模式分析与机器智能
工业和信息化部重点实验室
MIIT Key Laboratory of
Pattern Analysis & Machine Intelligence

ParNeC 模式识别与神经计算研究组
PAttern Recognition and NEural Computing

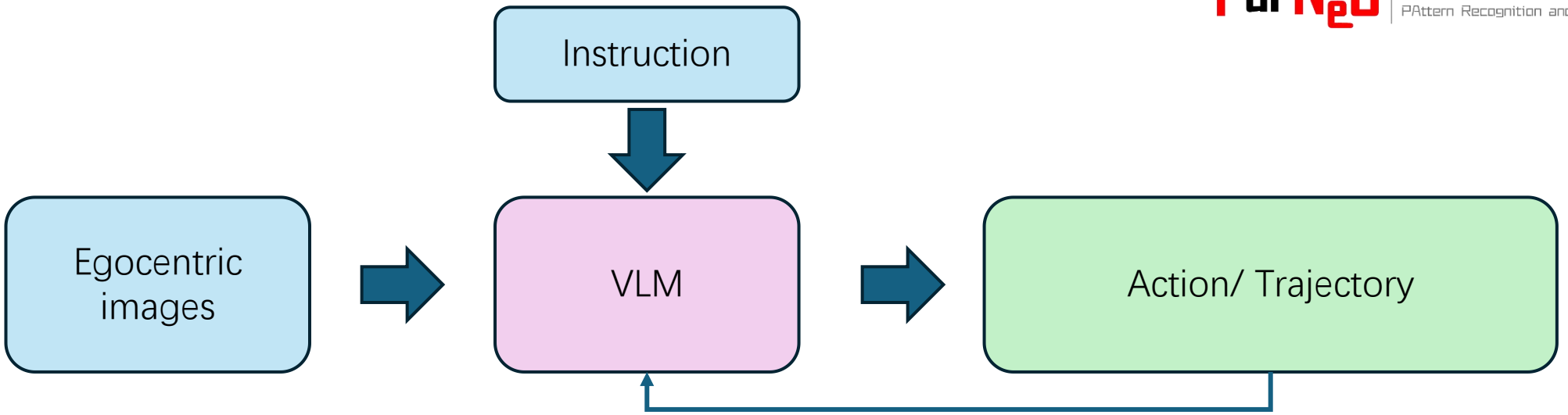
Recent Advances in Vision-and- Language Navigation

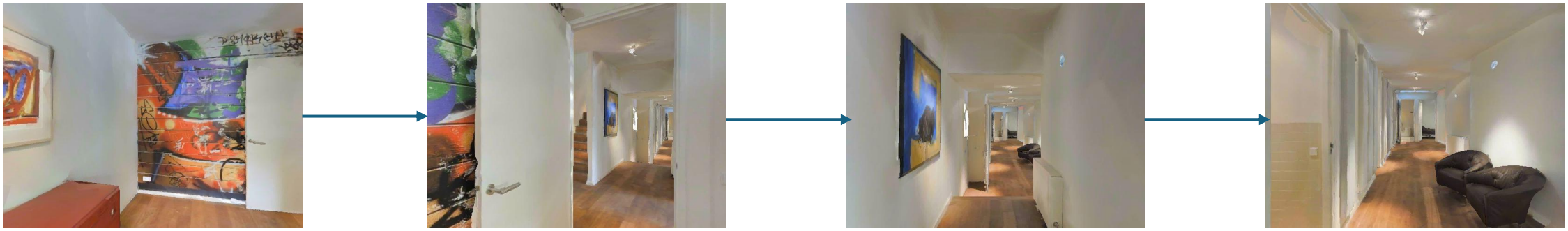
欣子豪

2026. 4. 20

Contents

VLN





Now you are facing towards the wall, slightly turn to your right, you can see a open door to your right, exit room through an open door and move forward, there is a wall art to the left side of the wall and few steps in-front of you, walk down the steps and move forward, there are two black couches to your right, that's your end point.



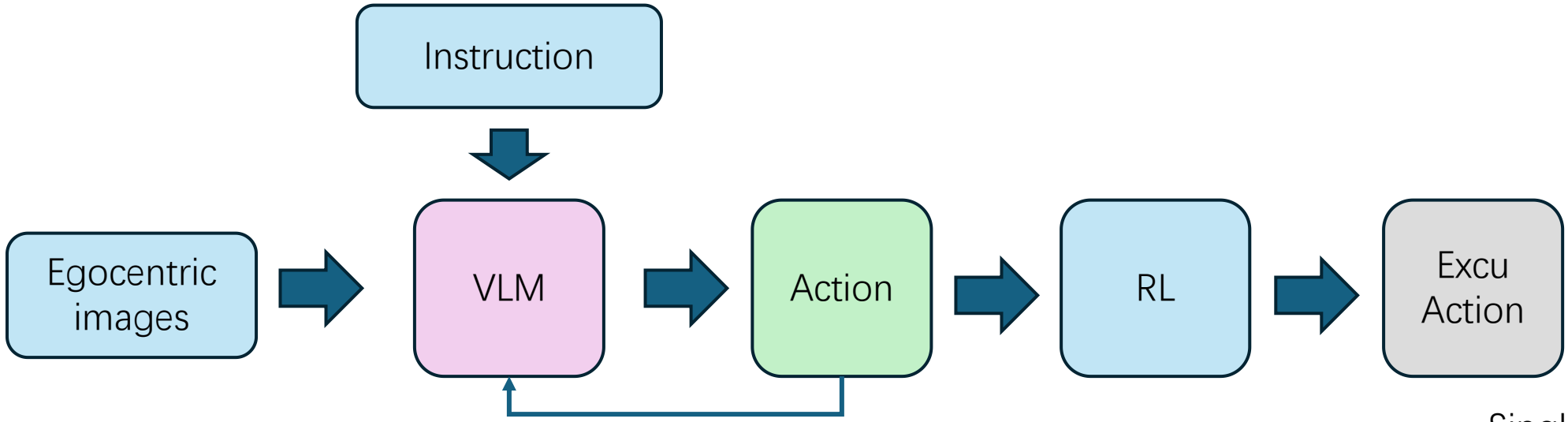

Now you are facing towards the wall, slightly turn to your right, you can see a open door to your right, exit room through an open door and move forward, there is a wall art to the left side of the wall and few steps in-front of you, walk down the steps and move forward, there are two black couches to your right, that's your end point.

What does VLM need to learn?

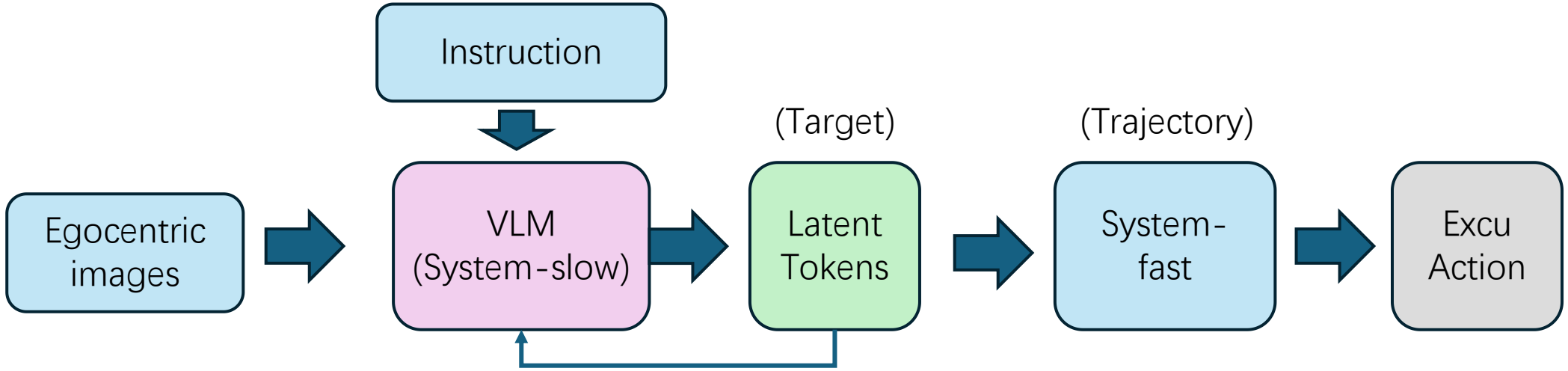
1. Depth estimation: How far is the wall/door?
2. Contextual semantic analysis: What exactly does "slightly" refer to in terms of perspective?
3. Long and short memory
4. Action Space

