https://doi.org/10.1038/s44182-025-00031-6

Egocentric visual self-modeling for autonomous robot dynamics prediction and adaptation

Yuhang Hu[1] ✉, Boyuan Chen[2] & Hod Lipson[1,3] ✉

The ability of robots to model their own dynamics is key to autonomous planning and learning, as well as for autonomous damage detection and recovery. Traditionally, dynamic models are pre-programmed or learned from external observations. Here, we demonstrate for the first time how a task-agnostic dynamic self-model can be learned using only a single first-person-view camera in a self-supervised manner, without any prior knowledge of robot morphology, kinematics, or task. Through experiments on a 12-DoF robot, we demonstrate the capabilities of the model in basic locomotion tasks using visual input. Notably, the robot can autonomously detect anomalies, such as damaged components, and adapt its behavior, showcasing resilience in dynamic environments. Furthermore, the model's generalizability was validated across robots with different configurations, emphasizing its potential as a universal tool for diverse robotic systems. The egocentric visual self-model proposed in our work paves the way for more autonomous, adaptable, and resilient robotic systems.

Human vision plays a pivotal role in facilitating a wide range of agile behaviors. The visual feedback it provides is instrumental in tasks ranging from basic locomotion to complex motor skills[1–3]. In contrast, individuals with visual impairments often face challenges in performing these behaviors due to the absence of this feedback. This reliance on vision is not unique to humans. Many sighted animals display more sophisticated and agile locomotion patterns than their counterparts with limited visual capabilities[4,5]. This observation emphasizes the importance of vision in locomotion and suggests that robots, especially legged robots, could benefit significantly from visual feedback.

Historically, most legged robots have been designed with a "blind" approach, either relying on pre-programmed control or learning model-free locomotion policies solely from proprioceptive feedback, without explicitly modeling the environment or visual input[6–10]. This design philosophy contrasts starkly with the natural world, where sighted beings, from humans to animals, utilize their visual perception to navigate complex terrains, avoid obstacles, and perform intricate tasks with remarkable agility. Our proprioceptive senses are, by themselves, insufficient for controlling our locomotion ability[11–13]. Most of us rely on visual feedback to provide signals for sustained locomotion. Vision not only helps us navigate and avoid obstacles but also helps us keep balance and fine-tune our own kinematic self-awareness[14,15]. For example, if we see patterns on the ground moving backward, our brain can implicitly deduce that we are moving forward. In addition, our brain can learn to predict what will be the effect of certain muscle actions—in other words, our brain uses visual feedback to create and maintain our predictive dynamic self-model[16,17].

In this paper, we explore the potential of leveraging purely visual data into robotic systems to achieve locomotion and damage resilience capabilities. Our approach goes beyond merely using vision as a pose estimator. We aim to enable robots to model their own dynamics, kinematics, and morphology using visual data instead of using explicit equations, CAD models, and information from IMUs and external optical trackers which requires extensive prior knowledge or infrastructure that may not exist when robots must learn to adapt to new conditions in situ.

Our work presents how a robot can learn to predict its future state, conditioned on motor commands, using a sequence of five frames from its recent past, as seen from a single onboard camera (Fig. 1). This visual self-model can be used in several ways. For example, a planning algorithm can use the model to test the consequences of various candidate motor commands and use these predictions to choose the action most suited for a particular task—before executing any of these tasks in reality. Alternatively, a damage detector can use this model to compare predicted motion to actual motion (from visual optometry). If predicted motion deviates from actual motion, then something must have gone wrong. These applications and other examples will be demonstrated in this paper. We believe that this

[1]Department of Mechanical Engineering, Columbia University, New York, NY, USA. [2]Department of Mechanical Engineering & Materials Science, Electrical and Computer Engineering and Computer Science, Duke University, Durham, NC, USA. [3]Data Science Institute, Columbia University, New York, NY, USA.
✉e-mail: yuhang.hu@columbia.edu; hod.lipson@columbia.edu

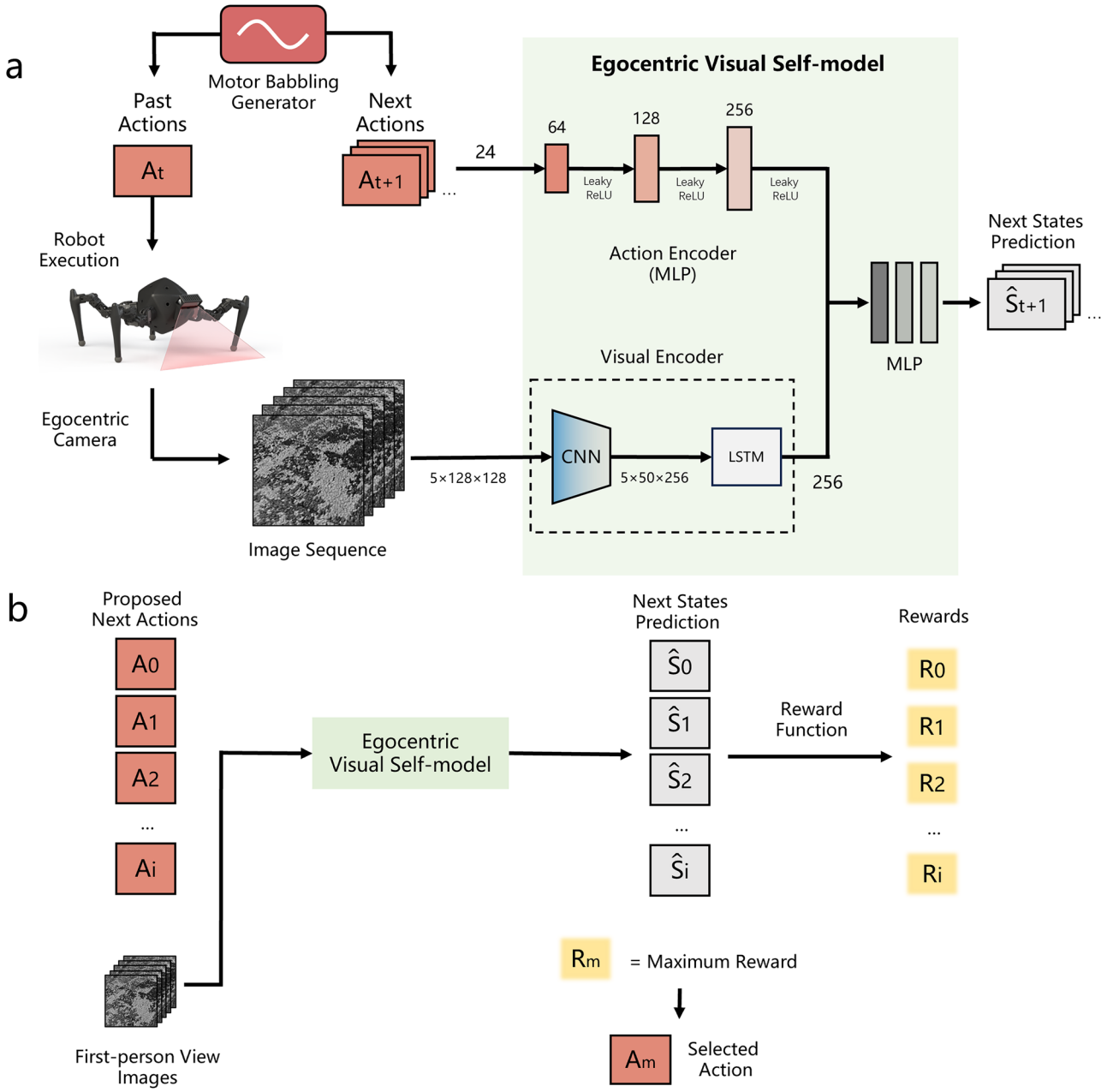

Fig. 1 | Egocentric Visual Self-Model for Predictive Control in Legged Robots. a Model Training Phase: The left image shows the process of predicting the robot's future state. Motor babbling generator produces random actions A_t for robot execution. The onboard camera captures five sequential images. These images are processed through a visual encoder. Meanwhile, the motor babbling also proposes a set of subsequent actionsA_{t+1}, which are processed through an action encoder. The outputs from both encoders are concatenated and then passed through MLP to predict the next robot state $\hat{S}_{t+1}$. **b** Deployment Phase: The right image illustrates the real-time control strategy for the robot. Multiple next-action proposals from the motor babbling generator are input into the egocentric visual self-model with first-person view images. This model predicts the future robot state for each proposed action$A_{0...i}$. Based on these predictions, rewards $R_{0...i}$ are computed for each predicted state$S_{0...i}$. The robot then selects the action $A_m$associated with the highest reward R_m for the next move.

approach can be extended, allowing robots to exhibit more agile and adaptive behaviors, much like their biological counterparts.

