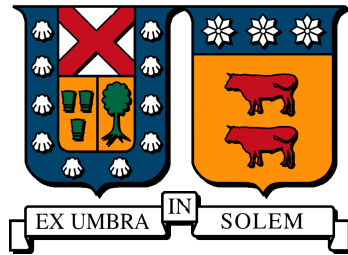


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INGENIERÍA MECÁNICA
SANTIAGO-CHILE



**SISTEMA DE CONTROL PARA PRÓTESIS
DE MANO MEDIANTE ALGORITMOS DE
APRENDIZAJE Y SISTEMA DE
MONITOREO EN AMBIENTE
CONTROLADO**

DANIEL IGNACIO PÉREZ SOTO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

PROFESOR GUÍA: DR. ING. DANILO ESTAY BARRIENTOS
PROFESOR CORREFERENTE: MG. ING. JAIME IBARRA GONZÁLEZ

JUNIO 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: SISTEMA DE CONTROL PARA PRÓTESIS DE MANO MEDIANTE ALGORITMOS DE APRENDIZAJE Y SISTEMA DE MONITOREO EN AMBIENTE CONTROLADO

Nombre del candidato(a): Daniel Ignacio Pérez Soto

Carrera / Grado: Ingeniería Civil Mecánica

Campus: Santiago San Joaquín ; **Departamento:** DIMEC

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Danilo Alejandro Estay Barrientos, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 11/08/2025

; Firma:

Estudiante o Candidato(a):

Fecha: 11/08/2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Abstract

This thesis presents a semi-autonomous control system for a prosthetic hand that employs monocular vision and deep learning to determine grip closure. The model was trained in simulation (PyBullet) and subsequently validated on a physical prototype outfitted with a YOLOv4 object detector ($>90\%$ precision) and an RGB-based depth estimator (Depth Anything). In simulation, the system achieved approximately 95% success, and in laboratory trials it reached around 85% accuracy. Throughout physical testing, no graspable object was ever missed, and erroneous activations occurred in about 15% of cases. These results confirm the viability and effectiveness of the proposed methodology, paving the way for enhanced autonomy and reliability in hand prostheses.

En este trabajo se desarrolla un sistema semiautónomo que decide el cierre de la garra mediante visión monocular y aprendizaje profundo. Los resultados muestran que el sistema propuesto es viable y efectivo: el modelo entrenado alcanzó aproximadamente un 95% de éxito en simulaciones, y logró alrededor de un 85% de acierto en las pruebas con el prototipo físico en laboratorio. El detector YOLO demostró una precisión superior al 90% en la identificación de los elementos. En las pruebas experimentales, el sistema nunca omitió un objeto presente, aunque presentó algunas activaciones erróneas en torno al 15% . Este desempeño confirma la relevancia de la metodología, lo que sienta un precedente para mejorar la autonomía y confiabilidad de las prótesis de mano en el futuro.

Índice general

1. Introducción	1
1.1. Propósito	2
1.2. Objetivos	2
1.2.1. Objetivo general	3
1.2.2. Objetivos específicos	3
1.3. Alcances y Limitaciones	3
2. Estado del Arte	4
2.1. Métodos de control para prótesis	4
2.1.1. Control Convencional	4
2.1.2. Control Mioeléctrico y bioseñales	5
2.1.3. Reinervación Muscular Dirigida	5
2.2. Inteligencia Artificial y Redes Neuronales	6
2.2.1. Aprendizaje por Refuerzo	7
2.2.2. Fundamentos	7
2.2.3. Aplicaciones en Control de Prótesis	8
2.3. Visión por Computador	8
2.3.1. Algoritmo de Detección de Objetos: YOLO	8
2.3.2. Algoritmo de Segmentación: DEPTH ANYTHING	9

3. Metodología	11
3.1. Diseño general de la investigación	11
3.2. Herramientas de desarrollo	12
3.3. Plan de recolección de datos	14
3.4. Plan de validación experimental	14
3.5. Otros aspectos metodológicos	15
4. Desarrollo de Sistema de Control	16
4.1. Arquitectura General	16
4.2. Configuración del Entorno de Simulación con PyBullet	17
4.2.1. Número de Iteraciones y Episodios de Entrenamiento	18
4.3. Módulo de Aprendizaje por Refuerzo (PPO)	18
4.3.1. Criterios de Convergencia	20
4.4. Módulo de Visión por Computadora	20
4.4.1. Dataset de Entrenamiento	21
4.4.2. Configuración e Hiperparámetros	21
4.4.3. Integración con la ROI	21
4.4.4. Algoritmo Depth Anything	21
4.5. Integración Final en un Pipeline	23
5. Validación Experimental	24
5.1. Generación de Ambiente Controlado	24
5.1.1. Montaje Físico	24
5.1.2. Evaluación Cualitativa	25
5.1.3. Pruebas en Tiempo Real	26
5.2. Criterios de Evaluación	26
6. Resultados	27

6.1. Desempeño del Entrenamiento	27
6.2. Desempeño del Algoritmo en Simulaciones	29
6.3. Desempeño de YOLO	30
6.4. Desempeño del Algoritmo Implementado	32
6.5. Discusión	33
6.5.1. Comparación con Métodos Alternativos	35
6.5.2. Fuentes de Error	36
7. Conclusión	38
7.1. Conclusiones del Trabajo	38
7.2. Trabajos Futuros	40

Índice de figuras

2.1.	Diagrama conceptual del ciclo de aprendizaje por refuerzo	7
2.2.	Extracto de video con utilización de Yolo v4 en tiempo real	9
2.3.	Comparación entre la imagen RGB original (izquierda) y el mapa de profundidad estimado por <i>Depth Anything</i> (derecha).	10
3.1.	Diagrama de flujo metodológico.	12
4.1.	Diagrama de bloques del flujo de información entre los módulos.	17
4.2.	Distribución de casos y recompensas del agente. De izquierda a derecha y de arriba hacia abajo.	19
4.3.	Visualización del cilindro fantasma (verde translúcido) ubicado entre los dedos del <i>gripper</i> . Su colisión con el cubo indica la posibilidad de captura exitosa.	20
4.4.	(a) Imagen estimada desde entorno real con <i>Depth Anything</i>	23
4.5.	(b) Imagen generada desde <i>PyBullet</i> (600x600, escala de grises)	23
4.6.	Comparación visual entre las imágenes de profundidad utilizadas como entrada por la red neuronal. Se observa una similitud estructural entre el entorno real (a) y el entorno simulado (b), lo cual facilita la generalización del modelo entrenado.	23
5.1.	Montaje físico del sistema de prueba. Se observa el gripper robótico montado frente al objeto, así como la cámara RGB encargada de capturar las imágenes de entrada al sistema.	25

6.1. Evolución de la pérdida de entropía durante el entrenamiento.	27
6.2. Evolución de la pérdida del gradiente de la política.	28
6.3. Recompensa promedio obtenida durante la evaluación.	28
6.4. Muestras de detección del modelo YOLOv4 aplicadas a distintas imágenes reales del entorno controlado. Se observa correcta identificación del gripper y del objeto bajo diversas condiciones de iluminación y orientación.	31
6.5. Curvas de entrenamiento y validación del modelo YOLOv4. Se muestra la evolución de las pérdidas (losses) y las métricas de desempeño como precisión, recall y <i>mean Average Precision</i> durante 100 épocas.	32
6.6. Secuencia de imágenes de la garra real al detectar y sujetar el objeto. . .	33
6.7. Secuencia de imágenes de la garra real al decidir la acción de abrir. . .	33

Índice de cuadros

6.1. Tabla resumen de resultados en prueba simulada.	29
6.2. Tabla con valores de las métricas.	30

Capítulo 1

Introducción

En las últimas décadas, el desarrollo de prótesis de miembro superior ha evolucionado significativamente, impulsado por los avances en robótica, biomecánica y neuroingeniería. Estos progresos han permitido diseñar dispositivos protésicos que no solo reemplazan la estructura física del miembro perdido, sino que buscan restaurar funciones motoras de manera confiable, acercándose cada vez más a la funcionalidad de una extremidad biológica [1].

Uno de los principales desafíos en este campo ha sido el diseño de mecanismos de control eficientes y adaptativos. Las prótesis de mano modernas requieren interpretar de manera precisa la intención de movimiento del usuario, adaptándose dinámicamente a distintos contextos de uso [2]. Esto es comparable al control de brazos robóticos. Sin embargo, mientras que estos últimos pueden basar su control en predicciones matemáticas, las prótesis enfrentan restricciones adicionales como la biocompatibilidad, la facilidad de uso diario, la variabilidad fisiológica de los usuarios amputados y el dinamismo de entornos no controlados [3].

Históricamente, los sistemas de control comenzaron con métodos simples mecánicos o con control por señal mioeléctrica, donde las señales eléctricas generadas por los músculos se utilizaban para accionar motores. Sin embargo, estos métodos presentan limitaciones importantes en cuanto a la cantidad de movimientos posibles, la precisión y la naturalidad del control [4]. Esto ha motivado la investigación en enfoques más complejos, como el reconocimiento de patrones mediante *machine learning* y la decodificación de señales neuronales pero aún no se ha desarrollado una investigación profunda en el uso de visión por computadora para asistir en la toma de decisiones del

dispositivo protésico [5, 6].

En este contexto, tecnologías como el aprendizaje profundo y la visión artificial cobran protagonismo como estrategias para mejorar la capacidad de percepción del entorno y la interacción con objetos. La posibilidad de incorporar cámaras y algoritmos de detección de objetos permite a las prótesis obtener información contextual del entorno, facilitando un control más inteligente y adaptativo [7]. Esto busca superar las limitaciones actuales y acercar el funcionamiento de las prótesis al nivel esperado por los usuarios.

A pesar del progreso, alcanzar un control verdaderamente natural y autónomo en prótesis de mano sigue siendo un problema abierto. Ningún estudio se ha realizado, debido a que factores como la escalabilidad de los modelos de control a diferentes condiciones y la capacidad de aprendizaje continuo del sistema son aspectos que aún requieren investigación y desarrollo [8]. De esta manera, se hace evidente la necesidad de explorar técnicas modernas de inteligencia artificial, para avanzar hacia un control más funcional, accesible y centrado en el usuario.

1.1. Propósito

Junto a lo mencionado en la sección anterior, este trabajo de memoria se enmarca en diseñar e implementar un sistema de control capaz de detectar la intención de captura de un objeto incorporando algoritmos de aprendizaje reforzado, redes neuronales convolucionales y visión por computadora, con el fin de incentivar una línea de investigación en esta área en busca de integraciones más robustas en entornos no controlados.

1.2. Objetivos

La propuesta se enmarca dentro del enfoque de control semiautónomo, donde se contempla tanto el desarrollo del modelo computacional como la validación del sistema en condiciones controladas. Para cumplir con estas generalidades se plantean diferentes objetivos.

