CAPE: A CLIP-Aware Pointing Ensemble of Complementary Heatmap Cues for Embodied Reference Understanding

Supplementary Material

A. Heatmap Generation

The heatmap generation method is a key component of our approach. Prior work [38] shows that the referent often aligns with the head-to-fingertip line, which inspired our Gaussian Ray Heatmap generation. While this generally holds when a person points clearly with an extended arm, it is not always reliable. Alignment can be disrupted if the person looks elsewhere (e.g., at a robot, camera, or another person), due to camera perspective, or when a person is close to the object and points using wrist motion. In such cases, the wrist-to-fingertip line aligns more accurately with the referent. To handle this variability, we generate separate heatmaps for both cues.

- x^H_{phm}: Heatmap from head-to-fingertip.
- x^W_{phm}: Heatmap from wrist-to-fingertip.

A.1. Head-to-Fingertip Heatmap

To generate this heatmap, we take the head and fingertip as reference points and construct a ray starting at the head and extending toward the image boundary in the fingertip direction. In the YouRefIt dataset [13], we use the detailed annotations provided in [13, 38], which include eye, elbow, wrist, and fingertip coordinates. These keypoints allow us to reliably define the reference direction for heatmap generation.

In real-world settings where such annotations are unavailable, estimating these points is straightforward. For eye position, face detection followed by using the center of the detected face provides a strong approximation. Similarly, modern pose estimation models offer accurate wrist and fingertip predictions. For our experiments on the ISL Pointing and CAESAR datasets, we obtain necessary coordinates using OpenPose [9], which delivers robust and reliable keypoint detection.

A.2. Wrist-to-Fingertip Heatmap

We follow the same pipeline used for head-to-fingertip heatmap generation, but replace the eye coordinates with wrist coordinates and the fingertip coordinates remain the same.

A.3. How to Create Heatmap

We evaluate the Conic Attention Heatmap using 15° and 30° cone angles, as well as the Gaussian Ray Heatmap, which produces more localized activation. As shown in Tab. A.7, the Gaussian Ray Heatmap achieves the best performance. This suggests that broader attention regions may be less effective in practice, likely due to the high density of objects in typical scenes (see Fig. A.5). Therefore, larger heatmaps tend to cover multiple distractors, reducing the discriminative value of the spatial signal.

Heatmap style	IoU=0.25	IoU=0.5	IoU=0.75	CLIP ↑	C_D ↓
Gaussian Ray Heatmap ($\sigma = 25$)	72.9	62.2	35.1	0.2457	0.2490
Conic Attention Heatmap (15°)	70.0	60.0	33.9	0.2464	0.2632
Conic Attention Heatmap (30°)	71.5	59.4	30.8	0.2463	0.2764

Table A.7. Comparison of different heatmap generation approaches.

B. Ensemble Methods

In this section, we present the details of the ensemble methods we explored. Each of our models produces N predictions, with each prediction accompanied by a confidence score. Before performing ensembling, we sort these N predictions in descending order based on their confidence scores. We denote the sorted prediction list from model M_H as $P^N_H \in \mathbb{R}^{4\times N}$, and from model M_W as $P^N_W \in \mathbb{R}^{4\times N}$. Similarly, we denote the confidence score of model M_H as $C^N_H \in \mathbb{R}^{1\times N}$, and from model M_W as $C^N_W \in \mathbb{R}^{1\times N}$.

B.1. Confidence-Only

In the Confidence-Only approach, we focus solely on the top-1 prediction from each model. We take the bounding box with the highest confidence score from M_H and another from M_W. These two predictions are then compared, and the one with the higher confidence score is selected as the final prediction.

Algorithm B.1 Algorithm of Confidence-Only ensembling method.

Require: Top-1 predictions from models M_H and M_W: (b^H, c^H) and (b^W, c^W)
Ensure: Final bounding box prediction b^*
1: **if** $c^H \geq c^W$ **then**
2: $\quad b^* \leftarrow b^H$
3: **else**
4: $\quad b^* \leftarrow b^W$
5: **end if**
6: **return** b^*

B.2. CLIP-Only (Top-1)

In this method, similar to the Confidence-Only approach, we select the bounding box with the highest confidence

