Imitation from Videos: Monocular 3D Motion Estimation for Agile Quadruped Locomotion

1st Liu Zhao
Department of Mechanical Engineering
The University of Hong Kong
Hong Kong
zhaol@connect.hku.hk

2nd Yuanhao Chen
Department of Mechanical Engineering
The University of Hong Kong
Hong Kong
jyhcan@connect.hku.hk

3rd Yidan Lu
Department of Mechanical Engineering
The University of Hong Kong
Hong Kong
ydlu@connect.hku.hk

4th Yunhui Liu*
T Stone Robotics Institute
The Chinese University of Hong Kong
Hong Kong
yhliu@mae.cuhk.edu.hk

5th Peng Lu*
Department of Mechanical Engineering
The University of Hong Kong
Hong Kong
lupeng@hku.hk

***Abstract*—Agile and explosive motions are extremely challenging for quadrupedal robots because rewards for aggressive behavior are complex to design and tune. Motion-capture can provide 3D reference motions, but it is costly and requires specialized equipment and annotation. Video-based learning is cheaper yet monocular videos offer only 2D pixels; during fast actions, appearance changes cause joint tracking failures and discontinuous trajectories that hinder locomotion learning. We present a video-to-motion framework. Robust 2D pose estimation and tracking build an undirected skeleton graph and fuse joint observations using a Kalman filter. A Spatial-Temporal Graph Convolutional Network aggregates spatial pose features via graph convolution and temporal dynamics via dilated temporal convolution to reconstruct 3D joint trajectories. The motions are retargeted to the robot joint space and learned with generative imitation learning. Deployed on a quadruped, the robot acquires gallop, tripod, bipedal, and backflip, reaching up to 3.5 m/s while tracking commands. Supplementary video: https://youtu.be/SGf0Nkx8t9A?si=cI098unO6MZ2Kpfv**

***Index Terms*—Quadrupedal robot locomotion, imitation learning, monocular video, 3D pose estimation, graph neural network**

I. Introduction

Deep reinforcement learning has improved quadrupedal locomotion on uneven and soft terrain and even parkour [2]–[7]. However, agility still lags behind animals, largely due to the complexity of learning aggressive motions: RL requires many reward terms (velocity tracking, orientation, joint velocity and acceleration, smoothness, foot clearance, body height, etc.), and tuning their weights is time-consuming; inappropriate tuning can cause learning to fail. Imitation learning from expert demonstrations avoids complex reward design by tracking reference motions [8]–[11]. Animal motions implicitly encode safety and energy efficiency. Reference motions are often obtained via motion capture, and several datasets have been collected to provide 3D joint trajectories [12], [13]. Acquiring such data, however, requires expensive equipment and professional animal trainers; training animals for aggressive motions is also time-consuming, and datasets are limited in size, motion diversity, and environment.

Learning from internet monocular videos reduces the cost of motion capture and offers diverse motions in varied environments. Generating reference motions from videos requires extracting and tracking 3D joint positions and mapping them to the robot joint space. Challenges include: (i) Joint extraction under occlusions, lighting, and especially motion blur and large pixel changes in aggressive motions. (ii) Monocular videos provide only 2D information while 3D joints are needed for locomotion. (iii) Online video frame rates are often lower than robot control frequency, so missing reference commands can harm stability.

We propose robust 2D pose estimation and tracking (skeleton graph plus Kalman filtering) and a Spatio-Temporal Graph Convolution Network (STGNet) to recover 3D joint trajectories from monocular video, then formulate generative imitation learning for real-robot deployment. Our contributions:

(i) A three-step framework for quadrupedal robots to learn aggressive motions from monocular videos.

(ii) Robust 2D pose estimation and joint tracking for aggressive motions.

(iii) STGNet for 3D pose estimation from 2D skeletons.

(iv) Real-robot validation of gallop, tripod, bipedal, and backflip on AlienGo.

II. Method

A. Overview

The framework has three steps: First, robust 2D pose estimation and tracking from video. Second, 3D pose estimation and reference motion generation, Third, generative imitation learning mapping reference motions to the robot and enabling velocity command tracking. Fig. 1 illustrates the pipeline.

Part of the foundational work has been published in npj Robotics [1]. This version, adapted for the workshop "Reinforcement Learning in the Era of Imitation Learning", includes expanded experimental results theoretical details.

