Abstract

This supplementary material includes (1) details of task definitions (§A.1), including a taxonomy of spatial aspects in Table 10 and curated functional programs in Table 11 and 12, (2) setups of the tabletop scene and the shelf scene (§A.2), (3) a discussion on the sim-to-real relevance (§A.3), (4) evaluation details, such as prompting procedures, essential prompts, and evaluation efficiency (§A.4), and (5) details of ESPIRE *assets, including their visualizations and dimensions (§A.5).*

A. Appendix

A.1. Participants of a robotics task

A robotics task, in its simplest form, can be defined by an action and a manipulable object (e.g., '*pick up the book*'). We employ two primitive actions, 'pick' and 'place', to initiate robotics tasks. Keeping the action space simple helps isolate spatial reasoning behaviors, allowing for a focus on their analysis.

To highlight the facets of spatial reasoning and support systematic task design and experimental analysis, we categorize key spatial aspects (S), reference frames (F), and reference objects (O) that characterize spatial reasoning and combine them to define task specifications (see Section 4.1).

- **Reference frames.** Reference frames refer to coordinate systems essential for describing one object in relation to another. They can be made explicit via linguistic specifications, but are usually implicitly conveyed within the context. Following [25], we consider three types of reference frames.
  - **Relative frames** are viewer-centered; for example, '*behind the mirror*' may refer to the space further from the viewer, from the viewer's perspective toward the mirror.
  - **Intrinsic frames** are object-centered; for example, '*behind the mirror*' may indicate the space opposite to the mirror's facing direction, independent of the viewer.
  - **Absolute frames** are defined with respect to fixed global coordinates, such as elevation and altitude (useful for describing *below* and *above*) and cardinal directions (e.g., *north* and *east*), but are only used in a few indoor scenarios.
- **Objects.** ESPIRE contains two primary object types: manipulable and reference objects.
  - **Manipulable objects** are instantiated as cuboid-shaped books (see Table 18). The regular geometries make it easier to verify their final states, facilitating automated evaluation. Moreover, they yield a relatively high likelihood of generating valid grasping/placement poses, without relying on external tools for pose proposal, yet remain sufficiently challenging for 6-DoF tasks.
  - **Reference objects** participate in describing an object in relation to another. In cases where a reference frame is not explicitly specified, the intrinsic frame of the referenced object may be used. Thus, we divide reference objects into *intrinsic-oriented* objects that have a clear front face (e.g., a chair or mirror) and *non-oriented* objects that do not (e.g., a jar or ball). To support fine-grained analysis of spatial reasoning, such as distinguishing units of measure in distance estimation (meter vs. centimeter), we further divide reference objects into *near* and *distant* categories (see Table 16 and 17). Near objects appear on the shelf or tabletop, whereas distant objects are located outside these areas.
- **Spatial aspects.** We group spatial aspects into four broad classes: attributes, distances, relationships, and orientations (see an overview in Table 10). Whenever applicable, we consider both coarse- and fine-grained expressions, such as relative distance and precise distance:
  - **Attributes** primarily refer to intrinsic size attributes (i.e., dimensions) of an object, such as *height*, *length*, *width*, *volume*, and *diameter/radius*. They may be implicitly used to check space fitness and describe object volume, e.g., *a large/small book*.
  - **Distances** describe the proximity between objects. Apart from relative distance descriptions like *nearest*, *farthest*, and *second farthest*, we include precise distance descriptions using different units of measure, e.g., *within 1 meter of the mirror* and *20 centimeters away from the jar.*'
  - **Relationships** primarily describes positional relations, i.e., how one object is positioned relative to another. They can be expressed in diverse ways in natural language, but we consider only the most commonly used basic forms like *left*, *right*, *in front of*, *behind*, *below*, and *above*, and their comparative and superlative forms like *leftmost*, *rightmost*, and *second leftmost*.
  - **Orientations** cover directional expressions, including coarse-grained state descriptions (e.g., *upright* and *at a tilt*) and fine-grained clock positions (e.g., *to your 6 o'clock*) and degrees of a tilt (e.g., *at a 45-degree tilt*).

We note that our definition of the task specification ($C = (S, F, O)$) primarily disentangles the complexity of spatial reasoning over 'Relationships' and 'Orientations', as they rely on a frame of reference, but 'Attributes' and 'Distances' do not. Nonetheless, we use this definition across all four spatial aspects to keep consistent.

A.2. Simulated Environment

We focus on tasks that involve picking up an object from the table and placing it on the shelf (see Figure 3). Though the reverse direction—picking up an object from the shelf and placing it on the table—is also feasible and would increase task diversity, we consider only the former because it

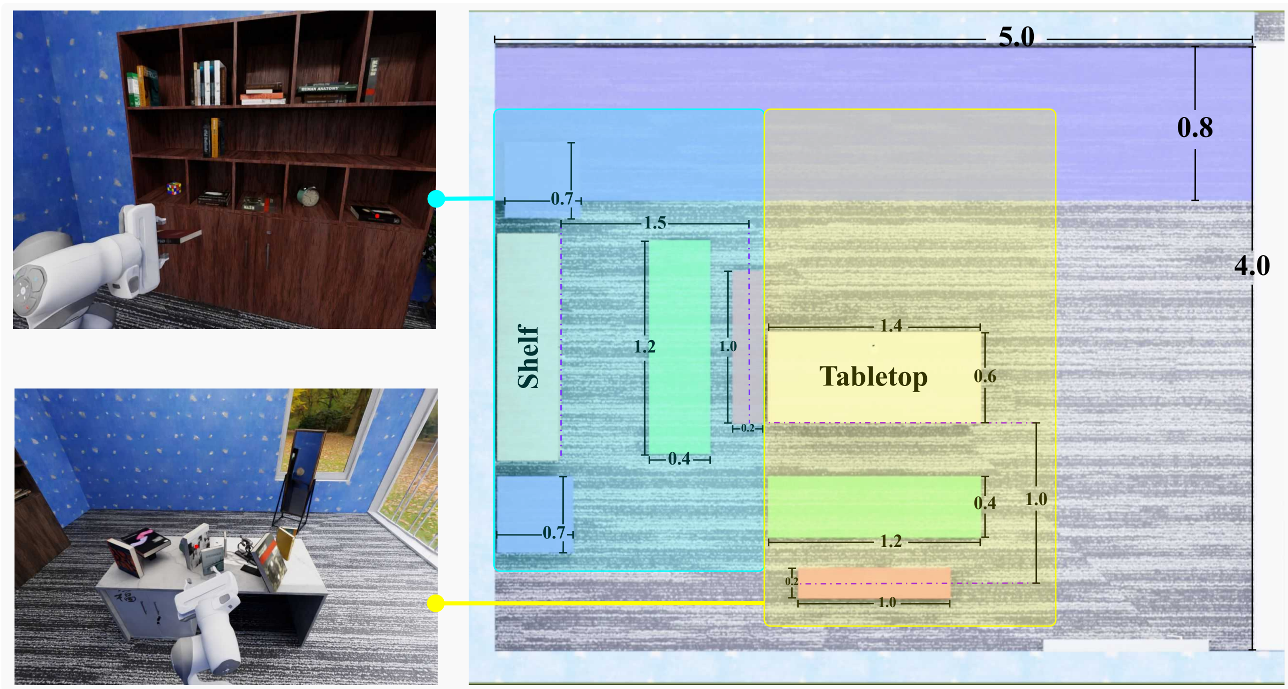


Figure 3. Layouts of the tabletop and shelf scenes within ESPIRE (best viewed in color). The light red region denotes the camera viewpoint sampling area, the light green region indicates where the robot end effector may appear, and the light blue region denotes where distant reference objects are placed. All labeled dimensions are in meters.

alone suffices to cover a diverse range of spatial reasoning scenarios.

Tabletop tasks. The table scene is initialized with manipulable books, support ornaments, and reference ornaments. For books, we consider three different sizes (i.e., *small*, *medium*, and *large*), and three different initial poses (i.e., *standing upright*, *lying flat*, and *at a tilt*). We use a small set of support ornaments to create *flat* and *tilting* poses while ensuring that the books are pickable via 6-DoF pose prediction. Near references are small-sized, appearing on the table (e.g., a picture frame or ceramic jar), whereas distant references are large-sized, located on the floor behind the table (e.g., a floor lamp or cheval mirror). Note that the reference ornaments are carefully selected to cover intrinsic-oriented and non-oriented categories, and the robot itself is an intrinsic-oriented reference, always facing the front of the table. We initialize the tabletop with two random near objects and one random distant object. The degree of clutter in the scene is controlled by varying the number of books on the table, while the overall complexity is driven by instructions and environmental factors like varying poses, lights, and textures.