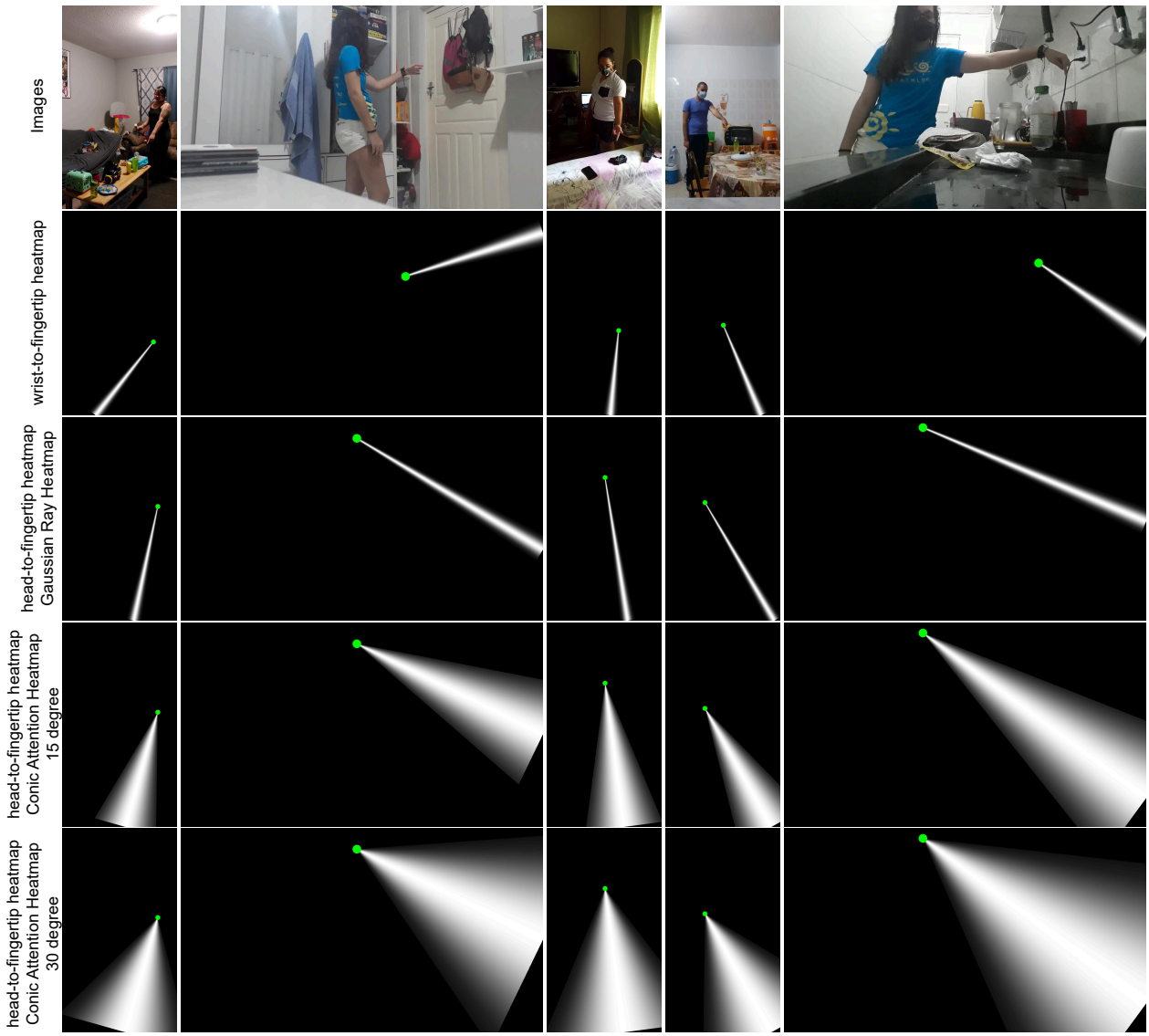

Figure A.5. This figure presents various heatmaps. The first row shows the input images. The second row displays wrist-to-fingertip heatmaps used for training M_W, while the third row contains head-to-fingertip heatmaps used for training M_H. The fourth and fifth rows show Conic Attention Heatmaps with 15° and 30° angles, respectively, which are used in the ablation study which we present in Tab. A.7.

score from each model. Next, we compute the CLIP similarity score between each predicted bounding box and the input text. The bounding box with the higher CLIP similarity score is then chosen as the final prediction.

$$CLIPSim_H = max(100 * cos(E_{I'_H, E_T}), 0) \quad (5)$$

where $E_{I'_H}$ is the embedded representation of cropped predicted bounding box of M_H with the highest score and E_T is the embedded representation of the text input.

$$CLIPSim_W = max(100 * cos(E_{I'_W, E_T}), 0) \quad (6)$$

where $E_{I'_W}$ is the embedded representation of cropped predicted bounding box of M_W with the highest score.

B.3. CLIP-Only (Top-2 + Threshold)

This method is similar to CLIP-Only (Top-1), where we take predictions from both models and compute the CLIP similarity score between the cropped bounding boxes and the input text. However, unlike the Top-1 variant, we also consider the Top-2 predictions from each model. However, we do not always include the second predictions; instead, we apply a confidence threshold determined empirically on

Algorithm B.2 Algorithm of CLIP-Only (Top-1) ensemble method.

Require: Top-1 predictions from models M_H and M_W: (b^H, c^H) and (b^W, c^W), input text t
Ensure: Final bounding box prediction b^*
Crop predicted regions from b^H and b^W to get image patches I'_H and I'_W
Compute text embedding E_T from t
Compute CLIP similarity scores using Eq. (5) and Eq. (6):
    $s_H = \max(100 \cdot \cos(E_{I'_H}, E_T), 0)$
    $s_W = \max(100 \cdot \cos(E_{I'_W}, E_T), 0)$
if $s_H \geq s_W$ **then**
    $b^* \leftarrow b^H$
else
    $b^* \leftarrow b^W$
end if
return b^*

the validation set, $T = 0.95$. If a model's second-highest prediction has a confidence score greater than or equal to this threshold, it is included in the comparison. Finally, we select the bounding box with the highest CLIP similarity score as the final prediction. The algorithm for this method is presented in Algorithm B.3.

B.4. CLIP Fusion

In this method, we leverage both the CLIP similarity score and the model's confidence score to make the final prediction. Specifically, we compute a hybrid score for each bounding box by summing its confidence score and its CLIP similarity score. We use the Top-2 predictions from both models for this process. However, the CLIP similarity scores and model confidence scores operate on different scales: while confidence scores are normalized in the range $[0, 1]$, CLIP similarity scores typically range around 25. To balance their influence and prevent the CLIP score from dominating, we scale the CLIP similarity scores by a factor of 0.04. The scaled CLIP score is then added to the corresponding confidence score to compute the final hybrid score for each bounding box. Finally, we select the bounding box with the highest hybrid score as the final prediction. The algorithm for this method is presented in Algorithm B.4.

