TRISHUL: Towards Region Identification and Screen Hierarchy Understanding for Large VLM based GUI Agents

Supplementary Material

1. Model Specifications and Endpoints

Since all our work leverages closed-source models like GPT-4V, GPT-4o, and Claude, we mention the model identifiers that we use for our API calls for clarity. For GPT-4V - "gpt-4-vision-preview", For GPT-4o - "gpt-4o-2024-08-06", and for Claude - "claude-3-5-sonnet-20241022". Unless otherwise noted, all experiments are conducted with a temperature setting of 0.0.

2. Hierarchical Screen Parsing Details

2.1. IoS Score

Similar to IoU score we define an IoS score as:

```
def IoS(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
    interArea = max(0, xB - xA) * max(0, yB
        - yA)
    boxAArea = (boxA[2] - boxA[0]) * (boxA
        [3] - boxA[1])
    ios = interArea / float(boxAArea + 1e
        -3)
    return ios
```

The IoS (Intersection over Size) score is a measure used to evaluate the overlap between two bounding boxes, typically in the context of object detection. It calculates the ratio of the intersection area between two boxes to the area of the first box. IoS(A, B) (also written as IOS_A), a score of 0.5 means 50% of A intersects with B.

2.2. Filtering Redundant Bounding boxes

The output of EasyOCR + SAM model combined is extremely cluttered (see 1) and contains numerous overlaps and false positive detections from both models. We deploy the following steps to parse the outputs of SAM and OCR (local elements as referred to in the main script) together.

- **Generate GROI, Icon and Button Candidate Proposals** Classify all SAM boxes based on $A_{thresh-GROI}$, $A_{thresh-icon}$, $A_{thresh-button}$. Let B represent the set of bounding boxes detected in the GUI.
  Global Region of Interest (GROI) Candidates: The set of boxes with an area greater than the GROI threshold is given by:

$$\text{GROI} = \{b \in B \mid \text{Area}(b) > A_{thresh-GROI}\}$$

  Icon Candidates:The set of boxes with an area between the Button and Icon thresholds is defined as:

$$\text{Icon} = \{b \in B \mid A_{thresh-button} < \text{Area}(b) < A_{thresh-icon}\}$$

  Button Candidates: The set of boxes with an area less than the Button threshold is:

$$\text{Button} = \{b \in B \mid \text{Area}(b) < A_{thresh-button}\}$$

- **Remove False Positive Text Bounding Boxes:** Remove text boxes using a predefined dictionary, that are likely OCR mis-detections for icons. These texts usually contain only special characters or short, meaningless words. If a word contains one of these characters and has a length of less than ¡ 5 that text bbox is ignored.
  Characters/Words to ignore:
  – "@", "#", "x", "?", "{", "}", "<", ">", "&", "`", "~", " ", "=", "C", "Q", "88", "83", "98", "15J", "^", "0e", "n", "E", "ya", "ch", "893"
- **Remove Icons Inside or Overlapping Text Bounding Boxes:** Remove icon bounding boxes that are either inside or intersect with text boxes, as they are likely to be text misidentified as icons by SAM.
- **Filter Square-like Icon Bounding Boxes:** Keep only icons that are roughly square-shaped, based on a specific aspect ratio range of [0.7, 1.3].
- **Remove Redundant Icon and Button Bounding Boxes:** Remove icon bounding boxes that are redundant, i.e., those that are inside or significantly overlap with $IoS > 0.6$ with other icons or text boxes.

2.3. Non Max Suppression for GROIs

- **Reject boxes with low Information score**S If the current bounding box has an information score $S < S_{thresh}$ it is rejected. S_{thresh} is set to 10 for the ScreenPoint and Read task and 25 for action grounding task.
- **Reject Overlapping BBoxes:** If the current bounding box intersects with a previously selected bounding box with a higher Infromation score and $IoS_{current} > IoS_{overlap-thresh}$ it is rejected. $IoS_{overlap-thresh}$ thresh is set to 0.5 for visual grounding task and 0 for ScreenPR task.
- **Reject Contained BBoxes (Smaller GROIs Inside Larger):** If the current bounding box is inside a previously selected bounding box with a higher Information

Score and if $IoS_{current} > IoS_{inside-thresh}$ it is rejected. $IoS_{overlap-thresh}$ thresh is set to 0.5 for visual grounding task and 0 for ScreenPR task.

