Dyna-0: Cuadrúpedo para aprendizaje automático

Gabriel Torre
LINAR
Universidad de San Andrés
San Fernando, Argentina
torreg@udesa.edu.ar

Agustín Grillo
LINAR
Universidad de San Andrés
San Fernando, Argentina
agrillo@udesa.edu.ar

Roberto Bunge
LINAR
Universidad de San Andrés
San Fernando, Argentina
rbunge@udesa.edu.ar

***Resumen*—En este artículo se presenta el Dyna-0, un robot cuadrúpedo para investigación en estrategias de locomoción y robustez de movimiento, que, basado en un diseño open-source impreso en 3D y mediante una computadora embebida, logra un movimiento robusto en una variedad de terrenos. Las estrategias de locomoción son generadas mediante aprendizaje reforzado en un ambiente simulado. En este trabajo se muestra cómo se pueden lograr marchas robustas valiéndose simplemente de estrategias simples de control lineal donde los sensores de entrada se mapean a los actuadores usando una matriz. Esta estrategia simple logra generalizar la robustez del andar a muchas clases de movimientos incluyendo roto-traslaciones en cualquiera de las direcciones paralelas al piso, aunque solamente haya sido entrenado caminando hacia adelante a velocidad constante. En particular, se trabaja sobre como robustecer un estilo de caminata predefinido, usando exclusivamente sensores propioceptivos, especialmente sensores inerciales. Otro foco de la investigación es la transferencia de lo aprendido entre el mundo simulado y la realidad, y posibles técnicas que ayuden a cerrar la brecha que separa los dos mundos, tales como aleatorización de parámetros o adaptación del dominio.**

***Keywords*—aprendizaje reforzado, cuadrúpedo, robótica móvil**

Figura 1: Fotografía de la plataforma robótica en posición de reposo

I. Introducción

I-A. Relevancia del problema

La robótica móvil tiene numerosas aplicaciones en la actualidad; en la industria, tareas de búsqueda y rescate, en la exploración espacial, y muchas otras. En particular, los robots terrestres han sido una de las plataformas que más auge han tenido, principalmente aquellos que utilizan ruedas para su locomoción. Sin embargo, en ciertas aplicaciones es necesario trasladarse en terrenos más adversos, y si bien existen variantes que les permiten a los robots con ruedas adaptarse a muchos de estos escenarios, para un gran número de aplicaciones este método de locomoción no resulta adecuado. Por ejemplo, en una aplicación de búsqueda y rescate, en la cual el robot debe trasladarse en un camino con rocas o escombros, los vehículos con ruedas no tienen la versatilidad necesaria. Incluso en muchas aplicaciones en las cuales actualmente se utilizan este tipo de vehículos resulta necesario adecuar el ambiente, esto ocurre en oficinas o galpones en los cuales hay escaleras o terrenos irregulares que presentan un desafío para los robots con ruedas.

En resumen, las ventajas de cuadrúpedos por sobre robots con ruedas son:

- Alta movilidad en suelos no estructurados.
- Buena capacidad de carga y eficiencia en superficies casi imposibles para otros tipos de robots.
- Altísima maniobrabilidad.
- Probado e inspirado en la naturaleza.

En los últimos años ha habido un considerable avance en el desarrollo de plataformas robóticas terrestres que utilizan patas para su locomoción. Estos robots han demostrado ser de utilidad para trasladarse en entornos complejos y realizar tareas de manera autónoma. Uno de los principales desafíos de este tipo de plataformas es el control, ya que sus dinámicas son altamente no lineales e inestables, y, por lo tanto, requieren estrategias capaces de lograr un buen rendimiento asegurando robustez. A su vez, el diseño del algoritmo de control que permite al vehículo seguir una trayectoria establecida también es un problema complejo, al ser este un problema sobredimensionado, y que depende del desempeño del algoritmo de control encargado de la estabilidad. Este proyecto tiene como objetivo el diseño de estrategias de control para robots cuadrúpedos. En particular el tipo de problema resulta interesante para explorar estrategias basadas en aprendizaje de máquina, más específicamente en aprendizaje por refuerzo y aprendizaje por imitación, las cuales han demostrado ser técnicas de utilidad para este tipo de aplicaciones.

I-B. Reseña histórica

En la década del 80, Marc Raibert (fundador de Boston Dynamics) y Shimoyama llevaron a cabo un estudio de la cinemática de robots cuadrúpedos y publicaron uno de los primeros trabajos en el área [1], que ha servido de referencia a muchos de los trabajos posteriores.