Self-modeling is a form of model-predictive-control (MPC) that decouples (deconvolves) a model of the robot (the "self-model") from a model of its environment and task. The key idea is that while the task and environment of a robot may change frequently, the robot itself is relatively constant across different tasks and environments. Therefore, separating out the self-model from the model of everything else and reusing the self-model for new tasks can simplify adaptation in situations that involve lifelong learning under variable tasks.

For example, if a tennis player must learn how to play badminton, they may need to learn new game rules, but will not need to relearn how to run across a court or grasp a racquet, because running and grasping are common actions performed by the same body. In this way, self-modeling facilitates transfer learning[18].

Importantly, the value of a self-model as compared to model-free learning increases with the complexity of the robot[19,20] and the complexity of the task. Therefore, finding efficient ways to create accurate self-models is extremely valuable and will offer a significant practical advantage as robot complexity continues to increase in the future.

The use of self-models for robot control is not new. Almost all robots today use some form of model predictive control. These forward models are either analytical or data-driven. Analytical self-models are programmed using equations, or using simulated physics and CAD geometries[21–23]. Data-driven models are learned using data captured from inertial measurement units (IMUs) and/or external tracking devices such as cameras and optical trackers[24–33].

Data-driven self-modeling acquires and maintains a computational representation of the robot morphology, kinematics, and dynamics through a machine-learning process that is typically bootstrapped during an initial period of safe "motor-babbling", and potentially continued over the lifetime of the machine.

Our own past work involved learning self-models using data from internal inertial sensors or external cameras[34,35]. Each of these self-modeling approaches however entailed its own limitation: Analytical models account only for limited dynamics. Data driven models that rely on proprioceptive sensors (e.g., IMUs) drift with time. Self-models that use external observations (e.g., external cameras) are not portable. Our goal here is to create self-models without proprioceptive or external sensors, and without relying on provided equations and existing geometric models.

Visual self-modeling is the creation of self-models using visual (camera) sensors to create robotic self-model[36,37]. In visual self-modeling, we aim to capture the entire dynamics of the robot using video images without any prior assumptions or knowledge about the robot's geometry, kinematics, or task. The visual self-modeling ability we explore here is similar, but fundamentally different, from "visual odometry". Visual odometry uses cameras to infer the position and orientation of a camera for applications such as SLAM, Augmented Reality and Virtual Reality goggle tracking. However, visual odometry does not involve predicting the dynamics of the system harboring the camera, as function of motor commands, and thus, cannot serve as a self-model for motion planning[38–40]. While we employ visual odometry as part of the learning process, we are ultimately interested in a model capable of predicting dynamics conditioned on motor actions.

The entire observation space of our robot contains only joint commands and the egocentric video feed (Fig. 1). Our key assumption is that visual data contains rich information about the ground, such as surface and texture, which will provide motion clues when viewed as a sequence of frames. Moreover, approaches based on visual modeling take advantage of the recent improvements in computer graphics, where modern simulators can offer image renderings sufficiently realistic that pre-training in simulation and sim-to-real transfer can work well for vision-based policies[41–43].

The self-model we introduce here learns the computational dynamics of the robot body in a self-supervised manner. During the modeling process, the self-model takes in a sequence of egocentric images, plus the motor commands, including current and future joint positions, to learn to predict the future states of the robot (Fig. 1). In doing so, the model implicitly learns such properties as mass distribution, geometry, friction, actuation and sensing delays, etc. Once acquired, the self-model can be used by any Model Predictive Control algorithms in real-time to satisfy various objectives.

In each timestep t_n, the commands A_t execute over a fixed period, during which time the camera takes five consecutive pictures $I_{t,0} - I_{t,4}$. These five images are fed into the Visual Encoder when the movement is complete. When the commands for the next step are generated, they are input into the action encoder. The outputs from two encoders are fused to predict the future robot states.

Our experiments reveal that legged robots with a visual egocentric self-model can predict future position and orientation changes of the robot body more accurately than robots without visual inputs. Our system can also be trained purely in simulation and transferred to the real-world without additional learning. Important to such zero-shot transfer is our domain randomization on both the simulated environments and robot dynamics, as well as data augmentation for neural network training.

Although it is no longer a challenge for a legged robot to walk on flat ground, our work is to provide evidence of the potential of vision in walking tasks to catalyze further development of more dynamic and adaptable robotic systems integrating visual perception with robotic gait control. Our key contributions can be summarized as follows:

1. This work demonstrates a legged robot successfully performing locomotion tasks using purely egocentric visual observations without relying on proprioceptive sensors. Our visual egocentric self-model network effectively fuses sequences of high-dimensional visual observations and actions to predict future robot states
2. A domain randomization and data augmentation pipeline for robot locomotion that enables effective zero-shot sim-to-real transfer.
3. Validation of the generalizability of our approach across multiple robots with varying morphologies and complexities, highlighting its potential as a universal tool for diverse robotic systems.
4. A demonstration of the ability to recover a damaged robot's locomotion capabilities by re-training the self-model using purely visual perception data collected in the real world, showcasing the adaptability and resilience of our approach in the face of unexpected challenges.

Results

Our evaluation includes various aspects of the implemented egocentric visual self-model, focusing on its ability to predict future robot states, adaptability to different robots or unseen terrains, and resilience in situations involving hardware anomalies.

Real-world experiments and robustness in unseen environments

We conducted real-world experiments to assess the performance and robustness of our egocentric visual self-model in various locomotion tasks and unseen environments (Fig. 2). The robot, equipped with an RGB camera, successfully navigated and maintained stable trajectories while moving forward, turning right, turning left, and moving backward. To validate the effectiveness of our method, we compared it with three baselines. Sinusoidal Gait baseline uses predefined, fixed sinusoidal motion patterns for robot control. It serves as the simplest method to generate locomotion without any adaptation to the environment. Gait Generator Baseline is built upon the sinusoidal gait. This baseline integrates a Central Pattern Generator (CPG) to dynamically produce new motion patterns. Unlike the static motion of the sinusoidal baseline, this generator allows more varied locomotion. Moreover, the gait generator acts as the policy to produce data for the IMU-only baseline and our method. IMU-Only Baseline leveraged the data generated by the gait generator. It replaces visual input with inertial measurement unit (IMU) data to predict robot dynamics. It allows us to isolate the contribution of high-dimensional visual information compared to lower-dimensional proprioceptive input.

To evaluate the robustness of our self-model in unseen environments and varying visual conditions, we tested our method on a rug texture used during training in simulation and a checkerboard texture that the robot had never encountered before. Figure 3 presents the mean scores by the robot for each method across three trials. Our method outperforms all other baselines, highlighting the effectiveness of leveraging egocentric visual information for robot self-modeling and control in real-world scenarios. Additionally, we conducted experiments on a carpet littered with slippery paper scraps (Fig. 4A–F) to assess the model's performance in challenging terrains. Besides, our method maintains strong performance even when tested on the unseen checkerboard texture, as shown in Fig. 4. These experiments show the adaptability of our approach to new environments and its robustness to visual perturbations. The results of the real-world experiment are presented in Supplementary Movie 1.

Evaluating the contribution of the visual encoder

To quantitatively evaluate the effectiveness of the Visual Encoder in our pipeline, we conducted an ablation study. First, we compared our method with the "Commands only—no vision" baseline. The goal of this baseline is to remove the contribution of images from the model and rely only on motor commands to predict future states, equivalent to open-loop control. Intuitively, it represents animals or humans learning to walk blindly without

Fig. 2 | Basic locomotion tasks in the real-world environment. We deployed an egocentric visual self-model on a legged robot. The robot only relies on image sequences from the front camera and action commands to achieve (**A**) moving forward, (**B**) turning right, (**C**) turning left, and even (**D**) moving backward. It learns the skill from the simulation to anticipate hundreds of possible future states by perceiving the latest visual information and motor commands it will actuate (see Supplementary Movie 1).

Fig. 3 | Evaluate the egocentric visual self-model in the real world. We compare our method with four baselines. In each experiment, the robot starts from the same pose, and each episode consists of 56 steps. After each run, we measure the final position and pose to calculate the scores.

visual input. This baseline model learns the robot's forward-kinematic body model through data-driven like previous work[35]. Therefore, contrasting our approach with this baseline, we aim to emphasize the importance of visual information. The second baseline replaces image sequences with explicit IMU data as input and remains consistent with our model in the motor action module. We used this baseline to compare the contribution of the high-dimensional visual information to the low-dimensional IMU information typically used in robotics. In the real-world experiment, we used high-accuracy IMU (HWT905-TTL MPU-9250 9-axis Gyroscope +Angle (XY 0.05° error).

