# AGENTE HUMANOIDE EN MUJOCO

**Profesor:**
Julio Godoy

# Integrantes

Mirko Peñailillo

Javier Reyes

Mario Salgado

# DESCRIPCIÓN DEL PROYECTO

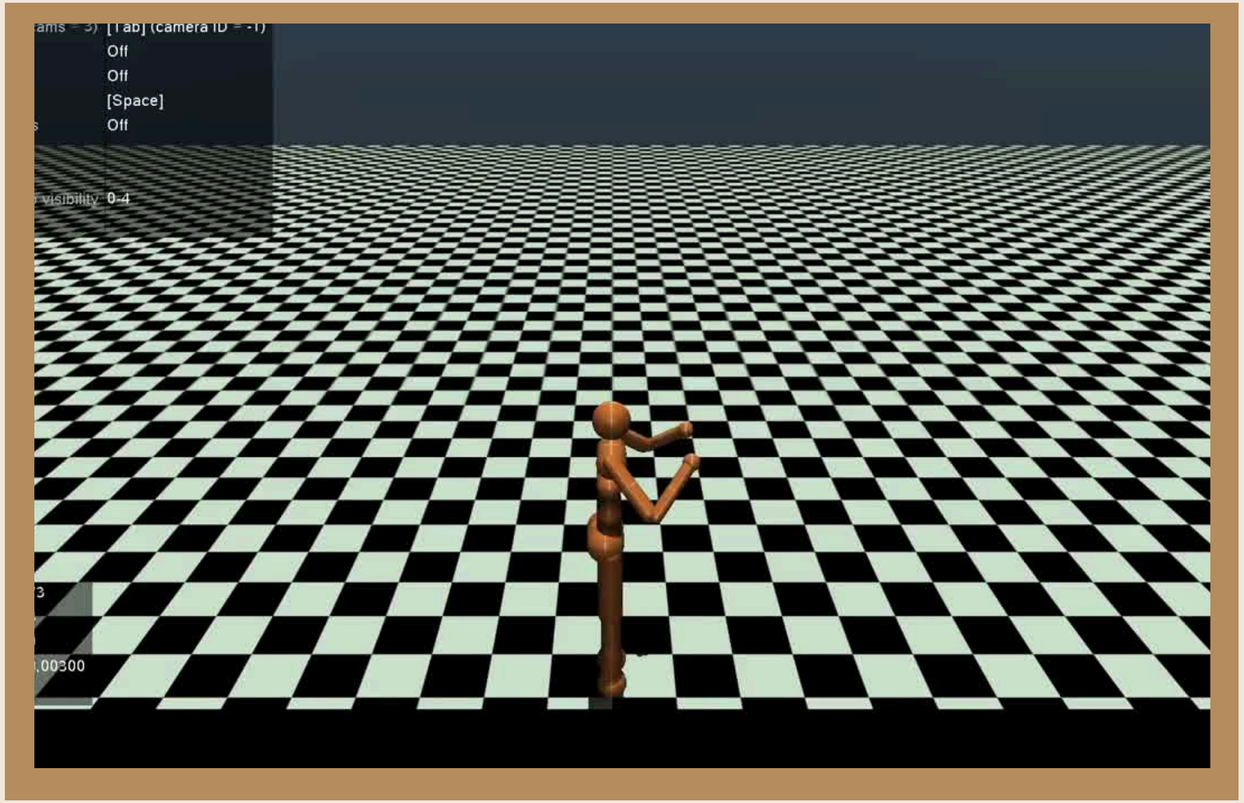
**Comparación de agentes capaces de controlar un humanoide en el simulador MuJoCo, centrado en tres habilidades:**

**01** Mantenerse de pie/equilibrio.

**02** Aprender a caminar correctamente a una velocidad deseada.

**03** Aprender a moverse en un entorno con obstáculos.

# INICIO GENERAL



# MÉTODOS IMPLEMENTADOS

| | |
|---|---|
| Proximal Policy Optimization (PPO) | Estabilidad y facilidad de entrenamiento. |
| Soft Actor-Critic (SAC) | Buen desempeño y capacidad de exploración. |
| Twin Delayed Deep Deterministic Policy Gradient (TD3) | Eficiencia y estabilidad. |

