VTInstructor: Visual Trajectory Prompting for Navigation Instruction Generation in Continuous Environments

Anonymous Author(s)

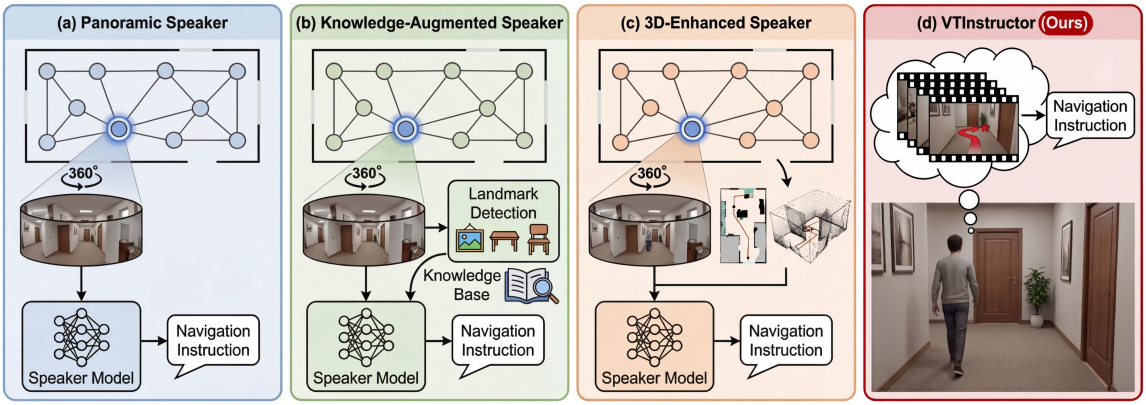

Figure 1: Evolution of navigation instruction generation paradigms. (a)–(c) Prior speakers rely on discrete viewpoint graphs with panoramic images, optionally augmented by external knowledge or 3D representations. (d) VTInstructor (Ours) generates instructions directly from ego-centric RGB video in continuous environments, using Visual Trajectory Prompts for spatial grounding without any navigation graph or 3D reconstruction.

Abstract

Navigation instruction generation from ego-centric RGB video in continuous environments is an important yet challenging task for human–robot interaction and scalable dataset construction. Prior instruction generators assume discrete viewpoint graphs with panoramic observations, where trajectory structure is explicit; in continuous environments, however, the agent receives only a dense RGB stream, making trajectory cues difficult to recover. We propose **VTInstructor**, the first VLN instruction generation framework for continuous environments. Our key idea is to convert implicit trajectory geometry into explicit visual trajectory prompts: EDTC condenses long RGB trajectories into navigation-critical keyframes, VTP overlays path, turn, and goal cues onto these anchors, VTMod injects the resulting trajectory signals into the visual encoder, and VT-GRPO further calibrates this spatial injection during training, all without requiring a navigation graph, pre-built map, or scene reconstruction. On the challenging R2R-CE and RxR-CE *Val Unseen* benchmarks, VTInstructor sets a new state of the art across all standard NLG metrics, surpassing the strongest baseline by +0.357 CIDEr and +0.109 CIDEr, respectively. Beyond automatic metrics, VTInstructor-generated instructions raise a frozen follower's success rate to 63.3%, a +14.7 percentage-point gain over the best competing instruction source, and provide consistent data augmentation gains of +3 SR points on downstream navigation tasks on downstream navigation tasks on both benchmarks.

CCS Concepts

• **Computing methodologies** → **Robotic planning**; **Vision for robotics**; • **Computer systems organization** → *Robotics*.

Keywords

visual trajectory, multimodal robotics, computer vision

1 Introduction

Navigation instruction generation (producing natural-language descriptions of traversed trajectories) is a fundamental task for human-robot interaction and the scalable construction of training data for embodied navigation models. Existing speaker models [5, 7, 11] have made substantial progress in discrete graph-based settings. Figure 1 summarizes the evolution of these paradigms, from panoramic speakers to knowledge-augmented and 3D-enhanced variants, all of which remain rooted in discrete viewpoint graphs. By contrast, navigation instruction generation in continuous environments remains largely unexplored.

Why has this setting remained unexplored? In discrete VLN environments, trajectories unfold over topological graphs whose nodes are associated with panoramic observations, making viewpoint relations explicit and trajectory structure easy to infer. In continuous environments, however, the agent receives only a dense ego-centric RGB stream, and adjacent frames may look highly similar even when they correspond to different motions or spatial states. This shift introduces two challenges. First, existing graph-based speaker models cannot be transferred directly to continuous environments. Second, trajectory cues such as path direction, turning behaviour, and goal progress are no longer explicit, and must instead be inferred from subtle visual changes. As a result, models struggle to recover directional information and often produce generic, spatially imprecise instructions.

To address these challenges, we propose **VTInstructor**, a unified navigation instruction generation framework for continuous environments that processes the entire pipeline within a single vision-language backbone. Our central idea is to convert trajectory

geometry into explicit visual trajectory prompts on ego-centric views, so that the model can perceive path structure directly instead of inferring it only from dense RGB streams. VTInstructor comprises three core components. (1) *Event-driven trajectory compression (EDTC)* first condenses long RGB trajectories into navigation-critical keyframes determined by the action sequence, providing visual anchors for subsequent prompting. (2) *Visual Trajectory Prompt (VTP) rendering* then overlays path, turn, and goal cues onto these keyframes, while GPT-based quality filtering (QF) retains only spatially reliable instruction-keyframe pairs for training. (3) *Visual Trajectory Modulator (VTMod) injection* feeds the resulting trajectory signals directly into the ViT encoder, strengthening spatial perception beyond what raw appearance alone can provide. On top of supervised fine-tuning, we further introduce VT-GRPO, which uses reinforcement learning to selectively calibrate the VTMod gates and explicitly refine how trajectory information is injected into the model.

We evaluate VTInstructor on the R2R-CE and RxR-CE *Val Unseen* benchmarks, where it achieves state-of-the-art performance across all standard NLG metrics (BLEU, METEOR, ROUGE-L, CIDEr, SPICE), outperforming the strongest baseline by +0.357 CIDEr and +0.109 CIDEr, respectively. Downstream navigation experiments further show that follower agents guided by VTInstructor-generated instructions achieve 14.7 percentage points higher success rate than those using competing instruction sources, and human evaluators consistently rate VTInstructor instructions higher on directional accuracy and overall followability.

Our contributions are:

- **The first VLN instruction generation framework for continuous environments.** VTInstructor generates instructions directly from raw ego-centric RGB trajectories in continuous environments, without relying on navigation graphs, panoramic renderings, pre-built maps, or scene reconstruction.
- **A visual trajectory prompting framework for explicit spatial grounding.** We convert implicit trajectory geometry in dense RGB streams into explicit spatial cues through EDTC for navigation-critical keyframe selection, VTP for path/turn/goal prompting on these anchors, VTMod for trajectory-aware visual encoding, and VT-GRPO for reward-driven calibration of spatial signal injection.
- **State-of-the-art performance with practical utility.** VTInstructor achieves state-of-the-art results on the R2R-CE and RxR-CE *Val Unseen* benchmarks, surpassing the strongest baseline by +0.357 and +0.109 CIDEr, respectively, improving frozen-follower success by 14.7 percentage points, and delivering +3 SR-point data augmentation gains on both benchmarks.

