

4DP-QA: Scalable QA for 4D Perception in Vision Language Models

–Supplementary Materials–

Seokju Cho^{1,3*} Abhishek Badki¹ Hang Su¹ Jindong Jiang¹ Ziyao Zeng^{1,2*}
 Seungryong Kim³ Sifei Liu¹ Orazio Gallo¹

¹NVIDIA ²Yale University ³KAIST AI

Contents

1. Additional Training Details	1	2.3.4. Point Tracking QAs	7
1.1. Architectural Details for the Model with L4P	1	2.4. Additional Answer Generation Details	8
1.2. Training L4P for True-Motion Point Tracking	2	2.4.1. Distractor Generation for Multiple-Choice Questions	8
2. 4DP-QA Details	2	2.4.2. Negative Question Augmentation	9
2.1. Common Design Principles	2	3. Dataset Statistics	9
2.2. Pipeline Details	3	3.1. Question Type Distribution	9
2.2.1. Data Standardization	3	3.2. Answer Format Distribution	9
2.2.2. Asset Sampling	4	4. Additional Results	9
2.2.3. Object Reference System	4	4.1. Additional Benchmark Results	9
2.3. Question Generation Pipeline	4	4.2. QA-Pair Bias Analysis	9
2.3.1. Camera Motion QAs	5	4.3. Quantitative Results of Tracking Tasks	10
2.3.2. Object Motion QAs	5	4.4. Visualization of the 4DP-QA-Bench	10
2.3.3. 3D Spatial Understanding QAs	7	5. Limitations	10

1. Additional Training Details

1.1. Architectural Details for the Model with L4P

Beyond improving the baseline VLM models, we also investigate the effect of incorporating a geometry encoder (L4P [1]) into the VLM model. Figure 1 depicts the architecture of the VLM model augmented with L4P. The features extracted from L4P serve as 4D geometric inputs to the VLM. Specifically, we take the output of the VideoMAE backbone in L4P with shape $(T, H_{4D}, W_{4D}, C_{4D})$ and project it into the LLM input space using an MLP projector, following the design of the NVILA-Lite-8B projector [12]. The projected features are then tokenized and flattened into a per-frame sequence of tokens with shape (T, L_{4D}, C_{LLM}) , where C_{LLM} is the LLM embedding dimension. For each frame t , we concatenate these L4P 4D tokens with the corresponding SigLIP [19] visual tokens from NVILA along the token dimension, and finally build the full input sequence by stacking these per-frame token blocks in temporal order ($t=0, 1, \dots, T-1$).

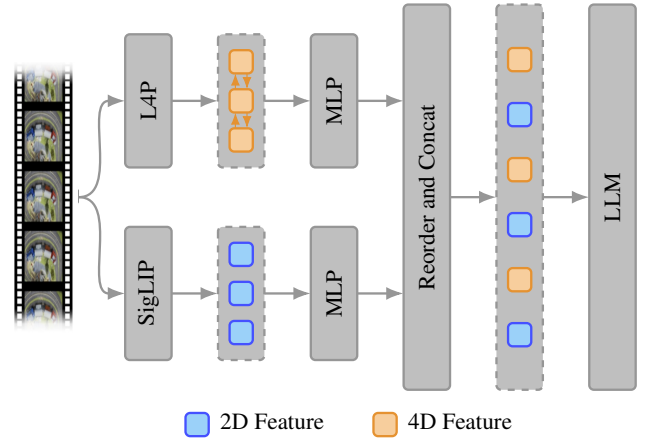


Figure 1. **Architecture of the VLM model enhanced with L4P.** We use the features extracted from L4P [1] as 4D geometry inputs to the MLLM [12]. The features from L4P are projected into the LLM input space and then used as input to the LLM along with visual tokens from SigLIP [19].

*Work done during internship at NVIDIA.

1.2. Training L4P for True-Motion Point Tracking

We describe how to train L4P to directly estimate true-motion point tracks without explicitly predicting depth or camera poses. The L4P model comprises a video encoder and task-specific decoders for both dense tasks (such as depth, optical flow, camera pose, and dynamic segmentation) and sparse tasks (including 2D and 3D visual point tracking). To enable 2D true-motion point tracking, we modify the sparse tracking head, which was originally designed for 2D visual point tracking.

The sparse tracking head in L4P uses the prompt encoding and mask decoding mechanism of Segment Anything [11]. It introduces special input and output tokens that interact with the video encoder’s output to estimate the 2D locations of points and their visibility. For a given input query pixel $\mathbf{p}[t_q] = (t_q, x_q, y_q)$, the system embeds it using 3D positional encoding alongside a learnable embedding to produce an input point token. In the case of 2D visual point tracking (P_{2D} in Equation 1 in the main paper), L4P utilizes two output tokens with learnable embeddings: one for the 2D track position and another for track visibility. These input and output tokens interact with each other and with the output of the video encoder through a two-way attention mechanism, allowing the model to decode the video features and generate the 2D visual point track P_{2D} .

To extend L4P for true-motion point tracking (M_{2D} in Equation 2 in the main paper) for the input query pixel $\mathbf{p}[t_q]$, we simply introduce two additional output tokens with learnable embeddings—one for the position of the 2D true-motion point track, and one for its visibility. We use the same two-way attention mechanism to simultaneously generate both the 2D visual point track and the 2D true-motion point track. Ground-truth annotations for the true-motion point track are generated by projecting the GT 3D trajectory onto the camera view at time t_q . The losses for the 2D true-motion point track mirror those used for the visual point track: an L1 loss for position and a binary cross-entropy loss for visibility. Training follows the same data and procedures as the original L4P model.

By adding true-motion to the L4P training and then incorporating it with the VLM, we saw an average increase from 85.3% to 87.7% on our benchmark and 59.5% to 60.0% on VLM4D. Figure 2 presents L4P results for both visual and true-motion point tracking on in-the-wild videos that involve complex camera and object movements. For each video, we select multiple 2D query points in the chosen frame and generate the corresponding visual and true-motion point tracks. These results demonstrate that true-motion point tracking offers an intuitive and interpretable motion representation. Moreover, these results indicate that, although true-motion point tracking requires the model to implicitly reason about depth and camera poses, it is possible to learn this task directly—without explicit depth and



Figure 2. **Visual vs. True-motion Point Tracking using L4P.**

In the first row, true-motion point tracks correctly identify that the cart remains stationary while the person moves toward it. In the second row, true-motion point tracks for the train are better aligned with the railway track. Across other examples, true-motion point tracks provide a more accurate and interpretable representation of object movement as compared to the visual point tracks.

pose computation or explicit geometric operations. This suggests that VLM models may also be capable of learning such representations in a similar manner.