1.2.1. Objetivo general

Desarrollar un sistema de control para una prótesis de mano con amputación a nivel de muñeca, basado en la detección de la intención de capturar objetos mediante visión por computador.

1.2.2. Objetivos específicos

- Analizar el estado del arte referente a métodos de control de prótesis.
- Desarrollar un algoritmo de control mediante aprendizaje reforzado capaz de detectar la intención de tomar un objeto.
- Validar de forma experimental el algoritmo desarrollado para la captura de objetos.
- Evaluar el rendimiento del algoritmo en el entorno de prueba.

1.3. Alcances y Limitaciones

El presente trabajo se enmarca en el desarrollo y validación de un sistema de control, por lo que se centra en evaluar la factibilidad de utilizar información visual para interpretar la intención del usuario al manipular objetos, esto configurado específicamente en el entorno de simulación y ambiente controlado.

Para esto se implementa utilizando una pinza robótica estilo gripper que simula el comportamiento funcional de una prótesis, permitiendo simplificar el análisis y costo computacional. Esta elección permite centrarse en la lógica de la toma de decisiones y percepción visual, sin complejidades mecánicas propias de una prótesis mano completa.

En cuanto al hardware, todo el desarrollo y la validación se realizan utilizando equipos personales. Se busca que los resultados obtenidos deben entenderse como una prueba de concepto en condiciones experimentales, que sienta las bases para desarrollos futuros en entornos más complejos.

Capítulo 2

Estado del Arte

El desarrollo de prótesis de mano con niveles cada vez mayores de destreza y autonomía ha requerido la conjunción de múltiples disciplinas, incluyendo mecánica, electrónica, ciencias biomédicas y ciencias de la computación. Este capítulo revisa los principales enfoques de control para prótesis, con énfasis en aquellos basados en *aprendizaje por refuerzo*, y el rol de la visión por computador en la detección y estimación de la posición de objetos para facilitar la captura.

2.1. Métodos de control para prótesis

A lo largo de la historia, el control de prótesis ha evolucionado desde dispositivos pasivos hasta sistemas con sensores y actuadores de alta precisión. Al día de hoy, se investigan métodos de control basados en bioseñales, reconocimiento de patrones y algoritmos de inteligencia artificial [1, 3].

2.1.1. Control Convencional

El término “control convencional” suele abarcar prótesis mecánicas o de accionamiento simple, donde el usuario gestiona de forma directa la apertura y cierre de la garra o mano protésica mediante sistemas de cableado o dispositivos electromecánicos de muy baja complejidad [8]. Si bien estos sistemas ofrecen robustez y facilidad de mantenimiento, carecen de la capacidad para reaccionar a la *intención* del usuario en

tiempo real de manera automatizada.

2.1.2. Control Mioeléctrico y bioseñales

La electromiografía de superficie (sEMG) es una técnica ampliamente estudiada para el control de prótesis, en la cual se captan señales eléctricas del músculo residual para inferir el comando motor que desea realizar el usuario [2, 5, 6]. Por ejemplo, Fougner et al. [5] abordaron la compensación de la posición del miembro para mejorar la exactitud del reconocimiento de patrones. Zhuang et al. [2] revisan métodos recientes que apuntan a un control más natural e intuitivo, mientras que Liu et al. [6] discuten cómo las técnicas de *deep learning* pueden incrementar la precisión y robustez ante variaciones de la señal mioeléctrica.

Aunque estos métodos mioeléctricos han logrado notables avances clínicos, también presentan desafíos. Entre ellos, la necesidad de un entrenamiento y calibración manual, la influencia de la fatiga muscular y la variabilidad inter-usuario [4].

2.1.3. Reinervación Muscular Dirigida

Otro avance relevante en el control protésico es la Reinervación Muscular Dirigida (del inglés *Targeted Muscle Reinnervation, TMR*). Este es un procedimiento quirúrgico en el que los nervios motores que antes controlaban la extremidad amputada se reconnected a músculos residuales o zonas específicas del cuerpo del paciente. Al realizar esta “reinnervación”, las señales nerviosas que originalmente iban hacia la mano ahora activan estos músculos reubicados, generando señales mioeléctricas más fuertes y distinguibles. Estas señales amplificadas pueden luego emplearse para controlar una prótesis de manera más intuitiva, utilizando los mismos principios del control mioeléctrico convencional pero con múltiples sitios de señal. Estudios clínicos reportan que la TMR puede mejorar el número de movimientos controlables y la naturalidad del control en pacientes amputados. Sin embargo, se trata de una técnica invasiva que requiere cirugía y rehabilitación, y no elimina por completo problemas como la fatiga o el ruido en las señales. Por tanto, si bien la TMR amplía las posibilidades del control basado en bioseñales, persiste la necesidad de enfoques complementarios que brinden un control aún más automático y robusto de la prótesis.

2.2. Inteligencia Artificial y Redes Neuronales

La Inteligencia Artificial (IA) es un campo de la informática que se dedica al desarrollo de sistemas capaces de realizar tareas que típicamente requieren inteligencia humana. Esto abarca problemas de percepción, toma de decisiones, aprendizaje y razonamiento. Dentro de la IA, un subcampo fundamental es el del aprendizaje automático, donde los algoritmos permiten a las máquinas aprender patrones a partir de datos en lugar de ser explícitamente programadas para cada tarea. En años recientes ha cobrado especial relevancia el aprendizaje profundo, que utiliza arquitecturas complejas de redes neuronales para extraer y representar características de alto nivel en los datos, logrando resultados sobresalientes en visión por computador, procesamiento de lenguaje natural, control robótico y otros dominios.

Por su parte, las redes neuronales artificiales son modelos computacionales inspirados en el sistema nervioso biológico. Están compuestas por capas de neuronas artificiales interconectadas, donde cada neurona realiza una operación matemática sencilla (una combinación lineal de sus entradas seguida de una función no lineal). A través de múltiples capas, una red neuronal puede aproximar funciones muy complejas. Durante un proceso de entrenamiento, la red ajusta los pesos sinápticos (los parámetros de conexión entre neuronas) para minimizar el error en la tarea que se le haya asignado, por ejemplo, clasificar imágenes o tomar decisiones de control. Este ajuste se logra típicamente con algoritmos de optimización como el descenso por gradiente, y requiere de muchos ejemplos (datos) para especializar la red en la tarea.

Un tipo particular de aprendizaje automático es el aprendizaje por refuerzo detallado en el capítulo 2.2.1. Este enfoque es muy útil para problemas de control donde no se dispone de ejemplos directos de la acción correcta, sino de un criterio para evaluar qué tan buena fue una acción.

En el contexto de este proyecto, se combinan estas ideas: utilizamos redes neuronales profundas tanto para interpretar la visión por computador como para tomar decisiones de control mediante aprendizaje por refuerzo profundo. Gracias al uso de redes neuronales entrenadas, nuestra prótesis obtiene la capacidad de entender su entorno visualmente y aprender a actuar en consecuencia.

2.2.1. Aprendizaje por Refuerzo

El *Aprendizaje por Refuerzo* (AR) es un paradigma de la inteligencia artificial que permite a un agente aprender una política de control a partir de la interacción con un entorno, recibiendo recompensas o penalizaciones [9]. En robótica, el AR se ha empleado con éxito para tareas de manipulación y control de brazos robóticos [10].

2.2.2. Fundamentos

En contraste con los métodos supervisados (donde se dispone de un conjunto grande de ejemplos etiquetados), el AR se basa en la búsqueda de acciones que maximicen la recompensa acumulada [11]. Para ello, se definen:

- **Estados (observaciones):** información percibida del entorno (p.ej., imágenes, posiciones articulares).
- **Acciones:** posibles comandos que puede ejecutar el agente.
- **Recompensa:** retroalimentación inmediata o diferida que indica la “bondad” de la acción.

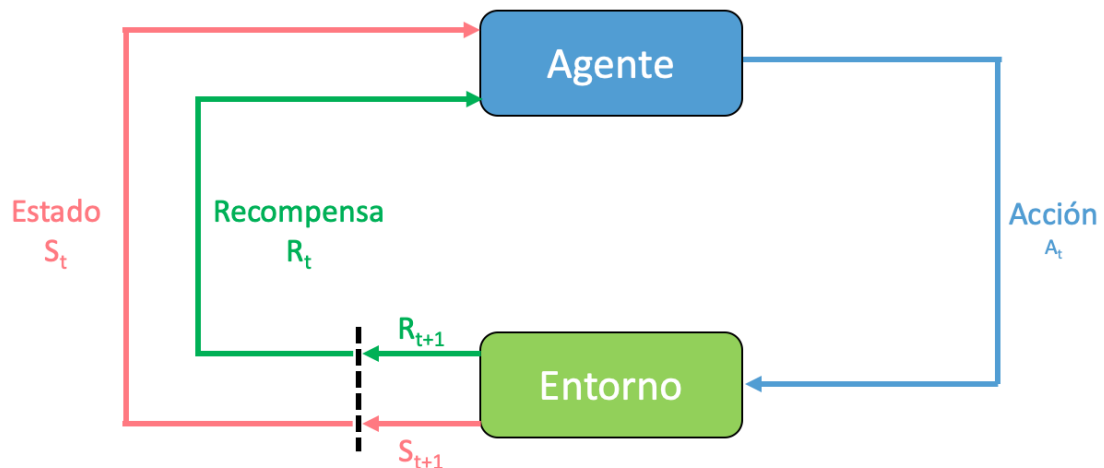


Figura 2.1: Diagrama conceptual del ciclo de aprendizaje por refuerzo

En la Figura 2.1 se ilustra el ciclo típico de un agente inteligente aprendiendo a controlar la prótesis mediante refuerzo. El agente (algoritmo de control) percibe el estado actual

del entorno (por ejemplo, imagen de profundidad con el objeto y el gripper) y decide una acción (por ejemplo, cerrar o no el gripper). Tras ejecutar la acción, recibe una recompensa en función del resultado (véase SubSección 4.3) y el entorno transita a un nuevo estado. Este bucle de estado $\rightarrow$ acción $\rightarrow$ recompensa se repite iterativamente durante el entrenamiento, permitiendo que el agente ajuste su estrategia para maximizar las recompensas.

2.2.3. Aplicaciones en Control de Prótesis

Aunque la literatura sobre AR en prótesis de mano es aún incipiente, se han reportado trabajos que combinan simulaciones físicas con transferencia posterior al hardware real. Levine et al. [10] demostraron la efectividad del *deep reinforcement learning* para mejorar la coordinación ojo-mano en un robot de agarre, dejando el precedente de que esta misma lógica puede aplicarse en sistemas protésicos con requerimientos similares de precisión y adaptabilidad.

2.3. Visión por Computador

Para que una prótesis pueda identificar la intención de capturar un objeto, resulta fundamental contar con un sistema de visión que reconozca la presencia de objetos y estime su posición relativa respecto a la mano protésica [7]. En los últimos años, han tomado relevancia las redes neuronales profundas (CNN) para tareas de detección, segmentación y estimación de profundidad.

