ARQUITECTURAS DE RED NEURO-CONVOLUCIONAL PARA APLICACIONES DE ROBÓTICA ASISTENCIAL

Robinson Jiménez Moreno
Diana Marcela Ovalle Martínez

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Doctorado en Ingeniería
UNIVERSIDAD DISTRITAL "FRANCISCO JOSÉ DE CALDAS"

Robinson Jiménez Moreno

Nació en Bogotá, Colombia, en 1978. Recibió su grado como ingeniero electrónico en la universidad Distrital Francisco José de Caldas en 2002, posteriormente su título de magíster en ingeniería de la Universidad Nacional de Colombia en 2012 y el título de Doctor en ingeniería de la universidad Distrital Francisco José de Caldas en 2018. Es actualmente docente de planta en el programa de ingeniería en mecatrónica de la universidad Militar Nueva Granada.

Diana Marcela Ovalle Martínez

Diana Marcela Ovalle Martínez nació en Bogotá, Colombia, en 1982. Recibió su título de Ingeniera Electrónica de la Universidad Distrital Francisco José de Caldas en 2005, recibió su título de magíster en Ingeniería Eléctrica de la Universidad de los Andes en 2007 y recibió su título de Doctora en Tecnologías Industriales de la Universidad Politécnica de Cartagena (España), en 2012. Actualmente, es docente de planta de la Facultad de Ingeniería de la Universidad Distrital Francisco José de Caldas.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Doctorado en Ingeniería
UNIVERSIDAD DISTRITAL "FRANCISCO JOSÉ DE CALDAS"

Arquitecturas de Red Neuro-convolucional para Aplicaciones de Robótica Asistencial

Robinson Jiménez Moreno
Diana Marcela Ovalle Martínez

Jiménez Moreno, Robinson
Arquitecturas de red neuro-convolucional para aplicaciones de robótica asistencial / Robinson Jiménez M., Diana Marcela Ovalle Martínez. -- 1a. ed. -- Bogotá : Universidad Distrital Francisco José de Caldas, 2020.
168 páginas ; 24 cm. -- (Doctorado en Ingeniería).

Incluye lista de figuras, lista de tablas y apéndices. -- Contiene bibliografía.

ISBN 978-958-787-237-8 (impreso) -- 978-958-787-238-5 (digital)

1. Robótica personal 2. Algoritmos (computadores) 3. Redes neuronales (computadores) I. Ovalle Martínez, Diana Marcela II. Título III. Serie

CDD: 629.8924019 ed. 23 CO-BoBN– a1057503

ISBN Impreso: 978-958-787-237-8
ISBN Digital: 978-958-787-238-5

Primera edición: Bogotá, octubre de 2020

Corrección de estilo y diseño gráfico:
Amadgraf Impresores Ltda.
Impresión:
Amadgraf Impresores Ltda.

Doctorado en Ingeniería
Carrera 7 # 40B-53
Bogotá
Correo electrónico: investigacion.doctoradoing@udistrital.edu.co

Hecho el depósito legal.

Impreso y hecho en Colombia
Printed and made in Colombia.

Agradecimientos

Los autores agradecen a la Universidad Distrital Francisco José de Caldas y al fondo editorial por la promoción de esta publicación.

A su vez los autores agradecen el apoyo en el aspecto investigativo al Grupo de Investigación, Desarrollo y Aplicaciones en Señales – IDEAS, de la Universidad Distrital Francisco José de Caldas, y al grupo de investigación Davinci de la Universidad Militar Nueva Granada.

Los autores extienden sus agradecimientos a la Universidad Distrital Francisco José de caldas, donde la autora Diana Ovalle es docente tiempo completo de planta, y a la Universidad Militar Nueva Granada, donde el autor Robinson Jiménez es docente tiempo completo de planta, por el apoyo en la dedicación de horas para la realización de la presente publicación.

