Generación de trayectorias robóticas mediante aprendizaje profundo por refuerzo

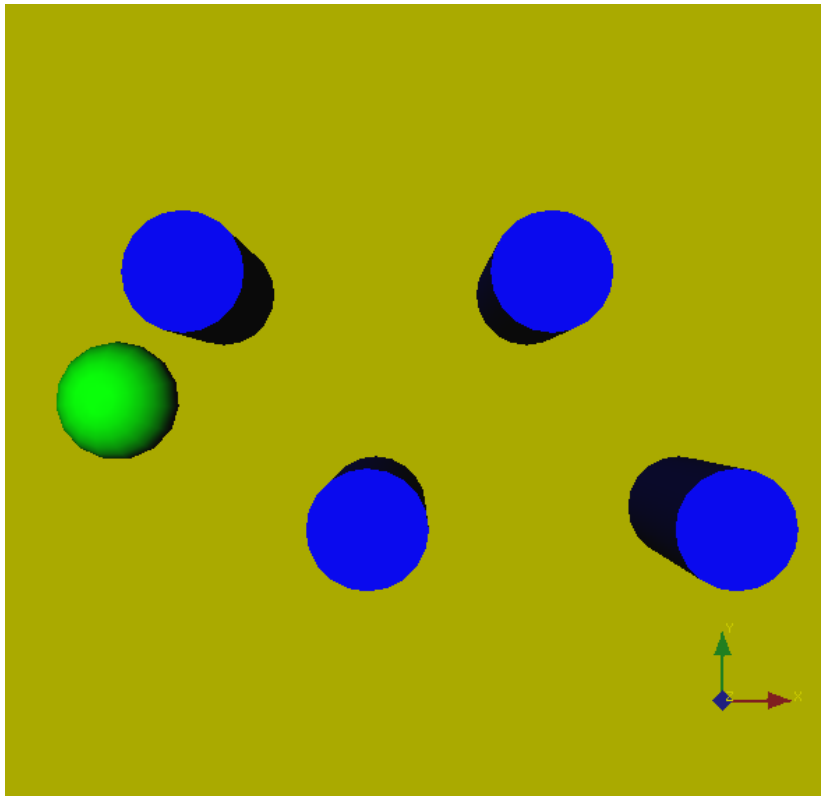
MAYO 2018

GUILLERMO URCERA MARTÍN

DIRECTOR: JAN ROSELL GRATACÒS


ABB
ABB
ORIGINAL TASTE
Coca-Cola
ORIGINAL TASTE
Coca-Cola
ORIGINAL TASTE
Coca-Cola
ORIGINAL TASTE
Coca-Cola
Estrella
Damm

Problema objetivo



Elemento	Descripción
Dimensiones del área de trabajo	19×38cm
Dimensiones de los obstáculos	Cilindros de $11,5$cm de altura, $6,6$cm de diámetro
Dimensiones del TCP	Esfera de 7cm de diámetro

Índice

Objetivos del trabajo

Implementar un sistema de planificación de movimientos utilizando el algoritmo DDPG

Comparar el rendimiento del sistema con otro planificador en simulación

Implementar el sistema en un robot real

Planificación de movimientos

Enfoque clásico:

- Métodos basados en la descomposición por celdas
- Métodos basados en campos de potenciales
- Métodos basados en mapas de ruta

Métodos basados en el muestreo

- Probabilistic Road Maps (PRM)
- Rapidly-exploring Random Trees (RRT)
- Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (KPIECE)

Problemas de los planificadores convencionales

Alto coste computacional / temporal

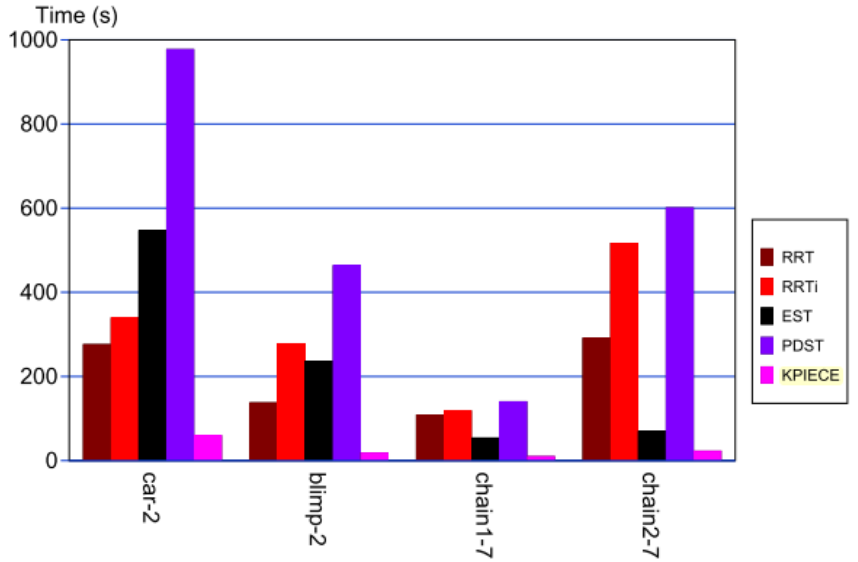
Curse of dimensionality

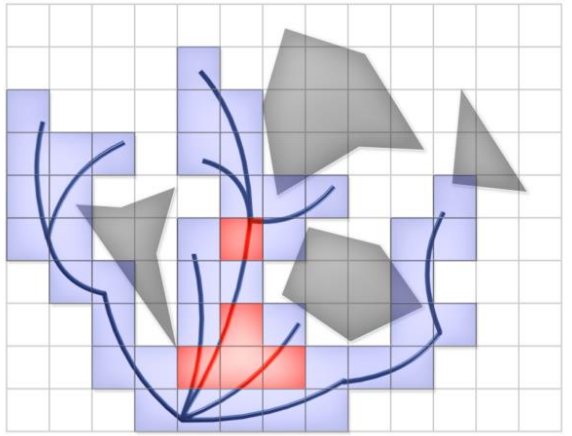
Precisión subordinada a una simulación física

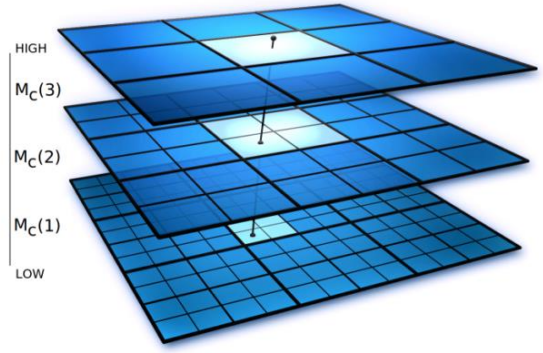
Superficies del espacio de configuraciones complejas

Kinodynamic Motion Planning by Interior-Exterior Cell Exploration (KPIECE)

- Basado en el muestreo
- Tiempos de cálculo y uso de memoria reducidos
- Planificación con simulación física
- Exploración eficaz:
  - Proyecciones a espacios de menor dimensionalidad
  - Discretización a varios niveles







Aprendizaje automático

Las redes neuronales permiten escapar del *curse of dimensionality*

Las redes neuronales permiten generalizar

Mayor adaptabilidad al entorno

- Parámetros iniciales
- Cambios

Algoritmo

Introducción
- Objetivos
- Motivación

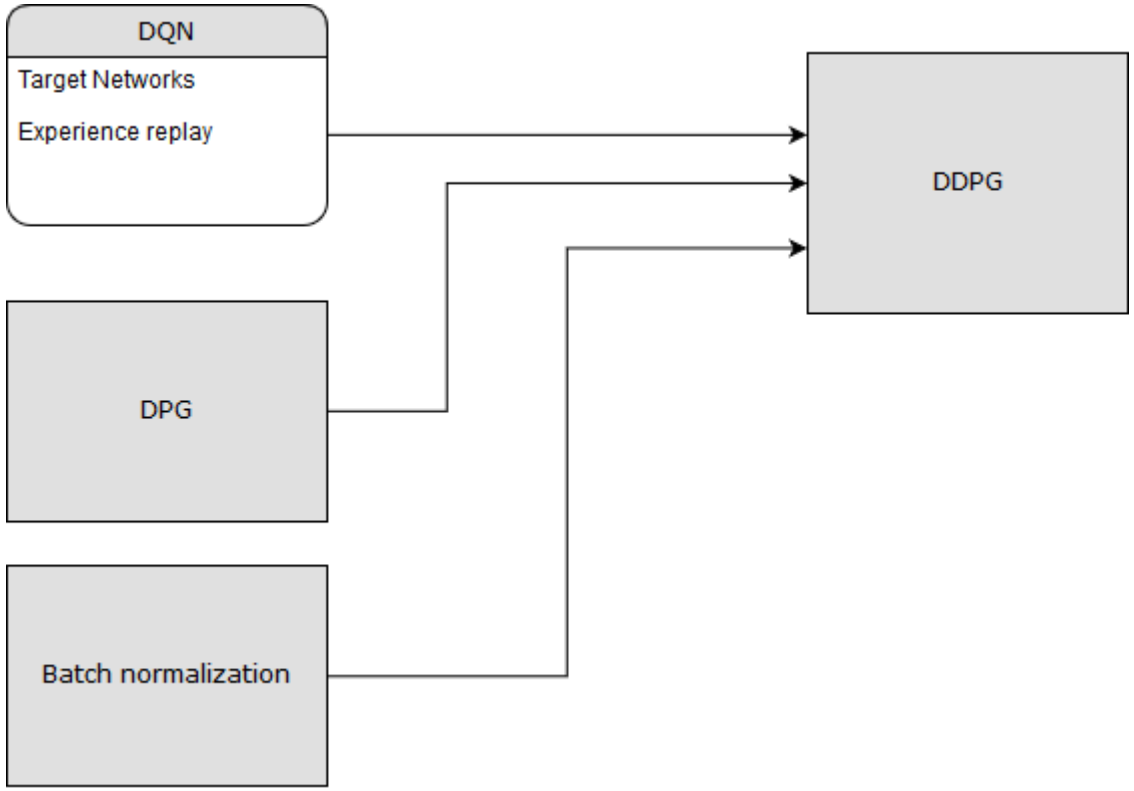
Algoritmo
- Antecedentes
- Puntos clave
- Núcleo

Implementación
- Entornos
- Arquitectura
- Detalles
- Experimentos

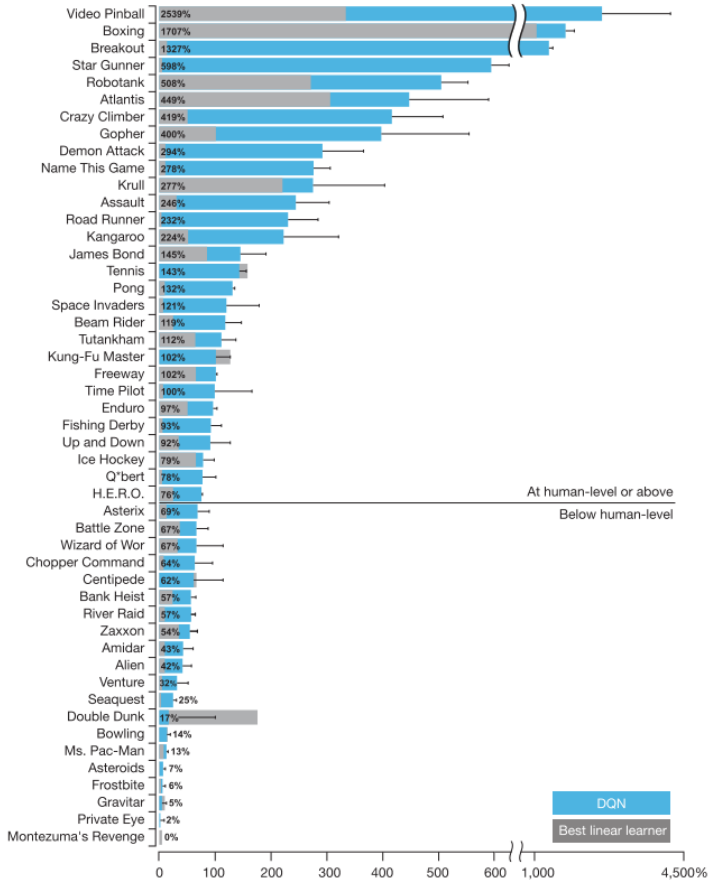
Resultados
- Pruebas con modificaciones
- Comparativa con KPIECE