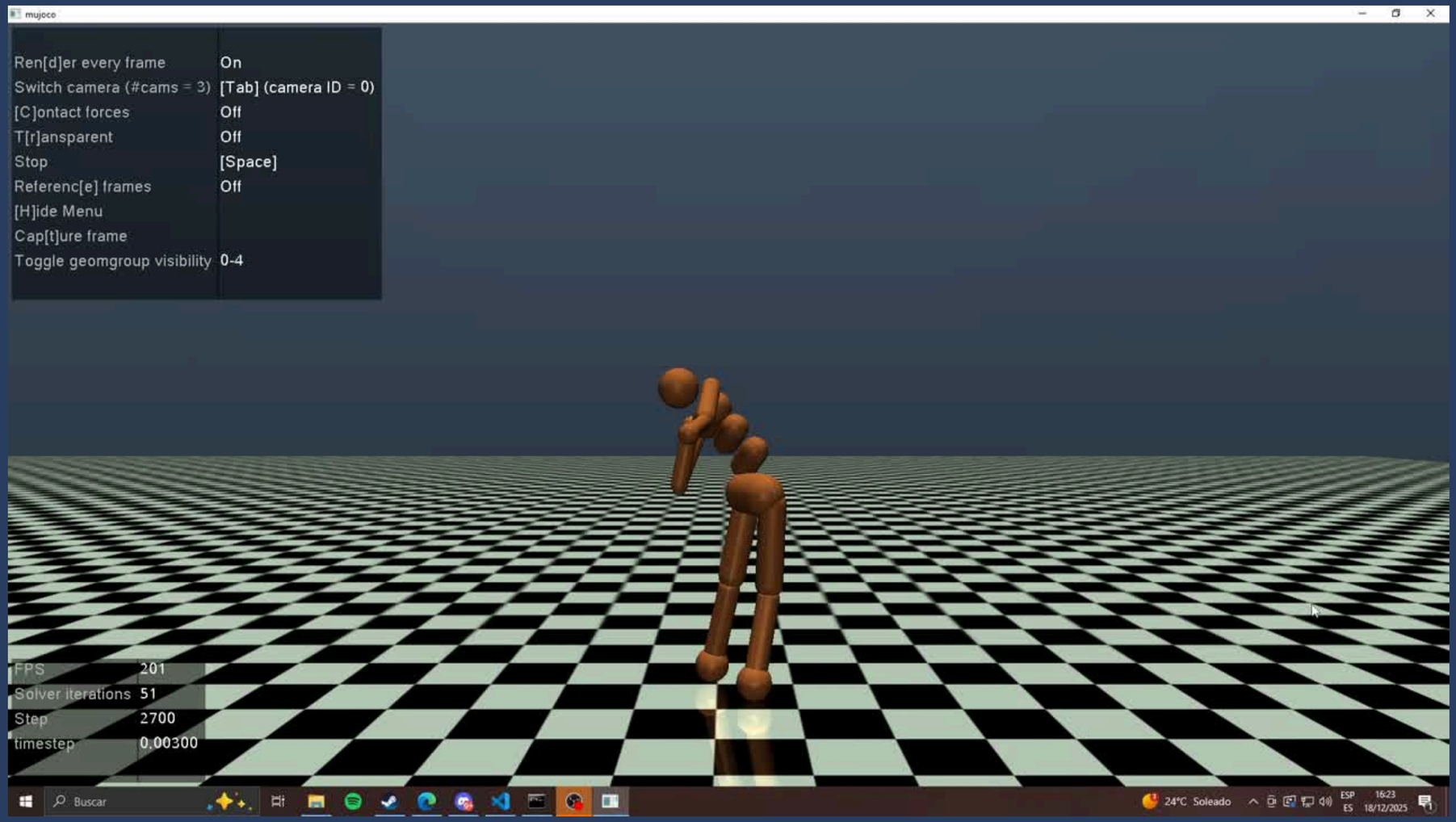
# DETALLES PPO

## DETALLES DE IMPLEMENTACIÓN

1. Arquitectura Actor-Critic
   - Redes Neuronales: Dos redes independientes.
   - Espacio de Acciones: Implementación de una distribución Tanh-Gaussiana.
2. Estabilización del Aprendizaje
   - Clipping: Uso de una función de pérdida por ratio (epsilon=0.2) para evitar actualizaciones que alteren drásticamente la política.
   - Generalized Advantage Estimation (GAE): Balance entre sesgo y varianza mediante parámetros gamma=0.995 y lambda=0.95 para una estimación precisa de recompensas a largo plazo.
3. Optimización y Eficiencia
   - Vectorización Paralela: Uso de AsyncVectorEnv para recolectar experiencias de múltiples ambientes MuJoCo simultáneamente.
   - LR Scheduling: Programador de tasa de aprendizaje con fase de Warmup (10 actualizaciones) seguida de un decaimiento de coseno.
   - Early Stopping: Monitoreo de la divergencia KL Aproximada para detener épocas de entrenamiento si el cambio en la política supera el umbral de seguridad (0.01).

