Trajectory Tracking and Imitation of Robotic Quadrupeds Using DeepLabCut

Martin Davies
Faculty of Technology
University of Sunderland
Sunderland, United Kingdom
0009-0003-1771-1618

John Murray
Faculty of Technology
University of Sunderland
Sunderland, United Kingdom
0000-0002-0384-9531

Umar Manzoor
Faculty of Technology
University of Sunderland
Sunderland, United Kingdom
0000-0001-7602-1914

***Abstract*—Quadrupedal robotics is an ever growing field with a wide range of applications. However, developing controllers for new behaviours can be challenging due to the complex nature of these robots. Imitation learning algorithms can help overcome some of these challenges, with robots learning from biological counterparts through motion capture data. Robots could also potentially use these techniques for copying behaviours from other robots/animals of similar morphology. Acquiring the required motion capture data from animals and remote locations can be difficult due to the bulky expensive equipment required. However, the use of pose estimation toolboxes such as DeepLabCut could negate this issue. This paper covers the methodology and results from initial proof of concept experiments for two key areas. Firstly, testing the feasibility of DeepLabCut in the use of tracking robotic quadrupeds. Secondly, if the data produced can be used to generate trajectories for deployment on a target robot. This will help to establish if the use of pose estimation toolboxes could potentially be useful in future imitation learning experiments.**

***Index Terms*—pose estimation, quadrupedal robotics, imitation learning.**

I. Introduction

In recent times, robotic quadrupedal platforms are being deployed for an increasingly wider range of tasks, mainly due to their ability to traverse a variety of environments. They are currently deployed in areas such as inspection of gas and oil rigs and nuclear power stations [1][2], and to enter potentially dangerous rescue situations [3]. However, as versatile as these platforms are, programming them to perform new tasks and behaviours can be challenging [4]. Having the ability to learn from and imitate their biological counterparts, or other robots with similar morphology could be advantageous in overcoming some of these hurdles. Research into imitation learning techniques for robotics is becoming more prevalent. However, acquiring the motion capture data for use with these algorithms can be tricky depending on subject and location [5]. The development of markerless pose estimation toolboxes increase the feasibility of tracking free moving animals on location with their ability to use novel videos taken from contactless cameras [6]. Toolboxes such as SLEAP, DeepLabCut (DLC), and DeepPoseKit have had excellent results in tracking animal behaviour without the need of bulky equipment or restriction of environment [7][8][9]. The work conducted using these toolboxes has tracked a wide variety of animals including horses, cows, dogs, and mice [5], all of which would make good candidates for the use in imitation learning for quadrupedal robotics.

II. Background

A. Markerless Pose Estimation

Pose estimation toolboxes such as DLC are predominantly used for tracking and analysis animal movements and behaviour. OkeyDogg3D, a mobile application developed using DLC by Yu and Rim to track and recognise stress related behaviours in dogs [10]. Lecomte et al. used DLC for the gait analysis of cats, allowing them to study areas such as stride and step length [11]. WiKA is a sign language recognition tool proposed by Osorio et al. It uses DLC to recognise joint positions of the hands and interpret their sign language counterpart [12]. The work conducted in these areas helps inform the use of pose estimation toolboxes for use in imitation of animals. However, little work has been conducted with these toolboxes being used to track biologically inspired robots. Although, Youseff et al. has recently used DLC in conjunction with reinforcement learning to track robotic fish, enabling the robots to reach a specific target [13].

B. Imitation Learning

The purpose of the experiments discussed in this paper is to determine whether using pose estimation toolboxes show potential for future use in this area. By regarding the techniques currently used in this field can help determine how the use of pose estimation toolboxes could be beneficial.