B.5. CLIP-Aware Pointing Ensemble (CAPE)

In this method, we combine the CLIP-Only (Top-2 + Threshold) and CLIP Fusion approaches. Through empirical analysis on validation set, we observe that for small objects, the CLIP Fusion method yields more accurate results than other ensemble strategies. This is primarily because CLIP tends to perform less reliably on smaller objects. Therefore, we apply CLIP Fusion to not rely on only CLIP-score but also consider the confidence score. To determine whether an object is small, we follow the definition provided by [13]: if the object occupies less than 0.48% of the total image area, we classify it as small and use the CLIP Fusion ensemble. For all other cases, we apply the CLIP-Only (Top-2 + Threshold) method. The algorithm is presented in Algorithm B.5.

Algorithm B.3 Algorithm for CLIP-Only (Top-2 + Threshold).

Require: Predictions from models M_H and M_W, confidence threshold $T = 0.95$, input text t
Ensure: Final bounding box prediction b^*
Extract top-2 predictions from each model:
    $P_H = \{(b_1^H, c_1^H), (b_2^H, c_2^H)\}$
    $P_W = \{(b_1^W, c_1^W), (b_2^W, c_2^W)\}$
Initialize candidate set $\mathcal{B} = \{b_1^H, b_1^W\}$
if $c_2^H \geq T$ **then**
    Add b_2^H to $\mathcal{B}$
end if
if $c_2^W \geq T$ **then**
    Add b_2^W to $\mathcal{B}$
end if
for each $b \in \mathcal{B}$ **do**
    Compute CLIP similarity score $s_b = \text{CLIP}(b, t)$
end for
$b^* = \arg\max_{b \in \mathcal{B}} s_b$
return b^*

Algorithm B.4 Algorithm of CLIP Fusion method.

Require: Top-2 predictions from models M_H and M_W, input text t, scaling factor $\lambda = 0.04$
Ensure: Final bounding box prediction b^*
Extract top-2 predictions:
    $P_H = \{(b_1^H, c_1^H), (b_2^H, c_2^H)\}$
    $P_W = \{(b_1^W, c_1^W), (b_2^W, c_2^W)\}$
Initialize candidate set $\mathcal{B} = \{(b_1^H, c_1^H), (b_2^H, c_2^H), (b_1^W, c_1^{W2F}), (b_2^W, c_2^W)\}$
for each $(b, c) \in \mathcal{B}$ **do**
    Compute CLIP similarity score $s = \text{CLIP}(b, t)$
    Compute hybrid score $h = c + \lambda \cdot s$
    Store (b, h)
end for
$b^* = \arg\max_{(b,h)} h$
return b^*

Algorithm B.5 CLIP-Aware Pointing Ensemble (CAPE) algorithm.

Require: Top-2 predictions from models M_H and M_W, input text t, object area threshold $\tau = 0.0048$
Ensure: Final bounding box prediction b^*

```
Extract candidate bounding boxes from both models
Estimate area ratio r of each candidate bounding box to the image size
if r < τ then
    b* = ConfidenceCLIPFusion(M_H, M_W, t)        // Refer to Algorithm B.4
else
    b* = CLIPOnlyTop2Threshold(M_H, M_W, t)       // Refer to Algorithm B.3
end if
return b*
```

C. Discussion

C.1. Why is pointing heatmap useful?

In ERU, the model must associate a textual instruction with the spatial region indicated by the person in the image. Without explicit spatial guidance, the model relies on implicit cues, such as hand position or object features, which become ambiguous when multiple candidate objects fit the text instruction. A pointing line provides a clear directional prior, narrowing the possible targets to regions intersecting or near the line. Formally, if $R(I, T)$ is the set of candidate regions without explicit pointing heatmap (P), then pointing guidance:

$$R'(I, T, P) \subseteq R(I, T) \tag{7}$$

where I denotes the image, T refers to the textual description, and P is the pointing line. This reduces ambiguity and helps the model converge to the correct region.

A second benefit is spatial alignment of multimodal embeddings. Without spatial cues, the latent representation of the intended object may blend with nearby distractors. The pointing line acts as a spatial attention prior, biasing the model toward regions consistent with the pointing direction.

Finally, using a pointing heatmap also strengthens the gradient signal during training. The heatmap provides an additional, well-structured supervisory cue, reducing overfitting to spurious text or background correlations. Effectively, P serves as a spatial inductive bias that stabilizes and accelerates learning.

In summary, the pointing line provides an explicit spatial prior, constrains the candidate space, improves multimodal alignment, and yields more efficient and accurate grounding.

Method	IoU=0.25	IoU=0.50	IoU=0.75
CAPE with SigLIP	74.5	65.1	35.4
CAPE with CLIP	75.0	65.4	35.7

Table C.8. We compare SigLIP with CLIP as part of the CAPE module. CLIP demonstrates higher performance than SigLIP. The experiments are conducted on the YouRefIt dataset.

C.2. Why does CAPE enhance the performance?

The head-to-finger line provides a global directional prior, while the wrist-to-finger line captures local articulation. Together, they offer complementary spatial cues, allowing each network to capture different aspects of the pointing geometry and reducing the risk of misalignment across varied pointing poses. Therefore, we introduce CAPE module to ensemble the outputs of both models (M_H and M_F). We incorporate CLIP in addition to model confidence scores because CLIP learns a joint image–text embedding space where images and text with matching semantics are close. Formally, for an image region I^C and a text description T, CLIP provides a similarity score:

$$S_{CLIP}(I^C, T) = \omega(\phi_I(I^C), \phi_T(T)) \tag{8}$$

where ϕ_I and ϕ_T are image and text encoders, and ω is cosine similarity. This score captures semantic alignment.