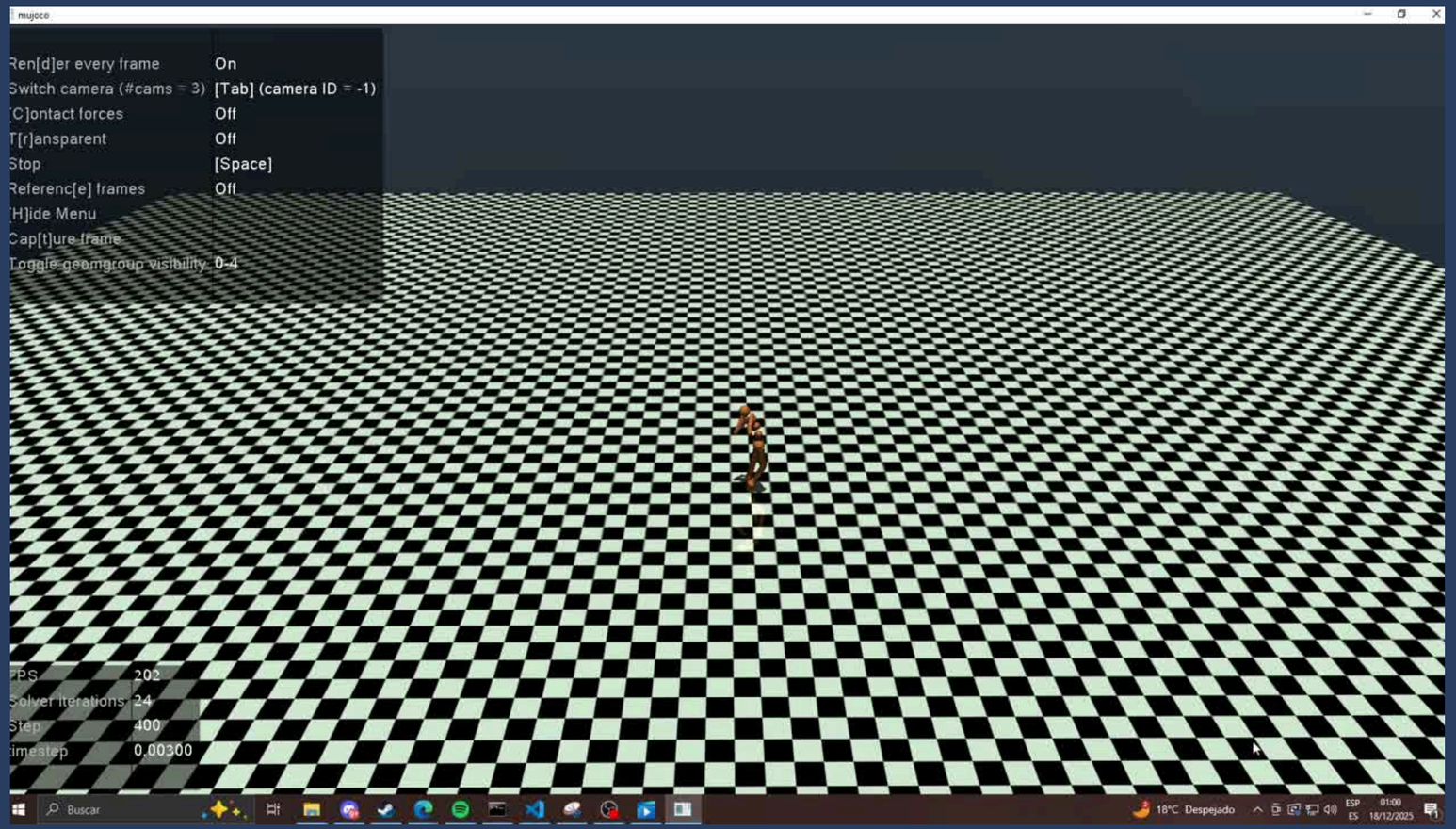
# RESULTADOS PPO

## BASE (DE CERCA)



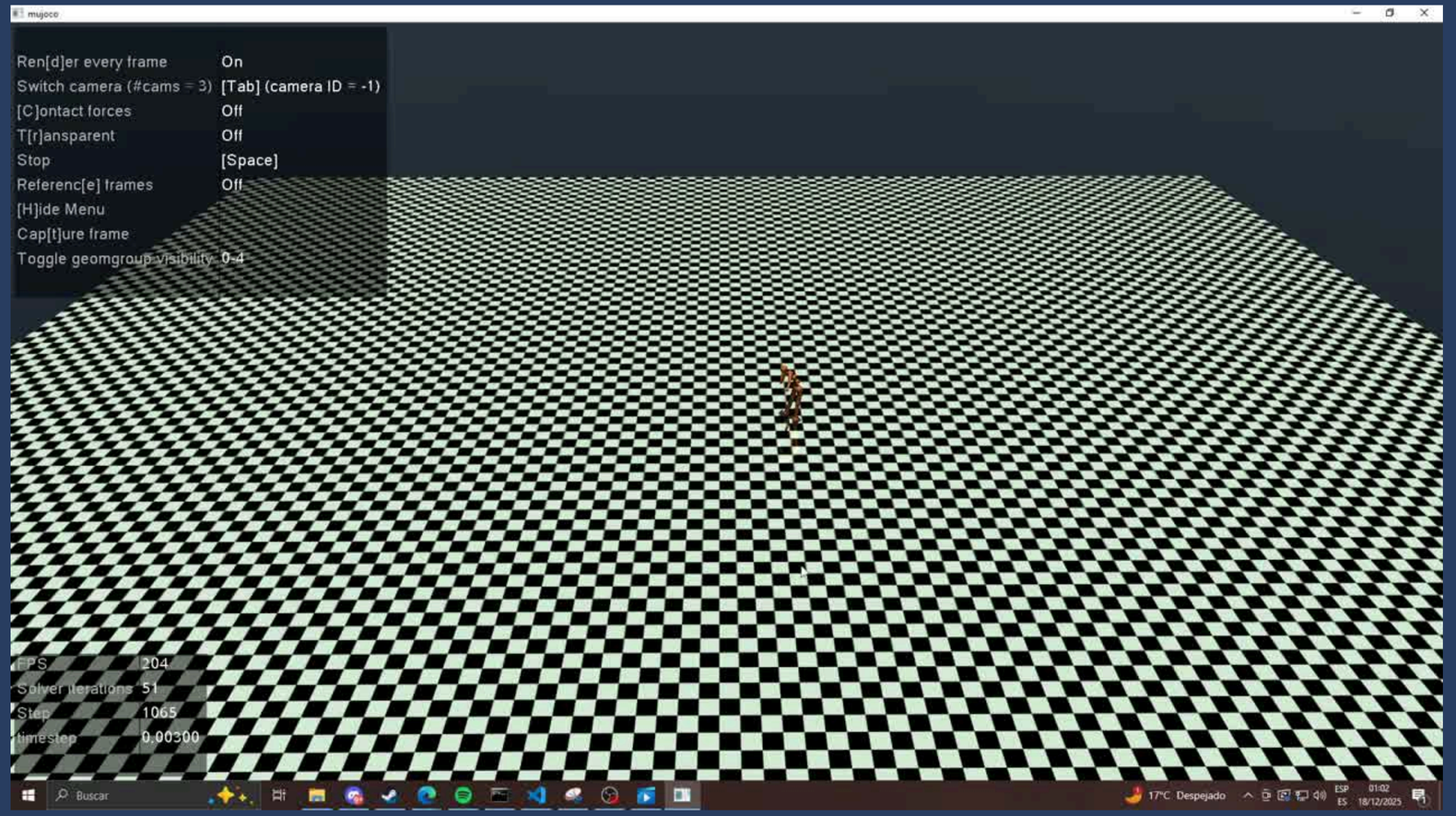
# RESULTADOS PPO

## BASE EN AMBIENTE ROBUSTEZ



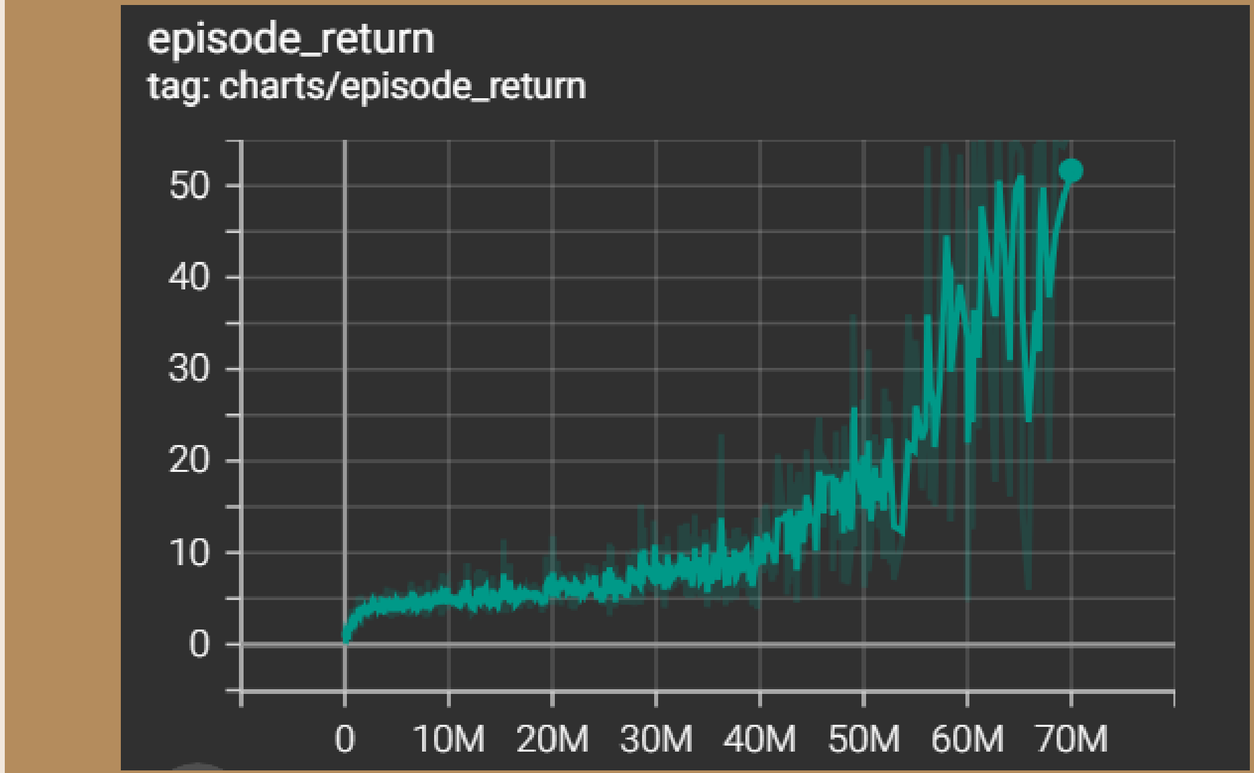
# RESULTADOS PPO

## AGENTE ROBUSTEZ



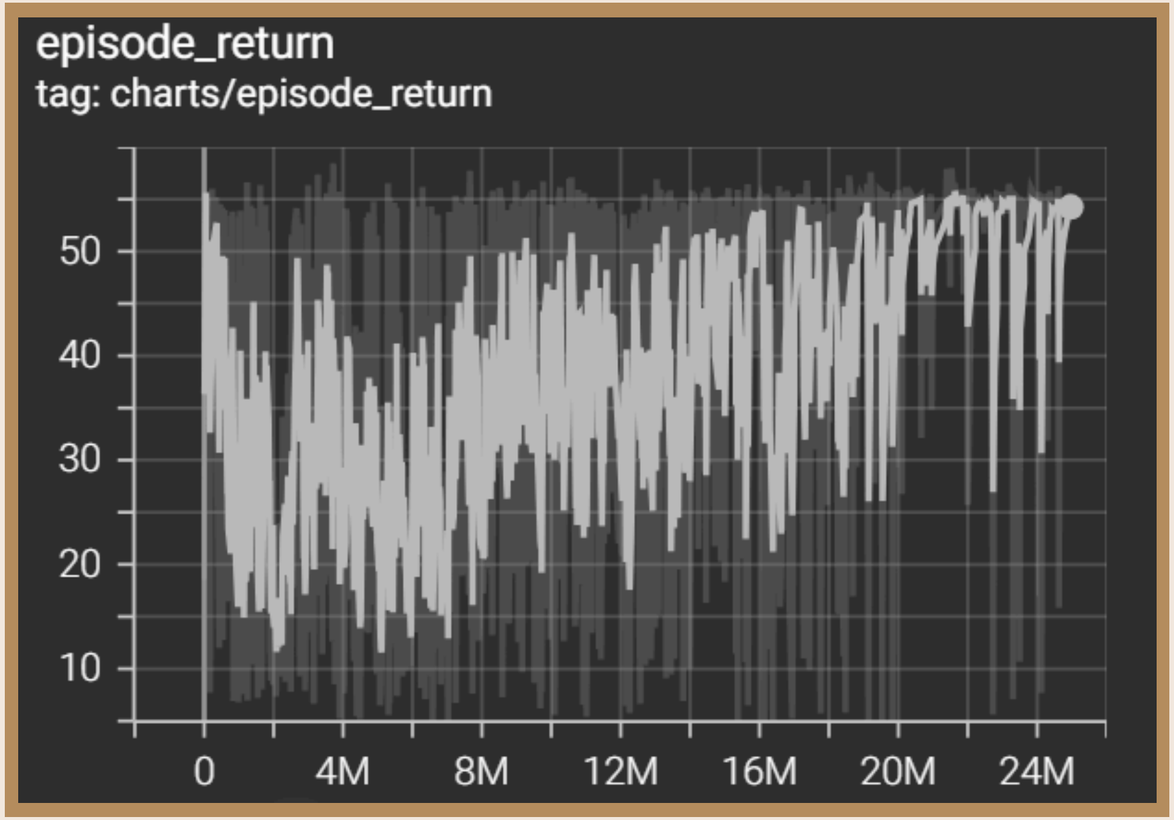
# ANALISIS DE RESULTADOS

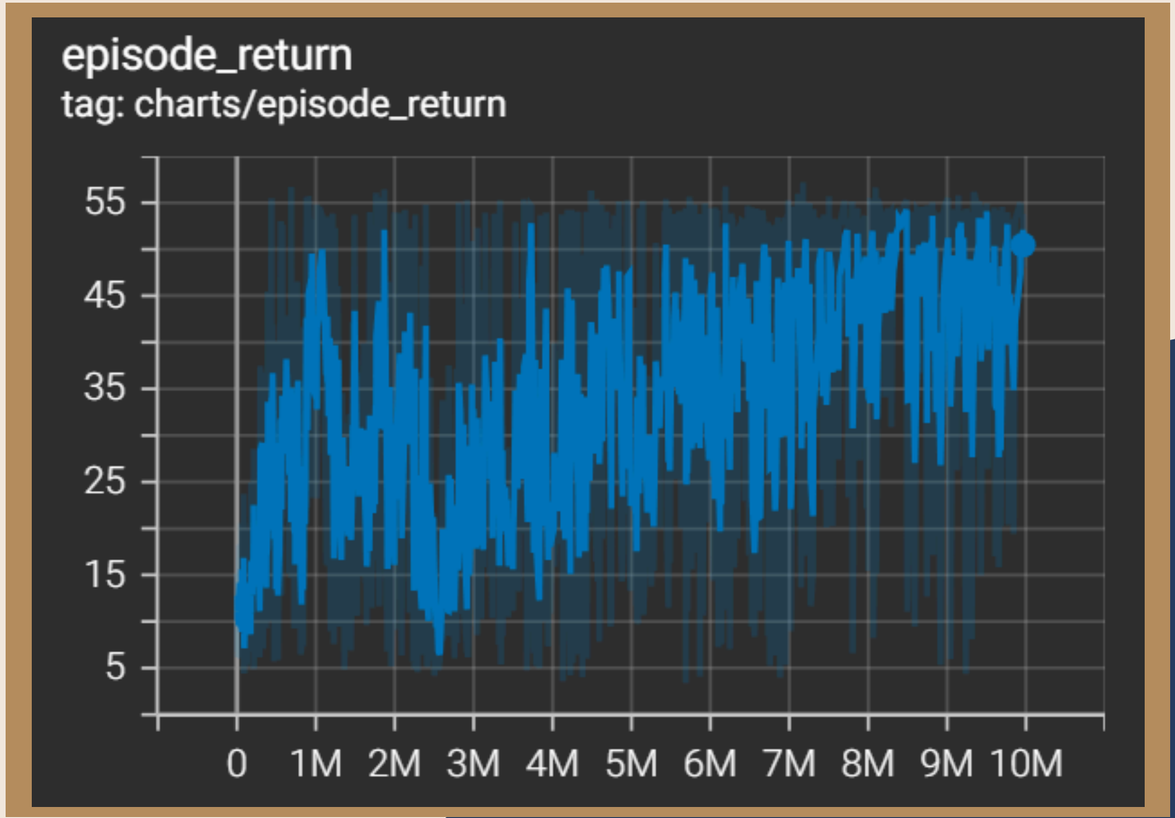
PPO



# ANALISIS DE RESULTADOS

PPO



# DETALLES SAC

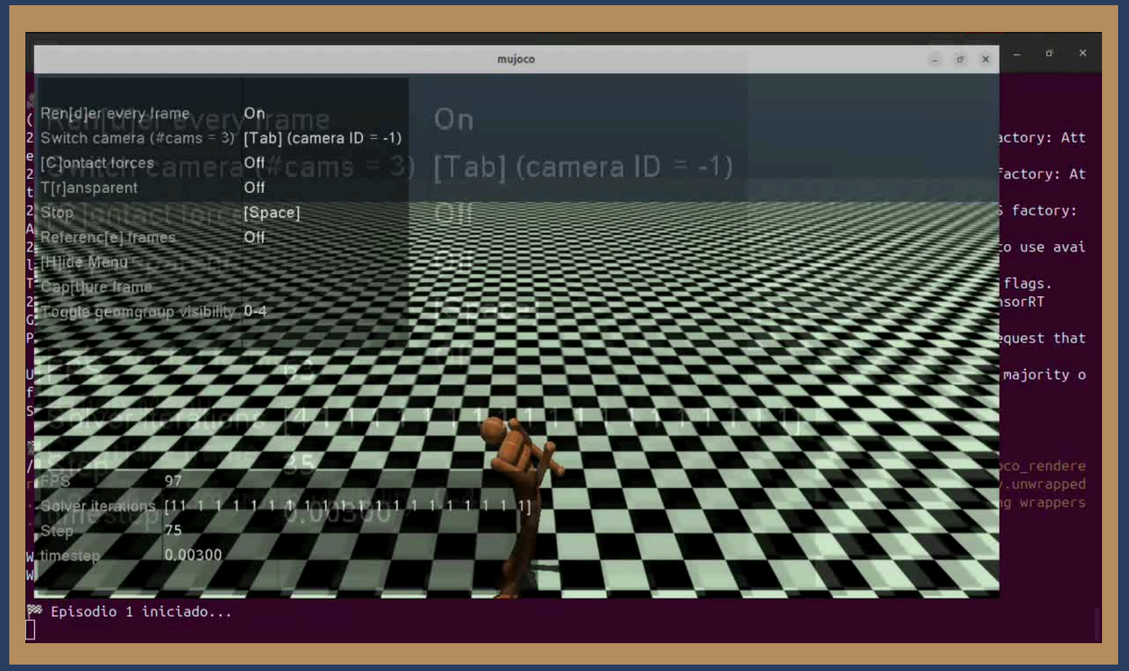
## DETALLES DE IMPLEMENTACIÓN

1. Arquitectura Actor-Critic
   - Doble Crítico (Twin Q-Networks): Uso de dos redes independientes para mitigar el sesgo de sobreestimación. Al tomar el valor mínimo entre ambas, el agente evita decisiones "demasiado optimistas" que causarían la caída del robot.
   - Espacio de Acciones Tanh-Gaussiana: . El agente genera una distribución  y la acota mediante una función tanh, asegurando que los movimientos de las 17 articulaciones respeten los límites físicos del simulador.
2. Estabilización del Aprendizaje:
   - Ajuste de entropía: La entropía actúa como instinto, se automatiza el  ajuste. Esto permite que el propio agente decida cuándo necesita explorar más (porque está fallando en su equilibrio) y cuándo debe ser preciso (porque ya ha encontrado una trayectoria estable).
   - Soft Updates : Se utiliza un factor de suavizado $\tau$=0.005 para actualizar las redes target de forma asíncrona, evitando cambios bruscos.
3. Optimización y Eficiencia
   - Replay Buffer : almacena un millón de experiencias, utilizando una máscara binaria para distinguir entre caídas  y límites de tiempo. Esta lógica permite penalizar la muerte y preservar el valor de los estados exitosos, priorizando el instinto de supervivencia.
   - Normalización y Vectorización: Se integra VecNormalize para estandarizar las observaciones y recompensas en tiempo real. Esto escala los 376 estados del Humanoid a un rango manejable, acelerando la convergencia.