The work conducted by Peng et al. proposed an imitation learning framework where motion capture data is taken from animals and deployed onto a Laikago quadrupedal robot using reinforcement learning techniques, and using simulation to retarget the motion capture data [4]. Grandia et al. presented the Differentiable Optimal Control (DOC) framework, which takes data from motion capture or animation and then retargets it onto simulated and real robotic quadrupeds of various morphology. As DOC is applied to model predictive control (MPC), there is no loss of dynamic stability [14]. Zhang et al. used motion capture data from a Labrador performing dynamic movements such as running and jumping, and transferred it to a robotic quadruped. These actions were also triggered from human interaction much like a real pet [15]. Li et al. proposed

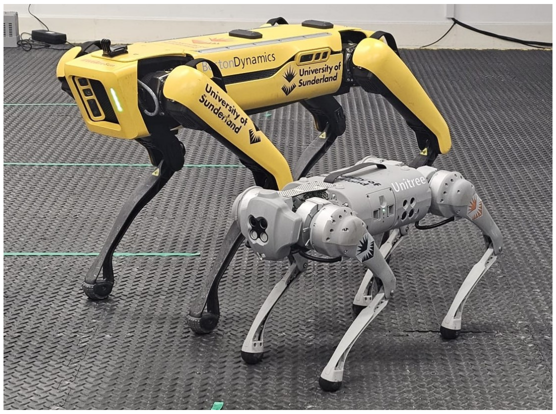


Figure 1: Spot by Boston Dynamics (Bernard) and the Unitree Go1 Edu (Rocky) stood side by side to demonstrate differences in morphology.

an adversarial imitation learning algorithm named WASABI in conjunction with partial hand held demonstrations of dynamic movements. These movements (including a back flip) were then deployed on their Solo8 quadrupedal robot [16]. The area of imitation learning in legged robotics is not limited to quadrupedal platforms. Recently Cheng et al. used motion capture data and deployed it onto a Unitree H1 bipedal robot, allowing the robot to imitate dynamic movements including dancing and giving high fives [17].

III. Methodology

A. Robots

The experiments covered in this paper make use of two robotic quadrupeds. The first being a Boston Dynamics' Spot, which we have named Bernard. The second is a Go1 EDU built by Unitree, which we have called Rocky (Fig 1). Both of these robots have a 12-Degree of Freedom (DoF) design, with 3-DoF present at each leg. These robots have a similar morphology but they have significant size, stability, and gait differences. This allows the conducted experiments to test across different platforms rather than have a single robot imitate itself.

Bernard is used as the demonstrator for tracking with DLC. This platform is objectively far more stable that its Go1 counterpart. This allows for better control when creating and recording the target trajectory. The robot is controlled manually using its controller to emulate discrepancies that might be found while in the field.

Rocky is used as the target robot for the deployment of the recorded trajectory. It was controlled via PC with the use of ROS2 Humble, and makes use of the ROS2 packages provided by Unitree. This includes their UDP connection protocol, and ROS legged messages, which can be found on their GitHub[1].

B. Method of Recording Video Footage

Training and test footage for DLC is collected via the use of a single GoPro 9. This was mounted to a tripod in a single fixed location at a height of 1.49m. Video footage of Bernard is captured at a frame rate of 23.98 frames per second (fps).

[1]https://github.com/unitreerobotics/unitree_ros2

C. DeepLabCut Setup

DLC's markerless pose estimation toolbox uses manually labelled frames extracted from videos to train a model capable of locating points of interest in novel footage of animals [18][8]. To prevent any conflict with ROS2 dependencies, DLC is deployed in a Docker container with shared access to files that were required by both ROS2 and DLC. All documentation on using DLC can be found on their GitHub[2].