Relying only on CLIP is insufficient because, while it provides semantic similarity, it does not encode geometric cues such as pointing direction. This can lead to imprecise localization in cluttered scenes or when multiple semantically plausible objects exist. Conversely, using only model confidence is also unreliable: pointing-based networks provide spatial priors but can be overconfident even when predictions are wrong due to occlusion, unusual poses, or network biases.

The two pointing models provide spatial priors, while CLIP provides semantic priors. Combining them reduces both geometric and linguistic ambiguity. Conceptually, the fusion functions like a Bayesian ensemble. The pointing models supply a geometry-driven likelihood over regions, while CLIP acts as a semantic prior. Integrating these signals produces more sharper and reliable posterior estimate.

As an alternative to CLIP model, we also experimented with SigLIP [80], which provides strong semantic grounding. However, replacing CLIP with SigLIP in the CAPE module yielded slightly lower performance (see Tab. C.8), so we continue using CLIP.

In summary, CAPE fuses pointing-based confidence with CLIP similarity, integrating complementary spatial and semantic signals to improve referent prediction in challenging poses.

Method	Heatmap Input	IoU=0.25	IoU=0.5	IoU=0.75
M_H	Head-to-finger	72.9	62.3	35.1
M_H	Wrist-to-finger	73.0	62.0	35.6
M_H	Ground-Truth	73.0	62.1	36.2
M_H	None	71.6	60.7	32.9
M_W	Wrist-to-finger	69.6	60.7	31.5
M_W	Head-to-finger	68.1	59.2	28.4
M_W	Ground-Truth	69.7	59.9	31.9
M_W	None	68.8	59.7	31.2

Table C.9. We analyze the performance of our models under different heatmap inputs during inference.

Method	Text	IoU=0.25	IoU=0.5	IoU=0.75
M_H	Original	72.9	62.3	35.1
M_H	Dummy	34.0	26.5	12.3
M_H	Random	24.9	19.0	9.4
M_H	No Text	39.2	27.9	10.2
M_W	Original	69.6	60.7	31.5
M_W	Dummy	34.6	25.5	11.0
M_W	Random	25.4	19.3	9.24
M_W	No Text	43.3	31.1	11.7

Table C.10. We analyze the performance of our models under different text inputs during inference.

C.3. Effect of Heatmap

We evaluate the robustness of our models (M_H and M_W) using different heatmap inputs (Table C.9). First, we test M_H using a wrist-to-finger heatmap. While IoU 0.25 and 0.75 show slight gains, integrating this into CAPE reduces overall performance, indicating a loss of complementary effectiveness. In our main experiments, we generate both heatmaps using predicted keypoints from a pose estimation model. In contrast, the *Ground-Truth* setup uses the annotated pointing coordinates, including the center of the ground-truth bounding box, to generate the heatmap. As expected, this yields a small performance boost since the line is guaranteed to intersect the target object's center. However, the improvement is marginal, indicating that our predicted heatmaps are already accurate and that the pointing line reliably guides the model. Finally, providing a zero heatmap (*None*) removes explicit pointing guidance. Performance drops but remains reasonable, demonstrating that the model can rely on text and visual context. Notably, since the model is trained with explicit pointing heatmaps, it has learned to interpret pointing cues; therefore, even without a heatmap at inference, it can still achieve meaningful performance. Repeating the same experiments for M_W shows similar trends, reinforcing the complementary nature of the two heatmaps.

Text Encoder	IoU=0.25	IoU=0.5	IoU=0.75	CLIP ↑	C_D ↓
Frozen, Setup A	69.8	59.5	**33.1**	0.2454	0.2994
Finetuned, Setup A	**71.2**	**60.1**	32.8	**0.2469**	**0.2662**
Frozen, Setup F	71.3	62.1	33.7	**0.2460**	0.2634
Finetuned, Setup F	**72.9**	**62.2**	**35.1**	0.2457	**0.2490**

Table C.11. Ablation study for frozen and finetuned text encoder.

C.4. Effect of Text Input

In summary, for the embodied reference understanding task, text serves as the primary cue for object identification, while pointing information, derived from both the input image and the heatmap, plays a complementary, assistive role.

In Table C.10, we evaluate our models under varying text input conditions to assess the role of textual guidance. The *Original* setup uses the dataset annotations. In the *Dummy* scenario, the text is replaced with the generic word "object," while in the *Random* scenario, text is randomly sampled from other test examples, sometimes contradicting the actual target. Both scenarios cause a notable performance drop, highlighting the importance of textual input. Without accurate text instructions, the pointing line often intersects multiple objects, making it insufficient to identify the correct referent in cluttered scenes.

In summary, text provides the primary cue for referent identification in ERU, while pointing information from the image and heatmap serves as a complementary signal.

C.5. Finetuning or Frozen Text Encoder?

We investigate the impact of freezing the text encoder during training. Based on multiple training experiments, we observed that finetuning the text encoder sometimes leads to instability, represented by the pink line in Fig. C.6. In contrast, freezing the text encoder results in consistently stable training, as shown by the green and orange lines in Fig. C.6. This indicates that finetuning introduces optimization sensitivity and may require multiple attempts to obtain a stable run. Despite this instability, finetuning ultimately yields better results. Table C.11 shows that, for both Setup A and Setup F, finetuning improves mAP at IoU 0.25 and 0.5, and also provides better C_D scores. These gains suggest that adapting the text representations to the specific semantics of embodied reference understanding provides a meaningful advantage, even though the training process becomes less stable.