A su vez extienden el agradecimiento al Doctorado en Ingeniería de la Universidad Distrital Francisco José de Caldas, que fue la cuna del desarrollo investigativo en el marco de la formación doctoral de que los autores fueron participes como tutora y estudiante, respectivamente.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Sistemas para Inteligencia Robótica

1.1. Introducción

El presente documento expone el desarrollo de un algoritmo orientado a esquemas de robótica asistencial trabajando en ambientes multi-herramientas. Esto se refiere a un robot de tipo antropomórfico, que se desenvuelve en un área de trabajo compartida por una persona, a quien asistirá en tareas como entrega de herramientas. Para ello, el robot debe identificar qué herramienta desea tomar dentro de un grupo de herramientas, lo que corresponde a una labor de reconocimiento de patrones. Cada herramienta presenta características particulares que deben ser aprendidas mediante un algoritmo de reconocimiento.

En este trabajo, se expone una dificultad en dicho reconocimiento aún no abordada en trabajos similares, y que surgió de los desarrollos propios de robótica asistencial realizados previamente. Al buscar reconocer una herramienta dentro de un grupo, el sistema de reconocimiento de patrones debe aprender las características que exhibe cada herramienta. Típicamente, esta tarea se realiza mediante la captura de la imagen del grupo de

herramientas por medio de una cámara, desde una posición dada, aprendiendo herramienta por herramienta.

Una vez reconocida y ubicada espacialmente la herramienta, se emplean algoritmos de planeación de trayectorias que, por medio de la cinemática del robot, permiten el desplazamiento del efector final hasta la herramienta. Pero, si se presenta el caso que en el área de trabajo un usuario interrumpa dicha trayectoria, el robot debe buscar una solución para alcanzar su objetivo. En la actualidad, dicha solución se ha orientado a detener el robot y esperar a que termine la interrupción por parte del usuario. Al buscar mejorar esta opción, por el ejemplo, a partir de que el robot sea capaz de generar la evasión del usuario, o de lo que le obstruye, se debe buscar una nueva trayectoria que este libre, partiendo desde el nuevo punto en que se encuentra luego de detenerse (robot desplazado) hasta la herramienta. Es aquí donde se presenta el problema de reconocimiento, desde la nueva posición se debe capturar la información de la herramienta para generar el nuevo desplazamiento, donde al cambiar el punto de captura, por cercanía o lejanía, la herramienta presenta más o menos características a nivel de la imagen original capturada, lo que varía el grado de reconocimiento desde el punto del aprendizaje inicial, dificultando el reconocimiento y confundiendo las herramientas presentes en la escena.

Los algoritmos de desplazamiento en función de una trayectoria libre, partiendo de la captura del objeto de agarre desde una posición siempre fija y equidistante, se han trabajado ampliamente. Sin embargo, desde esta perspectiva dinámica planteada (avanzar-detenerse y calcular, iterativamente), no presenta aún soluciones que se hacen necesarias para mejorar la interacción hombre-máquina, tal cual como lo hace un ser humano, que cambia la trayectoria a su destino cuando un obstáculo se detecta y se va aproximando a este. De manera que se realiza una revisión del estado del arte, que evidencia trabajos de reconocimiento de patrones en el campo de la robótica, e incluso de la robótica asistencial, sin encontrar soluciones al caso aquí expuesto. Motivo por el cual se considera un aporte al estado del arte y al conocimiento. Este aporte se da en la construcción de una arquitectura de reconocimiento de patrones, que considera variaciones de

la distancia del objeto a reconocer desde el punto de captura de la imagen. Dicha arquitectura se estructura desde técnicas de aprendizaje de tipo profundo (o de varias capas), que hacen parte de las técnicas más novedosas en el área de reconocimiento de patrones.

Después de presentada la arquitectura, en el documento se exponen algoritmos adicionales, que se requieren en la tarea de la robótica asistencial. Es el caso de un algoritmo de planeación de trayectorias, mediante técnicas de optimización no lineal; un algoritmo de evasión de obstáculos, basado en la arquitectura diseñada; un algoritmo de agarre, para la toma de la herramienta; y se sugiere un algoritmo de control para las variaciones de peso, que implica el tomar una herramienta u otra. Finalmente, se expone el resultado de la tarea conjunta del robot asistencial empleando estos algoritmos en un ambiente virtual.