The global camera looks at a random point on the front edge of the table. We randomly sample its elevation 0.5–1m above the table surface. The elevation of the end-effector is randomly sampled from 0.3–0.5m below the global camera, with pitch, yaw, roll randomly sampled from $[-22.5°, 22.5°]$, $[-22.5°, 22.5°]$, and $[-45°, 45°]$, respectively.

Shelf tasks. The shelf scene contains the same types of objects as in the tabletop scene. Analogously to the tabletop scenes, where books are initialized in random grasping poses, we initialize the shelf with random support ornaments to let books lean against, creating various placement poses. Compared with the tabletop scene, the shelf scene supports reasoning of two additional spatial relationships: *above* and *below*. We locate distant references on the floor, either to the left or to the right of the shelf. The robot always faces the front of the shelf. We control the complexity of shelf tasks by varying shelf layouts, including horizontal panels, grids of slots, and their combinations.

The global camera always faces the shelf center. We randomly sample its elevation 1.2–1.5m above the ground. The elevation of the end-effector is randomly sampled from 0.3–0.5m below the global camera, with pitch, yaw, roll randomly sampled from $[-22.5°, 22.5°]$, $[-22.5°, 22.5°]$,

CVPR #25

and $[-120°, -60°] \cup [60°, 120°]$, respectively.

Definitions of spatial relationships and orientations. In natural language, spatial relationships can exhibit ambiguity due to the reliance on reference frames and contexts. To address this issue, we use a unified definition to assign them unambiguous geometric interpretations. Specifically, under a given reference frame, we use its forward axis to represent the front-facing direction, then *left* and *right* are defined relative to it. The definition of *behind* is, however, more involved, as it depends on the reference frame. Suppose the description 'O_1 is *behind* O_2', when the reference frame is independent of O_2, it is interpreted as: O_1 is further than O_2 along the front-facing direction; when the reference frame is attached to O_2, the meaning changes to: $O1$ is further along the opposite of the front-facing direction.

We account for two fine-grained types of orientation: direction and tilt. Directions are represented using clock positions that provide a granular description relative to a specific reference frame. In this setup, the forward axis is assigned to 12 o'clock, with all other positions mapped relative to this heading. To describe precise tilts, we measure the tilt angle in degrees and define it as the deviation between the global up-axis and the upright axis of the object.

Definitions of 'above' and 'below'. The global up-axis corresponds to the up-direction of an absolute reference frame, defined as the surface normal of the floor in our simulation environment. We also rely on this global up-axis to define spatial relationships like *above* and *below*. Specifically, 'O_1 is *above* O_2' indicates that O_1 lies further along the global up-axis; equivalently, 'O_2 is *below* O_1'. Following the standard convention, we define the tilt angle as the angle between the up axis and the surface normal of an object.

Mitigation of ambiguity. For objects involved in tabletop tasks, we randomly initialize their locations while ensuring that they are spaced at least 5cm apart. Note that due to physical rendering constraints, the final spacing may be smaller than 5cm. We require that at least 20% of the pixels of each object are visible in the global view. For shelf tasks, we require that at least 50% of the pixels of the book in hand are visible. When a target satisfies multiple constraints (e.g., a book can be behind the picture while also being to its left), we select the most salient one for task generation.

Balanced task sampling. To ensure that ESPIRE tasks are approximately uniformly distributed across task families $\mathcal{T}$ and difficulty levels $\mathcal{L} = \{\text{easy}, \text{medium}, \text{hard}\}$, we propose a balanced task sampling strategy (see Algorithm 1). Specifically, we maintain a counter $C_{t,l}$ for each combination of task family $t \in \mathcal{T}$ and difficulty level $l \in \mathcal{L}$. We also record the number of times each scene $s \in \mathcal{S}$ has been attempted so far, denoted by A_s. These counters are used to dynamically adjust the sampling weights (lines 15 and 21).

During task generation, we first select a task family with preference for underrepresented families (lines 12–15). Given the selected task family, we collect all scenes that yield a non-empty answer set and randomly sample one, favoring underrepresented difficulty levels while penalizing scenes that have been repeatedly attempted (lines 16–19). We repeat this process until the desired number of tasks has been generated.

A.3. Sim-to-real relevance

To confirm ESPIRE serves as a reliable proxy for embodied spatial reasoning, we establish the benchmark's validity through the following two lens:

Performance alignment. We evaluated Qwen3-VL (8B/30B/235B), RoboBrain2.0-7B and Gemini2.5-Pro on ESPIRE and the pointing tasks of the natural-image benchmark RefSpatial [59]. We then compute Spearman's rank correlation between the model performance rankings on the two benchmarks. The resulting correlation is 96.4% (with $p = 0.00498$), indicating strong alignment between the two evaluations and suggesting that ESPIRE serves as a high-fidelity proxy for real-world embodied spatial reasoning.

Human study. We conducted a study with five humans to assess environment realism and model alignment (see Section 5.5). First, we observe an average $94.9 \pm 3.4\%$ human success rate across all tasks, suggesting that the simulated scenarios are readily interpretable and solvable by humans.

We further analyze agreement on the ground-truth reference frame across three reference categories: near oriented objects, distant oriented objects, and the table. Specifically, we measure the proportion of examples with unanimous agreement among the five annotators. For examples involving distant oriented references and the table, humans agree on the reference frame in more than 97% of cases, suggesting that the intended frame is clearly interpretable in these settings. However, only 31.03% of the examples involving near oriented references yield unanimous agreement. This lower agreement likely reflects the inherent ambiguity of reasoning with nearby oriented objects, as discussed in Section 5.5.

A.4. Evaluation

Task status checking. After execution, we obtain task status (e.g., failure or success) by checking if the final environment state satisfies the constraints specified in the instruction. (1) *Distance* is measured as the minimum distance between the boundaries of the target and the reference object (which can be a 3D point). The final distance within ± 3 cm of the

Algorithm 1 Balanced Task Sampling

Input: Task families $\mathcal{T}$, scenes $\mathcal{S}$, difficulty levels $\mathcal{L} = \{\text{easy, medium, hard}\}$, the number of tasks per family N_t, the number of tasks in total N_all.
Initialize task counts $\{C_{t,l}\}_{t\in\mathcal{T},\, l\in\mathcal{L}} \leftarrow 0$
Initialize scene attempts $\{A_s\}_{s\in\mathcal{S}} \leftarrow 0$
Initialize the task set $\mathcal{Q} \leftarrow \emptyset$
$N \leftarrow 0$
while $\textsc{EligibleFamily}(\mathcal{T}, \mathcal{S}) \neq \emptyset$ **do**
    if N_all is defined **and** $N \geq N_\text{all}$ **then**
        break
    end if
    $\mathcal{T}_\text{sub} \leftarrow \{t \in \mathcal{T} \mid \sum_l C_{t,l} < N_\text{t},\ \textsc{GetCompatibleScenes}(t, \mathcal{S}) \neq \emptyset\}$
    if $\mathcal{T}_\text{sub} = \emptyset$ **then**
        break
    end if
    for all $t \in \mathcal{T}_\text{sub}$ **do**
        $w_t \leftarrow \dfrac{1}{\sum_l C_{t,l} + 1}$ ▷ Under-sampled task families
    end for
    $t^\star \leftarrow \textsc{WeightedSampling}(\mathcal{T}_\text{sub}, \{w_t\})$
    $\mathcal{S}_{t^\star} \leftarrow \textsc{GetCompatibleScenes}(t^\star, \mathcal{S})$
    for all $s \in \mathcal{S}_{t^\star}$ **do**
        $l \leftarrow \textsc{GetDifficultyLevel}(t^\star, s)$
        $w_s \leftarrow \dfrac{1}{\sum_{t'} C_{t',l} + 1} \cdot \dfrac{1}{(A_s + 1)^2}$ ▷ Under-sampled scenes and difficulty levels
    end for
    $s^\star \leftarrow \textsc{WeightedSampling}(\mathcal{S}_{t^\star}, \{w_s\})$
    $q \leftarrow \textsc{GenerateAnswerSet}(t^\star, s^\star)$
    $A_{s^\star} \leftarrow A_{s^\star} + 1$
    if $|q| > 1$ **then** ▷ Retain only non-trivial tasks
        $l^\star \leftarrow \textsc{GetDifficultyLevel}(s^\star, t^\star)$
        $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(t^\star, s^\star)\}$
        $C_{t^\star,l^\star} \leftarrow C_{t^\star,l^\star} + 1$
        $N \leftarrow N + 1$
    end if
end while
return $\mathcal{Q}$