- **Reject Engulfing Bboxes (Larger GROIs Inside Smaller):** If the current bounding box completely engulfs a bounding box with a higher Information Score then it is rejected.

2.4. GROI analysis for ScreenSpot and VisualWebBench

We plot two additional statistics for the detected GROI's through our HSP block. In Figure 3 we plot the average number of GROIs per image, across the three different sub-categories of ScreenSpot and the full dataset of VisualWebBench. We observe that GUI screenshots from mobiles have the lowest average number of GROIs per image. This is due to the fact that mobile regions are not semantically coherent, therefore lesser number of GROIs are generated.

In Figure 4 we plot the average of the total area covered by all GROIs in an image to the total area of the image. Mobile GUI screenshots have the least dense GROI coverage, due to the fact that we also detect fewer GROIs in mobile screenshots. These studies further validate the fact that GROIs are not as useful for mobile GUI's however they offer more benefit for PC and web based GUIs.

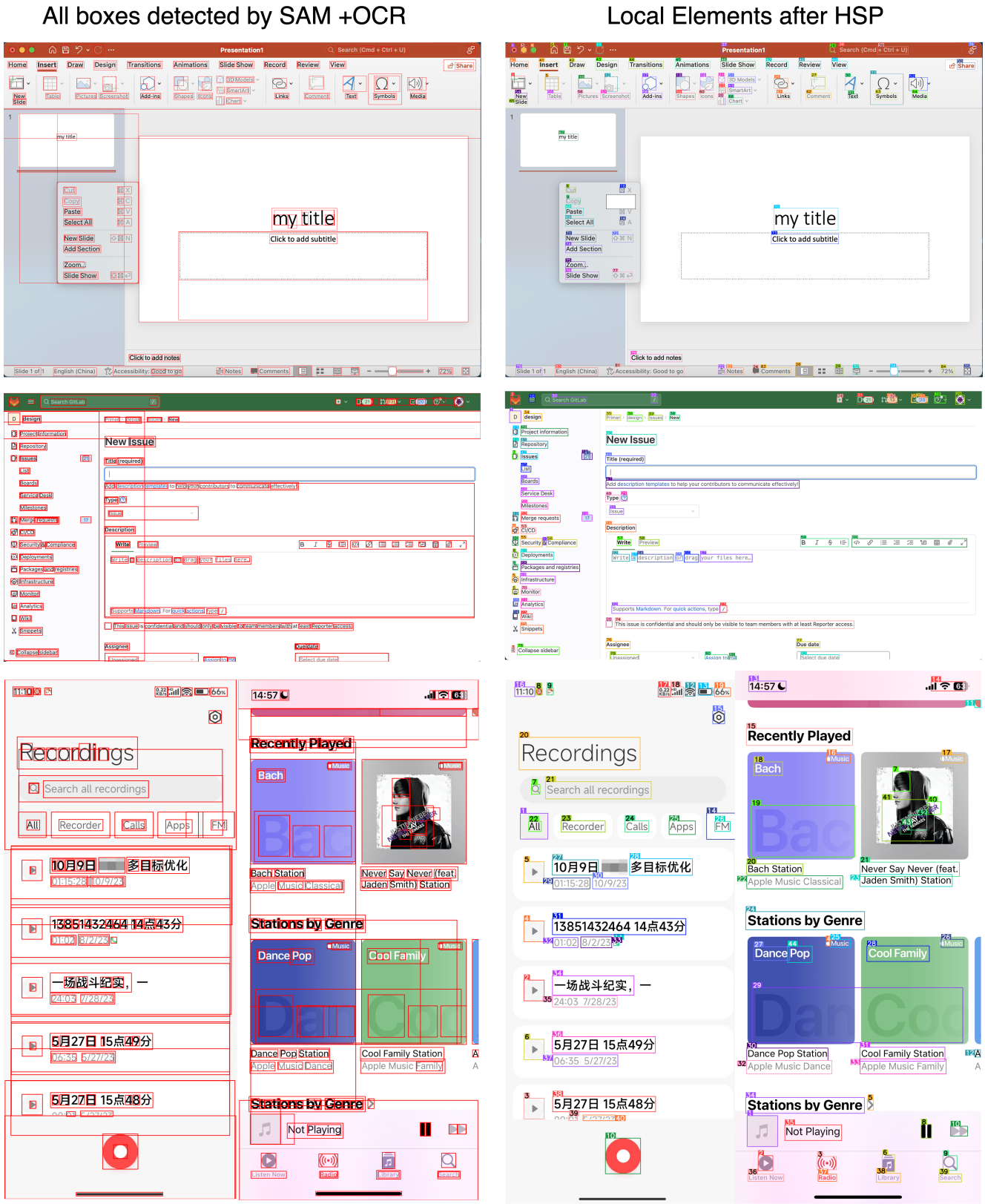

Figure 1. Candidate bounding boxes generated from SAM + OCR to the left and the corresponding HSP results (Icon + text + picture) to the right

You are a smart screen reader that outputs concise natural language to answer questions from users based on the area (box 1) pointed out by the user shown as a red dot on the screen. The red dot is inside the box 1 in the first image, at (x,y) = ({},{}), where x and y are the normalized coordinates.