# RESULTADOS SAC

## Balance

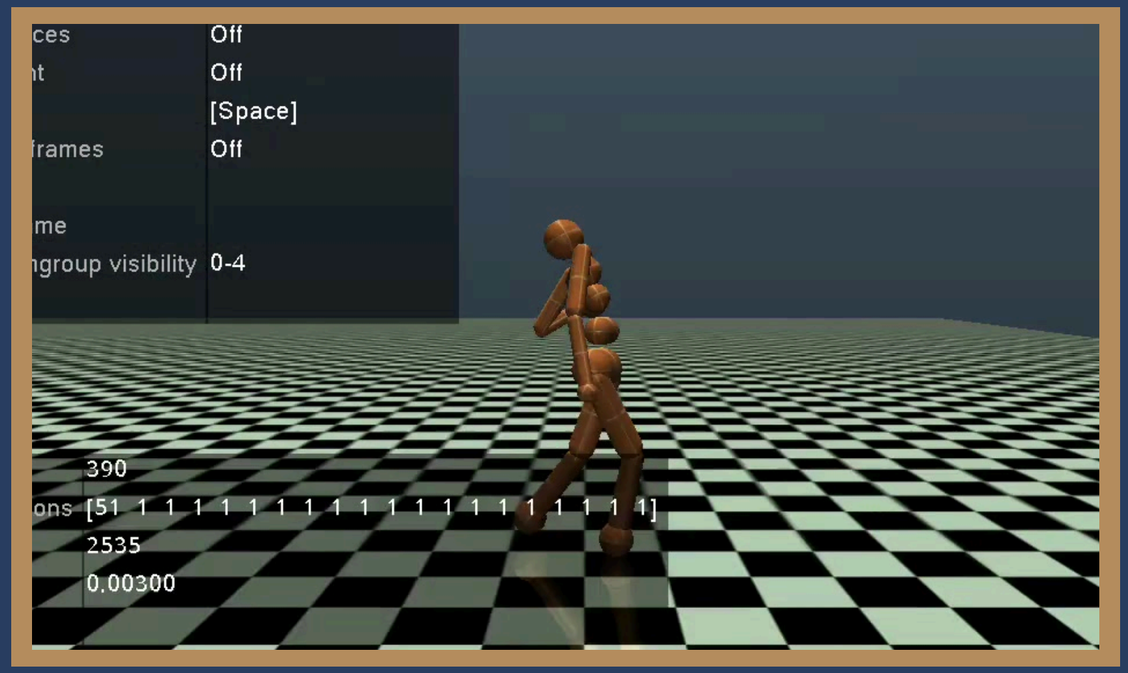
- En 400 mil pasos, el agente comenzó a lograr mantener el equilibrio y desplazarse con dificultad.



# RESULTADOS SAC

## Caminar

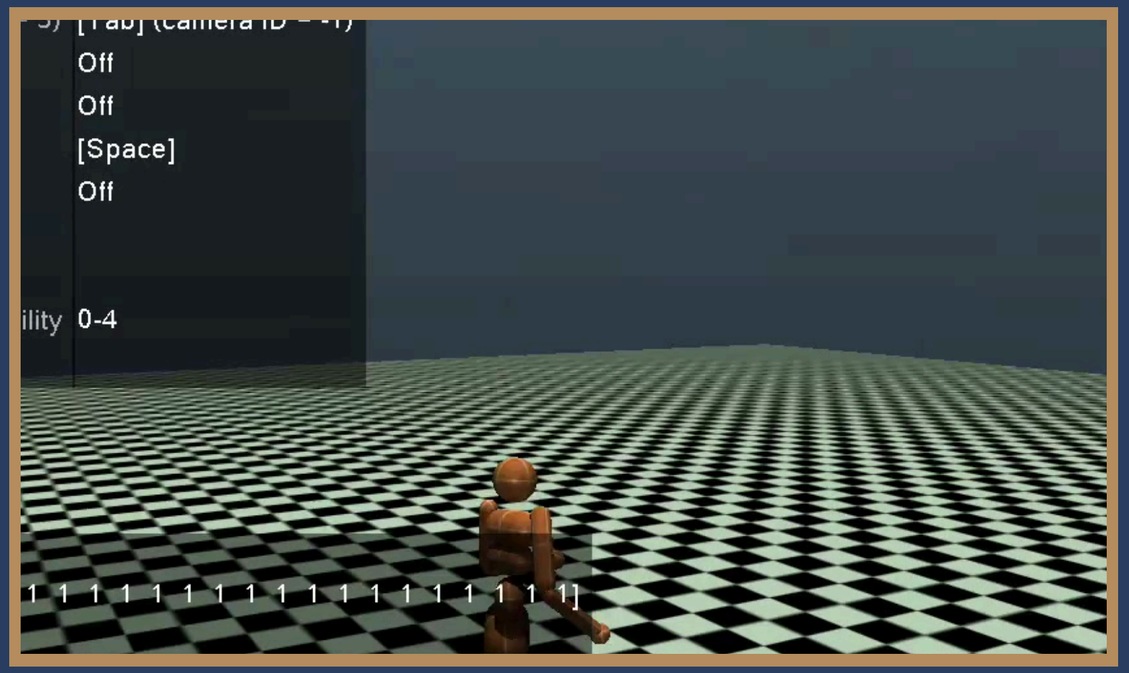
- En 1.6M pasos, el agente logra caminar de una forma más coordinada y estable.



# RESULTADOS SAC

## Robustez

- Enfrentamos el modelo anterior a la adversidad de viento, para validar robustez. Pese a que logramos desplazamiento y un par de pasos, notamos que logra derribarlo. Se espera que incorporar el viento al entrenamiento implique la robustez necesaria para superar esta robustez.



# ANALISIS DE RESULTADOS

## SAC

1. Recompensa Media

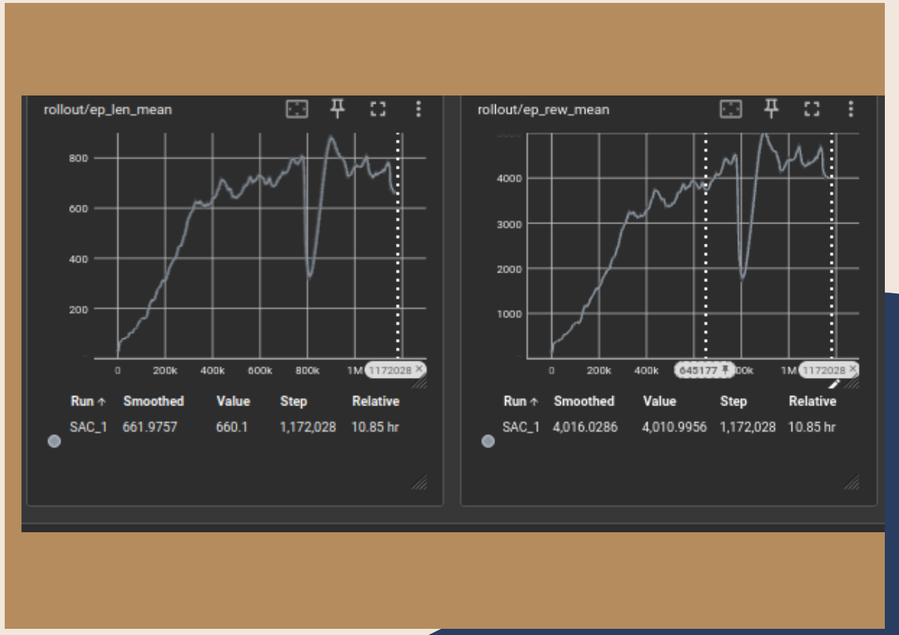
Progreso: Comienza en 0 y sube de forma constante