2 Related Work

2.1 Navigation Instruction Generation

Vision-and-Language Navigation (VLN) requires an agent to follow natural-language instructions in indoor environments [1, 2, 14]. Complementing instruction *following*, instruction *generation* (the speaker side) is critical for data augmentation and human–robot communication. Speaker-Follower [5] first trains an LSTM speaker for data augmentation; subsequent work improves generation through speaker–follower cycle consistency [19] and multi-task joint training [21]. Another line enriches the speaker with external knowledge or landmark grounding: SAS [6] and KEFA [24] introduce object-spatial attention and commonsense alignment, while landmark-based instruction generation methods such as Less is More [20] abstracts the trajectory into detected landmarks and generates instructions via a text-to-text model. More recently, C-Instructor [8] adopts chain-of-thought prompting with a multi-modal LLM, and MapInstructor [4] and BEV-LLM [3] leverage top-down maps or BEV representations for global spatial reasoning. Despite their diversity, existing instruction-generation methods are developed in discrete navigation graphs, where each state is typically represented by privileged panoramic observations—often instantiated in R2R-style VLN as a set of 36 discretized views per viewpoint—rather than raw first-person continuous perception. R2R-CE [9] and RxR-CE [10] extend VLN benchmarks to the continuous Habitat simulator, yet speaker models for this setting remain unexplored. VTInstructor is the first VLN instruction generation framework for continuous environments, taking only ego-centric RGB video as input without graph topology or panoramic privilege.

2.2 Visual Prompting

Visual prompting augments input images with task-relevant annotations to steer model attention without modifying model weights. In the general vision-language domain, coloured circles [16], numbered markers [22], and iteratively overlaid arrows and keypoints [13] have been used for referring expression comprehension, visual grounding, and action prediction. In robotic manipulation, TraceVLA [26] overlays end-effector trajectory traces on the current observation to enhance spatial-temporal awareness of VLA policies, and Robotic Visual Instruction [12] annotates images with visual cues to guide manipulation actions. Despite the growing adoption of visual prompting in perception and manipulation, it has not been explored for navigation instruction generation. VTInstructor is the first to introduce Visual Trajectory Prompts (VTP) into this task, with two key designs: (i) we render a path ribbon, rotation arrow, and endpoint marker in the ego-centric perspective, derived directly from the compressed action sequence; and (ii) rather than treating the annotated image as a standalone input token, we encode the VTP as a multi-channel binary mask and inject it into the ViT backbone via spatial modulation, enabling the language model to perceive trajectory geometry at the patch level.

3 Method

VTInstructor addresses navigation instruction generation in continuous 3D environments by converting implicit trajectory geometry in dense ego-centric RGB streams into explicit spatial cues. As shown in Figure 2, the framework has four components: (1) event-driven trajectory compression (EDTC), which condenses long trajectories into navigation-critical keyframes; (2) Visual Trajectory Prompt (VTP) rendering and data curation, which overlay path, turn, and goal cues onto these anchors; (3) VTMod, which injects trajectory-aware signals into the visual encoder; and (4) VT-GRPO, which further calibrates this spatial injection during training.

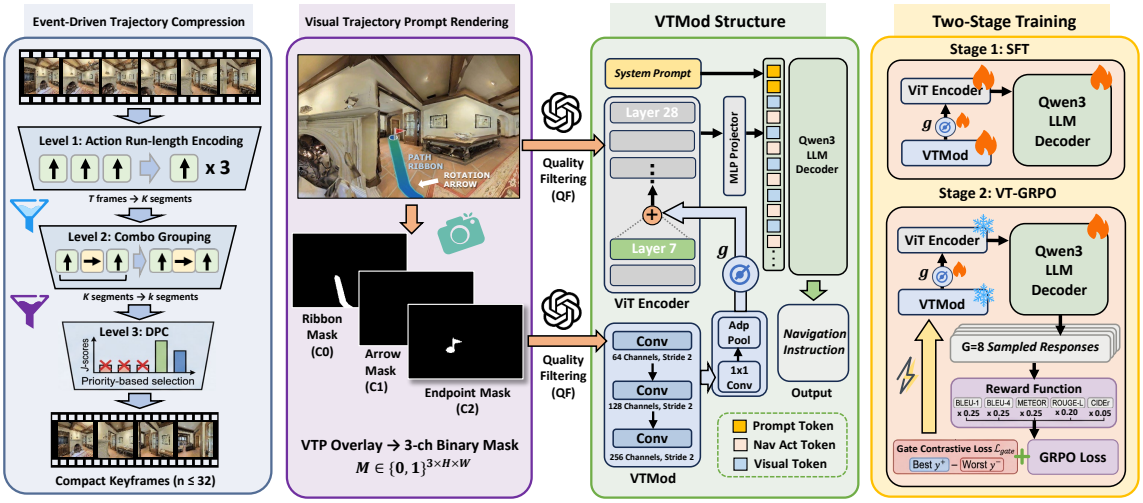

Figure 2: Overview of VTInstructor. Given a raw ego-centric RGB trajectory, the three-level event-driven compression pipeline produces a compact keyframe set. Each keyframe is annotated with a Visual Trajectory Prompt overlay (path ribbon / rotation arrow / endpoint marker), encoded as a 3 channels binary mask, and fed to the VTMod-augmented Qwen3-VL backbone. The VTP Encoder maps the mask to patch-aligned spatial features, which are injected into the ViT Encoder at layer 7 via per-token spatial modulation. The model is trained in two stages: an initial SFT stage, followed by VT-GRPO, which further optimizes the Qwen3 LLM decoder and the VTMod gates using rewards from G=8 sampled responses, defined as a weighted sum of NLG metrics, together with a gate contrastive loss.

