Egocentric RGB-D Perception for High-Level Locomotion Planning in Humanoid Robots

Tae Yang
Stanford University
taeyang@stanford.edu

Abstract

Humanoid robots navigating complex environments must select appropriate locomotion modes such as walking, crawling, or climbing. We propose a vision-based high-level planner that classifies locomotion mode from egocentric RGB-D input using a lightweight CNN-LSTM architecture. The model encodes spatial features via a CNN and captures temporal context through an LSTM. Training data is collected in a custom MuJoCo simulation with a humanoid robot equipped with stereo cameras. RGB and depth images are rendered using real-world camera intrinsics. Augmentations such as color jittering and horizontal flipping are applied to improve generalization. We conduct an ablation study across seven model variants, comparing RGB-only, depth-only, and RGB-D input, with or without LSTM and pretrained encoders. The best-performing model—RGB-D input, pretrained CNN, and LSTM—achieves the highest classification accuracy and runs over 500× faster than vision-language model baselines. Temporal context from the LSTM improves planning by leveraging past visual input and prior mode transitions. Our results demonstrate that a compact RGB-D classifier can serve as an efficient and accurate high-level planner for humanoid robots, with strong potential for real-time deployment in visually complex environments.

1. Introduction

Humanoid robots operating in complex real-world environments must dynamically switch between different locomotion modes, such as walking, crawling, or climbing, depending on terrain geometry and obstacles. This multimodal capability allows humanoids to traverse diverse terrain, including flat floors, confined underpasses, and elevated bins. However, selecting the correct mode of movement in real time—also known as high-level locomotion planning—remains a significant challenge. Unlike low-level motor control, this problem requires semantic understanding, spatial reasoning, and temporal context.

Recent research has leveraged vision-language models (VLMs) to address high-level robot decision-making through text-conditioned reasoning over egocentric views [4, 3]. While VLMs show impressive zero-shot generalization, their large computational footprint, slow inference time, and reliance on language prompts hinder real-time onboard deployment, especially for embedded systems.

To address this, we present a lightweight visual classifier for locomotion planning that operates solely on egocentric RGB-D input, removing the need for language and enabling efficient inference. The input to our algorithm is a time-series of synchronized RGB and depth frames captured from a head-mounted stereo camera. The model uses convolutional neural networks (CNNs) to extract modality-specific spatial features and a recurrent long short-term memory (LSTM) module to capture temporal dynamics. The output is a discrete locomotion mode: `walk`, `crawl`, or `climb`.

Depth perception provides critical geometric cues for assessing traversability and obstacle height, while RGB images capture semantic information, such as the presence of gaps or surfaces. Temporal information is essential for smooth transitions and disambiguating momentary postures—for instance, distinguishing a low crouch from a crawl initiation. Prior work in robot locomotion has explored multi-modal inputs [10, 5], hierarchical control [9], and egocentric visual reasoning [6] for terrain adaptation.

We build our approach on the ToddlerBot platform [12], a compact, open-source humanoid robot designed for simulation and learning. Our data is generated in a custom MuJoCo environment using robosuite [15], and carefully calibrated to match the real robot's fisheye camera intrinsics and stereo depth pipeline. This enables highly realistic synthetic RGB-D sequences for walking, crawling, and climbing behaviors, collected across diverse procedurally generated terrain layouts.

To validate our design, we conduct a comprehensive ablation study comparing models with and without depth,

temporal encoding, and pretrained backbones. Our best model—combining RGB-D input, pretrained CNNs, and an LSTM—achieves strong classification accuracy and runs over 500× faster than VLM-based approaches. These results highlight the effectiveness of compact visual classifiers for real-time high-level planning in legged humanoid systems.

2. Related Work

2.1. Visual Locomotion Planning in Robotics

High-level locomotion planning in robotics has traditionally relied on geometry-based path planning with terrain mapping, assuming global maps or structured environments [7]. However, recent advances use visual inputs to adapt locomotion strategies in an end-to-end or hybrid manner.

A major milestone is the development of modular locomotion pipelines that separate terrain perception from locomotion control. For example, the "Parkour in the Wild" framework [10] learns expert locomotion policies for walking, crawling, and climbing, and fuses them via a mode-switching policy informed by terrain embeddings. Inspired by this, our method simplifies the decision-making pipeline by replacing learned high-level planners with a vision-based classifier operating on egocentric RGB-D sequences.

Prior approaches to visual locomotion use either third-person terrain maps [2, 1] or exteroceptive sensors such as LiDAR or stereo [13]. In contrast, we rely entirely on first-person RGB-D input, which more realistically matches onboard sensing in embedded humanoids like ToddlerBot [12].