Note: box 1 is the box with label 1 and box 2 is the box with label 2, box 1 is located inside box 2
Note: the first image shows what is inside box 2, and the second image shows the complete screen.
Note: if the user asks about the location, focus on the layout, explain where box 1 is in box 2 and then explain where box 2 is in the overall complete screen.
Note: don't mention box 1, box 2 or the red dot in the output.

User question: (1) what is this? (2) where it is located in the screen?
Your output should in format (1) ... (2) ...

Figure 2. Prompt for Screen Point-and-Read

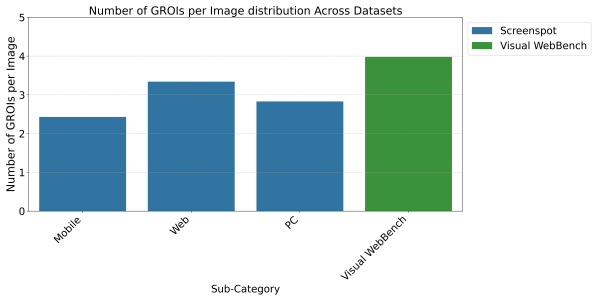

Figure 3. Distribution of Number of GROIs per image for ScreenSpot and Visual WebBench

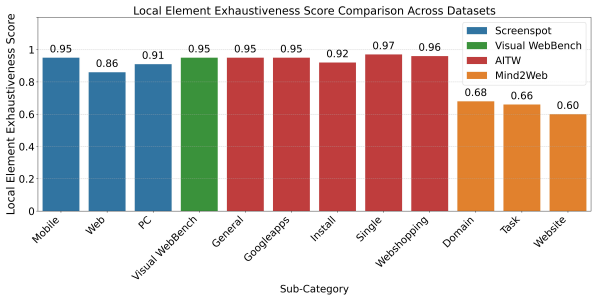

Figure 4. Distribution of Total GROI area / Image area for ScreenSpot and Visual WebBench

You are an expert User Interface (UI) navigation agent. You will receive a screenshot from a web, mobile, or PC interface, an instruction, and cropped Global Regions of Interest (GROIs) from the screenshot. Each GROI is annotated on the full image for context.

Your task:
1. Analyze each GROI individually using the full image for context.
2. Briefly describe the content and functionality of each GROI.
3. Identify the top 3 GROIs most likely to contain the UI element needed to complete the instruction successfully, listing them in order of relevance.
4. Incase of Overlap between 2 GROIs give preference to the smaller GROI.