D. Hyperparameters and Other Training Parameters

In Tab. C.12, we present empirically determined hyperparameters and other key training details to support reproducibility. For most hyperparameters, we follow established values from the literature, as they have been thoroughly an-

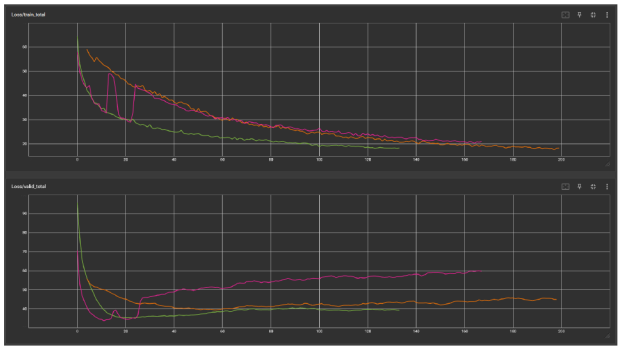

Figure C.6. Graphs of training and validation loss from three separate runs are shown. The green and pink lines represent training with a finetuned text encoder, while the orange line corresponds to training with a frozen text encoder.

Name	Value
Learning rate vision backbone	$5e^{-}5$
Learning rate - text backbone	$1e^{-}4$
Learning rate	$5e^{-}5$
λ_1	2
λ_2	1
λ_3	10
λ_4	10
λ_5	1
λ_6	1
Transformer encoder layer	6
Transformer decoder layer	6
Number of attention heads per layer	8
MLP dimension	2048
Dropout	p = 0.1
Weight decay	1e-4
Batch size	4
Text encoder	Roberta-Base
Image encoder	ResNet-101
Heatmap encoder	ResNet-18
Position embedding	Sine
Input dim of transformer encoder	256
Number of queries	20

Table C.12. Hyperparameters and other training details.

alyzed in prior work [11, 37–39, 43, 81, 85]. We also observe that training is not sensitive to the choice of loss coefficients, as the resulting accuracy differences are negligible.

E. More Results

In Tab. E.13, we report the results of our model on the CAESAR dataset, which is simulation-based. We compare our model with SOTA methods, including Touch-Line, GroundingDINO, PaliGemma2, and Qwen2.5vl. Despite the significant domain gap, our model achieves very accurate performance, attaining SOTA scores in most cases. Under the IoU threshold of 0.75, PaliGemma2 and GroundingDINO achieve higher scores.

In Tab. E.14, we report the mAP at three different thresholds for all ablation setups, categorized by object size. Similarly, in Tab. E.15, we present the CLIP scores and C_D metrics for the same setups, also with respect to object size.

In Fig. E.7, we present example failure cases from the ISL Pointing and CAESAR datasets. In the first column, all models predict the same object, as it aligns better with the pointing line and the text provides limited semantic guidance. Localization-related text is particularly challenging for the models to interpret. In the second column, only Touch-Line-VTL predicts the correct object. In the third column (from ISL Pointing), our models detect the scissors correctly, whereas Touch-Line-VTL detects only a small part of the spoon. Due to perspective and the lack of depth information, our models align the scissors with the pointing line.

In Fig. E.8, we present additional example images from the YouRefIt dataset and qualitatively compare our model with SOTA models. CAPE achieves high accuracy in most cases, whereas Touch-Line often fails to detect the correct object, and the LMMs largely ignore the pointing cue, relying only on the text instruction.

In Fig. E.9, we present example images from the ISL Pointing and CAESAR datasets. The first two columns show ISL Pointing images, while the remaining two columns contain CAESAR samples, which are simulation-based. In the first ISL example, all models except M_W predict the object correctly. M_W makes wrong prediction because its prediction aligns better with the wrist-to-fingertip pointing line than the GT object, and the provided text does not conflict with this prediction. In the second example, M_H fails, but CAPE correctly selects M_W's prediction. Touch-Line-VTL, PaliGemma2, Qwen2.5vl, and GroundingDINO make mistakes as they overlook the pointing cue. On the CAESAR dataset, despite the severe domain gap, our model and Touch-Line-VTL achieve strong performance. Our models closely follow the pointing cue and detect the correct object more accurately than other models. Given the challenging text instructions in both datasets, accurately and efficiently leveraging the pointing direction is key, explaining why CAPE surpasses all other models.

IoU Threshold for mAP	0.25				0.50				0.75			
Object Sizes	All	S	M	L	All	S	M	L	All	S	M	L
PaliGemma2	54.4	46.2	53.0	64.1	**51.2**	42.4	47.2	64.1	**44.5**	26.2	43.3	**64.1**
Qwen2.5vl	36.4	21.0	37.2	51.1	31.6	14.9	32.0	47.9	21.0	10.0	19.3	33.7
Grounding DINO	47.6	40.7	47.9	54.4	46.8	38.7	47.4	54.4	**44.5**	**33.1**	**46.1**	54.4
Touch-Line-EWL	50.7	49.8	53.2	45.6	46.5	45.5	49.3	41.3	24.6	20.3	30.9	29.3
Touch-Line-VTL	45.9	44.2	51.2	32.6	40.3	39.5	43.2	32.6	20.6	13.9	29.4	32.6
M_H	**55.5**	52.3	**57.6**	70.6	46.4	39.7	**55.6**	70.7	15.2	3.3	26.8	62.0
M_W	48.6	46.2	49.4	64.1	42.5	40.1	43.1	59.8	17.7	7.8	28.8	43.5
CAPE w/ M_H + M_W	**55.5**	**52.9**	55.8	**75.0**	50.5	**47.3**	50.9	**75.0**	18.6	9.1	26.6	57.6

Table E.13. Comparison of our model with prior works, SOTA LMMs, and Grounding-DINO in terms of mean Average Precision (mAP) at different IoU thresholds, across various object sizes, on the CAESAR dataset.