2.2. Vision-Language Models for Robot Planning

Recent work has explored vision-language models (VLMs) for robot planning and adaptation. These models, such as CLIP and its derivatives, are used to reason over egocentric visual observations using text-conditioned prompts. Commonsense Reasoning for Legged Robots [4] proposes using prompts like "Can the robot walk here?" and infers locomotion decisions by comparing image-text similarity. SpatialBot [3] improves spatial reasoning by enhancing CLIP with synthetic data and spatial grounding.

Although VLMs enable zero-shot generalization and interpretable decision making, their inference latency and computational footprint make them impractical for fast control loops. Additionally, they require prompt design and may not generalize to environments unseen during pretraining [8]. In contrast, we design a compact RGB-D model that is trained end-to-end for locomotion classification without prompt engineering.

2.3. Egocentric Visual Understanding

Egocentric vision has become a key modality for embodied AI. Jain et al. [6] use a transformer-based visual foresight model for terrain prediction and footstep planning. Prior work has also incorporated depth sensing and temporal aggregation to improve affordance recognition [14]. Our work is aligned with this direction, using temporal RGB-D inputs and LSTM-based aggregation to disambiguate short-horizon visual ambiguity and recognize behavioral intent.

2.4. Simulation Platforms and Synthetic Datasets

High-fidelity simulation plays a crucial role in training perception-based controllers. robosuite [15] provides a modular interface for robotic manipulation and locomotion in MuJoCo. ToddlerBot [12] offers an ML-compatible humanoid platform for simulation and real-world transfer. We extend this ecosystem by rendering egocentric RGB-D video and constructing a labeled dataset across locomotion behaviors. This enables large-scale training without manual annotation, similar to synthetic pipelines in visual navigation [11].

3. Methods

Our goal is to train a high-level planner that classifies locomotion mode—`walk`, `crawl`, or `climb`—from egocentric RGB-D visual input. To this end, we build a simulated data collection pipeline, design a lightweight vision-based classification architecture, and conduct an extensive ablation study to evaluate the impact of modality, pretraining, and temporal encoding. This section describes our dataset generation process, input encoding strategies, model architecture, and training procedures in detail.

3.1. Camera Modeling and Simulation Setup

Our approach begins with accurately replicating the vision system of the real humanoid robot in simulation. The physical platform uses two Arducam B0202 fisheye cameras with a diagonal field of view (FOV) of 160°, a resolution of 640×480, and a fixed-focus lens. These cameras are mounted on the robot's head to provide an egocentric stereo perspective, critical for visual terrain understanding.

To ensure the simulated images match the robot's perception pipeline, we first calibrate the real cameras using a standard checkerboard calibration pipeline. This produces undistorted and rectified image pairs suitable for stereo processing. Figure 1 shows an example of fisheye distortion before and after rectification.

We extract the camera intrinsics and extrinsics from the calibration process, including focal lengths, principal points, and baseline. These parameters are then used to define identical pinhole cameras within the MuJoCo simulator. Each simulated camera is attached to the head of the

Figure 1: (Top) Raw fisheye camera input with barrel distortion. (Bottom) Undistorted and rectified image used for stereo depth estimation.

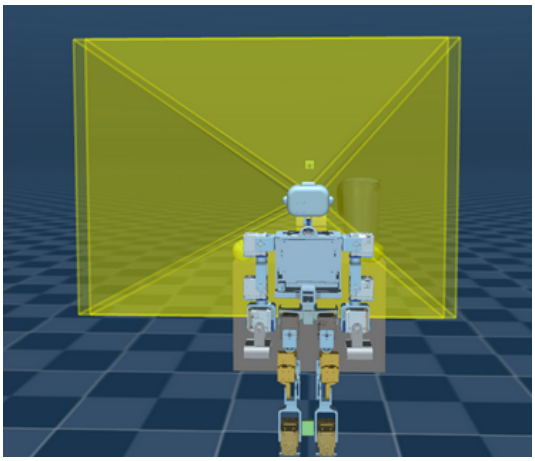


Figure 2: Simulated stereo camera setup in MuJoCo. Two cameras are attached to the robot's head with the same intrinsics and extrinsics as the real stereo rig. Frustums are visualized in yellow.