2. 4DP-QA Details

This section details the construction of 4DP-QA. We begin with our common design principles in Section 2.1 and pipeline details in Section 2.2. Section 2.3 describes the question generation framework, including specific protocols for camera motion (Section 2.3.1), object motion (Section 2.3.2), and 3D spatial understanding (Section 2.3.3). Point tracking questions are detailed in Section 2.3.4, followed by more details on QA generation in Section 2.4.

2.1. Common Design Principles

The use of hysteresis thresholding. To generate reliable ground truth annotations, we employ hysteresis thresholding with two threshold levels for each motion type. For

translational motion, we define an upper threshold $\tau_{\text{trans}}^{\text{upper}}$ and a lower threshold $\tau_{\text{trans}}^{\text{lower}}$. A motion is classified as positive if the displacement exceeds $\tau_{\text{trans}}^{\text{upper}}$, negative if it falls below $\tau_{\text{trans}}^{\text{lower}}$, and undecided otherwise. Similarly, for rotational motion we use angular thresholds $\tau_{\text{rot}}^{\text{upper}}$ and $\tau_{\text{rot}}^{\text{lower}}$. Importantly, we designed all question-answer pairs to exclude undecided cases. Specifically, no questions ask about motions that fall in the undecided range (i.e., between the lower and upper thresholds), and undecided motion states do not appear as distractors in multiple-choice questions. This approach ensures high annotation quality by excluding ambiguous cases.

Interval monotonicity constraint. For all motion-related questions, we enforce a monotonicity constraint to ensure that trajectories exhibit consistent directional change. Given a trajectory $\{\xi_t\}_{t=1}^T$ (representing translation along an axis or rotation angle), we require that all intermediate values lie between the endpoint values. Formally, let $\xi_{\min} = \min_t \xi_t$ and $\xi_{\max} = \max_t \xi_t$ denote the minimum and maximum values. The trajectory is considered monotonic if either $\xi_1 \approx \xi_{\min}$ and $\xi_T \approx \xi_{\max}$, or $\xi_1 \approx \xi_{\max}$ and $\xi_T \approx \xi_{\min}$.

To account for noise and minor fluctuations, we introduce a tolerance margin $\delta = \gamma(\xi_{\max} - \xi_{\min})$, where $\gamma = 0.1$ is the margin ratio. The approximation $\xi_i \approx \xi_j$ is satisfied if $|\xi_i - \xi_j| \leq \delta$. This constraint filters out trajectories with severe non-monotonic behavior, ensuring that the ground truth annotations correspond to clear, unambiguous motion patterns. The monotonicity check is applied to all directional translation components, rotation angles, and distance trajectories before generating questions.

Balanced sampling of static and dynamic objects. To avoid dataset bias toward moving objects, we implement balanced sampling when selecting object pairs. For questions involving two objects, we preferentially sample pairs consisting of one moving and one static object. This ensures that models learn to reason about both dynamic and static elements in the scene.

Gravity-aligned motion sensing. To ensure consistent geometric reasoning across datasets with different coordinate conventions, we identify the gravity axis for each dataset and align motion analysis to this axis for camera and object motion questions. We decompose translations and rotations into gravity-aligned components. Let $\mathbf{g} \in \mathbb{R}^3$ denote the gravity direction vector (pointing upward). For any 3D motion vector \mathbf{v} , we decompose it into vertical and horizontal components: $\mathbf{v}_{\text{vert}} = (\mathbf{v} \cdot \mathbf{g})\mathbf{g}$ and $\mathbf{v}_{\text{horiz}} = \mathbf{v} - \mathbf{v}_{\text{vert}}$. This enables intuitive motion descriptions such as “moving forward” and “moving upward” that are invariant to camera orientation.

2.2. Pipeline Details

2.2.1. Data Standardization

Component	Description
Video Frames	RGB video frames with shape $(T, H, W, 3)$ where T is the number of frames.
Frame Rate	Frames per second of the video sequence.
Camera Ininsics	Camera intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3 \times 3}$ (assumed shared across frames), encoding focal lengths and principal point.
Camera Extrinsics	Camera extrinsic matrices $\mathbf{T}[t] \in \mathbb{R}^{4 \times 4}$ for each frame, representing world-to-camera transformation.
Depth Maps	Per-pixel depth measurements $\mathbf{D}[t] \in \mathbb{R}^{H \times W}$ indicating distance from the camera. Depth values are not necessarily valid for all pixels; the maps may be sparse.
Depth Validity Masks	Binary masks $\mathbf{B}[t] \in \{0, 1\}^{H \times W}$ indicating reliable depth values, where 1 indicates a valid depth value and 0 indicates an invalid depth value.
Instance Segmentation	Per-pixel segmentation masks $\mathbf{S}[t] \in \mathbb{Z}^{H \times W}$ assigning each pixel to an object instance.
Instance Metadata	Semantic information for each tracked object, including its name and category. If the category is missing or incomplete in the dataset, it can be generated using an off-the-shelf VLM [17].
Object Poses	Object-to-world transformation matrices $\mathbf{O}_i[t] \in \mathbb{R}^{4 \times 4}$ encoding rigid-body pose for each tracked object i in frame t .

Table 1. **Standardized data format specification.** We unify the formats of datasets from various sources into a single standardized format to streamline the pipeline.

We standardize all datasets from various sources into a unified format to streamline the generation pipeline, as summarized in Table 1. Particular care is taken to unify the coordinate conventions, since different datasets adopt different standards. We adopt the OpenCV coordinate convention: in image space, the x-axis points to the right and the y-axis points downward, while in the camera coordinate system, the z-axis points forward. For depth, we convert all data to planar depth when a dataset provides only radial depth. We also observe that each dataset aligns gravity with one of the coordinate axes, but the specific axis differs across datasets. We identify the gravity axis for each dataset and incorporate this information during dataset generation to maintain consistent geometric reasoning.

Some datasets contain incomplete annotations for certain data types we require. For example, HOT3D [3] provides depth only for the hands and objects, leaving other regions invalid. In such cases, we avoid generating questions that reference regions where depth is unavailable.

