Appendix for PerceptionGPT: Effectively Fusing Visual Perception into LLM

In this appendix, we first introduce the detailed design choices of our perception encoder and decoders, as well as the training losses for perception tasks; then we elaborate more details of the inference procedure that was skipped in the main paper due to space limitation; afterwards, we illustrate more visualization results of our PerceptionGPT on spot captioning and reasoning-based detection and segmentation.

Algorithm 1 Inference Process of PerceptionGPT

Input: An image I and a textual instruction T
Output: Generated tokens and perception signals
$E_{\text{image}} \leftarrow \text{ImageEncoder}(I)$ {Extract visual tokens}
$E'_{\text{image}} \leftarrow \text{ProjectionLayer}(E_{\text{image}})$ {Map to LLM's embedding space}
$E_{\text{concat}} \leftarrow \text{Concatenate}(V', T)$ {Concatenate image and text features}
$E_{\text{new}} \leftarrow \text{LLM}(E_{\text{concat}})$ {Initial input to the LLM}
$Tokens \leftarrow []$
$Perceptions \leftarrow []$
token $\leftarrow$ TokenClassification(E')
while token is not <eos> **do**
  token $\leftarrow$ TokenClassification(E_{new})
  $Tokens \leftarrow \text{Concatenate}(Tokens, \text{token})$
  if token is *<vis>* **then**
    Signal $\leftarrow$ PerceptionDecoder(E')
    $E_{\text{percept}} \leftarrow$ PerceptionEncoder(Signal)
    $Perceptions \leftarrow \text{Concat}(Perceptions, \text{Signal})$
    $E_{\text{concat}} \leftarrow \text{Concatenate}(E, P)$
  else
    $E_{\text{text}} \leftarrow$ TextEncode(token)
    $E_{\text{concat}} \leftarrow \text{Concatenate}(E, E_{\text{text}})$
  end if
  $E_{\text{new}} \leftarrow \text{LLM}(E_{\text{concat}})$
end while
return $Tokens, PerceptionSignals$

1. Detailed Design Choices

1.1. Design of Perception Encoders and Decoders

In this section, we introduce the detailed design choices of our perception encoder and decoders. Notably, although we incorporate detection and segmentation tasks in this paper, other perception tasks can be supported in a similar manner by leveraging encoder and decoder particularly for the new task.

Bounding Box Encoder The bounding box encoder is implemented with a simple 3-layer MLP, with input dimension equal to 4, corresponding to the coordinates of the top-left and bottom-right bounding box coordinates; the output dimension is set to 4096, which equals to the dimension of the token embedding. The intermediate dimension is set to 512.

Bounding Box Decoder Similar to the bounding box encoder, the decoder is also implemented with a simple 3-layer MLP, with input dimension set to 4096 and output dimension set to 4, the intermediate dimension is set to 512.

Mask Encoder The mask encoder is implemented with a ResNet18-like architecture, the intermediate dimension of all blocks are set to 32. The input to the mask encoder should be mask images with 1 channel. The final linear layer maps the flattened image feature to the LLM's token embedding dimension, which is equal to 4096.

Mask Decoder The design of the mask decoder is inspired by Segment Anything model [1], which consists of two layers of two-way-transformer layers.The decoder block conducts self-attention and cross-attention in two directions (LLM token embedding-to-image embedding and image embedding-to-LLM token embedding). After conducting fusion through two blocks, the updated image embedding is up-sampled and generates the segmentation mask.

1.2. Choices of Perception Task Losses

In this section, we elaborate on the choices of perception task losses. Incorporating such losses during training helps leveraging the prior knowledge and characteristics of the tasks, which further helps learnining.

Bounding Box Loss We adopt a combination of L1-norm and GIoU [3] losses, which can be expressed as follows:

$$\mathcal{L}_{\text{box}}(b_{\text{gt}}, S_{in}, I) = \mathcal{L}_{\text{L1}}(b_{\text{gt}}, b_{\text{pred}}) + \mathcal{L}_{\text{GIoU}}(b_{\text{gt}}, b_{\text{pred}}(S_{in}, I)) \quad (1)$$