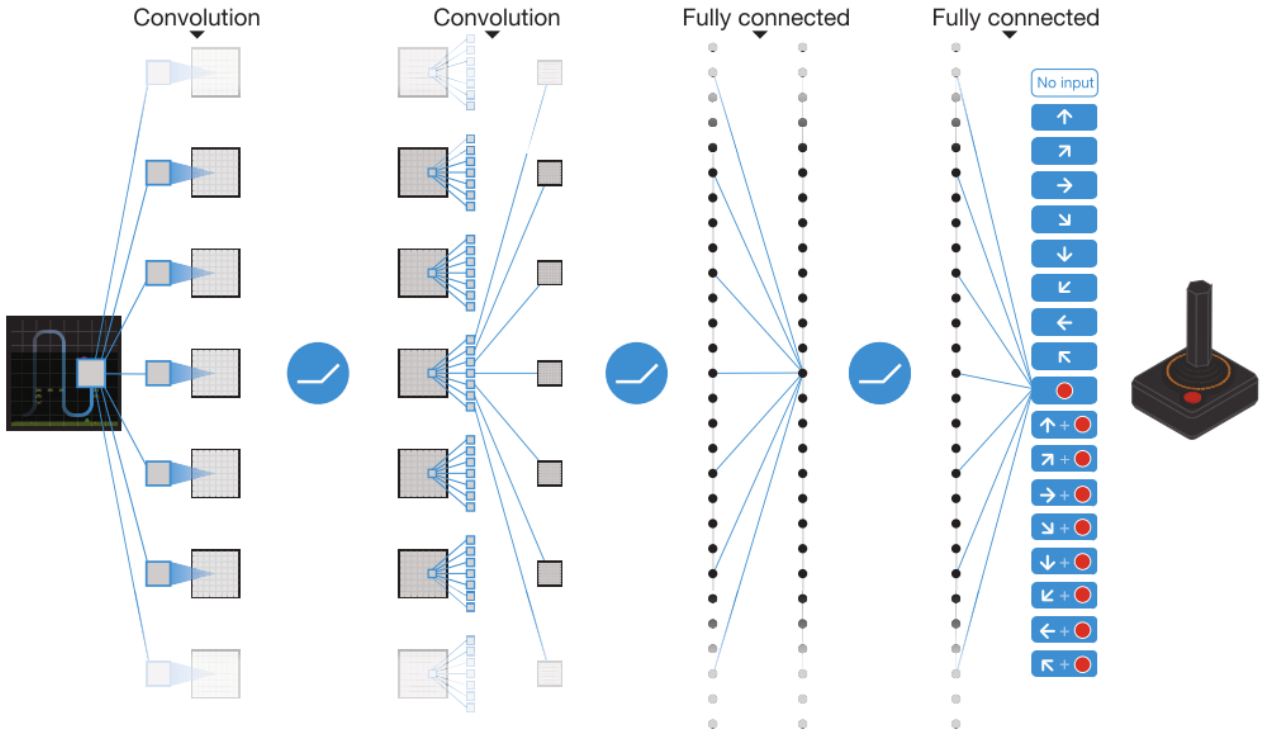
Conclusiones y trabajo futuro

Deep Deterministic Policy Gradient (DDPG)



Deep Q Network (DQN)





Deterministic Policy Gradient (DPG)

El gradiente de la política solo debe integrar sobre el espacio de estados

Rendimiento elevado en comparación con los gradientes de políticas estocásticas

$$\nabla_\theta J(\mu_\theta) = \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a) \mid_{a=\mu_\theta(s)} ds$$
$$= \mathbb{E}_{s\sim\rho^\mu}[\nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a) \mid_{a=\mu_\theta(s)}]$$

$$\nabla_\theta J(\pi_\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) \mathrm{d}a\mathrm{d}s$$
$$= \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)]$$

Puntos clave

Target networks

- Estabiliza el aprendizaje

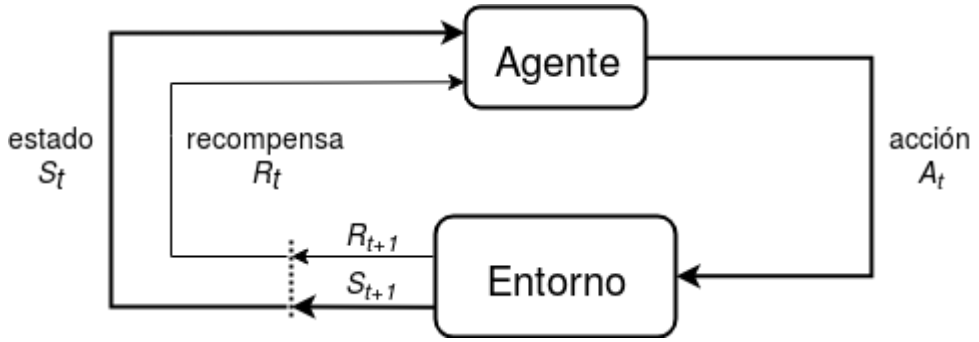
Experience replay

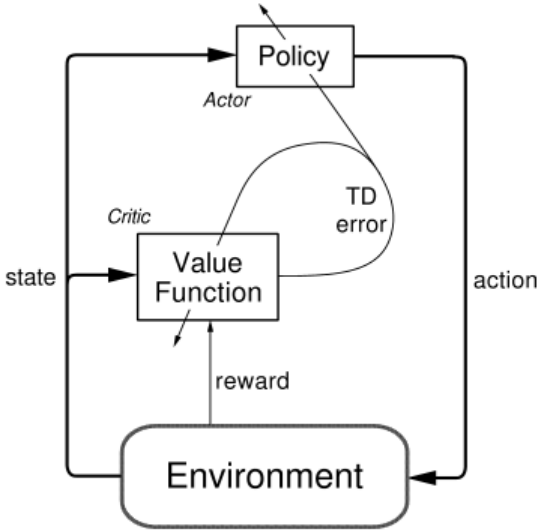
- Acelera el proceso de aprendizaje
- Aumenta la eficiencia de la exploración
- Estabiliza el aprendizaje

Batch normalization

- Reduce el *covariate shift*
- Acelera el aprendizaje

Deep Deterministic Policy Gradient (DDPG)





Algoritmo 1: Algoritmo DDPG[26]

Inicializar aleatoriamente la red del crítico $Q(s, a \mid \theta^{\mu})$ y el actor $\mu(s \mid \theta^{\mu})$ con pesos θ^{Q} y θ^{μ}.
Inicializar las redes objetivo Q' y μ' con pesos $\theta^{Q'} \leftarrow \theta^{Q}, \theta^{\mu'} \leftarrow \theta^{\mu}$
Inicializar el *buffer* de interacciones R
for *episodio=1, M* **do**
  Inicializar proceso aleatorio N para la exploración del espacio de acciones
  Recibir estado observacional inicial s_1
  for *t=1, T* **do**
    Seleccionar acción $a_t = \mu(s_t \mid \theta^{\mu}) + N_t$ de acorde con la política actual y el ruido exploratorio
    Ejecutar acción a_t y observar la recompensa r_t y el nuevo estado s_{t+1}
    Almacenar transición (s_t, a_t, r_t, s_{i+t}) en R
    Muestrear un *minibatch* aleatorio de N transiciones (s_i, a_i, r_i, s_{i+1}) de R
    Sea $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'})$
    Actualizar el crítico minimizando la pérdida:
    $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i \mid \theta^{Q}))^2$
    Actualizar la política del actor utilizando el gradiente de la política muestreado:
    $\nabla_{\theta^{\mu}} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a \mid \theta^{Q}) \mid_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu}) \mid_{s_i}$
    Actualizar las redes objetivo:
    $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$
    $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
  end
end

Deep Deterministic Policy Gradient (DDPG)

Implementación

Introducción
- Objetivos
- Motivación

Algoritmo
- Antecedentes
- Puntos clave
- Núcleo

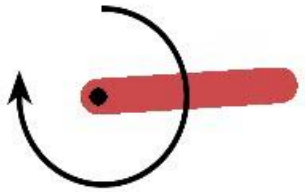
Implementación
- Entornos
- Arquitectura
- Detalles
- Experimentos