D. Training DLC

Using the method discussed in III-B, a total of eight videos of Bernard are created for the use of training the DLC model. These videos range in length from 33 to 61 seconds ($\approx$ 791 to 1458 frames), and contain a wide variety of Bernard's motions. This variety of movement helps increase robustness of the trained model. Recorded walking patterns include straight line, circular, and figure of eight trajectories. Other movements such as strafing and turning in place are also recorded, along with numerous static poses created by changing Bernard's pitch, roll and yaw angles. Although DLC documentation recommends training with footage recorded in different settings and lighting etc. For the purpose of these initial proof of concept experiments, it was deemed that recordings set in a single location would be sufficient. From each of the eight training videos, forty individual frames are extracted automatically using the DLC's in-built k-means clustering frame extraction algorithm. Points of interest are then manually located on each of the extracted frames. As this preliminary work focused on only recreating a generalised trajectory on the target robot, it was decided that only two key points of interest would be significant for marking at this stage. In each extracted frame a marker is placed at the "head" of Bernard, and one at the

[2]https://github.com/DeepLabCut/

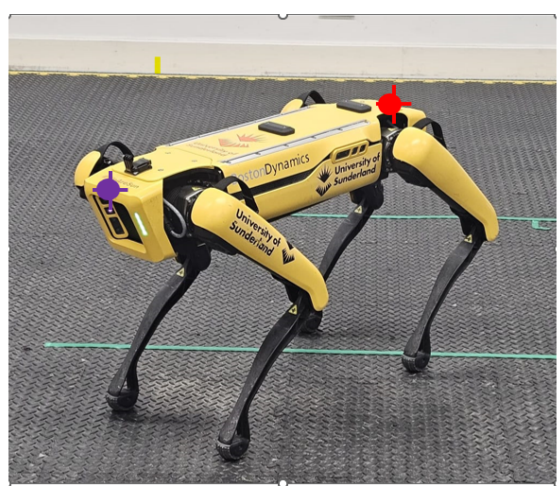


Figure 2: Example evaluation image provided by DeepLabCut.

Table I: DLC Model Combined Evaluation Results

Training Iterations	Train Error (px)	Test Error (px)	p-cuttoff Used
50000	1.51	2.46	0.8

”tail”. Once all frames were manually labelled, the DLC model can be trained.

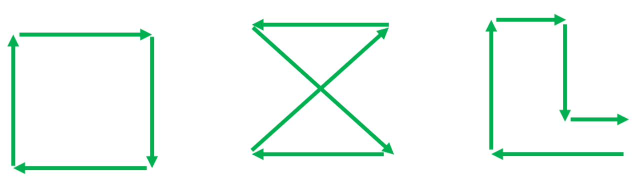
The DLC model is trained using the default option of the ResNet-50 convolutional neural network. Training takes place over 50000 epochs, with the prediction cutoff (p-cutoff) being raised from the default 0.6 to a value of 0.8. The model is then evaluated by visually comparing the manually located points of interest (ground truth) against their predicted location for each training frame. The manually located points are marked in each frame with ”+” and the predictions are marked with a ”•”. An example evaluation frame can be seen in Fig. 2, with a small discrepancy present between labeled and predicted locations. DLC also tests itself using reserved none labelled frames from the same set of videos, and provides error results for both training and testing. These values can be seen in Table I.

E. Testing the DLC Model

Another five novel videos of Bernard are recorded for the purpose of testing the trained model and obtaining data for imitation. As before, these are created using the same method as discussed in section III-B. These five novel videos each contain footage of Bernard performing a unique trajectory. The footage of Bernard includes walking in ”box”, ”egg timer”, and ”block L” patterns as shown in Fig. 3. The final two videos contain footage of Bernard walking in a straight line with pauses, and turning in place. These new recordings are then analysed with the trained model to evaluate how well it performs when presented with new footage. After these videos are analysed by the model using the DLC toolbox, the results can be plotted, and labelled videos created. The data extrapolated from these videos can then be tested by being transferred to Rocky, and an attempt made to imitate the generalised trajectory.

F. Calculating Trajectory

As the purpose of these initial experiments are to explore whether data provided from DLC can be deployed in a meaningful way to another robot, the decision was made that attempting to copy general trajectory over time would be applicable at this stage. The data extrapolated from the novel videos provide the x/y co-ordinates in pixels of both points of interest tracked by the model. These co-ordinates and their relationship to each other are used to attempt to determine four basic ”behaviours”. These are walking forward, rotating left, rotating right, and remaining stationary.