2. Longitud del Episodio

Sigue una curva casi idéntica a la recompensa, llegando a durar cerca de 800 pasos por episodio. El agente obtiene más puntos porque sobrevive más tiempo.

3. El Colapso

Caída brusca y profunda cerca del paso 900k donde la recompensa cae de 4,000 a casi 2,000.
Esto es común en SAC. A veces, la política se vuelve demasiado "confiada" o el coeficiente de entropía baja demasiado, haciendo que el agente intente una maniobra arriesgada que sale mal.



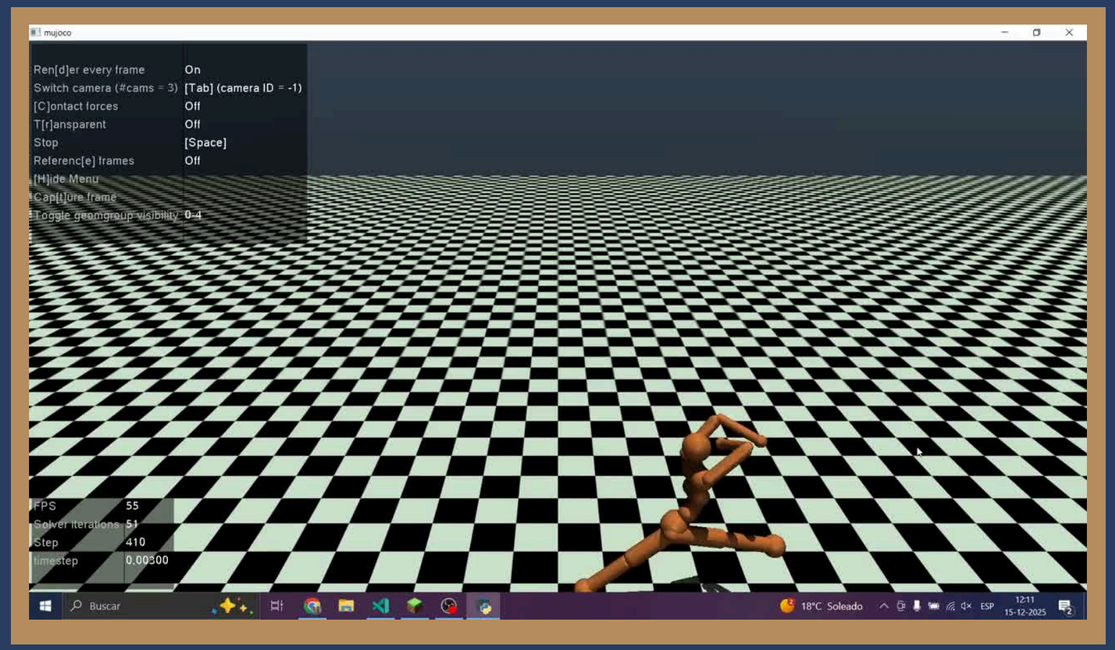
# DETALLES TD3

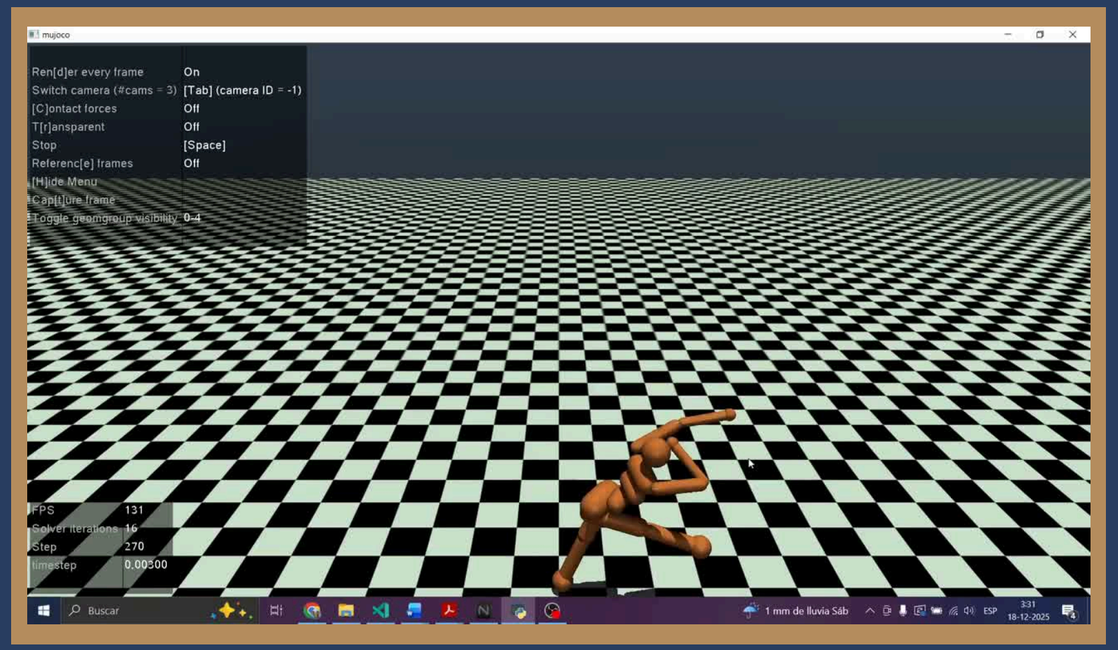
## DETALLES DE IMPLEMENTACIÓN

1. Arquitectura Actor–Critic
   - Actor determinístico con acciones continuas (tanh).
   - Dos críticos independientes ($Q_1$, $Q_2$) para reducir sobreestimación.
2. Estabilización del Aprendizaje
   - Clipped Double Q-Learning (min($Q_1$, $Q_2$)).
   - Target Policy Smoothing con ruido gaussiano acotado.
   - Delayed Policy Updates para mayor estabilidad.
3. Entrenamiento y Entorno
   - Curriculum Learning: balance → walk.
   - Reward shaping basado en avance (dx) y penalizaciones por retroceso, saltos y rotación del torso.
   - Detección explícita de contacto pie–suelo  con recompensa por apoyo estable y alternancia.
   - Grace period y control de terminación para evitar mínimos locales.