Several datasets also provide incomplete or missing instance names. To address this, we use the Qwen3-VL-32B-Instruct [17] model to annotate object descriptions. Specifically, for each object, we draw a red bounding box on the frame where the entire object is visible and appears largest in the image, and we prompt the VLM to identify the object’s name. The exact prompt used for this annotation process is as follows:

Prompt for Object Name Annotation

Give a short description of the object inside the red bounding box. Keep it concise and to the point. If applicable specify the object category, its attributes like colors. Describe in a way that is easy to distinguish from other objects in the image. Do not include the description of the surrounding of the object. Use less than 8 words.

In cases where the dataset contains noisy or unreliable metadata for certain objects, we append an additional prompt to guide the model. This prompt provides the existing metadata as a hint but instructs the model to use it only if it is helpful for producing an accurate visual description.

Prompt for Handling Noisy Metadata

I also have some metadata that may or may not be correct. For the object inside the red box the current metadata is {metadata}. Only use this metadata if it helps with visual description, otherwise ignore it.

2.2.2. Asset Sampling

We employ a two-stage approach for sampling valid question-answer pairs.

Stage 1: Interesting segment identification. We identify *interesting segments* where significant motion occurs to focus on informative video portions. First, we calculate the maximum spatial occupancy of each object across the video using pooled segmentation masks. Objects exceeding a specific area ratio threshold (default 0.1%) are classified as *large objects*. We then filter this set to include only objects that are moving, based on their 3D displacement. A contiguous sequence of frames is marked as an *interesting segment* if it contains at least one large, moving object in every frame for a minimum duration.

Stage 2: Snippet sampling within segments. For each sampled interesting segment, we select a sub-sequence (snippet) of frames. If the segment length is shorter than the target snippet length, we pad the selection by extending the start and end indices into the surrounding video to meet the required length. Otherwise, we sample a sub-segment with a randomized temporal span within the interesting segment boundaries. We then generate evenly spaced frame indices between the selected start and end frames. If the number of sampled frames exceeds the target input size (typically 32), we uniformly subsample the sequence. Finally, to increase data diversity, we apply temporal reversal augmentation with a probability of 0.5, flipping the chronological order of the frames.

2.2.3. Object Reference System

To refer to objects and points in video frames, we implement a flexible reference system that supports visual mark-

ers, coordinate-based specifications, and natural language names. This system enables clear, unambiguous references while maintaining diversity in question phrasing.

(1) Color-coded circular markers. Objects are referenced using colored circles drawn around their segmentation masks. For a given object with track ID i in frame f_t , we compute its bounding box from the segmentation mask $S[f_t]$ and draw an ellipse with center c_{2D} at the box center and axes determined by the box dimensions. We follow the format of the circle described in [15]. The circle is drawn with thickness $\tau_{\text{thick}} = 0.01 \cdot H$ where H is the image height. We use a fixed color palette that cycles through six colors: {red, green, blue, yellow, purple, orange}. Example references include “the red circle in frame 2” or “the object at the blue circle (frame 3).” For visual examples, please refer to the dataset visualizer described in Section 4.4.

To clearly indicate an object using a circular marker, we select the most suitable frame within the snippet based on two criteria. First, we check which frames show the entire object inside the image, without any part cut off at the image edges. Second, among those frames, we pick the one where the object covers the largest area in its segmentation mask. If no frame shows the object fully inside the image, we instead choose the frame where the object’s visible area is largest, even if it is partly outside the image. This approach ensures the object is both visible and clearly highlighted by the marker.

(2) Point markers with color. For point-based queries, following [15], we draw a circular marker at the specified pixel location (x, y) with radius $r = 0.06 \cdot H$ and thickness τ_{thick} , and add a filled dot of radius τ_{thick} at the center. The point is then referenced as “the red point at (0.451, 0.328) in frame 0” or simply “the green point (frame 1).”

(3) Coordinate-based references. Points can also be referenced purely by their normalized coordinates without visual markers: “the point at (0.451, 0.328)” or “point (0.451, 0.328) in frame 3.” Coordinates are normalized to $[0, 1] \times [0, 1]$ where (0, 0) is the top-left corner. For our benchmark evaluation, however, we do not include questions that involve these absolute coordinates, as coordinate conventions (e.g., origin and normalization) may differ across different VLMs.

(4) Natural language name. Objects can be referenced directly by their semantic class name or description if available in the dataset metadata or generated by a VLM. Example references include “the white sedan” or “the person in the red shirt.”

2.3. Question Generation Pipeline

The generation of each question-answer pair follows a structured pipeline that ensures diversity, balance, and annotation quality.

Question type and snippet selection. For each annotation, we first sample a question type from the available set (e.g., camera motion, object rotation, depth comparison) according to configured sampling weights. We then sample a snippet within a video $\mathcal{F} = \{f_1, \dots, f_T\}$ as described in the asset sampling section.

Template-based question generation. Each question type maintains a set of question templates and choice templates. For example, the camera motion question type has templates such as “Which of the following best describes the camera’s movement?” and “Considering the entire clip, which of the following describes the camera’s primary motion?” During generation, we randomly select a template and populate it with the analysis results (e.g., detected motions, object references). This template-based approach ensures linguistic diversity while maintaining semantic consistency.

Below, we detail each question-generation pipeline, beginning with the definition of notations, followed by camera motion QAs, object motion QAs, and finally, 3D spatial understanding QAs.

Notations. Throughout this section, we use the following notations aligned with Table 1: T denotes the number of frames in a snippet, indexed by $t \in \{1, \dots, T\}$. Camera intrinsics are denoted $\mathbf{K} \in \mathbb{R}^{3 \times 3}$ (shared across frames), and camera extrinsics at frame t are $\mathbf{T}[t] \in \mathbb{R}^{4 \times 4}$. The camera position in world coordinates is computed as $\mathbf{p}_{\text{cam}}(t) = -\mathbf{R}[t]^\top \mathbf{t}[t]$, where $\mathbf{T}[t] = [\mathbf{R}[t] | \mathbf{t}[t]]$. For object i , we denote its pose at frame t as $\mathbf{O}_i[t] \in \mathbb{R}^{4 \times 4}$ and its center in world coordinates as $\mathbf{c}_i(t) = \mathbf{O}_i[t]_{:3,3}$. The gravity direction is $\mathbf{g} \in \mathbb{R}^3$ with $\|\mathbf{g}\| = 1$.