3.1 Problem Formulation

An agent navigates a 3D scene by executing atomic actions from a discrete action space $\mathcal{A}$ of four primitives: forward +0.25 m, turn left -30°, turn right $+30^\circ$, and stop. At each timestep t, the agent receives an ego-centric RGB observation $f_t \in \mathbb{R}^{H\times W\times 3}$. A trajectory of length T is thus:

$$\tau = \big((f_1, a_1),\ (f_2, a_2),\ \ldots,\ (f_T, a_T)\big). \tag{1}$$

The *Navigation Instruction Generation* (NIG) task requires producing a natural-language instruction $y = (w_1, \ldots, w_L)$ that describes τ faithfully enough for a human or autonomous follower to reproduce the route:

$$P(y \mid \tau) = \prod_{l=1}^{L} P(w_l \mid w_{<l},\ \tau)\,. \tag{2}$$

3.2 Preliminaries

Backbone Model. We build on **Qwen3-VL-8B** [18]. Its ViT-based visual encoder partitions each input image into non-overlapping patches forming a spatial token grid of shape $(h_p \times w_p)$ with hidden dimension d_{vit}; critically, each patch token i retains a fixed spatial position (r_i, c_i) throughout all layers, enabling the pixel-accurate one-to-one correspondence that VTMod exploits. For N keyframes, each frame is encoded independently and projected to the language model's hidden dimension by a shared MLP connector, with per-image 2D RoPE encodings preserving spatial and temporal order. On the language side, the system prompt and interleaved action snippets are tokenized into decoder input embeddings and combined with the projected visual tokens in a single multimodal sequence. The decoder then generates the navigation instruction autoregressively.

Multimodal input format. The model receives a single interleaved sequence: a task-specific system prompt $\mathcal{P}$ is followed by alternating keyframe images (I) and textual action snippets:

$$\mathcal{P},\ I_1,\ [\text{Action 1: } d_1],\ I_2,\ [\text{Action 2: } d_2],\ \ldots$$

where each d_k is a natural-language description of the physical displacement or rotation at that step (e.g. "go forward 1.25 m", "turn left 90°"). This interleaved layout preserves the temporal alignment between visual observations and physical actions, allowing the decoder to attend to the relevant image–action pair at each generation step.

3.3 Event-Driven Trajectory Compression

Level 1: Action Run-Length Encoding (RLE). Let the raw action sequence be $\mathbf{a} = (a_1, a_2, \ldots, a_T)$ with $a_t \in \mathcal{A}$. RLE merges consecutive identical actions into *segments*:

$$s_k = (\text{type}_k,\ n_k), \quad \text{where } a_t = \text{type}_k \text{ for } n_k \text{ consecutive steps.} \tag{3}$$

Each segment maps to a physical displacement ($n_k \times 0.25$ m forward) or rotation ($n_k \times 30^\circ$), producing the compressed sequence $\mathbf{S} = (s_1, \ldots, s_K)$ with $K \ll T$. RLE is lossless with respect to navigational semantics: every physical displacement and rotation is exactly preserved in $\mathbf{S}$. To avoid overly long straight-line events, we further split any forward segment whose displacement exceeds 4.0 m (i.e., $n_k > 16$) into two shorter forward segments of approximately equal length.

Formally, for a forward segment $s_k = (\text{fwd}, n_k)$ with $n_k > 16$, we replace it with

$$(\text{fwd}, \lfloor n_k/2 \rfloor),\ (\text{fwd}, \lceil n_k/2 \rceil).$$

All other segment types remain unchanged. This preserves the total displacement while preventing excessively long forward motion from dominating a single event representation.

Level 2: Small-Step Combo Grouping. A segment s_k is considered *small-step* if it corresponds to either a short forward displacement of 0.25 m or 0.5 m ($n_k \in \{1, 2\}$) or a single-step rotation of 30° ($n_k = 1$). Consecutive small-step segments are merged into a *combo event*

$$e_c = (s_k, s_{k+1}, \ldots, s_{k+M-1}),$$

with $M \leq 6$. This grouping captures fine-grained turn-and-advance manoeuvres as a single semantic unit, preventing them from being fragmented into isolated snippets that may appear spatially incoherent to the language model. For example, a short forward step followed by a left turn and another short forward step is grouped into a single combo event: "go forward 0.25 m / turn left 30° / go forward 0.5 m".

Level 3: Dynamic Priority-based Compression (DPC). When the number of retained frames after Levels 1–2 still exceeds $F_{\max} = 32$, DPC selects the most informative keyframes by a joint visual–geometric priority score. For each adjacent event-frame pair (i, j):

$$J(i, j) = \lambda_{\text{vis}}\, \Delta_{\text{vis}}(i, j) + \lambda_{\text{geo}}\, \Delta_{\text{geo}}(i, j), \tag{4}$$

where Δ_{vis} measures frame-to-frame visual change in a frozen perceptual feature space, and Δ_{geo} aggregates the cumulative displacement and rotation magnitude between event boundaries. A high J score indicates a visually or geometrically significant transition that should be retained; event frames at boundaries with the *lowest* J scores are progressively removed until $|\mathcal{F}| \leq F_{\max}$, while always preserving the initial frame and the current observation frame. The final retained frame set is:

$$\mathcal{F} = \{f_1, f_T\} \cup \{f_{e_1}, f_{e_2}, \ldots, f_{e_{N'}}\}, \tag{5}$$

where f_1 is the initial observation, f_T is the current observation frame, e_i denotes the index of the i-th retained event boundary, and N' is the number of retained event frames.

3.4 Visual Trajectory Prompt Rendering and Data Curation

Overlay components. For each retained keyframe, a structured VTP is rendered in the corresponding ego-centric view to visualize the local trajectory state. Three complementary components provide exhaustive spatial coverage across the canonical navigation states (advancing, turning, approaching goal):

- **Path ribbon**: a colour ribbon tracing the upcoming route in the current view. Because the raw path is a jagged polyline induced by 0.25 m atomic forward steps, we smooth it in 2D through densified interpolation, Chaikin corner cutting, and two rounds of Gaussian smoothing. A final occlusion check is then applied to prevent smoothed points from drifting behind walls. Used when the current event is not a pure rotation; degrades gracefully to a short stub when fully occluded.
- **Rotation arrow**: a curved left/right arc annotated with the turn angle in degrees. Used when the current event is a pure rotation.
- **Endpoint marker**: a landmark flag indicating the goal location in the current view. Rendered near trajectory end-frames when the goal is unoccluded.

Mask representation. Each VTP overlay is stored as a three-channel binary semantic mask $\mathbf{M} \in \{0, 1\}^{3\times H\times W}$, where channel C_0 encodes the ribbon, C_1 the rotation arrow, and C_2 the endpoint marker. Representing overlays as independent binary channels eliminates colour ambiguity and lets the VTP Encoder learn channel-specific spatial patterns, a factorisation that would be conflated in a mixed-colour image. This mask is the direct input to the VTP Encoder; the coloured PNG visualisation is an artefact used only for qualitative inspection.

Task-specific prompting strategy. We design separate system prompts for R2R-CE and RxR-CE to match their distinct annotation styles: the R2R-CE prompt targets concise instructions (15–45 words) emphasising landmark references and a precise stop location, while the RxR-CE prompt elicits richer step-by-step narrations (50–120 words) with explicit orientation and transition cues. Both prompts share a critical *use-but-don't-mention* constraint: the model is informed that the input images contain grounded navigation cues (path ribbons, turn indicators, goal markers) and is instructed to leverage these overlays for path inference, yet is explicitly prohibited from referencing them in the generated instruction. This design ensures VTP functions as an implicit geometric prior that improves spatial grounding without leaking rendering artefacts into the output text.

