Locomoción en Simulación de Robot Hexápodo sobre Terrenos Irregulares utilizando Aprendizaje por Refuerzo Profundo

Tania Sofía Ferreyra*†, Nicolás Romero*†, Matías Sandacz†, Emanuel Slawinski‡ and Pablo De Cristóforis†
*Estos autores contribuyeron en partes iguales
† Departamento de Ciencias de la Computación, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires, Argentina
‡ Instituto de Automática, San Juan, Argentina

***Resumen*—El control de los robots con patas articuladas resulta complejo debido a la alta dimensionalidad del sistema. Este trabajo aborda el problema de locomoción en robots hexápodos ciegos mediante el uso de Aprendizaje por Refuerzo Profundo, como alternativa a las estrategias de control tradicional. Nos basamos en el enfoque de aprendizaje maestro-estudiante propuesto en otro trabajo para un robot cuadrúpedo de 12 grados de libertad y lo extendimos para un robot hexápodo de 16 grados de libertad. Implementamos y evaluamos el módulo del maestro para el robot PhantomX Mark II utilizando el simulador Gazebo, ROS para la comunicación entre los componentes de control y la librería StableBaselines para los algoritmos de aprendizaje por refuerzo. Los entrenamientos se realizaron con terrenos generados con ruido de Perlin y terrenos con bloques de altura aleatoria, adaptativamente seleccionados para aumentar la dificultad a medida que progresa el aprendizaje. Los resultados alcanzados muestran la factibilidad de utilizar Aprendizaje por Reuferzo Profundo para abordar el problema de la locomoción en robots hexápodos sobre terrenos irregulares.**

***Palabras clave*—robot hexápodo, locomoción, control, aprendizaje por refuerzo profundo**

I. INTRODUCCIÓN

Los robots con patas articuladas se destacan por su capacidad para realizar tareas en entornos desafiantes, hostiles o inaccesibles para otros robots terrestres con sistemas de locomoción basado en ruedas u orugas, al mismo tiempo que pueden transportar más carga y operar durante más tiempo que los vehículos aéreos no tripulados. Sus habilidades para realizar movimientos omnidireccionales y atravesar terrenos irregulares los convierten en una excelente opción para diversas aplicaciones, como las misiones de Búsqueda y Rescate Urbano (USAR por sus siglas en inglés) en construcciones colapsadas en situaciones de catástrofe, para operar en ambientes subterráneos como minas e incluso para la exploración extraplanetaria. Sin embargo, es sabido que el control de movimiento de este tipo de robots es sumamente desafiante debido a la cantidad de grados de libertad. Una alternativa a las estrategias de control tradicional es el uso de aprendizaje por refuerzo (RL del inglés Reinforcement Learning).

Existen algunos trabajos en el estado del arte que han explorado el uso de RL específicamente en robots hexápodos. Trabajos como [1] usan RL en simulación para que un hexápodo aprenda a caminar en terreno plano. En [2] se implementa RL para ajustar los parámetros de un generador de patrón central (CPG por sus siglas en inglés) en un robot real, pero los terrenos utilizados son prácticamente planos. En [3] el robot es solo simulado y el terreno tiene irregularidades no muy pronunciadas ni diversas. En [4], sí se consideran terrenos más desafiantes y se propone un sistema de dos etapas: una que aprende por DRL (Deep Reinforcement Learning) una caminata óptima para cada terreno dentro de N posibles tipos y un clasificador que aprende a predecir el tipo de terreno para seleccionar la política experta correspondiente. El trabajo presenta resultados muy interesantes, pero únicamente en simulación y utilizando un modelo de robot simplificado a figuras geométricas sencillas, lo que impide asegurar que el resultado sea trasladable al mundo real.

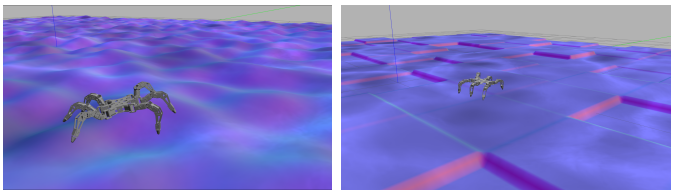

Figura 1: Ejemplo de los terrenos usados durante el aprendizaje: colinas (izquierda) y bloques (derecha).