Resultados
- Pruebas con modificaciones
- Comparativa con KPIECE

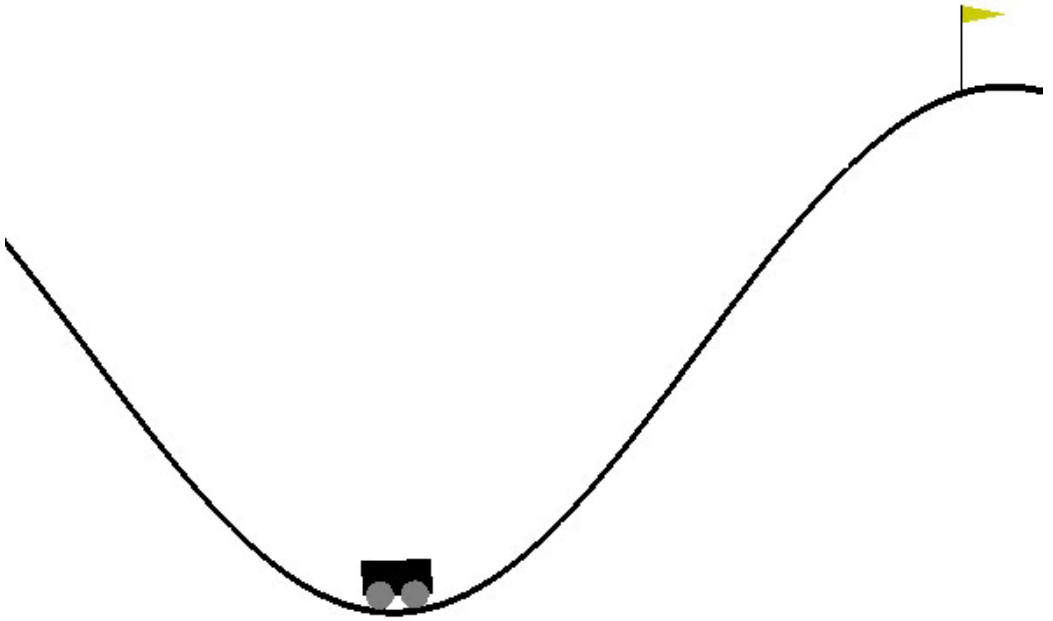
Conclusiones y trabajo futuro

Implementación en problemas sencillos

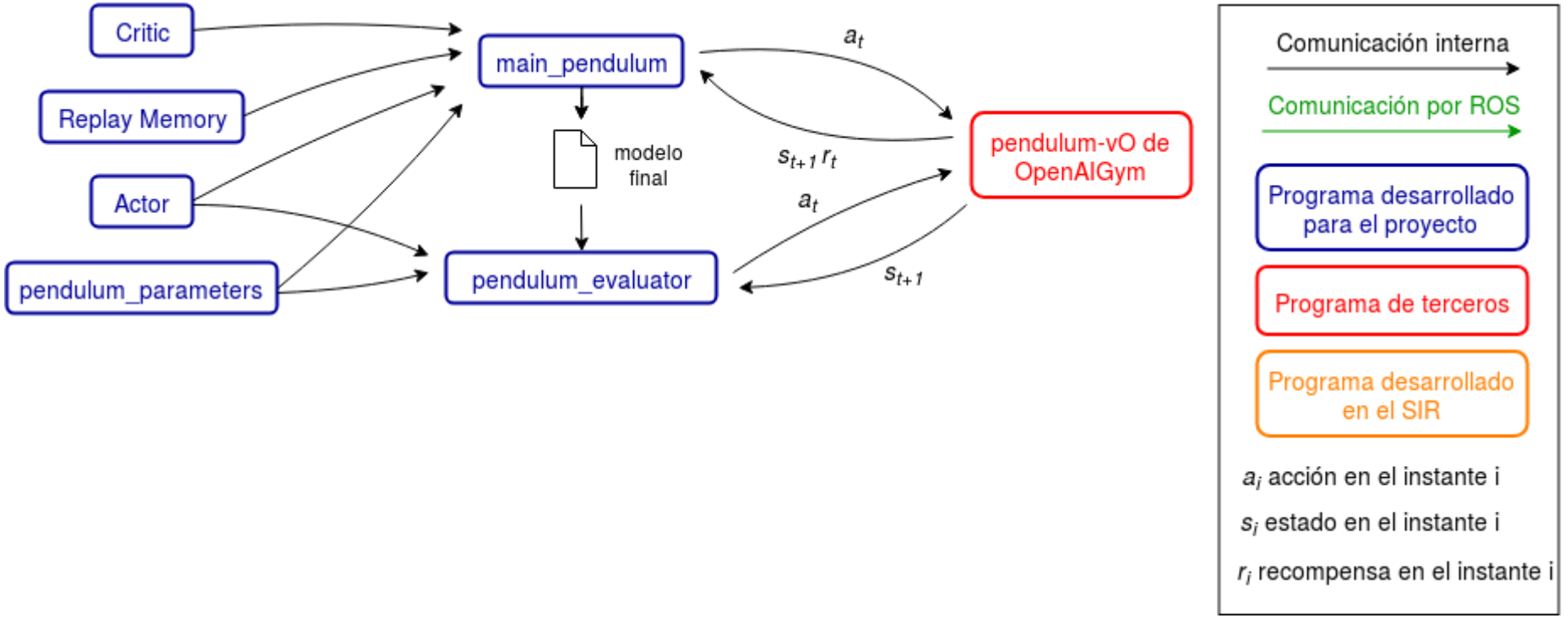
PENDULUM-V0



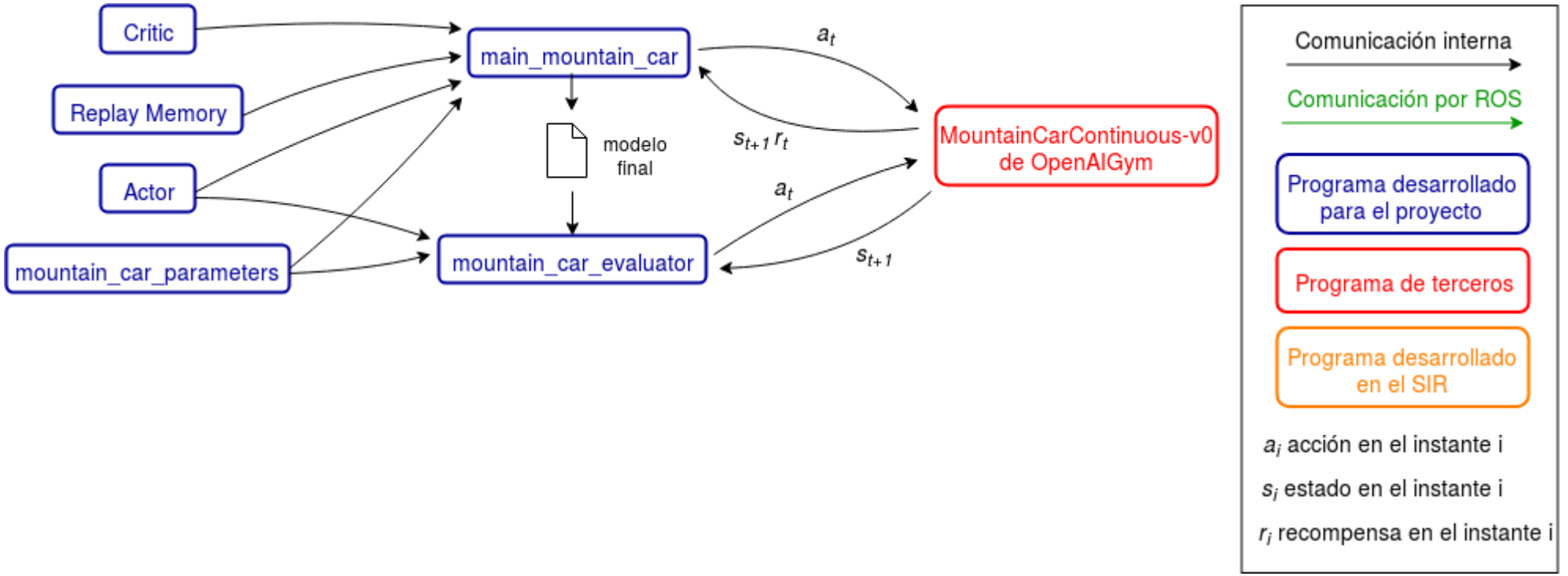
MOUNTAINCARCOUNTINUOUS-V0



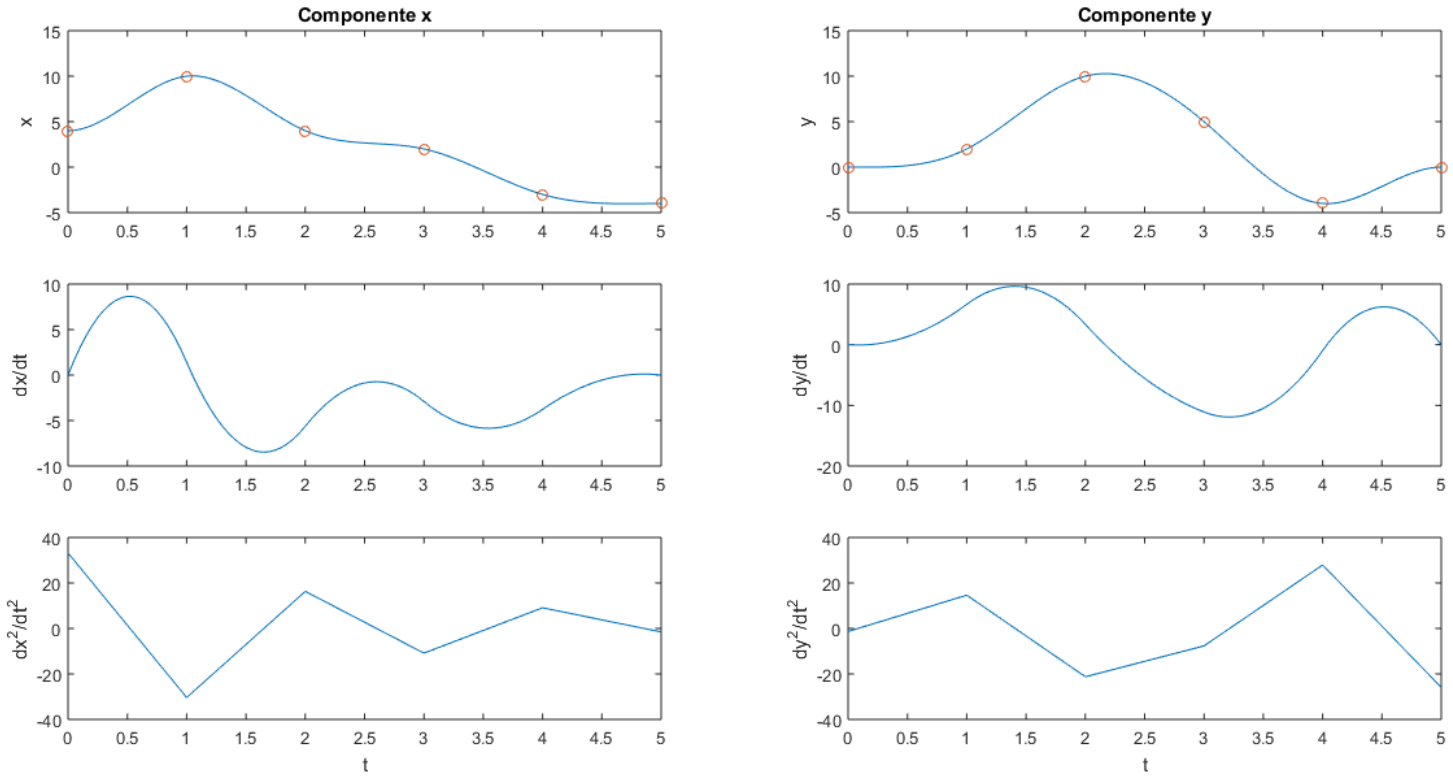
Pendulum


Critic
Replay Memory
Actor
pendulum_parameters
main_pendulum
modelo final
pendulum_evaluator
a_t
s_{t+1} r_t
a_t
s_{t+1}
pendulum-vO de OpenAIGym
Comunicación interna
Comunicación por ROS
Programa desarrollado para el proyecto
Programa de terceros
Programa desarrollado en el SIR
a_i acción en el instante i
s_i estado en el instante i
r_i recompensa en el instante i

MountainCarContinuous

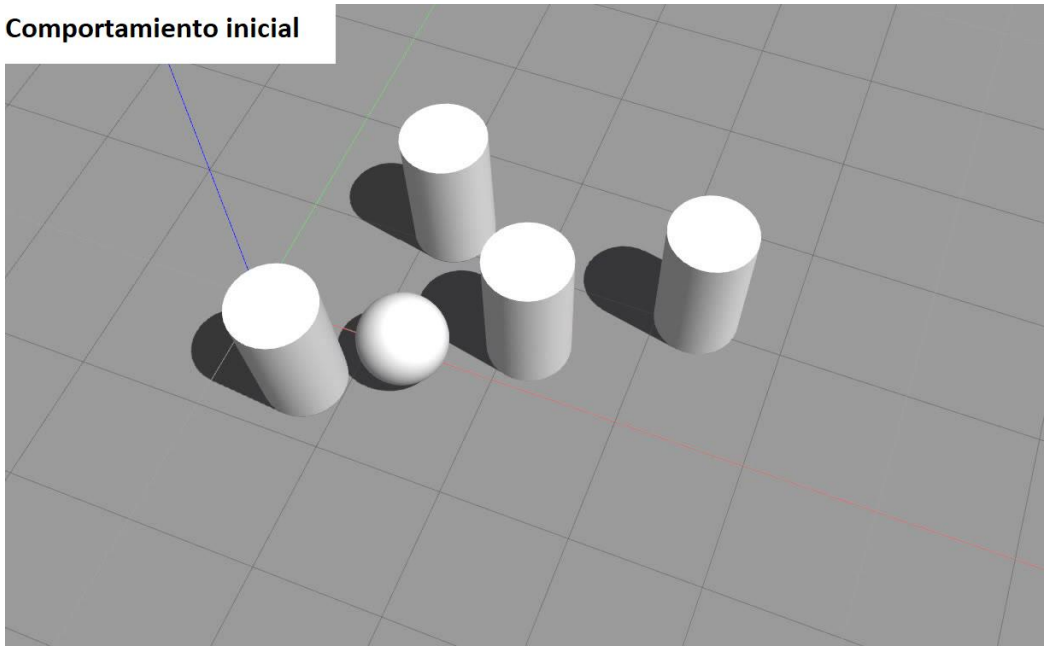


Generación mediante *splines*

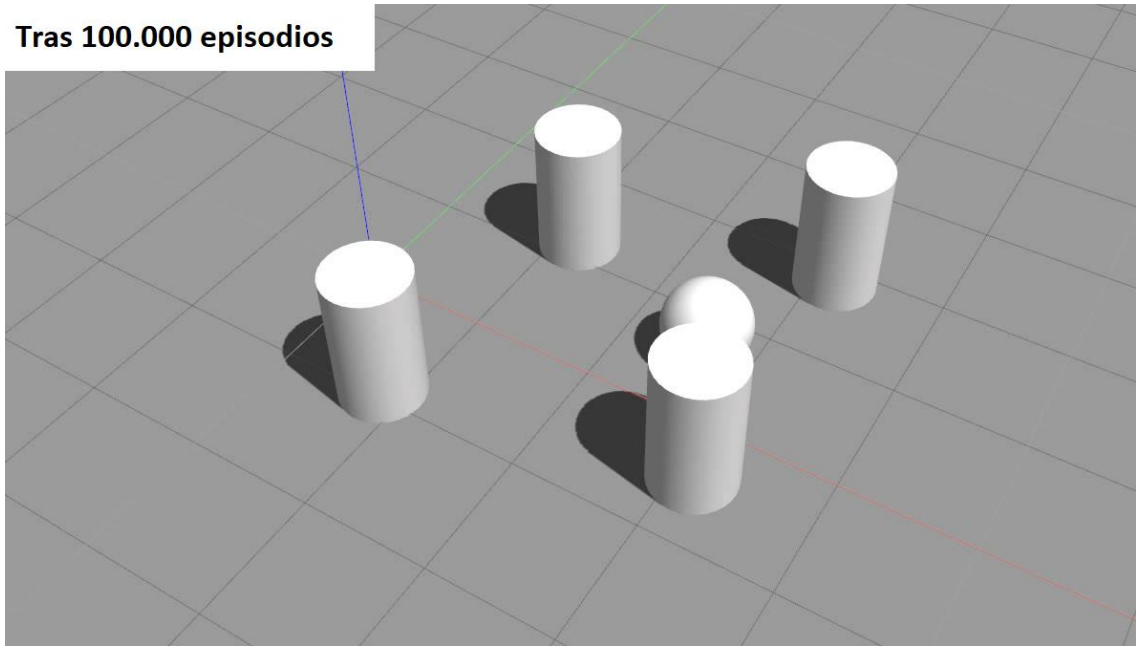


Planificador en simulación

COMPORTAMIENTO INICIAL



COMPORTAMIENTO TRAS 100000 EPISODIOS



Planificador en simulación

Sin *target networks*
Lazo abierto

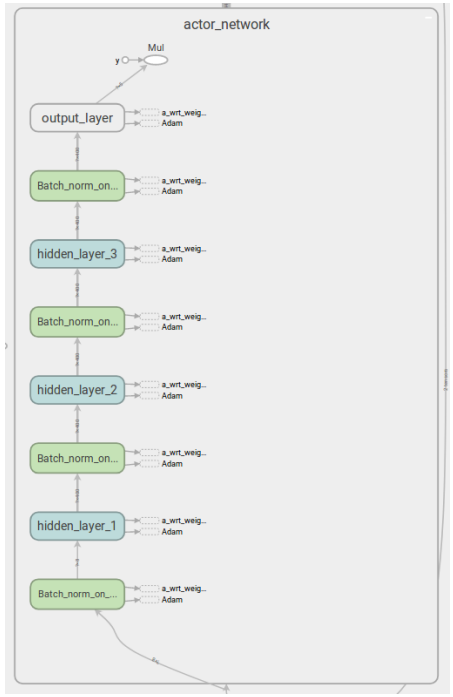
$$r_t = -o - 0,5d$$

Algoritmo 2: Algoritmo DDPG simplificado

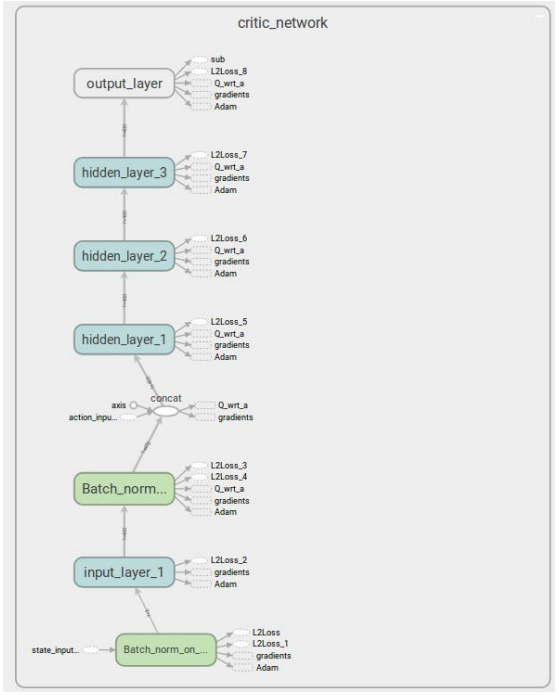
Inicializar aleatoriamente la red del crítico $Q(s, a \mid \theta^{\mu})$ y el actor $\mu(s \mid \theta^{\mu})$ con pesos θ^{Q} y θ^{μ}.
Inicializar el *buffer* de interacciones R
for *episodio=1, M* **do**
  Inicializar proceso aleatorio N para la exploración del espacio de acciones
  Recibir estado observacional inicial s_1
  for *t=1, T* **do**
    Seleccionar acción $a_t = \mu(s_t \mid \theta^{\mu}) + N_t$ de acorde con la política actual y el ruido exploratorio
    Ejecutar acción a_t y observar la recompensa r_t y el nuevo estado s_{t+1}
    Almacenar transición (s_t, a_t, r_t, s_{i+t}) en R
    Muestrear un *minibatch* aleatorio de N transiciones (s_i, a_i, r_i, s_{i+1}) de R
    Sea $y_i = r_i$
    Actualizar el crítico minimizando la pérdida: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i \mid \theta^{Q}))^2$
    Actualizar la política del actor utilizando el gradiente de la política muestreado: $\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a \mid \theta^{Q}) \mid_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s \mid \theta^{\mu}) \mid_{s_i}$
  end
end