(a) ”Box” Pattern (b) ”Egg Timer” Pattern (c) ”Block Capital L” Pattern

Figure 3: The three trajectory patterns

To determine whether the Bernard is turning right or left, the robot’s angle of orientation is calculated for each video frame. This is achieved by determining the gradient from the ”head” to the ”tail” points relative to the x-axis within the frame and using it to calculate the corresponding angle.

The gradient is calculated by:

$$m = \frac{(y2 - y1)}{(x2 - x1)} \tag{1}$$

Where m is the gradient, $x1$ and $y1$ represent the x and y co-ordinates of the head in pixels, and $x2$ and $y2$ represent the x and y co-ordinates of the tail in pixels.

The angle θ is then calculated by:

$$\theta = \tan^{-1}(m) \tag{2}$$

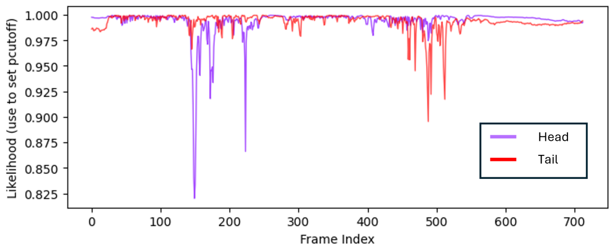
If the angle between frames increases or decreases significantly, then it can be determined that at that point in the video that Bernard is turning. As the experiments only require understanding of whether the robot is turning rather than how far it has, the exact angle value is arbitrary. If no significant change in angle is present, but significant relative change of both ”head” and ”tail” co-ordinates is apparent, then it can be determined that the robot is walking in a straight line.

If no significant change is present in angle or co-ordinates, then it can determined that the robot remains stationary.

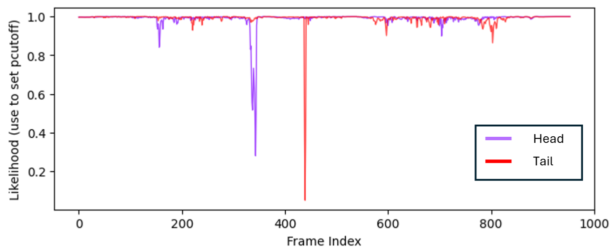
The training error shown in Table I can cause marked points to show slight movement between frames while Bernard remains stationary. If the frame step is too small difficulties can arise distinguishing between Bernard movement and the aforementioned error. To compensate for this comparison was completed on every sixth frame ($\approx$ 0.25 seconds). Once a behaviour at a frame is determined, the length of time it remains is calculated from the number of consecutive frames showing the same behaviour, and multiplying by the fps.

G. Deploying Trajectory on Rocky

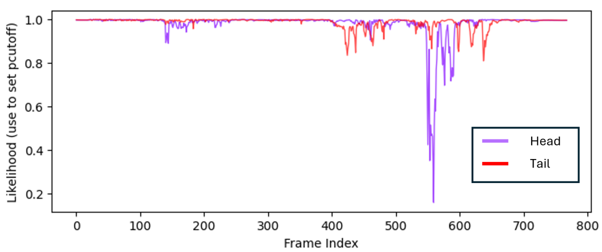
A ROS node is written to perform the determination of each behaviour from the extracted DLC data and the time period it lasts. A set of tuning parameters are also created in the node, including options to change variables such as time step length, walk speed, yaw speeds, and filters to eliminate any inaccuracy present in the data. The node also depends on packages written and provided by Unitree for working with ROS2, including their legged messages and high level UDP protocol. The node relates a behaviour to the Go1’s pre-programmed high level modes, and selects the correct mode and speed parameters. These are then transmitted for deployment onto Rocky.