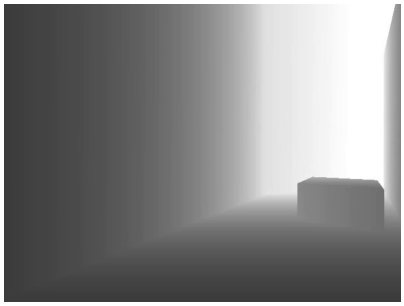
humanoid robot model (ToddlerBot [12]) at the same relative transform as the real-world stereo setup. The cameras are configured to output RGB images and render accurate depth maps in the left camera's frame.

This physically grounded camera simulation ensures that both RGB and depth inputs are consistent across real and simulated domains, enabling better generalization and simplifying future transfer to the real robot. Figure 2 illustrates the simulated stereo setup with visible camera frustums.

3.2. Simulation Environment and Visual Data Collection

To support diverse locomotion scenarios, we construct a custom simulation environment using `robosuite` [15], into which we load the ToddlerBot humanoid model [12].

(a) RGB image



(b) Depth image (normalized)

Figure 3: Simulated egocentric view from the left camera. RGB and depth frame pair for a walk sequence.

The scene is populated with semantically meaningful and physically interactive obstacles. These elements are designed to elicit distinct locomotion modes such as walking on flat terrain, crawling under low gaps, and climbing over obstacles.

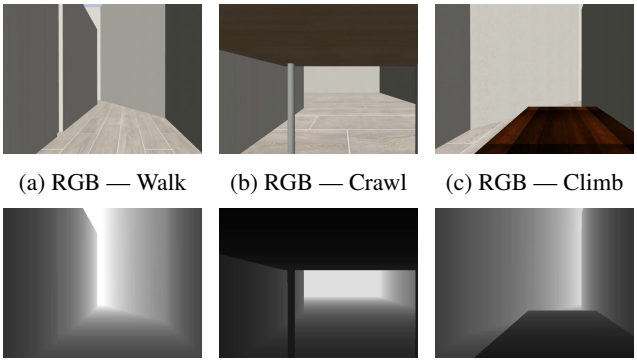
During simulation, we render both RGB and depth images at a resolution of 640×480. Depth maps are captured directly using MuJoCo's GPU-accelerated offscreen renderer from the left camera, consistent with the real robot's stereo-based depth estimation pipeline, which generates disparity in the left frame. Figure 3 shows sample outputs from the left camera in simulation.

To collect data across various environments and locomotion behaviors, we enable manual teleoperation of the robot using a keyboard-based control interface. The robot supports both translational and rotational base movements, allowing it to traverse narrow passages, crawl under bars, and approach climbable obstacles with realistic egocentric motion.

During data collection, RGB and depth video streams are captured simultaneously from the left head-mounted camera. Rendering is performed offscreen to ensure consistency and reproducibility. Each RGB-D video pair is synchronized and stored as a continuous recording of the robot's traversal through the scene. Figure 4 shows a snapshot of the robot actively traversing the simulated environment under human teleoperation.

(a) `move_1` (b) `move_2` (c) `move_3`

Figure 4: Sequence of robot movement frames during teleoperated navigation. The robot progresses through the environment over time.



(a) RGB — Walk (b) RGB — Crawl (c) RGB — Climb

(d) Depth — Walk (e) Depth — Crawl (f) Depth — Climb

Figure 5: Labeled RGB and depth frames sampled from each locomotion class. Depth maps are clipped to 3 meters and normalized for visualization.

3.3. Dataset and Feature Extraction

After collecting synchronized RGB and depth videos across different locomotion modes, we construct the training dataset by temporally segmenting the sequences and extracting labeled image frames.

Each recording corresponds to a distinct locomotion behavior: walking on flat terrain, crawling under bars, or climbing over bins. Since the videos are captured independently for each behavior, labeling becomes straightforward. We uniformly sample frames every 5 time steps from both RGB and depth videos, and save them as image pairs with consistent filenames encoding the label. This automated pipeline enables efficient dataset construction without requiring per-frame manual annotation.

The full dataset contains 1,056 labeled RGB-depth image pairs in total, spanning three classes: `walk`, `crawl`, and `climb`. We split the data randomly into 80 percent training and 20 percent validation sets. Each RGB and depth image is center-cropped and resized to a resolution of 128×128 pixels. RGB images are augmented using random horizontal flipping, color jitter, and Gaussian blur to increase robustness to lighting variation and camera perspective. Depth images are converted from grayscale, clipped to a maximum distance of 3 meters, and normalized to the $[0, 1]$ range.

If the model uses a temporal encoder (e.g., an LSTM), the dataset loader returns sequences of three consecutive frames. The label corresponds to the final frame in the sequence, allowing the model to incorporate short-horizon temporal context. During training, sequences are dynamically constructed from contiguous frames of the same locomotion class to ensure label consistency.