Planificador en simulación

ACTOR



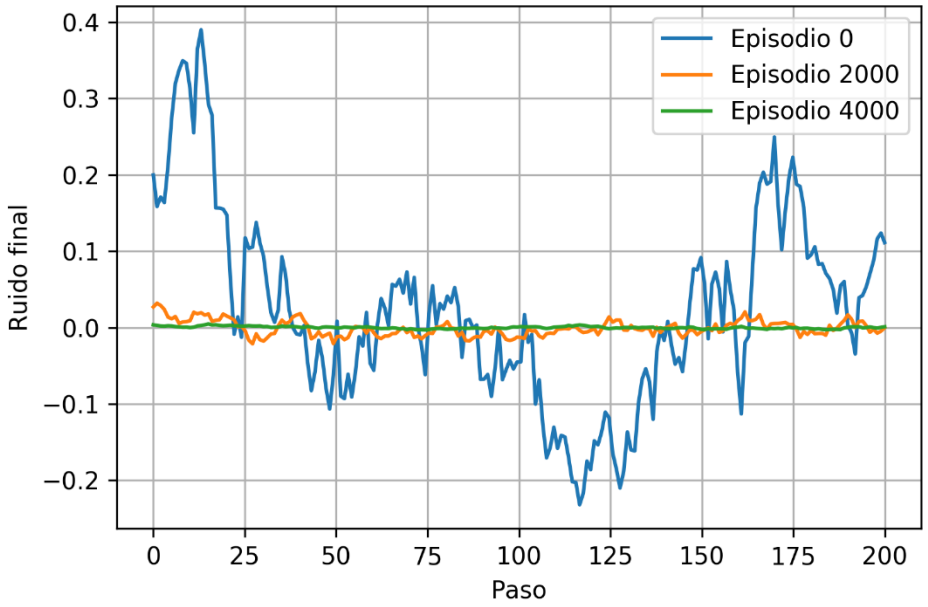
CRÍTICO

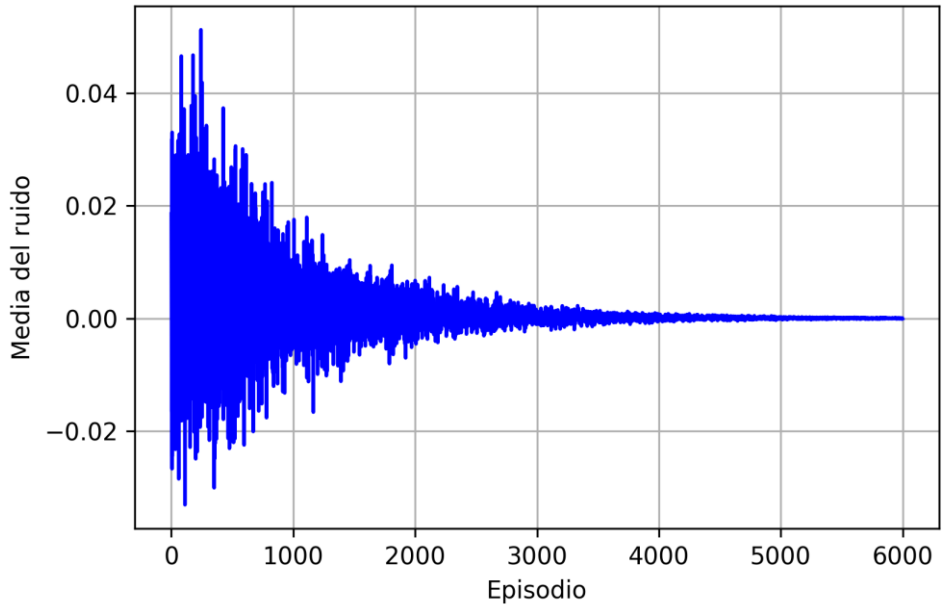


Elemento	Descripción
Arquitectura del crítico	2 capas internas de 400 unidades respectivamente. Todas las capas tienen activación ReLU.
Arquitectura del actor	2 capas internas de 400 unidades respectivamente. Todas las capas tienen activación ReLU excepto la salida, que usa una tanh.
Regularización L_2 *weight decay* para el crítico	10^{-2}
Optimizador para el crítico	ADAM
Optimizador para el actor	ADAM
Ratio de aprendizaje para el crítico	10^{-3}
Ratio de aprendizaje para el actor	10^{-4}
Factor de descuento γ	0.99
Factor τ para las actualizaciones de las *target networks*	0,01
Tamaño del *replay buffer*	10^5
Proceso Ornstein-Uhlenbeck	$\theta = 0{,}15$, $\sigma = 0{,}2$