2.3.1. Camera Motion QAs

(1) Camera movement. We analyze camera motion by decomposing it into gravity-aligned translation and rotation components. For a snippet with frames $\mathcal{F} = \{f_1, \dots, f_T\}$, we compute the camera trajectory $\{\mathbf{p}_{\text{cam}}(f_t)\}_{t=1}^T$ in world coordinates. We project this trajectory onto three axes: (1) the forward direction \mathbf{d}_{fwd} , obtained by projecting the initial camera’s forward vector onto the ground plane; (2) the right direction $\mathbf{d}_{\text{right}} = \mathbf{d}_{\text{fwd}} \times \mathbf{g}$; and (3) the vertical direction \mathbf{g} . The camera translation is then determined from the signed displacements along these three projected trajectories.

For rotation analysis, we compute gravity-aligned Euler angles. Let $\mathbf{R}[t]$ denote the camera rotation matrix at frame t (extracted from $\mathbf{T}[t]$). We define: (1) *pitch* as the angle between the camera’s forward vector and a plane perpendicular to the gravity axis, (2) *yaw* as the rotation around the gravity axis, measured relative to an arbitrary horizontal direction that remains fixed throughout the snippet, and (3) *roll* as the rotation around the camera’s viewing axis. All angles are computed relative to the first frame. Motion is

classified using hysteresis thresholding: $|\Delta| > \tau^{\text{upper}}$ indicates motion in one direction, $|\Delta| < \tau^{\text{lower}}$ indicates no motion, and intermediate values are discarded as undecided.

Example Question Template: Camera Movement

Question: Which of the following best describes the camera’s movement?

Choices:

- The camera is moving forward and panning right
- The camera is moving backward and tilting down
- The camera is stationary

(2) Camera-object distance change. For camera-object proximity analysis, we track the distance between the camera and an object’s center over time. Let $d_i(t) = \|\mathbf{c}_i(t) - \mathbf{p}_{\text{cam}}(t)\|$ denote the distance at frame t . We require the distance trajectory $\{d_i(f_t)\}_{t=1}^T$ to be monotonic. The change is classified based on $\Delta d = d_i(f_T) - d_i(f_1)$ and relative change $\rho = \Delta d / \max(d_i(f_1), d_i(f_T))$. The objects are getting closer if $\Delta d < -\tau_{\text{trans}}^{\text{upper}}$ and $\rho < -\tau_{\text{rel}}^{\text{upper}}$; moving apart if $\Delta d > \tau_{\text{trans}}^{\text{upper}}$ and $\rho > \tau_{\text{rel}}^{\text{upper}}$; and maintaining distance if $|\Delta d| < \tau_{\text{trans}}^{\text{lower}}$.

Additionally, we analyze the relative movement direction between camera and object on the ground plane. Let $\mathbf{v}_{\text{cam}} = \mathbf{p}_{\text{cam}}(f_T) - \mathbf{p}_{\text{cam}}(f_1)$ and $\mathbf{v}_{\text{obj}} = \mathbf{c}_i(f_T) - \mathbf{c}_i(f_1)$ be the displacement vectors, projected onto the ground plane. We compute the alignment $\alpha = \cos^{-1}(\hat{\mathbf{v}}_{\text{cam}} \cdot \hat{\mathbf{v}}_{\text{obj}})$. The directions are classified as: along (same direction) if $\alpha < \tau_{\text{align}}$, against (opposite) if $\alpha > 180 - \tau_{\text{align}}$, and perpendicular if $|\alpha - 90| < \tau_{\text{perp}}$.

Example Question Template: Camera-Object Distance

Question: Describe the change in proximity between the camera and the red cup (frame 5) throughout the clip.

Choices:

- The camera and the red cup got closer
- The camera and the red cup moved further apart
- The distance between them stayed roughly the same

2.3.2. Object Motion QAs

(3) Rotation. We analyze object rotation from a top-down perspective by tracking yaw changes around the gravity axis. For object i , we extract rotation matrices $\{\mathbf{R}_i[f_t]\}_{t=1}^T$ from its pose $\mathbf{O}_i[f_t]$. To measure yaw, we project the object’s local x-axis onto the ground plane at each frame, obtaining $\mathbf{x}_i^\perp(t) = \mathbf{x}_i(t) - (\mathbf{x}_i(t) \cdot \mathbf{g})\mathbf{g}$, where $\mathbf{x}_i(t) = \mathbf{R}_i[f_t]\mathbf{e}_1$. We measure angles against a fixed horizontal reference frame and unwrap the angle trajectory $\{\theta_i(f_t)\}_{t=1}^T$ to handle ± 180 discontinuities. The yaw change $\Delta\theta =$

$\theta_i(f_T) - \theta_i(f_1)$ is classified as clockwise if $\Delta\theta < -\tau_{\text{rot}}^{\text{upper}}$, counter-clockwise if $\Delta\theta > \tau_{\text{rot}}^{\text{upper}}$, and no rotation if $|\Delta\theta| < \tau_{\text{rot}}^{\text{lower}}$. We verify that pitch and roll changes remain below $\tau_{\text{rot}}^{\text{upper}}$ to ensure valid top-down analysis.

Example Question Template: Object Rotation

Question: From a top-down perspective, which of the following best describes the rotation of the blue toy car (frame 3)?

Choices:

- The blue toy car is rotating clockwise
- The blue toy car is rotating counter-clockwise
- The blue toy car is not rotating

(4) Direction. For directional motion analysis, we decompose object translation in a stable reference frame defined by the initial camera view. Let $\mathbf{v}_i = \mathbf{c}_i(f_T) - \mathbf{c}_i(f_1)$ be the object’s displacement vector. We project it onto three axes: (1) camera-aligned horizontal right $\mathbf{d}_{\text{right}}$, (2) gravity direction \mathbf{g} , and (3) camera-aligned horizontal forward \mathbf{d}_{fwd} . For each axis, we compute the projected trajectory and verify monotonicity. Motion is classified independently for each direction using thresholds that adapt to object depth: $\tau_i^{\text{upper}} = \max(\tau_{\text{trans}}^{\text{upper}} \cdot 2, \tau_{\text{depth-rel}}^{\text{upper}} \cdot d_i(f_1))$, where $d_i(f_1)$ is the initial depth. This accounts for the fact that objects farther from the camera require larger absolute displacements to be perceived as moving. Questions can ask about single-direction motion or combined motion (e.g., “moving forward and to the right”). We also generate counting questions asking how many objects exhibit a specific directional motion.

Example Question Template: Directional Motion

Question: From the camera’s initial perspective (where ‘forward/backward’ are relative to the camera’s forward viewing direction), which option best describes the movement of the yellow block (frame 2)?