Algorithm 2 Localization Procedures

(a) Localization w/o Reflection

1: **Input:** Task instruction T, Scene observation O, Maximum trials N
2: **for** $i = 1$ to N **do**
3:     $P \leftarrow \textsc{Predict}(T, O)$
4:     $R \leftarrow \textsc{Evaluate}(P)$
5:     **if** R.success **then**
6:         **break** ▷ Stop if successful
7:     **end if**
8: **end for**

(b) Localization w/ Reflection

1: **Input:** Task instruction T, Scene observation O, Maximum trials N
2: $F, R_0 \leftarrow$ None
3: **for** $i = 1$ to N **do**
4:     $P \leftarrow \textsc{Predict}(T, O, F, R_0)$
5:     $R_1 \leftarrow \textsc{Evaluate}(P)$
6:     **if** R.success **then**
7:         **break**
8:     **end if**
9:     $F \leftarrow \textsc{Reflect}(T, O, R)$
10:     $R_0 \leftarrow R_1$
11: **end for**

expected distance is considered correct in the evaluation. (2) *Orientation and Relationship* are determined by checking if the center of the target lies in the target area defined by a reference frame. The final tilt angle within $\pm 10^\circ$ of the expected angle is considered correct in the evaluation.

Algorithms. We illustrate the evaluation procedures used for localization in Algorithm 2a (w/o reflection) and Algorithm 2b (w/ reflection). Reflection is performed following a localization failure (line 9 of Algorithm 2b). The generated reflection tokens are added to the inputs to the next iteration (line 4 of Algorithm 2b). The evaluation procedures for execution are illustrated in Algorithm 3a (w/o reflection) and Algorithm 3b (w/ reflection). They are similar to those used for localization except that the reflection for execution relies on an additional view of the failure state (line 9–12 of Algorithm 3b).

Prompts. We provide our customized prompts for *pick* tasks with Qwen3-VL [2]. Figures 4a, 4b, 5a, and 5b show

Algorithm 3 Execution Procedures

(a) Execution w/o Reflection

Input: Task configuration T, Scene observation O, Maximum trials N
for $i = 1$ to N **do**
  $P \leftarrow \text{PREDICT}(T, O)$
  $R \leftarrow \text{EVALUATE}(P)$
  if R.success **then**
    break ▷ Stop if task is done
  end if
  $O \leftarrow \text{GETOBSERVATION}()$
end for

(b) Execution w/ Reflection

Input: Task configuration T, Initial observation O, Maximum trials N
$F, O_0, R_0 \leftarrow$ None
for $i = 1$ to N **do**
  $P \leftarrow \text{PREDICT}(T, O, F, O_0, R_0)$
  $R_1 \leftarrow \text{EVALUATE}(P)$
  if R_1.success **then**
    break
  end if
  $O_0 \leftarrow O$
  $R_0 \leftarrow R_1$
  $O \leftarrow \text{GETOBSERVATION}()$
  $F \leftarrow \text{REFLECT}(T, O, O_0, R_0)$
end for

the prompts used for localization, execution, localization w/ reflection, and rotation, respectively. They are different across VLMs primarily in the output format, e.g., Gemini2.5-Pro [44] outputs point coordinates in $[y, x]$ while others in $[x, y]$. The differences in the prompts for *pick* and *place* tasks arise primarily in the task descriptions. For example, a localization instruction for *place* tasks could be: '*Given a scene image and a textual description specifying the placement conditions for a book currently held by a robot gripper, you are required to determine the exact placement location in the image.*'

Evaluation Time. We break down the evaluation time along the evaluation stage (see Table 9). The results are averaged across all tasks and attempts. We record the time taken until a successful attempt is achieved. During execution, the environment updates after each move and in each observation query, so we also report the average number of environment updates over successful tasks. Compared to the model inference time in localization and execution, the environment update is quite quick, i.e., it takes an average of 11.65 seconds per update. Models with reflection enabled generally take longer because they require additional API calls. A higher number of environment updates indicates more execution attempts. For example, RoboBrain2.0-7B [38] not only requires the largest number of environment updates but also achieves the lowest success rate, suggesting its weaker capability in execution.

Running Examples. We provide running examples with Qwen3-VL-235B-A22B [2]. Figure 6 shows an excerpt of the localization (w/ reflection) logs of a *pick* task, and Figure 8 shows an example run on a *place* task without reflection.

Figure 9 shows an excerpt of the execution (w/o reflection) logs of a *pick* task. Note that, in this example, the model also needs to predict rotations for the *pitch* axis. Figure 10 presents an excerpt of the execution (w/ reflection) logs of a *place* task. In this task, the model must predict a goal position for the end-effector. Interestingly, the model fails in the first attempt but succeeds by leveraging reflection in the second attempt.

Qualitative analysis. We further conduct a qualitative analysis, including both successful runs and failed runs in Table 8.

A.5. Asset Visualization

We visualize primary assets of ESPIRE, including near reference objects in Table 16, distant reference objects in Table 17, tables in Table 13, shelf layouts in Table 14, shelf textures in Table 15, manipulable books in Table 18, and support ornaments in Table 19.

Table 8. Qualitative analysis. We categorize intermediate executions into following six types: T1 denotes a grasp-favorable viewpoint; T2 denotes a grasp-infeasible viewpoint; T3 denotes manipulator occlusion; T4 denotes object occlusion; T5 denotes unrecognizable target; and T6 denotes physically-achievable execution.

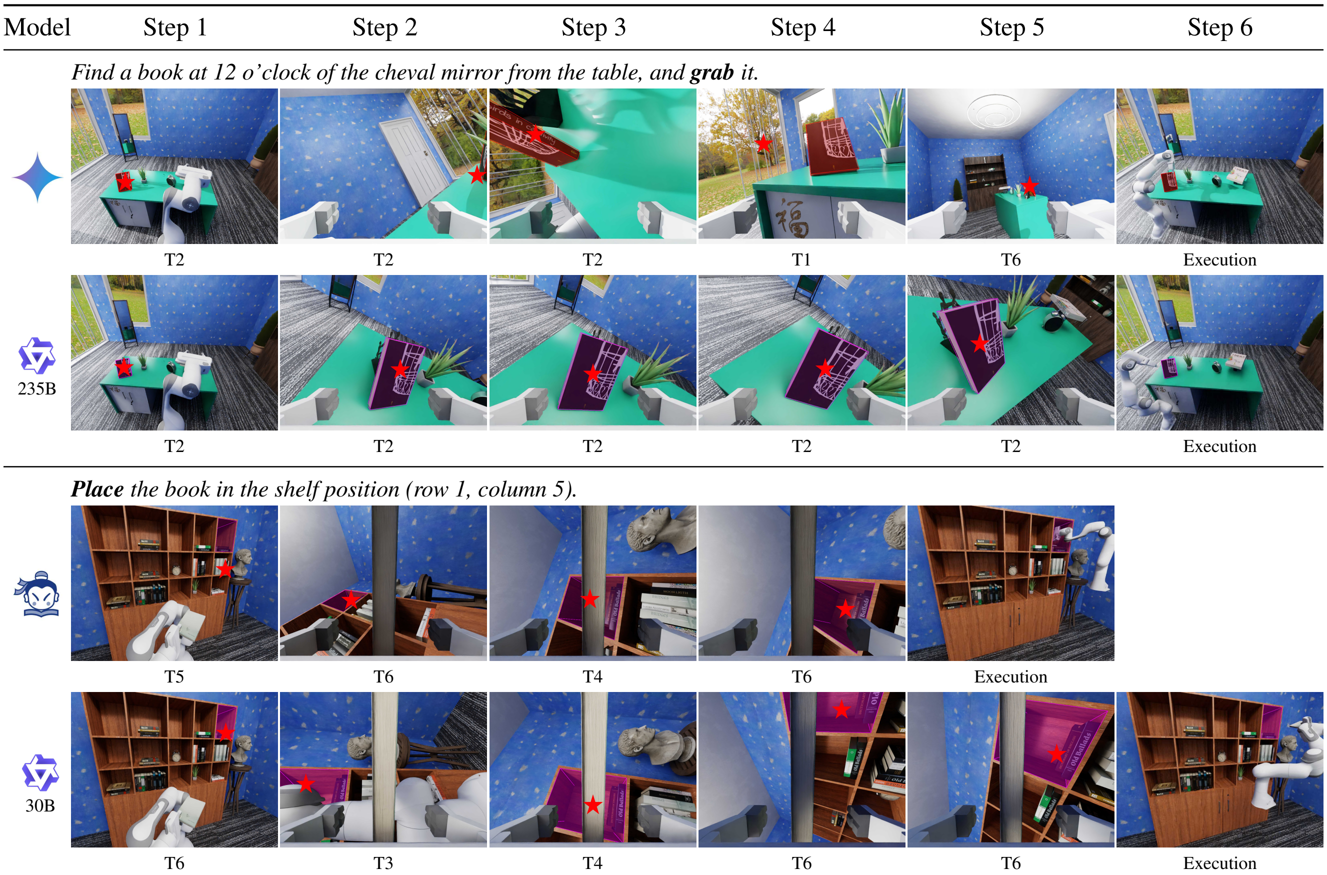


Table 9. Breakdowns of evaluation time in seconds (s).