	IoU Threshold for mAP			0.25				0.50				0.75			
Setup	Object Sizes	Heatmap Injection	Ensemble	All	S	M	L	All	S	M	L	All	S	M	L
	PaliGemma2	-	-	58.8	29.0	53.5	75.8	46.9	22.1	50.8	68.0	31.7	6.2	34.1	54.8
	Qwen2.5vl-3b	-	-	38.9	17.0	41.8	58.0	31.0	11.1	33.6	48.1	20.0	5.7	19.8	34.5
	Grounding DINO	-	-	57.9	38.0	60.9	74.9	54.9	35.7	59.3	69.6	**42.3**	**22.7**	**45.9**	**58.4**
	Touch-Line-EWL [38]	-	-	69.5	56.6	71.7	80.0	60.7	44.4	66.2	71.2	35.5	11.8	38.9	55.0
	Touch-Line-VTL [38]	-	-	71.1	55.9	75.5	81.7	63.5	47.0	70.2	73.1	39.0	13.4	45.2	57.8
1	Baseline	-	-	71.2	59.8	73.0	80.9	60.1	43.2	66.4	70.5	32.8	8.6	35.4	53.4
2	Setup 1 + object center prediction	-	-	70.8	59.5	73.2	79.5	61.7	45.8	67.4	71.5	34.6	14.0	38.1	51.1
3	Setup 1 + W2F heatmap	Embedding feature	-	68.9	55.5	74.0	77.4	59.7	42.0	68.1	69.3	32.4	8.7	37.8	50.6
4	Setup 1 + H2F heatmap	Embedding feature	-	71.9	57.5	74.1	84.1	62.8	43.3	70.1	75.1	33.8	10.0	35.6	55.3
5	Setup 2 + W2F heatmap	Embedding feature	-	69.6	57.7	74.9	76.4	60.7	45.5	69.0	68.2	31.5	12.1	35.3	47.0
6	Setup 2 + H2F heatmap	Embedding feature	-	72.9	59.5	77.8	81.4	62.3	44.3	70.9	42.0	35.1	10.4	39.3	55.3
7	Setup 1 + W2F heatmap + H2F heatmap	Embedding feature	-	70.2	59.0	71.4	80.2	60.2	44.5	65.6	70.5	33.8	11.9	39.3	50.0
8	Setup 6	Channel-wise input	-	68.7	58.5	71.9	75.6	58.5	44.8	65.0	65.9	33.4	13.9	37.4	48.6
9	Setup 6	Channel-wise feature	-	72.6	60.0	77.5	80.4	58.4	38.8	68.7	68.1	27.9	8.5	32.6	42.5
10	Setup 5 + Setup 6	Embedding feature	Confidence-Only	73.4	62.7	78.6	79.0	64.1	49.0	72.7	71.0	34.1	13.4	38.2	50.5
11	Setup 5 + Setup 6	Embedding feature	CLIP-Only (Top-1)	73.5	59.7	79.9	81.2	64.2	46.5	74.3	72.2	35.4	12.7	39.8	53.4
12	Setup 5 + Setup 6	Embedding feature	CLIP-Only (Top-2 + Threshold)	73.8	59.7	80.2	81.6	64.1	46.3	74.3	72.2	35.5	12.4	40.1	53.9
13	Setup 5 + Setup 6	Embedding feature	CLIP Fusion	73.6	63.2	78.4	79.6	64.4	49.3	72.8	71.7	34.5	13.4	38.5	51.7
14	Setup 5 + Setup 6	Embedding feature	CAPE	75.0	63.2	80.2	81.8	65.4	49.5	74.3	72.7	35.7	13.4	40.1	53.4
15	Setup 6 w/ frozen text encoder	Embedding feature	-	71.3	56.0	74.9	83.1	62.1	42.5	69.5	74.4	33.7	10.2	35.8	54.6
16	Setup 6 w/ Conic Attention Heatmap (15°)	Embedding feature	-	70.0	54.2	72.2	83.3	60.0	42.0	66.6	71.5	33.9	12.2	40.6	48.8
17	Setup 6 w/ Conic Attention Heatmap (30°)	Embedding feature	-	71.5	58.7	73.5	82.1	59.4	41.8	64.7	71.7	30.8	7.0	32.4	52.7
	Ours - Final	Embedding feature	CAPE	**75.0**	**63.2**	**80.2**	**81.8**	**65.4**	**49.5**	**74.3**	**72.7**	35.7	13.4	40.1	53.4

Table E.14. Detailed mAP results of all ablation setups with respect to the object size.