Output format:
Description:
- ID 1: <Brief description of GROI ID 1>
- ID 2: <Brief description of GROI ID 2>
- ... (Repeat for each GROI)

Proposed GROIs:
[<ID of the most relevant GROI>, <ID of the second most relevant GROI>, <ID of the third most relevant GROI>]
Output the IDs as a list of Integers.

Figure 5. Prompt for instruction guided GROI Proposal generation

```
You are an expert User Interface (UI) navigation agent. You will be given a screenshot of a UI of a mobile / desktop / web application.
The screenshot will contain bounding box annotation for UI elements such icon/picture/text etc type elements.
Additionally you will also be given an input prompt, containing information about the the UI elements. \n
Your task is to, improve the prompt by describing the functionality of the various elements, correcting mistakes and establishing
spatial and semantic relations between different types of elements tagged in the UI.\n
Input Description:\n
1.) There are 2 types on UI elements annotated onto the screenshot, "image" (generated using semantic segmentation) and "text" (generated using OCR).
2.) Each UI element will have a bounding box and a numeric ID associated with it.
3.) For "image" type UI elements, each bounding box and its respective ID tag will share the same color, The ID will be at the top-left corner of the bounding box.\n
4.) For "text' type UI elements, the associated bounding box  will be "red" in color, the corresponding ID tag will NOT be annotated onto the image to avoid cluttering. \n
5.) An input prompt with the description of all UI components in the following format will also be provided:\n
      [ID][[x1, y1, x2, y2]][text/image][Content]\n
6.) For each element [ID] represents the associated ID tag, [x1, y1, x2, y2] are the endpoint coordinates of the rectangular bounding box associated with the element. [text/image] represents the type of the UI element.
   [Content] is the text associated with the bounding box for text elements, the field will be left empty for image elements\n
Task:\n
For each instruction think carefully and follow these four steps in order:\n
   1.) Generate a brief description of the functionality and content of the UI screenshot provided.
   2.) For each "text" type UI element:\n
      2.1) Analyze its bounding box coordinates, and its content provided in the input prompt, use the UI screenshot
         as reference to spell check the content, in case of errors, fix the errors in place.\n
      2.2) For all "text-a" type elements append its functionality in context of the associated elements to the description.
         (e.g., "search" text is a search bar, "https://..." text link is a URL input field, "Jon" name text is an input field for name.)
      2.3) Classify the element as:
         2.2.1) "text-p": (text paired) if the text element is associated with a nearby "image" type element.
            Note:
            1.) Use semantic matching and proximity to determine if a "text" element should be paired with an "image" element. (e.g., "Settings" text should pair with a gear icon).
            2.) Each "text" type element can be mapped to one "image" type element at max.
         2.2.2) "text-a": (text actionable) If the "text" element is associated with any functionality classify it as "actionable" ( e.g. input field, search bar, hyperlink, font change, language etc.)
         2.2.3) "text-s": (text standalone) if the text element is NOT associated with any nearby "image" type elements or any functionality.
      2.4) For all "text-p" type elements append the ID of the associated "image" type element to the description.
   3.) For each "image" type UI element:\n
      3.1) Classification: Do a hierarchal classification of all "image" type elements in this order "icon-p", "icon-s", "text-o", "picture" or "irrelevant"\n
         3.1.1) "icon-p": (icon-paired) refers to clickable elements like application icons, buttons etc. that have a nearby text element associated with them in the UI (e.g. Wifi icon with a wifi text at its bottom).\n
         3.1.2) "icon-s": (icon-standalone) refers to all clickable elements like application icons, buttons etc. that are not associated with any text elements in the UI.(eg: play, pause, drop down, add to cart, back, wishlist, settings etc)\n
         3.1.3) "text-SAM" (text SAM) refers to text elements detected by the segmentation model that were missed by the OCR model.\n
         3.1.4) "picture" refers to static images or parts of image embedded in the UI that have been detected by segmentation and are useful. (eg: picture of a dog)\n
         3.1.5) If an element does not fit any one of the aforementioned descriptions  then and then only classify it as "irrelevant"
      3.2) Semantic and Functionality description generation:
         3.2.1) For each "icon-s" type UI element,
            3.2.1.1) Describe the elements looks and semantics based on its visual characteristics and your knowledge about common icons and buttons.(e.g., "circular button with a gear icon," or "square button with an arrow" or "facebook icon").
            3.2.1.2) Use the looks and semantic description generated to infer its functionality. (e.g., circular button with gear icon, for accessing settings)
         3.2.2) For each "icon-p" type UI element,
            3.2.1) Use the text element associated with it (1.2.1) in the UI, to add fine grained functionality description.(e.g. if a button is circular with a toggle icon associated with the text that says "Wi-Fi," it is a "toggle button for turning Wi-Fi on/off.")
            3.2.2) Append the ID of the "text" type element associated with the element to the description.
         3.2.3) For each "picture" type UI element output a one line description of the image the element is a part of or encloses completely.
            If necessary generate context description for the "picture" type element.\n
         3.2.4) For each "text" type element, perform OCR to extract the text enclosed in its bounding box and append it to in the [Content] section of the annotation.
         3.2.5) If the "image" type element is undecipherable or the element is neither an "icon" "button" "text" or "picture" classify it as "irrelevant"
      Note:
      1.) If an icon does not have a bounding box due to imperfect image detection, do not try to generate its description, only focus on the annotated elements.
      2.) Do not give redundant icon descriptions.
Note:
The following things are very important for a successful execution of the task:\n
   1.) Generate the output for all "text" elements first and then the "image" elements.
   2.) Maintain spatial and semantic relationships between UI elements to correctly determine functionality.
   3.) The ID tags associated with "image" type elements are unordered, so avoid sequencing bias when associating IDs in the prompt to IDs annotated in the screenshot.
   4.) Match elements to ID tags using only the color of the ID tag, color of the bounding box, and their spatial proximity.
   5.) The number of elements, bounding boxes, and IDs in the input must match the output exactly.
Output Format:\n
Description: Brief description of the content and functionality of the screenshot.
Annotations:\n
   [ID][x1, y1, x2, y2][text-p][ID of associated "image" element][Spell-corrected content for text type element]\n
   [ID][x1, y1, x2, y2][text-s][Spell-corrected content for text type element, context (if necessary)]\n
   [ID][x1, y1, x2, y2][text-a][Spell-corrected content for text type element][Functionality Description]\n
   [ID][x1, y1, x2, y2][icon-p][ID of associated "text" element][Semantic description][Functionality description]\n
   [ID][x1, y1, x2, y2][icon-s][Semantic and visual description[Functionality description]\n
   [ID][x1, y1, x2, y2][picture][Description of the image and context (if necessary)]\n
   [ID][x1, y1, x2, y2][irrelevant][]\n
   (Repeat for all UI elements)\n
Here are some examples:
```