Existe más desarrollo realizado para robots cuadrúpedos, donde hay múltiples ejemplos de sistemas llevados al mundo real, con resultados sumamente interesantes. En [5] se entrena con RL una política que luego se transfiere a un robot cuadrúpedo real y le permite moverse en escaleras, terrenos con agujeros y rocas. Aún más impresionante son los resultados obtenidos en [6], donde logran que un cuadrúpedo ANYmal pueda atravesar escaleras, terreno boscoso inclinado, vegetación, nieve entre otros terrenos. No obstante, ambos trabajos hacen uso de un sistema de visión, que escapa al alcance de este trabajo. Por otro lado, en [7] obtienen resultados similares para un robot ANYmal utilizando únicamente medidas propiocépticas, entrenando en simulación y pudiendo transferir la política aprendida al robot real sin ajuste adicional. Para ello proponen una estrategia maestro-estudiante, que consiste en entrenar en simulación un robot maestro con acceso a información privilegiada obtenida del simulador y luego usar la pólitica obtenida para entrenar por aprendizaje supervisado un robot estudiante que solo tiene acceso a las mediciones que estarían disponibles en el robot real.

En este trabajo presentamos el módulo maestro de un sistema de locomoción para robots hexápodos utilizando únicamente mediciones propioceptivas y Aprendizaje por Refuerzo Profundo, basado en el enfoque maestro-alumno que se presenta en [7]. Específicamente, el sistema está diseñado para el robot PhantomX Mark II. A diferencia del trabajo antes mencionado, hacemos uso de herramientas ampliamente utilizadas por la comunidad: Gazebo [8] como plataforma de simulación, ROS [9] para comunicación entre los distintos módulos del sistema y la librería Stable Baselines 3 [10] para los algoritmos de Aprendizaje por Refuerzo Profundo. Los terrenos utilizados para el entrenamiento, mostrados en la Figura 1, fueron terrenos de colinas, generados con ruido de Perlin, y terrenos de bloques, de altura aleatoria, seleccionados adaptativamente con un filtro de partículas para aumentar progresivamente la dificultad del terreno con el que se entrena.

II. SISTEMA

En este trabajo nuestro objetivo es enseñar a un robot hexápodo a caminar en terrenos irregulares usando únicamente sensores propioceptivos. El robot utilizado para este trabajo es el PhantomX Mark II, cuyo modelo para el simulador en formato URDF fue sacado de [11]. La red neuronal que se entrena mediante DRL modifica la trayectoria de cada pata fijada *a priori* por el controlador de bajo nivel, que por sí solo mantiene al robot caminando en el lugar en un terreno plano, para que se adapte al terreno irregular y siga el comando de velocidad deseado. Para ello se modifican la frecuencia de la trayectoria de cada pata, f_i, y la posición final de referencia adicionando un diferencial, Δr_i, como se explica más adelante en la sección II-C. El sistema aprende a partir de un currículo de terrenos adaptativo. Se definen dos tipos de terreno, bloques y colinas, en los que se desea entrenar, y se utiliza un filtro de partículas para seleccionar terrenos con dificultad creciente para la etapa de entrenamiento en la que se encuentre.

II-A. Estructura General

La estructura general del control, mostrada en la Figura 2, se compone, en primer lugar, de una red neuronal, que recibe comandos de velocidad lineal y angular, y cuya salida son los valores de frecuencia f_i de cada pata y los diferenciales de movimiento Δr_i. Los valores de frecuencia sirven de entrada al bloque de generación de trayectoria que define el movimiento a realizar por cada pata. El valor de posición objetivo para cada pata es el obtenido por el generador de trayectoria más el diferencial de movimiento Δr_i determinado por la red neuronal. Este valor final se procesa con las ecuaciones de cinemática inversa para obtener las posiciones deseadas para cada articulación, que son enviadas a los controladores PID de cada articulación del robot. Finalmente, el robot y el simulador devuelven las observaciones necesarias para armar el vector de observación de la red neuronal.

II-B. Modelo Cinemático

Como podemos ver en la Figura 3, es posible establecer la posición de la pata a partir de los ángulos de coxa, femur y

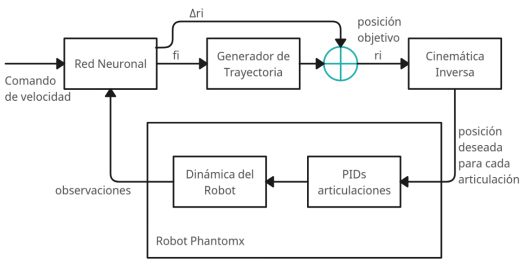

Figura 2: Diagrama en bloques del sistema de control del robot.