Models	Pick				Place			
	localization (s)	execution (s)	# update	success (%)	localization (s)	execution (s)	# update	success (%)
w/o reflection								
Gemini2.5-Pro	22.37	40.53	6.86	34.06	33.00	41.15	9.01	15.70
InternVL3-78B	14.44	32.21	6.92	17.26	19.06	34.04	7.59	9.67
RoboBrain2.0-7B	14.17	27.56	9.07	10.87	19.08	30.83	9.36	8.64
Qwen3-VL-8B	15.44	34.87	6.73	29.32	19.48	35.86	7.83	12.41
Qwen3-VL-30B-A3B	12.50	30.61	6.86	32.15	17.38	32.59	7.39	20.00
Qwen3-VL-235B-A22B	16.08	34.49	7.38	26.76	20.40	37.55	7.64	19.34
w/ reflection								
Qwen3-VL-8B	23.94	44.88	8.58	15.07	31.22	47.58	9.31	6.67
Qwen3-VL-30B-A3B	21.51	44.70	8.37	17.08	27.99	49.29	8.13	13.80
Qwen3-VL-235B-A22B	33.23	66.83	8.12	23.20	44.03	68.58	8.53	15.40

You are an expert Visual-Language Assistant specialized in embodied pointing tasks. Note that any occurrence of "You" denotes the first-person camera viewpoint of the RGB image (i.e., the perceptual perspective). It does NOT refer to any object, robot, or robotic end-effector visible in the image.
Task: Identify the object in an RGB image according to a detailed textual description and provide precise coordinates for the object that match the input conditions.
Inputs: 1. A text description specifying the book that needs to be located, including all required attributes and conditions. 2. An RGB image representing the scene, captured from your perceptual viewpoint.
Output: Locate the target object with a point, report its point coordinate in JSON format like this: "point_2d": [x, y]

(a) Localization in *pick* tasks.

You are an expert Visual-Language Assistant specialized for embodied robot-arm manipulation.
Task: Determine the target position of the robotic arm's end-effector to grasp a specific book, where the book is marked with a colored bounding box. Choose a grasp point that is easily reachable and suitable for the robot's end-effector, avoiding edges or positions that are obstructed or unstable. The grasp point should allow the robot to securely pick up the book without collisions or slippage.
Note: If the predicted target point lies on the target book, the robot will attempt to directly grasp the book at that location. If the predicted point lies on another object or a free-space region, the robot will move its end-effector to the indicated position.
Inputs: 1. An RGB image representing the scene, where the target book is marked with a colored bounding box. 2. A textual instruction that specifies how the robot should grasp the book.
Output: Locate the target position with a point, report its point coordinate in JSON format like this: "point_2d": [x, y]

(b) Execution in *pick* tasks.

Figure 4. Example prompts with Qwen3-VL (continued).

CVPR
#25

You are an expert Visual-Language Assistant specialized in reflective analysis of embodied pointing tasks.
Task: Analyze the previous prediction error in the image grounding task. The incorrect predicted point has been marked with a red circle on the RGB image. Reflect on why the model's prediction failed and explain what visual or textual cues were misunderstood or ignored.
Inputs: 1. The original textual description specifying the target object and its required attributes or conditions. 2. The same RGB image used in the grounding task, now containing the red circle marking the incorrect prediction.
Output: Provide a detailed textual reflection explaining: - The likely cause of the failure (e.g., visual confusion, incorrect attribute matching, spatial misinterpretation, etc.). - The correct reasoning process that should have been applied. - Suggestions for how to improve the next prediction.

(a) Localization w/ reflection in *pick* tasks.

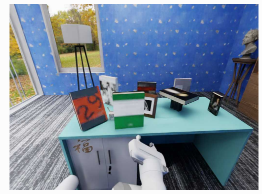
You are an expert Visual-Language Assistant specialized for embodied robot-arm manipulation and precise grasp orientation control.
Task: Given an RGB image with a colored bounding box marking the target book and a red dot indicating the moving target position, together with a textual instruction specifying the book attributes and required rotation axes (pitch, yaw, roll), predict the end-effector's **rotation offsets** relative to its current orientation. All rotations are defined in the **target book's local coordinate frame**.
Inputs: 1. RGB image: - Target book marked by a colored bounding box. - Moving target position marked by a red dot. 2. Text instruction describing the target book and specifying which rotation axes to output.
Reference Frame: Rotations follow the book's intrinsic axes (forward–backward, left–right, up–down): - **Pitch:** tilt up/down. - **Yaw:** swing left/right. - **Roll:** roll clockwise/counterclockwise. Positive/negative follow standard right-hand conventions.
Output: Return the predicted rotation offsets (degrees) in JSON: “‘json “rotation”: [pitch, yaw, roll]

(b) Rotation prediction in *pick* tasks, summarized by GPT-5.1 for the demonstration purpose.

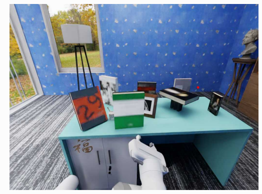
Figure 5. Example prompts with Qwen3-VL.

CVPR #25

ENV: {task_instruction: from the items on the table, find a book more than 120cm away from you, and fetch it. category: pick}

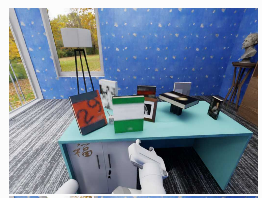


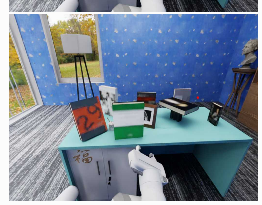
VLM: {system_prompt: [see Figure 4a], user: from the items on the table, find a book more than 120cm away from you, and fetch it. assistant: {point_2d: [757, 448], label: book more than 120cm away from you}}



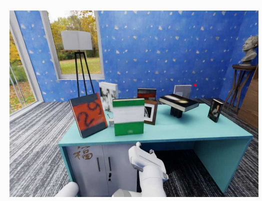
ENV: {moved: false, success: false}

VLM: {system_prompt: [see Figure 5a], user: The previous visual grounding attempt for the [task_instruction] was incorrect. The red circles in the image mark the wrong grounding regions. Please carefully analyze the image and the instruction again, understand why the previous prediction was wrong, and describe what should be corrected in the next attempt. assistant: The previous prediction was incorrect because it failed to properly interpret the spatial constraint in the instruction... Why the Previous Prediction Was Wrong: 1. Misinterpretation of Spatial Distance 2. Focus on Object Type Only... What Should Be Corrected in the Next Attempt: 1. Incorporate Spatial Reasoning 2. Re-evaluate All Books on the Table...}





VLM: {system_prompt: [see Figure 4b], user: Here is a reflection on the previous failed attempt and some suggestions: [reflection]. Now, complete this task: from the items..., assistant: {point_2d: [745, 400]}}



ENV: {moved: false, success: false}

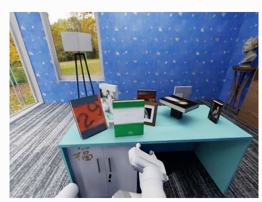
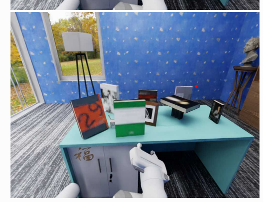


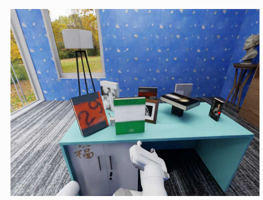
Figure 6. Pick localization example with reflection, where the prompt simplified for better demonstration.