…

VLN

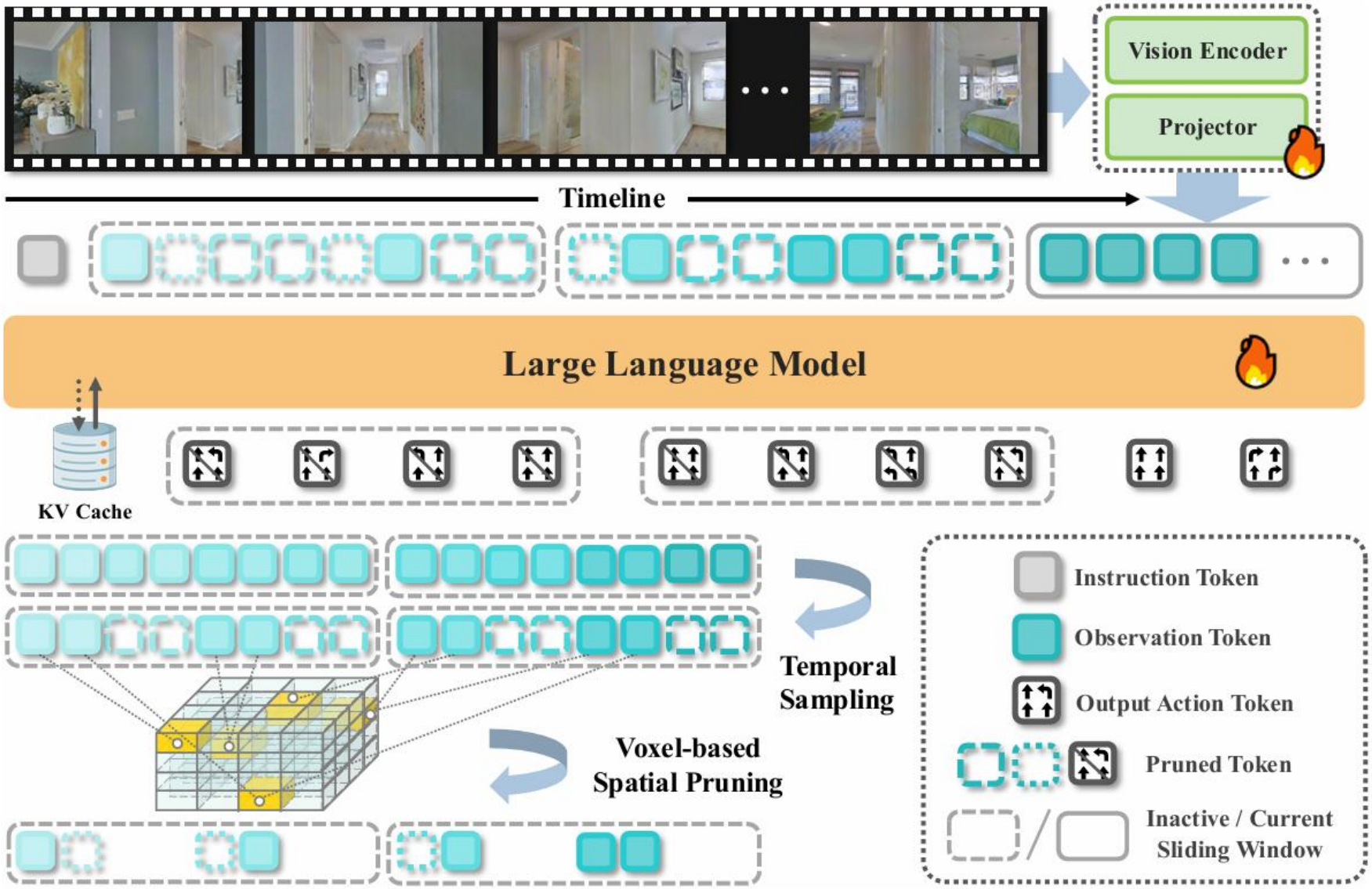


Single-System



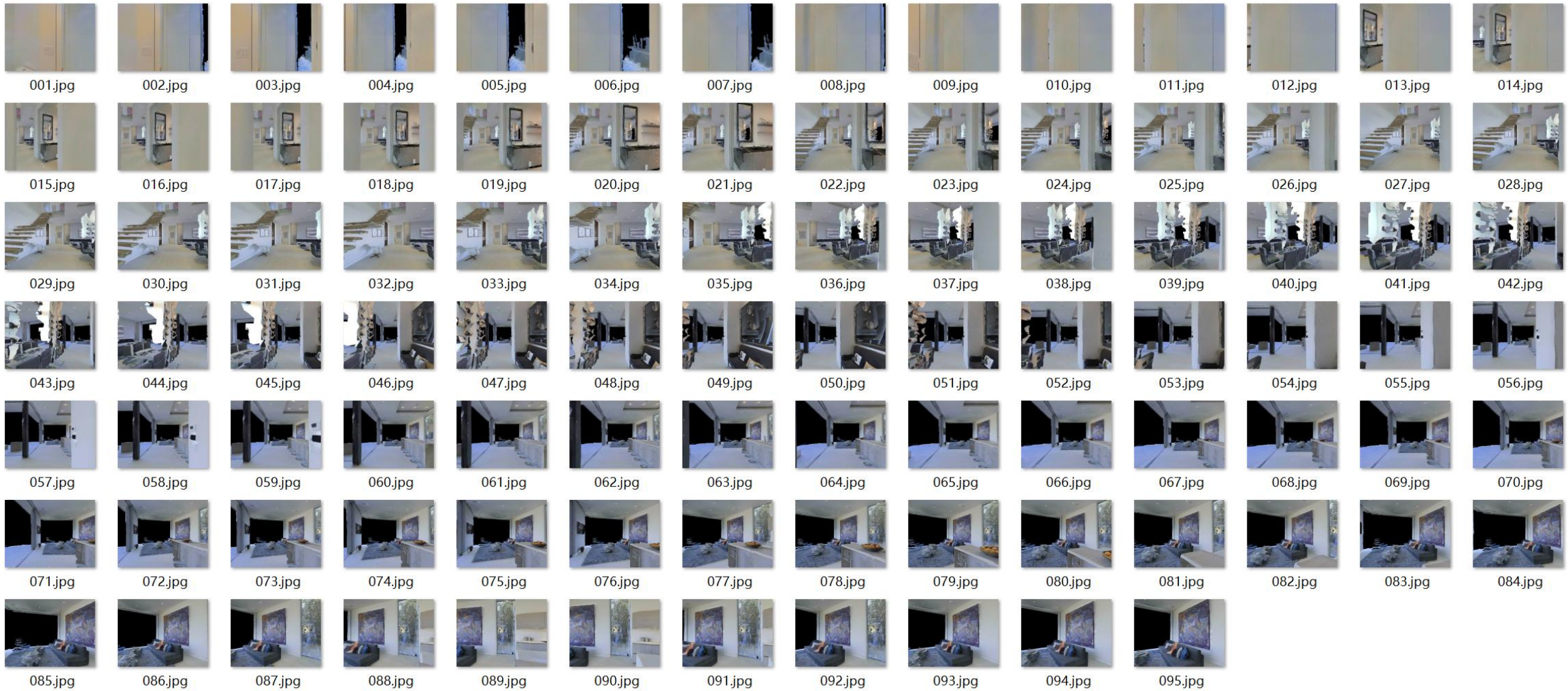
Dual-System

StreamVLN（Single-System, ICRA2026）



1. Long and short term memory
2. Positioning and planning navigation capability
3. Exploration and error correction

VLN – Long Term Memory



~15m path = ~100 ego-centric frames

StreamVLN（Single-System, ICRA2026）

1. Uniform Sampling
2. Voxel-based Cutting

Memory:

8 history frames

+

4 current frames

~4k tokens

Qwen2.5-VL-7B (BF16)

Vision Tower: 640 * 480 image = 46 * 34 patch = 1564 tokens

LLM (after merge): 640 * 480 image = 23 * 17 patch = 391 tokens

KV Cache = ~23 MB

Peak Inference Mem = Model Weight + Vision Tower Peak
+ LLM Prefill Peak + KV Cache + Allocator Reserve

↓

Hidden + QKV + MLP ≈ 100-300 MB / frame

VLN – Long Term Memory

Qwen2.5-VL-7B (BF16): *10 v.s. 100 frames*

Time of First Token (TTFT)

Vision Tower: 10x

LLM Prefill: ~90x

N = num_frames * 391 + text_tokens

Self-attn: O(N^2 · d)

Compute-bound

Time Per Output Token(TPOT)

KV Cache: 10x

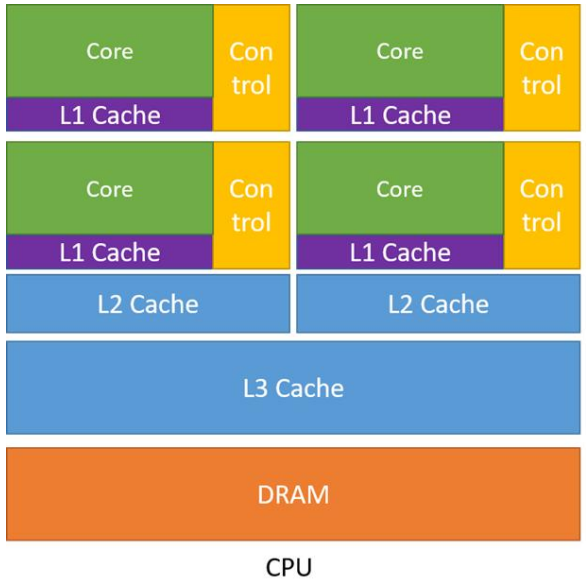
New Token Query (1 x d)

x

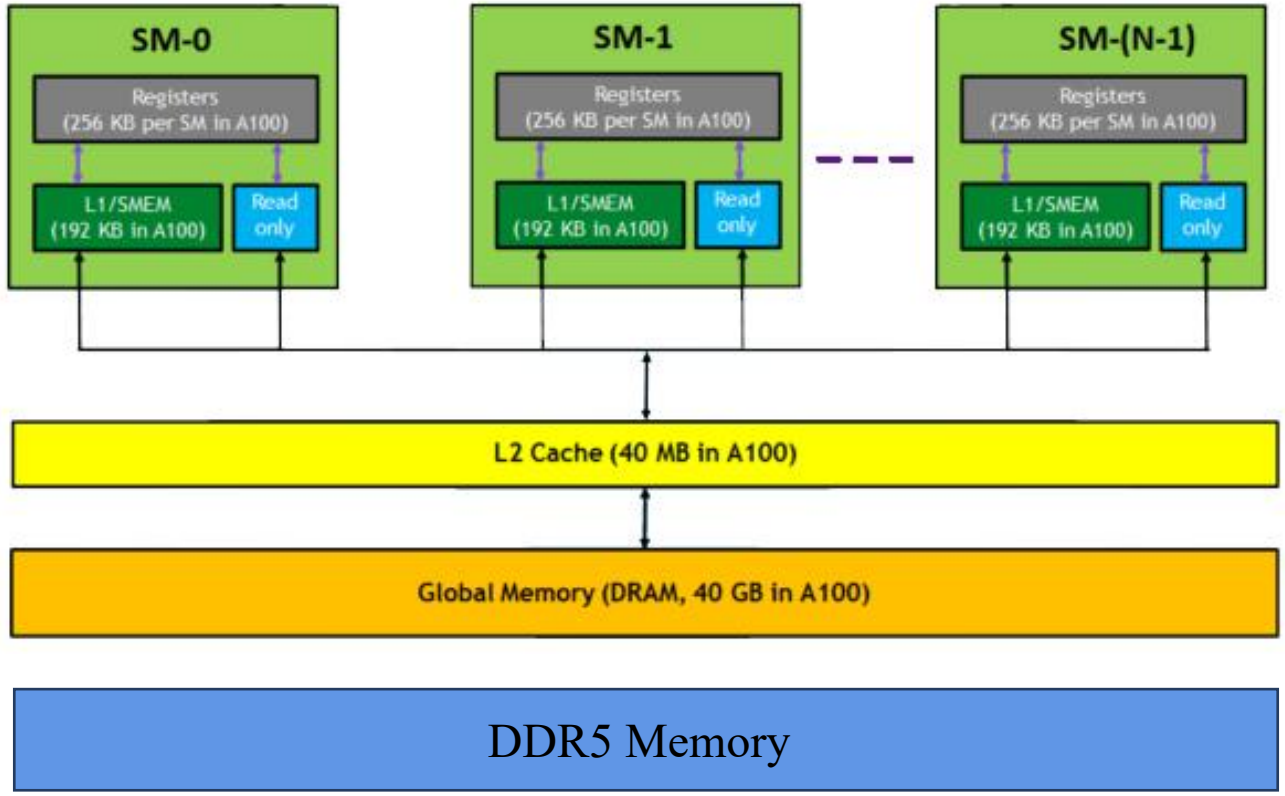
KV Cache (d x N)

Memory-bound

GPU Memory Hierarchy

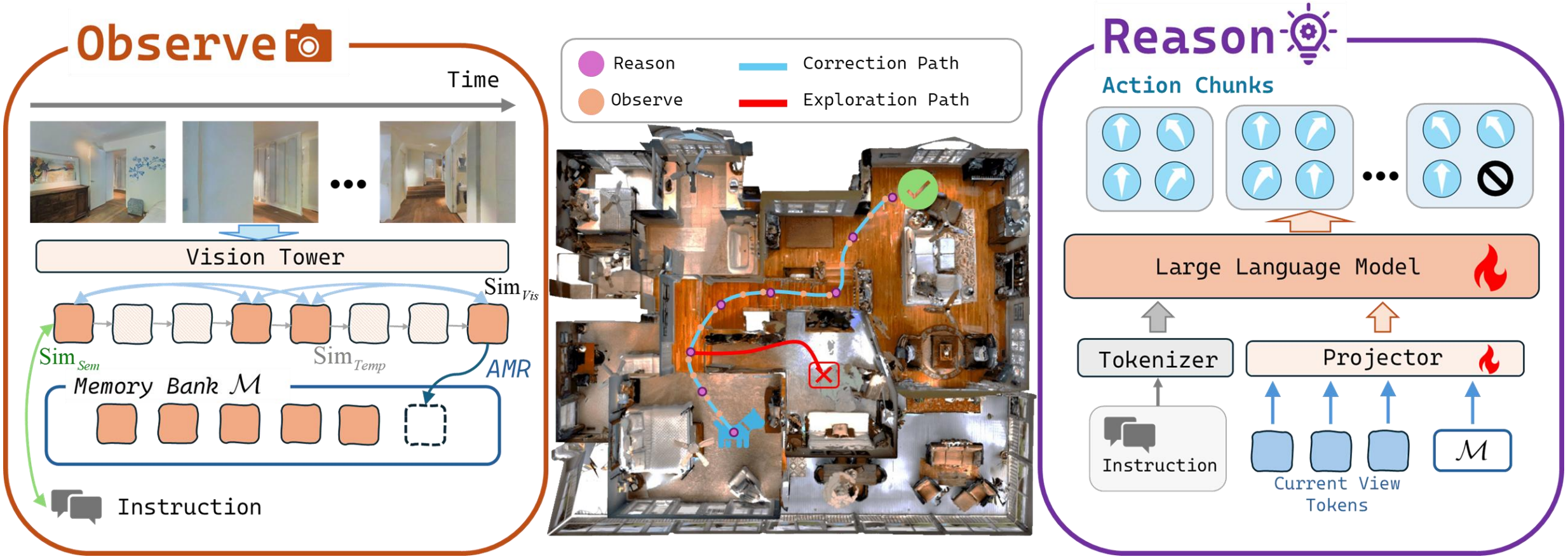


The GPU Devotes More Transistors to Data Processing



https://docs.nvidia.com/cuda/cuda-c-programming-guide/

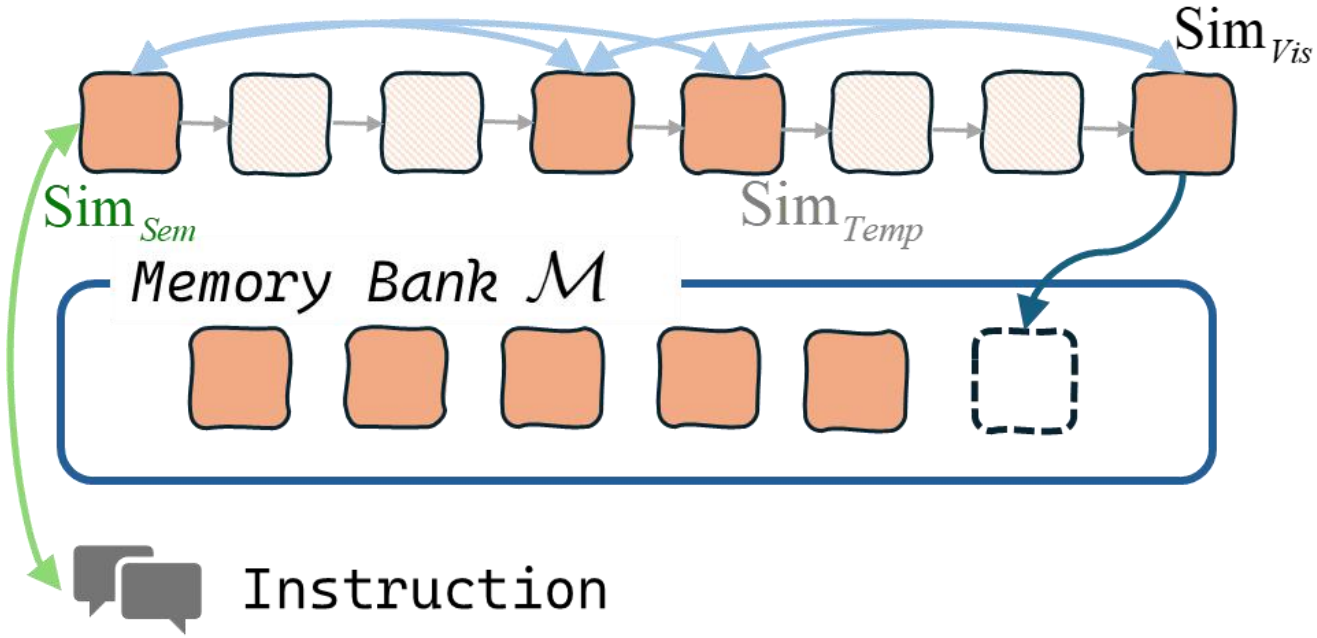
DecoVLN（Single-System, CVPR2026）



https://allenxinn.github.io/DecoVLN

DecoVLN（Single-System, CVPR2026）

Consider long-term memory as an optimization problem.