Choices:

- The yellow block is moving forward and up
- The yellow block is moving backward and left
- The yellow block is stationary

(5) Agent motion. For agent motion (e.g., a person or car), we combine perspective-based translation with egocentric rotation analysis. First, we determine if the agent is moving forward or backward from its own perspective by comparing its velocity vector \mathbf{v}_i with its forward direction $\mathbf{f}_i(f_1) = \mathbf{R}_i[f_1]\mathbf{e}_1$, both projected onto the ground plane. Let $\alpha = \cos^{-1}(\hat{\mathbf{v}}_i \cdot \hat{\mathbf{f}}_i)$ be the angle between them.

The agent is moving forwards if $\alpha < \tau_{\text{persp}}^{\text{lower}}$, backwards if $\alpha > 180 - \tau_{\text{persp}}^{\text{lower}}$, and neither if $\tau_{\text{persp}}^{\text{lower}} < \alpha < \tau_{\text{persp}}^{\text{upper}}$ or $180 - \tau_{\text{persp}}^{\text{upper}} < \alpha < 180 - \tau_{\text{persp}}^{\text{lower}}$.

Second, we analyze turning direction using the yaw trajectory as described in rotation analysis. However, if the agent is moving backward, we flip the turn direction (left \leftrightarrow right) to maintain consistency with the agent’s egocentric perspective. Both translation and rotation trajectories must be monotonic for valid classification.

Example Question Template: Agent Motion

Question: How’s the agent’s motion? Is it moving forwards, backwards, or turning left or right?

Choices:

- The person is moving forwards and turning left
- The person is moving backwards
- The person is not moving forwards or backwards

(6) Moved distance. To compare movement distances between objects, we compute the 3D displacement for each object: $\Delta_i = \|\mathbf{c}_i(f_T) - \mathbf{c}_i(f_1)\|$. For a pair of objects (i, j) , we classify their relative movement as: object i moved further if $\Delta_i > \Delta_j + \tau_{\text{comp}}$; object j moved further if $\Delta_j > \Delta_i + \tau_{\text{comp}}$; or neither moved significantly if $\Delta_i < \tau_{\text{trans}}^{\text{lower}}$ and $\Delta_j < \tau_{\text{trans}}^{\text{lower}}$. The comparison threshold $\tau_{\text{comp}} = 5 \cdot \tau_{\text{trans}}^{\text{upper}}$ ensures clear distinction between movements.

Example Question Template: Moved Distance

Question: Which object moved a greater distance, the red ball (frame 1) or the green cube (frame 2)?

Choices:

- The red ball moved further
- The green cube moved further
- Neither object moved significantly

(7) Object-object distance change. For analyzing distance changes between two objects, we compute the inter-object distance trajectory $\{d_{ij}(f_t)\}_{t=1}^T$, where $d_{ij}(t) = \|\mathbf{c}_i(t) - \mathbf{c}_j(t)\|$. We require valid 3D positions for both objects in at least 80% of overlapping frames and monotonicity of the distance trajectory. The change is classified based on $\Delta d_{ij} = d_{ij}(f_T) - d_{ij}(f_1)$ with adaptive thresholding: $\tau_{ij}^{\text{upper}} = \max(\tau_{\text{trans}}^{\text{upper}} \cdot 2, \tau_{\text{depth-rel}}^{\text{upper}} \cdot \max(d_i(f_1), d_j(f_1)))$, where $d_i(f_1)$ and $d_j(f_1)$ are the distances of each object from the camera at the first frame. Objects are getting closer if $\Delta d_{ij} < -\tau_{ij}^{\text{upper}}$, moving apart if $\Delta d_{ij} > \tau_{ij}^{\text{upper}}$, and staying at the same distance if $|\Delta d_{ij}| < \tau_{\text{trans}}^{\text{lower}}$.

Example Question Template: Object-Object Distance

Question: Which best describes the distance change between the white mug (frame 1) and the laptop (frame 3) over the course of the video?

Choices:

- They are getting closer
- They are getting farther apart
- They are staying at a similar distance

2.3.3. 3D Spatial Understanding QAs

(8) Depth. For single-frame depth comparison, we select two points \mathbf{p}_a and \mathbf{p}_b in frame f_t with valid depth values from $\mathbf{D}[f_t]$ and compute their distances to the camera: $d_a = \|\mathbf{p}_a - \mathbf{p}_{\text{cam}}(f_t)\|$ and $d_b = \|\mathbf{p}_b - \mathbf{p}_{\text{cam}}(f_t)\|$. The question asks which point (or object) is closer/farther to the camera. For objects, we use their center positions $\mathbf{c}_i(f_t)$ and $\mathbf{c}_j(f_t)$. Points are selected to ensure meaningful depth difference: $\max(d_a/d_b, d_b/d_a) > \tau_{\text{depth-ratio}}$, where $\tau_{\text{depth-ratio}} = 1.25$.

Example Question Template: Depth Comparison

Question: In frame 0, which is closer to the camera: the red point (A) or the blue point (B)?

Choices: A, B

(9) Object-object distance. For single-frame object-object distance comparison, we select three objects i , j , and k visible in frame f_t . We compute distances $d_{ij} = \|\mathbf{c}_i(f_t) - \mathbf{c}_j(f_t)\|$ and $d_{ik} = \|\mathbf{c}_i(f_t) - \mathbf{c}_k(f_t)\|$, and ask which of objects j or k is closer to object i . We ensure meaningful distance difference with the same ratio threshold.

Example Question Template: Object-Object Distance

Question: In frame 0, which point is closer to the green cup: the yellow point (A) or the purple point (B)?

Choices: A, B

(10) Multi-view depth. For cross-frame depth comparison, we select two objects i and j that appear in different frames f_a and f_b respectively, and compare their depths relative to a reference frame f_r . Let $\mathbf{p}_{\text{cam}}(f_r)$ be the reference camera position. We compute $d_i = \|\mathbf{c}_i(f_r) - \mathbf{p}_{\text{cam}}(f_r)\|$ and $d_j = \|\mathbf{c}_j(f_r) - \mathbf{p}_{\text{cam}}(f_r)\|$. The objects are annotated with their best visible frames f_a and f_b (where they appear largest), but the depth comparison is performed at the shared

reference frame f_r . This tests the model’s ability to reason about 3D positions across different viewpoints.

Example Question Template: Multi-View Depth

Question: In frame 2, which is closer to the camera: the orange bottle (A) shown in frame 0 or the blue pen (B) shown in frame 5?