All visual features are learned directly from raw image tensors using CNN backbones. No manual feature extraction is applied. Figure 5 shows representative RGB and depth frames from each of the three locomotion classes. Each modality contributes complementary information: RGB captures semantic context, while depth provides geometric cues critical for terrain understanding.

3.4. Model Architecture

We formulate high-level locomotion planning as a supervised classification problem over three classes: walking, crawling, and climbing. The input to our model is a sequence of $T = 3$ egocentric RGB and depth image pairs from the robot's left camera. Each RGB image is denoted $\mathbf{x}_t^{\text{rgb}} \in \mathbb{R}^{3\times H\times W}$, and each depth image $\mathbf{x}_t^{\text{depth}} \in \mathbb{R}^{1\times H\times W}$, where $t = 1, \ldots, T$ and $H = W = 128$. The goal is to predict a locomotion label $y \in \{0, 1, 2\}$ corresponding to the locomotion mode in the final frame $t = T$.

Visual encoders. RGB and depth images are processed by two independent convolutional encoders: - The RGB encoder is based on ResNet18 pretrained on ImageNet. Let $f_{\text{rgb}}(\cdot)$ denote this feature extractor, which maps $\mathbf{x}_t^{\text{rgb}} \rightarrow \mathbf{h}_t^{\text{rgb}} \in \mathbb{R}^d$. - The depth encoder, $f_{\text{depth}}(\cdot)$, is a lightweight CNN trained from scratch to accommodate the different distribution of simulated depth maps.

We concatenate both embeddings at each time step to produce $\mathbf{z}_t = [\mathbf{h}_t^{\text{rgb}}; \mathbf{h}_t^{\text{depth}}] \in \mathbb{R}^{2d}$, yielding a temporal sequence $\{\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3\}$.

Temporal module. To incorporate short-horizon temporal context, the concatenated feature sequence is passed through a single-layer LSTM:

$$\mathbf{h}_T^{\text{lstm}} = \text{LSTM}(\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3),$$

where $\mathbf{h}_T^{\text{lstm}} \in \mathbb{R}^h$ is the hidden state at the final timestep T, and $h = 128$ in our experiments. This recurrent module helps resolve transient ambiguity in single frames and promotes smooth mode selection by leveraging prior visual context.

Algorithm 1 Forward Pass of Locomotion Classifier

Require: RGB sequence $\{\mathbf{x}_t^{\text{rgb}}\}_{t=1}^{T}$, Depth sequence $\{\mathbf{x}_t^{\text{depth}}\}_{t=1}^{T}$
Require: Pretrained ResNet18 encoder f_{rgb}, Depth CNN f_{depth}, LSTM $\mathcal{L}$, Classifier ϕ
Initialize feature sequence $Z = [\,]$
for $t = 1$ **to** T **do**
  $\mathbf{h}_t^{\text{rgb}} \leftarrow f_{\text{rgb}}(\mathbf{x}_t^{\text{rgb}})$ ▷ Extract RGB features
  $\mathbf{h}_t^{\text{depth}} \leftarrow f_{\text{depth}}(\mathbf{x}_t^{\text{depth}})$ ▷ Extract depth features
  $\mathbf{z}_t \leftarrow [\mathbf{h}_t^{\text{rgb}}; \mathbf{h}_t^{\text{depth}}]$ ▷ Concatenate features
  Append $\mathbf{z}_t$ to Z
end for
$\mathbf{h}_T^{\text{lstm}} \leftarrow \mathcal{L}(Z)$ ▷ Pass sequence through LSTM
$\hat{\mathbf{y}} \leftarrow \phi(\mathbf{h}_T^{\text{lstm}})$ ▷ Predict class logits
$\hat{y} \leftarrow \arg\max(\hat{\mathbf{y}})$ ▷ Final locomotion prediction

Classification head. The final LSTM output is passed through a fully connected classifier with one hidden layer:

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{W}_2 \cdot \text{ReLU}(\mathbf{W}_1 \mathbf{h}_T^{\text{lstm}} + \mathbf{b}_1) + \mathbf{b}_2),$$

where $\hat{\mathbf{y}} \in \mathbb{R}^3$ is a probability distribution over locomotion classes. We train the model using cross-entropy loss:

$$\mathcal{L} = -\sum_{c=1}^{3} \mathbf{1}[y = c] \log \hat{y}_c.$$

Implementation details. We implemented the visual encoders, LSTM module, and classifier head using PyTorch. The pretrained ResNet backbone is imported from `torchvision`, while the depth encoder is trained from scratch using simulated depth maps. Our custom dataloader dynamically assembles fixed-length frame sequences from continuous labeled segments, ensuring that the target label corresponds to the locomotion mode in the final frame. Data augmentation—such as color jittering and horizontal flipping—is applied only to RGB images during training to simulate semantic diversity. Training and evaluation scripts are implemented from scratch.