1.2. Motivación

La inclusión de agentes robóticos en diversas actividades humanas es un componente del quehacer diario, que va cobrando mayor participación conforme se van dando avances tecnológicos. Es así, como la Organización para la Cooperación y el Desarrollo Económicos (OCDE) prevé un incremento en la fuerza laboral robótica en los próximos 10 años (2018-2028). Una de las labores en las que vamos encontrando más y más robots es la tarea de asistencia humana en diferentes ambientes y aplicaciones, lo que implicará la interacción hombre-máquina en un mismo espacio de trabajo (ver Figura **1-1**), impulsando así el concepto de robótica asistencial.

Cada aplicación robótica presenta diversos retos a cumplir. Pero, las capacidades que debe tener el robot están claras, en cuanto a la percepción e interacción que requiere con el medio en que se desenvolverá. Dentro de las aplicaciones de los robots asistenciales, estos cumplen tareas de alto riesgo, esfuerzo o repetitividad. Aunque, hoy en día su participación se ve restringida a una secuencia de movimientos pre-establecidos, o a un control vía sistemas inalámbricos, sistemas hápticos (de realimentación al usuario), o incluso por señas (procesamiento de imagen), que están condi-

cionados por las técnicas de percepción e interacción de máquina predominantes durante los últimos 15 años, algunas de las cuales son referidas más adelante.

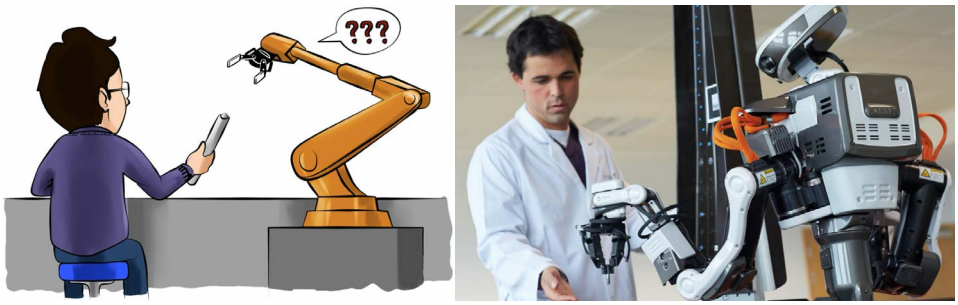


Figura 1-1: Interacción hombre-máquina[1].

La autonomía de los robots es limitada y, aunque los sistemas de visión de máquina actuales permiten aumentar dicha autonomía, la capacidad de discriminación de objetos para la interacción robótica se reduce a técnicas como conversión en espacios de color, cálculo de contornos, uso de clasificadores (tipo Haar por ejemplo), y redes neuronales de máximo una capa oculta (MPL, de sus siglas en inglés, Multi Layer Perceptron). Para este último caso, más de una capa ha presentado problemas de sobre-entrenamiento o perdida de la función de error. Estos algoritmos poseen problemas de reconocimiento, debido a restricciones como la profundidad limitada de las redes neuronales (número de capas ocultas), por las razones mencionadas previamente, y el manejo de sombras o cambios de iluminación, o de distancia, en los sistemas de visión, lo que limita mucho el desenvolvimiento de un robot en los ambientes asistidos.