2.3.1. Algoritmo de Detección de Objetos: YOLO

El *framework You Only Look Once* (YOLO) [12] es uno de los referentes en detección de objetos en tiempo real. Se caracteriza por:

- Procesar la imagen completa en una única pasada (*single shot*).
- Predecir *bounding boxes* y clases simultáneamente, lo que lo hace eficiente para aplicaciones en tiempo real.



Figura 2.2: Extracto de video con utilización de Yolo v4 en tiempo real

Al observar la Figura 2.2, podemos visualizar como con el algoritmo de Yolo v4 se pueden detectar clases de objetos en entornos bastante ruidosos, lo que lo hace especial en su uso para este trabajo de memoria permitiendo escalar a trabajos futuros en ambientes no controlados.

En el contexto de una prótesis, YOLO permite identificar si hay un objeto dentro de la “zona de agarre” de la mano, o bien si el espacio está libre [7]. Para robustecer la detección, se suelen entrenar modelos específicos con imágenes del entorno real, etiquetando el objeto y la propia garra protésica.

2.3.2. Algoritmo de Segmentación: DEPTH ANYTHING

Aunque existen cámaras RGB-D que capturan directamente mapas de profundidad, no siempre están disponibles. *Depth Anything* [13] surge como una alternativa de la familia de métodos de segmentación y estimación de profundidad que, a partir de una imagen RGB, infieren la distancia aproximada de cada píxel a la cámara.



Figura 2.3: Comparación entre la imagen RGB original (izquierda) y el mapa de profundidad estimado por *Depth Anything* (derecha).

Tal como se observa en la Figura 2.3, el modelo es capaz de reconstruir la geometría aproximada de la escena, identificando regiones más cercanas o más lejanas a la cámara.

Si bien *Depth Anything* no provee la exactitud milimétrica de un sensor dedicado, resulta particularmente útil para:

- Generar *depth maps* en tiempo real, sin requerir hardware adicional costoso.
- Facilitar la integración con algoritmos de *deep reinforcement learning*, donde la información de profundidad puede ser más relevante que el color para tareas de agarre.

Las aproximaciones recientes a la segmentación universal (*segment anything* [14]) y la “profundidad universal” parten de redes masivas entrenadas con datasets muy amplios, abarcando múltiples escenarios. Esto permite una transferencia razonable de entornos simulados a entornos reales, cuestión de alta relevancia en prótesis robóticas [7, 11].

El control de prótesis de mano ha progresado desde métodos puramente mecánicos y mioeléctricos hacia enfoques más inteligentes, que incorporan aprendizaje por refuerzo y visión por computador para la detección automática de objetos. Herramientas como YOLO y Depth Anything facilitan el reconocimiento y la estimación de profundidad, respectivamente, sentando las bases de una prótesis capaz de “interpretar” su entorno y ejecutar acciones de agarre de manera más autónoma y confiable. En el siguiente capítulo, se describirá la Metodología propuesta para desarrollar e integrar estos componentes, de manera que se cumplan los objetivos de la presente investigación.

Capítulo 3

Metodología

La presente investigación se llevó a cabo con el objetivo de diseñar e implementar un sistema de control para prótesis de mano, capaz de detectar un objeto mediante visión artificial y ejecutar la acción de captura utilizando una red neuronal entrenada con aprendizaje por refuerzo. Para cumplir con este objetivo, se planificaron tres grandes etapas:

1. Diseño y configuración de un entorno de simulación (basado en *PyBullet*) donde entrenar el algoritmo de control.
2. Implementación de un entorno físico controlado y replicable, que permita trasladar y comparar el desempeño del sistema entrenado virtualmente.
3. Protocolo de validación experimental para cuantificar la precisión y robustez del sistema en ambos entornos.

A continuación, se describen los aspectos metodológicos que guiaron cada etapa.

3.1. Diseño general de la investigación

La estrategia metodológica contempló:

- Revisar y adoptar herramientas de aprendizaje por refuerzo que permitan la toma de decisiones a partir de imágenes.

- Incorporar un módulo de visión por computador que detecte el objeto y estime su posición, a fin de desencadenar o no el cierre de la prótesis.
- Simular inicialmente todo el proceso (robot + visión) en un entorno virtual (Py-Bullet) para agilizar iteraciones de entrenamiento.
- Reproducir en la realidad las condiciones de la simulación, con una cámara y un modelo físico de garra o prótesis.
- Definir métricas de validación y un protocolo experimental para verificar la eficacia de la aproximación.

La Figura 3.1 muestra el esquema metodológico general seguido para alcanzar los objetivos planteados, incluyendo la progresión entre simulación, integración de módulos, pruebas físicas y validación experimental.

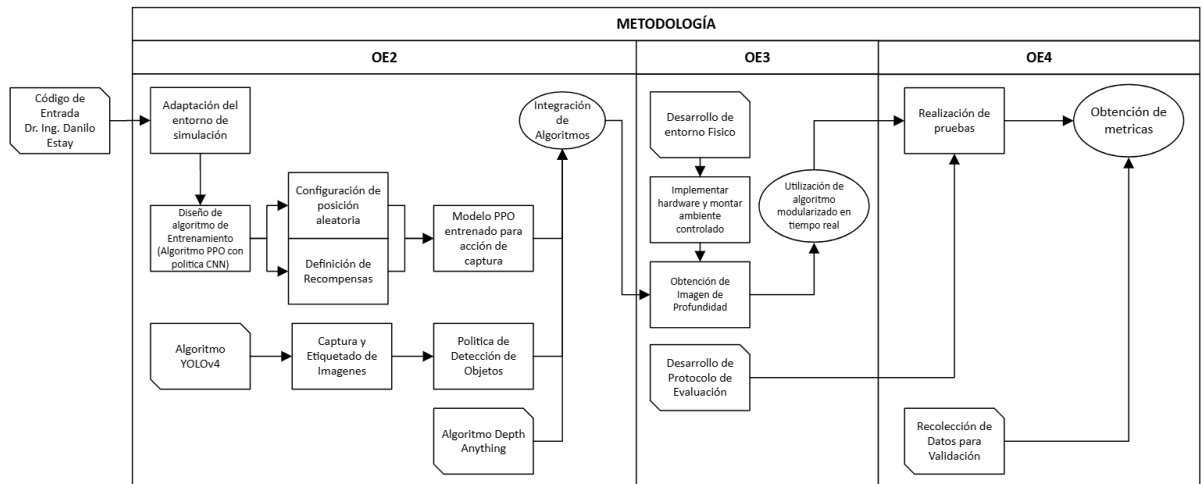


Figura 3.1: Diagrama de flujo metodológico.

Los detalles de implementación, configuración de parámetros y resultados se presentan en los capítulos 4 y 5.

3.2. Herramientas de desarrollo

Se seleccionaron las siguientes herramientas por su solidez, documentación disponible y eficacia comprobada en tareas relacionadas con visión por computador, simulación

física y aprendizaje por refuerzo profundo. A continuación se detalla la elección de cada una:

- **Python:** Se utilizó como lenguaje principal debido a su sintaxis accesible, amplia comunidad y compatibilidad con bibliotecas de inteligencia artificial, robótica y visión computacional. Python ha sido ampliamente adoptado en entornos académicos y de investigación, siendo el lenguaje base de frameworks como TensorFlow, PyTorch y OpenAI Gym [15].
- **PyBullet:** Este motor de simulación de física permite modelar cuerpos rígidos, colisiones, articulaciones y sensores. Fue seleccionado por su integración nativa con Python, bajo consumo computacional y facilidad para simular robots manipuladores en entornos 3D [16]. Además, PyBullet permite la extracción directa de imágenes RGB y mapas de profundidad, elementos clave para el entrenamiento basado en visión [17].
- **Stable-Baselines3:** Es una librería basada en PyTorch para implementar algoritmos de aprendizaje por refuerzo profundo. Se eligió por su código limpio, mantenido activamente, y su compatibilidad con PyBullet y Gym. Incluye implementaciones robustas de PPO, SAC y otros algoritmos modernos con soporte para entrenamiento paralelo y herramientas de monitoreo [18]. Fue preferida por su estabilidad y facilidad de ajuste de hiperparámetros en entornos personalizados.
- **YOLO:** El algoritmo You Only Look Once (YOLO) fue escogido por su alta precisión y velocidad en tareas de detección de objetos en tiempo real [19]. Su arquitectura de red convolucional permite realizar predicciones a partir de una única pasada por la imagen, lo cual es esencial para aplicaciones en robótica. Se utilizó YOLOv4 por su equilibrio entre rendimiento y disponibilidad de herramientas compatibles.
- **Depth Anything:** Esta herramienta fue empleada para generar mapas de profundidad a partir de imágenes RGB. Al no contar con cámaras físicas de profundidad en la etapa experimental, se utilizó este modelo para estimar profundidad con alta fidelidad [13]. Su uso permite emular los datos que en PyBullet se obtienen directamente, cerrando la brecha entre simulación y entorno real.

3.3. Plan de recolección de datos

La recolección de datos se dividió entre **datos de simulación** y **datos del entorno físico**. El plan contempló:

1. Generar un conjunto amplio de imágenes sintéticas en PyBullet, con variaciones en la posición del objeto y del gripper, y asociar cada imagen a la acción tomada por el agente (se describen detalles de la configuración en la Sección 4.2 del Capítulo 4).
2. Entrenar YOLO con un conjunto reducido de imágenes etiquetadas de la garra y el cubo en el mundo real, creando un *dataset* manual. Esto debido a que se interactúa en un ambiente controlado, por lo que se espera buen desempeño con este bajo banco de imágenes.
3. Aplicar un protocolo de validación con un banco de imágenes independientes (cerca de 500) para medir el desempeño de YOLO y de la política de red neuronal en condiciones reales.

3.4. Plan de validación experimental

Con el fin de medir el desempeño y robustez del sistema, se definieron cuatro grandes ejes de validación:

- **Comparación cualitativa** entre el entorno virtual y el real (posiciones relativas, iluminación, escalas).
- **Evaluación de YOLO**, midiendo métricas de precisión, recall y *mean Average Precision* (mAP) en la detección del objeto.
- **Evaluación del algoritmo de control**, registrando tasas de éxito, tasas de error (falsos positivos y omisiones) y analizando la capacidad de transferir el aprendizaje de la simulación a la realidad.
- **Rendimiento en tiempo real**, valorando la capacidad de captura de la imagen y procesamiento en tiempo inmediato.

Los resultados de estas evaluaciones se presentan y discuten en el Capítulo 5.