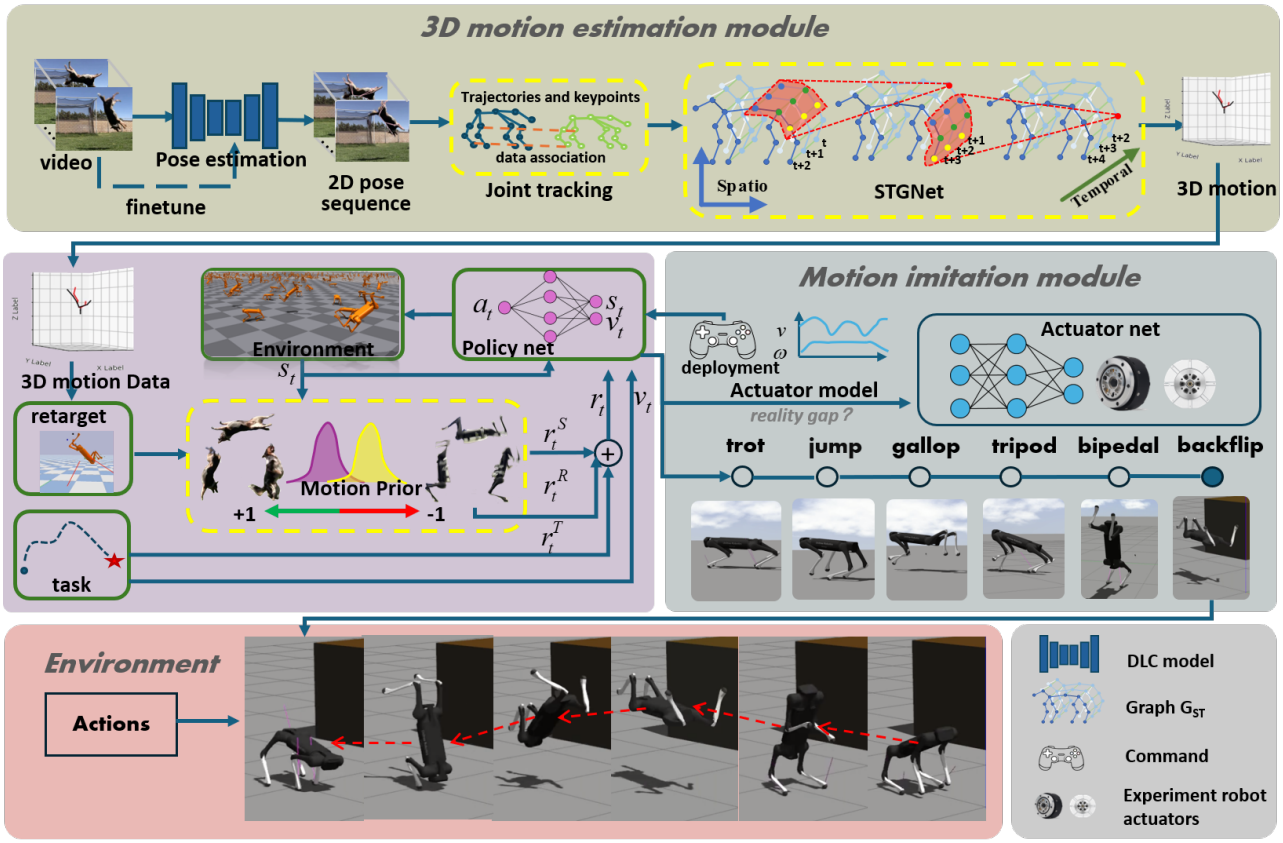

Fig. 1. Description of our learning from video framework. We split the system into two parts: The 3D motion estimation module acquires monocular videos of animals and extracts the spatial-temporal skeleton motion graph in 3D dimensions. The motion imitation module retargets the animal motion data to the robot's joint space and trains the robot to imitate those dynamic motions, enabling it to perform well in the environment.