We repeated each experiment 10 times ($N = 10$), where each run is an episode of 56 steps. Table 1 shows the results of our experiments in the simulation. We test our model with four terrain textures (Fig. 8). We compare our method to the baselines using mean prediction loss and

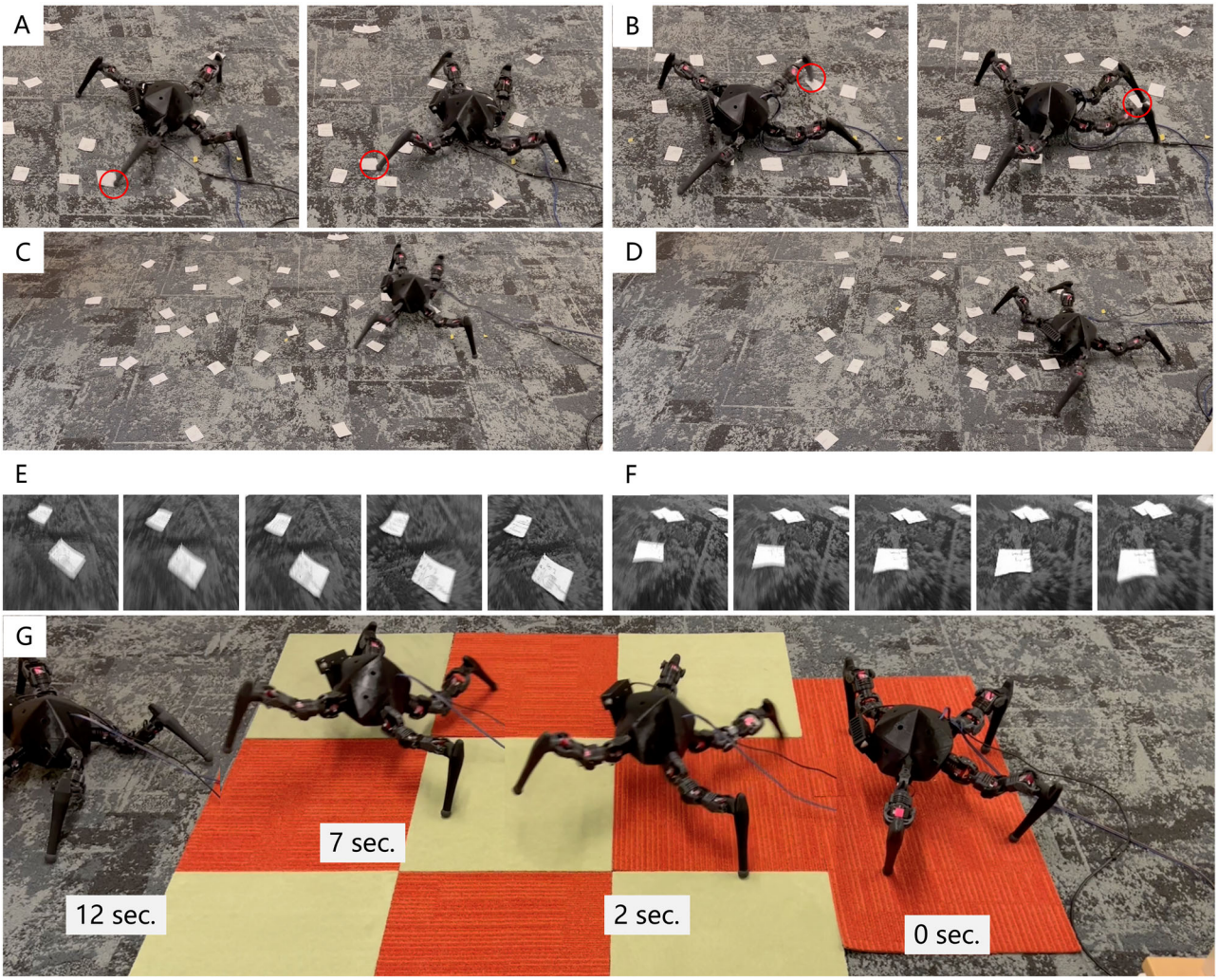

Fig. 4 | Assessing egocentric visual self-models on unseen terrain. A–E We performed the same experiments on carpets with pieces of slippery paper and terrain with checkerboard textures. **C** The robot is moving forward, and its first-view pictures are shown in (**E**). **D** The robot is turning right and its first-view pictures are shown in (**F**). **G** The robot moves forward with an egocentric visual self-model on unseen terrain.

evaluation metrics for each task. The results indicate our approach is better than any other baselines in four tasks across all terrains. We believe that high-dimensional observation should conclude the information from IMU data. Visual observation can also provide additional information about robot configuration, kinematics, and dynamics. Thus, the results of the experiment demonstrate the importance of visual information in predicting robot dynamics and kinematics.

To assess the statistical significance of the differences in prediction performance among the methods, we conducted a one-way Analysis of Variance (ANOVA) followed by Tukey's Honest Significant Difference (HSD) post-hoc tests for each task (moving forward, moving backward, turning right, and turning left). The ANOVA results confirmed highly significant differences among methods (p-values far below the 0.05 threshold in all cases). Notably, each of our four egocentric visual self-model variants ("Ours") performed significantly better than either of the two baselines ("Action Only" and "IMU Only"), while no statistically significant differences were observed among the four "Ours" models themselves. This outcome highlights the robust advantage of incorporating an egocentric vision in our self-modeling approach. Besides, it demonstrates that once trained with sufficient domain randomization, specific environmental textures did not materially affect final predictive performance. The full statistical experiment is provided in the supplementary materials.

Generalizability and transfer learning

In this section, we evaluate the generalizability of the visual encoder in our model by conducting experiments on three additional robots in the Pybullet simulation environment. The main goal is to determine whether the pre-trained visual encoder from robot 0 can be used to train the egocentric visual self-models of the new robots while maintaining performance and requiring less data during the training process. We designed two baselines and conducted quantitative evaluations to demonstrate the generalizability of our model.

To assess the generalizability of our model's visual encoder, we selected three different robots in the Pybullet simulation environment, as shown in Fig. 5. The experiment of our Robot 3 (with the second and third joints of each leg being rotated by 90°) has significant kinematic changes. These robots possess varying morphologies and complexities, ensuring a diverse and challenging set of test cases for our model. The pretrained visual encoder from robot 0 was used as the starting point for training the egocentric visual self-models of the new robots.

During the training process, the visual encoder weights were kept frozen, meaning that the model did not learn any new visual features specific to the new robots. Instead, it relied on the visual features learned from the initial robot. We used less data (200k steps) for training the new egocentric visual self-models compared to the data used for training the initial Robot 0 (400k steps).

We designed two baselines for quantitative evaluation. The first baseline, a non-visual method (NV), relies solely on motor commands and proprioceptive data for predicting robot dynamics, without utilizing any visual input. It represents traditional proprioceptive-based methods and provides a contrast to our approach by isolating the contribution of visual information to generalizability. The second baseline (IM) utilized the initial egocentric visual self-model of robot 0 without additional training for the new robots. The comparison between our method and the IM baseline highlights the substantial reduction in prediction errors achieved by fine-tuning the egocentric visual self-model with a small batch of data for each new robot. The pre-trained visual encoder's ability to capture crucial information about the robot's state, combined with the fine-tuning of the self-model, enables efficient adaptation to new robot configurations while significantly improving prediction accuracy.

The results (Fig. 5) demonstrate that our method (OM) consistently outperformed both baselines, exhibiting the lowest prediction errors across all terrains and tasks for the three new robots. These results highlight the visual encoder's ability to capture essential information about robot dynamics and facilitate efficient learning and adaptation for different robotic systems.