tibia (θ_C, θ_F y θ_T, respectivamente) conociendo las longitudes de cada parte (LC, LF y LT, respectivamente) haciendo uso del eje auxiliar r_{xy}:

$$r_{xy} = LC + LFcos(\theta_F) + LTcos(\theta_F + \theta_T)$$

$$z = LFsen(\theta_F) + LTsen(\theta_F + \theta_T)$$

$$x = r_{xy}cos(\theta_C)$$

$$y = r_{xy}sen(\theta_C)$$

A partir de estas relaciones podemos obtener las ecuaciones de cinemática inversa:

$$\theta_C = atan2(y, x)$$

$$u = \sqrt{x^2 + y^2} - LC$$

$$c2 = cos(\theta_T) = \frac{u^2 + z^2 - LF^2 - LT^2}{2LF\ LT}$$

$$s2 = sen(\theta_T) = -\sqrt{1 - c2^2}$$

$$c1 = cos(\theta_F) = \frac{u(LF + LT\ c2) + z\ LT\ s2}{u^2 + z^2}$$

$$s1 = sen(\theta_F) = \frac{z(LF + LT\ c2) - u\ LT\ s2}{u^2 + z^2}$$

$$\theta_F = atan2(s1, c1)$$

$$\theta_T = atan2(s2, c2)$$

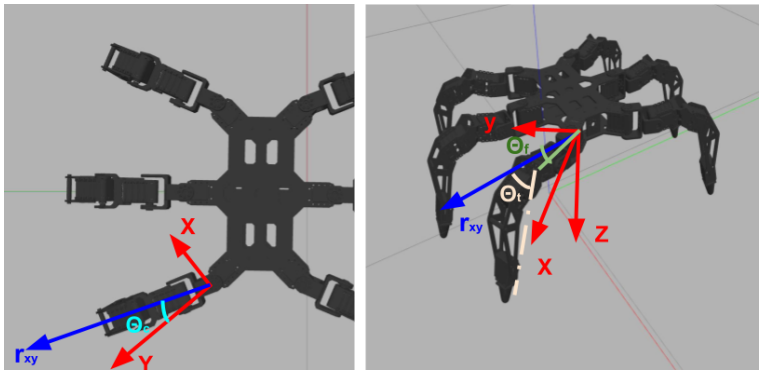

Figura 3: Sistema de coordenadas para obtener las ecuaciones cinemáticas para la pata derecha delantera.

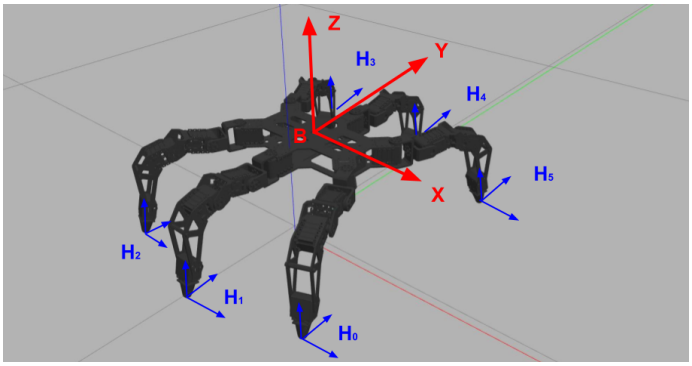

Figura 4: Robot PhantomX con los sistemas de coordenadas de referencia H_i (azul) y el sistema de coordenadas de la base B (rojo).

Como estas ecuaciones asumen cierta orientación del sistema de coordenadas de cada articulación y que todos los ángulos en cero corresponden a la pata estirada recta, los valores de comando real a enviar a Gazebo son: $\theta_{coxa} = \theta_C$, $\theta_{femur} = -\theta_F$ y $\theta_{tibia} = \theta_T + \pi/2$.

II-C. Control de Bajo Nivel

El control de bajo nivel consta de varias etapas. La primera es la generación de trayectoria de referencia para cada pata. En cada instante de tiempo la fase de la pata i está dada por:

$$\phi_i = (\phi_{i,0} + (f_0 + f_i)t)(mod\ 2\pi)$$

donde $\phi_{i,0}$ es la fase inicial, f_0 es una frecuencia base común a todas las patas y f_i es la frecuencia de offset de la i-ésima pata. La fase indica etapa de contacto para $\phi_i \in [0, \pi)$ y movimiento para $\phi_i \in [\pi, 2\pi)$. Dada la fase ϕ_i, la posición objetivo de la pata será $r_i = F(\phi_i) + \Delta r_i$, donde Δr_i es la salida de la red neuronal, así como f_i, y la función F viene dada por:

$$F(\phi_i) = \begin{cases} (h(-2k^3 + 3k^2) - 0{,}5)^{H_i z} & k \in [0, 1] \\ (h(2k^3 - 9k^2 + 12k - 4) - 0{,}5)^{H_i z} & k \in [1, 2] \\ -0{,}5^{H_i z} & sino \end{cases}$$

donde $k = 2(\phi_i - \pi)/\pi$, h es un parámetro para la altura máxima del paso y H_i es un sistema de coordenadas de referencia colocado a una distancia del marco de coxa dada por la postura de parada de referencia, cuya orientación es siempre horizontal, con el eje x paralelo a la proyección ortogonal del eje x de la base del robot sobre el plano xy, como se puede ver en la Figura 4. De esta forma si $f_i = 0$ y $\Delta r_i = \overrightarrow{0}$ para todo i, el robot se mantiene caminando en el lugar.

Una vez obtenido los r_i en sus sistemas de coordenadas H_i, se convierte al sistema de coordenadas de coxa de cada pata y se usan las ecuaciones de cinemática inversa para obtener los valores de ángulos a usar como comandos de los controles de posición de cada articulación (los controladores PID ya están incluidos en el modelo URDF del robot PhantomX).

II-D. Currículum de Terrenos

Para el entrenamiento se utilizan dos tipos de terrenos que buscan cubrir variedad suficiente para permitir que la política aprendida pueda utilizarse en terrenos más generales. Éstos se describen por medio de un vector de párametros $c_T = (c_{T,1}, c_{T,2}, c_{T,3})$.

- Terreno de colinas: basado en el ruido de Perlin, se genera a partir de 3 parámetros: rugosidad, frecuencia y amplitud del ruido. La altura de cada elemento del mapa de alturas se distribuye $Perlin(c_{T,2}, c_{T,3})[i, j] + \mathcal{U}(-c_{T,1}, c_{T,1})$.
- Terreno de bloques: consiste en bloques cuadrados de altura aleatoria. Todos los bloques tienen tamaño $c_{T,1} \times c_{T,1}$ con altura muestreada de $\mathcal{U}(0, c_{T,2})$.

Ambos terrenos se pueden apreciar en la Figura 1.

II-E. Filtro de Partículas

La generación de terrenos se realiza de forma adaptativa a través del uso de un filtro de partículas, siguiendo el enfoque presentado en [7]. En primer lugar veremos algunas definiciones necesarias.

La transitabilidad de un terreno generado mide la tasa de éxito al atravesarlo. La podemos utilizar para evaluar el progreso del aprendizaje de un agente. Se podría utilizar la función de recompensa, pero ésta métrica resulta ser mas intuitiva. En primer lugar, definimos la función v de etiquetado para la transición desde el estado s_t al estado s_{t+1} luego de aplicar la acción a_t:

$$v\left(s_t, a_t, s_{t+1}\right) = \begin{cases} 1 & \text{si} \quad v_{pr}\left(s_{t+1}\right) > 0{,}2 \\ 0 & \text{si} \quad v_{pr}\left(s_{t+1}\right) < 0{,}2 \vee \text{ terminación} \end{cases}$$

donde $v_{pr}(s_{t+1})$ es el producto interno entre el vector velocidad de la base y la dirección del comando en el tiempo $t+1$, que indica la velocidad con la cual el agente logra seguir la velocidad indicada por el comando. Si la política π puede moverse en la dirección dada más rápido que $0,2$ m/s, vamos a considerar al terreno transitable en esta dirección.

La transitabilidad de un terreno c_T bajo la política π se define como el siguiente valor esperado:

$$Tr(c_T, \pi) = \mathbb{E}_{\xi \sim \pi}\left\{v(s_t, a_t, s_{t+1} \mid c_T)\right\} \in [0.0, 1.0]$$

donde ξ son trayectorias generadas por π.

El objetivo del método de generación adaptativa de terrenos es obtener una distribución para muestrear parámetros c_T con transitabilidad en un rango medio $\mathrm{Tr}\left(c_T, \pi\right) \in [0{,}5, 0{,}9]$. La razón es generar terrenos que no sean ni demasiado fáciles ni demasiado difíciles para el agente. Con esto en mente, dada una política π y un terreno c_T definimos la deseabilidad de un terreno $\mathrm{Td}\left(c_T, \pi\right)$ como la probabilidad de que su transitabilidad esté en el rango deseado:

$$\begin{aligned} \mathrm{Td}\left(c_T, \pi\right) &:= \Pr\left(\mathrm{Tr}\left(c_T, \pi\right) \in [0.5, 0.9]\right) \\ &= \mathbb{E}_{\xi \sim \pi}\left\{Tr\left(c_T, \pi\right) \in [0.5, 0.9]\right\} \end{aligned}$$

Los terrenos con mayor probabilidad de tener una transitabilidad en el rango $[0.5, 0.9]$ serán los mas deseados.