Algunos trabajos previos han permitido evidenciar directamente las falencias de los algoritmos de procesamiento de imágenes [Jimenez Moreno, 2011] y reconocimiento de patrones mediante redes neuronales [Pinzón-Arenas *et al.*, 2019], en sistemas de aplicaciones de visión de máquina como el presentado en la Figura **1-2.**

1 Izquierda, tomada de: https://blog.robotiq.com/hubfs/eBooks/Part%20Presentation%20Playbook%20.pdf?hsLang=en-ca&t=1536232209651. Derecha, tomada de: https: //elpais.com/economia /2016/05/20/actualidad/1463769085_077235.html

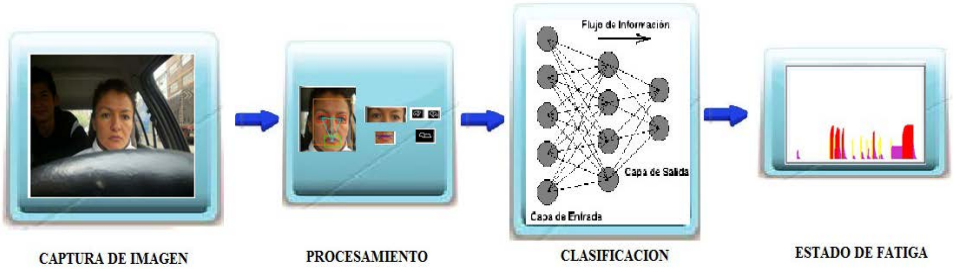


Figura 1-2: Sistema de vision de máquina para detección de cansancio en conductores [Jimenez Moreno, 2011].

Frente a estas falencias, recientes métodos de aprendizaje como el Deep Learning (DL), ofrecen nuevas posibilidades de entrenamiento de sistemas neuronales y de procesamiento de imágenes, que pueden ser aprovechados para la interacción con agentes robóticos (ver Figura **1-3**). Dichos métodos de aprendizaje han mostrado su eficiencia en el reconocimiento de patrones. Por ejemplo, a nivel de reconocimiento de comandos de voz [Pinzón-Arenas y Jiménez-Moreno, 2020], e identificación de objetos en imágenes [Velandia *et al*., 2019]. Dentro de estos métodos de aprendizaje se encuentran las redes neuronales convolucionales (CNN, de sus siglas en inglés, Convolutional Neural Networks), que son la principal técnica de Deep Learning orientada al reconocimiento de objetos en imágenes [Jimenez Moreno, 2011], con grandes ventajas sobre las técnicas convencionales de procesamiento de imágenes y problemas descritos en el párrafo anterior.

Sin embargo, las aplicaciones que emplean redes neuronales convolucionales siguen en desarrollo y dan lugar a mejoras en los algoritmos existentes. En el caso de la robótica asistencial, estas aplicaciones son recientes y las falencias de las redes neuronales convolucionales ya comienzan a abordarse. Una de estas falencias es la capacidad de las redes neuronales convolucionales en la identificación de objetos en ambientes dinámicos, donde la cámara se acerque o aleje del objeto. Para agentes robóticos asistenciales, los cuales pueden llegar a variar la perspectiva con la que ven un grupo de objetos en el espacio, debido a su naturaleza móvil, esta falencia genera problemas de confusión de clases, la cual no ha sido abordada en el estado del arte.

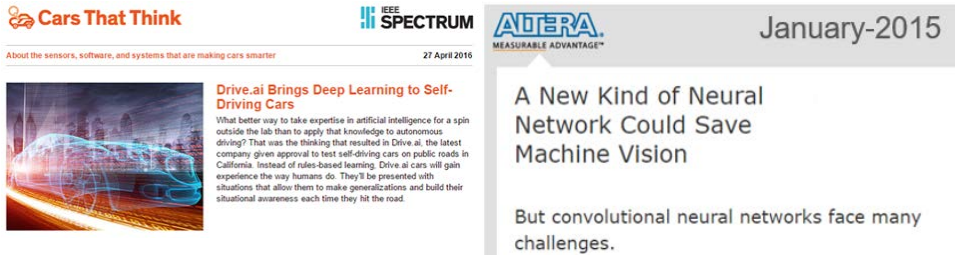


Figura 1-3: Técnicas recientes aplicables a sistemas de visión de máquina.

