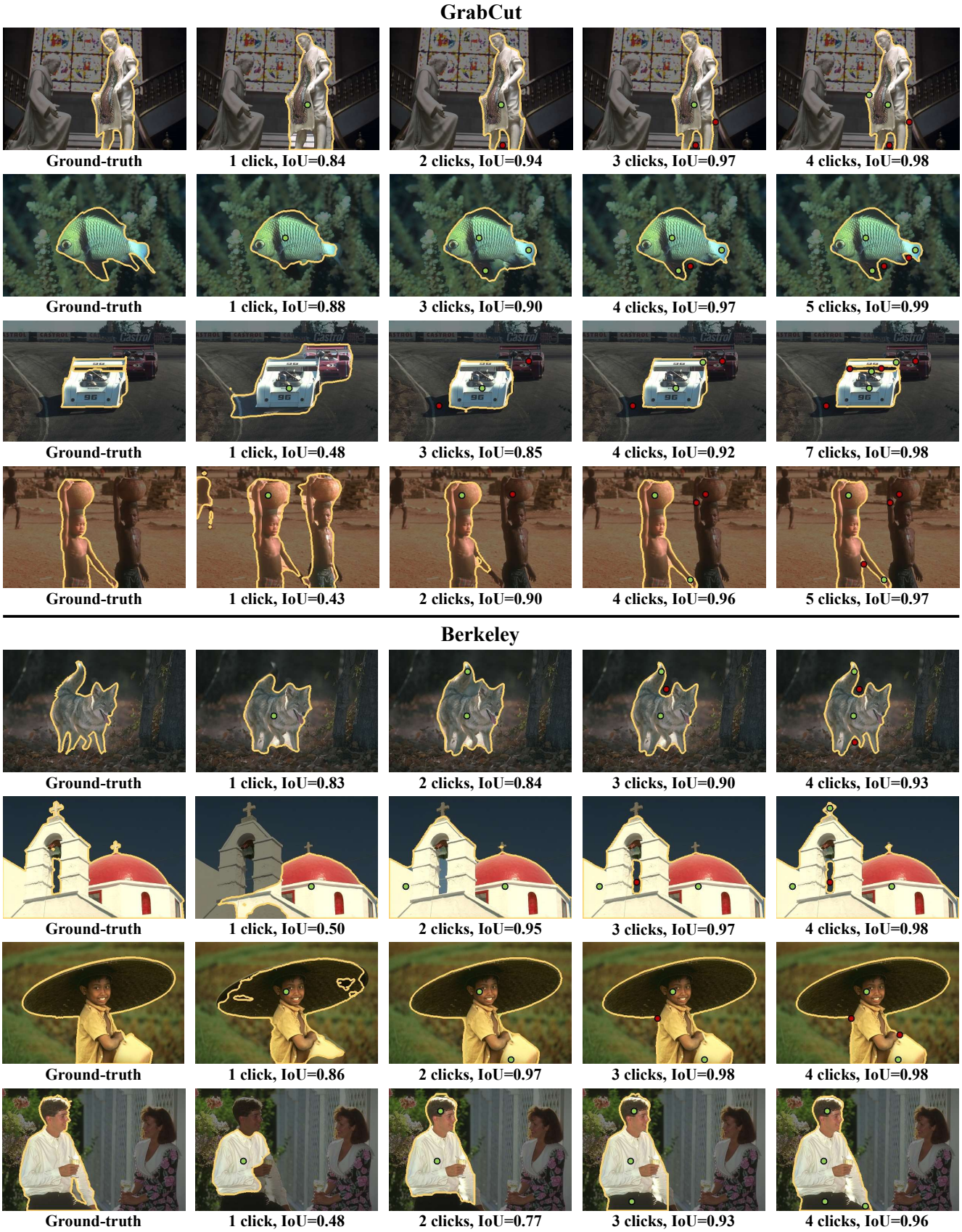

Figure IV. **Visualization of the interaction and segmentation.** The green and red points indicate positive and negative clicks respectively.