A partir del año 2000, el desempeño de los robots cuadrúpedos ha evolucionado de manera notable, principalmente debido al avance en dispositivos electromecánicos y a los sistemas microelectromecánicos, *MEMS*.

En 2015, el ETH desarrolló un nuevo robot cuadrúpedo llamado ANYmal [2], que está diseñado para un funcionamiento autónomo en entornos complejos. ANYmal puede percibir la información del entorno a su alrededor para la creación y localización de mapas utilizando sensores láser y cámaras. Por lo tanto, este robot móvil puede seleccionar puntos de apoyo adecuados mientras camina y planificar su ruta de navegación de forma autónoma.

II. Metodología

Se busca robustecer la caminata mediante el uso de estrategias simples que aprovechen el aprendizaje de máquina. En este caso, la estrategia de control está definida mediante una matriz, la cual, en conjunto con un generador de trayectorias, mapea un vector de estado a un conjunto de acciones aplicadas a cada actuador. En concreto, se buscó lograr un trote estable resistente a perturbaciones del terreno y medianamente rápido que utiliza sólo información de un sensor inercial *IMU* para lograr la robustez.

II-A. Simulación

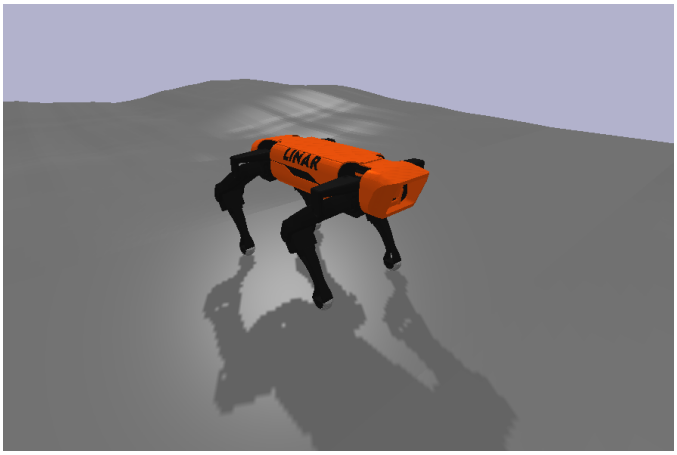
Se trabajó con el motor físico *open-source* llamado Bullet, corriendo con el simulador de robótica PyBullet [3]. Este es un simulador con foco en aprendizaje automático, especialmente en entornos de Gym [4] que facilitan y sistematizan la investigación en aprendizaje por refuerzo. En la Fig. 2a se puede ver el Dyna-0 simulado en Pybullet.

El simulador está basado principalmente en los ambientes implementados para el Minitaur [5] y el OpenQuadruped [6], con algunas funcionalidades extra que fueron adicionadas por los autores.

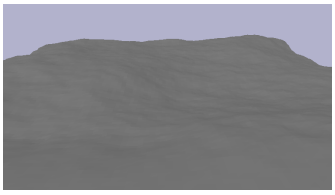
Las dos partes principales de la simulación son el robot articulado y el suelo.

La topología y parte física del robot está definida en un archivo formato URDF. Éste describe los aspectos visuales del cuadrúpedo, las colisiones que le permiten interactuar con el entorno, y los parámetros físicos como por ejemplo el tensor de inercia para cada parte móvil con su respectiva masa.

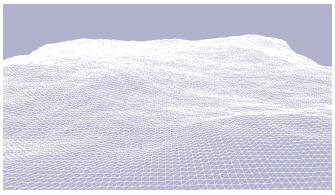
Se modeló la unidad de medición inercial (IMU) aplicando ruido gaussiano sobre el valor ideal simulado. Este modelado se explica más detalladamente en la sección II-D. También se modelaron los servomotores luego de una identificación de parámetros. En cuanto a software, se simuló latencia para así tener en cuenta el retardo entre que se comanda una acción y que el robot real se mueve.