La situación mencionada previamente se presenta de forma característica en aplicaciones de interacción hombre-máquina, donde el robot y el usuario comparten la misma zona de desplazamiento. Por ejemplo, para un robot que alimenta a una persona, o le alcanza instrumentos de trabajo, si la persona obstruye el desplazamiento del robot, este debe cambiar de trayectoria, de forma que al buscar de nuevo el punto de destino, demarcado por el objeto o herramienta de interés, la perspectiva desde este desplazamiento cambia, haciendo que aumenten o disminuyan las características de los objetos en su nuevo campo de visión, conllevando problemas como la confusión de clases. En el presente documento se expone un aporte al conocimiento, mediante la aplicación de sistemas de reconocimiento visual por redes neuronales convolucionales para agentes robóticos, en ambientes de trabajo compartido hombre-máquina, ya sea de forma colaborativa o asistencial, solventando las variaciones de perspectiva de un grupo de objetos, que puedan generar confusión entre estos por parte del robot.

1.3. Objetivos

1.3.1. Objetivo General

Estructurar una técnica de reconocimiento de herramientas en tres dimensiones, mediante el desarrollo de un algoritmo basado en Deep Learning para emplear un brazo robótico como asistente personal en escenarios multi-herramienta.

1.3.2. Objetivos Específicos

- Analizar las técnicas existentes de Deep Learning aplicables a la detección de objetos, validando su funcionalidad, y determinar la mejor arquitectura para discriminación de herramientas en imágenes.

- Desarrollar un algoritmo de entrenamiento basado en la mejor arquitectura de Deep Learning, que permita distinguir una herramienta particular de un grupo de herramientas, sometiéndola a variaciones de distancia.

- Desarrollar un algoritmo de posicionamiento que permita a un brazo robótico ubicar su efector final sobre una herramienta identificada, evitando colisionar con una persona que ingrese a su espacio de trabajo, o con quien interactúe.

- Implementar un ambiente de pruebas virtual o real, que permita validar la técnica desarrollada mediante la localización de una herramienta por parte de un agente robótico asistencial y entregarla a un usuario, empleando Deep Learning.

1.4. Marco Metodológico

Con el propósito de dar cumplimiento a los objetivos propuestos, el marco metodológico se establece, inicialmente, validando las prestaciones de las redes neuronales convolucionales en el aprendizaje de herramientas en un escenario particular, buscando una arquitectura de red, que permita la identificación de objetos deseados, mediante heurísticas soportadas en la literatura. Inicialmente, se entrena una red neuronal convolucional con una base de datos de un grupo de herramientas a una distancia fija y, posteriormente, se evalúa con imágenes de dichas herramientas a diferentes distancias del foco de la cámara, esto permite evidenciar la necesidad de una arquitectura diferente a la estructura neuro-convolucional convencional.

Para ofrecer una solución a esta necesidad, se establecieron dos esquemas en una arquitectura multi-convolucional paralela, validando dos métodos de aprendizaje diferentes, para definir la mejor arquitectura de red a establecer, que incluye un canal adicional de información de la distancia de la cámara al objeto. Una vez hecho esto, se determina una capa final, que permite la ponderación de las ramas, mediante dos propuestas: una aritmética y otra mediante un sistema de inferencia difusa. De forma que, el resultado obtenido es una arquitectura neuro-convolucional, que en función a una entrada RGB-D (imagen a color más la información de profundidad), identifica las herramientas de un escenario particular, que permite variaciones de ubicación espacial del sensor de captura de imagen.

Una vez lograda la identificación, se aplica un algoritmo de optimización de trayectoria a un brazo robótico de tipo académico, para el agarre de la herramienta objetivo en dicho escenario. Los puntos de desplazamiento del algoritmo realimentan la entrada de la red, en las diferentes perspectivas que el brazo va tomando de la herramienta. Se emplea la arquitectura diseñada para establecer un algoritmo de evasión de obstáculos dinámicos, como puede ser un usuario que ingresa en el área de trabajo del robot.

Por último, los algoritmos resultantes se prueban en ambientes simulados y reales, para evidenciar la utilidad de los mismos. De forma general, la metodología empleada es tipo experimental.