3.5. Otros aspectos metodológicos

- Se empleó control de versiones y seguimiento de experimentos con TensorBoard, a fin de documentar la evolución del entrenamiento de la red neuronal.
- Se definieron hiperparámetros iniciales (tasa de aprendizaje, tamaño de *batch*, entre otros) con base en la literatura previa [10] y se afinaron durante una etapa preliminar de pruebas (véase Capítulo 4).
- Se priorizó modularidad en el código para permitir la fácil sustitución de componentes (por ejemplo, cambiar YOLOv4 a YOLOv5, o Depth Anything a un sensor físico de profundidad).

En síntesis, la metodología define el plan general para abordar la investigación, y los aspectos detallados de configuración, implementación y resultados se describen en los capítulos posteriores.

Capítulo 4

Desarrollo de Sistema de Control

En este capítulo se describen en detalle las etapas de implementación y desarrollo realizadas para materializar la metodología planteada en el Capítulo 3. Se profundiza en la configuración de *PyBullet*, la integración de los algoritmos de *aprendizaje por refuerzo* (AR), la implementación de *YOLO* y el ensamblaje final de todos los componentes en un único pipeline funcional.

4.1. Arquitectura General

La arquitectura general del sistema puede verse como una cadena de procesamiento que inicia con la captura de imágenes (o la generación de *render* en el simulador), continúa con la detección del objeto, y finaliza con la orden al actuador mecánico. Para mantener la claridad, se separan los módulos en:

1. **Módulo de Simulación (PyBullet).**
2. **Módulo de Aprendizaje por Refuerzo (PPO).**
3. **Módulo de Visión por Computadora (YOLO).**
4. **Integración y Control Final.**

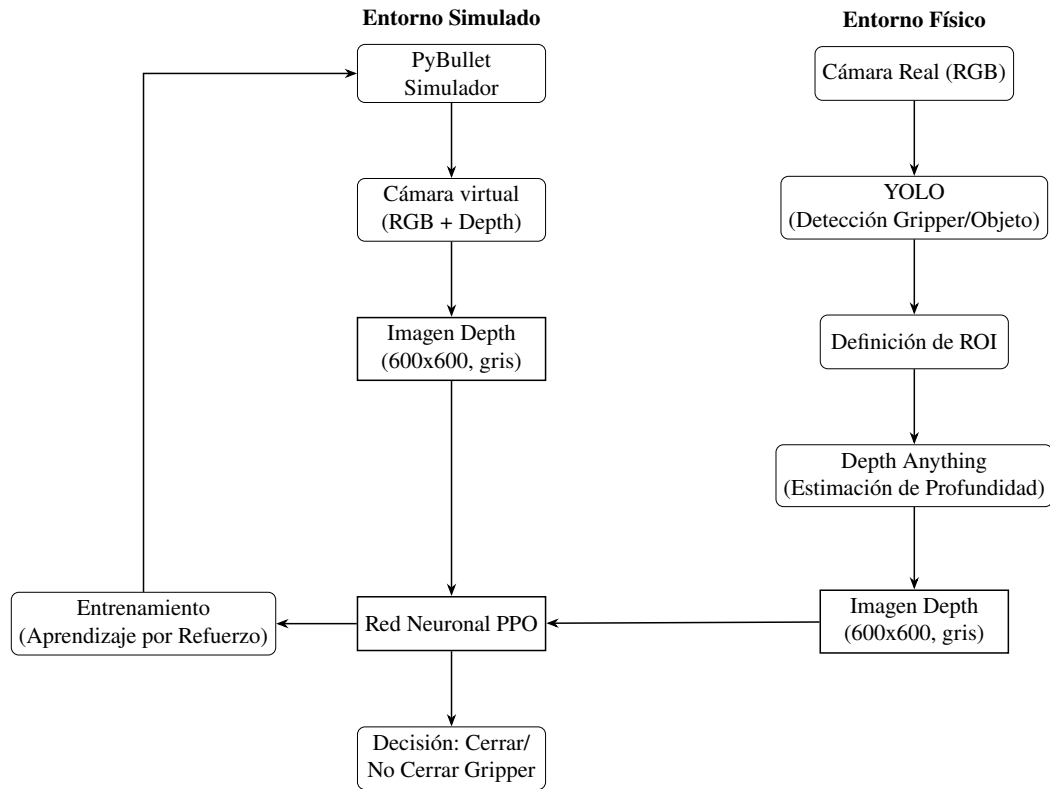


Figura 4.1: Diagrama de bloques del flujo de información entre los módulos.

La Figura 4.1 muestra un diagrama de bloques que ilustra cómo se comunican estos módulos. En la simulación, PyBullet entrega directamente la imagen de profundidad al modelo PPO. En el entorno real, se requiere detección con YOLO y estimación de profundidad con Depth Anything antes de ingresar la imagen al modelo.

4.2. Configuración del Entorno de Simulación con Py-Bullet

Para la primera etapa de entrenamiento y pruebas iniciales, se configuró un entorno en *PyBullet* que reprodujese fielmente un escenario controlado de interacción entre una garra robótica y un cubo:

- **Archivo de escena URDF:** Se creó un modelo del *gripper* y del cubo, utilizando los archivos reales para su posterior comparación en el entorno físico real.

- **Rango de posiciones aleatorias:** Se estableció que el cubo permaneciera con posición fija, mientras que el gripper se ubicaba en posiciones y orientaciones aleatorias en el rango $[-0,35; 0,35]$ en X e Y, y $[0,05; 0,5]$ en Z.
- **Captura de profundidad:** PyBullet provee una cámara virtual que genera un *buffer* de profundidad. Dicho *depth map* se recorta a 600x600 px y se normaliza antes de pasarlo al módulo de AR.
- **Parámetros de simulación:** Se definió una tasa de refresco de 240 Hz para garantizar una física estable, y se incluyó detección de colisiones con muros y suelo para “reubicar” el gripper cuando ocurriese una colisión no deseada.

Para garantizar reproducibilidad, se desarrollaron los scripts en Python de manera modularizada a modo de poder establecer las configuraciones necesarias.

4.2.1. Número de Iteraciones y Episodios de Entrenamiento

Con base en recomendaciones de investigación [10], se planeó originalmente entrenar 100,000 iteraciones. No obstante, tras un análisis preliminar, se observó que a las 55,000 iteraciones se alcanzaba una convergencia estable en el entorno simulado, con una tasa de éxito promedio cercana al 95 %. Se detallan estos criterios en la Sección 4.3.1.

4.3. Módulo de Aprendizaje por Refuerzo (PPO)

Se implementó el algoritmo **Proximal Policy Optimization (PPO)** utilizando la librería *Stable-Baselines3*. A continuación, se especifican algunos de los parámetros críticos:

- **Red Convolutacional (CNN):**
 - Tamaño de entrada: 600x600x1 (escala de grises).
 - Normalización previa al ingreso de datos.
- **Parámetros de PPO:**

- `learning_rate`: 3×10^{-4}
- `batch_size`: 256
- `n_steps` por iteración: 2048

■ **Recompensa (Reward):**

- +1: Si el agente decide cerrar la garra con el cubo dentro de la ROI.
- -1: Si el agente decide no cerrar la garra y deja pasar la oportunidad de capturar.
- -0,8: Si cierra la garra y no había objeto.
- 0,5: Si el agente decide no cerrar la garra cuando no hay objeto.

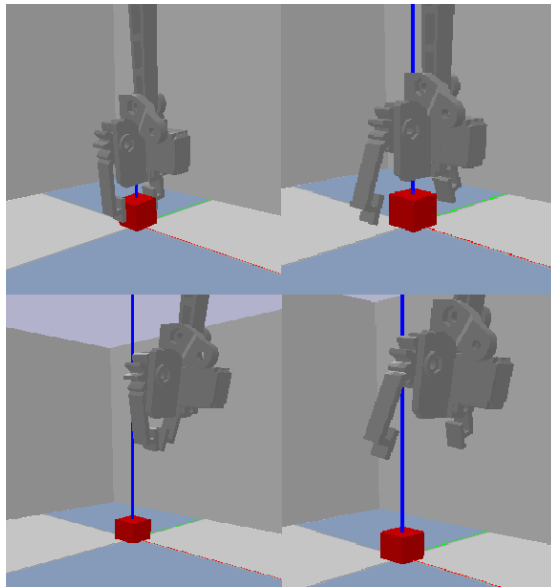


Figura 4.2: Distribución de casos y recompensas del agente. De izquierda a derecha y de arriba hacia abajo.

La Figura 4.2 ilustra las posiciones desde PyBullet, con la obtención de la recompensa correspondiente.

Para facilitar el aprendizaje de la red neuronal, se forzó la aparición de casos positivos durante el entrenamiento. Se definieron como casos positivos aquellas situaciones en las que el *gripper* era capaz de capturar el objeto, con una probabilidad forzada del 70 %.

Para validar esta condición, se implementó un objeto de colisión invisible, ubicado entre las puntas de los dedos del *gripper*. Este objeto, denominado *cilindro fantasma*, permitía detectar si el cubo se encontraba dentro de la región de interés (ROI) para una posible captura. Si el cilindro colisionaba con el cubo, se consideraba que la condición de captura estaba cumplida.

En la Figura 4.3 se ilustra el cilindro fantasma representado en color verde, mostrado únicamente con fines de visualización.

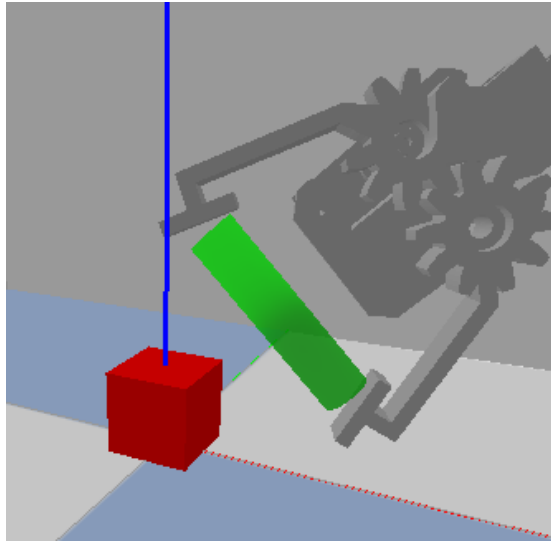


Figura 4.3: Visualización del cilindro fantasma (verde translúcido) ubicado entre los dedos del *gripper*. Su colisión con el cubo indica la posibilidad de captura exitosa.

4.3.1. Criterios de Convergencia

Se evaluó la política actual cada 2048 iteraciones, midiendo la tasa de aciertos en un *mini set* de 256 episodios. Al superar consistentemente el 90 % de éxito, se consideró que el agente había convergido en la tarea simulada.

4.4. Módulo de Visión por Computadora

Para la detección de objetos (cubo y posición del gripper) en imágenes reales, se utilizó el algoritmo YOLO. A continuación, se detallan los principales pasos de implementación:

4.4.1. Dataset de Entrenamiento

- Se recopilaron 200 imágenes del entorno real, donde se anotó la posición del gripper y del cubo.
- El dataset se dividió en 80 % para entrenamiento y 20 % para validación interna como lo recomienda UltraLytics en su documentación.