$$\mathrm{Sim}_{\mathrm{Vis}}(f, \mathcal{M}) = \max_{m \in \mathcal{M}} \left(\frac{\mathrm{embed}(f) \cdot \mathrm{embed}(m)}{\|\mathrm{embed}(f)\| \, \|\mathrm{embed}(m)\|} \right)$$

$$\mathrm{Sim}_{\mathrm{Sem}}(f, I) = \mathrm{sim}(\mathbf{e}_f, \mathbf{e}_I) = \frac{\mathbf{e}_f \cdot \mathbf{e}_I}{\|\mathbf{e}_f\| \, \|\mathbf{e}_I\|}$$

$$\mathrm{Sim}_{\mathrm{Temp}}(f, \mathcal{M}) = \frac{1}{\min_{m \in \mathcal{M}} |t_f - t_m| + \epsilon}$$

$$f^* = \arg \max_{f \in \mathcal{C} \setminus \mathcal{M}} [\lambda_R \cdot \mathrm{Sim}_{\mathrm{Sem}}(f, I) - (1 - \lambda_R) \cdot (w_V \cdot \mathrm{Sim}_{\mathrm{Vis}}(f, \mathcal{M}) + w_T \cdot \mathrm{Sim}_{\mathrm{Temp}}(f, \mathcal{M}))]$$

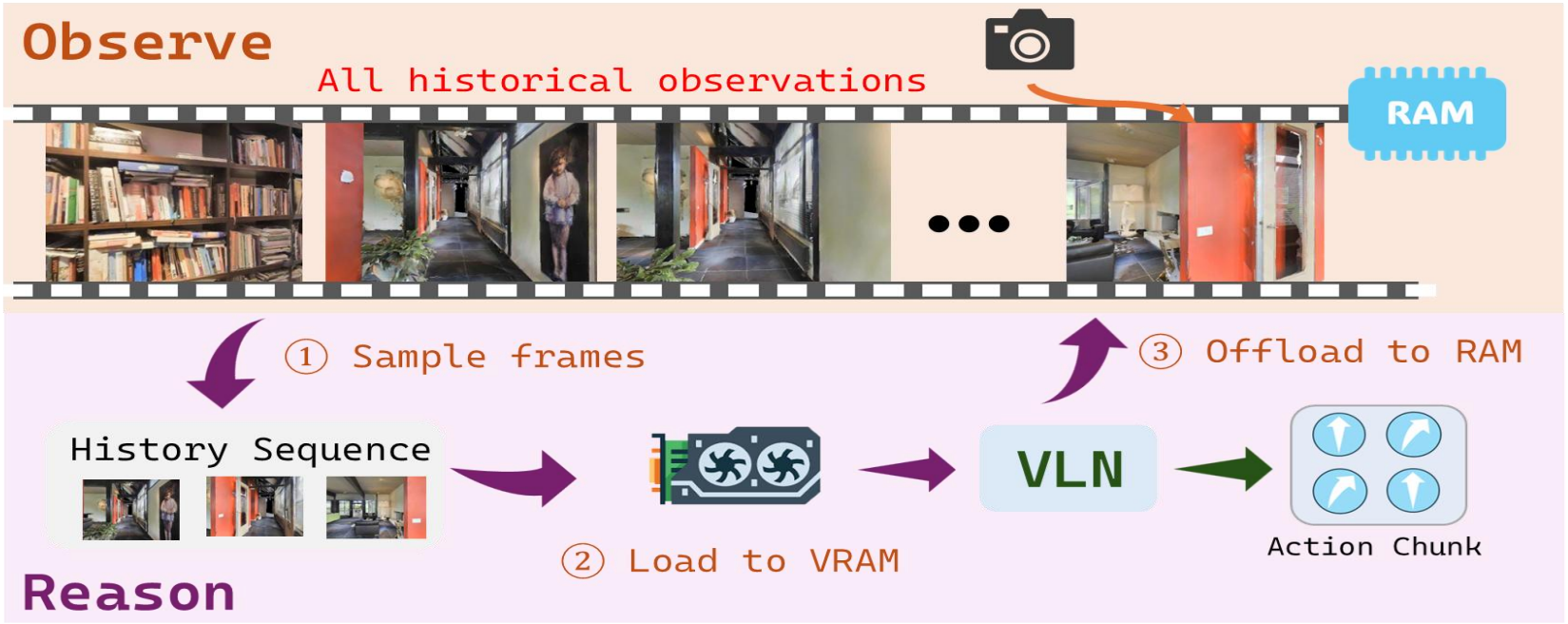
DecoVLN（Single-System, CVPR2026）

(a) Uniform Sampling

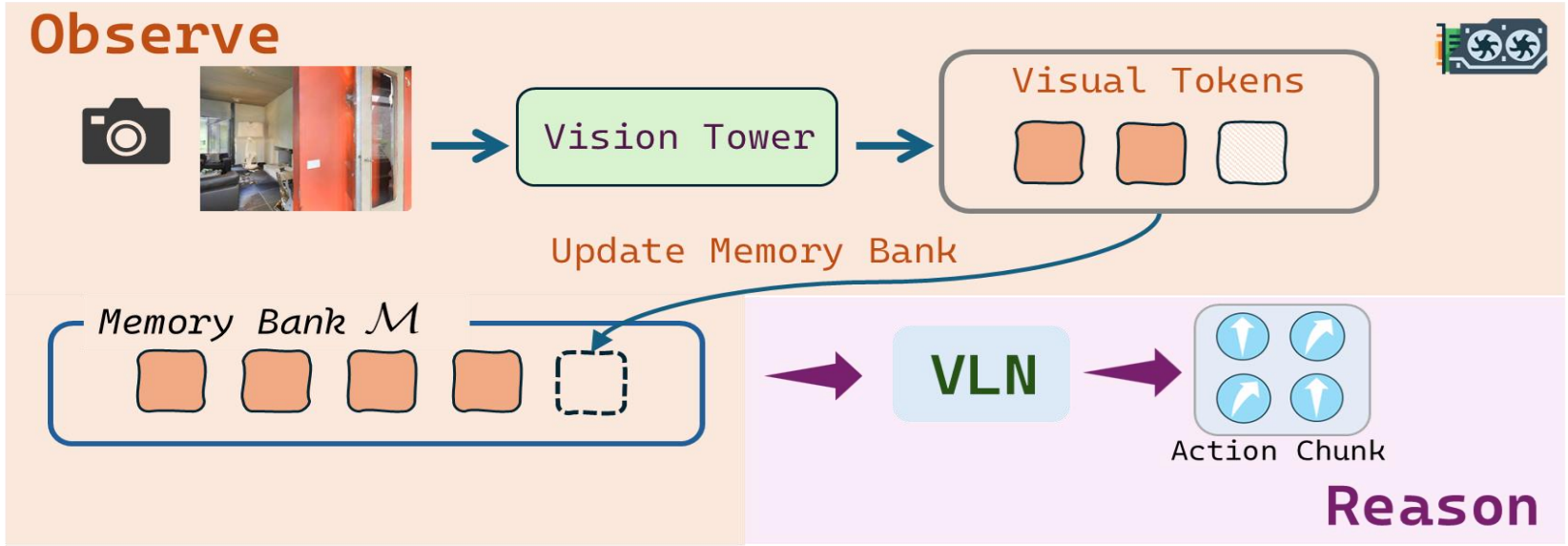
(b) Adaptive Memory Refinement

Now you are facing towards the wall, slightly turn to your right, you can see a open door to your right, exit room through an open door and move forward, there is a wall art to the left side of the wall and few steps in-front of you, walk down the steps and move forward, there are two black couches to your right, that's your end point.

DecoVLN（Single-System, CVPR2026）



Streaming Paradigm



DecoVLN

DecoVLN（Single-System, CVPR2026）

StreamVLN

DecoVLN

DecoVLN（Single-System, CVPR2026）

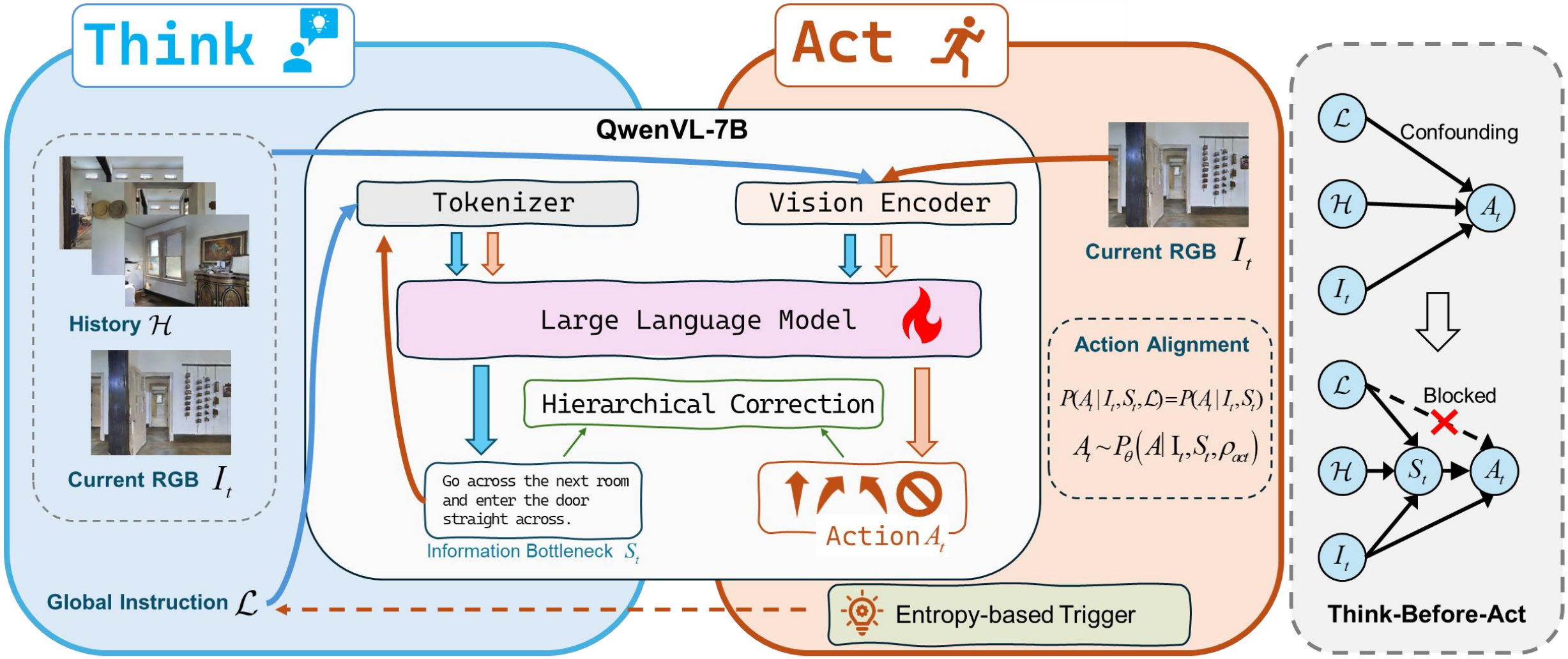
1. Expert Dataset: R2R, RxR (Stage 1)
2. Collect correction data using stage 1 model
3. Multi-Mode Dataset: Correction data + MM Datasets

Algorithm 1 Correction Data Collection

Require: Initial policy π_θ, expert policy π^*, dataset $\mathcal{D} = \{(P^i_{\text{exp}}, I^i)\}_{i=1}^{M}$, deviation threshold τ
Ensure: Correction dataset $\mathcal{D}_c$