# RESULTADOS TD3

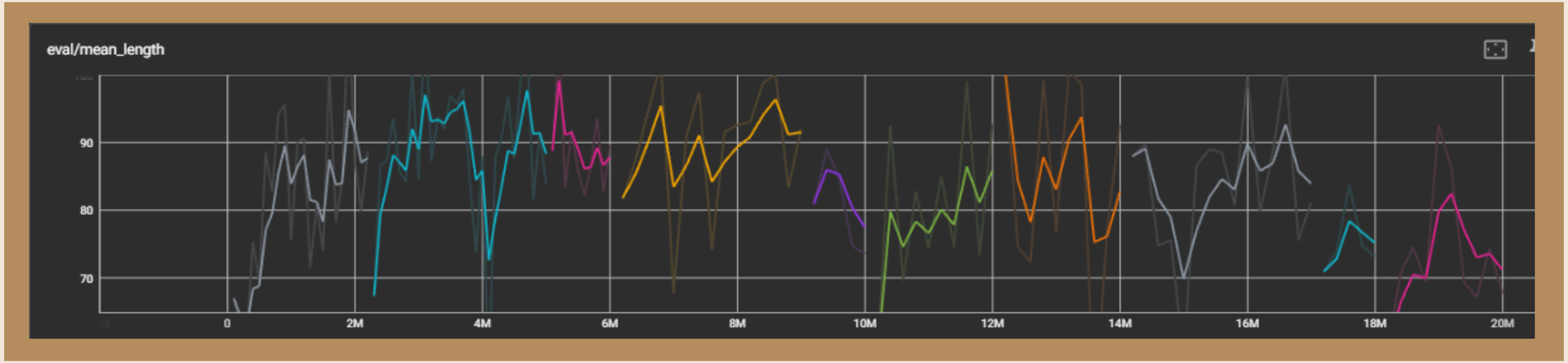
## BALANCE + CAMINAR

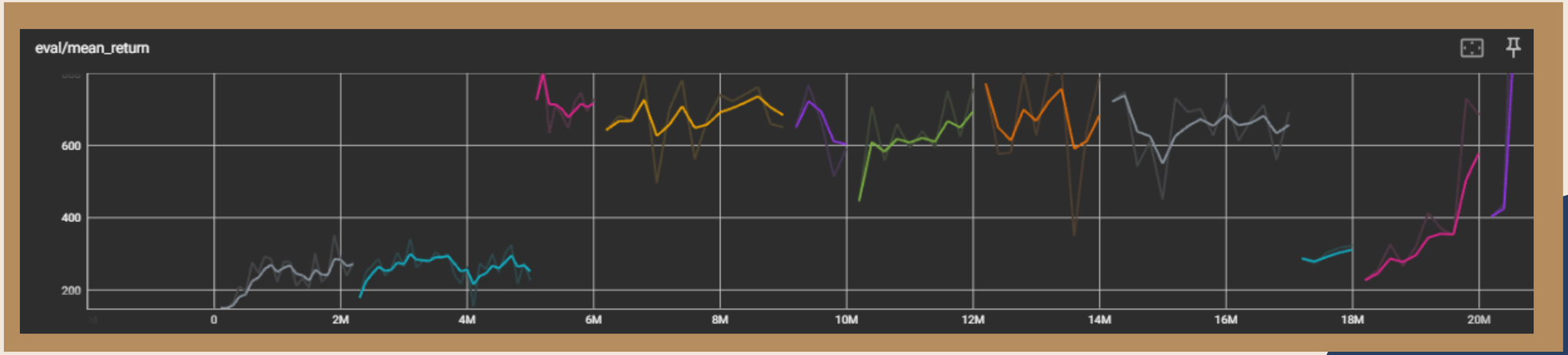


# ANALISIS DE RESULTADOS

TD3



# A P R E N D I Z A J E S

**01** No existe un único algoritmo óptimo: cada método presenta ventajas y limitaciones según la tarea.

**02** La complejidad del control continuo hace necesario un entrenamiento progresivo y un monitoreo constante del comportamiento del agente.

**03** Más allá de obtener altos valores de recompensa, es fundamental visualizar el comportamiento del humanoide para detectar y evitar soluciones no deseadas.

# CONCLUSIÓN

Para cerrar, nuestra principal conclusión es que PPO resultó ser el algoritmo más robusto para la locomoción humana, pero tenemos teorías claras de por qué sucedió esto.

Primero, mientras que SAC es más rápido al principio, sufre de inestabilidad al final. En nuestras gráficas vimos un colapso en SAC cerca del paso 900k; esto ocurre porque, a diferencia de PPO, SAC no tiene un 'freno de emergencia' como el Early Stopping por KL Divergence que detenga una actualización si el cambio es demasiado brusco.

Segundo, el control fino. El humanoide tiene 17 articulaciones muy sensibles. El código de PPO que implementamos utiliza un Scheduler de Coseno, lo que significa que al final del entrenamiento los pasos de aprendizaje son minúsculos, permitiendo al robot aprender equilibrio fino. SAC, con un Learning Rate fijo, a veces 'patea' los pesos de la red demasiado fuerte incluso cuando ya está cerca del óptimo.

Finalmente, aprendimos que la robustez se entrena, no se elige. Ningún algoritmo, por muy avanzado que sea, pudo superar el viento si no fue entrenado con él. La combinación de la estabilidad de PPO con entornos de entrenamiento adversos fue la clave del éxito en este proyecto.

Estudio Shonos

# MUCHAS GRACIAS

# Referencias


https://github.com/ProfessorNova/PPO-Humanoid/

https://github.com/xbpeng/DeepMimic/tree/master

https://github.com/DLR-RM/stable-baselines3