4.4.2. Configuración e Hiperparámetros

- **Versión de YOLO:** v4, con anclas (anchors) adaptadas al tamaño esperado de la garra y el cubo.
- **Batch size:** 16
- **Learning rate:** 10^{-3} con *scheduler* que reduce a la mitad cada 10 épocas si la pérdida no mejora.

4.4.3. Integración con la ROI

Tras la detección, se localizan las coordenadas (x1, y1, x2, y2) del rectángulo que delimita la garra. Luego, se define una ROI alrededor de este rectángulo. Si se detecta el cubo dentro de esa ROI, se genera una señal que activa el paso de la imagen al módulo de AR.

4.4.4. Algoritmo Depth Anything

Con el fin de contar con información de profundidad en el entorno real (similar a los mapas de profundidad que se obtienen en PyBullet), se integró la herramienta *Depth Anything* [13]. A continuación, se describen los puntos clave de esta implementación.

Alcance Dado que en la simulación los mapas de profundidad son generados directamente por PyBullet, en el entorno físico se requiere un modo de estimar la profundidad del objeto y el gripper sin recurrir a hardware de captura 3D especializado (como cámaras RGB-D). **Depth Anything** permite aproximar esta información a partir de imágenes

RGB convencionales, aprovechando modelos entrenados en grandes volúmenes de datos.

Integración en el *pipeline*

- **Entrada:** Recibe la imagen RGB capturada por la cámara Logitech (resolución 1280x720).
- **Salida:** Entrega un mapa de profundidad estimado, con la misma resolución que luego puede recortarse a 600x600 px para alimentar la red convolucional de AR.
- **Coordinación con YOLO:** YOLO se encarga de localizar el cubo y el gripper mediante *bounding boxes*. Una vez que se confirma que el cubo está dentro de la Región de Interés (ROI) del gripper, se procede a generar el mapa de profundidad correspondiente a esa región con *Depth Anything*.

Limitaciones Es importante mencionar que *Depth Anything* ofrece una aproximación de la profundidad, pero no una medida absoluta exacta. Al no utilizar un sensor real de profundidad, no se garantiza una exactitud milimétrica. No obstante, para la tarea de *cerrar/no cerrar* la garra, basta con una estimación de la forma y posición relativa del objeto.

En la fase de validación (Capítulo 5), no se llevarán a cabo mediciones cuantitativas específicas para *Depth Anything* de la profundidad estimada). Se usará principalmente como fuente de *depth map* para la red neuronal, asumiendo que cualquier error de estimación se reflejará en la tasa de éxito global del sistema.

Para ilustrar las similitudes entre ambas fuentes de entrada al sistema, en la Figura 4.6 se comparan dos imágenes utilizadas por la red neuronal: una obtenida mediante *Depth Anything* a partir de una imagen RGB real, y otra generada desde el entorno simulado en *PyBullet*. Ambas son recortadas a un tamaño de 600x600 píxeles en escala de grises antes de ser utilizadas por el modelo.



Figura 4.4: (a) Imagen estimada desde entorno real con *Depth Anything*

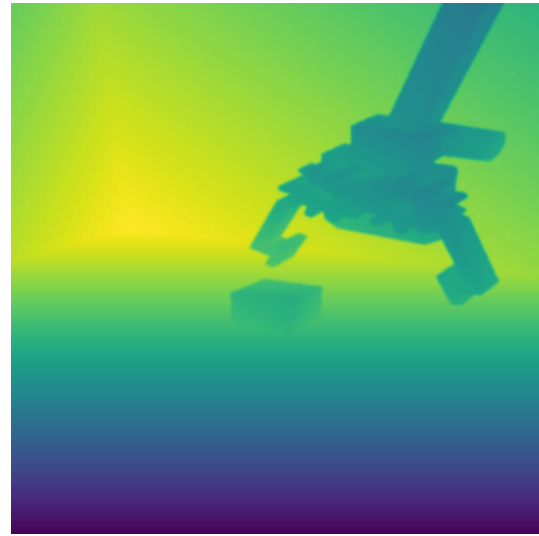


Figura 4.5: (b) Imagen generada desde *PyBullet* (600x600, escala de grises)

Figura 4.6: Comparación visual entre las imágenes de profundidad utilizadas como entrada por la red neuronal. Se observa una similitud estructural entre el entorno real (a) y el entorno simulado (b), lo cual facilita la generalización del modelo entrenado.

4.5. Integración Final en un Pipeline

Finalmente, se creó un archivo Python principal (`main.py`) que orquesta la comunicación:

1. **Captura de imagen:** desde la cámara Logitech.
2. **Detección con YOLO:** identifica el gripper y el cubo, extrae la ROI.
3. **Generación/preparación de imagen de profundidad:** Se emula con *Depth Anything* (ver Sección 4.4.4 del capítulo).
4. **Llamado al modelo PPO:** la CNN procesa la imagen y decide “cerrar” o “no cerrar”.
5. **Accionamiento:** si la decisión es “cerrar”, se envía la orden al microcontrolador para el gripper.

Capítulo 5

Validación Experimental

En este capítulo se describe el protocolo de pruebas y validaciones implementado para evaluar la efectividad del sistema descrito en el Capítulo 4. El énfasis aquí está en el procedimiento seguido para validar el comportamiento en condiciones controladas, pero sin entrar aún en los resultados numéricos (los cuales se discutirán en el Capítulo 6).

5.1. Generación de Ambiente Controlado

5.1.1. Montaje Físico

Se preparó un entorno físico que emula las características del simulador:

- **Entorno de fondo blanco:** Paneles de espuma forradas a modo de simular las murallas del entorno virtual.
- **Cubo real:** Construido con cartón de dimensión 1.5x1.5x1.5 cm, forrado en color rojo para asemejar el cubo virtual.
- **Iluminación:** lámpara LED orientada hacia uno de los muros para evitar luz directa.
- **Cámara Logitech 1280x720:** Fijada en un trípode, reproduciendo la altura y ángulo de la cámara virtual de PyBullet.

En la Figura 5.1 es posible ver el montaje real.



Figura 5.1: Montaje físico del sistema de prueba. Se observa el gripper robótico montado frente al objeto, así como la cámara RGB encargada de capturar las imágenes de entrada al sistema.

El protocolo de validación se diseñó para confirmar si el sistema es capaz de:

1. **Detectar el cubo** en el ROI del gripper, utilizando YOLO, en diversas condiciones de posición y distancia.
2. **Generar la imagen de profundidad** (simulada, en este caso) y procesarla con la CNN entrenada para decidir o no la acción de captura.

5.1.2. Evaluación Cualitativa

Se tomó un conjunto de casos de prueba donde se analiza visualmente si:

- El ángulo de la cámara en la realidad coincide con la proyección definida en PyBullet.

- La detección del cubo por YOLO coincide con la posición real del objeto.
- La profundidad simulada o la imagen generada posee un “ruido” aceptable, para no confundir a la red neuronal.

5.1.3. Pruebas en Tiempo Real

Para **pruebas en tiempo real**, se generó un *script* que:

1. Inicia el flujo de video.
2. En cada frame, corre YOLO y determina si el cubo está presente en la ROI.
3. Si lo está, extrae la imagen de profundidad (simulada) y la pasa al modelo PPO.

Se repitió la secuencia con distintas orientaciones y posiciones del gripper para obtener una métrica de medición de precisión.

5.2. Criterios de Evaluación

- **Exactitud de Detección (mAP-YOLO):** proporción de *frames* en los cuales se detecta correctamente el cubo dentro de la ROI.
- **Tasa de Éxito de Captura:** número de cierres correctos sobre el cubo dividido por el número de intentos totales.
- **Tasa de Falsas Capturas:** veces que la garra se cierra sin haber objeto presente en la ROI.
- **Reality Gap:** pérdida de conocimiento en la transferencia de información entre entorno virtual y real.

Capítulo 6

Resultados

6.1. Desempeño del Entrenamiento

Durante el proceso de entrenamiento, se realizaron un total de 55,000 iteraciones, momento en el cual se observó una convergencia en los resultados, lo que indicaba que el modelo había aprendido una política efectiva para la tarea planteada. Adicionalmente, se enfatiza en la duración aproximada de 55 horas en la duración de estas iteraciones lo cual indica un coste computacional elevado para el hardware que se estaba usando.

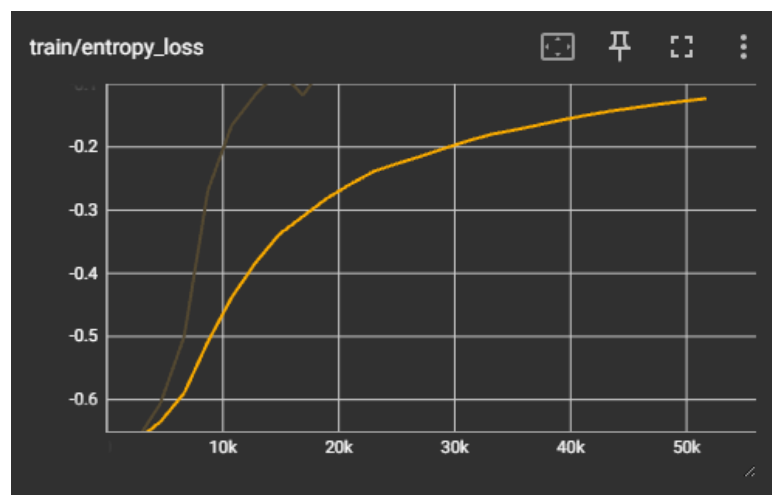


Figura 6.1: Evolución de la pérdida de entropía durante el entrenamiento.

En la Figura 6.1 se observa que esta métrica comienza con valores bajos y experimenta un aumento progresivo, lo que sugiere una mayor exploración en las primeras etapas

del entrenamiento y una estabilización a medida que el agente optimiza su estrategia.

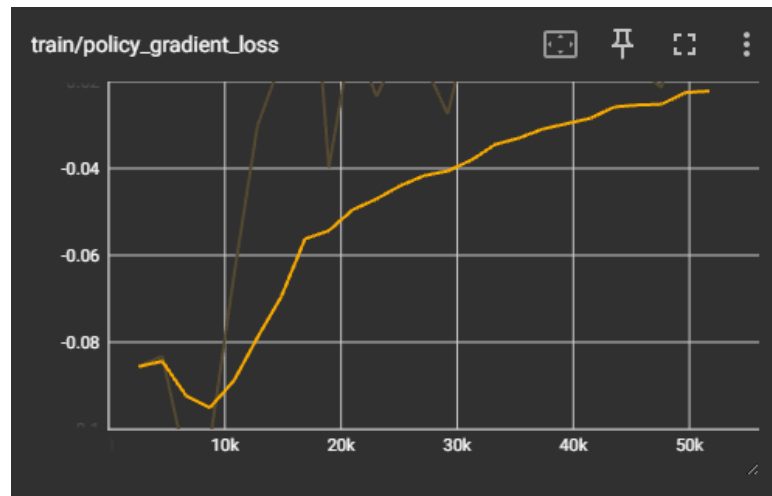


Figura 6.2: Evolución de la pérdida del gradiente de la política.