To clarify the model's forward computation, we summarize the inference pipeline in Algorithm 1. For each timestep, RGB and depth features are extracted using independent encoders, concatenated, and passed through a single-layer LSTM to capture short-horizon temporal dependencies. The final LSTM hidden state is fed into a fully connected classifier to predict the locomotion mode.

Motivation. Depth input is essential for spatial reasoning, particularly in distinguishing climbable obstacles from walkable paths. RGB complements this with semantic cues such as texture or obstacle type. The LSTM adds critical temporal reasoning by capturing motion transitions over time—e.g., distinguishing between pre- and post-climb postures. Finally, pretrained CNN weights improve performance and convergence speed given the limited visual diversity in our simulated environments.

To understand the contribution of each component, we conduct a comprehensive ablation study (Section 4) across seven variants, evaluating the effect of input modality (RGB vs depth), temporal modeling (with or without LSTM), and encoder initialization (random vs pretrained).

4. Results

We evaluate our model using both quantitative and qualitative analysis to measure classification performance, interpret model behavior, and compare against relevant baselines. All evaluations are conducted on a held-out test set consisting of 202 labeled sequences. This section outlines the metrics, training details, ablation experiments, and our analysis of learned behavior.

4.1. Evaluation Metrics

Our primary evaluation metric is classification accuracy on the test set:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left[\hat{y}_i = y_i\right],$$

where $\hat{y}_i$ is the predicted locomotion class for test example i, and y_i is the ground-truth label. We also analyze confusion matrices to understand class-specific performance and identify common misclassifications.

4.2. Training Details and Hyperparameters

We trained all models using the Adam optimizer with a learning rate of 1×10^{-3}, selected based on preliminary grid search over $\{10^{-2}, 10^{-3}, 10^{-4}\}$. A mini-batch size of 32 provided a good trade-off between convergence stability and GPU memory usage. Each model was trained for 20 epochs with early stopping based on validation loss. No additional cross-validation was performed due to the stability of our dataset, but we used a held-out 15 percent validation set.

The depth encoder was initialized randomly, while the RGB encoder used pretrained ResNet18 weights from ImageNet. We applied basic data augmentation (random horizontal flip and color jitter) on RGB images to reduce overfitting and improve generalization to lighting and terrain texture variation.

4.3. Ablation Study

To assess the contribution of each architectural component, we conduct a detailed ablation study across seven model variants, summarized in Table 1. Our final model

Table 1: Classification accuracy (%) on the test set across model variants.

Model	Input Modalities	LSTM	Accuracy
A1	RGB only	×	88.3
A2	Depth only	×	91.5
A3	RGB + Depth	×	92.6
A4	RGB + Depth	✓	93.6
B1	RGB only (pretrained)	×	95.7
B2	Depth only (pretrained)	×	97.9
B3 (Ours)	RGB + Depth (pretrained)	✓	**98.7**

(B3) uses both RGB and depth inputs, a recurrent LSTM module for temporal aggregation, and a pretrained ResNet backbone for RGB encoding.

From these results, we observe several important trends. First, depth-only models (A2, B2) consistently outperform RGB-only counterparts (A1, B1), suggesting that depth images provide more reliable cues for locomotion classification in our simulated domain. This is expected, as depth captures geometric structure invariant to illumination or texture—making it highly effective for distinguishing between actions like climbing over a bin or crawling under a bar. RGB alone, while helpful for capturing semantics (e.g., floor material or obstacle type), suffers under challenging visual conditions such as shadows or uniform textures.

Combining RGB and depth inputs (A3, A4, B3) generally leads to improved performance over single-modality inputs. The additional semantic information from RGB frames complements the spatial structure in depth, which helps resolve certain edge cases where geometry alone may be ambiguous (e.g., distinguishing a climbable bin from a high step).

Temporal reasoning through the LSTM module (A4, B3) further improves performance by maintaining short-term motion memory. This enables the model to disambiguate visually similar states based on motion context—such as differentiating a crouched posture during crawl from the start of a climb. It also promotes smoother, more temporally consistent predictions across frame sequences, reducing flickering in locomotion mode selection.