GPT-based quality filtering. Instruction candidates are scored by GPT against a rubric covering four dimensions: (i) *directional accuracy*: do described turns and path shape match the VTP overlay?; (ii) *landmark specificity*: are salient visual features referenced?; (iii) *distance plausibility*: do distance expressions correspond to actual trajectory length?; and (iv) *linguistic fluency*. Only samples exceeding quality threshold τ_{GPT} are retained, removing hallucinated or spatially imprecise instructions and ensuring the training corpus maintains consistent geometric fidelity.

3.5 Visual Trajectory Modulator Design

VTP Encoder. Let $\mathbf{M} \in \{0, 1\}^{3\times H\times W}$ denote the binary semantic mask with $\mathbf{F}^{(0)} = \mathbf{M}$. The VTP Encoder applies three stride-2 convolutional blocks, each comprising a convolution, group normalisation, and GELU activation, with output channel dimensions $C_l \in \{64, 128, 256\}$:

$$\mathbf{F}^{(l)} = \text{GELU}\Big(\text{GN}\Big(\text{Conv}^{(l)}_{s=2}\big(\mathbf{F}^{(l-1)}\big)\Big)\Big), \quad l = 1, 2, 3. \tag{6}$$

The hierarchical stride-2 design progressively expands the receptive field so that each spatial position encodes not only its local overlay pixel but also the surrounding trajectory context. A subsequent 1×1 convolution projects to dimension $d_v = 384$, and adaptive average pooling aligns the spatial resolution to the ViT patch grid, yielding:

$$\mathbf{V} = \text{AdpPool}\Big(\text{Conv}_{1\times1}\big(\mathbf{F}^{(3)}\big),\ (h_p, w_p)\Big) \in \mathbb{R}^{h_p w_p \times d_v}, \tag{7}$$

where each row $\mathbf{v}_i \in \mathbb{R}^{d_v}$ is in one-to-one spatial correspondence with the i-th ViT patch token.

Spatial modulation injection. At ViT layer $l^* = 7$, the hidden state $\mathbf{h}_i \in \mathbb{R}^{d_{\text{vit}}}$ of patch token i is updated as:

$$\mathbf{h}'_i = \mathbf{h}_i + \mathbf{g} \odot \text{LN}(\mathbf{W}\mathbf{v}_i), \tag{8}$$

where $\mathbf{W} \in \mathbb{R}^{d_{\text{vit}}\times d_v}$ is a learnable linear projection, LN denotes layer normalisation, $\odot$ denotes element-wise multiplication, and $\mathbf{g} \in \mathbb{R}^{d_{\text{vit}}}$ is a channel-wise learnable gate. We set $l^* = 7$ as the default injection site. As shown in Table 4, single-layer injection at layer 7 performs better than distributing injection across layers 7/15/23, so we use layer 7 in all experiments. The gate $\mathbf{g}$ is initialised to $\mathbf{g}_0 \approx \mathbf{0}$, implementing a *lazy activation*: the pre-trained ViT gradient

landscape is preserved at training onset, preventing catastrophic forgetting of visual priors while the backbone adapts to the new task during early SFT.

3.6 Training Strategy

Two-stage rationale. The two stages address complementary limitations. SFT teaches the task distribution (what high-quality navigation instructions conditioned on VTP-annotated keyframes look like) via maximum-likelihood estimation on GPT-filtered data. However, MLE treats all reference tokens equally regardless of navigational informativeness, and cannot directly optimise full-sequence metrics such as CIDEr or METEOR that are evaluated at test time. VT-GRPO then uses NLG reward signals to calibrate the gate $\mathbf{g}$'s injection strength while continuing to update the LLM backbone, sharpening which spatial channels of the VTP are amplified for reward-relevant generation.

Stage 1: Supervised Fine-Tuning (SFT). During SFT, all modules are trainable, including the ViT backbone, VTP Encoder, VTMod, MLP connector, and Qwen3 decoder. We use a dual learning-rate scheme: a higher rate η_{new} for newly initialised VTP/VTMod parameters and a lower rate η_{backbone} for the pretrained Qwen3-VL backbone. The SFT objective is standard next-token prediction cross-entropy over ground-truth instruction tokens.

Stage 2: VT-GRPO. Group Relative Policy Optimisation (GRPO) [15] is applied to refine instruction quality using NLG metrics as reward. For each input, $G = 8$ candidate completions are sampled. The reward for completion y is a weighted combination of automatic metrics:

$$r(y) = w_{B1}\,\text{B-1}(y) + w_{B4}\,\text{B-4}(y) + w_M\,\text{METEOR}(y) + w_R\,\text{ROUGE-L}(y) + w_C\,\text{CIDEr}(y), \tag{9}$$

with $w_{B-1} = w_{B-4} = w_M = 0.25$, $w_R = 0.20$, $w_C = 0.05$. The weights reflect the complementary coverage of the metrics: B-1 (BLEU-1) captures unigram precision, B-4 (BLEU-4) rewards multi-word phrase fidelity, METEOR additionally accounts for synonym overlap, and ROUGE-L measures longest-common-subsequence structural similarity; CIDEr is down-weighted because its large absolute scale would otherwise dominate the composite reward. A KL penalty $\beta = 0.04$ and clip ratio $\varepsilon = 0.2$ regularise the policy update. Let $\mathcal{L}_{\text{GRPO}}$ denote the corresponding GRPO objective induced by these rewards and regularizers.

Gate contrastive loss. Because $\mathcal{L}_{\text{GRPO}}$ averages over all G completions weighted by their respective advantages, positive and negative signals partially cancel, leaving the gate $\mathbf{g}$ with a diffuse gradient that is insufficient for precise calibration. We therefore introduce a contrastive loss on the best (y^+) and worst (y^-) completions within each group:

$$\mathcal{L}_{\text{gate}} = -\log\,\sigma\left(\frac{\bar{\ell}(y^+) - \bar{\ell}(y^-)}{\tau_g}\right), \tag{10}$$

where $\bar{\ell}(y)$ is the mean per-token log-probability and $\tau_g = 1.0$. Unlike the group-averaged GRPO signal, this loss provides a focused contrastive gradient that directly pushes $\mathbf{g}$ to amplify VTP channels correlated with higher-reward generations and suppress those correlated with lower-reward ones. The total training objective is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{GRPO}} + \alpha_{\text{gate}}\,\mathcal{L}_{\text{gate}}, \quad \alpha_{\text{gate}} = 0.05. \tag{11}$$

During VT-GRPO, the ViT backbone, VTP Encoder, and modulator projection layers are frozen; the LLM backbone and the gate vector $\mathbf{g}$ remain trainable. Detailed hyperparameters for both stages are reported in §4.1.

4 Experiments

4.1 Implementation Details

Hardware. All experiments are conducted on 8×NVIDIA H200 GPUs with DeepSpeed ZeRO-2 parallelism.