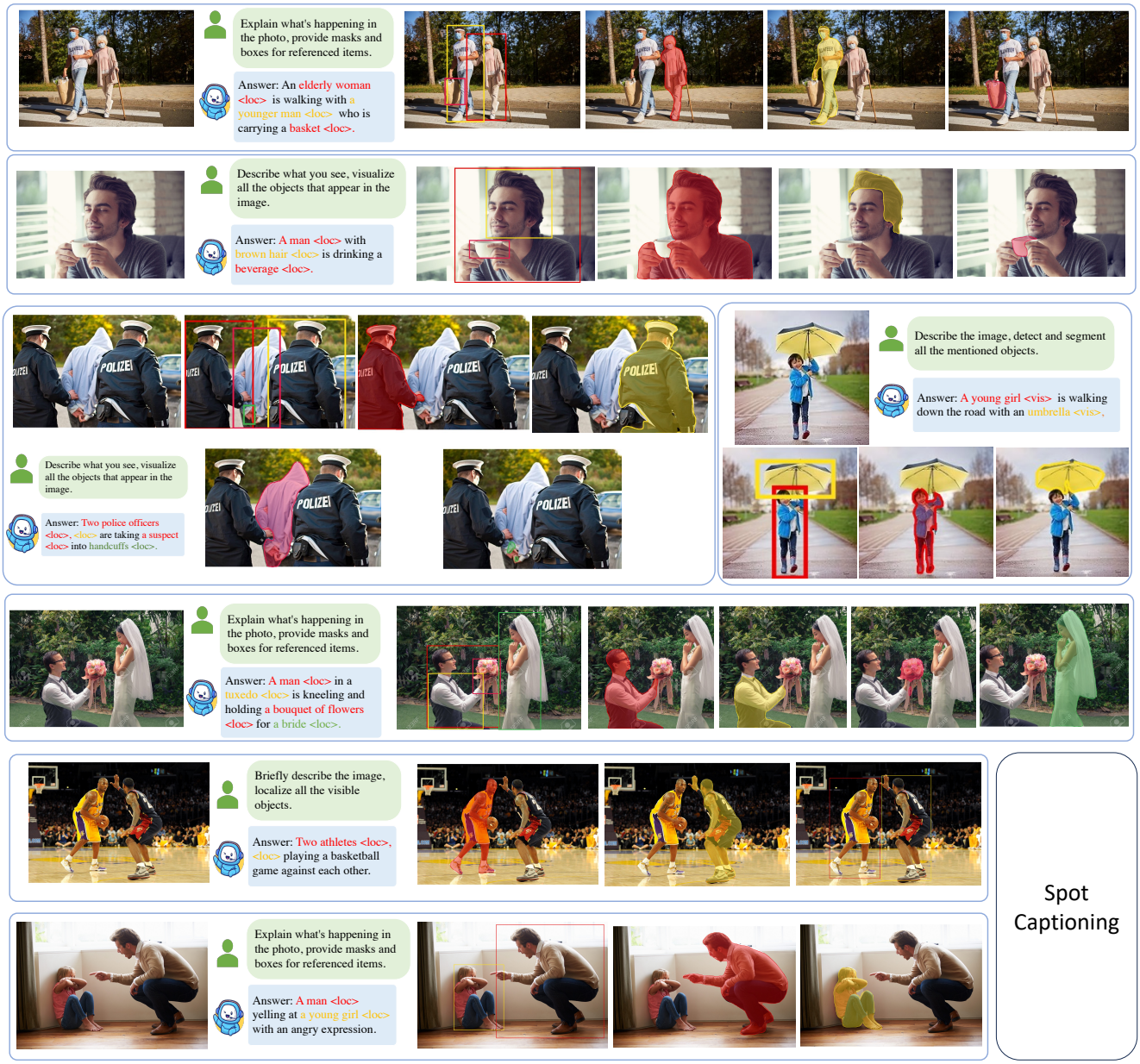

Figure 1. Visualization of results from PerceptionGPT on spot captioning task, where the model is asked to generate a description regarding the image content, while detecting and segmenting the mentioned objects at the same time.

The Generalized IoU (GIoU) loss is then defined as:

$$\mathcal{L}_{\text{GIoU}}(b_{\text{gt}}, b_{\text{pred}}) = 1 - \left(\text{IoU} - \frac{\text{area}(C) - \text{area}(b_{\text{pred}} \cup b_{\text{gt}})}{\text{area}(C)} \right) \tag{2}$$

where C is the smallest enclosing box that contains both b_{pred} and b_{gt}.

Mask Loss We adopt a combination of L1-norm and GIoU losses, which can be expressed as follows:

$$\mathcal{L}_{\text{mask}}(m_{\text{gt}}, S_{in}, I) = \mathcal{L}_{\text{L1}}(m_{\text{gt}}, m_{\text{pred}}) + \mathcal{L}_{\text{DICE}}(m_{\text{gt}}, m_{\text{pred}}(S_{in}, I)) \tag{3}$$

The DICE loss can be formulated as follows:

$$\mathcal{L}_{\text{DICE}}(m_{\text{gt}}, m_{\text{pred}}) = 1 - \text{Dice} = 1 - \frac{2 \times |m_{\text{pred}} \cap m_{\text{gt}}|}{|m_{\text{pred}}| + |m_{\text{gt}}|} \tag{4}$$