B. Robust 2D Pose Estimation and Joint Tracking

We fine-tune a 2D pose estimation network (DeepLabCut model) [14], [15] on annotated data to extract joint pixel positions. An undirected skeleton graph is built with joints as nodes and physical connections as edges. In the presence of occlusion and fast motion, raw detections can be noisy or swapped. We use a Kalman filter with a variable-acceleration motion model to predict joint positions; a cost matrix (position distance plus velocity-direction consistency and detection confidence) and the Hungarian algorithm [16] associate predictions with observations frame-by-frame. The filter update yields smooth 2D joint trajectories. To quantify tracking accuracy, we use the 2D MPJPE metric in Eq. (1),

$$L_{pose}^{2d}(f,S) = \frac{1}{N_s}\sum_{i=1}^{N_s}\left\| J_{f;S}^{(i)} - J_{f;S,gt}^{(i)} \right\|_2^2. \tag{1}$$

where N_s is the number of joints in the 2D skeleton S, f denotes the video frame index, $J_{f;S}^{(i)}$ is the estimated 2D pixel position of joint i at frame f, $J_{f;S,gt}^{(i)}$ is the corresponding ground-truth pixel position, and $\|\cdot\|_2$ is the Euclidean norm.

The state-space joint evolution and observation model are given in Eqs. (2)–(6),

$$J_f = F_f J_{f-1} + w_f \tag{2}$$
$$J_f^o = H_f J_f + v_f \tag{3}$$
$$w_f \sim \mathcal{N}(0, Q_f), \quad v_f \sim \mathcal{N}(0, R_f) \tag{4}$$
$$\hat{J}_{f|f-1} = F_f \hat{J}_{f-1|f-1} \tag{5}$$
$$P_{f|f-1} = F_f P_{f-1|f-1} F_f^\top + Q_f \tag{6}$$

where J_f is the (stacked) joint state at frame f, F_f is the state transition matrix, H_f is the observation model (mapping the true state to the measurement space), J_f^o is the observation, w_f and v_f are process and observation noises with zero mean and covariances Q_f and R_f, respectively; $\hat{J}_{f|f-1}$ and $P_{f|f-1}$ are the predicted state estimate and prediction error covariance, while $\hat{J}_{f-1|f-1}$ and $P_{f-1|f-1}$ denote the previous posterior estimates.

The matching cost matrix is defined by Eqs. (7)–(10), where C is the $N \times N$ cost matrix for data association; $c(\cdot,\cdot)$ combines a position-distance term $c_p(i,j)$ and a velocity-direction consistency term $c_d(i,j)$ using scalar weights λ; σ scales the joint-confidence penalty $J_{f,\text{conf}}$ (confidence values on the diagonal, zeros elsewhere). Here $J_f^{o(i)}$ is the observed pixel position of joint i at frame f, $\hat{J}_{f|f-1}^{(j)}$ is the predicted

position for joint j at frame f, and $T_{f-1}^{(j)}$ is the tracked position of joint j from frame $f-1$; $\mathrm{cosd}(\cdot,\cdot)$ denotes the cosine-distance-type measure used to compare the direction consistency between observation and prediction.

$$C = \left[c\left(J_f^{o(i)}, \hat{J}_{f|f-1}^{(j)}\right)\right]_{N\times N} - \sigma\, J_{f,\mathrm{conf}} \tag{7}$$

$$c\left(J_f^{o(i)}, \hat{J}_{f|f-1}^{(j)}\right) = c_p(i,j) + \lambda\, c_d(i,j) \tag{8}$$

$$c_p(i,j) = \left\| J_f^{o(i)} - \hat{J}_{f|f-1}^{(j)} \right\|_2^2 \tag{9}$$

$$c_d(i,j) = \mathrm{cosd}\left(J_f^{o(i)} - T_{f-1}^{(j)},\ \hat{J}_{f|f-1}^{(j)} - T_{f-1}^{(j)}\right) \tag{10}$$

Finally, the Kalman filtering update is performed using Eqs. (11)–(13),

$$K_f = P_{f|f-1}H_f^\top \left(H_f P_{f|f-1} H_f^\top + R_f\right)^{-1} \tag{11}$$

$$\hat{J}_{f|f} = \hat{J}_{f|f-1} + K_f\left(J_f^o - H_f \hat{J}_{f|f-1}\right) \tag{12}$$

$$P_{f|f} = (I - K_f H_f)\, P_{f|f-1} \tag{13}$$

where K_f is the Kalman gain at frame f, $\hat{J}_{f|f}$ is the updated (posterior) joint state estimate, $P_{f|f}$ is the updated posterior error covariance, I is the identity matrix, and $(J_f^o - H_f \hat{J}_{f|f-1})$ is the innovation term.

Outlier refinement via interpolation is applied before tracking. This reduces discontinuities that would harm 3D estimation and imitation.