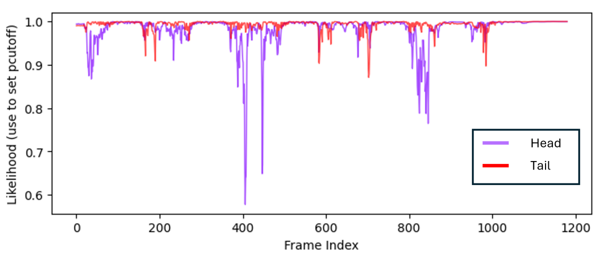
(a) DLC Box likelihood Plot



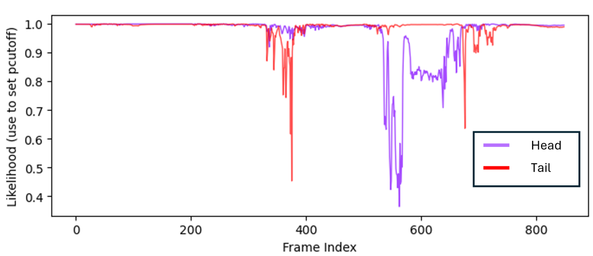
(b) DLC Egg Timer Likelihood Plot



(c) DLC L Likelihood Plot



(d) DLC Spin Likelihood Plot



(e) DLC Walk Stop Likelihood Plot

Figure 4: Likelihood plots for points of interest at each frame

IV. Results

A. The DLC Model

The DLC toolbox is also used to create labelled videos from the analysed test footage. Graphs containing the likelihood predictions at each frame are also created, and can be seen in Fig. 4. These plots show the accuracy of both points of interest at each frame.

Table II: Tuning parameters for each imitated trajectory

Trajectory	walk buffer	turn buffer	right yaw	left yaw	vel
Initial	1.0	2.2	-0.7	0.65	0.3
”L”	1.0	2.2	-0.7	0.65	0.3
”Box”	1.0	2.7	-0.8	0.65	0.3
”Egg Timer”	1.0	2.2	-0.7	0.7	0.3
”Spin”	2.8	2.2	-0.7	0.65	0.3
”Walk Stop”	1.0	2.2	-0.7	0.5	0.3

B. Trajectory Replication

Each trajectory is then tested on Rocky using the method discussed in section III-G. All trajectories initially use the same set of parameters that are chosen from tuning the ”Block L” trajectory data. Each trajectory is then deployed to Rocky, and the results recorded. Individual tuning parameters are then established for each trajectory to increase performance accuracy, and a second set of video recordings created. Exact tuning parameters for each trajectory can be found in Table II. Where walk and turn buffers are values in pixels to eliminate jitter. The right yaw, left yaw and vel values represent velocities for turning left, right and moving forward respectively. These velocity values use implementation from the Unitree ROS2 high level control examples and range in value from -1 to 1. Recordings of the tuned trajectories are also run through the DLC model to produce x/y co-ordinate data for comparison.

Both demonstrator and target robot DLC data is used to create x/y co-ordinate plots of each trajectory. Fig. 5 shows the path of each robot over time, and is calculated using an average of ”head” and ”tail” positions at each frame. By comparing the path of both robots for each trajectory, it can be determined how well the target robot performed against the demonstrator.

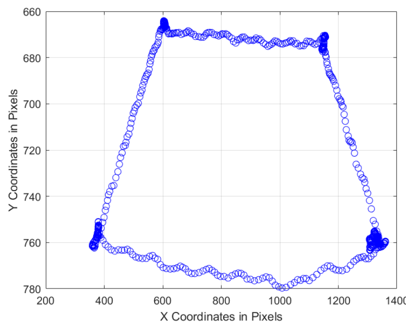
Recordings of all labelled DLC videos, and deployed trajectories can be found online[3].