Table 1 | Quantitative evaluations

		Move forward	Move backward	Turn right	Turn left
Action Only	Mean	3.10E-03	1.90E-03	2.80E-03	2.30E-03
	Std	3.90E-03	1.90E-03	3.80E-03	2.40E-03
IMU Only	Mean	2.40E-03	1.80E-03	2.10E-03	2.20E-03
	Std	3.00E-03	1.90E-03	1.80E-03	2.50E-03
Ours (rug)	Mean	1.90E-03	1.40E-03	1.60E-03	1.50E-03
	Std	2.30E-03	1.70E-03	1.70E-03	1.60E-03
Ours (grid)	Mean	1.90E-03	1.40E-03	1.60E-03	1.50E-03
	Std	2.30E-03	1.70E-03	1.70E-03	1.60E-03
Ours (Color Dots)	Mean	1.40E-03	1.30E-03	1.60E-03	1.40E-03
	Std	2.20E-03	1.60E-03	1.50E-03	1.30E-03
Ours (grass)	Mean	1.40E-03	1.30E-03	1.60E-03	1.30E-03
	Std	1.50E-03	1.40E-03	1.60E-03	1.40E-03

Applicability to a humanoid robot

To further demonstrate the versatility and adaptability of our egocentric visual self-model, we conducted an additional experiment on a significantly different robot: the humanoid robot Atlas. Controlling such a sophisticated system is challenging due to its complex morphology and dynamics. To focus on testing the basic locomotion capabilities based on our egocentric visual self-model and its adaptability to different robots, we froze the upper body joints and used only 6 degrees of freedom, including the hip, knee, and ankle joints. We evaluated the performance of our method on the Atlas robot in a simulated environment. The results, shown in Fig. 6, demonstrate that our method can be successfully deployed on this robot, enabling it to move forward and turn left and right. This highlights the generalizability of our approach to robots with vastly different morphologies and complexity compared to the initial robot used in our experiments.

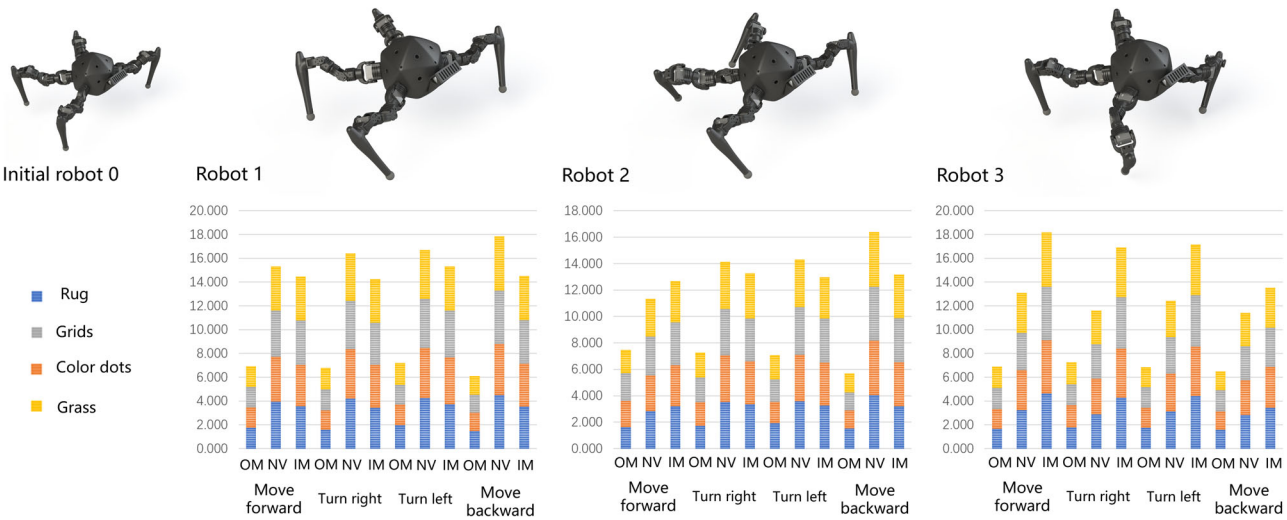

Fig. 5 | New robot configurations and quantitative evaluations in various terrains and locomotion tasks. Robot 0 is modified to create three new configurations, robot 1, robot 2, and robot 3, by altering leg-body connection orientations, resulting in significant kinematic changes due to the serial connection of leg motors, thus providing a diverse set of test cases for our visual encoder's generalizability. The lower plots show prediction errors for our method (OM), a non-visual method (NV), and the initial model (IM). Our method consistently outperforms the other methods, exhibiting the lowest errors across all terrains and tasks for the three new robots.

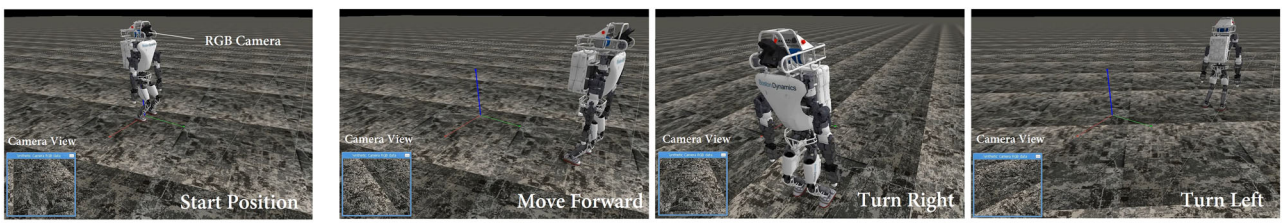

Fig. 6 | Egocentric visual self-model deployed on the humanoid robot Atlas in simulation. The Atlas robot successfully performs basic locomotion tasks such as moving forward, turning right, and turning left, demonstrating the adaptability and generalizability of our approach to complex robotic systems with vastly different morphologies.

We also conducted a quantitative evaluation of our method's performance on the Atlas robot. First, we assessed the prediction errors of our egocentric visual self-model compared to a baseline that does not use visual input (Table 2). Our method significantly outperformed the baseline, achieving much lower prediction errors across all locomotion tasks (forward, right, and left). We also compared the locomotion performance of our method against the two baselines in terms of the distance traveled in each desired direction. The performance metric is calculated using the same approach described in the Methods section under Reward Function and Score Metrics, which captures the net displacement along (or around) the specified axis of motion. Specifically, each trial's movement score reflects the robot's displacement in the desired direction, as well as any deviation or rotation from the intended heading. Table 3 shows these locomotion performance scores for each method.

These experiments further validate the adaptability and generalizability of our egocentric visual self-model to various robotic systems, including humanoid robots with complex morphologies and dynamics. The successful deployment of our method on the Atlas robot highlights its potential as a universal tool for enabling basic locomotion capabilities in diverse robotic platforms, paving the way for more advanced and agile behaviors in the future.

Autonomous anomaly identification and adaptation

The egocentric visual model can endow the robot with abilities to recognize hardware anomalies, like broken legs or damaged actuators, and recover by adapting to the new body configuration. We conducted anomaly-detection experiments using a different leg module, which has a broken end-link as shown in Fig. 7(A–C). Such damage typically happens when subjected to external force impact or fatigue fracture during testing or deployment, and is generally difficult to detect using direct sensors. Recovery usually requires human intervention to determine the cause and shut down the robot for repair. The normal and leg-damaged robot attempting forward locomotion is shown in Supplementary Movie 3 and Supplementary Movie 4.

To detect the anomaly, we first trained a deep visual odometry model using the same data as the egocentric visual self-model. The model takes 7 consecutive pictures captured by the camera between two steps as input and returns the state changes ($\Delta x, \Delta y, \Delta z, \Delta roll, \Delta pitch, \Delta yaw$) between every two steps. This information is not predictive and does not factor in motor commands. The visual odometry captures the current (not the future) state of the robot. The visual odometry model has the same architecture as our egocentric visual self-model model, consisting of the visual encoder and MLPs, but omitting the motor command encoder.

We deployed the learned visual odometry model directly on the physical robot. As shown in the first two columns of Fig. 7D, the robot can recognize that an abnormal situation (leg broken in this case) may have occurred based on a large discrepancy between predicted and actual motion. The first two rows in Fig. 7E demonstrate the difference when we use the egocentric visual self-model to control the normal robot and the damaged robot to do the same forward motion task. Under normal circumstances, the robot will choose a forward gait according to its predicted future state to achieve forward motion. However, since its right rear leg was broken, following the original gait caused it to deflect to the right. The discrepancy between predicted and actual motion can trigger an alarm. It can also trigger auto-recovery processes as follows.

In order for the robot to regain its previous ability to accurately predict future states based on visual information and motion commands, the damaged robot needs to re-learn its self-model to match its new, altered body configuration. By running motor babbling in the real environment for about 30 min, the robot collected 7000 steps of data. This data includes the motor commands of the robot and the images captured by the camera. The Visual Odometry (VO) model outputs the change of robot state from the visual information as the ground truth label for updating the visual self-model. Egocentric visual self-models can be re-trained with this new data to successfully and accurately predict future states. Once the robot states predicted by the self model match the actual robot states measured by the visual odometry, the self-model retraining has been completed. Figure 7D shows how the error becomes lower after updating the model, and the robot can recover to perform forward locomotion in Fig. 7E. Supplementary Movie 5 demonstrates the successful recovery of the robot after retraining the self-model on data from its new damaged configuration.