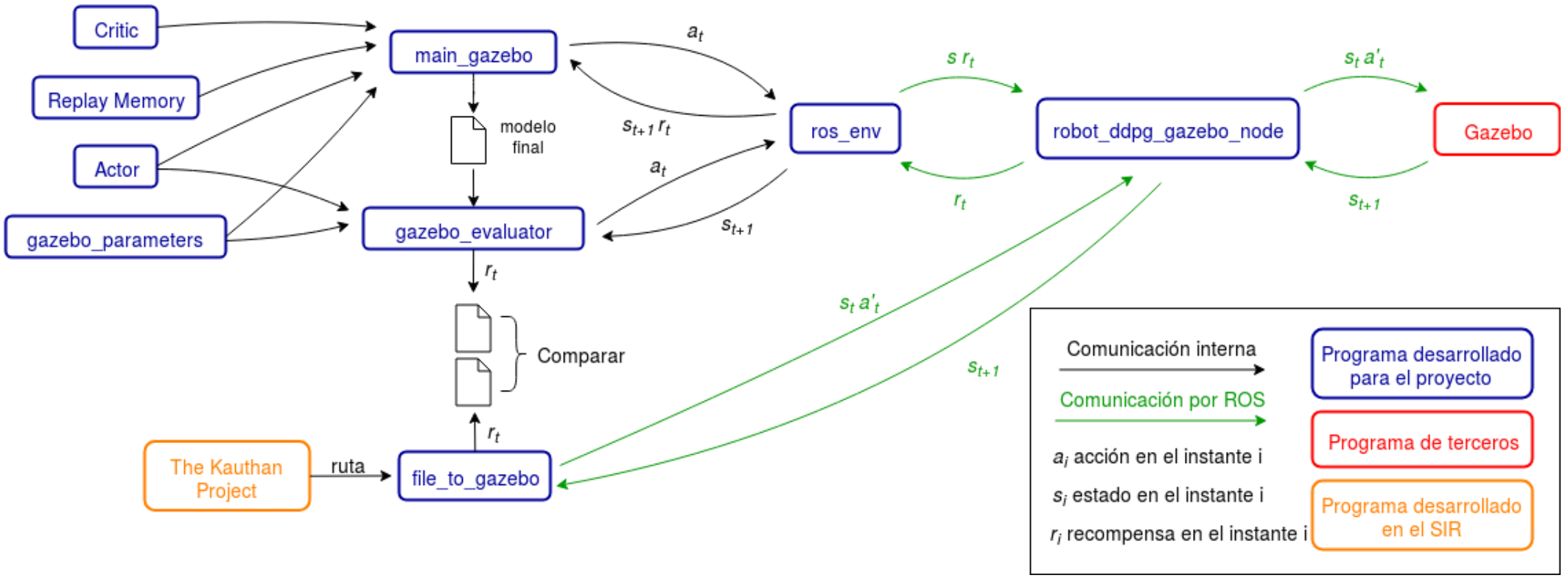
Planificador en simulación

Proceso Ornstein-Uhlenbeck

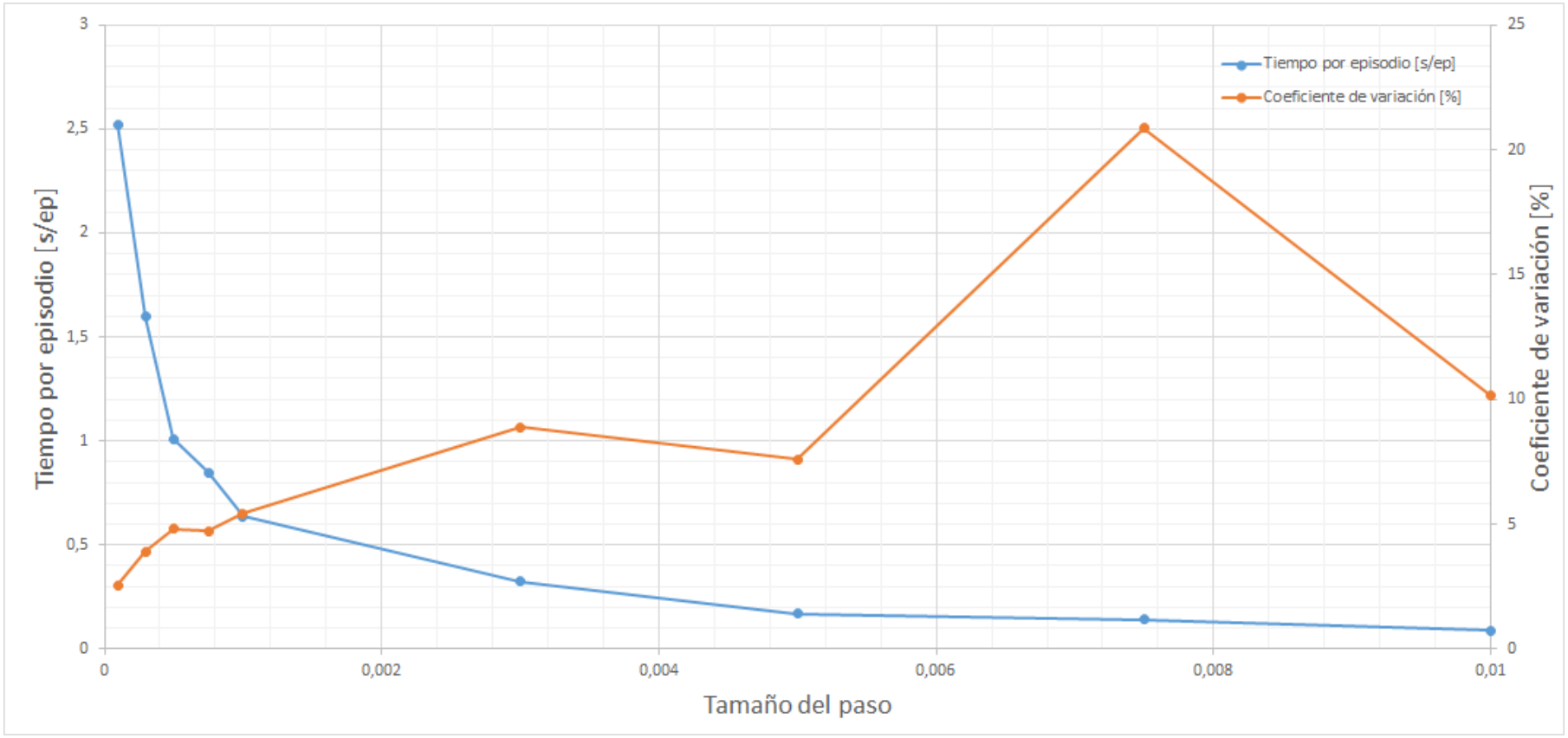




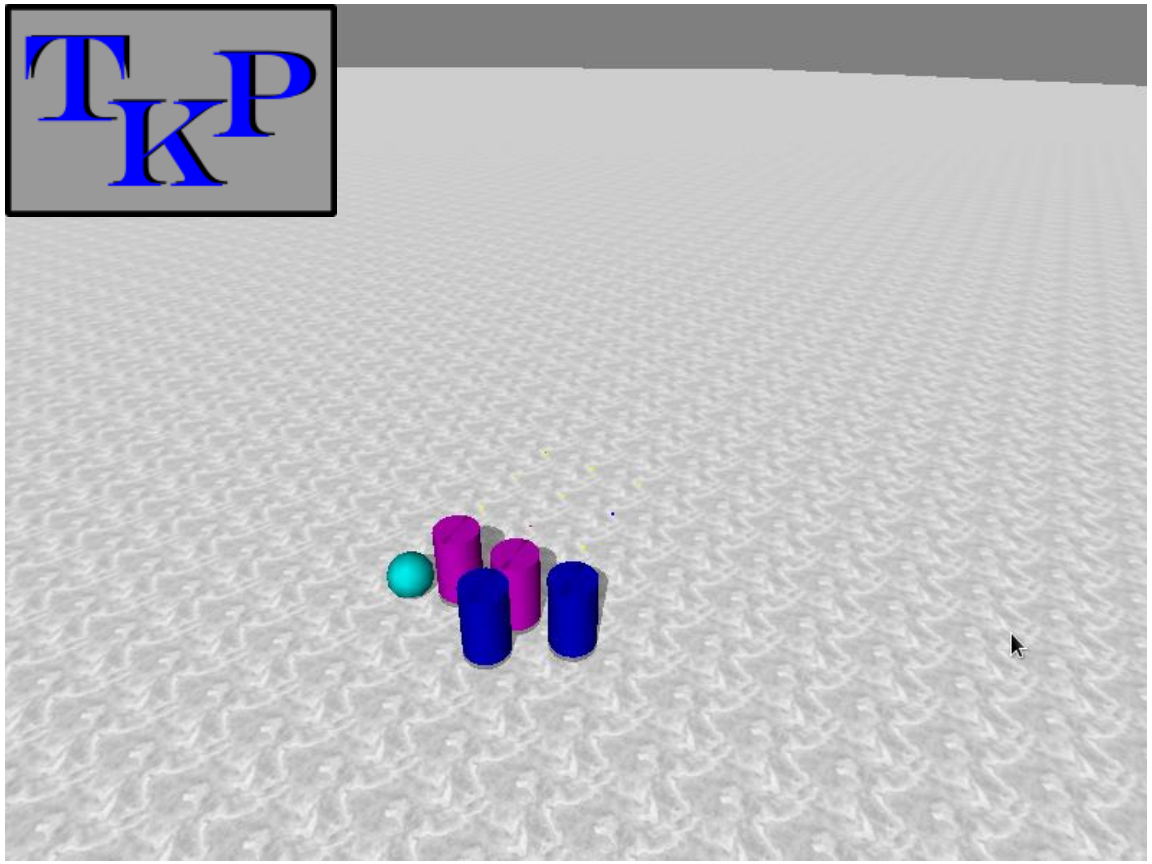
Planificador en simulación

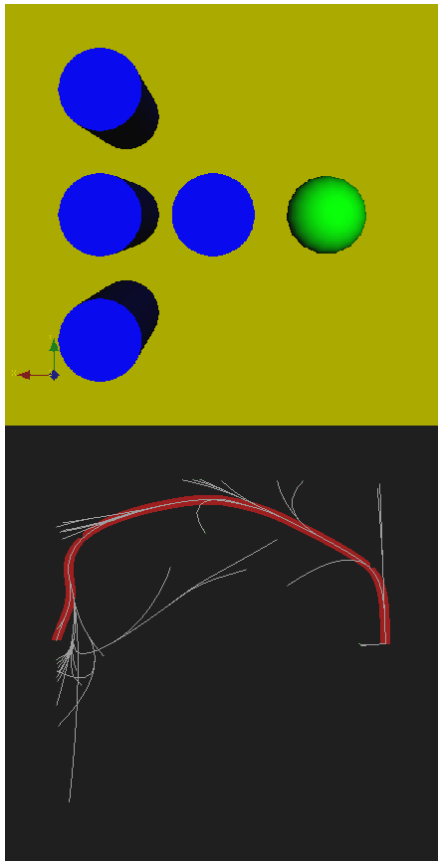


Planificador en simulación

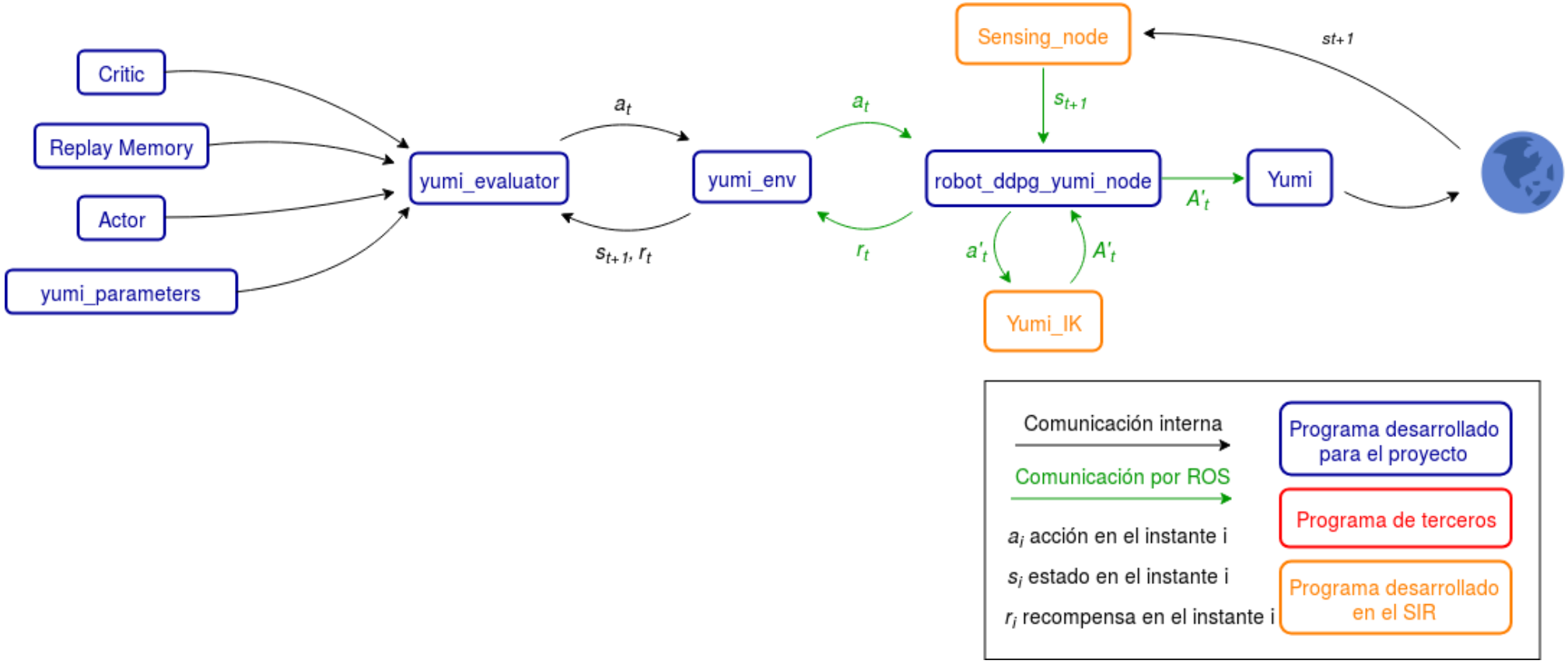


The kautham Project: KPIECE





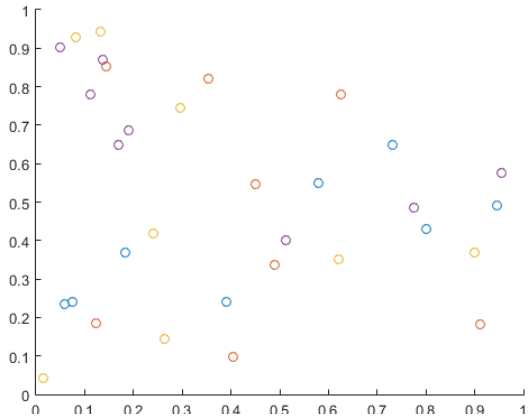
Implementación en YuMi



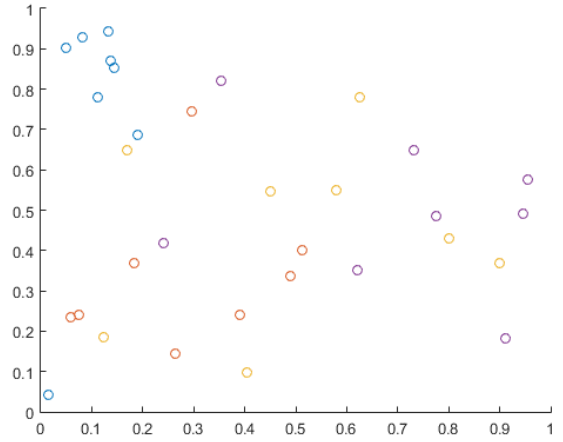
Experimentos

Experimento	Entornos	Descripción
Normal	Pendulum Mountain Car Simulación	DDPG básico, sin batch normalization.
Acciones en la segunda capa	Pendulum Mountain Car	Como en el algoritmo original, las acciones se añaden en la segunda capa del crítico.
Batch normalization	Pendulum Mountain Car	DDPG con batch normalization.
Sin *warmup*	Pendulum Mountain Car	Sin episodios de calentamiento.
Como el artículo original	Pendulum Mountain Car	DDPG tal y como se describe en el artículo original.
Entradas reordenadas	Simulación	DDPG básico, sin batch normalization pero con las entradas reordenadas.
Como el artículo original y con entradas reordenadas	Simulación	DDPG tal y como se describe en el artículo original y con las entradas reordenadas.

Original



Reordenado



Resultados

Introducción
- Objetivos
- Motivación

Algoritmo
- Antecedentes
- Puntos clave
- Núcleo

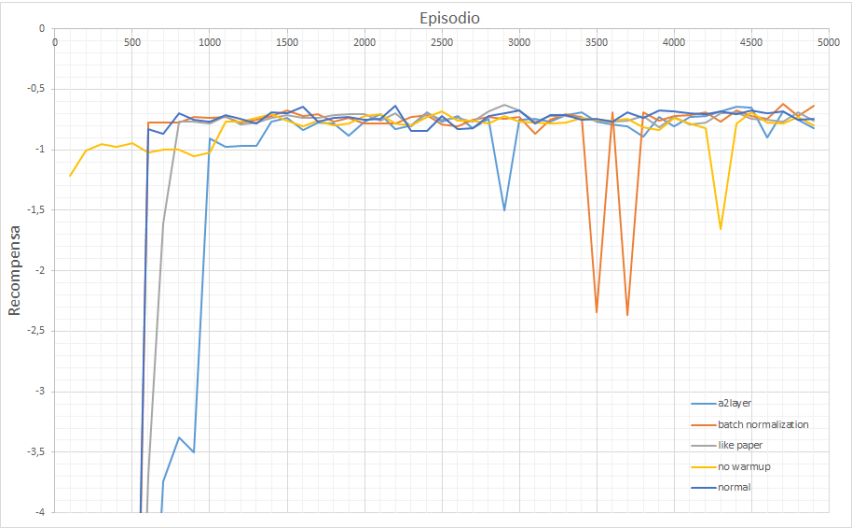
Implementación
- Entornos
- Arquitectura
- Detalles
- Experimentos