V. Discussion

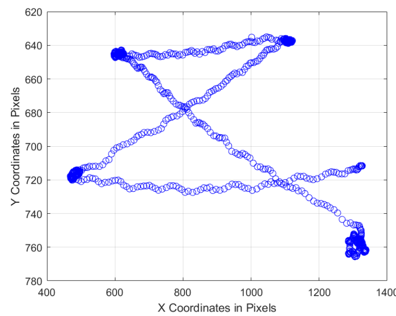
A. DLC Model Performance

From analysis of labelled videos and the produced graphical plots, it can be argued that at this early stage of experimentation the use of DLC for tracking robotic quadrupeds shows promise. The labelled videos show both the ”head” and ”tail” being tracked objectively well over the course of an entire video. The co-ordinate plots of the demonstrator robot in Fig.5 also support these findings, each showing the desired trajectory albeit a slight distortion pertained to the angle of recording, and the ”fish bowl” effect of the camera lens. This gives the appearance in all trajectories of a curvature in the x-axis and tapering in the y-axis. The likelihood plots in Fig.4 also show a good overall performance of the model. However, some frames show significant loss of accuracy for both ”head” and ”tail” markers. This can also be observed when watching the marked

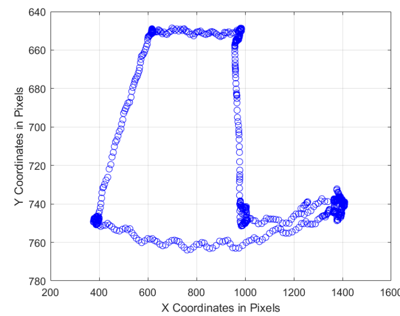
[3]https://tinyurl.com/9utd2m6e



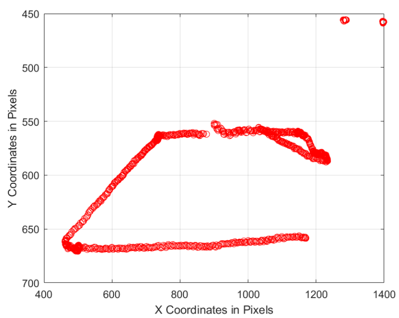
(a) "Box" Coordinate Plot of Demonstrator Robot



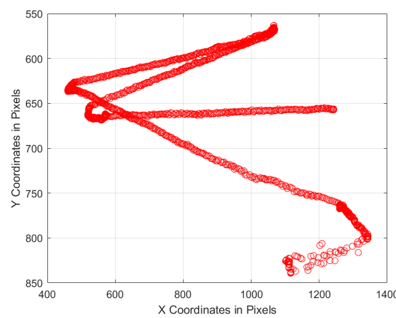
(b) "Egg Timer" Coordinate Plot of Demonstrator Robot



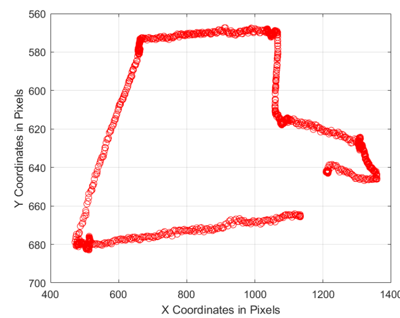
(c) "Block L" Coordinate Plot of Demonstrator Robot



(d) "Box" Coordinate Plot of Target Robot



(e) "Egg Timer" Coordinate Plot of Target Robot



(f) "Block L" Coordinate Plot of Target Robot

Figure 5: x/y coordinate plot of each novel video and target robot trajectories

videos carefully, where it can be seen that in certain frames the markers appear in the wrong location. Further experimentation and analysis is required to ascertain the exact reasons for the DLC model performing inadequately for these specific frames. However, as these experiments use a relatively small data set in a single environment, creating a more robust model using a larger data set could possibly improve results.