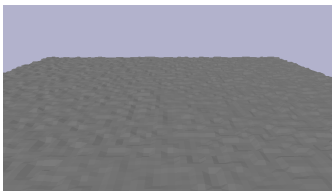
(a)



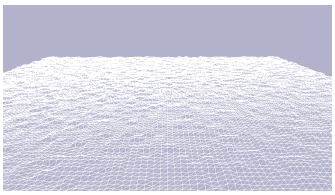
(b)



(c)



(d)



(e)

Figura 2: (a) Simulador de robótica basado en PyBullet con foco en aprendizaje reforzado. (b) Representación sólida del terreno simulado 'colinas'. (c) Estructura alámbrica de la malla que simula el terreno 'colinas'. (d) Representación sólida del terreno simulado 'escalones'. (e) Estructura alámbrica de la malla que simula el terreno 'escalones'.

Se modelaron también diferentes superficies de terreno con el objetivo de simular la mayor cantidad posible de configuraciones de terreno. En el simulador, se decidió utilizar dos geometrías diferentes entre sí, a las que se llamó colinas y escalones. Estas pueden apreciarse en las Fig. 2b y 2d.

II-B. Marcha

La estrategia de aprendizaje reforzado no produce una marcha por sí misma, sino que modula una marcha preexistente que funciona a lazo abierto como una coreografía que mueve las patas sin importar lo que ocurra en el entorno. Esto último será explayado más adelante en la sección II-C.

En este trabajo sólo se analizó trote (movimiento alternado de patas de a pares diagonalmente opuestos), puesto que, éste produce los resultados más robustos a velocidades medias, en contraste con caminatas que dependen de equilibrio estático y están limitadas a velocidades muy bajas. Esta clase de marcha

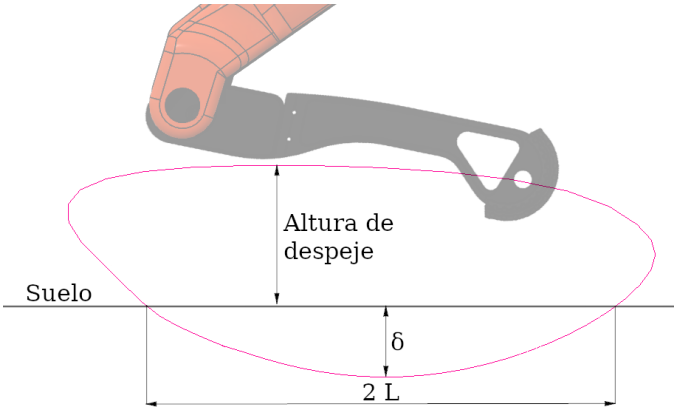


Figura 3: Parámetros de marcha que modulan el generador de trayectorias. Nótese que la penetración δ fue magnificada con fines ilustrativos.

es considerada una de las más eficientes en caninos y es aquella que se utiliza típicamente para evaluación de la marcha en animales [7].

La marcha usada consiste en un ciclo completo partido en dos componentes: apoyo y balanceo. La fase $S(t)$ indica en qué parte del ciclo está cada pata, donde $0 \leq S(t) < 2$, estando en el tramo de apoyo cuando $S(t) < 1$ y el de balanceo cuando $S(t) \geq 1$. Siguiendo el trabajo [6], se modeló el apoyo como una función sinusoidal, Ec. 1, y el balanceo como una curva de Bézier de 12 puntos, Ec. 2 , basado en las trayectorias desarrolladas por el MIT para esta aplicación [8].

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{cases} L(1-2S(t))\cos(\rho) \\ L(1-2S(t))\sin(\rho) \\ -\delta\cos(\dfrac{\pi(1-2S(t))}{2}) \end{cases} \tag{1}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{cases} cos(\rho)\sum_{i=1}^{N}\binom{N}{i}P_i^{xy}(2-S(t))^{N-i}(S(t)-1)^i \\ sin(\rho)\sum_{i=1}^{N}\binom{N}{i}P_i^{xy}(2-S(t))^{N-i}(S(t)-1)^i \\ \sum_{i=1}^{N}\binom{N}{i}P_i^{z}(2-S(t))^{N-i}(S(t)-1)^i \end{cases} \tag{2}$$

Siendo:

L: Largo del paso
$S(t)$: Fase
δ: Profundidad de penetración
ρ: Rotación de las curvas en $\hat{z}$
N: Cantidad de puntos de Bézier
P_i^j: Componente j del punto de Bézier número i

En la Fig. 3 se observan los parámetros relacionados a la generación de trayectorias para la marcha.