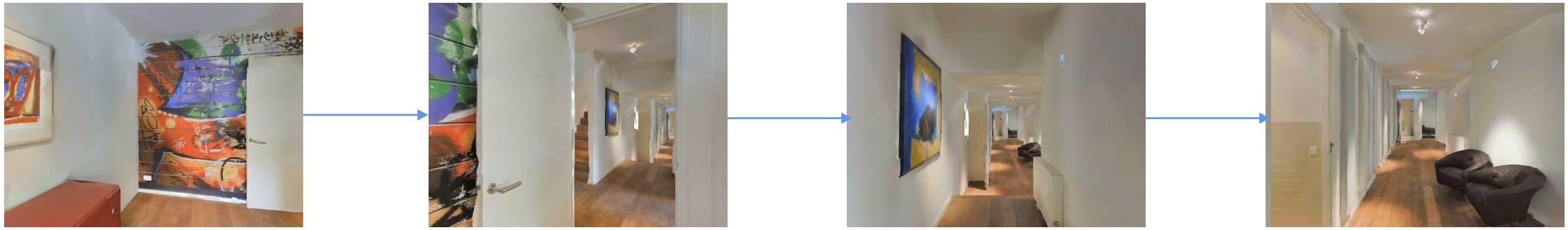
$\mathcal{D}_c \leftarrow \varnothing$ ▷ Initialize correction dataset
for all episode $(P_{\text{exp}}, I) \in \mathcal{D}$ **do**
  $P_{\text{exp}} \leftarrow \{s^*_0, s^*_1, \ldots, s^*_N\}$ ▷ Expert reference trajectory
  $H \leftarrow \varnothing$ ▷ Initialize memory state
  for $t = 0$ **to** $T_{\max}$ **do**
    $DM \leftarrow \min_{s^* \in P_{\text{exp}}} d_g(s_t, s^*)$ ▷ Compute deviation from expert trajectory
    if $0 < DM(s_t) \leq \tau$ **then**
      $a_t^{\text{exp}} \leftarrow \pi^*(s_t, P_{\text{exp}})$
      $\mathcal{D}_c \leftarrow \mathcal{D}_c \cup \{(s_t, a_t^{\text{exp}}, f_t)\}$ ▷ Store expert correction
    else if $DM(s_t) > \tau$ **then**
      break ▷ Abort episode due to large deviation
    end if
    $a_t \sim \pi_\theta(f_t, I, H)$ ▷ Roll out current policy
    $s_{t+1} \leftarrow \text{Env.Step}(a_t)$ ▷ Environment state update
    $f_{t+1} \leftarrow \text{Env.Observe}(s_{t+1})$ ▷ Acquire next observation
    $H \leftarrow \text{AMF}(H, f_{t+1})$ ▷ Adaptive memory refinement
    if target reached **or** episode failure **then**
      break ▷ Terminate episode
    end if
  end for
end for
return $\mathcal{D}_c$

IDEAL-VLN（Single-System）



Global Instruction Turn to your left and exit the room. Go across the next room and enter the door straight across. Stop once you enter the bathroom door.

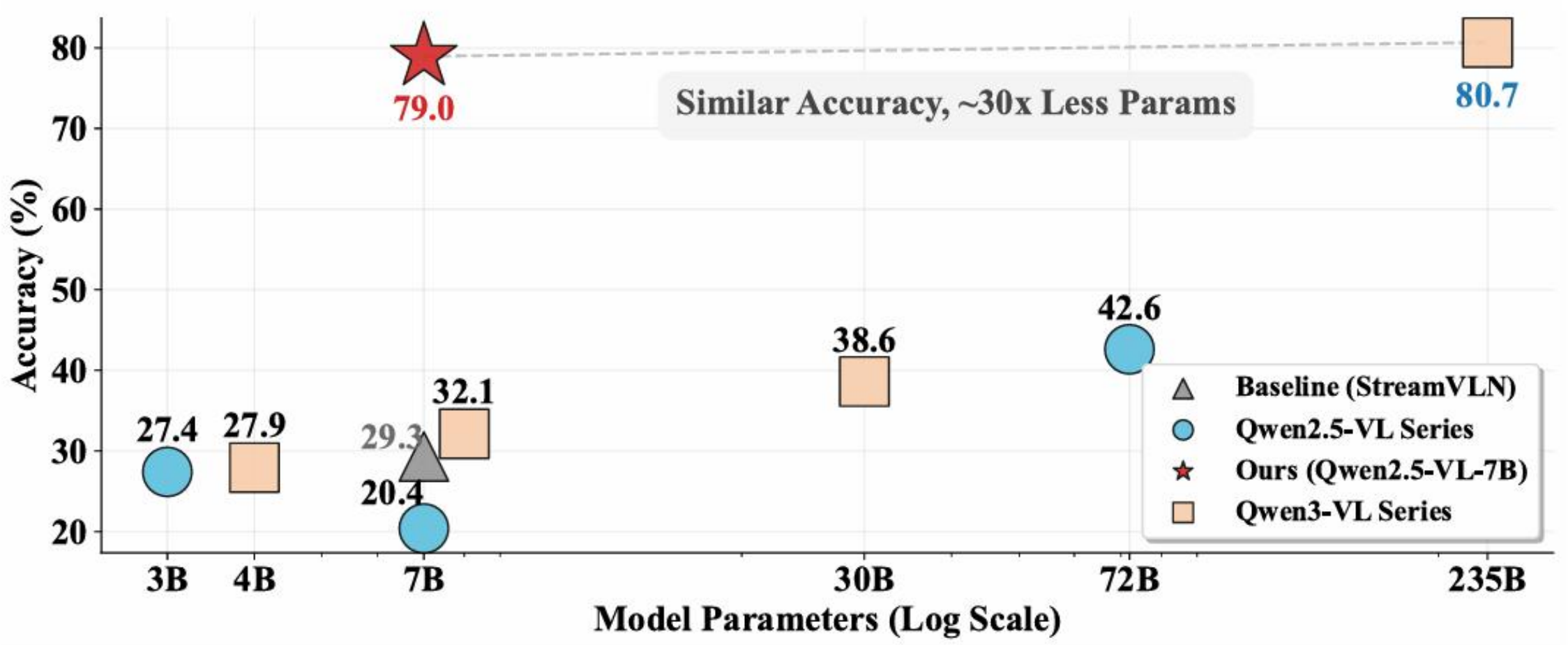
IDEAL-VLN（Single-System）



Now you are facing towards the wall, slightly turn to your right, you can see a open door to your right, exit room through an open door and move forward, there is a wall art to the left side of the wall and few steps in-front of you, walk down the steps and move forward, there are two black couches to your right, that's your end point.

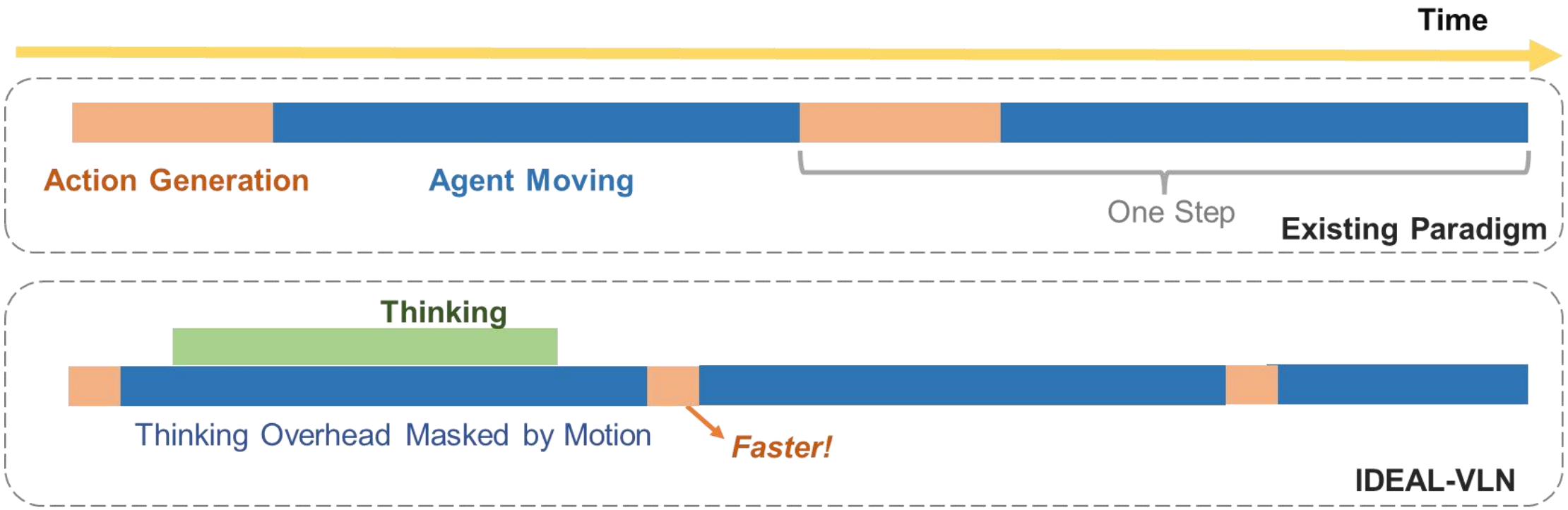
- ✗ Rule-Based Split (word count or period)
- ✗ Text Analysis and Split
- ✗ LLM-based Split

IDEAL-VLN（Single-System）



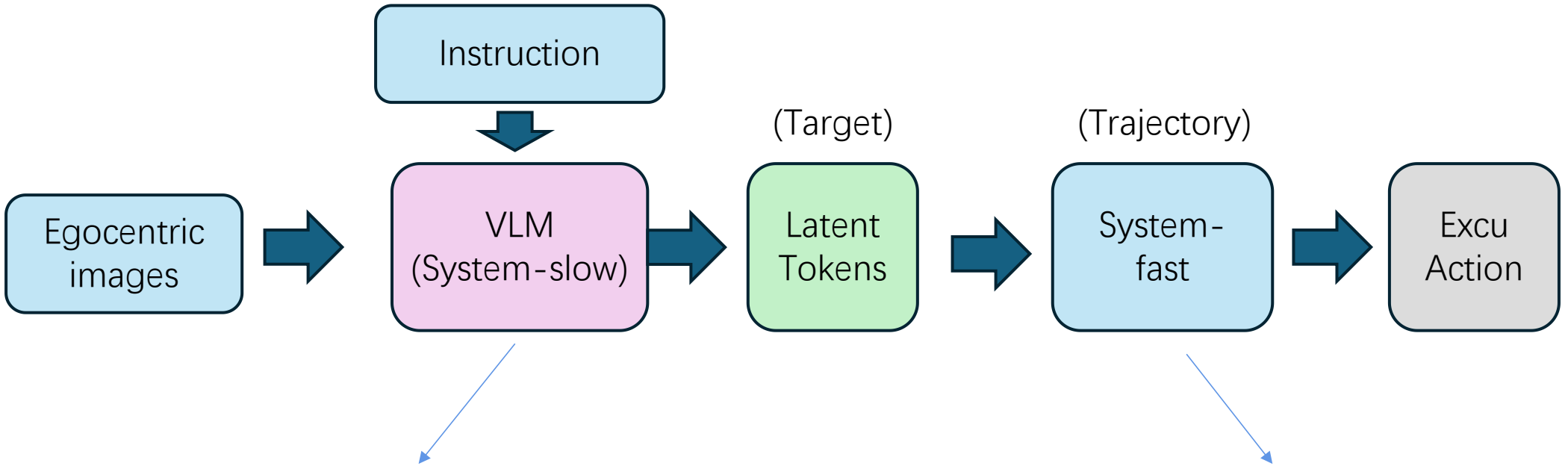
The existing models cannot comprehend the relationship between instructions and ego-centric images.