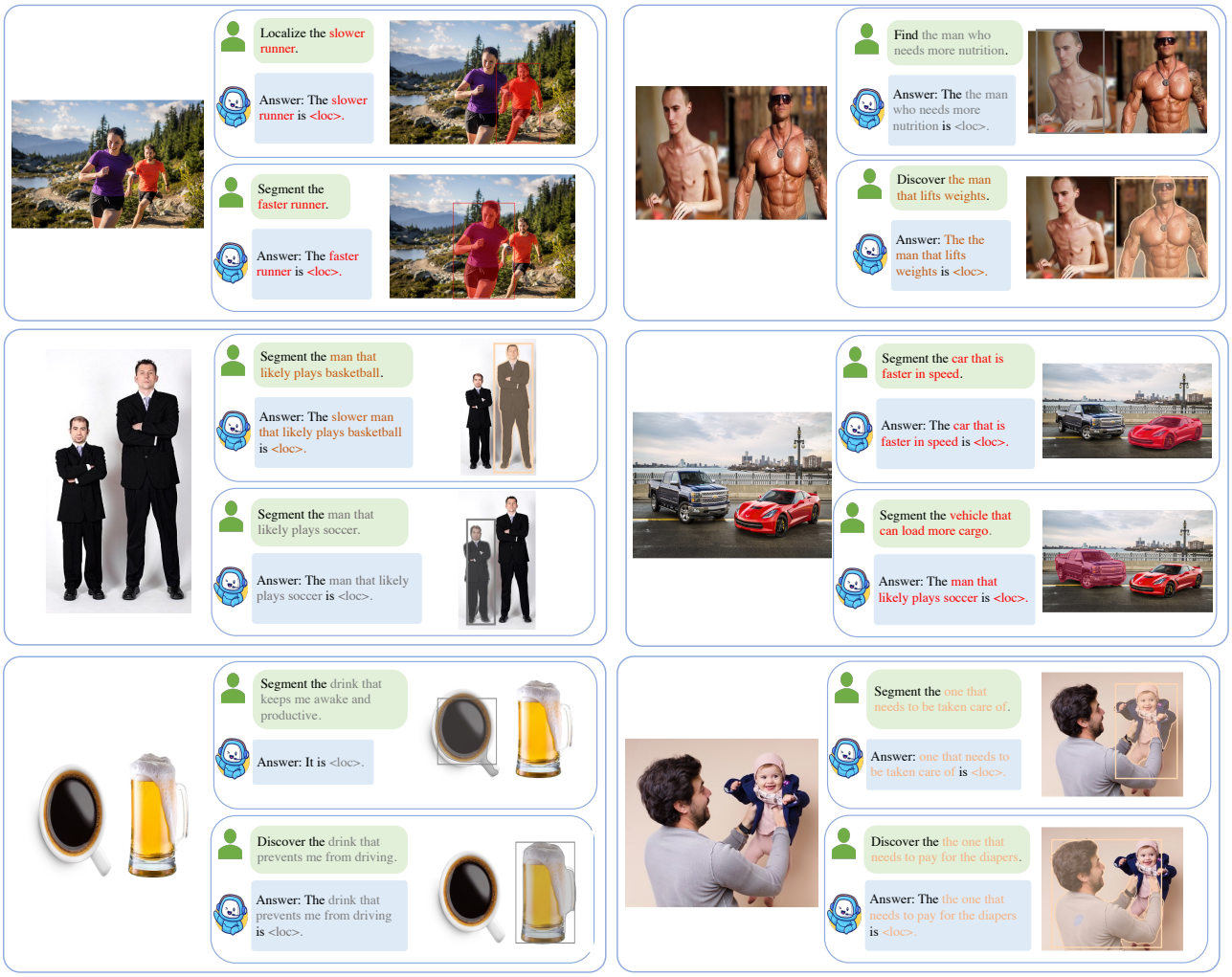

Figure 2. Visualization of results from PerceptionGPT on reasoning-based detection and segmentation task. Our PerceptionGPT is able to detect and segment the object related to the instruction via reasoning, even if the object is not explicitly givn by the user.

where

$$\text{Dice} = \frac{2 \times |m_{\text{pred}} \cap m_{\text{gt}}|}{|m_{\text{pred}}| + |m_{\text{gt}}|} \tag{5}$$

2. Detailed Inference Procedure

We demonstrate the detailed inference procedure in Algorithm 1. During inference, given an image and a textual instruction, the image encoder first extracts the visual tokens from the image, which are then mapped to the dimension of LLM's embedding space via the projection layer. Then, the mapped image features are concatenated with text embeddings to serve as the input to the LLM. Subsequently, the LLM begins to iteratively perform next-token-generation similar to previous VLLMs [2, 4], until the end of sentence token is generated.

Perception Signal as Input. When the input contains a perception signal, our lightweight perception encoder maps it into the embedding space of the large language model (LLM), which is treated as the embedding for <*vis*> token. This embedding is then concatenated with other embeddings before being processed by the LLM.

Perception Signal as Output. During inference, when a token is decoded as <*vis*> , its associated embedding is extracted and processed by the perception decoder to reconstruct the original signal.

Notably, during inference, when a <*vis*> is generated, we use the embedding of bounding box in place of <*vis*> to generate the next token, while this choice is arbitrary and the mask embedding can also be used.

References

[1] 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, 2023. 1

[2] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. 3

[3] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression, 2019. 1

[4] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models, 2023. 3