CVPR
#25

VLM: {system_prompt: [see Figure 5a], user: The previous visual grounding attempt for the [task_instruction] was incorrect. The red circles in the image mark the wrong grounding regions. Please carefully analyze the image and the instruction again, understand why the previous prediction was wrong, and describe what should be corrected in the next attempt. assistant: The previous prediction was incorrect because it focused on a region on the wall... Why the Error Occurred: - Misinterpretation of “items on the table” - Failure to filter by object class... What Should Be Corrected in the Next Attempt: 1. Identify all books on the table 2. Estimate distance from the agent (robot arm)...}



VLM: {system_prompt: [see Figure 4b], user: Here is a reflection on the previous failed attempt and some suggestions: [reflection]. Now, complete this task: from the items..., assistant: {point_2d: [835, 555]}}



ENV: {moved: false, success: true}

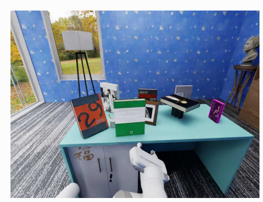
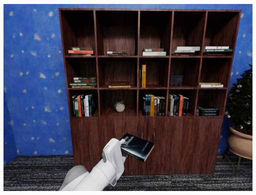
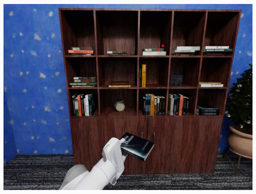


Figure 7. Pick localization example (subsequent attempts and successful execution).

ENV: {task_instruction: position the book in row 1, column 3 of the shelf. category: place}



VLM: {system_prompt: [see Figure 4a], user: position the book in row 1, column 3 of the shelf. assistant: {point_2d: [640, 224]}}



ENV: {moved: false, success: true}

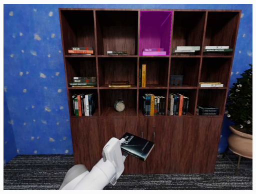
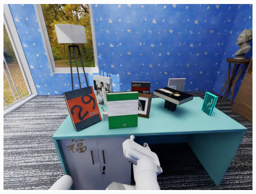
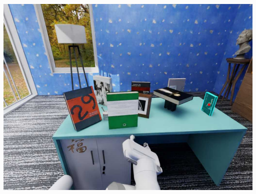


Figure 8. Place localization example without reflection, where the prompt simplified for better demonstration.

ENV: {task_instruction: from the items on the table, find a book more than 120cm away from you, and fetch it. category: pick}



VLM: {system_prompt: [see Figure 4b], user: reach for the book that needs to be grabbed; this book is marked by a bounding box of the red color. assistant: {point_2d: [825, 541], label: target book}} {system_prompt: [see Figure 5b], user: [task_instruction] + The rotation delta(s) you should predict: pitch, assistant: {rotation: [5.0, null, null]}}



ENV: {moved: true, success: true}

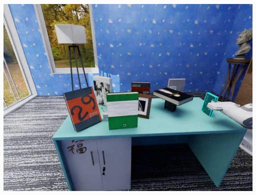
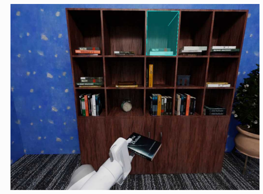
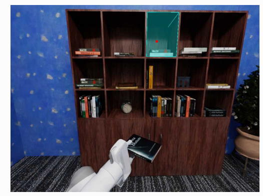


Figure 9. Pick execution example without reflection, where the prompt simplified for better demonstration.

ENV: {task_instruction: position the book in row 1, column 3 of the shelf. category: place}

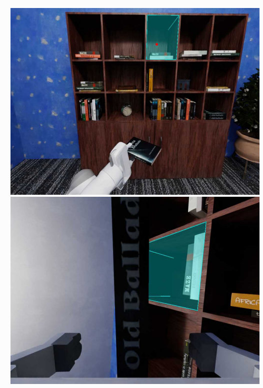


VLM: {system_prompt: [see Figure 4b], user: Reach for the location where the book in your hand needs to be placed. The book in your hand should be placed in the vertical manner, and the placement area is marked by a bounding box in the cyan color. assistant: {point_2d: [587, 194]}}



ENV: {moved: true, success: false}

VLM: {system_prompt: [see Figure 5a], user: The previous visual moving attempt for the [task_instruction] was incorrect. The red circles in the image mark the wrong moving regions. Please carefully analyze the image and the instruction again, understand why the previous prediction was wrong, and describe what should be corrected in the next attempt. assistant: The failure in the previous attempt stems from a misinterpretation of the spatial relationship between the cyan bounding box and the robot's reachable workspace... Likely Cause of Failure: Target Point Misplacement... Kinematic limits... Suggestions for Improving the Next Prediction: Focus on the shelf surface, not the back wall... Prioritize reachable points...}



VLM: {system_prompt: [see Figure 4b], user: Here is a reflection on the previous failed attempt and some suggestions: [reflection]. Now, complete this task: Reach for the location ..., assistant: {point_2d: [684, 418]}}

ENV: {moved: true, success: true}

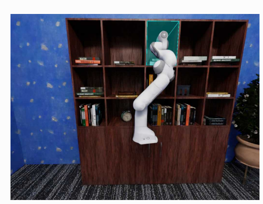


Figure 10. Place execution example with reflection, where the prompt simplified for better demonstration.

Table 10. Spatial aspects of varying granularities. `nil` indicates no input parameters are required. Note that *left*, *right*, *front*, *behind*, *above*, and *below* are reloaded as *directional* relations in 'Orientation' (*cf. positional* relations in 'Relationship').

Spatial Aspect	Granularity	Type	Input	Example Instruction
Attribute	Coarse	Small	nil	*take a small book from the table*
		Medium	nil	*take a medium-sized book*
		Large	nil	*take a large book*
		Empty	nil	*place the book in an empty slot*
		NonEmpty	nil	*place the book in a partly occupied slot*
		Emptiest	nil	*place the book in the emptiest slot*
	Fine	Height	(#,)	*take a book around 20 centimeters high*
		Width	(#,)	*place the book in a slot around 45 centimeters wide*
		Index1D	(#,)	*place the book at row 2 of the shelf*
		Index2D	(#,#)	*place the book at row 2, column 3 of the shelf*
Distance	Coarse	Closest	nil	*take a book among the books closest to you*
		Farthest	nil	*take a book among the books farthest from you*
		LessThan	(#,)	*place the book in a slot within 1.5 meters of you*
		MoreThan	(#,)	*take a book more than 1.5 meters away from you*
	Fine	RankClosest	(#,)	*take a book among the books second closest to you*
		RankFarthest	(#,)	*take a book among the books second farthest from you*
		EqualTo	(#,)	*take a book about 1.5 meters away from you*
		Range	(#,#)	*take a book 1.5 to 2 meters away from you*
Relationship	Coarse	Left	nil	*take a book on the left of the table*
		Right	nil	*place the book in a slot on the left of the shelf*
		Front	nil	*take a book at the front of the table*
		Behind	nil	*take a book at the back of the table*
		Upper	nil	*place the book in a slot in the upper part of the shelf*
		Lower	nil	*place the book in a slot in the lower part of the shelf*
		LeftMost	nil	*take the leftmost book from the table*
		RightMost	nil	*place the the book in a leftmost slot on the shelf*
	Fine	RankLeftMost	(#,)	*take the second leftmost book on the table*
		RankRightMost	(#,)	*place the book in a second rightmost slot on the shelf*
		Between	(#,#)	*place the book between the alarm clock and the succulents*
Orientation	Coarse	Flat	nil	*take a flat-lying book from the table*
		Vertical	nil	*place the book upright on the shelf*
		Tilted	nil	*place the book at a tilt on the shelf*
		Left	nil	*take a book to your left*
		Right	nil	*place the book in a slot to your right*
		Front	nil	*place the book in front of the teddy bear*
		Behind	nil	*take a book behind the picture frame*
		Above	nil	*place the book in a slot above the picture frame*
		Below	nil	*place the book in a slot below the picture frame*
	Fine	DirectLeft	nil	*place the book immediately to the left of the alarm clock*
		DirectRight	nil	*place the book immediately to the right of the succulents*
		DirectAbove	nil	*place the book in a slot directly above the alarm clock*
		DirectBelow	nil	*place the book in a slot directly below the picture frame*
		ClockPosition	(#,)	*place the book in a slot to your 6 o'clock*
		TiltDegree	(#,)	*place the book at a tilt angle of about 30 degrees*

Table 11. Example instruction families of *pick* tasks. The outermost `pick(·)` is discarded for simplicity. `unique(·)` ensures a unique item from the input set. `TABLE` returns all items in the tabletop scene.