C. 3D Pose Estimation: STGNet

From the 2D skeleton sequence we build a spatio-temporal graph G_{ST}: nodes are 2D joint positions per frame; edges include intra-frame skeleton links and inter-frame links for the same joint across time. To construct the spatio-temporal skeleton graph and encode STG module feature transforms, we use Eqs. (14)–(18),

$$G_{ST} = (V, E), \qquad V = V_f, \quad E = E_f^s \cup E_f^t, \quad f \in \mathcal{T}. \tag{14}$$

where G_{ST} is the spatio-temporal skeleton graph, V denotes the set of nodes (2D joint states indexed by frame), E is the set of edges composed of intra-frame edges E_f^s (skeleton connectivity at frame f) and temporal edges E_f^t (links for the same joint across time); $\mathcal{T}$ denotes the set of frames in the input video clip.

STGNet consists of stacked Spatio-Temporal Graph (STG) modules. Each STG module:

(i) applies graph convolution to aggregate spatial features within a frame [17];

(ii) applies dilated temporal convolution to aggregate temporal features across frames.

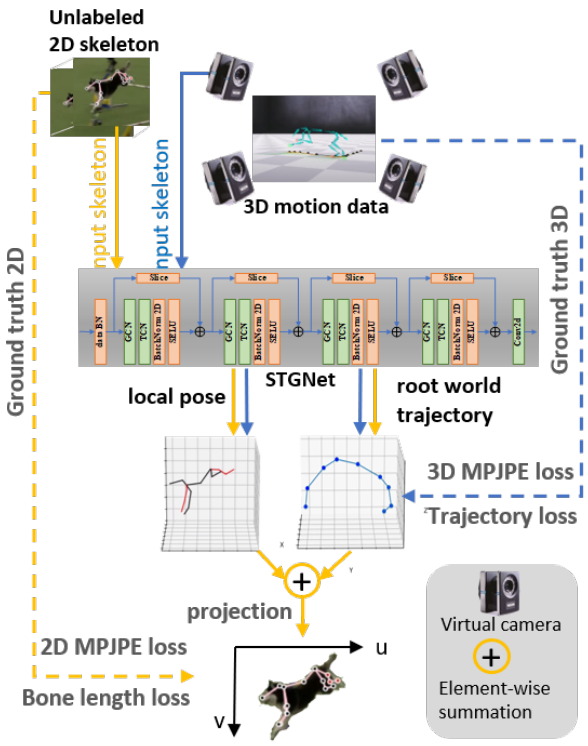

Fig. 2. The STGNet structure, with blue arrow sections representing supervised learning, and yellow arrow connections representing unsupervised learning for 3D pose estimation. Dashed lines indicate the source of supervised data during training.

Residual connections and batch normalization are used [18]. Given input feature map F, the STG module can be expressed via Eqs. (15)–(18),

$$F_{\mathrm{GCN}} = \Lambda^{-1/2}(A+I)\Lambda^{-1/2} F W \tag{15}$$

$$F_{\mathrm{TCN}} = \mathrm{Conv}(F_{\mathrm{GCN}}) \tag{16}$$

$$\Lambda_{ii} = \sum_j (A_{ij} + I_{ij}) \tag{17}$$

$$\mathrm{STG}(F) = \mathrm{SELU}(\mathrm{BN}(F_{\mathrm{TCN}})) + \mathrm{slice}(F) \tag{18}$$

where A is the adjacency matrix of the skeleton graph, I is the identity matrix, and Λ is the degree matrix with diagonal entries $\Lambda_{ii} = \sum_j (A_{ij} + I_{ij})$; W is a trainable weight matrix, $\mathrm{Conv}(\cdot)$ denotes (dilated) temporal convolution over frames, $\mathrm{BN}(\cdot)$ is batch normalization, $\mathrm{SELU}(\cdot)$ is the activation function, and $\mathrm{slice}(F)$ denotes the residual slicing operation to match feature/channel dimensions. Fig. 2 shows 3D motion estimation module.