Training data. Training data consists of GPT-quality-filtered VTP-annotated trajectories (score ≥ 6 on a 10-point rubric) drawn exclusively from the Train splits of R2R-CE and RxR-CE.

Input representation. Each observation is represented as a single wide-aspect egocentric image, with VTP overlays rendered only in the central region. Trajectories are compressed to at most $F_{\max} = 32$ keyframes using the event-driven pipeline (§3.3).

SFT hyperparameters. 3 epochs; batch size 1; gradient accumulation 12; base LR 3×10^{-5} (backbone), VTMod LR 5×10^{-4}; 10% linear warm-up. Training takes approximately 9 hours on the above hardware.

VT-GRPO hyperparameters. 1 epoch; LR 10^{-6}; gradient accumulation 4; group size $G = 8$; KL $\beta = 0.04$; clip $\varepsilon = 0.2$; top-$p = 0.9$; gate-contrastive weight $\alpha_{\text{gate}} = 0.05$. Training takes approximately 30 hours on the above hardware.

4.2 Evaluation Benchmarks and Metrics

R2R-CE [9] extends the Room-to-Room benchmark [1] to the continuous-action Habitat simulator with photorealistic Matterport3D (MP3D) scenes. The agent navigates via four atomic actions and receives only a raw ego-centric RGB stream, with no navigation graph or pre-built map available. The *Val Unseen* split covers environments entirely held out from training, providing a stringent test of instruction generalisation to novel scenes. Since VTInstructor is trained exclusively on the Train splits of R2R-CE and RxR-CE, neither the environments nor the reference instructions in *Val Unseen* have been seen during training.

RxR-CE [10] adapts the Room-across-Room benchmark to the same Habitat continuous setting. RxR instructions are substantially longer and more spatially detailed than R2R (averaging over 70 words per instruction), with fine-grained descriptions of turn angles, landmark sequences, and relative distances. This verbosity and spatial precision make RxR-CE a more demanding benchmark for evaluating instruction generation quality. We report results on the English *Val Unseen* split.

Instruction metrics. We report BLEU-1/4, METEOR, ROUGE-L, CIDEr, and SPICE; higher is better for all. These metrics collectively capture n-gram precision, recall, synonym overlap, sequential similarity, and semantic propositional content.

Navigation metrics. For downstream experiments (§4.5), we report the standard VLN-CE metrics [1]: Success Rate (SR), Oracle Success Rate (OSR), SPL, and Navigation Error (NE, in metres; lower is better), all with a 3 m success threshold.

4.3 Main Results

Tables 1 and 2 present the main results. Table 1 compares VTInstructor against proprietary and open-source models on R2R-CE

Table 1: Navigation instruction generation results on R2R-CE and RxR-CE *Val Unseen*.

Method	R2R-CE *Val Unseen*						RxR-CE *Val Unseen*					
	BLEU-1	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE	BLEU-1	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE
Proprietary models												
Qwen3.5-Plus	0.557	0.131	0.223	0.387	0.137	0.180	0.527	0.080	0.157	0.254	0.033	0.156
GPT-5.4	0.482	0.075	0.197	0.348	0.078	0.144	0.493	0.056	0.146	0.241	0.029	0.143
Gemini-3.1-Pro-Preview	0.631	0.166	0.209	0.392	0.203	0.167	0.393	0.073	0.128	0.242	0.029	0.135
xAI Grok 4	0.486	0.084	0.207	0.354	0.073	0.160	0.430	0.049	0.181	0.249	0.014	0.144
Claude Opus 4.6	0.484	0.084	0.190	0.337	0.068	0.119	0.367	0.043	0.176	0.223	0.005	0.123
Open-source models												
Qwen3.5-397B-A17B	0.525	0.113	0.216	0.382	0.120	0.174	0.422	0.059	0.128	0.249	0.027	0.142
Kimi-K2.5	0.478	0.088	0.195	0.357	0.098	0.146	0.432	0.071	0.195	0.259	0.011	0.145
LLaVA-Video-7B-Qwen2	0.506	0.100	0.146	0.317	0.108	0.109	0.064	0.012	0.063	0.169	0.002	0.074
Qwen3-VL-8B-Instruct	0.524	0.085	0.175	0.334	0.147	0.137	0.366	0.053	0.119	0.230	0.022	0.113
Qwen3.5-9B	0.579	0.121	0.194	0.373	0.181	0.151	0.515	0.077	0.171	0.259	0.027	0.164
GLM-4.1V-9B-Thinking	0.490	0.080	0.156	0.321	0.130	0.115	0.074	0.014	0.054	0.144	0.003	0.064
GLM-4.6V	0.556	0.120	0.171	0.348	0.168	0.122	0.489	0.065	0.143	0.242	0.027	0.145
Ours												
Qwen3-VL-8B (SFT-only)	0.720	0.282	0.232	0.485	0.484	0.201	0.630	0.201	0.210	0.357	0.060	0.162
VTInstructor	**0.765**	**0.320**	**0.263**	**0.511**	**0.560**	**0.245**	**0.774**	**0.308**	**0.265**	**0.431**	**0.142**	**0.206**

Table 2: Cross-setting comparison with prior instruction generation methods. Prior methods are developed in discretized panoramic viewpoint-graph environments with privileged 36-view observations, whereas VTInstructor operates in a more challenging continuous environment with only ego-centric RGB input.

Method	R2R / R2R-CE *Val Unseen*						RxR / RxR-CE *Val Unseen*					
	BLEU-1	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE	BLEU-1	BLEU-4	METEOR	ROUGE-L	CIDEr	SPICE
Prior methods on R2R / RxR (discretized viewpoint-graph environments)												
BT-speaker [5]	0.658	0.250	0.209	0.440	0.391	0.178	0.311	0.064	0.160	0.243	0.022	–
EDrop-speaker [17]	0.660	0.260	0.215	0.455	0.413	0.184	0.294	0.056	0.143	0.253	0.027	–
CCC-speaker [19]	0.679	0.254	0.226	0.456	0.401	0.183	0.266	0.042	0.113	0.206	0.016	–
Lana [21]	0.689	0.260	0.219	0.463	0.419	0.194	0.314	0.111	0.126	0.267	0.045	–
Lana+ [21]	0.692	0.262	0.220	0.462	0.424	0.199	0.308	0.115	0.125	0.268	0.046	–
C-Instructor [8]	0.711	0.263	0.237	0.469	0.450	0.211	0.366	0.124	0.181	0.301	0.053	–
BEVInstructor [3]	0.699	0.264	0.230	0.467	0.449	0.208	0.351	0.112	0.174	0.292	0.043	–
MapInstructor [4]	0.715	0.285	0.234	0.485	0.490	0.209	0.411	0.159	0.191	0.323	0.057	–
Ours on R2R-CE / RxR-CE (continuous environments)												
VTInstructor	**0.765**	**0.320**	**0.263**	**0.511**	**0.560**	**0.245**	**0.774**	**0.308**	**0.265**	**0.431**	**0.142**	**0.206**

and RxR-CE *Val Unseen*. Table 2 further compares with prior instruction generation methods developed in discretized panoramic viewpoint-graph environments.