S	F	O	R	I	Example Program
Attribute			Small Large Medium		filterAttr\$R(filterBook(TABLE))
			Height Width	*float*	filterAttr\$R(I, filterBook(TABLE))
Distance		*viewer* *distant obj.* *near obj.*	RankClosest RankFarthest	*int*	filterDist\$R(I, filterBook(TABLE), O)
			LessThan MoreThan EqualTo Range	*list*	filterDist\$R(I, filterBook(TABLE), O)
Relationship	intrinsic	*table*	Left Right Front Behind		filterRel\$R(filterBook(TABLE), O)
			RankLeftMost RankRightMost	*int*	filterRel\$R(I, filterBook(TABLE), O)
			Between	*list*	filterRel\$R(filterBook(TABLE), filter(I_1, TABLE), filter(I_2, TABLE))
	relative	*viewer*	Left Right		filterRel\$R(filterBook(TABLE), O)
			RankLeftMost RankRightMost	*int*	filterRel\$R(I, filterBook(TABLE), O)
Orientation	intrinsic	*viewer* *oriented*	Left Right Front Behind		filterOri\$R(filterBook(TABLE), O)
			RankLeftMost RankRightMost	*int*	filterOri\$R(I, filterBook(TABLE), O)
			Flat Vertical Tilted		filterOri\$R(filterBook(TABLE))
			ClockPosition	*int*	filterOri\$R(I, filterBook(TABLE), O)
			TiltDegree	*float*	filterOri\$R(I, filterBook(TABLE), O)
	relative	*viewer* *non-oriented*	Left Right Front Behind		filterOri\$R(filterBook(TABLE), O)
			RankLeftMost RankRightMost	*int*	filterOri\$R(I, filterBook(TABLE), O)
			ClockPosition	*int*	filterOri\$R(I, filterBook(TABLE), O)
			TiltDegree	*float*	filterOri\$R(I, filterBook(TABLE), O)

Table 12. Example instruction families of *place* tasks. The outermost `place(·)` is discarded for simplicity. `unique(·)` ensures a unique item from the input set. `SHELF` returns all shelf-scene items.

S	F	O	R	I	Example Program
Attribute			Small Large Medium		filterAttr\$R(filterSlot(SHELF))
			Height Width	*float*	filterAttr\$R(I, filterSlot(SHELF))
Distance		*viewer* *distant obj.* *near obj.*	RankClosest RankFarthest	*int*	filterDist\$R(I, filterSlot(SHELF), O)
			LessThan MoreThan EqualTo Range	*list*	filterDist\$R(I, filterSlot(SHELF), O) filterDist\$R(I, filterSpace(SHELF), O)
Relationship	intrinsic	*shelf*	Left Right Upper Lower		filterRel\$R(filterSlot(SHELF), O) filterRel\$R(filterSpace(SHELF), O)
			RankLeftMost RankRightMost	*int*	filterRel\$R(I, filterSlot(SHELF), O)
			Between	*list*	filterRel\$R(filterSpace(SHELF), filter(I_1, SHELF), filter(I_2, SHELF))
	relative	*viewer*	Left Right		filterRel\$R(filterSlot(SHELF), O) filterRel\$R(filterSpace(SHELF), O)
			RankLeftMost RankRightMost	*int*	filterRel\$R(I, filterSlot(SHELF), O)
Orientation	intrinsic	*viewer* *oriented*	Left Right Front Behind		filterOri\$R(filterSlot(SHELF), O) filterOri\$R(filterSpace(SHELF), O)
			RankLeftMost RankRightMost	*int*	filterOri\$R(I, filterSlot(SHELF), O)
			ClockPosition	*int*	filterOri\$R(I, filterSpace(SHELF), O)
	relative	*viewer* *non-oriented*	Left Right Front Behind		filterOri\$R(filterSlot(SHELF), O) filterOri\$R(filterSpace(SHELF), O)
			RankLeftMost RankRightMost	*int*	filterOri\$R(I, filterSlot(SHELF), O)
			ClockPosition	*int*	filterOri\$R(I, filterSpace(SHELF), O)
	absolute	*distant obj.* *near obj.*	Flat Vertical Tilted TiltDegree	*float*	placeOri\$R(I, unique(filterSlot(SHELF))) placeOri\$R(I, unique(filterSpace(SHELF)))
			Above Below		filterOri\$R(filterSlot(SHELF), O) filterOri\$R(filterSpace(SHELF), O)

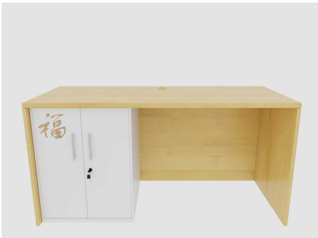


Table 1

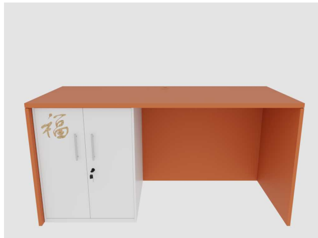



Table 2

Table 3

Table 4

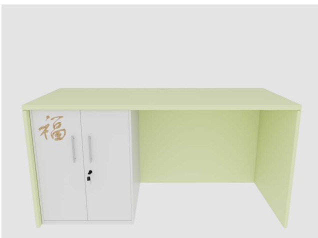



Table 5

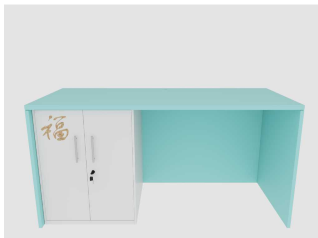



Table 6

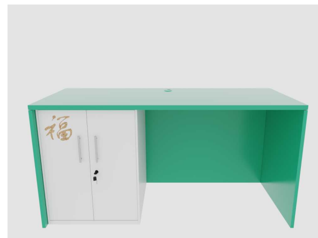



Table 7

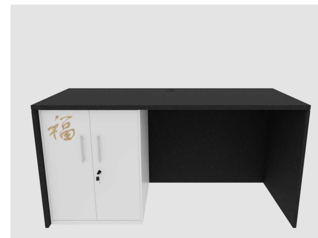



Table 8

Table 13. Tables with different colors and textures.

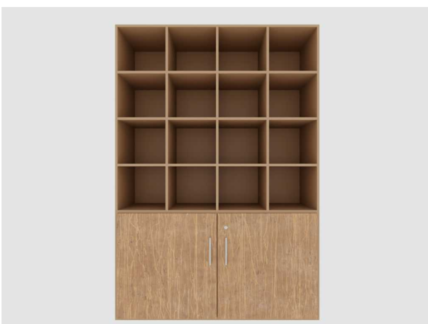
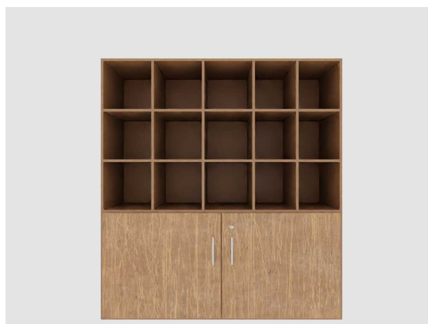
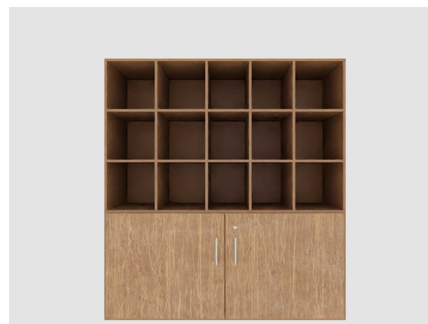
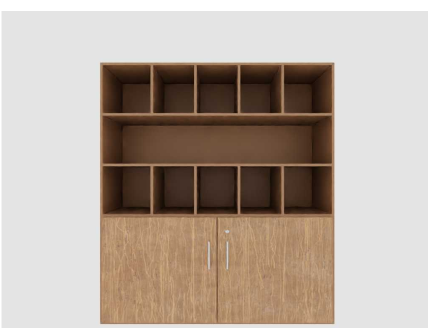
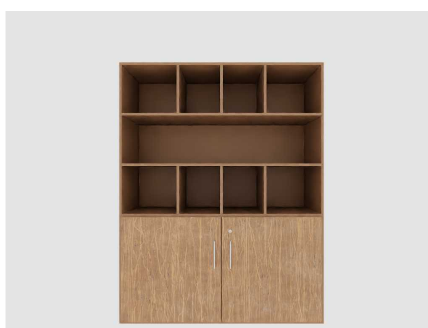
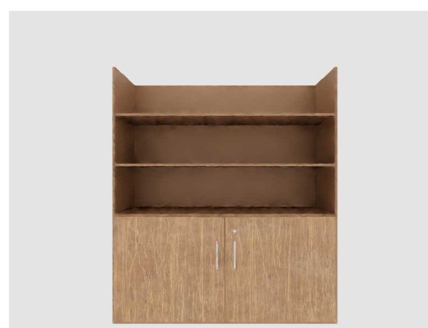

Shelf 1

Shelf 2

Shelf 3

Shelf 4

Shelf 5

Shelf 6

Table 14. Shelf with different layouts.