STGNet performs two regression tasks: local positions of key joints and root trajectory for global position. Global joint positions are derived from local positions and root pose. The network is trained with motion capture data (e.g. AI4Animation, Digdogs) projected to 2D via virtual cameras; 3D data serve as supervision. The projection from 3D motion to 2D virtual camera coordinates follows Eq. (19),

$$P_{f,j}^{2d} = K_j E_j P_f^{3d}. \tag{19}$$

where P_f^{3d} is a 3D joint point (in the world/reference frame) at frame f, $P_{f,j}^{2d}$ is its 2D projected pixel coordinate under the j-th virtual camera, E_j is the camera extrinsic transform, and K_j is the camera intrinsic matrix. And the training objective is optimized using Eqs. (20)–(23),

$$L_{\text{pose}}(f,S) = \frac{1}{N_s}\sum_{i=1}^{N_s}\left\|J_f^{(i)} - J_{f,gt}^{(i)}\right\|_2^2 \tag{20}$$

$$L_{\text{trj}}(f,S) = \frac{1}{k}\left\|J_f(J_r) - J_{f,gt}(J_r)\right\|_2^2 \tag{21}$$

$$L_{\text{bone}}(f,S) = \frac{1}{N_b}\sum_{i=1}^{N_b}\left\|E_f^{(i)} - E_{f,gt}^{(i)}\right\|_2^2 \tag{22}$$

$$L(f,S) = \lambda_p L_{\text{pose}}(f,S) + \lambda_t L_{\text{trj}}(f,S) + \lambda_b L_{\text{bone}}(f,S) \tag{23}$$

where N_s and N_b are the numbers of joints and bones (graph edges) in skeleton S, respectively; $J_f^{(i)}$ and $J_{f,gt}^{(i)}$ are the estimated and ground-truth joint positions of joint i at frame f; J_r is the root-joint variable used to form the global root trajectory term L_{trj} (with normalization factor k); $E_f^{(i)}$ and $E_{f,gt}^{(i)}$ are the estimated and ground-truth bone lengths/quantities for bone i; and $\lambda_p, \lambda_t, \lambda_b$ are scalar weights balancing the three loss components.

D. Motion Imitation and Deployment

Reconstructed 3D motions are retargeted to the robot joint space (hip and foot as corresponding joints; knee solved via inverse kinematics and constraints). Retargeting is performed by solving the constrained optimization in Eq. (24), with morphological scaling in Eq. (25),

$$q_{0:T}^* = \arg\min_{q_{0:T}} \sum_t \Big[\sum_i \|\hat{x}_i(t) - x_i(q_t)\|_2^2 + (\bar{q} - q_t)^\top W(\bar{q} - q_t)\Big]. \tag{24}$$

$$\lambda = \alpha\left(\frac{L_r}{L_b}\right)^\beta. \tag{25}$$

where $q_{0:T}$ denotes the robot joint-angle sequence from time 0 to T, q_t is the joint configuration at time t, $\hat{x}_i(t)$ is the source (animal) keypoint position for keypoint i at time t, $x_i(q_t)$ is the corresponding robot keypoint position computed via robot kinematics given q_t, $\bar{q}$ is the nominal joint configuration, W is a diagonal (regularization) weight matrix that penalizes deviation from $\bar{q}$ while the first term enforces keypoint tracking; λ is the kinematic/morphological scaling factor, L_r and L_b are normalized morphological parameters representing the robot and biological (source) joint lengths, respectively; α is the kinematic fidelity coefficient and β is a torque compensation factor, both treated as dimensionless scalars.

The locomotion problem is formulated as generative imitation learning: the policy learns reference motions while tracking omnidirectional velocity commands. Joint torque limits are penalized to respect hardware. An actuator net bridges simulation and reality for stable real-robot deployment [19]. In particular, the adversarial motion priors reward is written as Eq. (26), the discriminator objective is optimized as Eq. (27), and the style reward takes the form of Eq. (28) [20], [21].

$$r_t = \alpha_T r_t^T + \alpha_S r_t^S + \alpha_R r_t^R. \tag{26}$$

$$\begin{aligned}\arg\min_\psi\ & \mathbb{E}_{(s_t,s_{t+1})\sim M}\left[\left(D_\psi(s_t,s_{t+1})-1\right)^2\right] \\ & + \mathbb{E}_{(s_t,s_{t+1})\sim \pi_L}\left[\left(D_\psi(s_t,s_{t+1})+1\right)^2\right] \\ & + \frac{\beta_{gp}}{2}\mathbb{E}_{(s_t,s_{t+1})\sim M}\left[\left\|\nabla_{(s_t,s_{t+1})} D_\psi(s_t,s_{t+1})\right\|_2^2\right].\end{aligned} \tag{27}$$

$$r_t^S(s_t,s_{t+1}) = \max\left(0,\ 1 - 0.25\left(D_\psi(s_t,s_{t+1})-1\right)^2\right). \tag{28}$$