Comparison with existing models (Table 1). All baseline models are evaluated under a few-shot setting: each prompt includes a small number of high-quality reference instructions sampled from the training split that exemplify the target annotation style of R2R-CE or RxR-CE, ensuring that every model receives sufficient task context before generation. VTInstructor surpasses all baselines by a large margin across both benchmarks. On R2R-CE, VTInstructor achieves a CIDEr of 0.560, outperforming the strongest model Gemini-3.1-Pro-Preview (0.203) by +0.357; BLEU-4 reaches 0.320 versus 0.166 for Gemini, an improvement of over 90%. On RxR-CE, the gap widens further: most baselines obtain CIDEr below 0.03, while VTInstructor reaches 0.142. Notably, LLaVA-Video-7B and GLM-4.1V-9B-Thinking nearly collapse on RxR-CE (BLEU-1 of 0.064 and 0.074, respectively), suggesting that these models struggle with the longer observation sequences required by RxR trajectories. An interesting observation is that proprietary models do not consistently outperform smaller open-source models on this task—for instance, GPT-5.4 (CIDEr 0.078 on R2R-CE) falls below Qwen3.5-9B (0.181)—indicating that navigation instruction generation cannot be solved by model scale alone and benefits from task-specific training with spatial grounding.

Cross-setting comparison (Table 2). Despite operating in the more challenging continuous environment with only ego-centric RGB input, VTInstructor outperforms all prior discrete-setting methods that have access to panoramic images and pre-built navigation graphs. On R2R, VTInstructor surpasses the previous best MapInstructor in BLEU-4 (0.320 vs. 0.285), METEOR (0.263 vs. 0.234), and CIDEr (0.560 vs. 0.490). The advantage is even more pronounced on RxR, where VTInstructor achieves BLEU-4 of 0.308 compared to MapInstructor's 0.159 and CIDEr of 0.142 versus 0.057, suggesting that the VTP-based spatial grounding is particularly effective for generating the longer, more spatially detailed instructions characteristic of RxR. Crucially, R2R-CE/RxR-CE and R2R/RxR share identical navigation paths and reference instruction sets; the only difference lies in the observation and action spaces—prior methods consume privileged 36-view panoramic images at discrete graph nodes, whereas VTInstructor receives only a single ego-centric RGB stream with small-step actions. We follow the same evaluation protocol as C-Instructor [8], aggregating per-sub-path instructions to ensure metric-level comparability, so the scores in Table 2 are directly comparable despite the different input modalities. Figure 3 provides qualitative examples illustrating how VTInstructor generates spatially accurate instructions while the strongest baseline produces directional errors and hallucinated landmarks.

Table 3: Component-wise ablation on R2R-CE *Val Unseen*.

#	VTMod	EDTC	QF	BLEU-4	METEOR	ROUGE-L	SPICE
1				0.282	0.232	0.485	0.201
2			✓	0.286	0.241	0.492	0.220
3		✓	✓	0.288	0.242	0.493	0.223
4	✓	✓		0.302	0.250	0.500	0.232
5	✓	✓	✓	**0.308**	**0.256**	**0.504**	**0.238**

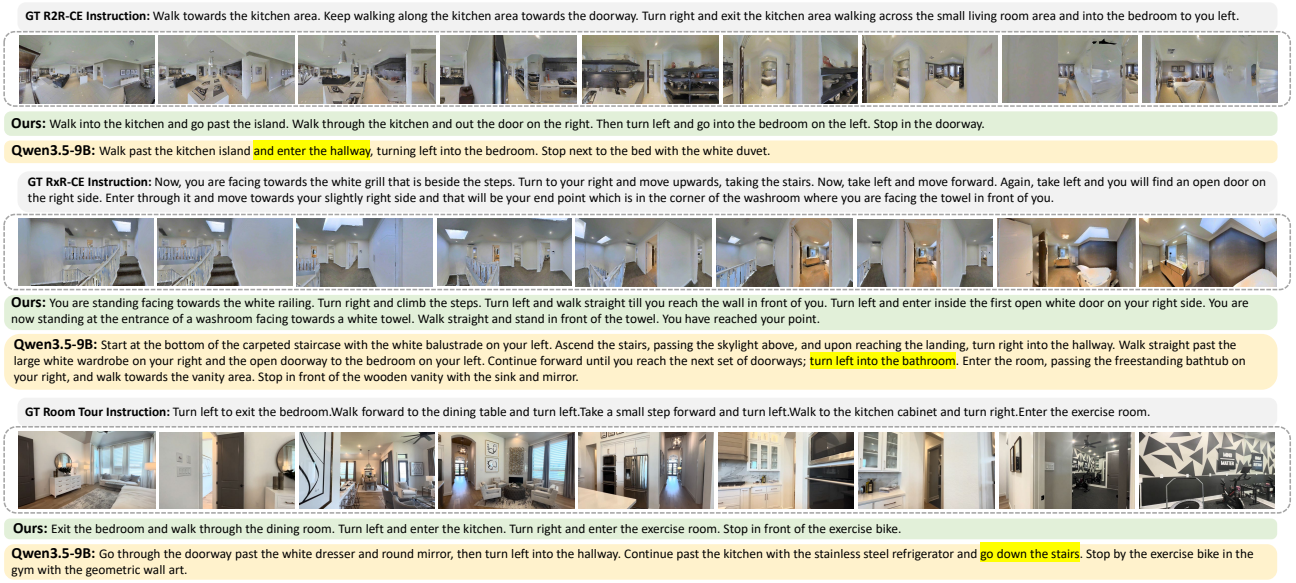

Figure 3: Qualitative comparison on R2R-CE *Val Unseen*. Each case shows the compressed keyframes with VTP overlays (top) and instructions from the ground truth, the strongest baseline, and VTInstructor (bottom). Yellow highlights mark factual errors in the baseline output (e.g. wrong turn direction, hallucinated landmarks). VTInstructor produces spatially accurate descriptions that align with the VTP-indicated trajectory structure.

Table 4: Design ablations on R2R-CE *Val Unseen*.

Configuration	BLEU-4	METEOR	ROUGE-L	SPICE
(a) EDTC: frame sampling				
Stride-4 (avg. 13.0 fr.)	0.738†	0.240	0.490	0.220
EDTC (avg. 10.5 fr.)	**0.743**†	**0.242**	**0.493**	**0.223**
(b) VTMod: injection layer				
Layer 7	**0.299**	**0.248**	**0.497**	**0.224**
Layers 7, 15, 23	0.295	0.244	0.491	0.221
(c) VTMod: injection design				
Cross-attention	0.295	0.244	0.494	0.222
Gated projection	**0.300**	**0.247**	**0.497**	**0.226**
(d) VT-GRPO: RL strategy				
Pure GRPO	0.314	0.260	0.508	0.240
VT-GRPO	**0.320**	**0.263**	**0.511**	**0.245**

† BLEU-1 reported for frame sampling comparison.