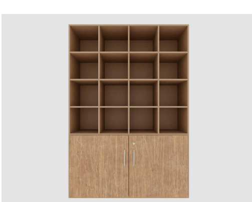
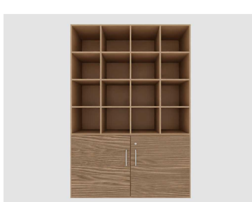
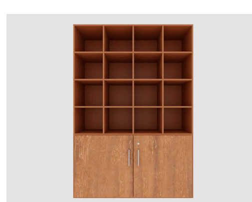
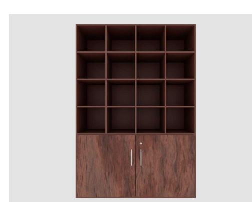
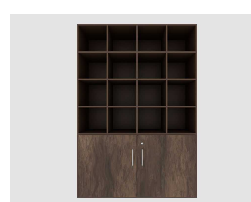
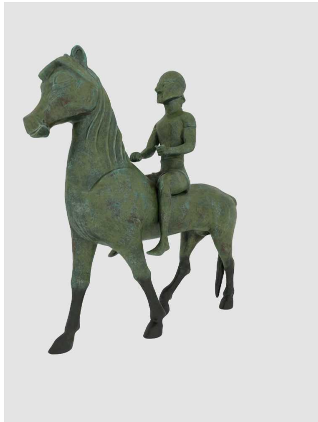
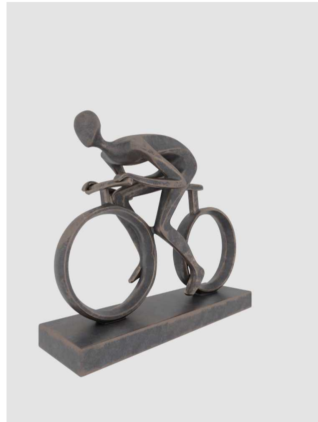
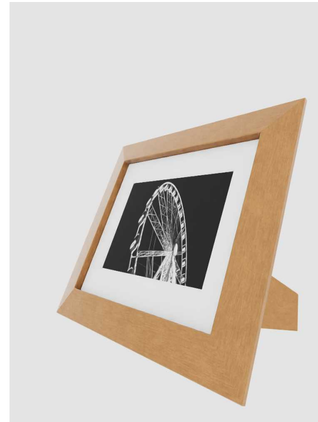






Table 15. Shelf with different textures.

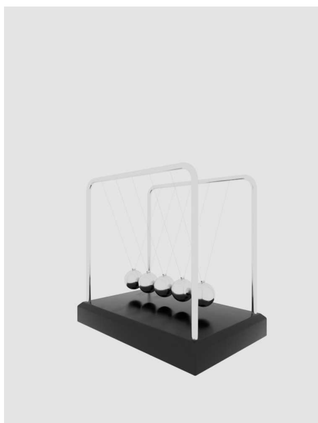

Alarm Clock

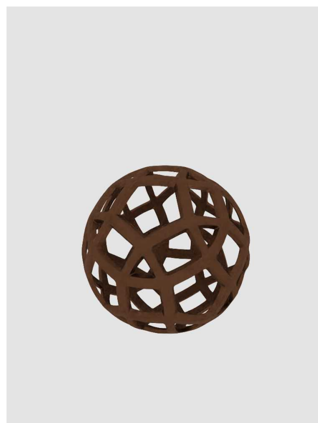

Armento Rider


Bicycle Sculpture

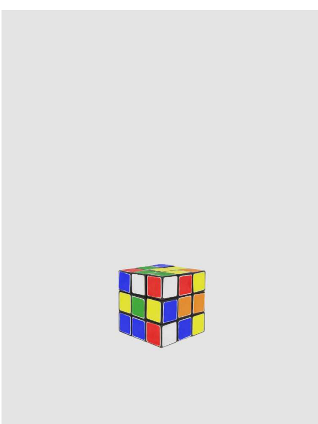

Picture Frame

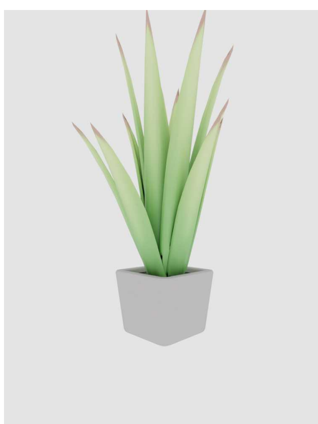

Teddy Bear

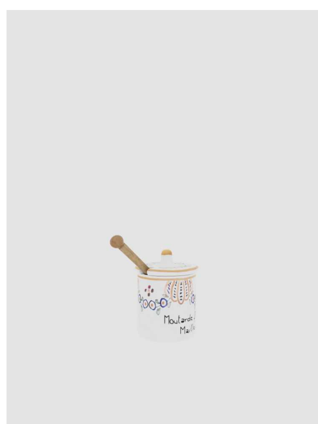

Newton’s Cradle


Geosphere Sculpture


Pillar Bookend


Rubik’s Cube

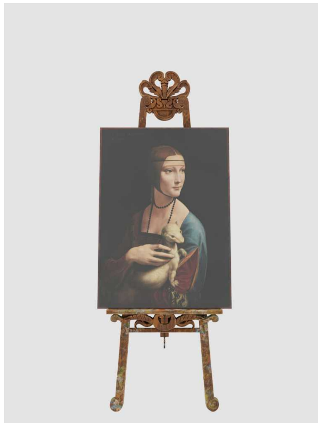

Succulents

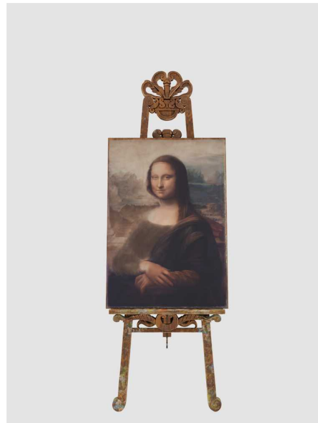

Ceramic Jar

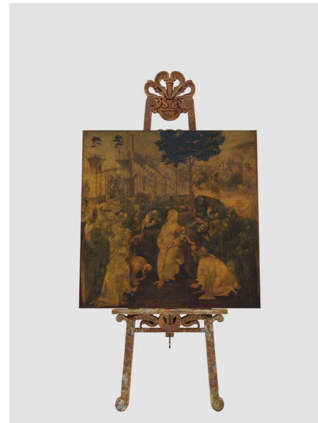

Pagoda Statue

Table 16. Near reference objects (w/ and w/o an intrinsic frame of reference) appear on the table or shelf.