La Figura 6.2 muestra la evolución de la pérdida del gradiente de la política. En las primeras iteraciones, se presentan fluctuaciones, pero con el avance del entrenamiento, se observa una tendencia de crecimiento progresivo hasta alcanzar una estabilidad relativa. Este comportamiento indica que el modelo ha ajustado gradualmente su política, lo que respalda el aprendizaje eficiente del agente.

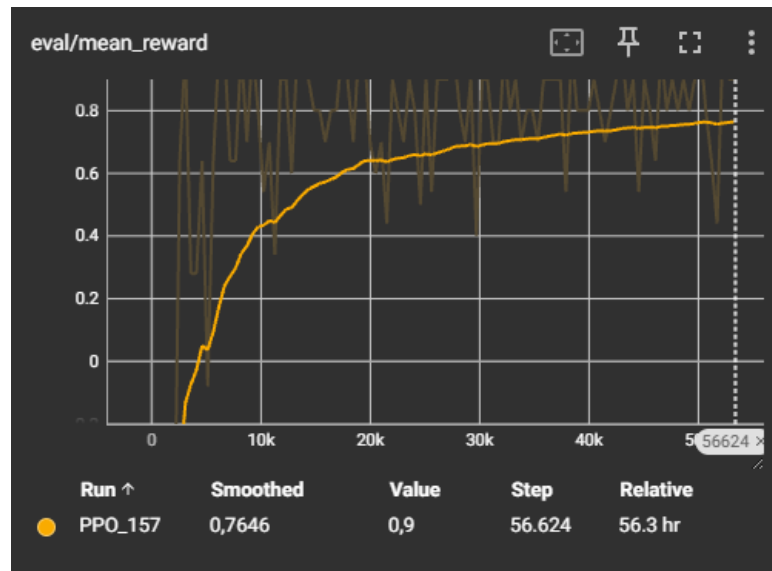


Figura 6.3: Recompensa promedio obtenida durante la evaluación.

Finalmente, el desempeño del modelo durante la evaluación se refleja en la Figura

6.3, donde se presenta la recompensa promedio obtenida a lo largo de las iteraciones. Se observa una tendencia creciente en la recompensa, alcanzando valores cercanos a 0.9 en las últimas iteraciones, lo que indica que el agente ha aprendido una política efectiva para la tarea asignada. A pesar de algunas variaciones, la estabilidad en los valores altos de la recompensa confirma que el modelo ha convergido satisfactoriamente. Hay que considerar que el rango de valores para la premiación están presente entre -1 y 1, por lo que valores cercanos a 0.9 demuestran la convergencia.

Es importante notar que el archivo final que contiene la política de entrenamiento en su totalidad tiene un peso de archivo en el orden de 2GB, lo que refleja la complejidad del modelo aprendido y la cantidad de información almacenada para su implementación.

A modo de salida se genera una carpeta con un campo de imágenes obtenidas desde el entrenamiento realizado. Esta carpeta posee alrededor de 30,000 imágenes.

6.2. Desempeño del Algoritmo en Simulaciones

En esta sección se presentan los resultados obtenidos al probar el algoritmo propuesto en un entorno de simulación. El entrenamiento se llevó a cabo utilizando un entorno robótico simulado en PyBullet junto con la política evaluada en la sección 6.1.

En las pruebas realizadas con el modelo entrenado, se registró la frecuencia de los valores de recompensa obtenidos durante la simulación que contó con 2500 iteraciones con una duración de aproximadamente 5 horas. Los resultados mostraron que en 87 ocasiones se obtuvo una recompensa de -0.8, en 1803 ocasiones se alcanzó una recompensa de 0.50, en 579 ocasiones se logró la recompensa máxima de 1.0 y en solo 31 oportunidades se obtuvo la recompensa mínima de -1.0. Esto se visualiza de manera compacta en la Tabla 6.1.

Premio	Cant. de Veces
1.0	579
0.5	1803
-0.8	87
-1.0	31

Cuadro 6.1: Tabla resumen de resultados en prueba simulada.

En esta simulación, los datos se obtuvieron a partir de posiciones completamente

aleatorias, sin modificar la probabilidad para favorecer escenarios de captura del objeto, a diferencia de lo realizado durante el entrenamiento. Como resultado, es esperable que en muchos casos el objeto no esté al alcance, haciendo que la decisión óptima sea abrir el gripper y recibir una recompensa de 0.5.

Para evaluar de manera más precisa el desempeño del algoritmo, se utilizó la tasa de éxito como métrica principal, la cual mide la capacidad del modelo para tomar la decisión correcta, ya sea cerrar o abrir el gripper según la situación presentada. Adicionalmente, se definieron dos métricas complementarias: Cerrado Negativo y Abierto Negativo, que reflejan los casos en los que el modelo tomó decisiones incorrectas, cerrando cuando debía abrir o abriendo cuando debía cerrar, respectivamente.

Métrica	Valor
Tasa de Éxito	95.28 %
Cerrado Negativo	3.48 %
Abierto Negativo	1.24 %

Cuadro 6.2: Tabla con valores de las métricas.

La Tabla 6.2 muestra una tasa de éxito del 95.28 %, lo que indica un desempeño sólido del modelo en la tarea de agarre. Sin embargo, se observa un 3.48 % de casos de “Cerrado Negativo”, donde el gripper se cierra sin tener el objeto al alcance, representando un margen de error que, aunque menor, sigue siendo relevante para mejorar la precisión del sistema. Por otro lado, los casos de “Abierto Negativo” son apenas del 1.24 %, lo que sugiere que las decisiones incorrectas en las que el gripper abre cuando el objeto está disponible son poco frecuentes y no afectan significativamente el rendimiento general.

6.3. Desempeño de YOLO

Para la detección de objetos, en este caso tanto el gripper como el cubo, se empleó la familia de modelos YOLO (*You Only Look Once*), específicamente la versión YOLOv4 de Ultralytics. Se entrenó la red con un conjunto de alrededor de 300 imágenes generadas en el entorno de simulación y alrededor de 100 imágenes extraídas desde la web.

Los resultados de la etapa de validación muestran un valor de *mAP* (mean Average Precision) de aproximadamente 95 %, con las siguientes especificaciones:

- Precisión en la clase *objeto*: 94 %
- Precisión en la clase *gripper*: 96 %

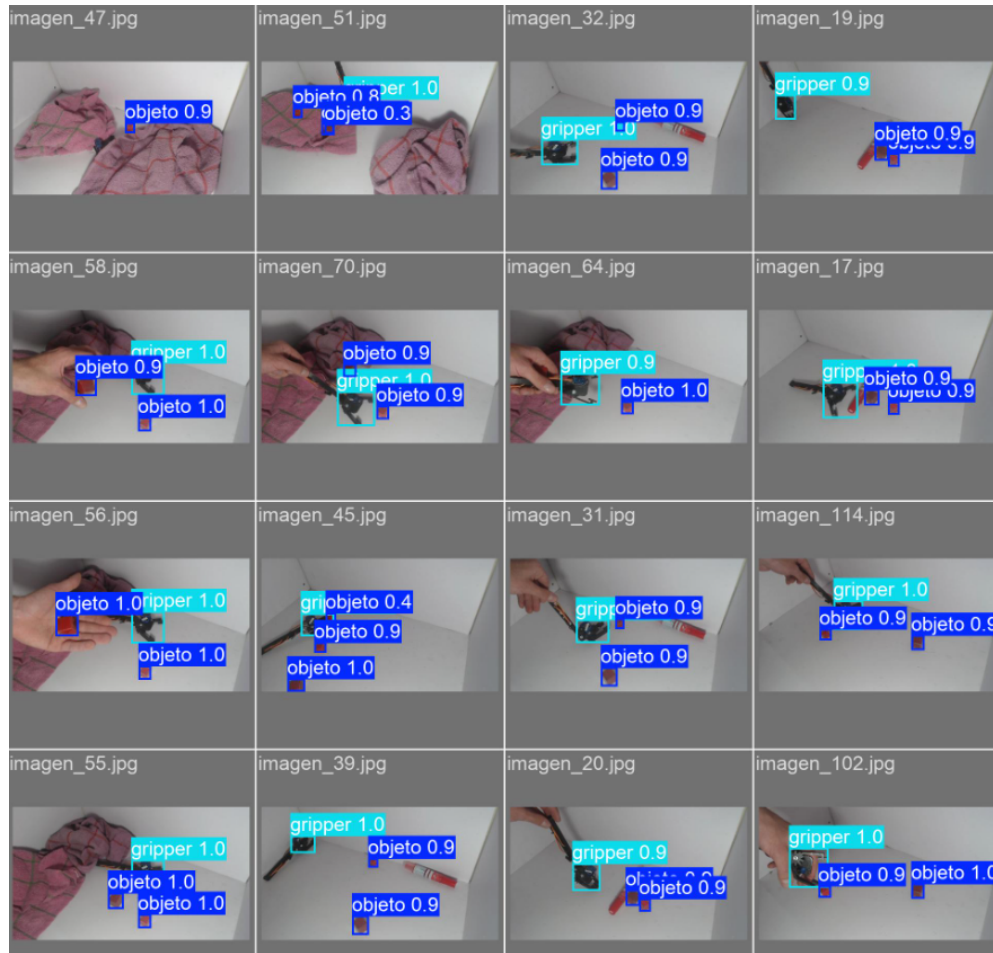


Figura 6.4: Muestras de detección del modelo YOLOv4 aplicadas a distintas imágenes reales del entorno controlado. Se observa correcta identificación del gripper y del objeto bajo diversas condiciones de iluminación y orientación.

El tiempo de inferencia por fotograma se mantuvo cercano a 20 ms, lo cual confirma la viabilidad del uso de YOLO en tiempo real para el presente proyecto.