Table 5: Downstream navigation performance on R2R-CE *Val Unseen* with a frozen CorrectNav follower.

Instruction Source	SR↑	OSR↑	SPL↑	NE↓
Human Annotations				
Official instruction	61.6	67.2	53.3	4.53
Proprietary models				
Qwen3.5-Plus	48.6	61.8	39.0	5.70
GPT-5.4	35.9	51.4	28.6	6.14
Gemini-3.1-Pro-Preview	48.5	56.4	39.9	5.59
xAI Grok 4	45.2	63.9	33.3	6.15
Claude Opus 4.6	21.7	48.8	15.2	9.56
Open-source models				
Qwen3.5-397B-A17B	45.2	63.9	33.3	6.15
Kimi-K2.5	43.9	58.6	33.6	6.49
LLaVA-Video-7B-Qwen2	30.2	44.8	24.3	7.91
Qwen3-VL-8B-Instruct	33.4	46.8	25.9	7.09
Qwen3.5-9B	36.2	45.2	29.1	6.82
GLM-4.1V-9B-Thinking	31.1	42.9	24.4	7.37
GLM-4.6V	29.9	40.6	24.0	7.11
Ours				
VTInstructor	**63.3**	**70.0**	**52.7**	**4.47**

4.4 Ablation Study

We conduct ablation experiments on R2R-CE *Val Unseen*, whose shorter reference instructions make component contributions easier to isolate.

Component-wise ablation. Table 3 progressively adds VTMod, EDTC, and quality filtering (QF). The base SFT model without any of these components (Row 1) achieves BLEU-4 of 0.282 and SPICE of 0.201. Adding VTMod yields the largest single improvement: comparing Row 3 to Row 5, BLEU-4 jumps from 0.288 to 0.308 (+0.020) and SPICE from 0.223 to 0.238 (+0.015), confirming that VTMod is the core contributor by injecting trajectory-grounded spatial cues into the vision encoder. The full model (Row 5) achieves BLEU-4 of 0.308, METEOR of 0.256, ROUGE-L of 0.504, and SPICE of 0.238, representing cumulative gains of +0.026, +0.024, +0.019, and +0.037 over the base model. These results are obtained solely with supervised fine-tuning (SFT), without applying the subsequent VT-GRPO optimization process.

Design ablations. As shown in Table 4, EDTC reduces the average frame count from 13.0 to 10.5 while improving all metrics over stride-4 sampling, confirming that event-driven selection retains more informative keyframes. Injecting VTMod at a single early layer (layer 7) outperforms distributing it across layers 7/15/23 (BLEU-4 0.300 vs. 0.295), as subsequent ViT layers can jointly refine the fused representation without redundant modulation. Gated patchwise projection outperforms cross-attention injection (BLEU-4 0.300 vs. 0.295), preserving spatial locality of VTP cues that cross-attention would dilute. Finally, VT-GRPO improves over pure GRPO (BLEU-4 0.314→0.320, SPICE 0.240→0.245), demonstrating that the gate contrastive reward yields more spatially grounded instructions.

Table 6: CorrectNav (LLaVA-Video-7B backbone) trained under two augmentation settings on R2R-CE *Val Unseen*.

Set.	Training Data	SR↑	OSR↑	SPL↑	NE↓
A	R2R-CE + RxR-CE (human)	0.45	0.52	0.44	6.20
B	Setting A + VTInstructor-generated	**0.48**	**0.54**	**0.46**	**5.85**

Table 7: CorrectNav (LLaVA-Video-7B backbone) trained under two augmentation settings on RxR-CE *Val Unseen*.

Set.	Training Data	SR↑	OSR↑	SPL↑	NE↓
A	R2R-CE + RxR-CE (human)	0.41	0.51	0.39	8.34
B	Setting A + VTInstructor-generated	**0.44**	**0.53**	**0.41**	**7.74**

Table 8: Human evaluation on real-world navigation videos (1–5 scale, mean±std over 3 annotators).

Method	Action	Landmark	Direction	Follow.
GPT-5.4	2.65 ± 0.08	3.55 ± 0.06	2.53 ± 0.10	2.29 ± 0.06
Qwen3-VL-8B	2.63 ± 0.12	3.65 ± 0.09	2.67 ± 0.05	2.49 ± 0.08
VTInstructor	**4.38 ± 0.05**	**4.21 ± 0.05**	**4.25 ± 0.12**	**4.35 ± 0.04**

4.5 Downstream Navigation Success Rate

NLG scores measure lexical similarity to reference instructions but do not directly reflect navigational utility. To bridge this gap, we sample all trajectories from R2R-CE *Val Unseen*, generate instructions with each model under the same prompt, and feed the resulting instructions to a frozen CorrectNav [23] follower. Table 5 reports navigation performance under each instruction source.

VTInstructor-generated instructions achieve SR of 63.3 and NE of 4.47 m, *surpassing even human-written instructions* (SR 61.6, NE 4.53 m) and closely matching their SPL (52.7 vs. 53.3). Among baselines, the best proprietary model Qwen3.5-Plus reaches only SR 48.6, lagging VTInstructor by nearly 15 percentage points. Open-source VLMs perform substantially worse (SR 29.9–36.2), and Claude Opus 4.6 obtains the lowest SR of 21.7. These results confirm that NLG metrics and downstream navigation performance are positively correlated, and that VTInstructor's spatially grounded instructions translate directly into improved follower behaviour.

4.6 Training Gain from Generated Data

We further assess whether VTInstructor-generated instructions provide greater training benefit than instructions from competing generators. A CorrectNav [23] follower (LLaVA-Video-7B [25] backbone) is trained from scratch under three data settings and evaluated on R2R-CE *Val Unseen* (Table 6) and RxR-CE *Val Unseen* (Table 7). Setting A uses only the original human-annotated R2R-CE and RxR-CE training data. Setting B augments Setting A with 20K instructions generated by our proposed VTInstructor from Table 5.

Compared to the human-only baseline (Setting A), Setting B improves SR from 0.45 to 0.48 (+3,pp) and reduces NE from 6.20 m to 5.85 m (−0.35 m) on R2R-CE *Val Unseen*, while also improving SR from 0.41 to 0.44 (+3,pp) and reducing NE from 8.34 m to 7.74 m (−0.60 m) on RxR-CE *Val Unseen*, demonstrating that VTInstructor-generated data provides meaningful training augmentation for downstream navigation agents.