II-C. Aprendizaje por refuerzo

Se siguió el esquema propuesto por [6], donde la aplicación de aprendizaje por refuerzo a cuadrúpedos se aplica mediante una *policy* que modifica una caminata base a lazo abierto. La misma, añade un corrector que permite agregar flexibilidad y responder a perturbaciones externas, trabajando a lazo cerrado. En la Fig. 4 se puede ver el diagrama de control del sistema.

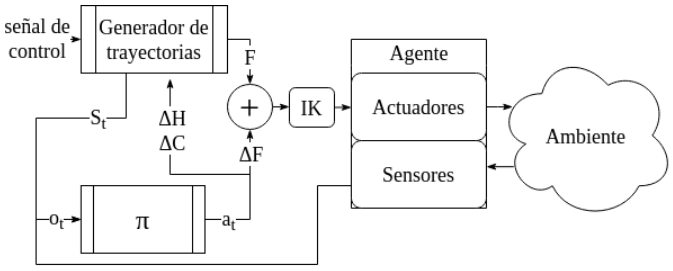


Figura 4: Diagrama de control, la *policy* π corrige las posiciones de las patas dadas por el generador de trayectorias

El problema se planteó como un proceso de Markov parcialmente observable (PMPO). Un PMPO es un proceso estocástico en tiempo discreto formado por un estado S, una observación O de dicho estado, una acción A, una matriz de transición $P_a(s, s')$ y una función recompensa $R_a(s, s')$. Dado un estado s, el agente puede tomar una acción a que conduce a un nuevo estado s', dado por la matriz de transición P_a, y obteniendo una recompensa r. El objetivo del aprendizaje por refuerzo es encontrar una *policy* óptima $\pi^*(a|o)$, la cual maximiza la suma descontada de las recompensas R_γ, siendo γ el factor de descuento.

Al trabajar con una *policy* lineal, la misma puede ser definida mediante una matriz M. De esta forma, podemos obtener las acciones en base a las observaciones del estado como:

$$a = \pi(o) = Mo \tag{3}$$

Se usó Augmented Random Search (ARS) [9] para encontrar la *policy* M. Este método es conocido por ser eficiente tanto durante el entrenamiento como durante la inferencia. Es, en resumen, una búsqueda aleatoria básica en el espacio de parámetros (M_{ij}) con algunas mejoras adicionales. Una de ellas, por ejemplo, asegura un entrenamiento parejo entre todos los parámetros cuando estos tienen distintas escalas, lo que es común en el caso de la robótica.

Esencialmente, se aleatorizan los parámetros M_{ij} en ciertas direcciones $\nu(\delta_k)_{ij}$. Siendo el hiper-parámetro ν, la desviación estándar del ruido de exploración; δ_k la matriz de desviación multiplicativa de la iteración k-ésima, muestreada de una distribución normal; e ij el elemento en la fila i-ésima y columna j-ésima.

En cada actualización de parámetros, un grupo de k *policies* aleatorizadas son evaluadas como se muestra en la Ec. 4, y la matriz M es actualizada en la dirección de las que más recompensa obtuvieron.

$$\pi^k(o) = (M \pm \nu\delta_k)o \tag{4}$$

La evaluación está hecha mediante la función recompensa promediada por paso temporal ($R_t = R/T_{ep}$). Dicha función recompensa consiste en tres términos ponderados, cuya suma ponderada puede verse en la Ec. 5

- D_x: Distancia recorrida hacia adelante.
- p, r: Rotación del chasis en cabeceo y alabeo, (*pitch* y *roll* en inglés).
- ω_p, ω_r: Velocidad angular del chasis en cabeceo y alabeo.

donde:

$$\mathcal{R} = \sum_{t}^{T_{episode}} (\lambda_D D_x^t - \lambda_{rot}(|r^t| + |p^t|) - \lambda_\omega(|\omega_r^t| + |\omega_p^t|)) \quad (5)$$

El espacio de observación consiste en:

- p, r: Rotación del chasis en cabeceo y alabeo.
- ω: Velocidad angular del chasis (Giróscopo).
- a: Aceleración lineal del chasis (Acelerómetro).
- S: Fase correspondiente al movimiento de cada pata en el generador de trayectorias.