Definimos la variable de medida y_j^k tal que $y_j^k = 1$ si $\mathrm{Tr}\left(c_{T,j}^k, \pi\right) \in [0.5, 0.9]$. Podemos reescribir la deseabilidad del terreno en términos de la probabilidad de la variable de medida:

$$\Pr\left(y_j^k \mid c_{T,j}^k\right) = \Pr\left(\mathrm{Tr}\left(c_{T,i}^k, \pi\right) \in [0.5, 0.9]\right) = \mathrm{Td}\left(c_{T,j}^k, \pi\right)$$

En la implementación, dicha probabilidad de medida se computa reemplazando la esperanza matemática por un promedio de datos obtenidos durante el entrenamiento:

$$\Pr\left(y_j^k \mid c_{T,j}^k\right) \approx \sum^{N_{\text{traj}}} \frac{\chi_{[0.5,0.9]}\left(\mathrm{Tr}\left(c_{T,j}^k, \pi\right)\right)}{N_{\text{traj}}}$$

en donde N_{traj} denota la cantidad de trayectorias obtenidas utilizando $c_{T,j}^k$ y χ es la función característica, que da 1 si $\mathrm{Tr}\left(c_T, \pi\right) \in [0.5, 0.9]$ y 0 sino.

El filtro de partículas es un algoritmo utilizado para aproximar una distribución de probabilidad no paramétrica de forma iterativa. En este caso, nos interesa modelar la distribución de terrenos c_T tal que $\mathrm{Tr}\left(c_T, \pi\right) \in [0.5, 0.9]$. Se comienza con un conjunto de terrenos, también llamados partículas, aleatorios e iterativamente se actualizan dichas partículas hasta converger a la distribución objetivo. Inicialmente, aproximamos la distribución con un conjunto finito de partículas, $c_T^k \in \mathcal{C}, k \in 1, \cdots, N_{\text{particle}}$. Estas partículas se inicializan de manera uniforme.

El algoritmo se basa en las siguientes suposiciones:

- Los parámetros de terrenos con $\mathrm{Tr}\left(., \pi\right)$ similares están cerca en Espacio Euclídeo.
- Una política π que fue entrenada en un área en C podrá interpolar a terrenos cercanos.

Basado en nuestras suposiciones, el filtro de partículas irá iterativamente incrementando la dificultad de los terrenos hasta converger a la distribución de c_T tal que $\mathrm{Tr}\left(c_T, \pi\right) \in [0.5, 0.9]$ y nuestro agente se irá progresivamente adaptando a los nuevos terrenos generados asegurando un aprendizaje continuo.

II-F. Vector de Estado

El vector de estado es la entrada a la red neuronal, que se construye a partir del comando de velocidad, mediciones del simulador y variables internas del sistema. Se actualiza a $50Hz$, que es la frecuencia de trabajo del sistema de control de alto nivel. Para el robot maestro implementado en este trabajo, siguiendo [7], el vector de estado está compuesto por:

- Dirección de velocidad lineal deseada ($({}^B_{IB}\hat{v}_d)_{xy}$).
- Dirección de velocidad angular deseada ($({}^B_{IB}\hat{\omega}_d)_z$).
- Vector de gravedad.
- Velocidad angular de la base (${}^B_{IB}\omega$).
- Velocidad lineal de la base (${}^B_{IB}v$).
- Posición/velocidad de cada articulación ($\theta_i, \dot{\theta}_i$).
- Fases para la generación de trayectorias ($sin(\phi), cos(\phi)$).
- Frecuencias para la generación de trayectorias ($\dot{\phi}$).
- Frecuencia base (f_0).
- Últimos dos valores de error de posición de cada articulación.
- Últimos dos valores de velocidad de cada articulación.
- Últimos dos valores de posición deseada de cada pata ($(r_{i,d})_{t-1,t-2}$).
- Normal del terreno en cada pata.
- Altura de cada pata y de 8 puntos alrededor sobre un círculo de radio $10cm$.
- Fuerzas de contacto de cada pata.
- Estados de contacto de cada pata.
- Estados de contacto de cada fémur.
- Estados de contacto de cada tibia.
- Coeficientes de fricción pata-suelo.