Table 2 | Humanoid Robot State Prediction Errors

	Our Method		No Images Input Baseline	
	MEAN	STD	MEAN	STD
Forward	5.99E-04	3.16E-05	3.38E-01	2.16E-01
Right	7.05E-04	2.24E-05	4.61E-01	1.49E-01
Left	5.77E-04	4.04E-05	4.71E-01	1.95E-01

Discussion

In this study, we introduced an innovative egocentric visual self-model that leverages sequences of images combined with motion commands to predict future robot states. The results underscore the potential of visual input in comprehending robot dynamics, especially when contrasted with traditional robot self-modeling methods that rely on IMUs or external sensors.

One of the most compelling outcomes of our research is the robot's ability to autonomously detect anomalies, such as damaged components, and adapt accordingly. This resilience is crucial for real-world applications where robots might face unpredictable challenges. The ability to autonomously detect and adapt to damage, as demonstrated in our experiments, signifies a significant step towards creating robots that can operate in dynamic and potentially hazardous environments without constant human intervention.

Furthermore, the generalizability of our model was evident when applied to robots with varying morphologies and complexities. This adaptability is paramount, especially in scenarios where robots might need to share learned behaviors or when one robot replaces another. The successful transfer of the egocentric visual self-model across different robotic configurations emphasizes its potential as a universal tool for a wide range of robotic systems.

The findings of our work highlight the importance of visual input in learning robot dynamics and the value of generalizable egocentric visual self-models for complex robotic systems. As robot complexity continues to increase and the demand for versatile, adaptable machines grows, our approach presents a significant step towards enabling robots to efficiently learn and adapt to new tasks and environments using only visual input. In future work, we aim to explore the applicability of our method to even more complex and advanced robotic systems, as well as to investigate its potential for enhancing lifelong learning and adaptation in real-world scenarios.

Table 3 | Humanoid Robot Locomotion Performance Comparisons

	Our Method		No Images Input Baseline		Gait Generator	
	MEAN	STD	MEAN	STD	MEAN	STD
Forward	1.27E+00	1.34E-01	4.06E-01	8.13E-01	4.50E-01	1.39E+00
Right	2.01E+00	3.16E-01	-6.96E-01	2.08E-01	3.46E-01	1.54E+00
Left	1.67E+00	1.94E-01	1.53E+00	1.89E-01	4.10E-01	7.95E-01

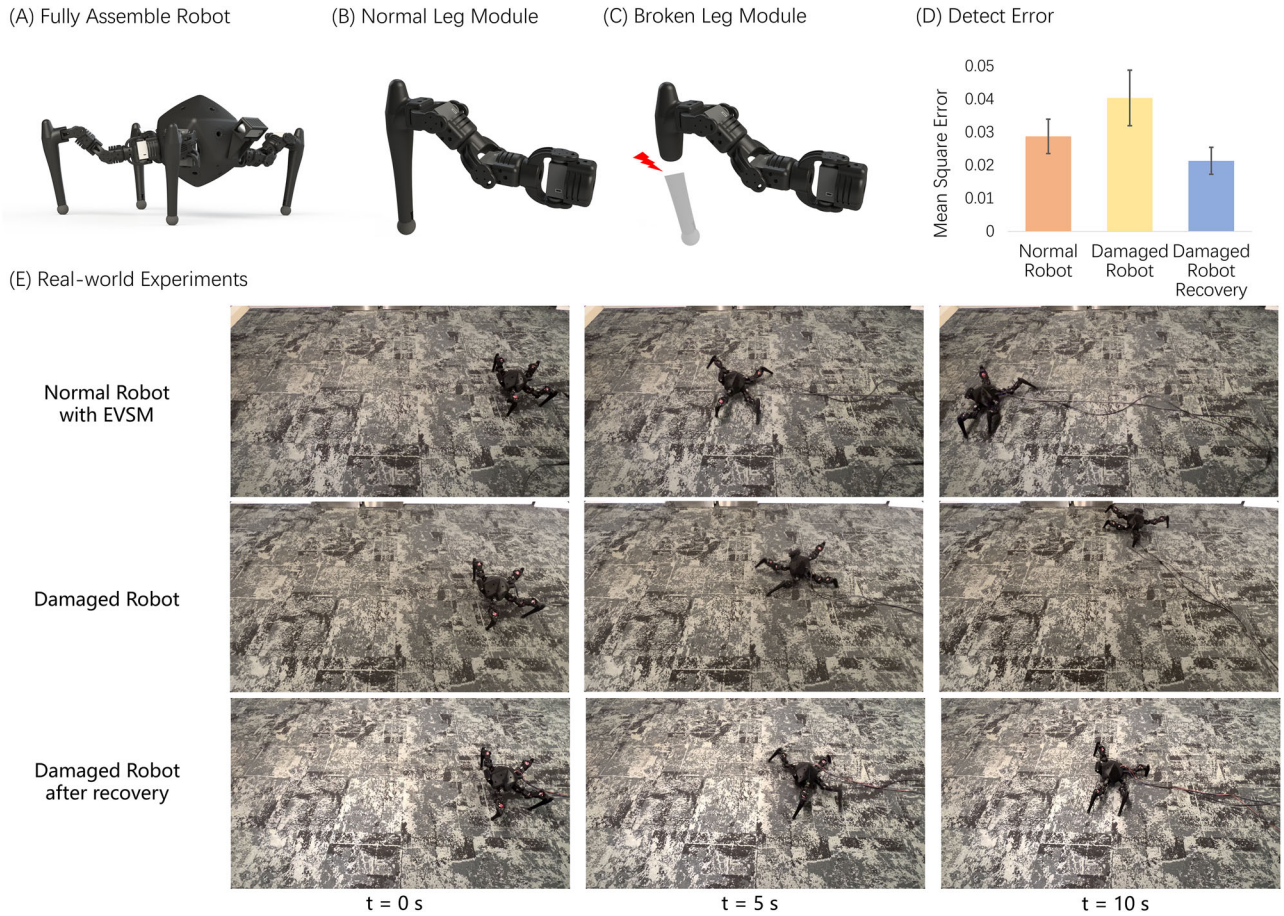

Fig. 7 | Description of anomalies. A Our fully assembled robot has four legs and a camera aimed at the ground. **B** The robot leg module consists of three motors and a rubber ball foot in contact with the ground. **C** To test the resilience ability of the robot, we cut the end-link model in half to act as a damaged robot leg. **D** Error detected by the visual odometry model. **E** Resilience experiments in the real world. The first row shows the normal robot moving forward with the egocentric visual self-model. In the second row, we replaced the right hind leg of the robot with a damaged leg and deployed the same egocentric visual self-model to control the robot. The last row presents using the updated egocentric-visual self-model on the damaged robot to control the robot to move forward.

The primary limitation of the presented method is that the robot's predictive accuracy depends on static ground. Dynamic environments, such as moving surfaces or significant terrain deformation, can introduce inconsistencies in visual cues, thereby reducing the model's reliability. Another limitation is the inability to predict long-term future states. The model relies on short sequences of visual data, which provide limited temporal information. Furthermore, the model may face challenges under extreme visual conditions, such as heavy occlusions and rapid lighting changes, which can disrupt the extraction of motion cues from visual inputs. To overcome these limitations, future work could integrate a confidence estimation mechanism into the egocentric visual self-model.

Humans use much more information from their vision to enable locomotion. We can quickly identify objects as our world coordinate system. We select static objects as reference and ignore moving objects—For example, we use the ground instead of a moving car as the origin. We also use persistent distance landmarks (such as the horizon or a distant building) as long-term visual references.

We expect that similar multi-modal strategies could also enhance visual self-modeling for robots. When properly trained, visual self-models may be able to distinguish between reliable static textures and unreliable moving textures. A combination of ground-facing cameras as well as forward-facing cameras can combine short-term and long-term visual planing.

In conclusion, our work offers a fresh perspective on robot learning and adaptation, emphasizing the power of visual data. As the field of robotics continues to evolve, approaches like ours could redefine how robots are trained, making them more autonomous, adaptable, and resilient in the face of real-world challenges.