Choices: A, B

(11) Multi-view object-object distance. For cross-frame object-to-object distance comparison, we select three objects i , j , k where object i serves as reference and objects j and k are presented in their best visible frames. All distance measurements are computed at a common reference frame f_r : $d_{ij} = \|\mathbf{c}_i(f_r) - \mathbf{c}_j(f_r)\|$ and $d_{ik} = \|\mathbf{c}_i(f_r) - \mathbf{c}_k(f_r)\|$. The question asks which of objects j or k is closer to reference object i in frame f_r , even though the objects are shown in different frames.

Example Question Template: Multi-View Object Distance

Question: In frame 1, which is closer to the white keyboard: the mouse (A) shown in frame 0 or the phone (B) shown in frame 3?

Choices: A, B

2.3.4. Point Tracking QAs

Point tracking questions require generating temporally consistent 3D trajectories and their 2D projections for query points across frames. We first recover a unified 3D trajectory $\{\mathbf{X}[f_t]\}$ for each query point using our standardized 3D scene representation, and then obtain different 2D track representations by projecting this trajectory into appropriate camera views.

Given a query point $\mathbf{p}[f_q]$ at reference frame f_q , we generate its trajectory across all frames f_t . The algorithm consists of three main steps:

Step 1: 3D Initialization. We first unproject the query point to 3D world coordinates. Let $z_q = \mathbf{D}[f_q](\mathbf{p}[f_q])$ be the depth at the query point. We compute the 3D point in the camera coordinate system at frame f_q :

$$\mathbf{X}_{\text{cam}}[f_q] = z_q \mathbf{K}^{-1}[\mathbf{p}[f_q]; 1] \quad (1)$$

We then transform to world coordinates using the camera extrinsics: $\mathbf{X}[f_q] = \mathbf{T}[f_q]^{-1}[\mathbf{X}_{\text{cam}}[f_q]; 1]$. This gives us the 3D position of the query point in the world coordinate system at the query time.

Step 2: Motion Association. We determine whether the point belongs to a moving object or the static scene by

querying the segmentation mask. Let $i = \mathbf{S}[f_q](\mathbf{p}[f_q])$ be the instance ID at the query point. If the point belongs to the background, it is part of the static environment and maintains the same world coordinates $\mathbf{X}[f_t] = \mathbf{X}[f_q]$ across all frames. Conversely, if the point belongs to a moving object, we transform it to the object’s local coordinate system at frame f_q : $\mathbf{X}_{\text{local}} = \mathbf{O}_i[f_q]^{-1}[\mathbf{X}[f_q]; 1]$. This local representation is invariant to the object’s motion. For each frame f_t , we then transform back to world coordinates using that frame’s object pose: $\mathbf{X}[f_t] = \mathbf{O}_i[f_t]\mathbf{X}_{\text{local}}$, which correctly handles object motion, rotation, and deformation.

Step 3: Projection and Visibility. For each frame f_t , we project the 3D world point to 2D image coordinates using the projection operator Π :

$$\mathbf{p}[f_t] = \Pi(\mathbf{K}, \mathbf{T}[f_t], \mathbf{X}[f_t]) \quad (2)$$

To determine visibility, we perform occlusion checking using depth comparison. We sample the observed depth map at the projected location using bilinear interpolation: $\hat{z}_t = \mathbf{D}[f_t](\mathbf{p}[f_t])$. The point is classified as visible if three conditions are satisfied: (1) the point projects within frame bounds; (2) the point is in front of the camera; and (3) the point is not occluded: $|z_t - \hat{z}_t| < \epsilon \cdot z_t$, where z_t is the depth of $\mathbf{X}[f_t]$ in camera f_t , and $\epsilon = 0.1$ is the visibility threshold. Together, these steps yield a single 3D trajectory $\{\mathbf{X}[f_t]\}$ with per-frame visibility.

On top of this unified 3D trajectory, we define two complementary track representations. *Visual point tracking* uses the time-varying camera poses to show how the point appears in each observed frame (P_{2D}), while *true-motion point tracking* re-projects the trajectory into a fixed reference view to isolate object motion from camera motion (M_{2D}).

(12) Visual point tracking. Visual point tracking asks the model to predict $P_{2D} = \{\mathbf{p}[f_t]\}$. Given a query point $\mathbf{p}[f_q]$, we generate its full trajectory and create questions in two formats: In the *single-target* format, the question asks for the location of the point $\mathbf{p}[f_t]$ in one specific target frame f_t , and the answer is either its 2D coordinates or an “occluded” label. In the *full-trajectory* format, the question instead asks for the entire sequence.

Example Question Template: Single-Target Visual Tracking

Question: Given the red point at (0.451, 0.328) in frame 0, where is this point in frame 5?

Example Answers:

- (0.804, 0.384)
- The point is occluded in frame 5.

Example Question Template: Full-Trajectory Visual Tracking

Question: Given the red point at (0.451, 0.328) in frame 0, what is its trajectory across all frames?

Example Answer:

- Frame 0: (0.451, 0.328), Frame 1: occluded, Frame 2: (0.486, 0.354), Frame 3: (0.522, 0.384)

(13) True-motion point tracking. True-motion point tracking projects the 3D trajectory $\{\mathbf{X}[f_t]\}$ onto a fixed reference camera view at time f_q , isolating object motion from camera motion. This corresponds to computing $M_{2D} = \{\mathbf{m}_{f_q}[f_t]\}$:

$$\mathbf{m}_{f_q}[f_t] = \Pi(\mathbf{K}, \mathbf{T}[f_q], \mathbf{X}[f_t]) \quad (3)$$

This produces a trajectory that shows how the point moves in a fixed camera reference frame. Questions follow the same format as visual point tracking but explicitly state the reference frame.

Example Question Template: True-Motion Tracking

Question: What does the 3D trajectory of the blue point at (0.386, 0.414) (frame 2) look like when projected to frame 2?

Example Answer:

- Frame 0: (0.421, 0.454), Frame 1: (0.402, 0.436), Frame 2: (0.386, 0.414), Frame 3: (0.359, 0.386)

2.4. Additional Answer Generation Details

2.4.1. Distractor Generation for Multiple-Choice Questions

For multiple-choice questions, we generate distractors (incorrect choices) tailored to each question type:

(1) Motion questions: We generate distractors by creating plausible but incorrect combinations of motion states. For example, if the correct answer is “moving forward and up,” distractors might be “moving backward and up” or “moving forward and down.” We use a two-tier prioritization: *mixed distractors* (containing both correct and incorrect motion components) are preferred over *fully false distractors* (containing only incorrect components), as mixed distractors are more challenging.