Figure 6. Prompt for SEED

```
You are a helpful AI assistant. You will be given a screenshot of a User Interface (UI) of a mobile / desktop / web application,
along with an instruction. Your task is to find the correct element from the UI that will lead to a successful
completion of the instruction.\n
Input Format:\n
[ID][x1, y1, x2, y2][text-p][ID of associated "image" element][Content inside the text element]\n
[ID][x1, y1, x2, y2][text-s][Content inside the "text" element]\n
[ID][x1, y1, x2, y2][text-a][Content inside the "text" element][Functionality Description]\n
[ID][x1, y1, x2, y2][icon-p][ID of associated "text" element][Structural and Semantic description of icon][Functionality description]\n
[ID][x1, y1, x2, y2][icon-s][Structural and Semantic description of icon][Functionality description]\n
[ID][x1, y1, x2, y2][picture][Description of the image and context]\n
[ID][x1, y1, x2, y2][irrelevant][]\n
Input Description:\n
1.) Prompt:
   1.1) There are eight types on UI elements annotated onto the screenshot, "text-a" (text actionable), "text-p" (text paired), "text-s" (text standalone), "icon-p" (icon paired), "icon-s" (icon standalone), "picture",
"irrelevant".\n
   1.2) For each element [ID] represents the associated ID tag, [x1, y1, x2, y2] are the endpoint coordinates of the rectangular bounding box associated with the element.\n
   1.3) "text-p": (text paired) are the "text" elements associated with a nearby "icon" type element.
   1.4) "text-a": (text actionable) are the "text" elements associated with a functionality.
   1.5) "text-s": (text standalone) are the "text" elements NOT associated with any nearby "icon" type elements or any functionality.
   1.5) "text-SAM": (text SAM) are the "text" elements NOT detected by OCR but SAM Segmentation algorithm.
   1.6) "icon-p": (icon paired) are the "icon" elements associated with a nearby "text" type element.
   1.7) "icon-s": (icon standalone) are the "icon" elements NOT associated with a nearby "text" type element.
   1.8) "picture":  are the images embedded in the UI.
   1.9) "irrelevant": elements are false positive detections and should be ignored.
2.) Input Screenshot:
   2.1) For "icon", "picture" "text-SAM" and "irrelevant"  type UI elements, each bounding box and its respective ID tag will share the same color, The ID will be at the top-left or the bottom-left corner of the bounding
box.\n
   2.2) For "text-p", "text-a" and "text-s"  type UI elements, the associated bounding box  will be "red" in color, the corresponding ID tag will NOT be annotated onto the image to avoid cluttering. \n

Task:\n
   Using the descriptions of the UI elements provided in the annotation, their classifications, and the screenshot as reference
   correct element along with the reasoning behind their selection, clicking which will lead to the completion of the instruction in a single shot.\n

Note:\n
The following pointers are very important for a successful execution of the task:\n
   1.) The selected element should lead to the completion of the instruction in a single shot.
   2.) The correct element does not necessarily correspond to the element with the strongest string match, lay strong emphasis on the functionality requirement of the instruction.
   3.) Correct semantic, structural, and functional understanding of elements should be established using the descriptors and image provided.
   4.) Preference should always be given to icons and actionable text (e.g. texts associated with input fields, search bars, buttons, drop downs etc.) over static text.
   5.) Pay attention on the action words in the instruction like "Search", "Input" "Select", "Enter" to determine the type of element required for the task.



Output Format:\n
{"id": <ID of the correct element>, "reasoning": <semantic and functional reasoning behind the elements selection>}

Do not output anything else.
"""
```