Pretrained encoders (B1–B3) provide a substantial boost across all configurations. Compared to training from scratch (A1–A4), models with pretrained ResNet backbones learn more effective visual representations with fewer data, achieving faster convergence and higher final accuracy. This highlights the importance of leveraging transfer learning even in simulated environments with limited visual diversity.

Overall, our full model (B3) achieves the highest test accuracy of 98.7%, demonstrating the effectiveness of combining depth and RGB inputs, temporal modeling, and pretrained encoders for egocentric locomotion classification.

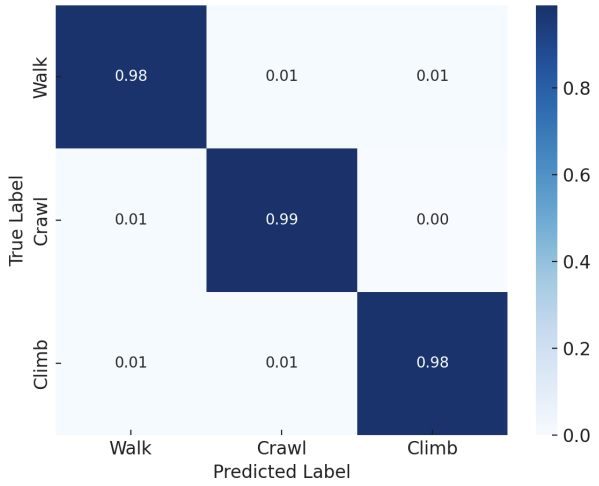


Figure 6: Confusion matrix on test set using our full RGB-D + LSTM model.

4.4. Confusion Matrix Analysis

Figure 6 presents the normalized confusion matrix for our final model (B3) evaluated on test examples. The classifier demonstrates strong performance across all three locomotion classes, with accuracies exceeding 98% per class. The model classifies crawling behavior with the highest precision (99%), while minor confusion is observed between walk and crawl, likely due to transitional postures or low-angle egocentric views. Climb predictions remain robust, benefiting from distinctive geometric cues captured in the depth modality. Overall, the model exhibits consistent and high-confidence predictions with minimal cross-class misclassification.

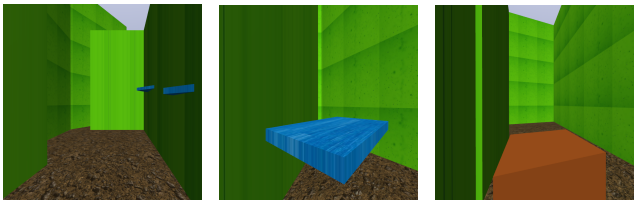
4.5. Qualitative Results

To understand what cues the model uses, we visualize sample RGB-D inputs with predicted labels in Figure 7. Walk predictions typically show open terrain and upright view, while crawl corresponds to tunnel-like perspectives. Depth is especially helpful for climbing detection, revealing large vertical surfaces or discontinuities in the ground.

4.6. Inference Efficiency

We benchmark inference latency across models in Table 2, using a single NVIDIA RTX 3060 Ti GPU. Our lightweight CNN + LSTM classifier achieves real-time inference at 4.4 ms per frame, making it suitable for onboard control in closed-loop robotic systems.

To contextualize our efficiency claims, we compare against recent vision-language models (VLMs) that perform a similar high-level planning task from egocentric vision.



(a) Walk (b) Crawl (c) Climb

Figure 7: Qualitative examples from the test set showing RGB-D input and predicted locomotion mode. Each example corresponds to a correct prediction by our B3 model.

Table 2: Average inference time (ms) per frame (NVIDIA RTX 3060 Ti).

Method	Architecture	Time (ms)
CommonSense [4]	GPT-4o	2500
SpatialBot [3]	Phi-2 + SigLIP	4680
Ours (B3)	CNN + LSTM	**4.4**

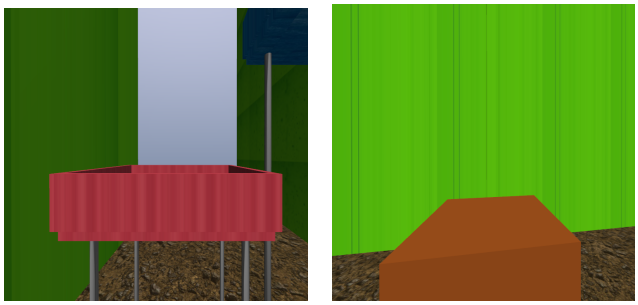
CommonSense VLM [4] was proposed as a high-level locomotion planner for quadruped robots, using GPT-4o to generate symbolic actions from RGB-D inputs. SpatialBot [3], a state-of-the-art VLM for spatial reasoning, uses RGB-D inputs with the Phi-2 language model and SigLIP vision backbone to infer spatial relations and action plans.