(2) Comparison questions: For depth and distance comparisons, distractors are simply the alternative choices (e.g., if A is closer, then B is the distractor).

Table 2. **Additional benchmark results.** We evaluate the performance of the models trained on 4DP-QA on additional benchmarks.

Methods	BLINK [8]	MMSI [18]	OmniSpatial [10]	SAT-Test [14]	SAT-Val [14]	VSTI [7]	SPAR [20]
Qwen2.5-VL-3B	50.9	26.6	41.8	64.0	54.2	41.3	34.0
Qwen2.5-VL-3B + 4DP-QA	49.3 -1.6	26.9 +0.3	40.4 -1.4	68.7 +4.7	68.0 +13.8	39.1 -2.2	37.7 +3.7
Qwen2.5-VL-7B	52.5	24.2	37.7	46.7	57.0	25.9	41.7
Qwen2.5-VL-7B + 4DP-QA	57.8 +5.3	26.2 +2.0	42.7 +5.0	74.0 +27.3	66.0 +9.0	42.6 +16.7	46.0 +4.3

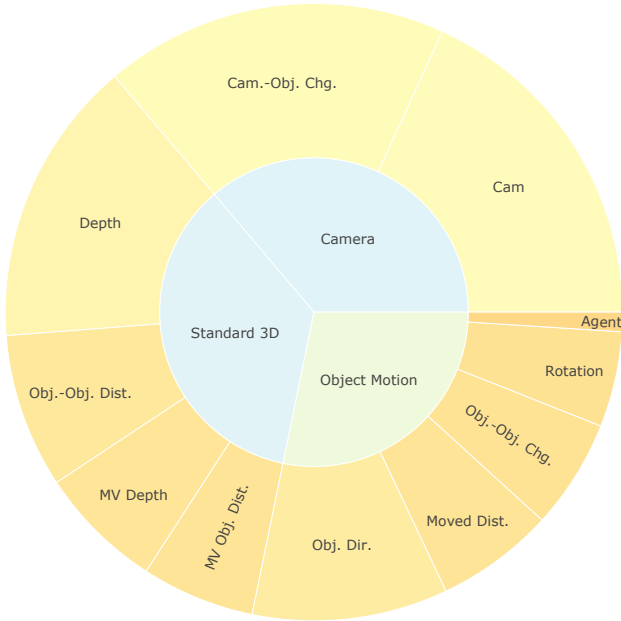


Figure 3. **Dataset statistics of 4DP-QA-Bench.** The dataset contains an even distribution of the three question categories: camera motion, object motion, and 3D spatial understanding.

2.4.2. Negative Question Augmentation

To improve robustness and test whether models can recognize non-existent objects, we augment a fraction of questions with *negative questions*. With probability $p_{\text{neg}} = 0.1$, we replace one or more object references in the question with fabricated descriptions (e.g., “the purple elephant” when no such object exists). For multiple-choice questions, the correct answer becomes a template like “There is no such object in the video” or “The object does not exist.” For positive questions (where the object exists), we occasionally add “There is no such object” as a distractor with the same probability p_{neg} , ensuring the model learns to correctly reject this choice when the object is present.

3. Dataset Statistics

3.1. Question Type Distribution

Figure 3 illustrates the distribution of question types in 4DP-QA-Bench. The dataset is balanced across the three

question types: camera motion, object motion, and 3D spatial understanding.

3.2. Answer Format Distribution

Our training set consists of 400K QA pairs, approximately 20% of which use multiple-choice format with 2-4 options, while the remaining 80% are open-ended questions requiring coordinate outputs (point tracking) or per-frame classifications (detailed distance analysis). Negative questions comprise approximately 10% of multiple-choice questions, ensuring models learn to recognize non-existent objects.

4. Additional Results

4.1. Additional Benchmark Results

To further assess the generalization of 4DP-QA, we evaluate a baseline VLM fine-tuned solely on 4DP-QA across a diverse suite of established benchmarks (Table 2). Specifically, BLINK [8] evaluates a broad range of general visual perceptual abilities. OmniSpatial [10] and MMSI [18] assess comprehensive spatial reasoning across diverse question types. SAT [14] and VSTI [7] cover both static and dynamic spatial scene understanding in video. SPAR [20] evaluates 3D spatial perception and reasoning from 2D inputs. Achieving strong performance across this broad set of benchmarks generally benefits from training on dataset mixtures combining task-specific data sources [2, 12, 17], which we leave as future work. Nonetheless, we report results of fine-tuning exclusively on 4DP-QA to demonstrate its generalization without any task-specific mixture training. We observe consistent performance improvements for Qwen2.5-VL-7B across all benchmarks, demonstrating that the low-level 4D understanding signals learned from 4DP-QA transfer to a wide range of spatial and perceptual tasks. Importantly, this generalizability is achieved without exposure to any of these target benchmarks during fine-tuning.

4.2. QA-Pair Bias Analysis

To assess the quality of our QA pairs and investigate potential template biases inherent to synthetically constructed datasets, we evaluate a set of carefully designed baselines on 4DP-QA-Bench. Results are summarized in Table 3.

Since our benchmark covers both multiple-choice and yes/no question formats, the expected random-choice ac-

Table 3. **QA-pair bias analysis on 4DP-QA-Bench.** Blind baselines replace all video frames with fully blacked-out images. Performance close to the random baseline (40.8%) confirms that language priors alone cannot solve the task, while the large gain from full visual input demonstrates that visual reasoning drives performance.

Model	Camera Motion	Object Motion	3D Spatial	Overall
Random	41.5	28.5	50.0	40.8
Gemini-2.5-Pro (blind)	37.7	44.7	40.1	40.0
NVILA-Lite-8B	42.4	26.0	55.4	42.3
NVILA-Lite-8B + 4DP-QA (blind)	55.5	45.1	63.6	55.1
NVILA-Lite-8B + 4DP-QA	83.5	81.6	88.6	84.4

Table 4. **Quantitative results of tracking tasks.** We evaluate the performance of the model on the point tracking tasks, namely, visual point tracking (VPT) and true-motion point tracking (TMPT). We use average Jaccard (AJ) score [6] to evaluate the performance of the model on the tracking tasks.