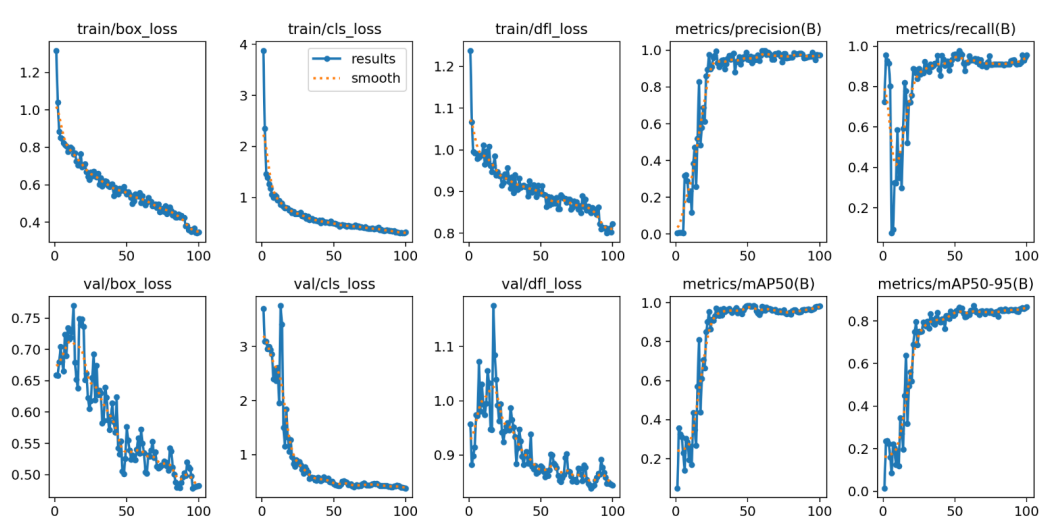


Figura 6.5: Curvas de entrenamiento y validación del modelo YOLOv4. Se muestra la evolución de las pérdidas (losses) y las métricas de desempeño como precisión, recall y *mean Average Precision* durante 100 épocas.

En la figura 6.5 se aprecia una convergencia progresiva del modelo, con mejoras constantes en precisión, *recall* y métricas de *mAP*, lo que confirma una adecuada capacidad de generalización. El valor final de *mAP50-95* se aproxima al 83 %.

6.4. Desempeño del Algoritmo Implementado

Tras completar la etapa de entrenamiento y validación en simulaciones, se procedió a integrar la política entrenada en el sistema real.

Las pruebas experimentales incluyeron 500 ejecuciones mediante la captura de imágenes etiquetadas para ser comparado el resultado obtenido con el esperado. Los resultados pueden resumirse en:

- Tasa de éxito promedio en detección y decisión de cierre: 85.2 %.
- 74 falsos positivos (cerrar la garra sin objeto presente).
- 0 falsos negativos (no cerrar la garra cuando el objeto estaba presente).

Adicionalmente, se realizaron pruebas de ejecución en tiempo real, donde se posicionaba el gripper de manera progresiva, a modo de que este se cierre al momento que la Política

Neuronal tome esta decisión. En la Figura 6.6 es posible visualizar una secuencia progresiva del gripper cerrándose.

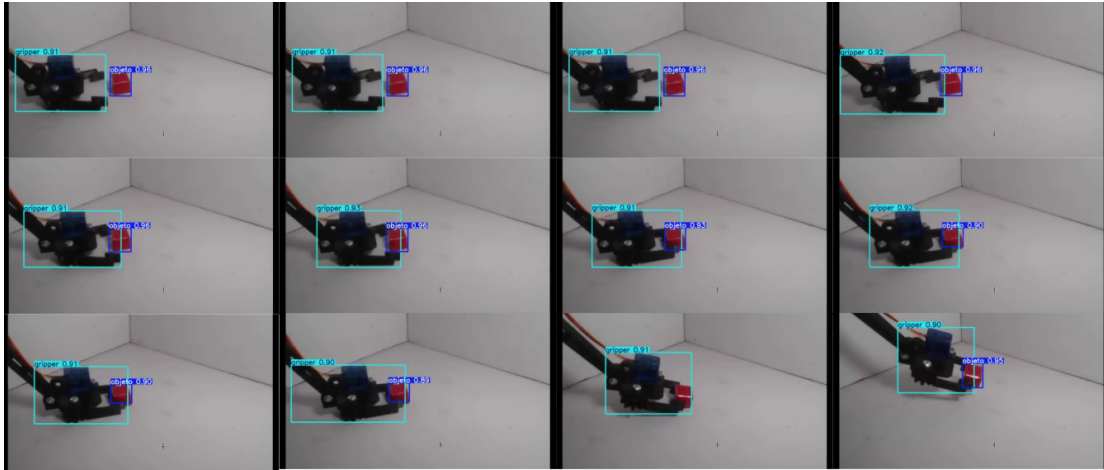


Figura 6.6: Secuencia de imágenes de la garra real al detectar y sujetar el objeto.

Asimismo, la Figura 6.7 presenta una secuencia de imágenes que ilustran el momento exacto en que la red neuronal decide abrir el gripper.

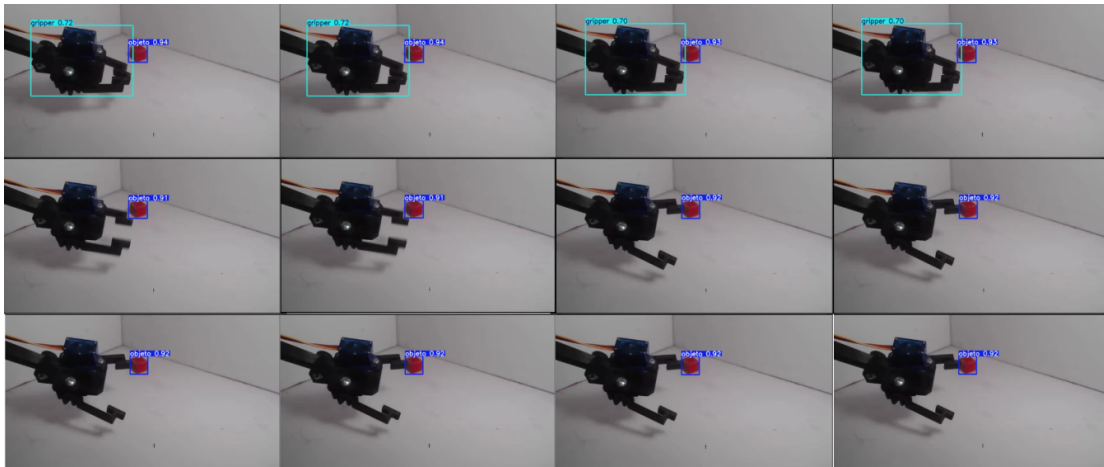


Figura 6.7: Secuencia de imágenes de la garra real al decidir la acción de abrir.

6.5. Discusión

Tras implementar y evaluar el sistema de control propuesto tanto en simulaciones como en pruebas reales, se pueden extraer aprendizajes relevantes sobre su desempeño.

En general, el algoritmo PPO entrenado en simulación logró un alto rendimiento, alcanzando convergencias estables con tasas de éxito cercanas al 90 % en el entorno controlado. Este nivel de desempeño en simulaciones concuerda con lo reportado en literatura. Sin embargo, al transferir la política entrenada al mundo real, se observó una disminución en el rendimiento, manifestada en una menor tasa de agarre exitoso y una mayor variabilidad en los resultados. Esta brecha entre simulación y realidad, conocida como reality gap, es un fenómeno ampliamente documentado en robótica.

La literatura menciona, en tareas de manipulación con simuladores y manipuladores reales se observa un reality gap que oscila entre un 10 % y un 20 % en escenarios complejos. Ellos señalan que, para prototipos iniciales, un gap inferior al 10 % suele considerarse aceptable. Dado que en nuestro caso nuestro reality gap aproximado es del 10 %, este valor se asimila a lo reportado y confirma la viabilidad del enfoque en esta etapa exploratoria. [20]

En nuestras pruebas, la reducción de desempeño se atribuye principalmente a las imperfecciones de los módulos de visión. Por ejemplo, se detectaron casos donde el agente no lograba capturar el objeto en la prueba real pese a haberlo hecho consistentemente en simulación, debido a ligeros errores en la estimación de profundidad del objeto o del gripper. A continuación, se discuten detalladamente estos hallazgos, contrastándolos con otros enfoques recientes, y se analizan las fortalezas y limitaciones de la solución junto con las posibles fuentes de error y oportunidades de mejora.

Desde una mirada crítica, el desempeño en simulación superó las expectativas iniciales, evidenciando la capacidad del algoritmo PPO de aprender una secuencia eficaz de movimientos de la prótesis para alinear y cerrar los dedos sobre el objeto objetivo. El uso de deep learning permitió capturar una política de control flexible sin necesidad de modelar manualmente el complejo espacio de estados de la mano robótica. Una de las fortalezas principales del enfoque es su adaptabilidad: a diferencia de los controladores basados en cinemática inversa o en patrones preprogramados, el agente de RL puede ajustarse a variaciones del entorno a través del aprendizaje. Esto le confiere potencial para enfrentar situaciones no vistas durante el desarrollo, como cambios moderados en la posición u orientación del objeto. Además, la integración de visión por computador (YOLO y Depth Anything) aporta un carácter autónomo al sistema, permitiendo detectar y localizar objetos de forma automática. Esta sinergia entre visión e inteligencia artificial es coincidente con tendencias recientes en robótica de servicio, donde se busca que los manipuladores perciban su entorno y tomen decisiones informadas.

En nuestro trabajo, se cumplió un rol similar otorgando conciencia espacial al agente de control, lo que constituye una ventaja técnica significativa frente a enfoques de prótesis tradicionales que no incorporan visión. Asimismo, el hecho de basarse en aprendizaje por refuerzo profundo supone una contribución innovadora en el contexto de prótesis de mano controladas automáticamente. Estudios previos en la última década suelen apoyarse en señales musculares (EMG) del usuario para el control de prótesis, lo que requiere largos períodos de entrenamiento personal del paciente.

No obstante, junto con estas fortalezas, se identifican varias limitaciones importantes en el sistema propuesto. En primer lugar, la dependencia en los módulos de visión introduce un cuello de botella en el rendimiento global. Si el detector de objetos YOLO falla en reconocer el objeto o si la estimación de profundidad es imprecisa, el controlador de RL recibirá información defectuosa, derivando en intentos de agarre mal dirigidos. Efectivamente, durante las pruebas reales observamos casos de omisiones, debido a una mala detección en la diferencia de profundidad entre el gripper y el objeto. Estas incidencias degradaron la tasa de éxito real.

6.5.1. Comparación con Métodos Alternativos

Al comparar el desempeño del algoritmo propuesto con los métodos actualmente utilizados en el control de prótesis de mano, se observa que las prótesis mioeléctricas (EMG) presentan un rendimiento altamente variable, condicionado por la habilidad del usuario y la disponibilidad de retroalimentación sensorial. Diversos estudios clínicos han documentado que, bajo control exclusivamente humano, las fallas de agarre son frecuentes, incluso en usuarios entrenados. Estas limitaciones se traducen en tasas de éxito de agarre relativamente modestas; por ejemplo, aún con entrenamiento intensivo, lograr una alta consistencia en los agarres resulta difícil, siendo común un desempeño en torno al 70 %. Esto evidencia que el éxito del control mioeléctrico tradicional depende en gran medida de la destreza individual y de la capacidad de mantener una concentración constante durante el uso de la prótesis.