Methods

Traditionally, visual observation is used for high-level control of legged robotic systems, such as planning, localization, and object detection, as we discussed above. The motion controller is a separate module that operates at a higher frequency than the camera. Our goal is to demonstrate the effectiveness of using egocentric visual observations for legged robot motion and propose a way to directly control the robot locomotion using vision, even though the overall controller operates at a lower frequency. An intuitive way to address these problems is to train a network that takes images as input and outputs action commands. However, the main challenges of using RGB images as observations are the high computational cost of visual information and various environmental changes, resulting in incoherent actions and the inability to perform locomotion. We propose a method for training a visual self-model that takes a sequence of images with a list of commands produced by a prior action generator as inputs and outputs the future state of the robot. Specifically, we use convolutional neural networks and memory-based networks to process sequential visual data and map action commands into a high-dimensional latent space. Finally, we can use the objective function to access different action commands to perform various tasks.

Legged robot hardware

We designed a four-leg robot with 12 degrees of freedom from 3D printed and commonly available electronic parts. Specifically, each leg contains

three Hiwonder LX-224 serial bus servos with a 0.3° accuracy and 20 kg.m torque at 7.4 voltage. We used position control to command the actuators from an onboard Raspberry Pi 4 controller. We equipped the robot with a front RGB camera with 87° × 58° field of view and 60 frames per second facing down to the ground by 41 degrees. The onboard controller and the camera are connected to a desktop with one NVIDIA 3090 GPU.

Data representation

The robot observation includes the five most recent visual images from the front camera. Instead of directly using the camera readings, we crop the input RGB images from 320 × 240 × 3 to 240 × 240 × 3 and resize them to 128 × 128 × 3. Furthermore, we only use the gray-scale image rather than the raw RGB input to reduce the dimension of the visual observation by a large margin. Such a process is critical for real-time control of the physical robot. This is because all program steps are serial. By having a smaller network to process the reduced visual inputs, we can save the cost of the decision-making process during inference time, thus avoiding losing too much information about the robot states while the robot moves constantly.

In addition to the egocentric visual observations, the robot also takes both its previous–step and its next–step actions as inputs. Since we use position control for the robot, the action commands are encoded as a vector with 24 numbers, containing 12 angles from the previous step and 12 angles from the next step. Our egocentric visual self-model aims to predict the robot states from its visual observation and next-step action. However, since we do not rely on any GPS-like systems to localize the robot, we do not have the pose of the robot in the global world coordinate. We therefore define the robot state as the change of the robot position and orientation in the robot coordinate frame, known as Δx, Δy, Δz, $\Delta roll$, $\Delta pitch$, Δyaw.

We collect data in the Pybullet [1], a robotics physics simulation (see Supplementary Movie 2). As we presented above, the objective of the visual self-model is to predict the future robot state given the latest sequence of five frames and action commands. The robot needs to have the ability to determine the motion from the five frames captured by the first-person camera aiming at the ground and to predict the influence of the action commands it will take. Initially, we used a random policy that uniformly samples actions in an action space normalized to [−1,1] representing robot joints -90 degrees to 90 degrees, and record images and robot state. However, this method is ineffective in reducing model loss during training because the data we collected do not match the data required for the robot to achieve locomotion. Besides, the high-dimensional observation space combined with action space is too large and even more extensive when we transfer the trained model to the real world using domain randomization.

Architecture and usage of the model

The architecture of the egocentric visual-self model is implemented as a hybrid of residual networks, and recurrent neural networks[44] as shown in Fig. 1a. The robot receives two types of inputs at each step: five consecutive grayscale images and a vector of 24 action commands.

The visual information comprises five consecutive images with 128 × 128 size. In our implementation, these five frames are captured at 60 FPS, covering a total of approximately 83 ms (5 frames × 16.7 ms per frame). This short temporal window is sufficient to capture local motion cues (akin to optical flow) while remaining computationally manageable for real-time inference. The action command comprises a vector of 24 motor angle values. To merge action commands with the images, we designed two encoders responsible for each type of observation: a Visual encoder and an action encoder. The visual encoder consisted of 48-convolutional layers[44,45], followed by five Long Short-Term Memory (LSTM) modules[46], and an action encoder consisting of three fully connected layers with a Leaky Rectified Linear Unit.

The Action Encoder maps the action vectors into some high-dimensional latent space through the fully-connected layer networks. The Visual encoder extracts information using a convolutional neural network and leverages LSTM units to process the features and embed the concept of time, allowing the model to learn the temporal dynamics within the sequential pictures[47]. Then, both outputs are concatenated and passed through five fully connected layers. Each layer is followed by a layer of Leaky ReLU activation function, and the output comprises six numbers representing the robot future change in position and orientation.

To train the model, we used actions generated by a CPG sinusoidal gait generator optimized for forward locomotion in simulation using simple Hill Climbing[48]. While the model will be used later for activities other than forward motion, we found that training the model using actions from a CPG is better than training the model on actions from a purely random motion generator. This suggests that purely random actions are inefficient for self-supervision.

During deployment, the robot proposes candidate motor commands. Each set of motor commands is fused with five recent gray-scale images to form the input data. Since the egocentric visual self-model runs in parallel on the GPU, we can obtain future states of the robot corresponding to multiple sets of candidate actions in a single cycle. This process is like a robot imagining its own future states through different trials. The robot then selects the action most suited for its objective functions. Therefore, a planner needs enough action candidates to allow the model to select the optimal action. The maximum number of motor commands per model run is the batch size on the GPU. We were able to test 100 candidate motor commands in parallel on an NVIDIA Geforce RTX 3090.

On our system, the visual self-model pipeline ran at about 5 Hz during our experiments. We trained the egocentric visual self-model in a simulation with a rug-like texture similar to the real world. Unlike plain ground texture, the rug texture has many features that can give the Visual Encoder plenty of information for optical flow. We modified the imported terrain and texture maps in simulation for each episode to avoid overfitting to a single texture.

The CPG (used during the training phase only) created gaits for moving forward. However, once the self-model was trained, the robot could use the self-model to accomplish other tasks, such as moving backward or turning, suggesting that our method can provide consequences of action significantly different from those generated forward-moving gait. Therefore, we believe that the egocentric visual self-model can extrapolate somewhat outside the space of actions used in training and hopefully be useful for a broad range of locomotion tasks.

Table 4 | Effect of sequence length on model performance

Sequence length (Frames)	Mean	Std
1	0.042308	0.001156
2	0.028885	0.000137
3	0.027686	0.000281
4	0.027272	0.000508
5	0.027072	0.000764
6	0.027008	0.000166

Effect of sequence length on model performance

To evaluate the impact of visual sequence length on the model performance, we conducted additional experiments where the model was fed with different numbers of consecutive frames. The sequence lengths ranged from 1 to 6 frames, and the model was trained and tested on rug-textured data using 50,000 steps for each configuration. Each experiment was repeated three times to compute the mean and standard deviation of the final testing loss.

The results, summarized in Table 4, demonstrate that increasing the sequence length improves the predictive accuracy, as indicated by the decreasing mean loss values. Notably, the performance gain diminishes as the sequence length increases beyond four frames, suggesting diminishing returns for additional frames in this specific setup. The standard deviation of the loss also decreases with longer sequences, indicating increased stability and robustness of the model.

These findings align with the expectation that longer visual sequences provide richer temporal information, allowing the model to better infer

motion dynamics. However, the diminishing performance gains suggest that practical considerations, such as computational cost and real-time constraints, should guide the choice of sequence length in specific applications.

Robot gait generator

In order to allow the model to be trained on data that is more representative of the robot locomotion, we design a gait generator to produce the action sequences for our robot to select for locomotion. The gait generator is based on Central Pattern Generator (CPG), used widely in the legged robot system[49,50]. Each motor action θ_i is generated by a single sinusoidal function with the same period as shown below: Locomotion

$$\theta_i(t) = a_i \times sin(t/T \times 2\pi + b_i) + c_i. \quad (1)$$

We use a mathematical optimization algorithm Hill Climbing[48] to optimize the parameters a_i, b_i, c_i in sinusoidal gait functions. Initially, we randomize the coefficients and test the gait in the simulation using 200 steps. The best gait parameters will be replicated into 20 copies. Among 16 copies, one of the coefficients is initialized randomly, and the other four individuals are initialized entirely from scratch. After each epoch, the rewards of all individuals are computed, and the set of parameters that can reach the highest reward is selected to iterate this process. The formal objective function is:

$$R(\Delta x, \Delta y) = \sum_{t=1}^{n} (100 \times \Delta y - 50 \times |\Delta x|). \quad (2)$$

We finish the optimization by 2 h parallel computation on a 16 cores CPU core computer and obtain an ideal sinusoidal gait that can make the robot move forward in the simulation. We applied Gaussian[51] function to our sinusoidal gait to produce new action a_n close to our optimized sinusoidal gait action a_o. These new actions can be written as:

$$a_n(i,t) = Gaussian(\mu = a_o(i,t), \sigma) \quad (3)$$

Compared to random motor babbling, using the optimized sinusoidal function with Gaussian noise provides action sequences that are closed to the optimized gait action and avoid the robot submerging in useless movement data, like crawling on the ground or flipping its body. In our tests, $\sigma = 0.2$ can provide good performance. Under this parameter, the robot can reach various states on the ground without falling easily. The gait generator is not only for collecting data for training visual self-model but also sampling action during the testing procedure. Because it can satisfy our purpose of generating a gait similar to the initially optimized gait by adding certain randomness, it can propose new and better gaits to accomplish different tasks by defining an objective function.