				CLIP Score				C_D			
Setup	Object Sizes	Heatmap Injection	Ensemble	All	S	M	L	All	S	M	L
	Touch-Line-EWL [38]	-	-	0.2456	0.2312	0.2435	0.2615	0.3168	0.3006	0.2903	0.3564
	Touch-Line-VTL [38]	-	-	0.2456	0.2308	0.2440	0.2615	0.2843	0.2809	0.2276	0.3393
1	Baseline	-	-	0.2470	0.2334	0.2447	0.2620	0.2662	0.2182	0.2506	0.3266
2	Setup 1 + object center prediction	-	-	0.2446	0.2316	0.2426	0.2596	0.2707	0.2201	0.2692	0.3222
3	Setup 1 + W2F heatmap	Embedding feature	-	0.2451	0.2311	0.2432	0.2604	0.2984	0.2763	0.2616	0.3533
4	Setup 1 + H2F heatmap	Embedding feature	-	0.2458	0.2312	0.2444	0.2617	0.2694	0.2649	0.2233	0.3172
5	Setup 2 + W2F heatmap	Embedding feature	-	0.2448	0.2313	0.2435	0.2593	0.2770	0.2568	0.2319	0.3382
6	Setup 2 + H2F heatmap	Embedding feature	-	0.2458	0.2318	0.2448	0.2606	0.249	0.222	0.2202	0.3024
7	Setup 1 + W2F heatmap + H2F heatmap	Embedding feature	-	0.2448	0.2322	0.2426	0.2593	0.2744	0.2540	0.2314	0.3335
8	Setup 6	Channel-wise input	-	0.2449	0.2311	0.2424	0.2609	0.2959	0.2437	0.2642	0.3767
9	Setup 6	Channel-wise feature	-	0.2457	0.2318	0.2437	0.2613	0.2598	0.2289	0.2119	0.3339
10	Setup 5 + Setup 6	Embedding feature	Confidence-Only	0.2578	0.2480	0.2614	0.2641	0.247	0.2176	0.2042	0.3141
11	Setup 5 + Setup 6	Embedding feature	CLIP-Only (Top-1)	0.2644	0.2563	0.2670	**0.2699**	**0.2391**	0.2242	0.2011	**0.2878**
12	Setup 5 + Setup 6	Embedding feature	CLIP-Only (Top-2 + Threshold)	0.2460	**0.2656**	0.2448	0.2606	0.2422	0.2158	**0.2009**	0.3060
13	Setup 5 + Setup 6	Embedding feature	CLIP Fusion	0.2581	0.2642	0.2485	0.2614	0.2434	0.2137	0.2028	0.3090
14	Setup 5 + Setup 6 (Final Model)	Embedding feature	CAPE	**0.2661**	0.2642	**0.2672**	0.2670	0.2476	**0.2137**	0.2241	0.3023
15	Setup 6 w/ frozen text encoder	Embedding feature	-	0.2461	0.2318	0.2441	0.2618	0.2634	0.2626	0.2233	0.3006
16	Setup 6 w/ Conic Attention Heatmap (15°)	Embedding feature	-	0.2462	0.2321	0.2442	0.2620	0.2695	0.2685	0.2344	0.3028
17	Setup 6 w/ Conic Attention Heatmap (30°)	Embedding feature	-	0.2464	0.2323	0.2444	0.2621	0.2765	0.2750	0.2499	0.3026
	Ours - Final	Embedding feature	CAPE	**0.2661**	0.2642	**0.2672**	0.2670	0.2476	**0.2137**	0.2241	0.3023

Table E.15. Detailed CLIP scores and C_D metrics of all ablation setups with respect to the object size.