Resultados
- Pruebas con modificaciones
- Comparativa con KPIECE

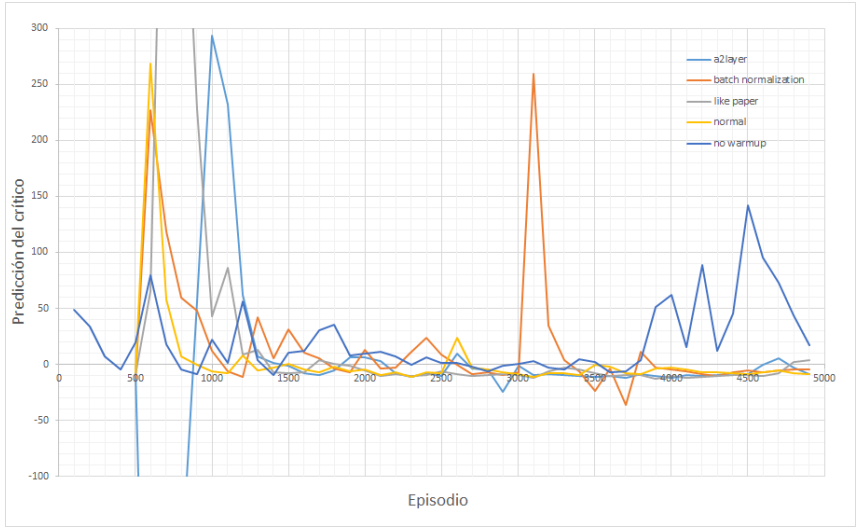
Conclusiones y trabajo futuro

Pendulum

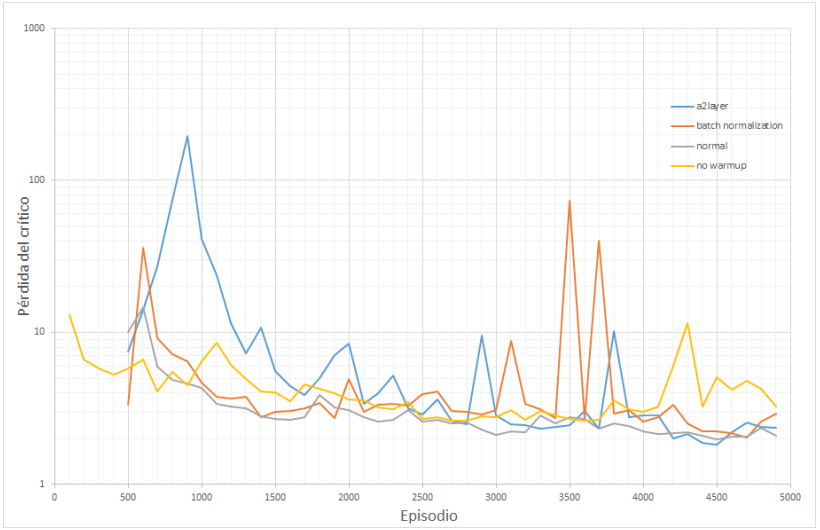
Recompensa



Predicción del crítico

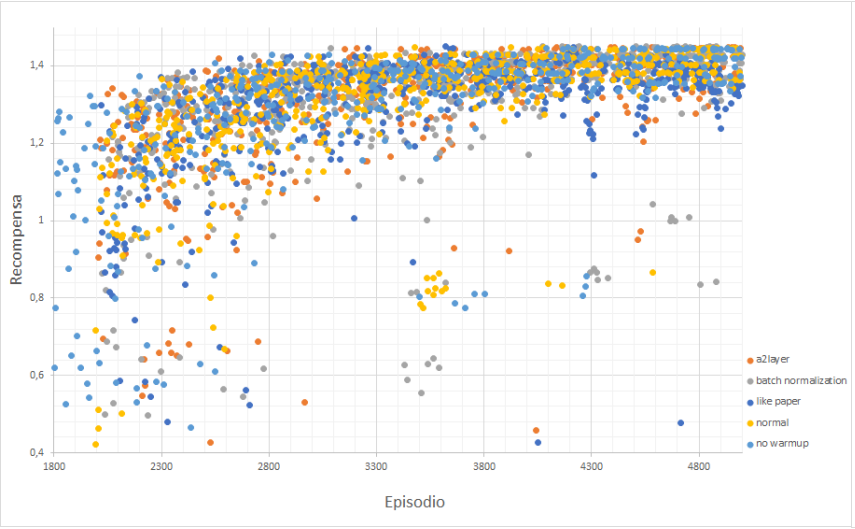


Pérdida del crítico

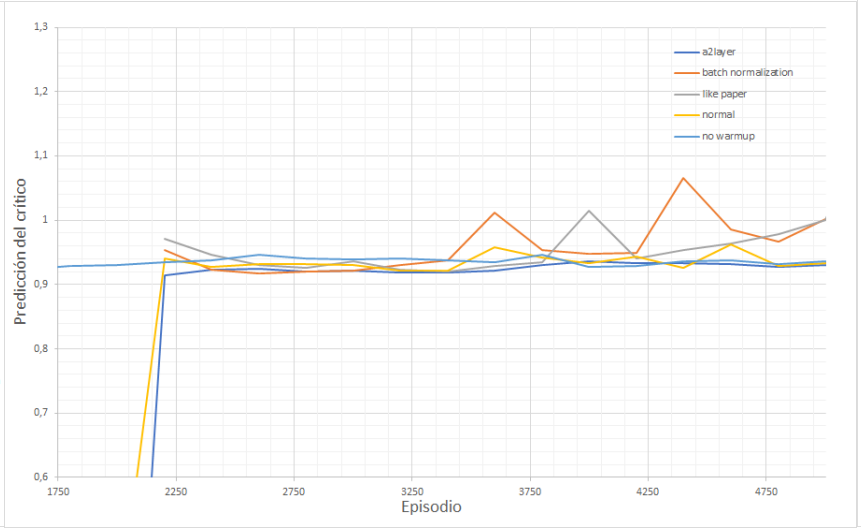


Mountain Car

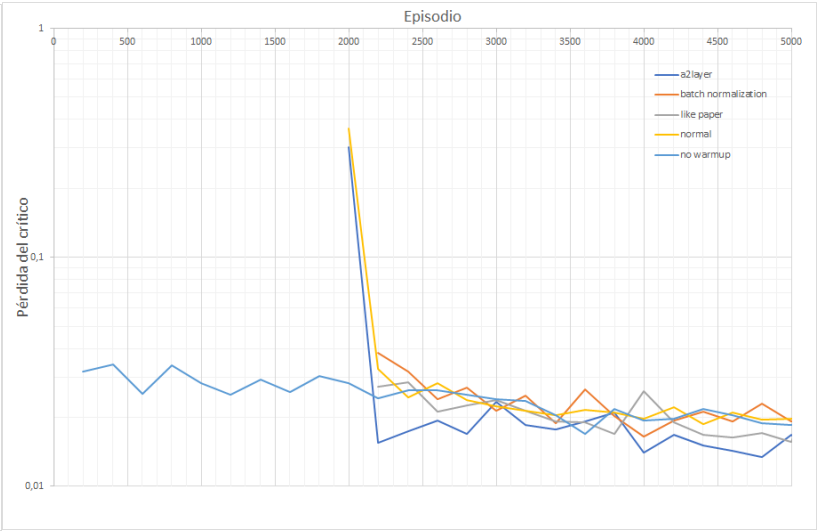
Recompensa



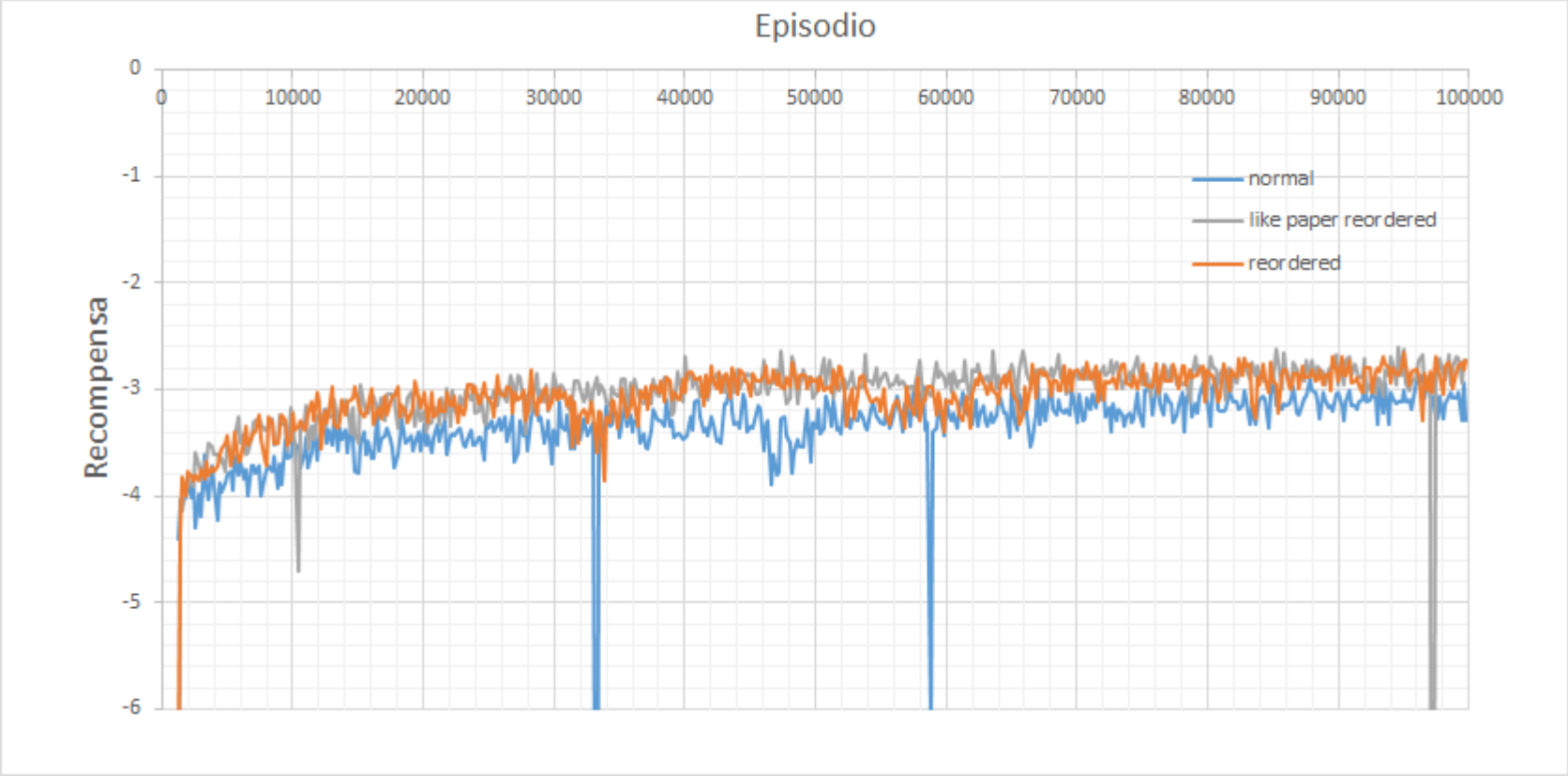
Predicción del crítico