Para cada paso de tiempo t, véase el espacio de observaciones en la Ec. 6.

$$\begin{aligned} o_t = (&r_t, p_t, \omega_t^x, \omega_t^y, \omega_t^z, a_t^x, a_t^y, a_t^z, \\ &S_t^{FL}, S_t^{FR}, S_t^{BL}, S_t^{BR}) \end{aligned} \quad (6)$$

El espacio de acciones a consiste en un vector de 14 dimensiones, donde se encuentran dos parámetros que modulan la marcha base (altura de despeje de las patas C y altura del chasis H) y 12 correctores para las patas, véase la Ec. 7.

$$\begin{aligned} a_t = (&\Delta C_t, \Delta H_t, \\ &\Delta Fx_t^{FL}, \Delta Fy_t^{FL}, \Delta Fz_t^{FL}, \\ &\Delta Fx_t^{FR}, \Delta Fy_t^{FR}, \Delta Fz_t^{FR}, \\ &\Delta Fx_t^{BL}, \Delta Fy_t^{BL}, \Delta Fz_t^{BL}, \\ &\Delta Fx_t^{BR}, \Delta Fy_t^{BR}, \Delta Fz_t^{BR}) \end{aligned} \quad (7)$$

II-D. Pase de simulación a realidad

Como se mencionó, el entrenamiento del controlador se realizó exclusivamente en simulación. Dado que la misma no es exactamente igual a la realidad, se implementaron diversas técnicas para mitigar los efectos que esto puede generar al evaluar el controlador en el prototipo real. En particular, usamos dos técnicas conocidas como Identificación de Sistema y Aleatorización de Dominio. La primera busca modelar los componentes del cuadrúpedo con alta precisión, mientras que la segunda busca contemplar posibles errores de modelado, extendiendo el dominio del sistema, e intentando abarcar el dominio real dentro del mismo.

II-D1. Identificación del sistema: En lo que respecta a la identificación del sistema, se buscó modelar los actuadores, el IMU y la latencia.

Por el lado de los actuadores, dado que los mismos son servomotores, se modela su control interno mediante un control de posición PD véanse Ec. 8, 9 y 10.

$$\Delta\theta = \theta_o - \theta_a \quad (8)$$

$$v = \frac{\partial \theta_a}{\partial t} \quad (9)$$

$$u_{pwm} = K_p\ \Delta\theta - K_d\ v \quad (10)$$

A partir de lo cual, en la Ec. 11 modelamos el torque generado.

$$T = K_t\ \frac{(u_{pwm}\ V - K_t\ v)}{R} \quad (11)$$

Donde (valores reales usados en nuestra simulación):

- θ_o: ángulo objetivo del motor.
- θ_a: ángulo actual del motor.
- $V = 6{,}7 \pm 0{,}08$ v: Tensión que alimenta a los motores.
- $R = 0{,}02\Omega$: Resistencia de armadura.
- $K_p = 1$: Constante proporcional.
- $K_d = 0{,}03$: Constante derivativa.
- $K_t = 0{,}1$: Constante de torque.

Respecto a la latencia, también se caracterizó la misma, obteniendo los siguientes resultados:

- Desde un movimiento del cuerpo a lectura de esa perturbación por el IMU dentro de ROS: $11{,}3 \pm 3{,}5$ ms
- Desde un comando dado en ROS al movimiento de uno de los miembros: $25{,}8 \pm 2{,}8$ ms
- Latencia total: $37{,}1 \pm 6{,}3$ ms

Por otro lado, se modeló el ruido del IMU como un desvío gaussiano en torno a la media, con los siguientes desvíos estándar:

- $\sigma_r = \sigma_p = 0{,}00065\ rad$
- $\sigma_a = 0{,}078\ m/s^2$
- $\sigma_\omega = 0{,}20\ rad/s$

II-D2. Aleatorización de dominio: En lo que respecta a la aleatorización de dominio, se decidió aleatorizar tanto la fricción del terreno como su geometría. A su vez, se agregó una fuerza externa para simular posibles perturbaciones al agente, la cual también es aleatorizada.

En particular, la magnitud de dicha fuerza se obtiene de una distribución uniforme $\mathcal{U}[0N,\ 0{,}4N]$, mientras que la dirección es completamente aleatoria, aunque manteniéndose ortogonal al vector gravedad.

La fricción del terreno es muestreada a partir de una distribución uniforme $\mathcal{U}[0{,}2,\ 0{,}5]$. Respecto a la geometría, se utilizaron principalmente dos tipos de terreno: escalones y colinas. Los mismos son definidos mediante una grilla rectangular, donde cada elemento de la grilla representa la altura de ese punto del terreno.