II-G. Función de Recompensa

La función de recompensa o refuerzo utilizada en este trabajo sigue lo propuesto en [7] y es:

$$R = 0{,}05r_{vl} + 0{,}05r_{va} + 0{,}05r_b + 0{,}01r_{ep} + 0{,}02r_c + 0{,}025r_s \tag{1}$$

donde los términos se explican a continuación, considerando que (${}^B_{IB}v$) es la velocidad lineal de la base respecto del sistema global expresada en coordenadas de la base, (${}^B_{IB}\hat{v}_d$) es el correspondiente comando de velocidad lineal normalizado, análogo para la velocidad lineal ω, y el subíndice xy indica la proyección ortogonal sobre este plano mientras que z indica la proyección ortogonal sobre esta dirección.

- Recompensa por velocidad lineal (r_{vl}):

$$r_{vl} = \begin{cases} e^{-\alpha_{vl}(v_p - 0,1)^2} & v_p < v_{threshold} \\ 1 & v_p \geq v_{threshold} \end{cases}$$

  donde $v_p = ({}^B_{IB}v)_{xy}({}^B_{IB}\hat{v}_d)_{xy}$ es la proyección ortogonal de la velocidad lineal de la base en xy sobre la dirección del comando de velocidad lineal.
- Recompensa por velocidad angular (r_{va}):

$$r_{va} = \begin{cases} e^{-\alpha_{va}(\omega_p - 0,1)^2} & \omega_p < \omega_{threshold} \\ 1 & \omega_p \geq \omega_{threshold} \end{cases}$$

  donde $\omega_p = ({}^B_{IB}\omega)_z({}^B_{IB}\hat{\omega}_d)_z$ es la proyección ortogonal de la velocidad angular de la base en z sobre la dirección del comando de velocidad angular.
- Recompensa por movimiento de la base (r_b):

$$r_b = e^{-\alpha_b v_0^2} + e^{-\alpha_b ||({}^B_{IB}\omega)_{xy}||^2}$$

  donde $v_0 = ||({}^B_{IB}v)_{xy} - v_p({}^B_{IB}\hat{v}_d)_{xy}||$.
- Recompensa por elevación de patas (r_{ep}):

$$r_{ep} = \sum_{i \in I_{swing}} \chi_{\mathcal{F}_{clear}} / |I_{swing}|$$

  donde I_{swing} es el conjunto de patas en movimiento (en inglés swing), χ es la función característica y $\mathcal{F}_{clear}$ es el conjunto de patas cuya posición es mayor o igual en altura que los 8 puntos medidos a su alrededor.
- Recompensa por colisión (r_c):

$$r_c = -|I_{c,cuerpo} \setminus I_{c,patas}|$$

  es decir, sumamos las partes del robot que tienen una fuerza ejercida sobre ellas, exceptuando las patas.

- Recompensa por suavidad (r_s):

$$r_s = -\sum_i ||(r_{i,d})_t - 2(r_{i,d})_{t-1} + (r_{i,d})_{t-2}||$$

donde $(r_{i,d})_t$ es la posición deseada para la pata i-ésima en el instante t.

Los parámetros de la función de recompensa, incluyendo las constantes que acompañan cada término de la ecuación (1), se ajustaron manualmente para obtener mejores resultados. En función de los ensayos realizados, se definió usar $\alpha_{vl} = \alpha_{va} = 100$, $\alpha_b = 500$ y $v_{threshold} = \omega_{threshold} = 0{,}2$.

II-H. Entrenamiento mediante DRL y curriculum de terrenos

Para la red neuronal utilizamos la misma arquitectura propuesta en [7]. Sin embargo, el esquema de aprendizaje por refuerzo resultante no se entrenó utilizando el algoritmo Trust Region Policy Optimization (TRPO) [12] como sugiere el citado trabajo, sino que se utilizó el algoritmo Proximal Policy Optimization (PPO) [13]. PPO utiliza una aproximación de proximalidad para mejorar gradualmente la política sin hacer grandes cambios en cada paso de optimización mediante la aplicación de una restricción en la actualización de políticas para asegurar que no se desvíen demasiado de la política anterior. Por otro lado, TRPO utiliza un enfoque de región de confianza (*trust region*) para garantizar que las actualizaciones de políticas sean conservadoras y no introduzcan cambios drásticos que puedan desestabilizar el proceso de entrenamiento. Esto requiere calcular y resolver problemas de optimización más complejos y conlleva una mayor sensibilidad a los hiperparámetros. Por lo tanto, PPO tiende a ser más estable y computacionalmente menos costoso que TRPO, lo que lo hace que requiera tiempos de entrenamiento menor y resulte más escalable para entrenar en entornos de mayor tamaño y complejidad [13]. El pseudocódigo de entrenamiento completo, incluyendo el currículum de terrenos adaptativo, se muestra en el Algoritmo 1.