Figure 7. SoM grounding Prompt for ScreenSpot and VisualWebBench

```
You are an expert at completing instructions on Android phone screens.
You will be presented with two images. The first is the original screenshot. The second is the same screenshot with bounding box annotations.
You will additionally also be provided with a prompt with the descriptions of all the elements tagged on the screen.
1. Input Screenshot Description:
   2.1) For all "icon"/"picture" type UI elements, each bounding box and its respective ID tag will share the same color, The ID will be at the top-left or the bottom-left corner of the bounding box.
   2.2) For all "text" type UI elements, the associated bounding box  will be "red" in color, the corresponding ID tag will NOT be annotated onto the image to avoid cluttering. \n
2. Input Prompt Format:
   [ID][x1, y1, x2, y2][text-p][ID of paired "icon"/"picture" element][Content inside the text element]
   [ID][x1, y1, x2, y2][text-s][Content inside the "text" element]
   [ID][x1, y1, x2, y2][text-a][Content inside the "text" element][Functionality Description]
   [ID][x1, y1, x2, y2][icon-p][ID of paired "text" element][Structural and Semantic description of icon][Functionality description]
   [ID][x1, y1, x2, y2][icon-s][Structural and Semantic description of icon][Functionality description]
   [ID][x1, y1, x2, y2][picture][Description of the image and context]
   [ID][x1, y1, x2, y2][irrelevant][]
3. Annotation Description:
   -s stands for "static" -a stands for "actionable", -p stands for "paired".

4. Task:
   If you decide to click somewhere, you should choose the numeric idx that is the closest to the location you want to click.
   The screenshot are most likely an intermediate step of this instruction, so in most cases there is no need to navigate back home.
   You should decide the action to continue this instruction.
   Here are the available actions:
   {"action_type": "click", "idx": <element_idx chosen from the second screen>, "bboxes": <bbox coordinates of the the chosen element as a singular list>}
   {"action_type": "type", "text": <the text to enter>}
   {"action_type": "navigate_home"}
   {"action_type": "navigate_back"}
   {"action_type": "scroll up"}
   {"action_type": "scroll down"}
   {"action_type": "scroll left"}
   {"action_type": "scroll right"}
   Your final answer must be in the above format.
```