B. Trajectory Replication Performance

Although the methods used to replicate general trajectory from demonstration video to target robot were rudimentary, and lacking in the use of an imitation learning algorithm. The results displayed in the video evidence and Fig.5 show potential for the use of a pose estimation toolbox in future imitation learning research. By comparing co-ordinate plots of the demonstrator against the target robot, it can be seen that Rocky recreated the desired path, albeit with some discrepancy. Incorrect turn speeds, distances of travel, and pauses were present in all videos. For example, it was noted that before before Bernard comes to a halt, it marches briefly in place. This then created unwanted forward movement in the target robot. It can be argued that at this stage of experimentation, artifacts such as these are to be expected. Many of these unwanted artifacts, and the need to individually tune each trajectory appears to stem from the limited functionality applied to Rocky. This can hopefully be eliminated in future work by using a more robust DLC model, implementing of a form of imitation learning algorithm, and filtering any outlier data. It should also be noted that some initial errors originated from Rocky itself, as it began to drift when turning. An attempt to compensate for this was made by implementing a slight forward velocity as the robot turned.

C. Future Work

The creation of a more robust DLC model will include creating a larger video data set. This will contain footage of various robotic quadrupedal platforms performing a wide array of movements in a variety of environments. The number of points of interest will also be increased to include each of the robots joints rather than the "head" and "tail" only. This will allow future experimentation to also test low level control of the target robot. Using the data produced by DLC in conjunction with a form of imitation learning algorithm will also establish whether this provides any improvement to accuracy of trajectory imitation.

VI. Conclusion

The purpose of the experiments covered in this paper are to determine if DLC shows potential for use in tracking robotic quadrupeds, and if the data produced can be transferred to a target robot for trajectory imitation. It can be argued that the results presented show that DLC has promise in both of these areas. The DLC model successfully tracks the demonstrator robot, and the data deployed to the target robot results in the trajectory being replicated. However, both the DLC model and the imitation results can be significantly improved. The likelihood graphs in Fig. 4 show the model losing accuracy at certain frames, and although Fig. 5 shows general trajectory being imitated successfully, discrepancies are present in each case. It has been determined that the cause of some these errors requires further investigation. However, loss of model accuracy, outlier data, limited behaviour functionality, and hardware issues are all factors that should be addressed to improve results. Once these issues are resolved, the use of DLC can be tested in conjunction with an imitation learning algorithm. If successful it will allow imitation learning methods to be deployed in a wider setting, for imitation of both animals and robots.

References

[1] Scarlett Evans, *Hannover Messe 2022: Spot the Robot Dog, ANYmal Inspecting Oil and Gas Rigs | IoT World Today*, en, Jan. 2022. [Online]. Available: https://www.iotworldtoday.com/robotics/hannover-messe-2022-spot-the-robot-dog-anymal-inspecting-oil-and-gas-rigs (visited on 21/04/2024).

[2] Jim Scott, 'Sellafield nuclear waste clear up helped by robotic dog', en-GB, *BBC News*, Nov. 2023. [Online]. Available: https://www.bbc.com/news/uk-england-cumbria-67390398 (visited on 21/04/2024).

[3] J. Capellini, *Mayor Adams: Robotic dog, drone used to assess Lower Manhattan parking garage collapse - CBS New York*, en-US, Apr. 2023. [Online]. Available: https://www.cbsnews.com/newyork/news/mayor-adams-robotic-dog-drone-used-to-assess-lower-manhattan-parking-garage-collapse/ (visited on 21/04/2024).

[4] X. Bin Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan and S. Levine, 'Learning Agile Robotic Locomotion Skills by Imitating Animals', in *Robotics: Science and Systems XVI*, Robotics: Science and Systems Foundation, Jul. 2020, ISBN: 978-0-9923747-6-1. DOI: 10.15607/RSS.2020.XVI.064.

[5] L. Jiang, C. Lee, D. Teotia and S. Ostadabbas, 'Animal pose estimation: A closer look at the state-of-the-art, existing gaps and opportunities', en, *Computer Vision and Image Understanding*, vol. 222, p. 103 483, Sep. 2022, ISSN: 10773142. DOI: 10.1016/j.cviu.2022.103483.