4.7 Human Evaluation on Real-World Navigation Videos

Automatic NLG metrics capture lexical overlap with reference instructions but cannot assess whether a generated instruction would actually guide a human follower in the real world. To evaluate real-world applicability, we collect **50 room-tour video clips sourced from YouTube**, capturing diverse indoor environments (offices, corridors, and multi-room apartments) filmed from a first-person perspective. For each clip, we first extract ego-centric frames at a fixed interval. Taking the starting frame as the origin of the local coordinate system, we estimate per-frame depth and camera poses using the estimated depth together with camera parameter information, and derive the geometric relationships between consecutive frames. Based on these geometric cues, we apply our event-driven compression pipeline to obtain keyframes and render VTP overlays. VTInstructor takes the resulting keyframes with VTP overlays as input, whereas the two strongest baselines (GPT-5.4 and Qwen3-VL-8B-Instruct) receive only the cropped keyframes without any VTP overlays or additional geometric annotations, and are prompted to generate instructions from visual content alone.

Annotation protocol. Three human annotators independently rate each generated instruction on a 1–5 scores across four dimensions:

- **Action**: does the instruction correctly describe the sequence of movements and turns (e.g. step count, turn type)?
- **Landmark**: are visually distinctive objects or regions in the scene accurately referenced?
- **Direction**: does the instruction correctly convey directional cues such as heading, turn angles, and left/right orientation?
- **Followability**: could a naïve human follower reproduce the route using only this instruction?

Each annotator is shown the corresponding keyframe sequence (without VTP overlays) alongside the instruction being rated, so scores reflect the instruction's standalone informativeness rather than reliance on the overlay visualisation. Final scores are the mean across annotators.

Table 8 shows that VTInstructor achieves the highest scores on all four dimensions. The largest gains over baselines to appear on Action and Direction, as the VTP overlay provides explicit movement-sequence and turn-angle cues that general-purpose models lack. Landmark scores are closer across methods, since large VLMs already possess strong visual recognition for common indoor objects. The Followability dimension, which integrates all aspects into a holistic judgement, can show the clearest overall improvement, confirming that spatially grounded instructions translate to practical navigational utility beyond what automatic metrics capture.

5 Limitations and Future Work

The VT-GRPO reward is composed entirely of automatic NLG metrics computed against reference instructions. While this avoids the prohibitive cost of running a navigation follower in the loop, it means the policy is not directly optimised for navigational success rate. Future work could incorporate sparse follower feedback (e.g., success/failure signals from a lightweight frozen follower evaluated on a small trajectory buffer) as an additional reward term, more directly bridging instruction quality and downstream navigation performance.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sunderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments. In *CVPR*.

[2] Shizhe Chen, Pierre-Louis Guhur, Makarand Tapaswi, Cordelia Schmid, and Ivan Laptev. 2021. History Aware Multimodal Transformer for Vision-and-Language Navigation. In *NeurIPS*.

[3] Sheng Fan, Rui Liu, Wenguan Wang, and Yi Yang. 2024. Navigation Instruction Generation with BEV Perception and Large Language Models. (2024).

[4] Sheng Fan, Rui Liu, Wenguan Wang, and Yi Yang. 2025. Scene Map-based Prompt Tuning for Navigation Instruction Generation. (2025), 6898–6908.

[5] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-Follower Models for Vision-and-Language Navigation. In *NeurIPS*.

[6] Muraleekrishna Gopinathan, Martin Masek, Jumana Abu-Khalaf, and David Suter. 2024. Spatially-Aware Speaker for Vision-and-Language Navigation Instruction Generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Bangkok, Thailand, 13601–13614. doi:10.18653/v1/2024.acl-long.734

[7] Aishwarya Kamath, Peter Anderson, Su Wang, Jing Yu Koh, Alexander Ku, Austin Waters, Yinfei Yang, Jason Baldridge, and Zarana Parekh. 2022. A New Path: Scaling Vision-and-Language Navigation with Synthetic Instructions and Imitation Learning. *arXiv preprint arXiv:2210.03112* (2022).

[8] Xianghao Kong, Jinyu Chen, Wenguan Wang, Hang Su, Xiaolin Hu, Yi Yang, and Si Liu. 2024. Controllable Navigation Instruction Generation with Chain of Thought Prompting. In *ECCV*.

[9] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. 2020. Beyond the Nav-Graph: Vision-and-Language Navigation in Continuous Environments. In *ECCV*.

[10] Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. 2020. Room-Across-Room: Multilingual Vision-and-Language Navigation with Dense Spatiotemporal Grounding. In *EMNLP*.

[11] Jialu Li, Hao Tan, and Mohit Bansal. 2022. EnvEdit: Environment Editing for Vision-and-Language Navigation. In *CVPR*.

[12] Yanbang Li, Ziyang Gong, Haoyang Li, Xiaoqi Huang, Haolan Kang, Guangping Bai, and Xianzheng Ma. 2025. Robotic Visual Instruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12155–12165.

[13] Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. 2024. PIVOT: Iterative Visual Prompting Elicits Actionable Knowledge for VLMs. arXiv preprint arXiv:2402.07872.

[14] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. 2020. REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments. In *CVPR*.

[15] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300* (2024).

[16] Aleksandr Shtedritski, Christian Rupprecht, and Andrea Vedaldi. 2023. What Does CLIP Know about a Red Circle? Visual Prompt Engineering for VLMs. In *ICCV*.

[17] Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to Navigate Unseen Environments: Back Translation with Environmental Dropout. In *NAACL*.

[18] Qwen Team. 2025. Qwen3-VL Technical Report. arXiv preprint.

[19] Hanqing Wang, Wei Liang, Jianbing Shen, Luc Van Gool, and Wenguan Wang. 2022. Counterfactual Cycle-Consistent Learning for Instruction Following and Generation in Vision-Language Navigation. In *CVPR*.

[20] Su Wang, Ceslee Montgomery, Jordi Orbay, Vighnesh Birodkar, Aleksandra Faust, Izzeddin Gur, Natasha Jaques, Austin Waters, Jason Baldridge, and Peter Anderson. 2022. Less is More: Generating Grounded Navigation Instructions from Landmarks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] Xiaohan Wang, Wenguan Wang, Jiayi Shao, and Yi Yang. 2023. LANA: A Language-Capable Navigator for Instruction Following and Generation. In *CVPR*.

[22] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. 2023. Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V. arXiv preprint arXiv:2310.11441.

[23] Zhuoyuan Yu, Yuxing Long, Zihan Yang, Chengyan Zeng, Hongwei Fan, Jiyao Zhang, and Hao Dong. 2026. CorrectNav: Self-Correction Flywheel Empowers Vision-Language-Action Navigation Model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 18737–18745.

[24] Haitian Zeng, Xiaohan Wang, Wenguan Wang, and Yi Yang. 2023. KEFA: A Knowledge Enhanced and Fine-grained Aligned Speaker for Navigation Instruction Generation. *arXiv preprint arXiv:2307.13368*.

[25] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. 2024. LLaVA-Video: Video Instruction Tuning With Synthetic Data. In *NeurIPS*.

[26] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. 2025. TraceVLA: Visual Trace Prompting Enhances Spatial-Temporal Awareness for Generalist Robotic Policies. In *The Thirteenth International Conference on Learning Representations (ICLR)*.