Figure 8. Agentic task following prompt for AITW

You are an expert at completing instructions on websites. You will be given an instruction and two images. The first is the original screenshot.
The second is the same screenshot with some bounding box annotations. Additionally you will also be given a prompt giving details about the annotations on the screenshot. The format for the same is specified below.
1. Input Prompt Format:
   [ID][x1, y1, x2, y2][text-p][ID of associated "image" element][Content inside the text element]
   [ID][x1, y1, x2, y2][text-s][Content inside the "text" element]
   [ID][x1, y1, x2, y2][text-a][Content inside the "text" element][Functionality Description]
   [ID][x1, y1, x2, y2][icon-p][ID of associated "text" element][Structural and Semantic description of icon][Functionality description]
   [ID][x1, y1, x2, y2][icon-s][Structural and Semantic description of icon][Functionality description]
   [ID][x1, y1, x2, y2][picture][Description of the image and context]
   [ID][x1, y1, x2, y2][irrelevant][]
2. Input Screenshot Description:
   2.1) For all icon/picture/irrelevant type UI elements, each bounding box and its respective ID tag will share the same color, The ID will be at the top-left or the bottom-left corner of the bounding box.\n
   2.2) For all "text" type UI elements, the associated bounding box will be "red" in color, the corresponding ID tag will NOT be annotated onto the image to avoid cluttering. \n
3. History:
   You will also be provided a brief summary history of all actions taken so far.
Task:
   Choose the best action to continue the instruction given the present screen. If you decide to click somewhere, Using the 2 UI screenshots and the annotation descriptors in the prompt as reference suggest the top 3 elements (text or icon) in decreasing order of relevance to the instruction, along with the reasoning behind their selection.
   Here are the available actions:
   {"action_type": "CLICK", "idx": <element_idx of chosen element from the annotated screen>, "bbox": <bbox coordinate of chosen element from the annotated screen>}
   {"action_type": "TYPE", "idx": <element_idx oof chosen element from the annotated screen>, "bbox": <bbox coordinate of chosen element from the annotated screen>, "value": <the text to enter> }
Note:
   1. **Important**: The sequence of screenshots are sampled from a pre-executed trajectory, and may not necessarily be a result of the previous action, choose the best action given the current screen.
   2. Use {"action_type": "TYPE"} when a specific value needs to be typed into the input field
   3. Output the answer as string and not json formatted
Output Format:
Action Reasoning:
<Reasoning behind the selection of action>
UI Element Selection Reasoning :
[ID][x1, y1, x2, y2][element type]: <Semantic and Functional reasoning behind the selection of the most relevant UI component>
(Repeat for top 3 elements)
Action:
{"action_type": "CLICK", "idx": <element_idx of chosen element from the annotated screen>, "bbox": <bbox coordinate of chosen element from the annotated screen>}
{"action_type": "TYPE", "idx": <element_idx oof chosen element from the annotated screen>, "bbox": <bbox coordinate of chosen element from the annotated screen>, "value": <the text to enter> }

Figure 9. Agentic task following prompt for Mind2Web