Algoritmo 1 Entrenamiento con currículum automático de terrenos

Inicializar la memoria de replay, muestrear $N_{particle}$ $c_{T,0}$s uniformemente de $\mathcal{C}$, $i, j = 0$.
while Todavía no hay convergencia **do**
  for $0 \leq k \leq N_{evaluate}$ **do**
    for $0 \leq l \leq N_{particle}$ **do**
      for $0 \leq m \leq N_{traj}$ **do**
        Generar un terreno usando $c^l_{T,j}$.
        Inicializar el robot.
        Correr la política π_i.
        Computar la traversibilidad para cada transición de estado.
        Guardar la trayectoria con sus valores de traversabilidad
      end for
    end for
    Actualizar la política con PPO.
    $i = i + 1$
  end for
  for $0 \leq l \leq N_{particle}$ **do**
    Computar la probabilidad de medición para cada parámetro $c^l_{T,j}$.
  end for
  for $0 \leq l \leq N_{particle}$ **do**
    Actualizar los pesos $\omega_j = \frac{P(y^l_i|c^l_{T,j})}{\sum_m P(y^m_i|c^m_{T,j})}$.
  end for
  Muestrear nuevamente $N_{particle}$ parámetros.
  Agregar los $c_{T,j}$s a la memoria de replay.
  for $0 \leq l \leq N_{particle}$ **do**
    Muestrear de la memoria de replay con probabilidad p_{replay}.
    Mover $c^l_{T,j}$ a un valor adyacente en $\mathcal{C}$ con probabilidad $p_t ransition$.
  end for
  $j = j + 1$
end while

III. RESULTADOS

En primer lugar, se entrenó el sistema propuesto en terreno plano con comando de velocidad lineal estrictamente hacia adelante, es decir, sin giros. El resultado de dicho entrenamiento puede verse en la Figura 5, donde se muestra el refuerzo por paso de entrenamiento. Como material suplementario se presenta un video con un episodio de caminata con la política obtenida. La caminata resultante mantiene leves irregularidades en la forma de mover las patas, pero es aceptable y capaz de trasladarse en la dirección deseada.

Una vez alcanzada una política para caminar en plano, se procedió a probar con y sin el curriculum de terrenos adaptativo. En el primer caso se entrenó con terrenos aleatorios con parámetros uniformemente distribuidos en todo el rango y en el segundo caso con el Filtro de Partículas. En la Figura 6 se pueden ver ambas curvas de aprendizaje, donde el resultado para el Filtro de Partículas se finalizó ni bien contamos con información suficiente para realizar la comparación. Se puede observar que el resultado con curriculum de terrenos

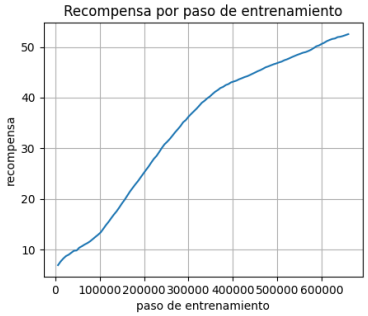

Figura 5: Curva de aprendizaje para terreno plano.

Figura 6: Curvas de aprendizaje para terreno plano utilizando el filtro de partículas para seleccionar los terrenos (celeste) y con terrenos aleatorios (azul).

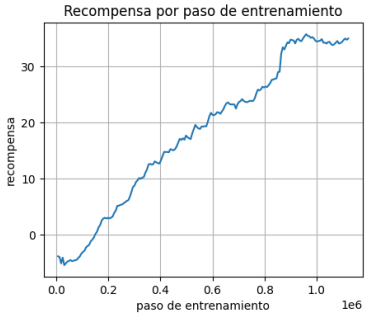

Figura 7: Curva de aprendizaje para terrenos irregulares.

adaptativo obtiene mayor recompensa final, y cualitativamente se observó que la política final obtenida es mejor.

Confirmada la utilidad del currículo de terrenos, se procedió a entrenar con éste al robot por un periodo de entrenamiento mayor. En la Figura 7 se puede ver la curva de aprendizaje obtenida. En el material suplementario se presentan videos con ejemplos de caminata con la política obtenida. Se observa que el robot logra aprender a caminar en los terrenos irregulares generados.