where r_t^T is the task reward, r_t^S is the style (imitation) reward, and r_t^R is the regularization reward; $\alpha_T, \alpha_S, \alpha_R$ are scalar weights controlling the contribution of each component; ψ denotes the discriminator parameters, D_ψ is the discriminator function, M is the expert/reference motion dataset, π_L is the current locomotion policy distribution, and β_{gp} is the gradient-penalty coefficient; the gradient penalty term is computed with respect to the state-transition inputs (s_t, s_{t+1}) to stabilize discriminator training; $r_t^S(s_t, s_{t+1})$ is the style reward computed from the discriminator score on the state transition (s_t, s_{t+1}).

Four skill types are demonstrated: agile quadruped (gallop), fault-tolerant (tripod), challenging (bipedal), and explosive (backflip).

III. Results

A. 2D Pose and Tracking

Refined joint tracking reduces mean per-joint position error (MPJPE) compared to using raw 2D estimates; using both position and direction in the cost matrix improves results over position-only. Reconstructed joint trajectories are smooth and continuous, enabling reliable 3D estimation and imitation.

B. 3D Pose Estimation

STGNet is compared with VideoPose3D and related methods [22]–[25]; it achieves lower training loss and lower MPJPE on validation data. Ablations on the number and width of STG modules show that four stacked middle-size STG modules give best 3D pose accuracy. Qualitative results on six videos (bipedal, backflip, trot, gallop, hurdle, tripod) show accurate 3D reconstruction and retargeting even when 2D skeletons overlap.

To verify the effectiveness of the designed spatio-temporal graph convolution module, we adjust the number and size of STG modules and study their optimal stacking quantity.According to Table I, the 3D pose MPJPE performance of STGNet is best when the number of STG module stacks is four with a receptive field of 243 frames.

To validate the effectiveness of our designed 3D pose estimation algorithm, we compare it with the VideoPose3D [22] algorithm to evaluate the accuracy of the model in 3D skeleton estimation.

We observe that our model achieves a lower loss in 3D pose during the training process and a lower MPJPE on the

TABLE I
THE ABLATION EXPERIMENT OF THE STG MODULE NUMBER

N_{STG}	Params	Loss	MPJPE(mm)	MPJVE
1	14793693	44.64	67.522	8.400
2	15188445	31.43	36.214	4.656
3	15582429	20.30	26.253	3.566
4	15587055	14.41	24.365	3.148

TABLE II
ABLATION EXPERIMENT OF THE GCN MODULE EFFICIENCY

Algorithm	Params	$\mathcal{L}_{pose}$	MPJPE	MPJVE
Ours	15585516	8.891	23.948	3.247
[22]	16952371	17.05	22.275	4.395

validation dataset, demonstrating that STGNet is capable of extracting features more effectively from G_{ST}, leading to more accurate predictions in 3D pose estimation.

We qualitatively analyze the results of 3D skeleton reconstruction in Fig. 3. Keyframes from six videos are displayed at the top, with their corresponding 3D estimation results shown in the middle. The bottom section presents the retargeted result on the robot in PyBullet, where the reference 3D pose of the robot motion is depicted as points.

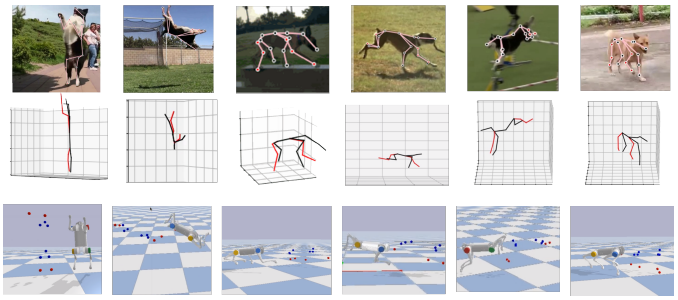

Fig. 3. Qualitative results for six videos. From left to right are bipedal, backflip, trot, gallop, hurdle, and tripod motions. **Top**: video frames with 2D pose overlay. **Middle**: 3D reconstruction. **Bottom**: Retarget result on robot.

C. Real-Robot Locomotion on AlienGo

AlienGo (12 joints, torque limits 45–55 N·m, mass 23.24 kg) is used for deployment.