[6] G. Tang, Y. Han, Q. Liu and P. Wei, *Anti-drift pose tracker (ADPT): A transformer-based network for robust animal pose estimation cross-species*, en, Feb. 2024. DOI: 10.1101/2024.02.06.579164.

[7] T. D. Pereira, N. Tabris, A. Matsliah *et al.*, 'SLEAP: A deep learning system for multi-animal pose tracking', en, *Nature Methods*, vol. 19, no. 4, pp. 486–495, Apr. 2022, ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-022-01426-1.

[8] A. Mathis, P. Mamidanna, K. M. Cury *et al.*, 'DeepLabCut: Markerless pose estimation of user-defined body parts with deep learning', en, *Nature Neuroscience*, vol. 21, no. 9, pp. 1281–1289, Sep. 2018, ISSN: 1097-6256, 1546-1726. DOI: 10.1038/s41593-018-0209-y.

[9] J. M. Graving, D. Chae, H. Naik *et al.*, 'DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning', *eLife*, vol. 8, I. T. Baldwin, J. W. Shaevitz, J. W. Shaevitz and G. Stephens, Eds., e47994, Oct. 2019, Publisher: eLife Sciences Publications, Ltd, ISSN: 2050-084X. DOI: 10.7554/eLife.47994.

[10] R. Yu and Y. Choi, 'OkeyDoggy3D: A Mobile Application for Recognizing Stress-Related Behaviors in Companion Dogs Based on Three-Dimensional Pose Estimation through Deep Learning', en, *Applied Sciences*, vol. 12, no. 16, p. 8057, Aug. 2022, ISSN: 2076-3417. DOI: 10.3390/app12168057.

[11] C. G. Lecomte, J. Audet, J. Harnie and A. Frigon, 'A Validation of Supervised Deep Learning for Gait Analysis in the Cat', en, *Frontiers in Neuroinformatics*, vol. 15, p. 712 623, Aug. 2021, ISSN: 1662-5196. DOI: 10.3389/fninf.2021.712623.

[12] T. M. Osorio, R. A. Mozo, K. D. Sabanilla and N. S. Gunay, 'WiKA: A Vision Based Sign Language Recognition from Extracted Hand Joint Features using DeepLabCut', en, *Journal of Engineering, Environment, and Agriculture Research*, vol. 3, no. 1, pp. 19–28, Apr. 2024, ISSN: 2651-8260, 2599-4395. DOI: 10.34002/jeear.v3i1.75.

[13] S. M. Youssef, M. Soliman, M. A. Saleh, A. H. Elsayed and A. G. Radwan, 'Design and control of soft biomimetic pangasius fish robot using fin ray effect and reinforcement learning', en, *Scientific Reports*, vol. 12, no. 1, p. 21 861, Dec. 2022, ISSN: 2045-2322. DOI: 10.1038/s41598-022-26179-x.

[14] R. Grandia, F. Farshidian, E. Knoop, C. Schumacher, M. Hutter and M. Bächer, 'DOC: Differentiable Optimal Control for Retargeting Motions onto Legged Robots', en, *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, Aug. 2023, ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3592454.

[15] C. Zhang, J. Sheng, T. Li *et al.*, *Learning Highly Dynamic Behaviors for Quadrupedal Robots*, en, arXiv:2402.13473 [cs], Feb. 2024.

[16] C. Li, M. Vlastelica, S. Blaes, J. Frey, F. Grimminger and G. Martius, *Learning Agile Skills via Adversarial Imitation of Rough Partial Demonstrations*, en, arXiv:2206.11693 [cs], Nov. 2022.

[17] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang and X. Wang, *Expressive Whole-Body Control for Humanoid Robots*, en, arXiv:2402.16796 [cs], Mar. 2024.

[18] T. Nath, A. Mathis, A. C. Chen, M. Bethge and M. W. Mathis, 'Using DeepLabCut for 3D markerless pose estimation across species and behaviors', en, Nov. 2018.