IDEAL-VLN（Single-System）



Method	Input tokens	Prefill Time(ms)	Steps/ Episode
Vanilla	7k	1625	21
Ours (Think)	7k	1832	5
Ours (Act)	2k	462	21

Dual-System VLN

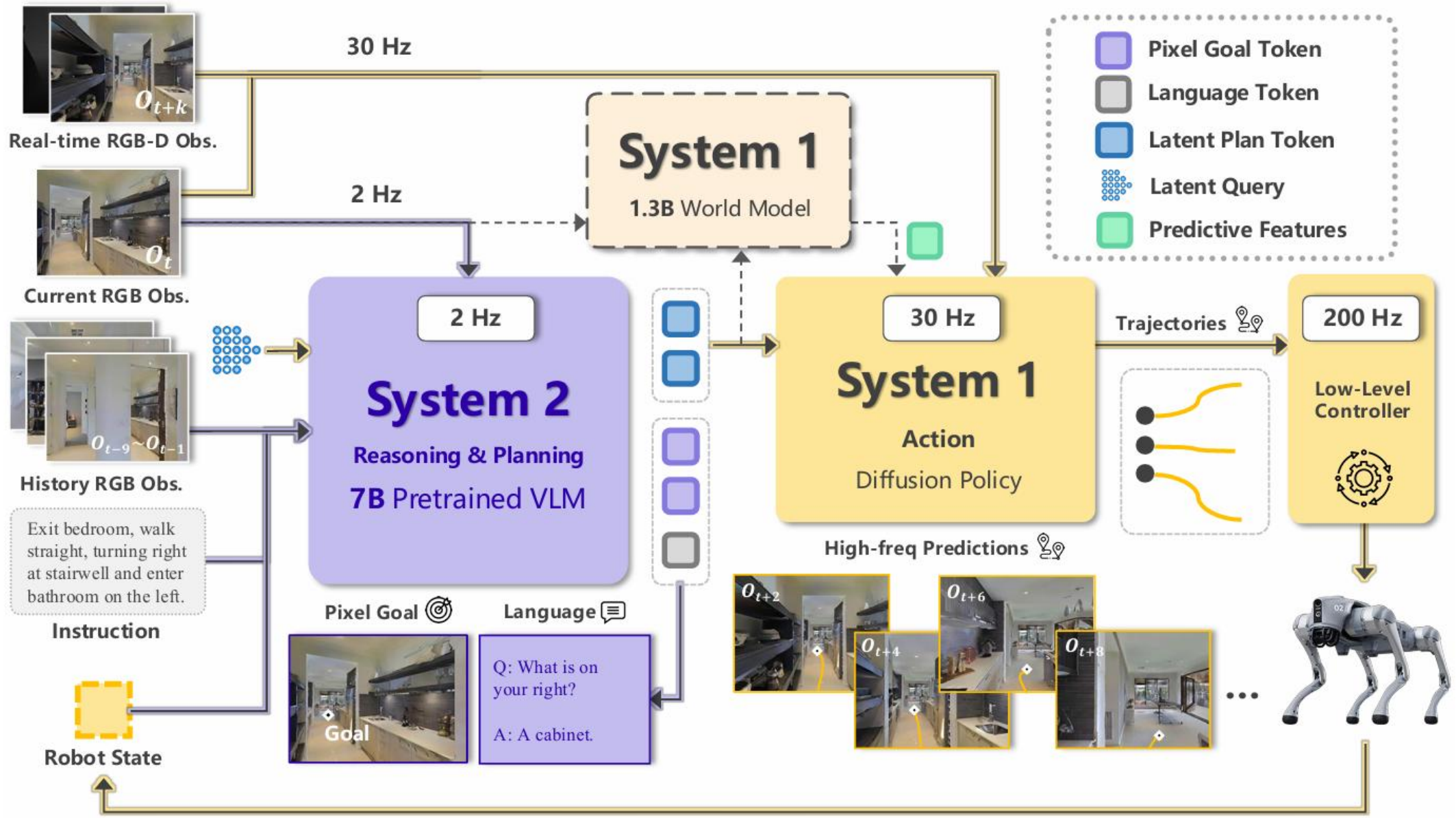


Only responsible for generating navigation points (image coordinate system)

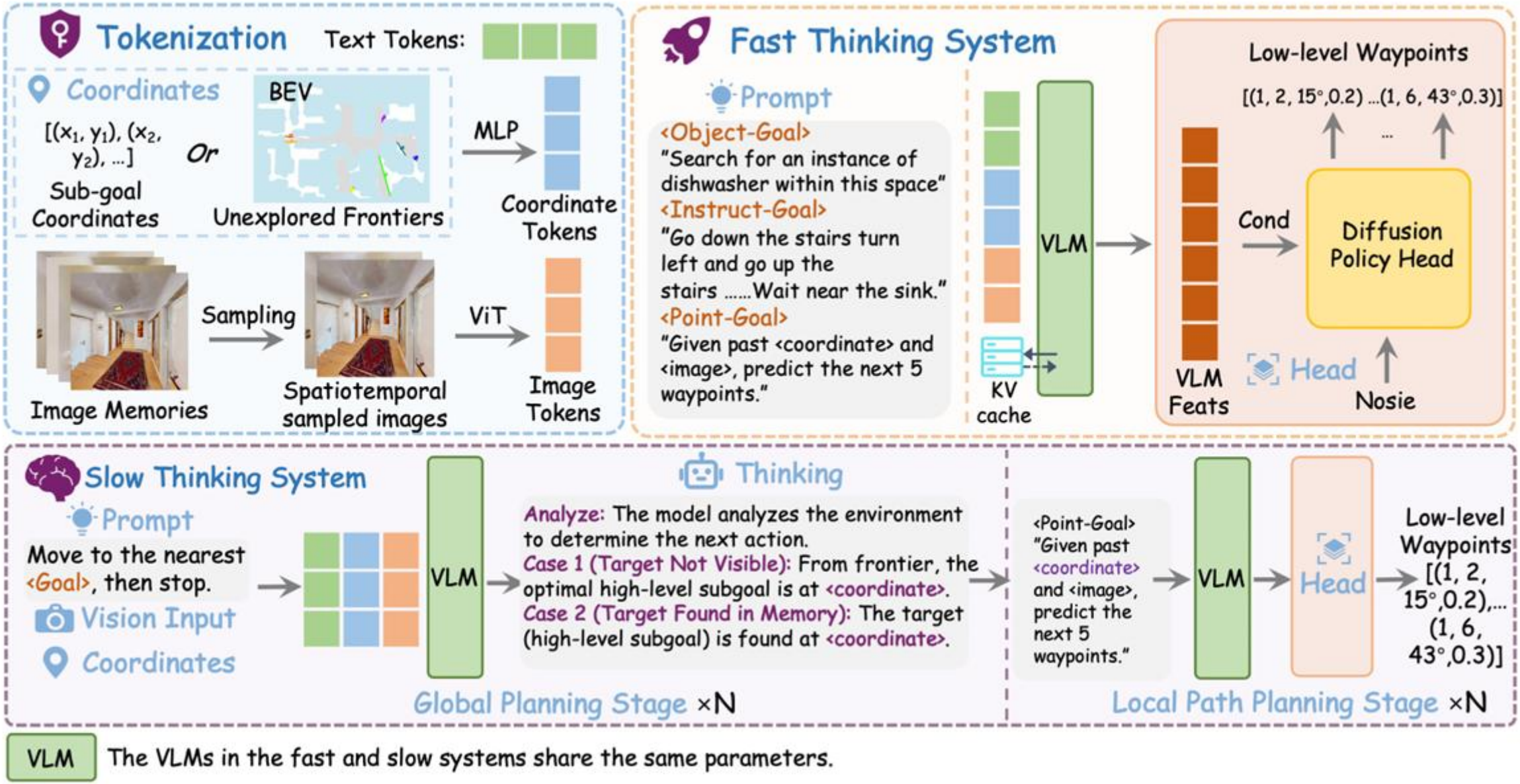
Convert the image coordinate into an executable trajectory

VLM is action-independent

DualVLN (ICLR 2026)



OmniNav (ICLR 2026)



Dual-System VLN

1. Joint training of the two systems is challenging
2. Incorrect navigation points may cause trajectory errors
3. Only available for specific robots
4. Very, very expensive

Dual-System VLN

A joint training scheme is particularly critical for Vision-and-Language Navigation (VLN), which requires strong general knowledge and semantic understanding, leading to substantial improvements in success rates in open environments. Stage 1 is trained with 96 NVIDIA H20 GPUs for 120 hours, and Stage 2 is trained with 64 NVIDIA H20 GPUs for 48 hours with lower learning rate.

-- OmniNav

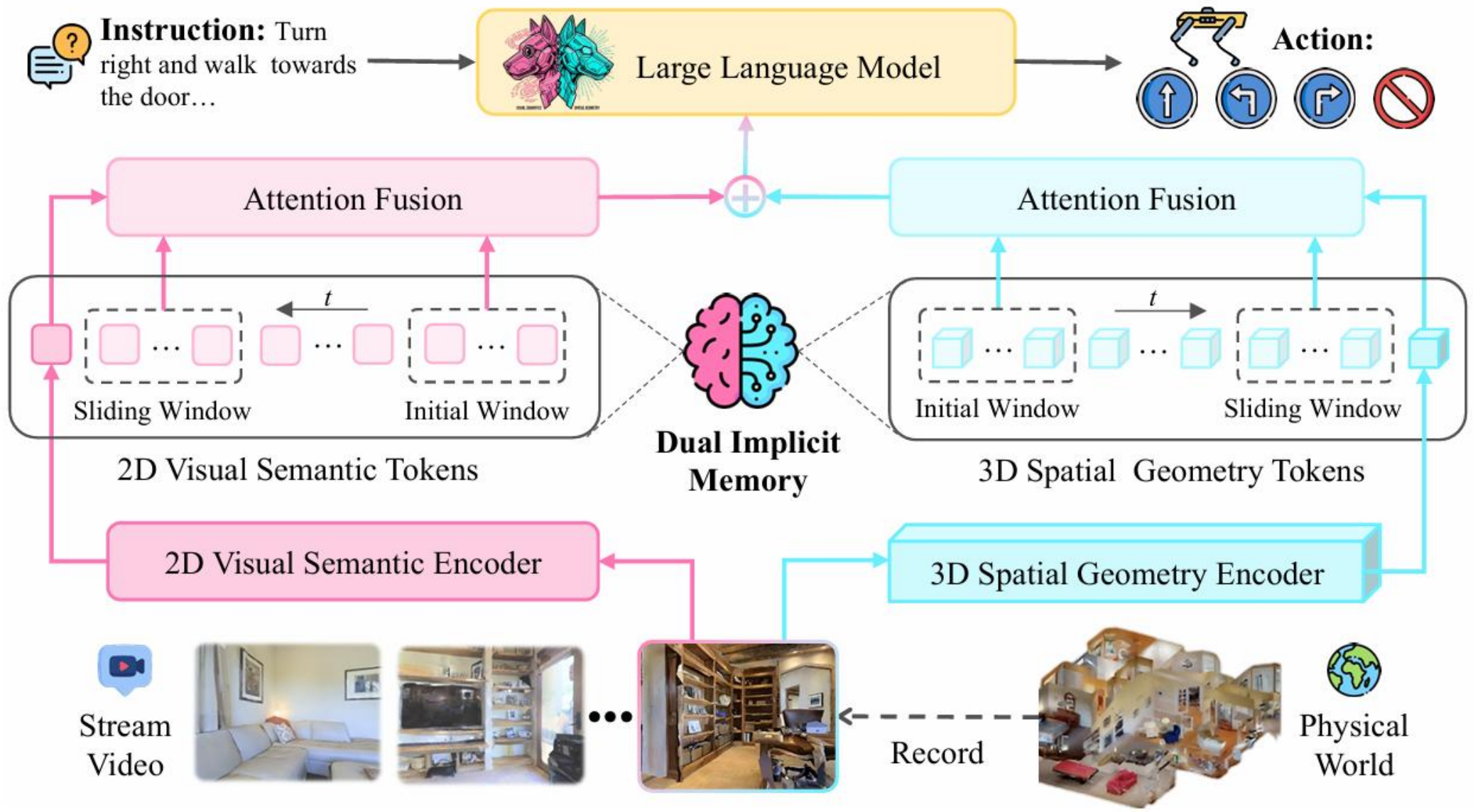
18000+ H20 hours
Dual-sys Training Cost: 15W+ RMB

Reproduction is extremely time-consuming

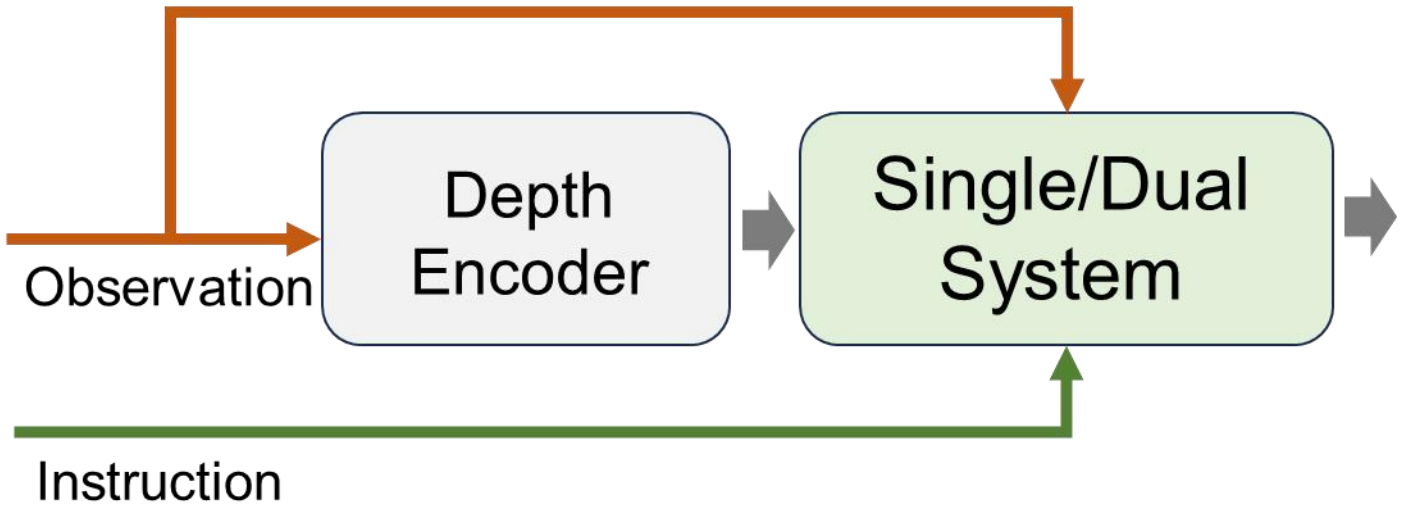
DecoVLN/ IDEAL-VLN: ~ 3k RMB

Depth Encoder-based Sys

JanusVLN (ICLR2026)