- Gallop: mean speed 2.5 m/s, max 3.45 m/s; max height 0.6 m, stepping frequency 1.95 Hz.
- Tripod: stable fault-tolerant locomotion, max velocity 1.495 m/s, stepping frequency 2.62 Hz.
- Bipedal: speed 0.45 m/s, 3.3 Hz; transitions between bipedal and quadrupedal are demonstrated.
- Backflip: consecutive backflips at 1.15 Hz.

Fig. 4 and Fig. 5 shows simulation ans real-world gallop, tripod, bipedal, and backflip respectively.

Gait analysis shows that the robot's motion phases and contact sequences match the source video patterns. Joint

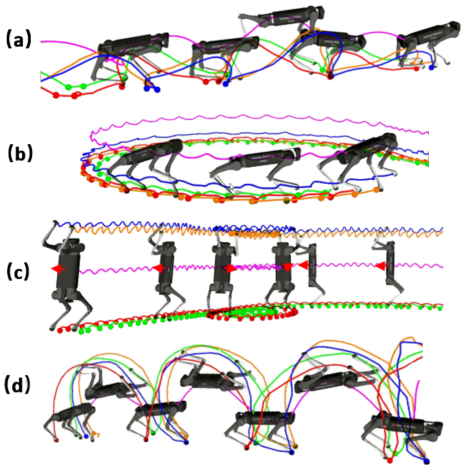

Fig. 4. AlienGo performing (a) gallop, (b) tripod, (c) bipedal, (d) backflip in simulation.

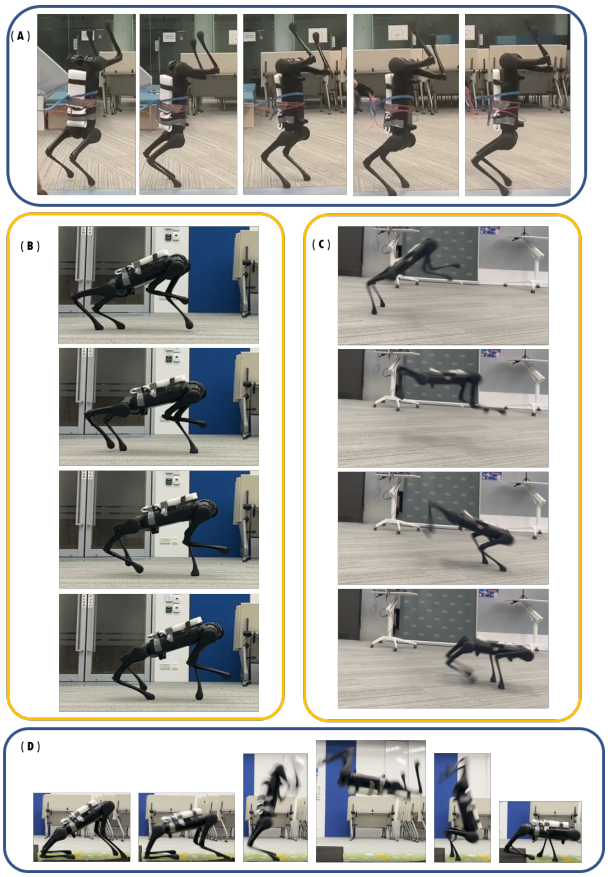

Fig. 5. Real-world experiments on AlienGo robot. (a) bipedal (b) tripod (c) gallop and (d) backflip locomotion that learns from monocular videos. The robot performs challenge locomotion stably.

torques remain within safe limits during aggressive motions. Gait analysis of backflip can be found in Fig. 6.

IV. DISCUSSION AND CONCLUSION

We proposed a three-step framework that enables quadrupedal robots to learn aggressive motions from monoc-