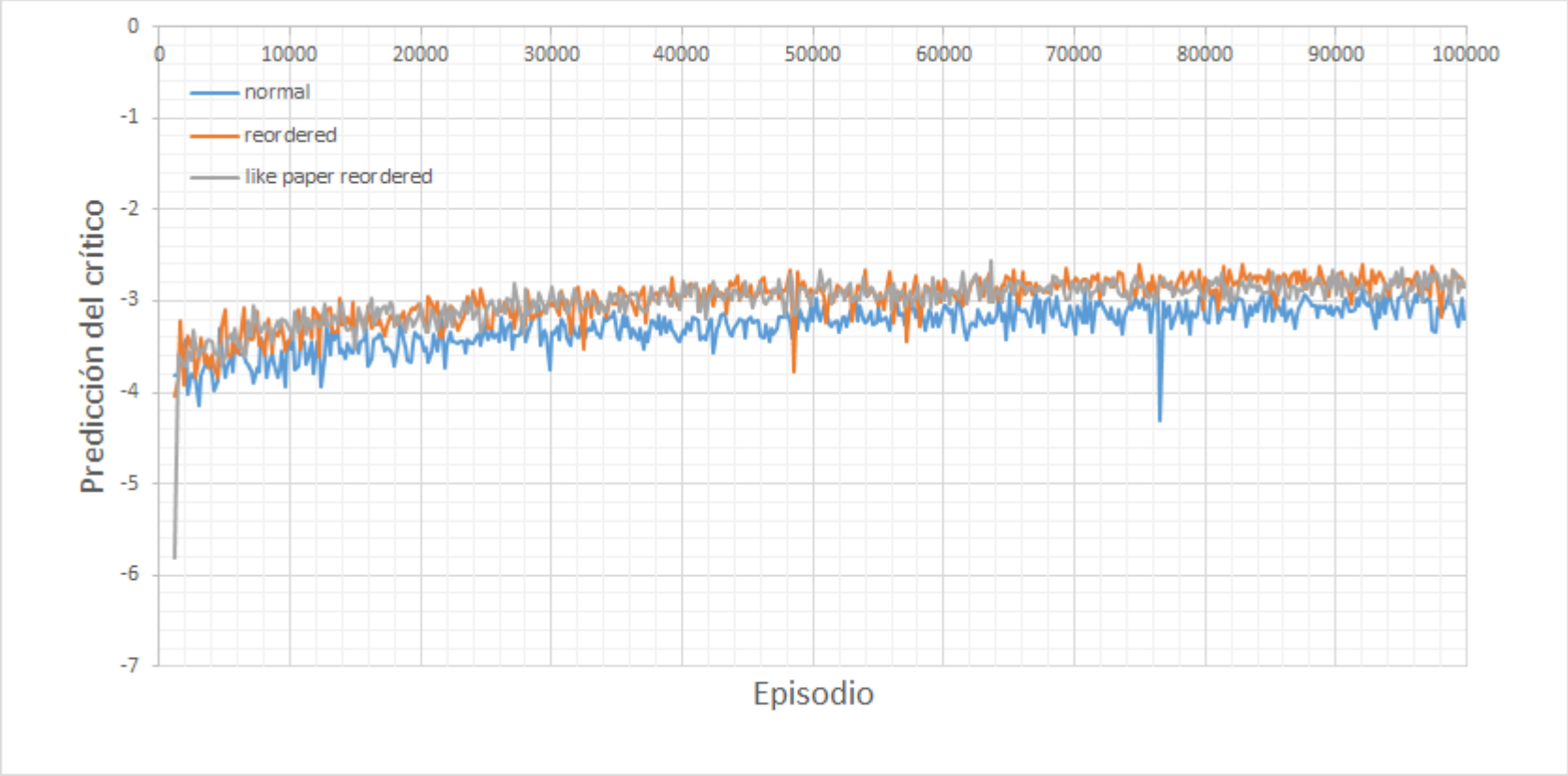
Pérdida del crítico



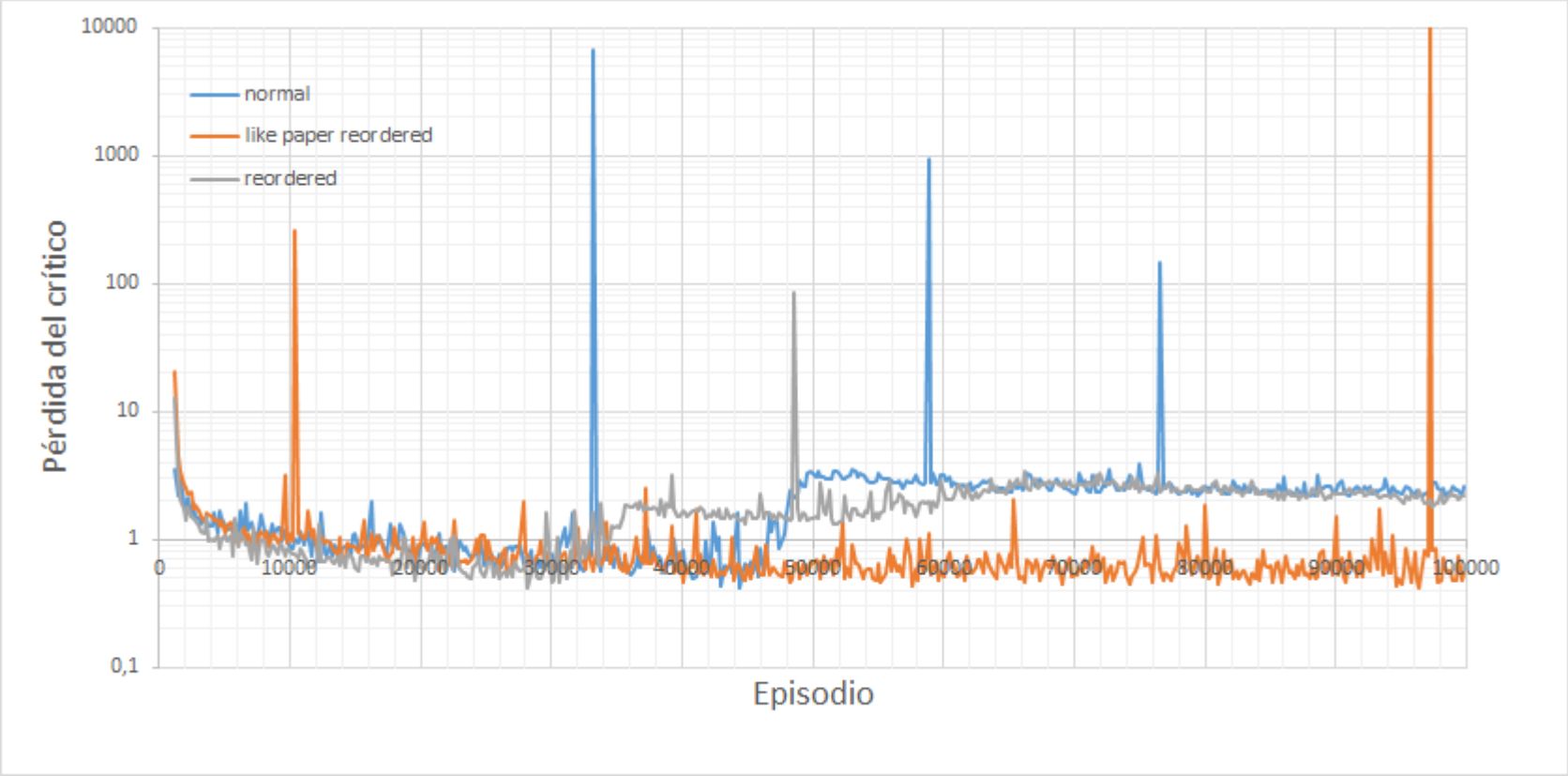
Planificador en simulación



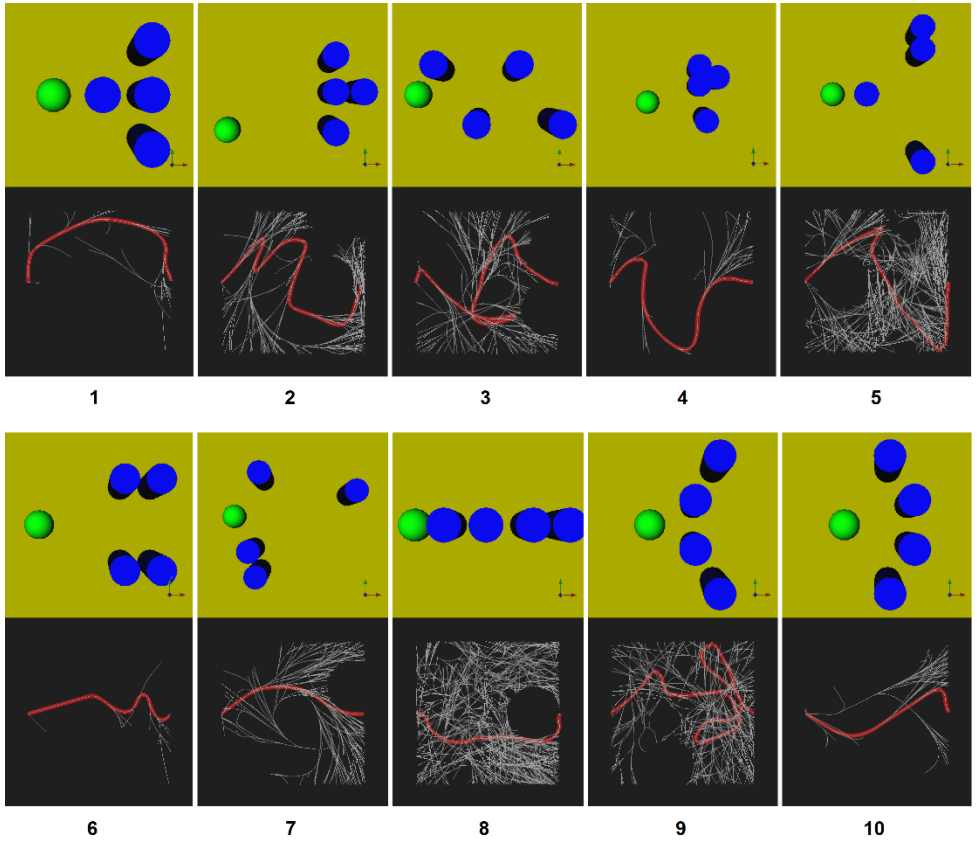
Planificador en simulación



Planificador en simulación

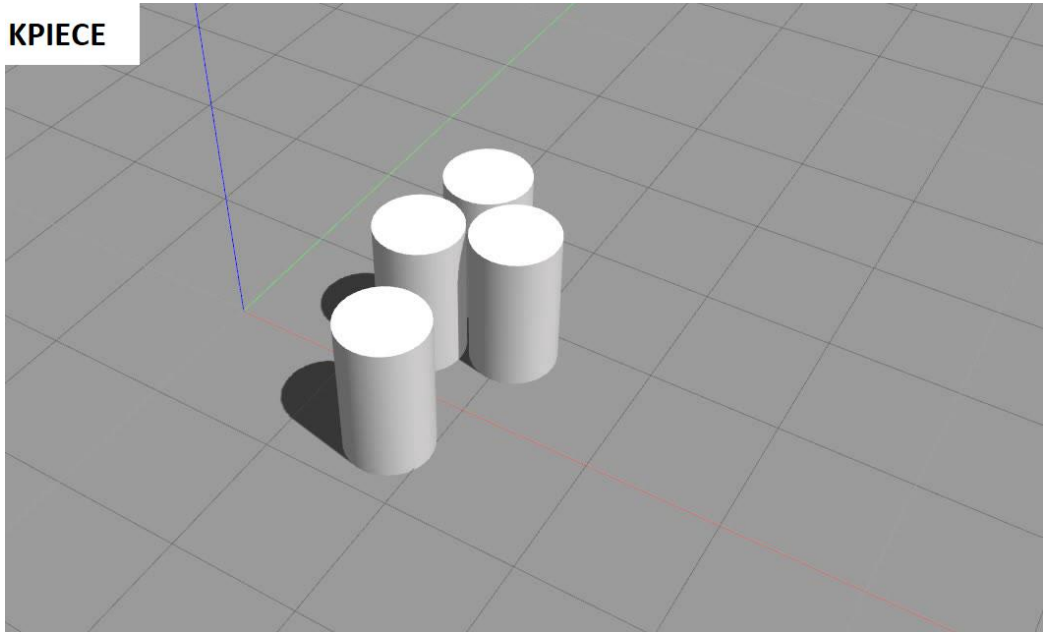


DDPG vs KPIECE

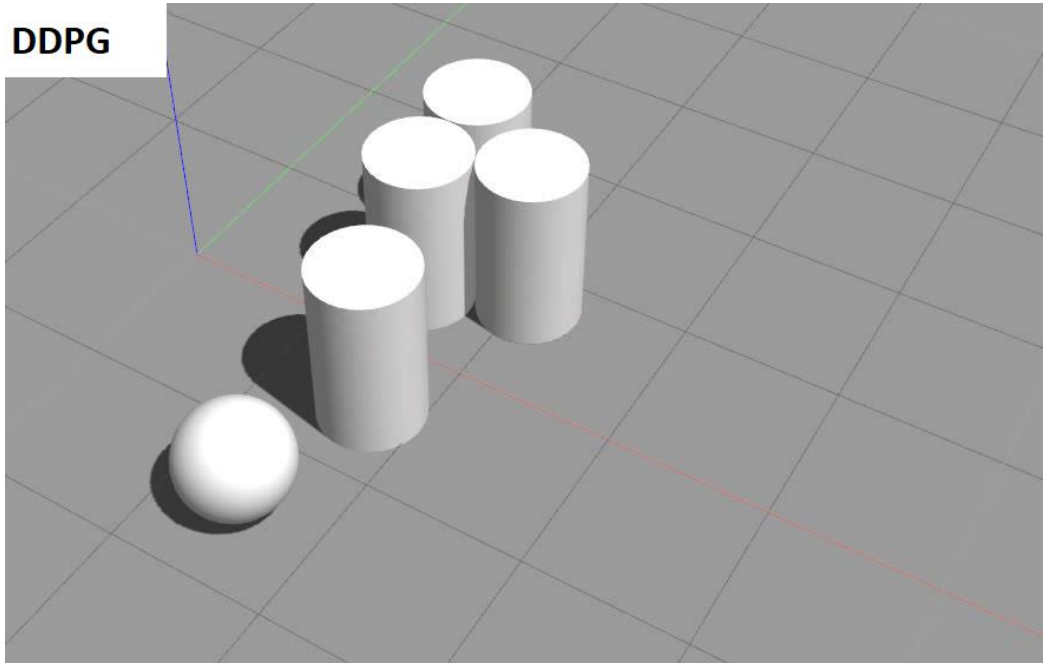


DDPG vs KPIECE

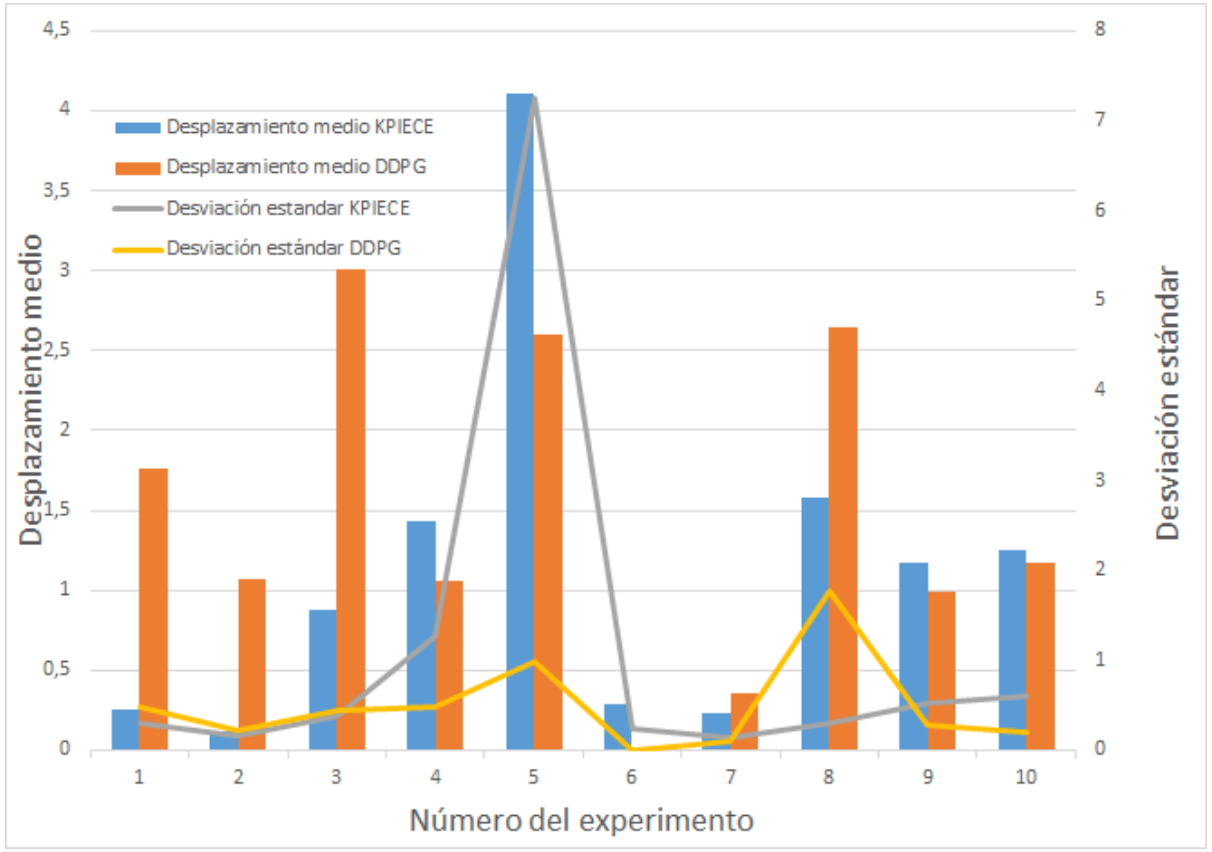
KPIECE



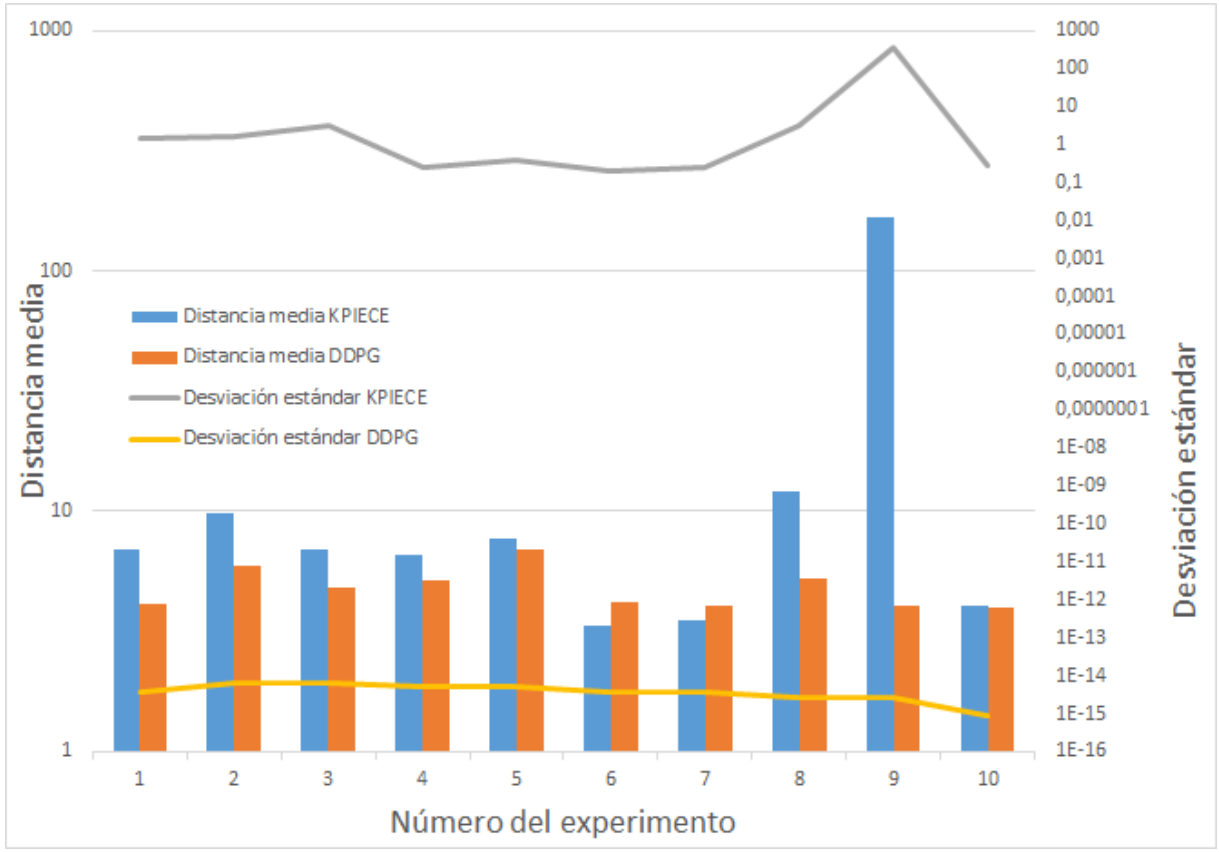
DDPG