Cheval Mirror


Lady with an Ermine

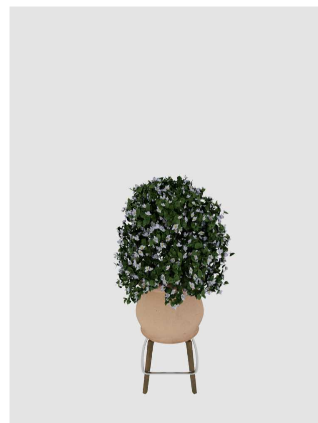

Mona Lisa


Adoration of the Magi

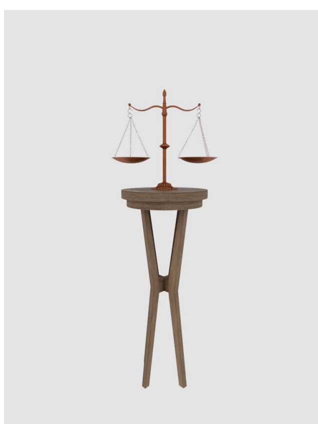

Floor lamp 1

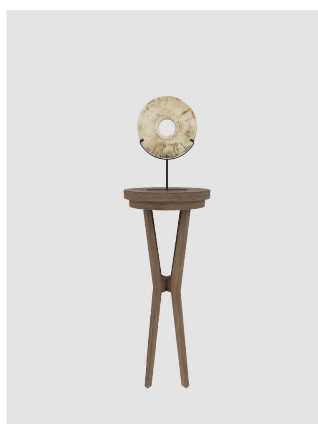

Floor lamp 2


Magnolia sieboldii


Philadelphus shrub


Juniperus communis


Copper scale


Decorative disk

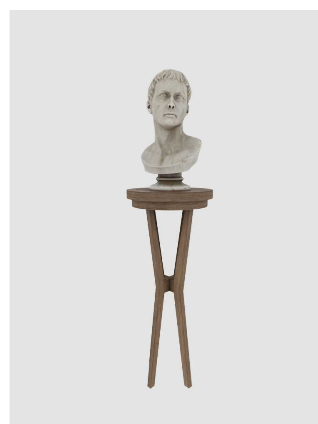

Marble bust

Table 17. Distant reference objects (w/ and w/o an intrinsic frame of reference) appear behind the table or besides the shelf.

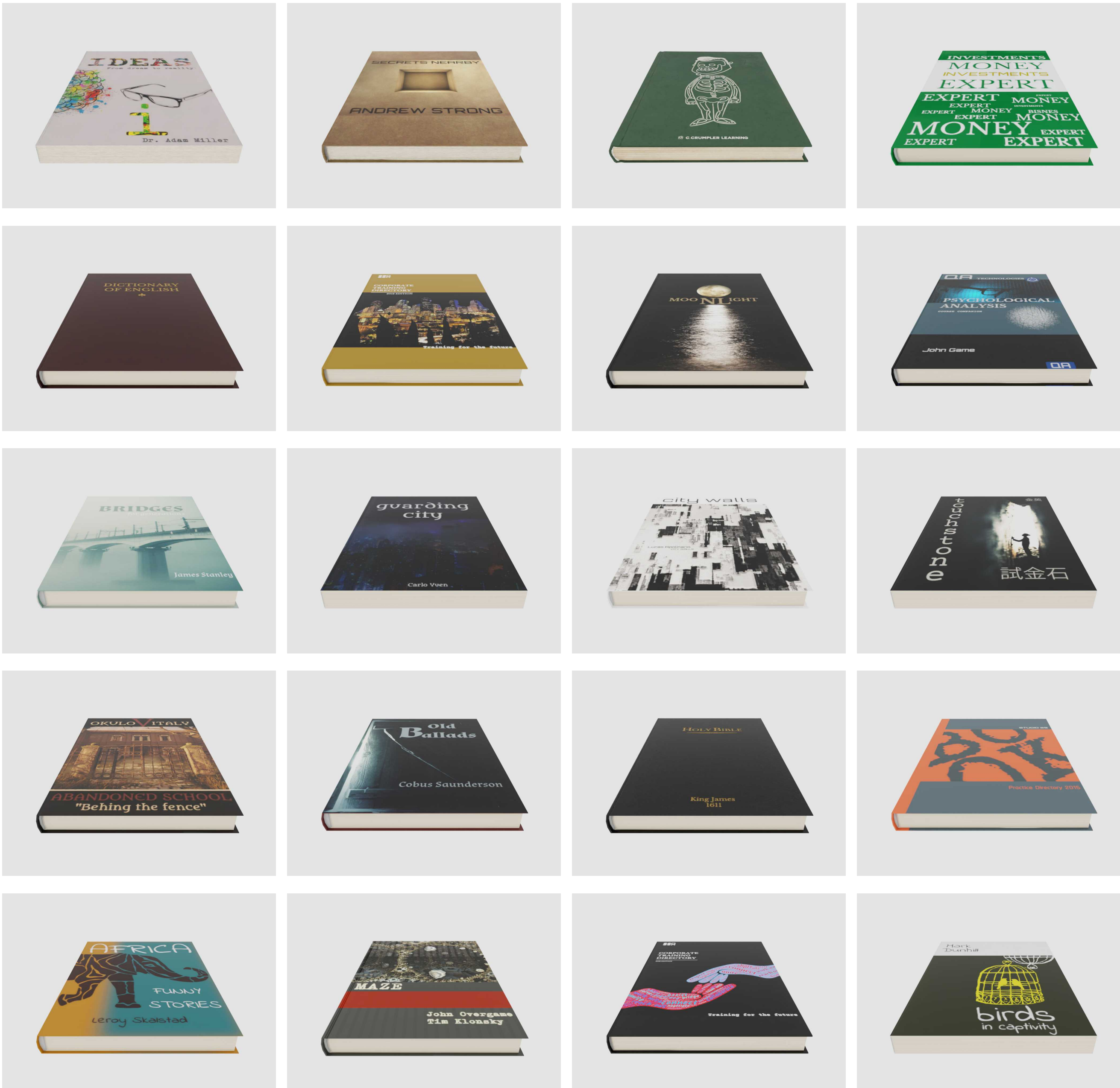


Table 18. Manipulable books. They will be auto-scaled to match three pre-defined sizes: small, medium, and large.

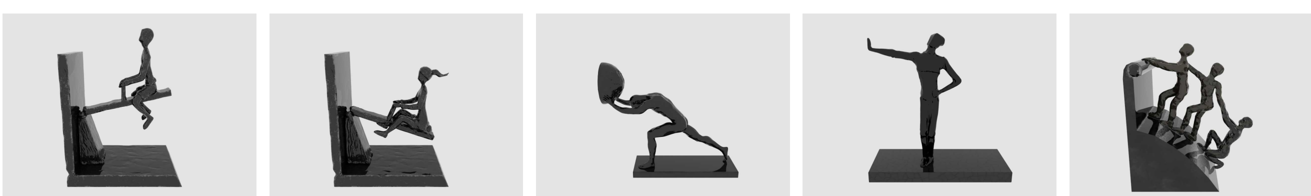


Table 19. Different bookends are used to help create tilting poses of books.

Table 20. Attributes of ESPIRE assets. Oriented assets have an intrinsic frame while non-oriented assets do not.

Name	Type	Oriented	L (cm)	W (cm)	H (cm)
Alarm clock	near	✓	7	13	17
Armento Rider	near	✓	24	7	24
Bicycle sculpture	near	✓	21	8	18
Picture frame	near	✓	13	22	18
Teddy bear	near	✓	20	23	25
Newton's cradle	near	✗	10	15	14
Geosphere sculpture	near	✗	15	15	15
Pillar bookend	near	✗	7	16	13
Rubik's cube	near	✗	6	6	6
Succulents	near	✗	17	15	29
Ceramic jar	near	✗	6	6	8
Pagoda statue	near	✗	13	14	21
Cheval mirror	distant	✓	40	42	43
Lady with an Ermine	distant	✓	62	59	52
Mona Lisa	distant	✓	62	54	52
Adoration of the Magi	distant	✓	62	79	52
Floor lamp 1	distant	✗	48	48	59
Floor lamp 2	distant	✗	51	51	60
Magnolia sieboldii	distant	✗	61	65	51
Philadelphus shrub	distant	✗	63	61	06
Juniperus communis	distant	✗	46	45	34
Copper scale	distant	✗	52	61	46
Decorative disk	distant	✓	52	52	43
Marble bust	distant	✓	52	52	53
Table $1-8$	table	✓	60	140	70
Shelf 1	shelf	✓	45	149	215
Shelf 2	shelf	✓	45	176	190
Shelf 3	shelf	✓	45	176	190
Shelf 4	shelf	✓	45	171	190
Shelf 5	shelf	✓	45	149	190
Shelf 6	shelf	✓	45	164	187
Book-small	book	✗	17.5 – 18.8	10.8 – 13.0	1.5 – 1.8
Book-medium	book	✗	21.6 – 25.0	14.0 – 17.6	2.0 – 2.5
Book-large	book	✗	25.4 – 30.5	20.3 – 24.1	3.7 – 4.0