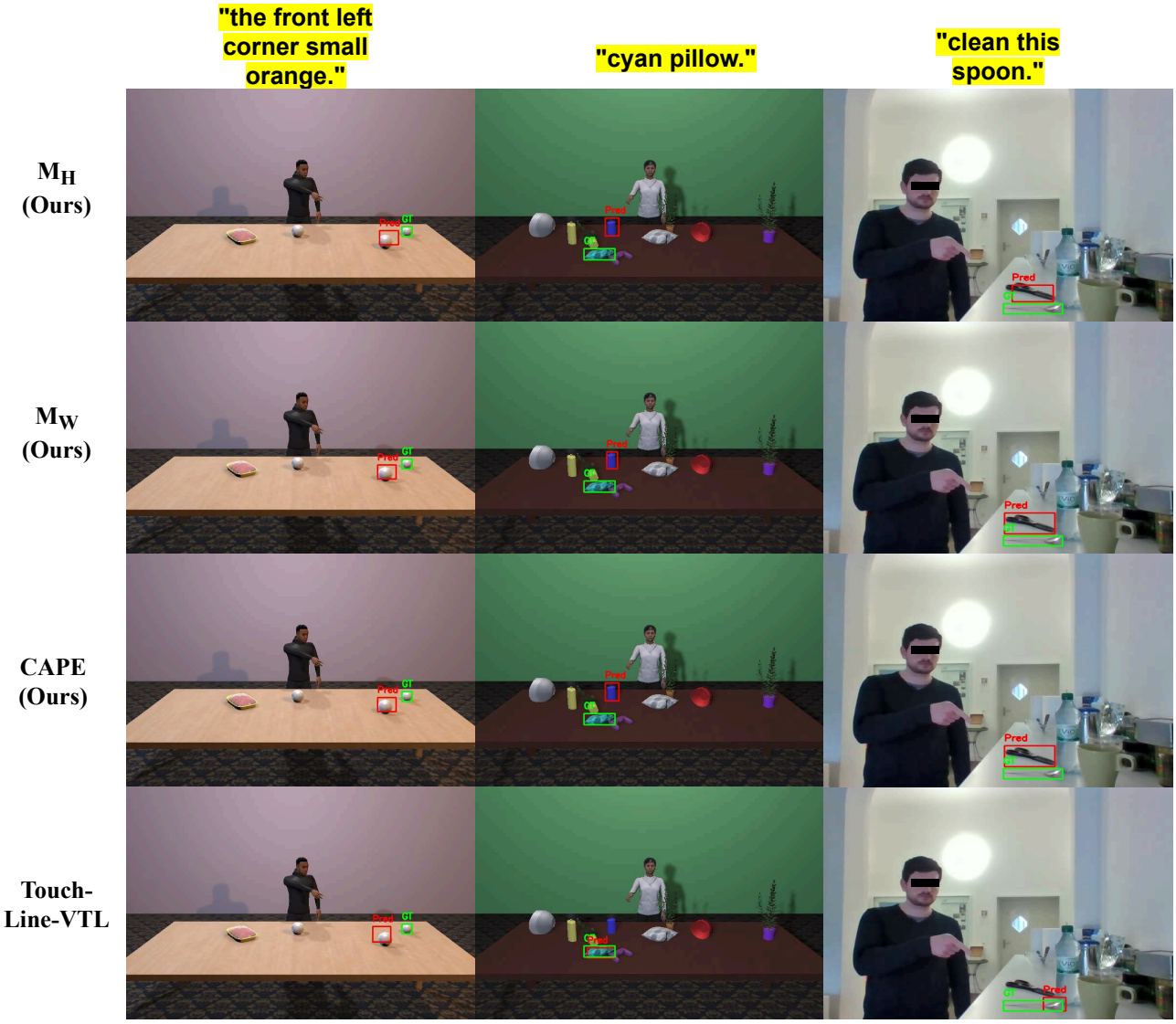

Figure E.7. Some examples for failure cases from ISL pointing and CAESAR datasets.

Figure E.8. Example results from different models on the YouRefIt dataset.

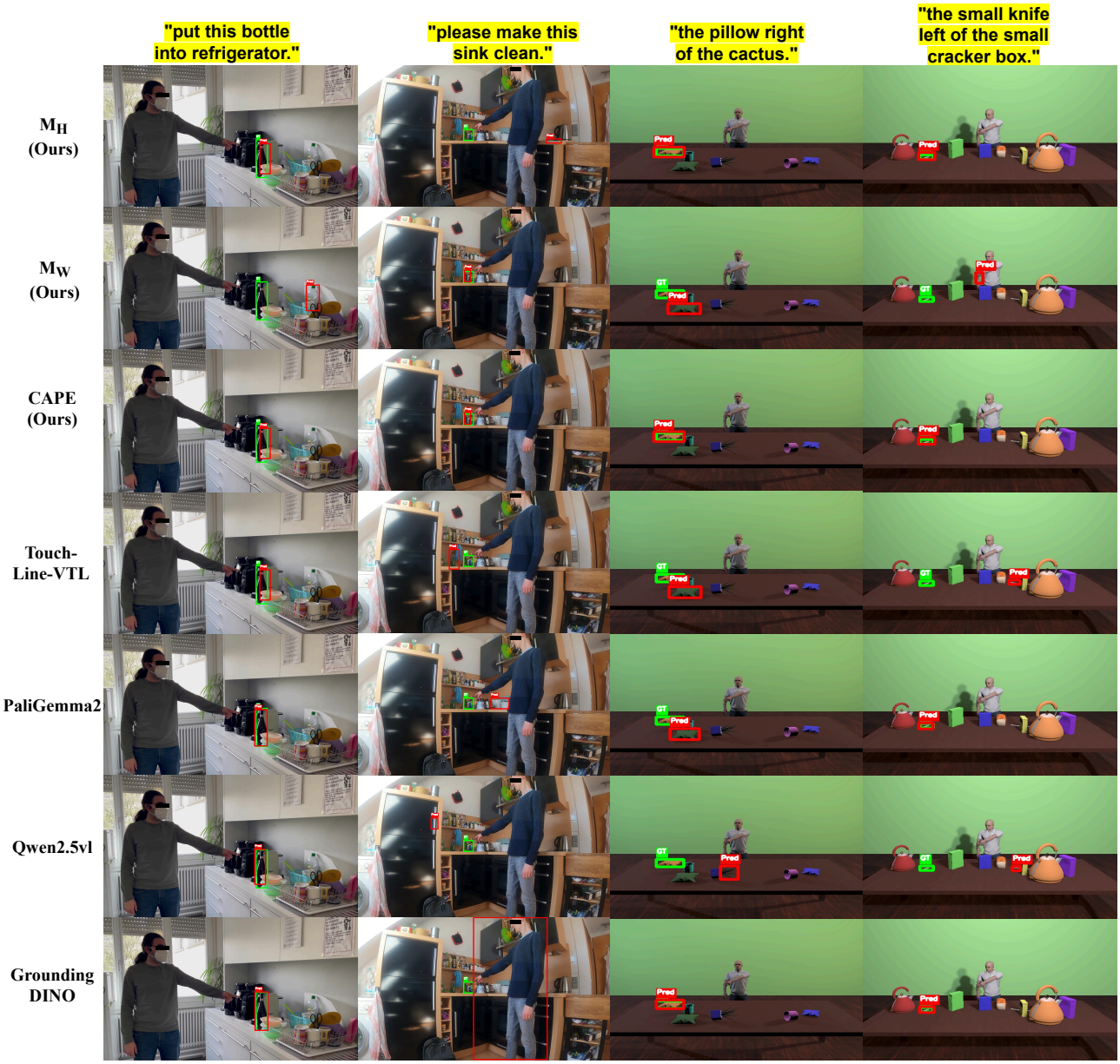

Figure E.9. Example results from different models on the ISL Pointing and CAESAR datasets. While first two columns have examples from ISL Pointing dataset [16], remaining two columns contain sample images from CAESAR dataset [29].