Depth Encoder-based Sys



Additional depth encoder required

Rely on depth encoder -> poor performance in real-world

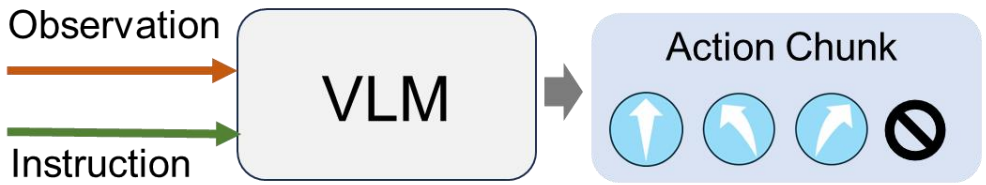
Significantly increases computational load and storage overhead

Depth Encoder-based Sys

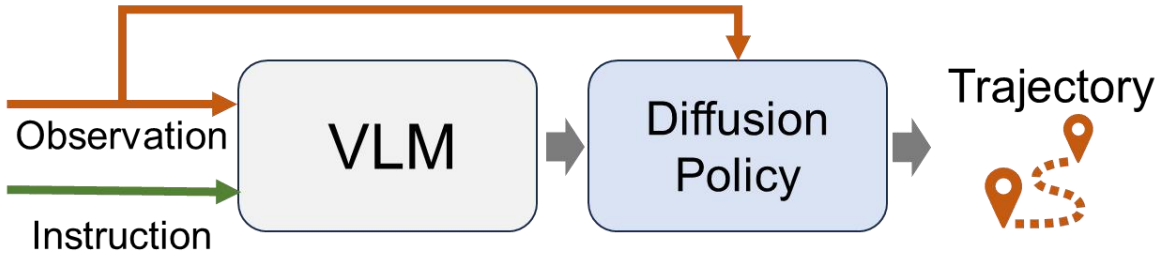
Why not use real depth map?

Depth map by realsense camera

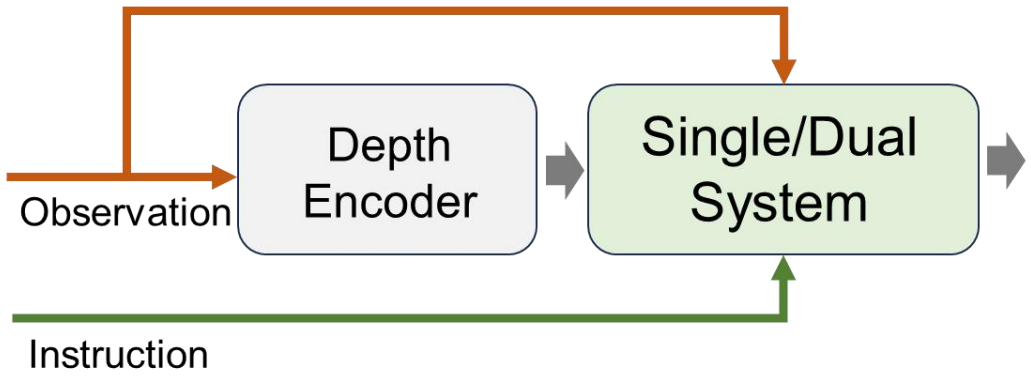
DepthAnything



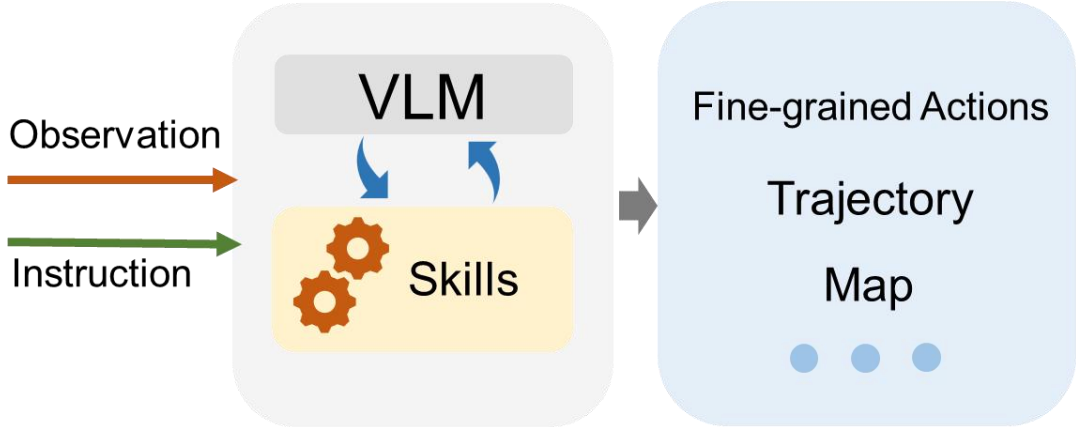
(a) Single-System



(b) Dual-System



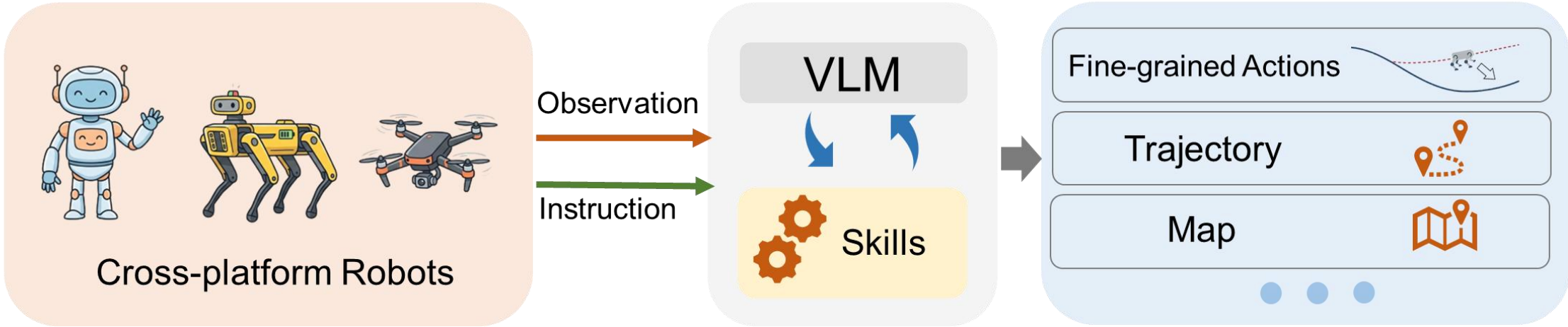
(c) Depth Encoder-based System



(d) AgentVLN

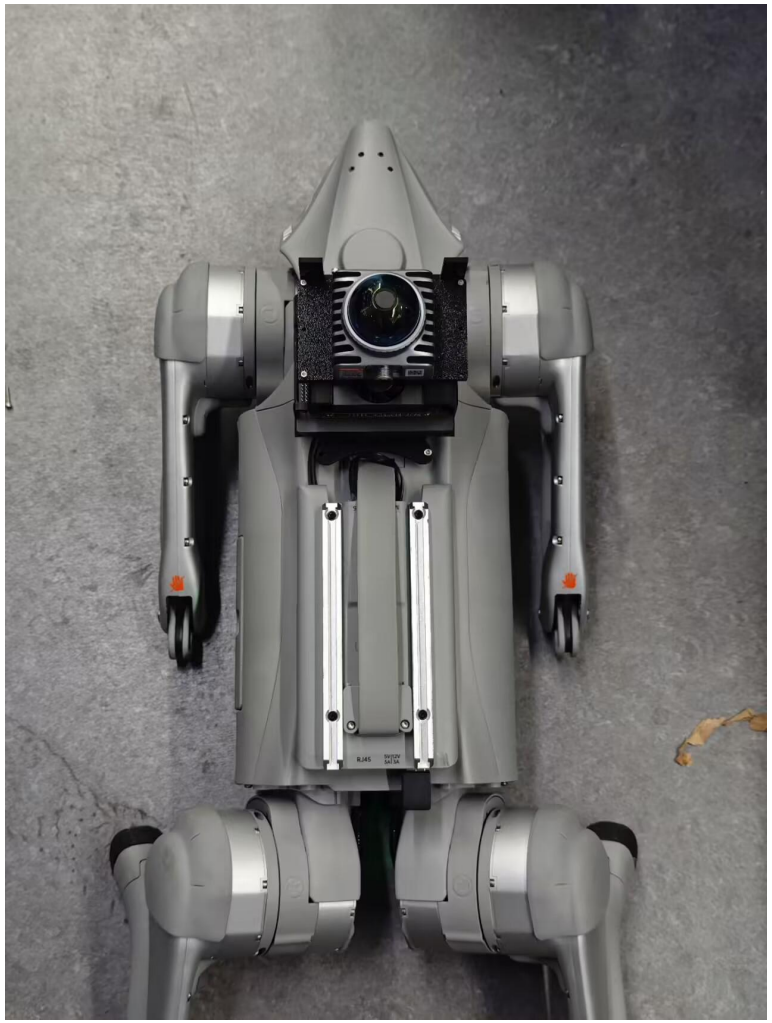
https://allenxinn.github.io/AgentVLN

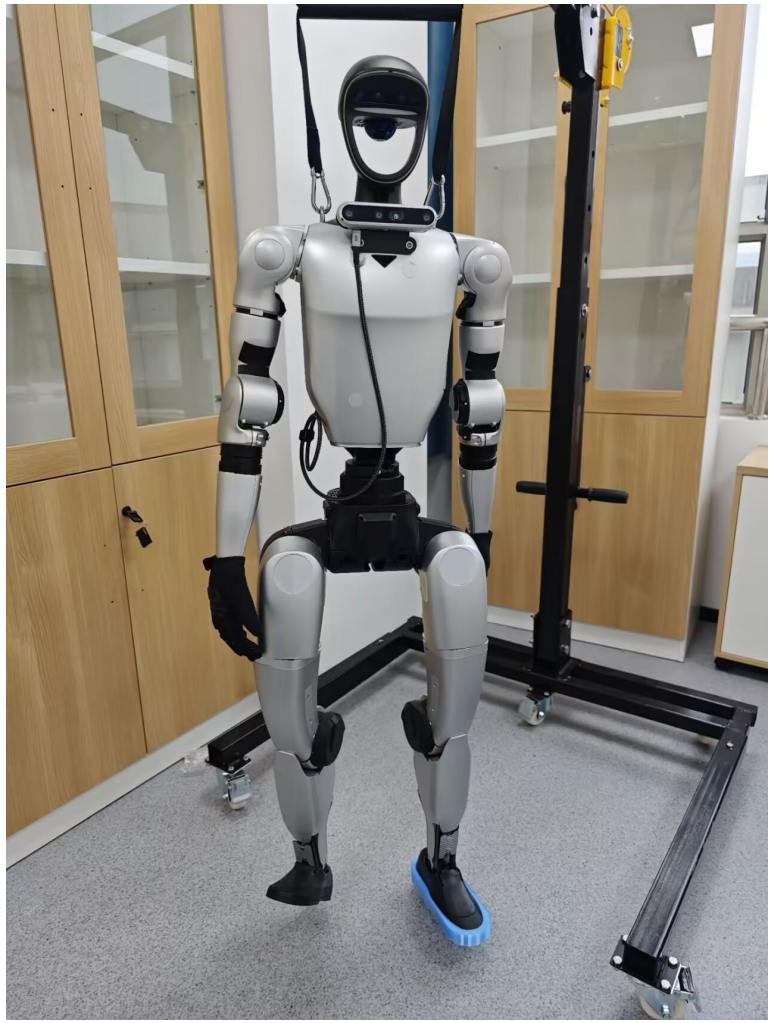
AgentVLN


Cross-platform Robots
Observation
Instruction
VLM
Skills
Fine-grained Actions
Trajectory
Map