Environments details

In this work, we have seven different environment domains, as shown in Fig. 8. To reduce the gap between the simulation and the real-world environment, we use the same outer morphology of the main body CAD of our physical robot. Other robot CAD parts are simplified when building our URDF files to speed up the simulation computation. Also, we measure the mass of all the components and set the same number in the file. Besides, we calibrate the number of simulation timestamps in each step by running the same motor trajectories in the real world and the simulation. The trajectories are hand-designed to move each joint from a lower limited position to a limited upper position. We set 60 simulated steps for each action command to ensure the motor commands can be fully executed in the simulation. We collect 100 thousand steps for each texture (A-D) in the simulation to train the visual-self model.

Randomization and data augmentation

To perform better results, we randomize three aspects of the simulated environment: observation, action, and friction, and use data augmentation during the training procedure.

Observation noise: The rug picture contains many features that our model can learn to utilize, but we only have one rug picture. Therefore, we randomly crop the original image into 1000 images and scale them to the same size. When collecting the data from the simulation, we sample these 1000 cropped images and also rotate and translate them randomly. During the training, the brightness of five observation images is changed arbitrarily. Apart from the brightness, we also blur the images by adding Gaussian noise. The details of noise levels are described in Table 5. Except for reducing the gap between simulation and the real environment, all of these processes

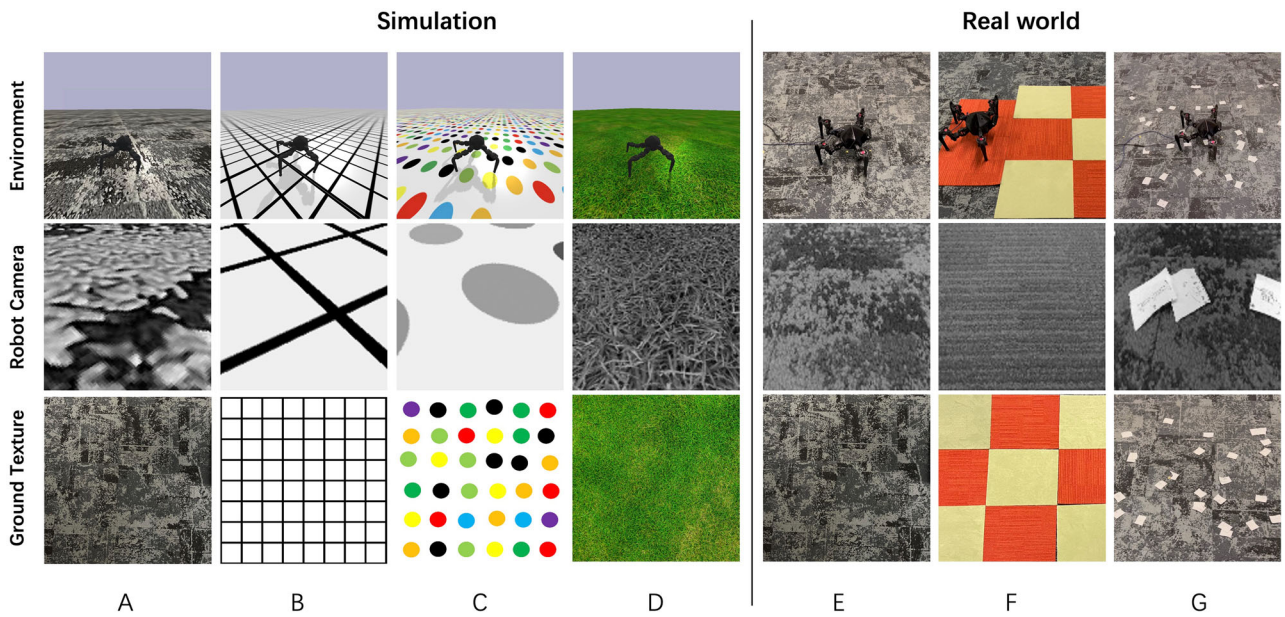

Fig. 8 | Five environments of our experiments. The left four columns are simulated environments with different terrains: rug (**A**), grids (**B**), color dots (**C**), grass (**D**). The columns on the right side are terrains in the real-world environment: rugs (**E**), checkerboard (**F**), and rugs with scraps of paper (**G**).

Table 5 | Details of the randomization

Observation noise parameter	Range
Crop range (ratio of the original image)	uniform [0.1, 1]
Rotation Range	uniform $[-\pi, \pi]$
Translation Range	uniform [−3, 3]
Brightness Range	uniform [0.1,10]
Gaussian Blur Kernel Size	uniform [3, 41]
Gaussian Blur Kernel Standard Deviation	uniform [0.1,5]
Motor Torque Range	uniform [1.5,2]
Motor Position Range	$N(0, 0.05)$
Friction Range	uniform [0.5,0.99]

Table 6 | Hyperparameters for training visual self-model

Hyperparameters	Value
Optimizer	Adam
Initial learning rate	1e-4
Batch size	128
Input size	[5 × 128 × 128, 24]
Output size	6
Hardware configuration	16 cores CPU + 1 GPU

are designed to force our model to learn the relationships among the five pictures rather than the patterns within the pictures.

Action noise: The robot is powered by a battery in the physical robot platform. The battery cannot always provide constant voltage due to its property, so motors are not performed flawlessly like in the simulation. During data collection, we add the noise to the robot joint position and torque in the simulation. The specific values of action noise can be found in Table 1.

Friction noise: The robot foot is built with a soft rubber knob, so it should be a relatively high friction coefficient. We randomize the friction coefficient (shown in Table 5) in the simulation when the robot starts a new epoch during data collection.

Training egocentric visual self-model

Before training the model, we normalized the ground-truth value to the same standard deviation since the output values contain positive and negative numbers and different ranges. This pre-process effectively avoids model bias, which may predict specific output values better than others that do not significantly impact the loss. We use an Adam optimizer with a learning rate of 1e-4 at the beginning and would be scaled by 0.1 if the validation loss does not decrease over twenty epochs. The hyperparameters used for training the visual self-model are listed in Table 6. The loss function can be written as:

$$L = MSE(f_p(I_t, A_t) - S_{t+1}). \tag{4}$$

Reward function and score metrics

Once we have a trained model, we can make the robot perform various tasks by defining the reward function. For example, the robot can perform moving forward by selecting the action that can maximize the forward reward function. We define the following objective function for the robot to complete the basic walking motion in the experiment.

$$Forwardtask : maximizef(\delta x, \delta y) = 2 * \delta y - |\delta x| \tag{5}$$

$$Turnlefttask : maximizef(\delta yaw) = \delta yaw \tag{6}$$

$$Turnrighttask : maximizef(\delta yaw) = -\delta yaw \tag{7}$$

$$Backwardtask : maximizef(\delta x, \delta y) = -2\delta y - |\delta x| \tag{8}$$

In moving forward and backward tasks, the robot obtains scores by moving along the y-axis (in meters) and reduces scores if there is an offset on the y-axis. The robot needs to rotate its body on the yaw axis (in radians) during rotation tasks. The score metrics used to evaluate locomotion performance are shown below. In the real-world task, a, b are set as 10:

$$r_{forward} = a * y - b * |yaw| \tag{9}$$

$$r_{left} = a + b * yaw \tag{10}$$

$$r_{right} = a - b * yaw \tag{11}$$

$$r_{backward} = -a * y - b * |yaw| \tag{12}$$

Data availability

Data, Code, and pre-trained models that support the findings have been deposited in public repositories: https://github.com/H-Y-H-Y-H/Egocentric_VSM.