In both cases, VLMs are prompted with egocentric visual inputs and task instructions to infer locomotion actions. While powerful, these models are computationally heavy and exhibit significant latency at inference time. For reference, the prompt templates used for both VLM baselines are provided in Appendix A. As shown in Table 2, our supervised CNN + LSTM approach offers more than 500× faster inference than VLM-based methods, without sacrificing accuracy.

4.7. Failure Cases and Discussion

Figure 8 illustrates common failure cases observed in our final model. Misclassifications typically occur near subtle transition points between locomotion modes. For example, the model occasionally predicts crawling slightly too early or climbing slightly too late. These errors often stem from inconsistencies in training labels, where mode transitions are not precisely aligned across demonstrations. Additionally, environmental factors such as shadows or textureless surfaces can degrade depth reliability, and RGB frames in low-light or overexposed scenes further exacerbate ambiguity.

Despite these challenges, the model performs consistently well across most test scenarios. The use of pretrained visual encoders improves robustness to out-of-distribution textures and lighting, while depth provides strong spatial priors. Finally, temporal smoothing via LSTM helps stabilize predictions across frames, reducing mode flickering commonly seen in frame-wise baselines.



(a) Crawl predicted early before obstacle. (b) Climb predicted late at the obstacle.

Figure 8: Representative failure cases from the test set. Misclassifications typically arise near transition zones due to labeling ambiguity or sensor noise.

4.8. Overfitting and Mitigation

While training accuracy converges quickly due to the relatively clean labels, we observe minimal overfitting on the validation set. Pretrained backbones and moderate augmentations (jitter, flips) mitigate this risk. We deliberately exclude stronger augmentations like cutout or random erasing to avoid domain shift away from realistic robotic perception.

5. Conclusion

In this work, we proposed a lightweight and efficient locomotion mode classifier for humanoid robots using egocentric RGB-D perception. Our approach combines a CNN-based encoder, temporal reasoning via LSTM, and multimodal inputs (RGB and depth) to achieve accurate and real-time decision-making. Through extensive ablations, we showed that depth-only models provided strong spatial awareness, RGB-only models were sensitive to lighting and texture, and their combination offered moderate gains by introducing semantic context. Temporal modeling with LSTM further improved stability in ambiguous transition zones by incorporating short-term history.

Our best model (B3), which integrates pretrained CNN encoders, dual-modality input, and LSTM memory, achieved 98.7% accuracy on the test set. It also significantly outperformed state-of-the-art vision-language planners such as SpatialBot and CommonSense VLM in inference speed by over 500×, making it suitable for real-time deployment in control loops.

6. Future Work

While our model performs well in simulation, extending its robustness to real-world environments remains an important direction. Future efforts could focus on domain adaptation using sim-to-real transfer techniques, data augmentation with photorealistic renderings, or fine-tuning on real-world egocentric datasets. Improving robustness under perceptual noise—such as motion blur, camera tilt, or overexposure—would further enhance real-world reliability.

Additionally, integrating high-level language-conditioned control could enable more interpretable and flexible planning. Rather than fixed classification, a language-guided policy could allow dynamic goals like "climb to the top bin" or "crawl under the bar." Finally, scaling to more complex scenes and supporting additional locomotion modes (e.g., jump, sidestep, or squeeze) would broaden applicability to more diverse terrains and missions.

7. Contributions & Acknowledgements

I, Tae Yang, performed all aspects of this project independently. This includes conceptualization, literature review, simulation environment setup, camera modeling and data collection, model design and implementation, extensive ablation studies, result analysis, and manuscript preparation.

I acknowledge the use of the 'robosuite' framework [15] for setting up the MuJoCo simulation environment and the 'ToddlerBot' [12] humanoid robot model as the foundation for the simulated agent.

References

[1] D. Belter and P. Skrzypczyński. Terrain analysis for rough terrain mobile robots using proprioceptive sensors—a review. *Robotics and Autonomous Systems*, 85:172–201, 2016.

[2] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, B. Gerkey, K. Conley, W. Meeussen, and J. K. Salisbury. Towards autonomous robotic systems for emergency response. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 592–597. IEEE, 2010.

[3] W. Cai, I. Ponomarenko, J. Yuan, X. Li, W. Yang, H. Dong, and B. Zhao. Spatialbot: Precise spatial understanding with vision language models, 2025.