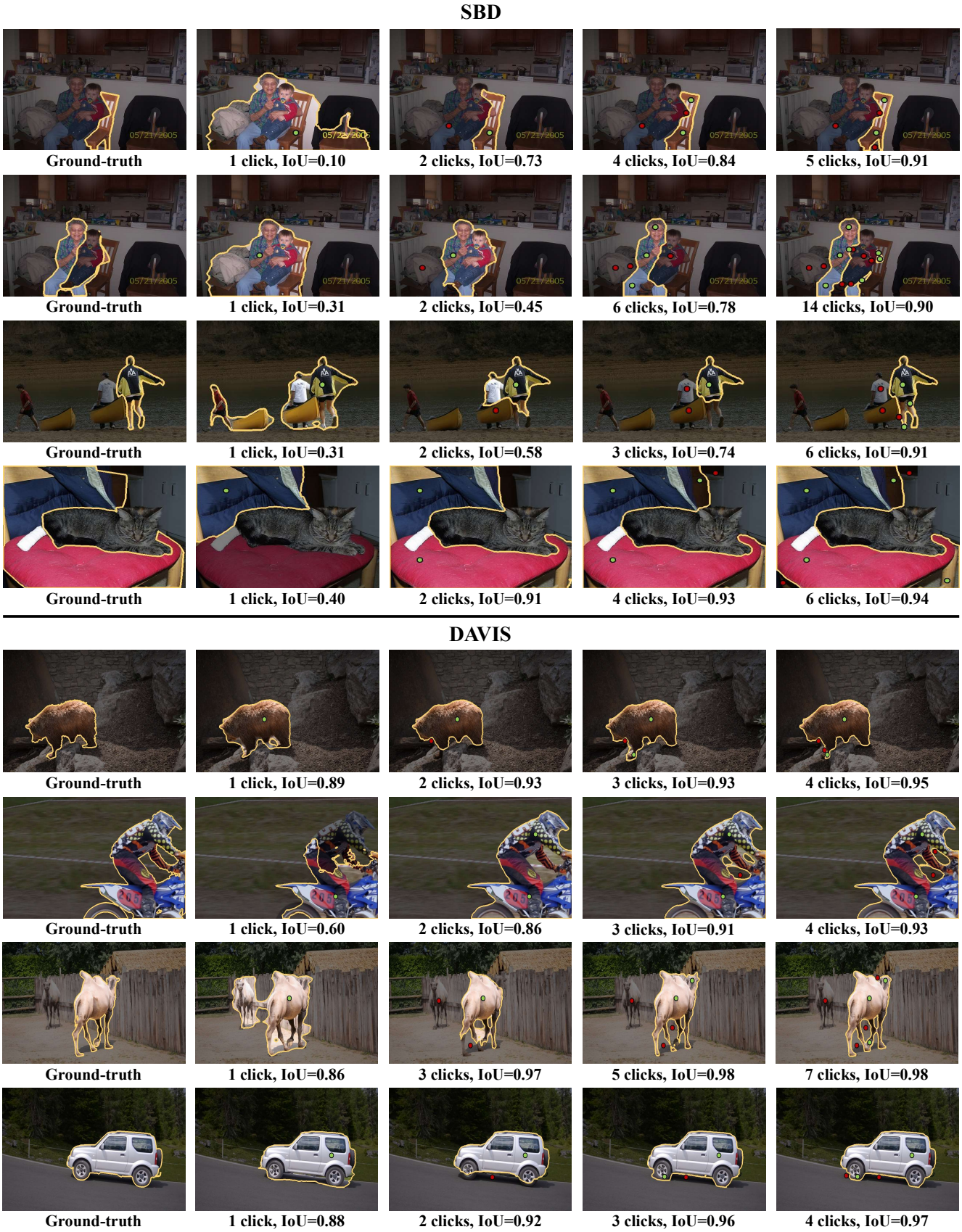

Figure V. **Visualization of the interaction and segmentation.** The green and red points indicate positive and negative clicks respectively.

References

[1] Felix Ambellan, Alexander Tack, Moritz Ehlke, and Stefan Zachow. Automated segmentation of knee bone and cartilage combining statistical shape knowledge and convolutional neural networks: Data from the osteoarthritis initiative. *Medical Image Analysis*, 52:109–118, 2019. 2

[2] Ujjwal Baid, Satyam Ghodasara, Suyash Mohan, Michel Bilello, Evan Calabrese, Errol Colak, Keyvan Farahani, Jayashree Kalpathy-Cramer, Felipe C Kitamura, Sarthak Pati, et al. The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification. *arXiv preprint arXiv:2107.02314*, 2021. 2

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021. 1, 3

[4] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021. 1

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 3

[6] Mark Hamilton, Zhoutong Zhang, Bharath Hariharan, Noah Snavely, and William T Freeman. Unsupervised semantic segmentation by distilling feature correspondences. In *International Conference on Learning Representations*, 2022. 3

[7] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *2011 International Conference on Computer Vision*, pages 991–998, 2011. 2, 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 4

[9] Qin Liu, Zhenlin Xu, Gedas Bertasius, and Marc Niethammer. Simpleclick: Interactive image segmentation with simple vision transformers. *arXiv preprint arXiv:2210.11006*, 2022. 2

[10] Kevin McGuinness and Noel E O'connor. A comparative evaluation of interactive segmentation algorithms. *Pattern Recognition*, 43(2):434–444, 2010. 4

[11] Luke Melas-Kyriazi, Christian Rupprecht, Iro Laina, and Andrea Vedaldi. Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8364–8375, 2022. 1, 3, 4

[12] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 1

[13] Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 724–732, 2016. 4

[14] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004. 4

[15] Oriane Siméoni, Gilles Puy, Huy V Vo, Simon Roburin, Spyros Gidaris, Andrei Bursuc, Patrick Pérez, Renaud Marlet, and Jean Ponce. Localizing objects with self-supervised transformers and no labels. In *British Machine Vision Conference*, 2021. 3

[16] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In *European Conference on Computer Vision*, pages 649–665. Springer, 2020. 4

[17] Xinlong Wang, Zhiding Yu, Shalini De Mello, Jan Kautz, Anima Anandkumar, Chunhua Shen, and Jose M Alvarez. Freesolo: Learning to segment objects without annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14176–14186, 2022. 4

[18] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3024–3033, 2021. 4

[19] Yangtao Wang, Xi Shen, Shell Xu Hu, Yuan Yuan, James L Crowley, and Dominique Vaufreydaz. Self-supervised transformers for unsupervised object discovery using normalized cut. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14543–14553, 2022. 1, 3, 4

[20] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963. 2