Tanto en terreno plano como en terrenos irregulares podemos observar que el robot alcanza una trayectoria subóptima, pero suficiente para poder transitar, con un movimiento de patas razonable, aunque no simétrico, ya que no hay nada en el esquema de aprendizaje por refuerzo ni en la función de recompensa utilizada que contemple la simetría del robot. Si observamos por varios segundos cómo el robot responde al comando de velocidad deseada, veremos que eventualmente se desvía de la dirección original, pero esto se debe a que el comando de velocidad se da a lazo abierto para el sistema global, es decir, ante una perturbación que corra al robot de su camino en una dirección éste continuará hacia adelante sin corregir que su postura se desvió de la orientación inicial. En un sistema que utilizara la política obtenida esto lo podría corregir el teleoperador o el sistema de planificación de trayectoria que sí tendrían disponible información de la posición global del robot.

IV. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se presentó un sistema de locomoción para robot hexápodo basado en aprendizaje por refuerzo profundo en simulación utilizando únicamente mediciones propiocépticas. El sistema obtenido pudo aprender a caminar en terrenos irregulares; la caminata final presenta ciertas limitaciones, pero logra transitar todo el terreno a pesar de ellas.

Una de las mayores dificultades encontradas durante la realización de este trabajo fue la demanda temporal del simulador para el entrenamiento. A pesar de las simplificaciones realizadas al sistema, el entrenamiento para un único comando de velocidad demanda alrededor de un día en una computadora con las siguientes prestaciones: Intel i7-5600U, 2.6 – 3.2GHz, dual-core 64-bit, 16GB de memoria RAM y una tarjeta gráfica NVIDIA GeForce RTX 3070 Ti. Esto se debe a que no se pudieron obtener velocidades de simulación mayores a real time en Gazebo Classic, y al requerirse miles de episodios para el entrenamiento el tiempo real escala a valores que impiden realizar pruebas eficientes, ajustar mejor los hiperparámetros y/o entrenar con más comandos de velocidad. Por lo tanto, una de las lineas de trabajo futuro consiste en explorar otras plataformas de simulación para el entrenamiento.

En cuanto al curriculum de terrenos adaptativo, se planea incorporar terrenos de tipo escalera, tal como en [7], así como ajustar más finamente los parámetros de los terrenos generados para obtener una variedad más acorde a los tipos de terrenos que se encontrará la política obtenida en su aplicación final. Por otro lado, como se mencionó previamente, la función de recompensa no contempla explícitamente la simetría del robot, ni tampoco la estabilidad estática y/o dinámica del mismo; estas también son cuestiones a abordar en un futuro.

El presente trabajo se enmarca en un objetivo más ambicioso que consiste en implementar un sistema de locomoción basado en aprendizaje maestro-estudiante en un robot hexápodo. Los resultados obtenidos presentan un primer paso en esta dirección. Una vez desarrollado el sistema completo maestro-estudiante, se evaluará la política obtenida con el estudiante en el robot real PhantomX Mark II.

V. AGRADECIMIENTOS

Este trabajo fue financiado parcialmente gracias al proyecto PIP 11220200101675CO otorgado por el Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina.

REFERENCIAS

[1] T. Li, N. Lambert, R. Calandra, F. Meier, and A. Rai, "Learning generalizable locomotion skills with hierarchical reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 413–419.

[2] W. Ouyang, H. Chi, J. Pang, W. Liang, and Q. Ren, "Adaptive locomotion control of a hexapod robot via bio-inspired learning," *Frontiers in Neurorobotics*, vol. 15, p. 627157, 2021.

[3] M. Schilling, K. Konen, F. W. Ohl, and T. Korthals, "Decentralized deep reinforcement learning for a distributed and adaptive locomotion controller of a hexapod robot," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5335–5342.

[4] T. Azayev and K. Zimmerman, "Blind hexapod locomotion in complex terrain with gait adaptation using deep reinforcement learning and classification," *Journal of Intelligent & Robotic Systems*, vol. 99, no. 3, pp. 659–671, 2020.
[5] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*. PMLR, 2023, pp. 403–415.
[6] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.
[7] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.
[8] Gazebo. [Online]. Available: https://gazebosim.org/
[9] Ros. [Online]. Available: https://www.ros.org/
[10] Stable baselines 3. [Online]. Available: https://stable-baselines3.readthedocs.io/en/master/
[11] Phantomx urdf. [Online]. Available: https://github.com/HumaRobotics/phantomx_gazebo.git
[12] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.
[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.