Para escalones se le asigna la misma altura a grupos de 4 celdas vecinas. Siendo T, la matriz que representa el mapa de alturas como se ve en la Ec. 12. Se usa este modo de generar terreno para que haya muchas superficies planas donde pararse sin deslizar a modo de mesetas de distintas alturas.

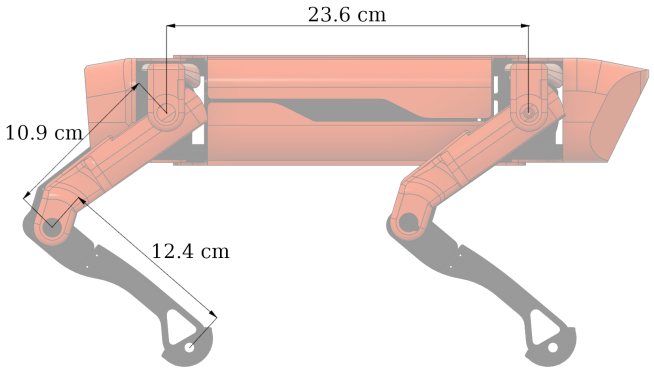


Figura 5: Medidas principales del robot relevadas en el robot físico

$$T_{i,j} = T_{i+1,j} = T_{i,j+1} = T_{i+1,j+1} \sim \mathcal{U}[h_{min}, h_{max}] \tag{12}$$

En cuanto a colinas, este tipo de terreno es modelado mediante la implementación de ruido Perlin, el cual es utilizado para obtener un terreno con un aspecto natural. En especifico, se usaron los siguientes valores para cada parámetro en la implementación.

- Octavas: [3, 4, 5, 6]
- Lacunaridad: 0.3
- Persistencia: [2.15, 2.3]

II-E. Plataforma robótica

La plataforma robótica Fig. 1 está basada en el proyecto de código abierto Spot Micro [10], con ciertas modificaciones. Los cambios mecánicos incluyen una base más larga, soportes adicionales para el hombro, cambios en el montaje de los servos, cambios en el montaje interno de las baterías y cambios menores tanto en la nariz como en la parte trasera. En cuanto a la electrónica, se rediseñó el esquema de drivers y bus I²C para no requerir de placas custom y admitir un procesador más potente. Se seleccionaron motores servo con un torque especificado de $30\,kg\,cm$ y una velocidad mayor a $0{,}13\,s/60°$. Las medidas principales del robot pueden verse en la Fig. 5.

El procesador principal consiste en una Raspberry Pi 4 corriendo ROS Noetic sobre Ubuntu. Dado que la frecuencia del lazo de control está limitada por la velocidad de refresco de los servomotores, la cual es menor o igual a 50 Hz (periodo: 20 ms), no se utilizó ningún sistema operativo en tiempo real (RTOS) o parche correspondiente para el Linux debido a que carece de sentido darle tanta importancia a los retardos en el computo, teniendo en cuenta el retardo inherente que inducen los actuadores electro-mecánicos.

III. Resultados

Para probar los comportamientos aprendidos, se usó un escalón descendente de distintas alturas. Las *policies* no fueron entrenadas explícitamente para este tipo de obstáculo pero de todas maneras pueden superarlo de manera robusta, mostrando así las capacidades de generalización del aprendizaje obtenido. Aunque este proyecto está pensando para terreno no estructurado, un escalón es un ambiente controlado y discreto en el que se puede variar la dificultad de manera simple, cambiando la altura del mismo. Un enfoque de testeo similar fue usado en [11]. El setup experimental puede verse en la Fig. 6. En las tablas I y II se muestra el porcentaje de supervivencia a la prueba del escalón, para las distintas *policies* evaluadas. El ensayo se considera exitoso si la plataforma logra descender el escalón sin tumbarse, aplicándole un comando de caminata frontal constante.

Figura 6: Setup experimental, escalón de altura variable que permite variar la dificultad de la prueba de manera simple

Como se mencionó anteriormente, se evaluaron y compararon diversas *policies*. Entre ellas se encuentran las siguientes. RDR* y Fast Policy, creada por los autores; Baseline, una *policy* ideada para una plataforma robótica similar en [6], que fue re-entrenada para esta plataforma robótica, con los parámetros originales propuestos por los autores correspondientes; Open Loop, no usa ningún tipo de control, simplemente muestra el mal desempeño del robot a lazo abierto al enfrentarse con obstáculos, y la importancia de la *policy* para superar los mismos.