En contraste, el sistema propuesto en este trabajo logra tasas de éxito comparables, pero con ventajas significativas en cuanto a escalabilidad, repetibilidad e independencia del usuario. Al automatizar el proceso de decisión y ejecución mediante visión artificial y aprendizaje por refuerzo, se elimina la necesidad de entrenamiento prolongado del usuario, así como la dependencia de su nivel de coordinación o concentración.

Este enfoque permite abordar una de las principales desventajas del control EMG: la curva de aprendizaje. En los sistemas mioeléctricos, dicha curva es generalmente prolongada y suele verse dificultada por la presencia de contracciones musculares residuales o involuntarias, las cuales pueden desencadenar movimientos no deseados. En la práctica clínica, este proceso de adaptación es tan exigente que una proporción significativa de usuarios termina abandonando el uso de la prótesis antes de lograr un control funcional eficaz.

En cambio, el sistema automatizado presentado en esta memoria permite alcanzar un rendimiento aceptable sin requerir sesiones de entrenamiento específicas para cada usuario. Asimismo, ofrece una plataforma que puede ser escalada y mejorada mediante investigación adicional, sin necesidad de repetir los ciclos de aprendizaje desde cero. Esto posiciona a los sistemas de control basados en inteligencia artificial como una alternativa prometedora frente a las limitaciones intrínsecas de las prótesis mioeléctricas tradicionales.

6.5.2. Fuentes de Error

Se identifican dos principales fuentes de error que explican la disminución en la tasa de éxito del algoritmo de control al ser implementado en el entorno real. La más relevante corresponde a las deficiencias en la estimación de profundidad entregada por el modelo Depth Anything. Si bien este algoritmo proporciona mapas de profundidad con métricas aceptables en condiciones generales, se observó que en ciertos casos no lograba diferenciar correctamente los niveles de profundidad entre el gripper y el objeto. Como resultado, al ingresar esta imagen a la red neuronal, el sistema interpretaba erróneamente que ambos elementos se encontraban a la misma distancia, ejecutando la acción de cierre pese a no haber contacto con el objeto. Esta falla se evidencia en la alta frecuencia de cierres negativos (sin éxito de agarre) en comparación con los obtenidos durante la simulación. Parte de esta discrepancia se explica por el hecho de que, en el entorno simulado, el campo de profundidad es ideal: los bordes entre el gripper, el objeto y los límites del entorno están perfectamente definidos. Esta condición, sin embargo, no puede replicarse fielmente en ningún entorno real, incluso utilizando cámaras de profundidad físicas.

Una segunda fuente de error probable está relacionada con la cantidad de iteraciones de entrenamiento empleadas durante la simulación. Aunque se alcanzó una tasa de

éxito cercana al 95 % en dicho entorno, diversos estudios en la literatura recomiendan duplicar el número de pasos de entrenamiento para garantizar una mayor robustez y generalización de la política de control. Un número limitado de iteraciones podría provocar que el agente no haya aprendido con suficiente consistencia las variaciones sutiles que aparecen al extrapolar el modelo a condiciones reales.

No obstante, la evidencia más concluyente sugiere que la causa principal del descenso en el rendimiento proviene de las limitaciones en la percepción visual de profundidad. Mejorar esta etapa del pipeline, ya sea optimizando el modelo de estimación o incorporando sensores de profundidad físicos calibrados, podría reducir significativamente la tasa de errores en el entorno real.

Capítulo 7

Conclusión

7.1. Conclusiones del Trabajo

En esta memoria se propuso y desarrolló un sistema de control para una prótesis de mano, capaz de inferir la intención de agarre de un objeto a partir de información visual y actuar de forma autónoma sobre la prótesis. A lo largo del trabajo se cumplieron todos los objetivos planteados inicialmente. Se realizó un análisis detallado del estado del arte en métodos de control protésico y técnicas de visión artificial aplicadas, lo que permitió fundamentar el enfoque seleccionado. Luego, se diseñó e implementó un algoritmo de control basado en aprendizaje por refuerzo profundo (PPO), entrenado en un entorno de simulación física para aprender a decidir el cierre de una garra robótica. Posteriormente, se validó experimentalmente el algoritmo integrándolo con un sistema de visión por computador en un ambiente controlado: mediante YOLOv4 se detectó la presencia y posición del objeto y de la garra protésica en la escena real, y con Depth Anything se estimó un mapa de profundidad aproximado para alimentar la red neuronal de control. Finalmente, se evaluó el rendimiento del sistema tanto en simulaciones como en pruebas reales, recopilando métricas cuantitativas de su desempeño.

Los principales hallazgos indican que el sistema es efectivo para la tarea propuesta. En el entorno simulado, el agente de aprendizaje por refuerzo alcanzó tasas de éxito cercanas al 95 % al decidir el cierre correcto de la garra. En las pruebas con el prototipo real, se logró un porcentaje de aciertos en torno al 85 %, evidenciando una buena generalización del modelo desde la simulación hacia un entorno físico, aunque con una

ligera disminución en el desempeño. Esta diferencia se atribuye principalmente a las limitaciones en la etapa de percepción visual: la estimación de profundidad basada en una sola cámara no alcanza la precisión de un sensor de profundidad real, introduciendo incertidumbre que derivó en algunos casos de cierre erróneo (falsos positivos). No obstante, es relevante destacar que el sistema no incurrió en falsos negativos durante las pruebas, es decir, nunca dejó de cerrar la garra cuando el objeto estuvo presente en la posición indicada. Esto implica que, ante las condiciones ensayadas, el controlador siempre reaccionó ante una oportunidad de agarre válida, cumpliendo el propósito fundamental de detectar la intención de captura del usuario.

Entre las contribuciones de este trabajo se cuenta la integración exitosa de técnicas de visión por computador con algoritmos de aprendizaje por refuerzo en el contexto de control protésico. Se demostró la viabilidad de entrenar una política de control en simulación y transferirla a un prototipo físico simple utilizando únicamente una cámara RGB estándar y modelos de procesamiento de imágenes. Además, el uso de YOLOv4 permitió dotar al sistema de una capacidad robusta de detección de objetos en tiempo real, mientras que la incorporación de mapas de profundidad estimados facilitó que la red neuronal tomara decisiones informadas sobre la presencia y posición relativa del objeto respecto a la prótesis. En conjunto, el sistema logrado es un prototipo de prueba de concepto que valida la posibilidad de implementar control semiautónomo en prótesis de mano empleando inteligencia artificial.

Como toda investigación exploratoria, este trabajo presenta algunas limitaciones. El escenario de validación fue restringido a un ambiente controlado de laboratorio, con un único tipo de objeto estático y condiciones de iluminación favorables. Si bien esto permitió aislar variables y comprobar la factibilidad básica, será necesario extender las pruebas a entornos más complejos y con mayor diversidad de objetos para consolidar la robustez del enfoque. Asimismo, la prótesis empleada se simplificó a una pinza de dos dedos; por tanto, los resultados obtenidos deben interpretarse como un primer paso hacia una aplicación en prótesis de mano más sofisticadas y con movimientos más complejos. Pese a estas limitaciones, los objetivos iniciales se cumplieron satisfactoriamente y los resultados respaldan la relevancia del método. En conclusión, se ha logrado establecer un sistema de control autónomo básico para una prótesis de mano mediante visión y aprendizaje, aportando evidencia de que este camino tecnológico puede mejorar la autonomía y eficacia de las prótesis en el futuro.

7.2. Trabajos Futuros

Como proyección de este trabajo, se identifican diversas líneas futuras de desarrollo para avanzar hacia una implementación más robusta y funcional en contexto real. En primer lugar, sería conveniente mejorar la etapa de percepción de profundidad: la integración de un sensor de profundidad dedicado como una cámara RGB-D. Asimismo, se propone ampliar el conjunto de pruebas a objetos de distintas formas y tamaños, bajo condiciones ambientales variables (iluminación, fondos complejos), con el fin de evaluar y reforzar la capacidad de generalización del sistema fuera del entorno controlado.

Bibliografía

- [1] Dario Farina and Oskar C. Aszmann. Bionic limbs: clinical reality and academic promises. *Science Translational Medicine*, 6(257):257ps12, 2014.
- [2] Kuo Zhuang, Ziqiang Yang, Yong Wu, and Gang Li. Towards natural control of intelligent prosthetic hand using surface electromyography: A review. *BioMed Research International*, 2019.
- [3] Manfred Atzori and Henning Müller. Control capabilities of myoelectric robotic prostheses by hand amputees: A scientific research and market overview. *Frontiers in Systems Neuroscience*, 9:162, 2015.
- [4] Claudio Castellini and et al. Proceedings of the first workshop on peripheral machine interfaces: Going beyond traditional surface electromyography. *Frontiers in Neurorobotics*, 8:22, 2014.
- [5] Anders Fougner, Erik J Scheme, Adrian D Chan, Kevin Englehart, and Øyvind Stavdahl. Resolving the limb position effect in myoelectric pattern recognition. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 20(4):676–685, 2012.
- [6] Hao Liu, Xiangyang Zhu, and Jingjin Yu. Deep learning for myoelectric control: A review of recent advances. *IEEE Transactions on Biomedical Engineering*, 68(6):1983–1995, 2021.
- [7] Mateusz Nowak, Łukasz Kołodziej, Paweł Gnutek, and Mariusz Bieńkowski. Computer vision-based object detection for myoelectric prosthetic hand. *Sensors*, 20(17):4743, 2020.

- [8] Max Ortiz-Catalan and et al. Clinical evaluation of advanced artificial limbs for amputees: From bench to bedside. *Nature Biomedical Engineering*, 4(7):698–710, 2020.
- [9] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
- [10] Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data. *The International Journal of Robotics Research*, 37(4–5):421–436, 2018.
- [11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [12] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.
- [13] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
- [14] Alexander Kirillov, Jasper R. R. Uijlings, Ross Girshick, and Kaiming He. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.
- [15] Guido Van Rossum. *Python para todos*. Python Software Foundation, 2011. Versión traducida y adaptada.
- [16] Erwin Coumans. Pybullet, a python module for physics simulation in robotics, games and machine learning. *GitHub Repository*, 2021.
- [17] Yufei Duan, Ke Lin, et al. Sim-to-real transfer for dexterous manipulation via learning from demonstrations. *IEEE Robotics and Automation Letters*, 7(4):11050–11057, 2022.

- [18] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Noah Ernestus, and Raphael Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Open Source Software*, 2021. <https://github.com/DLR-RM/stable-baselines3>.
- [19] Glenn Jocher. Yolo by ultralytics. <https://github.com/ultralytics/yolov5>, 2020. Accedido en 2025.
- [20] Lukas Elster, Philipp Rosenberger, Martin Holder, Ken Mori, Jan Staab, and Steven Peters. Introducing the double validation metric for radar sensor models. *Automotive and Engine Technology*, 9, 2024. Open Access.
- [21] Alexey Nekrasov, Tianyu Shen, Zhenguo Shi, Hao Tang, and Vladlen Koltun. Depth anything. <https://github.com/isl-org/Depth-Anything>, 2023. Accedido en 2025.