Method	Tracking Task	
	VPT	TMPT
NVILA-Lite-8B + Std-4DP-QA + VPT	43.2	–
NVILA-Lite-8B + Std-4DP-QA + TMPT	–	51.0
NVILA-Lite-8B + Std-4DP-QA + VPT + TMPT	44.1	48.8

curacy is 40.8%. To probe whether language priors or template-level cues alone can be exploited without visual information, we evaluate two blind baselines by replacing all video frames with fully blacked-out images. First, the proprietary Gemini-2.5-Pro achieves 40.0% accuracy under blind inputs, closely matching the random baseline. This indicates that language priors or template-level cues alone are insufficient to solve our benchmark. Second, we train Qwen2.5-VL-7B on blind input videos, allowing the model to potentially exploit any dataset-specific distributional biases or shortcuts. This variant achieves 52.5% accuracy, revealing a limited but measurable distributional bias. Such bias likely arises from inherent tendencies present in the underlying data sources used to construct our dataset.

In contrast, Qwen2.5-VL-7B with full visual input reaches 85.1% accuracy, confirming that visual reasoning is the primary driver of performance. While the model can exploit minor linguistic biases and correlations, as commonly observed in VQA datasets [4], the substantial gain with visual input (+32.6%) underscores that language priors alone cannot solve the task.

4.3. Quantitative Results of Tracking Tasks

In this experiment, we extract 200 tracking instances, with 100 cases of visual point tracking (VPT) and 100 cases of true-motion point tracking (TMPT). We then evaluate model performance on these tracking tasks using the average Jaccard (AJ) score [6], which measures both the accuracy of the predicted point locations and the occlusion

predictions. The results are shown in Table 4.

4.4. Visualization of the 4DP-QA-Bench

We include an interactive 4DP-QA-Bench visualizer in the supplementary material to facilitate a better understanding of the question and answer distribution. This tool displays QA pairs from each dataset: ADT [13], HOT3D [3], Virtual KITTI2 [5], Kubric [9], and SHIFT [16]. Due to space constraints, we present a subset of 212 QA pairs. To access the visualization, please open the `index.html` file in a web browser.

5. Limitations

We highlight some of the limitations of the 4DP-QA-Bench and the 4DP-QA dataset. Our current framework focuses primarily on rigid object motion, potentially limiting its applicability to complex non-rigid dynamics or highly deformable objects. Moreover, the use of hysteresis thresholding to filter out ambiguous motion helps ensure annotation reliability but also means that the dataset does not cover subtle motion cases that fall between the defined thresholds. We believe that these limitations can be addressed by extending the framework to cover more complex and subtle motion cases, and we leave this as future work.

References

- [1] Abhishek Badki, Hang Su, Bowen Wen, and Orazio Gallo. L4P: Low-level 4D vision perception unified. In *International Conference on 3D Vision (3DV)*, 2026. 1
- [2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibao Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL: Technical report. *ArXiv Preprint*, 2025. 9
- [3] Prithviraj Banerjee, Sindi Shkodrani, Pierre Moulon, Shreyas Hampali, Shangchen Han, Fan Zhang, Linguang Zhang, Jade Fountain, Edward Miller, Selen Basol, Richard Newcombe, Robert Wang, Jakob Julian Engel, and Tomas Hodan. HOT3D: Hand and object tracking in 3D from egocentric multi-view videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3, 10
- [4] Ellis Brown, Jihan Yang, Shusheng Yang, Rob Fergus, and Saining Xie. Benchmark designers should” train on the test set” to expose exploitable non-visual shortcuts. *ArXiv Preprint*, 2025. 10
- [5] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *ArXiv Preprint*, 2020. 10
- [6] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. TAP-Vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 10
- [7] Zhiwen Fan, Jian Zhang, Renjie Li, Junge Zhang, Runjin Chen, Hezhen Hu, Kevin Wang, Huaizhi Qu, Dilin Wang, Zhicheng Yan, et al. VLM-3R: Vision-language models augmented with instruction-aligned 3D reconstruction. *arXiv preprint arXiv:2505.20279*, 2025. 9
- [8] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. In *European Conference on Computer Vision (ECCV)*, 2024. 9
- [9] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanaprasagam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 10
- [10] Mengdi Jia, Zekun Qi, Shaochen Zhang, Wenyao Zhang, Xinqiang Yu, Jiawei He, He Wang, and Li Yi. Omnispacial: Towards comprehensive spatial reasoning benchmark for vision language models. In *International Conference on Learning Representations (ICLR)*, 2026. 9
- [11] 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. SAM: Segment anything. *ArXiv Preprint*, 2023. 2
- [12] Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, Xiuyu Li, Yunhao Fang, Yukang Chen, Cheng-Yu Hsieh, De-An Huang, An-Chieh Cheng, Vishwesh Nath, Jinyi Hu, Sifei Liu, Ranjay Krishna, Daguang Xu, Xiaolong Wang, Pavlo Molchanov, Jan Kautz, Hongxu Yin, Song Han, and Yao Lu. NVILA: Efficient frontier visual language models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 9
- [13] Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Yuheng Carl Ren. Aria Digital Twin: A new benchmark dataset for egocentric 3D machine perception. In *IEEE International Conference on Computer Vision (ICCV)*, 2023. 10
- [14] Arijit Ray, Jiafei Duan, Ellis Brown, Reuben Tan, Dina Bashkirova, Rose Hendrix, Kiana Ehsani, Aniruddha Kembhavi, Bryan A Plummer, Ranjay Krishna, et al. SAT: Dynamic spatial aptitude training for multimodal language models. In *Conference on Language Modeling (COLM)*, 2024. 9
- [15] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does CLIP know about a red circle? Visual prompt engineering for VLMs. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 4
- [16] Tao Sun, Mattia Segu, Janis Postels, Yuxuan Wang, Luc Van Gool, Bernt Schiele, Federico Tombari, and Fisher Yu. SHIFT: A synthetic driving dataset for continuous multi-task domain adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 10
- [17] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *ArXiv Preprint*, 2025. 3, 9
- [18] Sihan Yang, Runsen Xu, Yiman Xie, Sizhe Yang, Mo Li, Jingli Lin, Chenming Zhu, Xiaochen Chen, Haodong Duan, Xiangyu Yue, et al. MMSI-Bench: A benchmark for multi-image spatial intelligence. *ArXiv Preprint*, 2025. 9
- [19] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1
- [20] Jiahui Zhang, Yurui Chen, Yanpeng Zhou, Yueming Xu, Ze Huang, Jilin Mei, Junhui Chen, Yu-Jie Yuan, Xinyue Cai, Guowei Huang, et al. From Flatland to Space: Teaching vision-language models to perceive and reason in 3D. *ArXiv Preprint*, 2025. 9