Code availability

All code to train and evaluate egocentric visual self-models is open-sourced at: https://github.com/H-Y-H-Y-H/Egocentric_VSM.

Received: 15 September 2024; Accepted: 26 April 2025;
Published online: 13 June 2025

References

1. Pearson, K. G. Common principles of motor control in vertebrates and invertebrates. *Annu. Rev. Neurosci.* **16**, 265–297 (1993).
2. Alexander, R. M. *Principles of animal locomotion* (Princeton University Press, 2003).
3. Marigold, D. S. & Patla, A. E. Visual information from the lower visual field is important for walking across multi-surface terrain. *Exp. Brain Res.* **188**, 23–31 (2008).
4. Gibson, J. J. Visually controlled locomotion and visual orientation in animals. *Br. J. Psychol.* **49**, 182–194 (1958).
5. Marigold, D. S. & Patla, A. E. Gaze fixation patterns for negotiating complex ground terrain. *Neuroscience* **144**, 302–313 (2007).
6. Carpentier, J. & Wieber, P.-B. Recent progress in legged robots locomotion control. *Curr. Robot. Rep.* **2**, 231–238 (2021).
7. Torres-Pardo, A. et al. Legged locomotion over irregular terrains: state of the art of human and robot performance. *Bioinspiration Biomim.* **17**, 061002 (2022).
8. Maroger, I., Stasse, O. & Watier, B. Walking human trajectory models and their application to humanoid robot locomotion. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3465–3472 (IEEE, 2020).
9. Peng, X. B. et al. Learning agile robotic locomotion skills by imitating animals. Robotics: Science and Systems (RSS), 12–16 (2020).
10. Gangapurwala, S., Mitchell, A. & Havoutis, I. Guided constrained policy optimization for dynamic quadrupedal robot locomotion. *IEEE Robot. Autom. Lett.* **5**, 3642–3649 (2020).
11. Loomis, J. M. et al. Nonvisual navigation by blind and sighted: assessment of path integration ability. *J. Exp. Psychol.: Gen.* **122**, 73 (1993).

12. Kallie, C. S., Schrater, P. R. & Legge, G. E. Variability in stepping direction explains the veering behavior of blind walkers. *J. Exp. Psychol. Hum. Percept. Perform.* **33**, 183 (2007).
13. Gibson, J. J. *The ecological approach to visual perception: classic edition* (Psychology Press, 2014).
14. Warren, W. H., Kay, B. A., Zosh, W. D., Duchon, A. P. & Sahuc, S. Optic flow is used to control human walking. *Nat. Neurosci.* **4**, 213–216 (2001).
15. Paulus, W., Straube, A. & Brandt, T. Visual stabilization of posture: physiological stimulus characteristics and clinical aspects. *Brain* **107**, 1143–1163 (1984).
16. Wolpert, D. M. & Flanagan, J. R. Motor prediction. *Curr. Biol.* **11**, R729–R732 (2001).
17. Shadmehr, R. & Krakauer, J. W. A computational neuroanatomy for motor control. *Exp. Brain Res.* **185**, 359–381 (2008).
18. Yosinski, J., Clune, J., Bengio, Y. & Lipson, H. How transferable are features in deep neural networks? *Advances in neural information processing systems* **27** (2014).
19. Kwiatkowski, R. *Deep Self-Modeling for Robotic Systems*. Ph.D. thesis, Columbia University (2022).
20. Hu, Y., Kwiatkowski, R., Chen, B., Wyder, P. & Lipson, H. Scalable quasi-static self-modeling for physical legged locomotion (2023).
21. Todorov, E., Erez, T. & Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033 (IEEE, 2012).
22. Coumans, E. & Bai, Y. 2019. pybullet, a Python module for physics simulation for games, robotics and machine learning (2016).
23. Koenig, N. & Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS) (IEEE Cat. No. 04CH37566)* **3**, 2149–2145 (2004).
24. Kumar, A. et al. Adapting rapid motor adaptation for bipedal robots. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (IEEE, 2022).
25. Di Carlo, J., Wensing, P. M., Katz, B., Bledt, G. & Kim, S. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 1–9 (IEEE, 2018).
26. Siekmann, J., Green, K., Warila, J., Fern, A. & Hurst, J. Blind bipedal stair traversal via sim-to-real reinforcement learning. Robotics: Science and Systems. (2021).
27. Rudin, N., Hoeller, D., Reist, P. & Hutter, M. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, 91–100 (PMLR, 2022).
28. Miura, H. & Shimoyama, I. Dynamic walk of a biped. *Int. J. Robot. Res.* **3**, 60–74 (1984).
29. Raibert, M. H. Hopping in legged systems-modeling and simulation for the two-dimensional one-legged case. *IEEE Transactions on Systems, Man, and Cybernetics* 451–463 (1984).
30. Geyer, H., Seyfarth, A. & Blickhan, R. Positive force feedback in bouncing gaits? *Proc. R. Soc. Lond. Ser. B: Biol. Sci.* **270**, 2173–2183 (2003).
31. Bledt, G. et al. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2245–2252 (IEEE, 2018).
32. Hutter, M. et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 38–44 (IEEE, 2016).
33. Haarnoja, T. et al. Learning to walk via deep reinforcement learning. *Robotics: Science and Systems. (2019)*
34. Bongard, J., Zykov, V. & Lipson, H. Resilient machines through continuous self-modeling. *Science* **314**, 1118–1121 (2006).
35. Kwiatkowski, R. & Lipson, H. Task-agnostic self-modeling machines. *Sci. Robot.* **4**, eaau9354 (2019).
36. Chen, B., Kwiatkowski, R., Vondrick, C. & Lipson, H. Fully body visual self-modeling of robot morphologies. *Sci. Robot.* **7**, eabn1944 (2022).
37. Hu, Y., Lin, J. & Lipson, H. Teaching robots to build simulations of themselves. Nature Machine Intelligence. 1-11 (2025).
38. He, M., Zhu, C., Huang, Q., Ren, B. & Liu, J. A review of monocular visual odometry. *Vis. Comput.* **36**, 1053–1065 (2020).
39. Zhou, L., Huang, G., Mao, Y., Wang, S. & Kaess, M. Edplvo: efficient direct point-line visual odometry. In *2022 International Conference on Robotics and Automation (ICRA)*, 7559–7565 (IEEE, 2022).
40. Bao, Y., Yang, Z., Pan, Y. & Huan, R. Semantic-direct visual odometry. *IEEE Robotics and Automation Letters* (2022).
41. Staranowicz, A. & Mariottini, G. L. A survey and comparison of commercial and open-source robotic simulator software. In *Proc. of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, 1–8 (2011).
42. Andrychowicz, O. M. et al. Learning dexterous in-hand manipulation. *Int. J. Robot. Res.* **39**, 3–20 (2020).
43. OpenAI et al. Solving Rubik's Cube with a robot hand. *CoRR*abs/1910.07113 (2019).
44. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proc. of the IEEE conference on computer vision and pattern recognition*, 770–778 (2016).
45. Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. IEEE Transactions on Neural Networks and Learning Systems, vol. 33, no. 12, pp. 6999-7019 (2022)
46. Hochreiter, S. & Schmidhuber, J. Lstm can solve hard long time lag problems. *Advances in neural information processing systems* **9** (1996).
47. Oh, S. L., Ng, E. Y., San Tan, R. & Acharya, U. R. Automated diagnosis of arrhythmia using combination of cnn and LSTM techniques with variable length heart beats. *Comput. Biol. Med.* **102**, 278–287 (2018).
48. Russell, S. & Norvig, P. Artificial intelligence: a modern approach (2002).
49. Fukuoka, Y., Kimura, H. & Cohen, A. H. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *Int. J. Robot. Res.* **22**, 187–202 (2003).
50. Righetti, L. & Ijspeert, A. J. Programmable central pattern generators: an application to biped locomotion control. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*, 1585–1590 (IEEE, 2006).
51. Rasmussen, C. E. Gaussian processes in machine learning. In *Summer school on machine learning*, 63–71 (Springer, 2003).

Acknowledgements

This work was supported in part by the US National Science Foundation (NSF) AI Institute for Dynamical Systems (DynamicsAI.org), grant 2112085.

Author contributions

Y.H., B.C. and H.L. proposed the research. Y.H. developed the robot hardware, algorithm designs, implementations, simulations, and real-world experiments. B.C. and H.L. provided advice on algorithms and experiments. All authors participated in the writing of the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s44182-025-00031-6.

Correspondence and requests for materials should be addressed to Yuhang Hu or Hod Lipson.