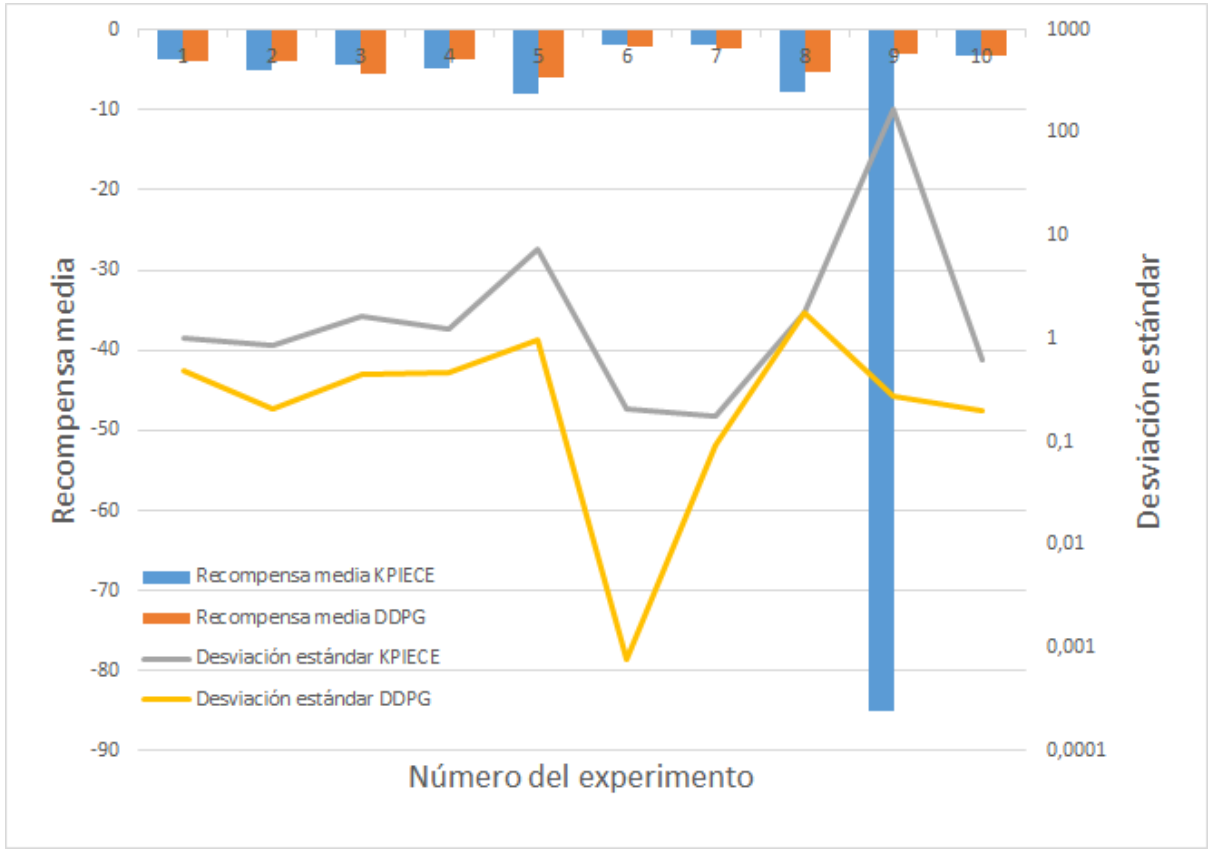
DDPG vs KPIECE



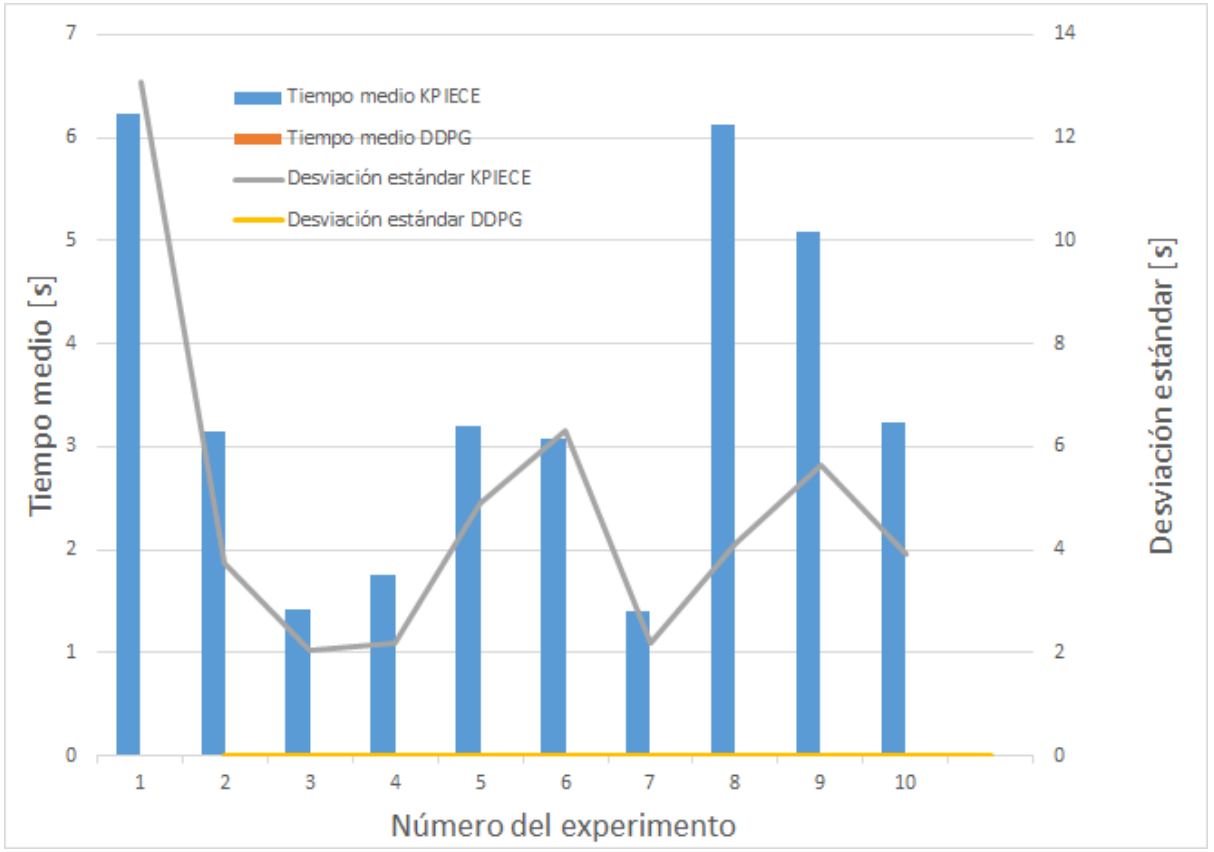
DDPG vs KPIECE



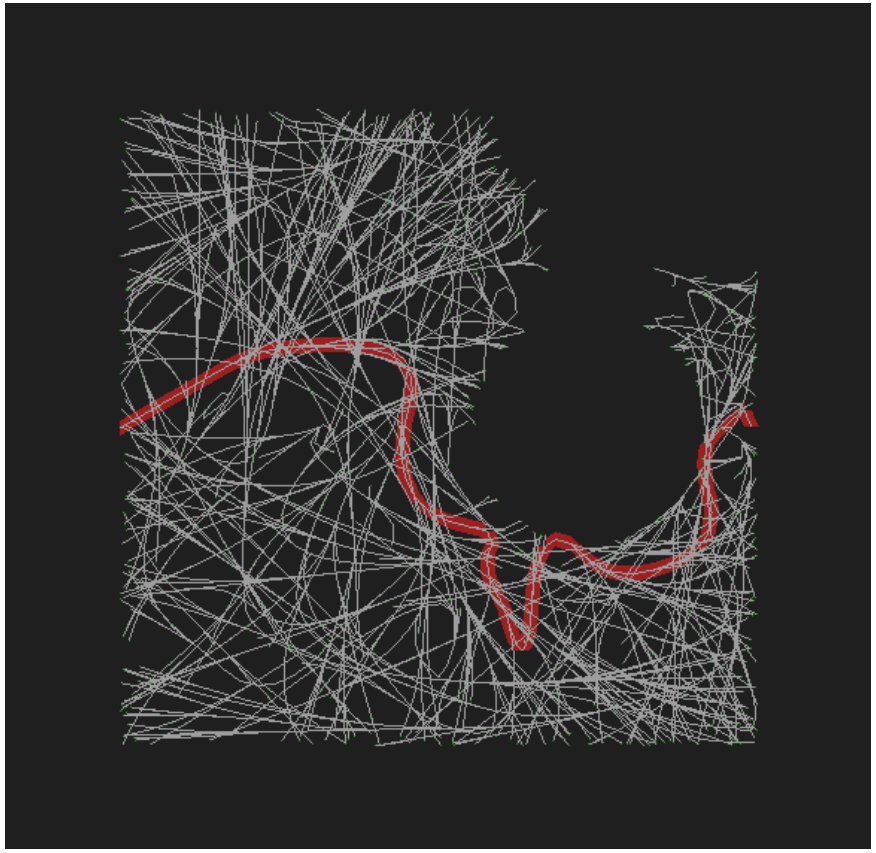
DDPG vs KPIECE



DDPG vs KPIECE



DDPG vs KPIECE



Conclusiones y trabajo futuro

Gran potencial

- Rapidez de respuesta
- Generalización
- Trabajo en entornos de alta dimensionalidad

Trabajo futuro

- Mejoras al *experience replay*
- Planificador en lazo cerrado

Muchas gracias por su atención