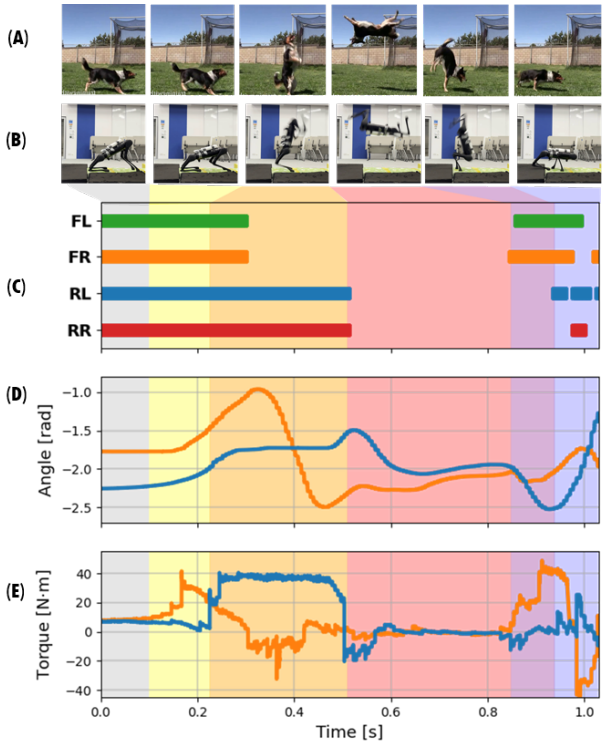

Fig. 6. Real Legged Robot Motion Data Analysis: The figure includes source video clips (A) and real robot motion imitation performance (B), gait analysis (C), joint angle analysis (D), and temporal joint torque analysis (E) of backflip motion.

ular videos, reducing the need for motion capture. Robust 2D pose estimation and Kalman-filter-based joint tracking yield smooth trajectories even under motion blur and fast motion. STGNet recovers 3D joint trajectories from 2D skeletons via spatio-temporal graph convolution and dilated temporal convolution. Generative imitation learning with retargeting and torque limits allows the robot to learn reference motions while tracking velocity commands. Domain randomization is adopted to reduce the sim-to-real gap. Validated on AlienGo, the framework successfully learns gallop, tripod, bipedal, and backflip from videos without object shape, camera pose, or intrinsic parameters. Limitations include cross-species morphological adaptation (joint-point selection must match target morphology) and possible degradation under extreme camera angles or erroneous 2D input. The approach provides a general pipeline for learning aggressive locomotion from monocular video and can support applications in legged robotics.

References

[1] L. Zhao et al., "Learning aggressive animal locomotion skills for quadrupedal robots solely from monocular videos," *npj Robotics*, vol. 3, no. 1, p. 32, 2025.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, Oct. 2020. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.abc5986

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv:2107.04034*, 2021.

[4] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[5] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5078–5084.

[6] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "Dtc: Deep tracking control," *Science Robotics*, vol. 9, no. 86, p. eadh5401, 2024. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.adh5401

[7] Z. Zhuang et al., "Robot parkour learning," in *Conference on Robot Learning (CoRL)*, 2023.

[8] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philos Trans R Soc Lond B Biol Sci.*, vol. 358, no. 1431, pp. 537–547, 2003.

[9] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics and Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010.

[10] T. D. Sergey Levine, Chelsea Finn and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, pp. 1–40, 2016.

[11] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," 2020.

[12] H. Zhang, S. Starke, T. Komura, and J. Saito, "Mode-adaptive neural networks for quadruped motion control," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–11, 2018.

[13] M. Shooter, C. Malleson, and A. Hilton, "Digidogs: Single-view 3d pose estimation of dogs using synthetic training data," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, January 2024, pp. 80–89.

[14] A. Mathis et al., "Deeplabcut: markerless pose estimation of user-defined body parts with deep learning," *Nature Neuroscience*, 2018. [Online]. Available: https://www.nature.com/articles/s41593-018-0209-y

[15] S. Ye et al., "Superanimal pretrained pose estimation models for behavioral analysis," 2023.

[16] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017. [Online]. Available: https://arxiv.org/abs/1609.02907

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[19] J. Hwangbo et al., "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[20] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[21] A. Escontrela et al., "Adversarial motion priors make good substitutes for complex reward functions," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 25–32.

[22] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3d human pose estimation in video with temporal convolutions and semi-supervised training," 2019. [Online]. Available: https://arxiv.org/abs/1811.11742

[23] A. Gosztolai et al., "LiftPose3D, a deep learning-based approach for transforming two-dimensional to three-dimensional poses in laboratory animals," *Nature Methods*, vol. 18, no. 8, pp. 975–981, 8 2021. [Online]. Available: https://doi.org/10.1038/s41592-021-01226-z

[24] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. N. Metaxas, "Semantic graph convolutional networks for 3d human pose regression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3425–3435.

[25] B. X. Yu, Z. Zhang, Y. Liu, S.-h. Zhong, Y. Liu, and C. W. Chen, "Glagcn: Global-local adaptive graph convolutional network for 3d human pose estimation from monocular video," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2023, pp. 8818–8829.