633.0

AgentVLN



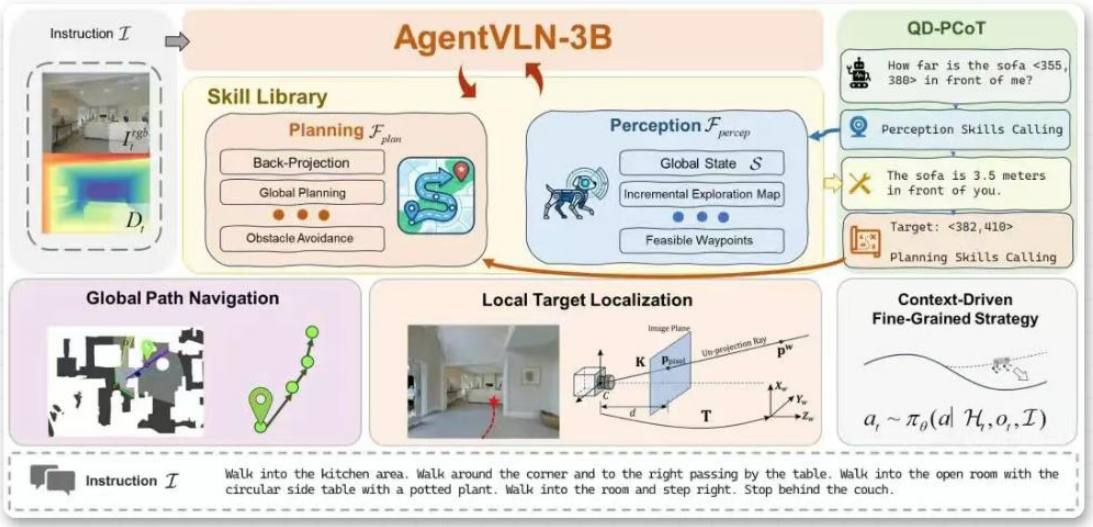


TODO

Quadrotor

小参数量，大能力！AgentVLN：轻量部署的下一代具身导航新范式

VLNer 视觉语言导航 2026年3月25日 21:50 湖南



- **作者**：Zihao Xin[1], Wentong Li[1*], Yixuan Jiang[1], Ziyuan Huang[1], Bin Wang[2], Piji Li[1], Jianke Zhu[3], Jie Qin[1], Shengjun Huang[1]
- **单位**：[1]南京航空航天大学，[2]山东大学，[3]浙江大学
- **论文标题**：AgentVLN: Towards Agentic Vision-and-Language Navigation
- **论文链接**：https://arxiv.org/abs/2603.17670v1
- **代码链接**：https://github.com/Allenxinn/AgentVLN

10k + views, 1k+ shares

AgentVLN：学会“自主导航”，轻量化具身视觉语言导航新范式！

原创 第一具身范式 具身智能之心TechDaily 2026年3月20日 17:32 上海

点击下方卡片，关注“第一具身范式”公众号

第一时间获取具身智能

具身智能之心TechDaily

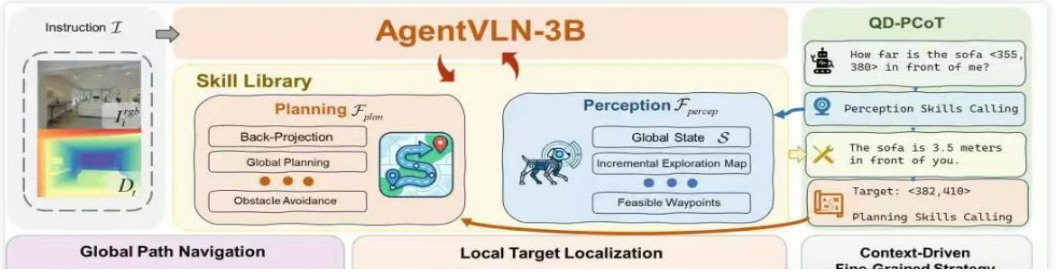
具身智能之心技术分享平台，最新前沿内容全收录～

427篇原创内容

公众号

> 行业探讨：自主语言导航会终结室内SLAM，还是补齐SLAM最后的短板？

在具身智能的研究版图中，视觉语言导航（VLN）是**极具挑战性**(ps：好发文章)的方向之一——它要求智能体依托第一视角的动态视觉观测，遵循自然语言指令，在未知的三维物理环境中完成长时程的自主导航。*视觉语言模型（VLM）虽在二维语义理解上展现出强大能力，却始终难以突破三维空间感知不足、2D-3D表征错位、单目尺度模糊的瓶颈，让VLN系统的实际落地困难重重。*

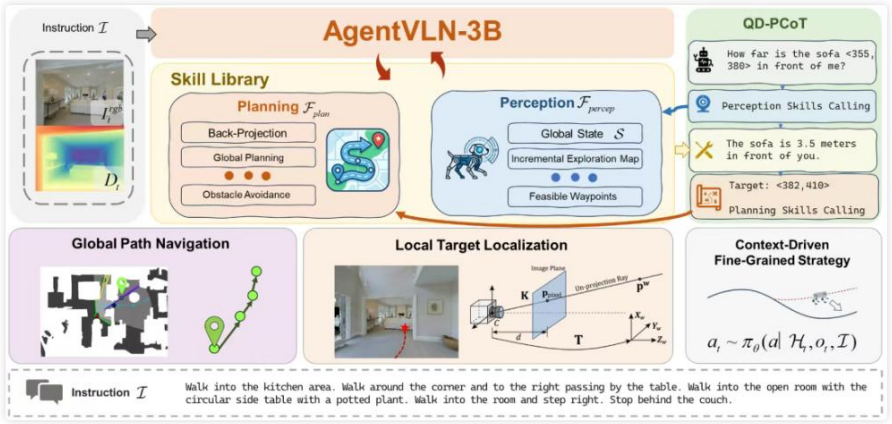


智能体也能“看图说话”自主导航？AgentVLN突破视觉语言导航新边界

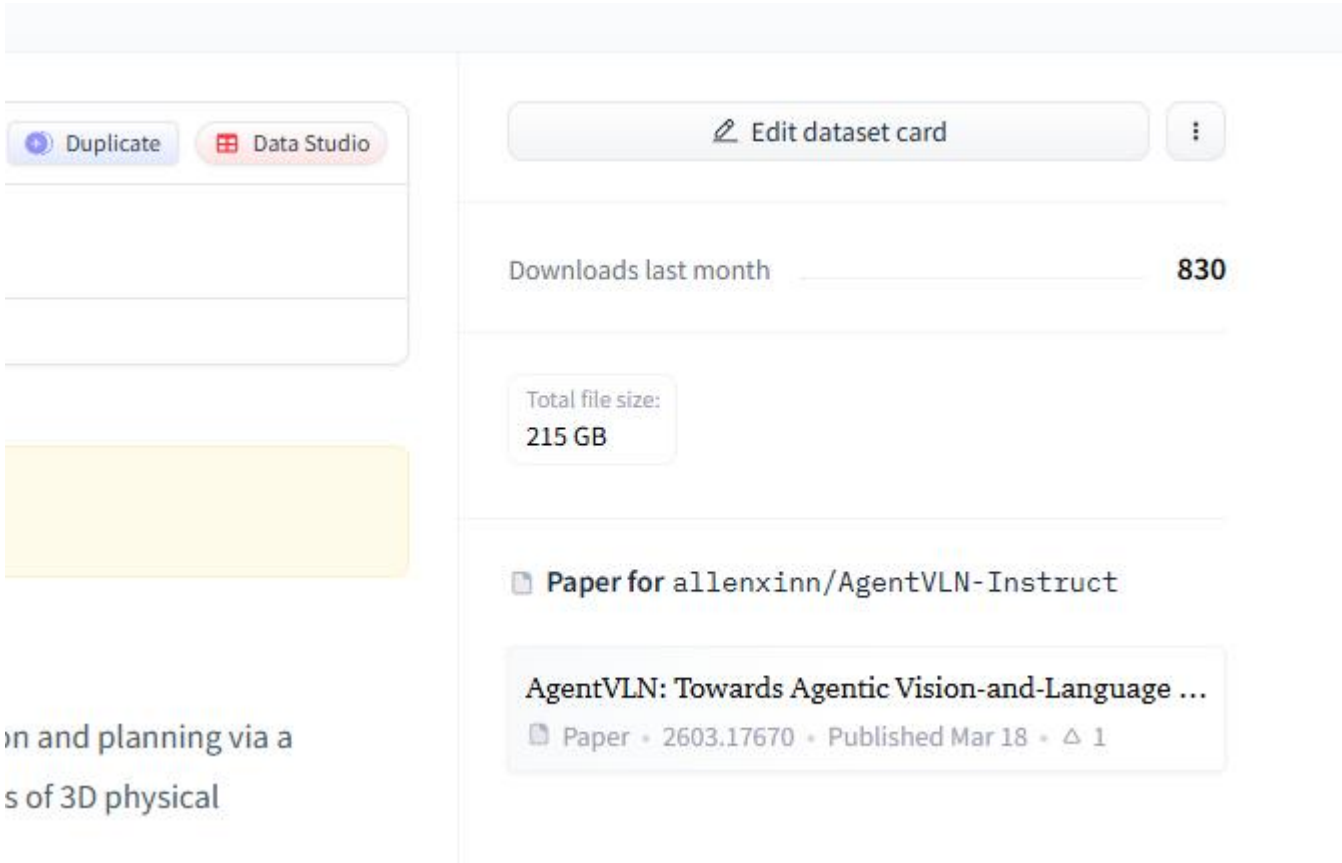
基本信息

- **作者**：Zihao Xin, Wentong Li, Yixuan Jiang, Ziyuan Huang, Bin Wang, Piji Li, Jianke Zhu, Jie Qin, Shengjun Huang
- **单位**：南京航空航天大学，山东大学，浙江大学
- **论文标题**：AgentVLN: Towards Agentic Vision-and-Language Navigation
- **论文链接**：https://arxiv.org/abs/2603.17670v1
- **代码链接**：https://github.com/Allenxinn/AgentVLN

简要介绍



视觉语言导航（VLN）要求智能体根据复杂语言指令在未知环境中进行长距离导航。然而，现有系统常受



AgentVLN

Instruction $\mathcal{I}$

AgentVLN-3B

Skill Library

Planning $\mathcal{F}_{plan}$

- Back-Projection
- Global Planning
- ● ● ●
- Obstacle Avoidance

Perception $\mathcal{F}_{percep}$

- Global State $\mathcal{S}$
- Incremental Exploration Map
- ● ● ●
- Feasible Waypoints

QD-PCoT

How far is the sofa <355, 380> in front of me?

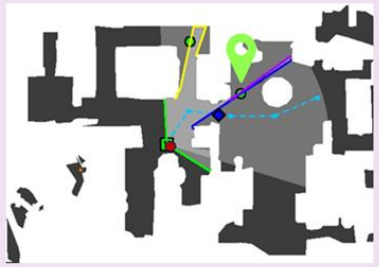
Perception Skills Calling

The sofa is 3.5 meters in front of you.

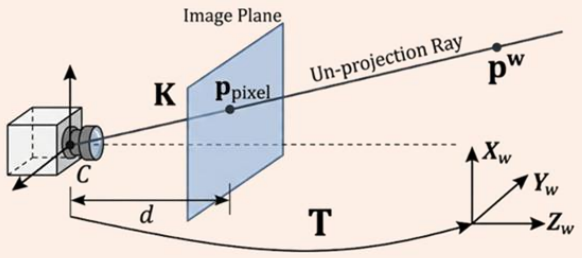
Target: <382,410>

Planning Skills Calling

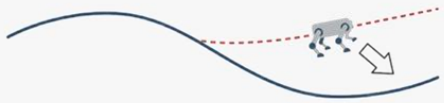
Global Path Navigation



Local Target Localization



Context-Driven Fine-Grained Strategy



$$a_t \sim \pi_\theta(a \mid \mathcal{H}_t, o_t, \mathcal{I})$$

Instruction $\mathcal{I}$

Walk into the kitchen area. Walk around the corner and to the right passing by the table. Walk into the open room with the circular side table with a potted plant. Walk into the room and step right. Stop behind the couch.