[4] A. S. Chen, A. M. Lessing, A. Tang, G. Chada, L. Smith, S. Levine, and C. Finn. Commonsense reasoning for legged robot adaptation with vision-language models, 2024.

[5] H. Fu, B. Hannaford, D. Fox, and J. Müller. Coupling hierarchical planners and control policies for active locomotion perception. In *Conference on Robot Learning (CoRL)*, 2021.

[6] A. Jain, E. Weng, B. Ichter, T. Zhang, A. Srinivas, I. Mordatch, and S. Levine. Learning visual foresight for locomotion planning on challenging terrain. In *Robotics: Science and Systems (RSS)*, 2023.

[7] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal. Fast, robust quadruped locomotion over challenging terrain. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2665–2670. IEEE, 2010.

[8] Y. Liu, K. Lu, D.-A. Huang, C. Finn, T. Darrell, and L. Fei-Fei. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *Transactions on Machine Learning Research (TMLR)*, 2023.

[9] X. B. Peng, G. Berseth, K. Yin, and M. van de Panne. Learning motor control primitives for humanoid locomotion. In *Robotics: Science and Systems (RSS)*, 2019.

[10] N. Rudin, J. He, J. Aurand, and M. Hutter. Parkour in the wild: Learning a general and extensible agile locomotion policy using multi-expert distillation and rl fine-tuning, 2025.

[11] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and A. Gupta. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9339–9347, 2019.

[12] H. Shi, W. Wang, S. Song, and C. K. Liu. Toddlerbot: Open-source ml-compatible humanoid platform for loco-manipulation, 2025.

[13] L. Wellhausen, R. Ranftl, and M. Hutter. Rough terrain locomotion with a quadrupedal robot using height maps and deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(2):2874–2880, 2021.

[14] T. Ye, J. Zhang, K. Fu, A. Zeng, D.-A. Huang, E. Yumer, L. Fei-Fei, and S. Savarese. Egocentric vision-based affordance learning for locomotion planning. In *Conference on Robot Learning (CoRL)*, 2022.

[15] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2025.

A. Vision-Language Prompt Templates

A.1. SpatialBot Prompt

We use the following prompt for inference with SpatialBot [3]:

```
Given the RGB and depth image, decide
what locomotion mode the robot should use
to move forward safely and efficiently.
The robot can choose between 'walk',
'crawl', or 'climb'.  If the path ahead
is clear for at least 0.6 meters, choose
'walk'.  If there is a low obstacle or
gap that the robot can crawl under and
it is within 0.6 meters, choose 'crawl'.
If there is an obstacle that the robot
can climb (robot height 0.6m) and it
is within 0.6 meters, choose 'climb'.
Otherwise choose the safest mode.  Always
choose the locomotion mode in advance,
switching at least 0.6 meters before
reaching an obstacle or gap.  Answer ONLY
with the word:  walk, crawl, or climb.
```

The input consists of two images: a head-mounted RGB image and a corresponding depth image encoded in RGB format.

A.2. GPT-4o Prompt

We use the following detailed prompt with GPT-4o [4]:

```
You are a high-level decision-making
module for a humanoid robot navigating
complex terrain.  The robot must choose
one locomotion mode:  walk, crawl, or
climb.  The robot receives two inputs:
1.  An egocentric RGB image from a
head-mounted camera 2.  A depth image,
where pixel values range from 0 (black,
0.0 meters) to 255 (white, 2.0 meters).
Pixel value 128 is approximately 1.0
meter away Robot context:  The robot's
height is about 0.6 meters The camera is
mounted at the head, so it is also about
0.6 meters above the ground Use the depth
image to analyze the space directly in
front of the robot and detect obstacles
or free space.  Focus on whether there is
open space on the floor, raised obstacles
with space underneath, or full-height
barriers Interpretation rules:  - If the
lower region of the image is bright and
unobstructed, the floor is open - If a
dark band floats above a bright floor,
that indicates a raised obstacle with
empty space below - If the obstacle is
dark from bottom to top and touches the
ground, estimate if its height is less
than or equal to 0.6 meters Decision
logic:  - If there is open space for
0.6 meters, choose walk - If there is a
raised obstacle with a gap of at least
0.3 meters underneath, choose crawl -
If there is a grounded obstacle within
0.6 meters and its height is less than
or equal to 0.6 meters, choose climb -
If none apply, choose the safest option
Important constraint:  Never choose
climb if there is space underneath the
obstacle.  Prefer crawl instead in such
cases Return your answer only in walk,
crawl, or climb.
```

The model receives both RGB and depth images in base64-encoded format and is queried through the OpenAI API.