Tabla I: Supervivencia en Test de escalón descendente 60mm

Policy	**Caidas**
RDR*	1/10
Baseline [6][a]	1/10
Open Loop	8/10

[a] re-entrenado para nuestro robot

Otro test que se realizó fue de velocidad hacia adelante en terreno plano, donde se promedió la velocidad de tres carreras de 5m. Debe tenerse en cuenta que una *policy* más lenta usualmente es más estable, por lo que una mejora en robustez y velocidad a la vez es algo deseable, los resultados pueden ser vistos en la tabla III.

Se puede observar como una policy diseñada para ser más

Tabla II: Supervivencia en Test de escalón descendente 75mm

Policy	Caidas
RDR*	1/10
Baseline [6][a]	5/10
Open Loop	9/10
Fast Policy	4/10

[a] re-entrenado para nuestro robot

Tabla III: Velocidad frontal en terreno plano

Policy	Velocidad[m/s]
RDR*	0.25
Baseline [6][a]	0.21
Open Loop	0.30
Fast Policy	0.33

[a] re-entrenado para nuestro robot

rápida (Fast policy) desarrolla mayor velocidad pero tiene una tasa mayor de caídas, evidenciando el *trade-off* antes mencionado entre velocidad y robustez.

IV. Discusión

El sistema resultó capaz de aprender comportamientos que robustecen considerablemente el andar usado una simple matriz como *policy*. El entrenamiento del robot se lleva a cabo exclusivamente con una caminata hacia adelante. Sin embargo, los comportamientos aprendidos generalizan bien para otras direcciones de caminata, como caminatas laterales o roto-traslaciones, e incluso rotar caminando hacia atrás. Dado que ambos pares de patas apuntan frontalmente la caminata hacia atrás es especialmente difícil para esta configuración de cuadrúpedo.

V. Conclusiones y trabajo futuro

Actualmente se está investigando respecto a la transferencia del aprendizaje entre simulación y realidad. En particular, en técnicas que ayuden a achicar la brecha que separa a éstas, específicamente en una implementación cuidadosa de aleatorización de dominio.

Una línea de investigación interesante es analizar otras marchas alternativas al trote, como ambladura, paso o las distintas clases de galope que se observan en cánidos y otro cuadrúpedos como caballos.

La continuación más natural para este trabajo es remplazar el simple modelo lineal actual por una red neuronal que pueda representar mejor comportamientos complejos o no lineales. También se puede utilizar una secuencia de observaciones como entrada al modelo, para sí poder estimar mejor el estado completo, ya que como se explicó, el problema a priori es un PMPO.

Otra línea de investigación interesante es realizar un ajuste fino en el robot real, luego de haber entrenado en simulación. Esto podría ser algo muy beneficioso para evitar totalmente cualquier problema en la transferencia de simulación a realidad.

Referencias

[1] M. Raibert, *Legged Robots that Balance*, ser. Artificial Intelligence Series. MIT Press, 1986. [Online]. Available: https://books.google.com.ar/books?id=GV_lHAAACAAJ

[2] "Anymal." [Online]. Available: https://rsl.ethz.ch/robots-media/anymal.html

[3] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[5] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots," *arXiv e-prints*, p. arXiv:1804.10332, Apr. 2018.

[6] M. Rahme, I. Abraham, M. L. Elwin, and T. D. Murphey, "Dynamics and domain randomized gait modulation with bezier curves for sim-to-real legged locomotion," *CoRR*, vol. abs/2010.12070, 2020. [Online]. Available: https://arxiv.org/abs/2010.12070

[7] B. J. Carr and D. L. Dycus, "Canine gait analysis," Aug 2020. [Online]. Available: https://todaysveterinarypractice.com/recovery-rehab-canine-gait-analysis/

[8] D. J. Hyun, S. Seok, J. Lee, and S. Kim, "High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah," *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1417–1445, 2014. [Online]. Available: https://doi.org/10.1177/0278364914532150

[9] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *CoRR*, vol. abs/1803.07055, 2018. [Online]. Available: http://arxiv.org/abs/1803.07055

[10] [Online]. Available: https://spotmicroai.readthedocs.io/en/latest/

[11] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning Quadrupedal Locomotion over Challenging Terrain," *arXiv e-prints*, p. arXiv:2010.11251, Oct. 2020.