Duple Camera Sys

$$o_t = \{I_t^{rgb}, D_t\}$$

$$T_t = [R_t \mid \mathbf{t}_t] \in SE(3)$$

$$\mathbf{P}^w = R_t(d \cdot K^{-1}\mathbf{p}_{img}) + \mathbf{t}_t$$

$$s \cdot \mathbf{p}_{path}^{img} = KR_t^{-1}(\mathbf{P}_{path}^w - \mathbf{t}_t)$$

$$\mathbf{P}_{target}^c = d_{target} \cdot K^{-1}\mathbf{p}_{target}^{img} = \begin{bmatrix} \frac{(u_{target}-c_x)\cdot d_{target}}{f_x} \\ \frac{(v_{target}-c_y)\cdot d_{target}}{f_y} \\ d_{target} \end{bmatrix}$$

$$\mathbf{P}_{target}^w = R_t\mathbf{P}_{target}^c + \mathbf{t}_t$$

Lidar + Monocular Camera Sys

$$\mathbf{P}_{L,i} = R_W^L\mathbf{P}_{w,i} + \mathbf{t}_W^L$$

$$\mathbf{P}_{L,i} = R_W^L\mathbf{P}_{w,i} + \mathbf{t}_W^L$$

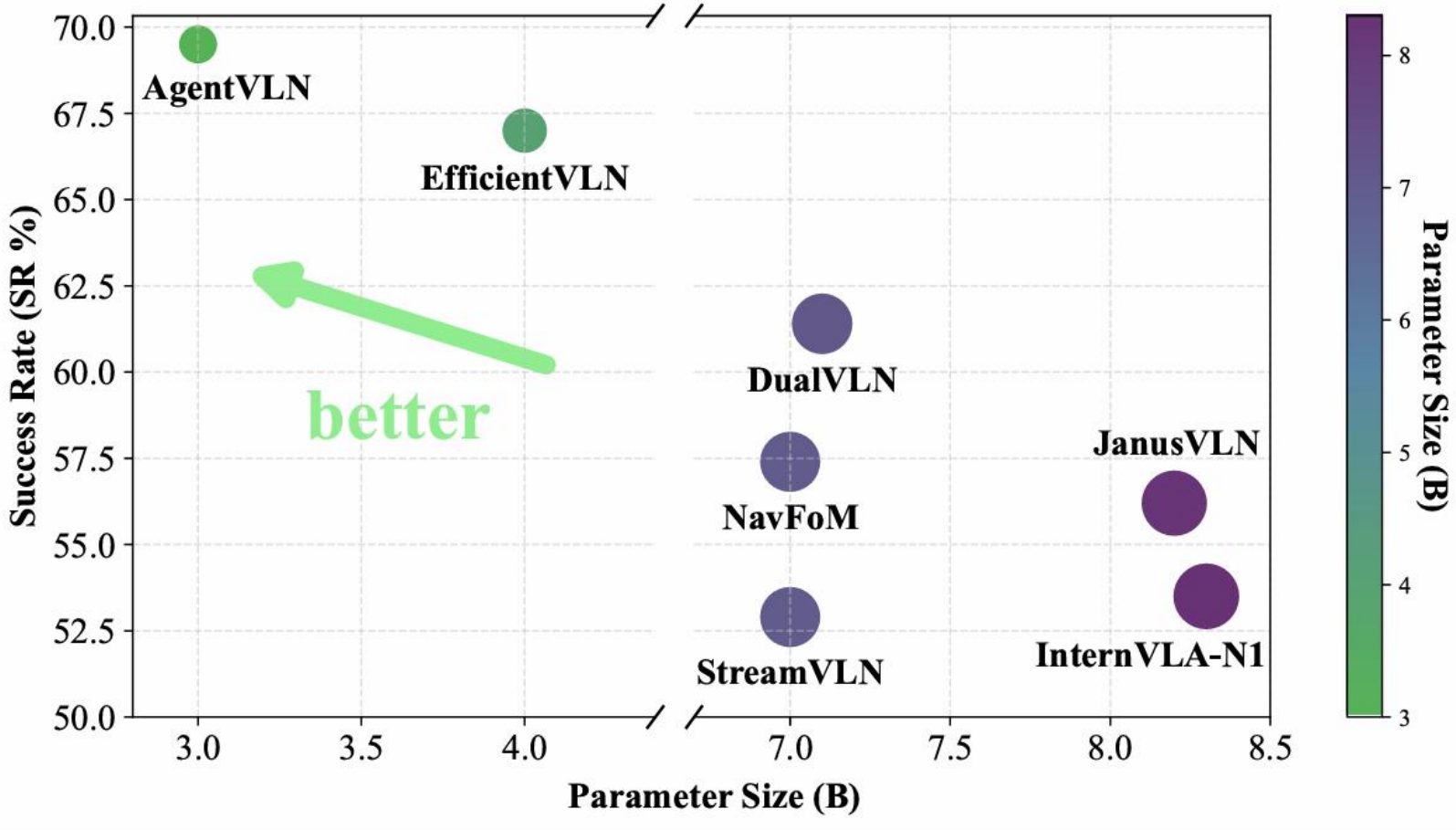
$$\mathbf{P}_{C,i} = R_L^C\mathbf{P}_{L,i} + \mathbf{t}_L^C$$

$$\hat{d} = \frac{\sum_{m=1}^{k} \omega_m \cdot d_m}{\sum_{m=1}^{k} \omega_m}$$

$$\omega_m = \frac{1}{\|\mathbf{p}_{target}^{img} - \mathbf{p}_m^{img}\|_2}$$

$$\mathbf{P}_{target}^C = \hat{d} \cdot K^{-1} \begin{bmatrix} u^* \\ v^* \\ 1 \end{bmatrix}$$

AgentVLN



Inference peak memory:

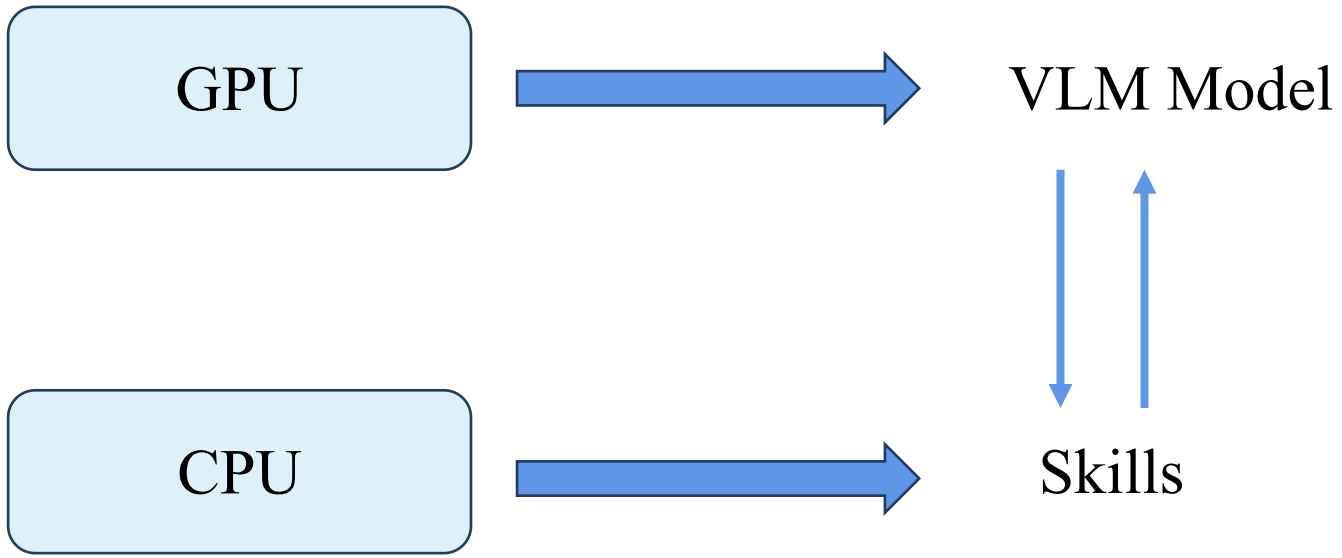
AgentVLN: ~9G

DualVLN: ~20G

StreamVLN: ~18G

The first VLN model deployable on embedded devices.

Outstanding load balancing



AgentVLN


Step: 0 [Plan]
Type: action
Dist: 13.7m
Pred: <action> TURN_RIGHT TURN_RIGHT TURN_RIGHT TURN_RIG


Start
End
Expert Path
Traveled
Targets
Nav Target

AgentVLN

Table 1: Comparison results with SOTA methods on the Val-Unseen dataset for R2R-CE. Our method outperforms other approaches on the same benchmarks, even having fewer parameters.

Method	Observation				R2R-CE			
	S.RGB	Pano.	Depth	Odo.	NE ↓	OS ↑	SR ↑	SPL ↑
HPN+DN [20]		✓	✓	✓	6.31	40.0	36.0	34.0
VLN BERT [17]		✓	✓	✓	5.74	53.0	44.0	39.0
CMA [17]		✓	✓	✓	6.20	52.0	41.0	36.0
Reborn [2]		✓	✓	✓	5.40	57.0	50.0	46.0
Ego^2-Map [18]		✓	✓	✓	5.54	56.0	47.0	41.0
DreamWalker [39]		✓	✓	✓	5.53	59.0	49.0	44.0
ETPNav [1]		✓	✓	✓	4.71	65.0	57.0	49.0
Seq2Seq [21]	✓		✓		7.77	37.0	25.0	22.0
RGB-CMA [21]	✓				9.55	10.0	5.0	4.0
AG-CMTP [8]		✓	✓		7.90	39.0	23.0	19.0
R2R-CMTP [8]		✓	✓		7.90	38.0	26.0	22.0
LAW [35]	✓		✓	✓	6.83	44.0	35.0	31.0
CM2 [14]	✓		✓		7.02	41.0	34.0	27.0
WS-MGMap [9]	✓		✓		6.28	47.0	38.0	34.0
ETPNav+FF [42]	✓		✓		5.95	55.8	44.9	30.4
AO-Planner [7]		✓	✓		5.55	59.0	47.0	33.0
NaVid-7B [51]	✓				5.47	49.0	37.0	35.0
Uni-NaVid-7B [50]	✓				5.58	53.3	47.0	42.7
NaVILA-7B [12]	✓				5.22	62.5	54.0	49.0
StreamVLN-7B [44]	✓				5.10	64.0	55.7	50.9
NavFoM-7B [49]	✓				5.01	64.9	56.2	51.2
JanusVLN-8.2B [48]	✓				4.78	65.2	60.5	56.8
EfficientVLN-4B [53]	✓				4.18	**73.7**	64.2	55.9
DualVLN-7.1B [43]	✓				4.05	70.7	64.3	58.5
StreamVLN-7B [44]	✓		✓		4.98	64.2	56.9	51.9
InternVLA-N1-8.3B [38]	✓		✓		4.83	63.3	58.2	54.0
AgentVLN-3B	✓		✓		**3.88**	73.5	**67.2**	**64.7**

Table 2: Comparison with SOTA methods on RxR-CE Val-Unseen split.

Method	S.RGB	Pano.	Depth	Odo.	NE ↓	SR ↑	SPL ↑	nDTW ↑
VLN BERT [17]		✓	✓	✓	8.98	27.0	22.6	46.7
CMA [17]		✓	✓	✓	8.76	26.5	22.1	47.0
Reborn [2]		✓	✓	✓	5.98	48.6	42.0	63.3
ETPNav [1]		✓	✓	✓	5.64	54.7	44.8	61.9
Seq2Seq [21]	✓		✓		12.10	13.9	11.9	30.8
LAW [35]	✓		✓	✓	10.90	8.0	8.0	38.0
ETPNav+FF [42]	✓		✓	✓	8.79	25.5	18.1	-
AO-Planner [7]		✓	✓		7.06	43.3	30.5	50.1
Uni-NaVid-7B [50]	✓				6.24	48.7	40.9	-
NaVILA-7B [12]	✓				6.77	49.3	44.0	58.8
NavFoM-7B [49]	✓				5.51	57.4	49.4	60.2
JanusVLN-8.2B [47]	✓				6.06	56.2	47.5	62.1
StreamVLN-7B [44]	✓				6.16	51.8	45.0	61.9
InternVLA-N1-7B [13]	✓				6.41	49.5	41.8	62.6
EfficientVLN-4B [53]	✓				**3.88**	67.0	54.3	68.4
DualVLN-7.1B [43]	✓				4.58	61.4	51.8	70.0
StreamVLN-7B [44]	✓		✓		6.22	52.9	46.0	61.9
InternVLA-N1-8.3B [38]	✓		✓		5.91	53.5	46.1	65.3
AgentVLN-3B	✓		✓		3.92	**69.5**	**61.3**	**74.6**

Future Work: Infinite Memory

Model architecture specifically designed for VLN tasks.

Static Memory: $H^I = \text{EncodeOnce}(I), \qquad M_I = K_I^{(l)}, V_I^{(l)}$

Dynamic Memory: $h_t^{loc} = \text{SWA}(o_{t-K:t}, a_{t-K:t-1}, p_{t-K:t})$

Memory Update:

$$e_t = \text{WriteMLP}(h_t^{loc}, a_{t-1}, p_t)$$

$$S_t = \text{GatedDeltaNetWithDecay}(S_{t-1}, e_t)$$

$$h_t^{hist} = \text{Read}(S_t, q_t^{hist})$$

$$S_t = \rho_t \odot S_{t-1}\left(I - \beta_t k_t k_t^\top\right) + \beta_t v_t k_t^\top$$

Future Work: Infinite Memory

Model architecture specifically designed for VLN tasks.

Sliding Attention + GDN Attention

Peak memory usage: 5.5G
Achieve 10 FPS
Inference Delay: 0.45s
No inference optimization was performed!