1.5. Línea de investigación

El desarrollo de esta investigación está enmarcado dentro del Grupo de Investigación, Desarrollo y Aplicaciones en Señales - IDEAS, de la Universidad Distrital Francisco José de Caldas, en las líneas de investigación: Procesamiento de Imágenes y de video, y Control y Automatización. Así mismo, dentro del grupo de investigación Davinci, de la Universidad Militar Nueva Granada, en la línea de investigación de robótica híbrida.

1.6. Contribución del trabajo

Un robot asistencial busca desplazarse con un fin determinado. Por ejemplo, en el caso de una plataforma multi-herramientas, para tomar una herramienta entre un grupo de estas, y, debido a que parte de la labor que debe solventar el robot, al interactuar en un ambiente compartido, es el desplazamiento hacia su objetivo, en el que puede encontrar posibles obstrucciones en su trayectoria, o tener que manejar ciertos límites de seguridad, para evitar colisiones. Al establecer una trayectoria, para alcanzar el punto en el espacio donde se encuentra la herramienta, dado que se busca tener un entorno compartido hombre-máquina, el robot debe validar permanentemente que no se presenten obstáculos en dicha trayectoria. De ser así, debe replantear su desplazamiento desde la posición en la que detecta el obstáculo. El determinar dicho cambio implica moverse lateralmente fuera del alcance del obstáculo, donde debe reevaluar la posición de la herramienta desde su nueva perspectiva. Al haberse desplazado la distancia del nuevo punto de percepción las herramientas exhiben nuevas, o pierden, características respecto a la percepción inicial, desde el punto de origen. Lo que deja al entrenamiento inicial de la arquitectura de red, encargada de reconocer la herramienta, susceptible a variaciones y, por consiguiente, a errores en la clasificación.

El aporte al conocimiento logrado consiste en el desarrollo de una arquitectura de red basada en Deep Learning, para entrenamiento de robots asistentes en reconocimiento de objetos orientado a plataformas multi-herramienta. En donde se robustezca, en función de la distancia, el reconocimiento de dichos objetos, sin importar las variaciones que pueda sufrir el punto de captura de la imagen, respecto al objeto. El algoritmo base será tal que determinará la arquitectura de Deep Learning, que ofrece una solución a este problema y permite el desarrollo de una aplicación de robótica asistencial, con un grado de inteligencia mayor al actualmente encontrado en el estado del arte.

Para el problema que se desea abordar, debe ser claro que se requiere una cámara alojada en el efector del brazo, que oriente la relación espacial entre el punto del actuador y su destino final, como lo es la localización espacial de la herramienta. Una cámara fija supervisora esta susceptible a obstrucciones del brazo y del usuario, lo cual al ocluir la herramienta impide su relocalización en función al desplazamiento del efector. Una analogía clara del problema puntual a abordar se da con respecto a la visión humana que, dependiendo de la distancia, logra reconocer ciertos objetos, según se evidencia en la Figura **1-4**. Un sistema de reconocimiento de patrones en imágenes sufre el mismo problema, si la cámara de adquisición de la imagen se acerca, o aleja, del objeto podrá reconocer ciertas características y otras no. Por lo que se busca diseñar una arquitectura de redes neuronales convolucionales que genere el mismo valor de confidencia en el reconocimiento de un objeto, sin importar si se da dicho cambio.

Figura 1-4: Discriminación de caracteres con cambio de escala

1.7. Algoritmo empleado para el desarrollo propuesto

El procedimiento desarrollado para abordar el problema previamente establecido, se muestra en el Algoritmo 1.

Algoritmo 1: Desarrollo del trabajo

Begin

Paso 1: Establecer las características de las redes convolucionales para detección de objetos en ambientes dinámicos.

Paso 2: Diseñar una arquitectura de red robusta a cambios dinámicos del punto focal mediante una estructura paralela.

Contribución

Begin

Paso 1: Diseñar una arquitectura base de red convolucional para reconocimiento según profundidad, mediante:

a. Entrenamiento por transferencia de aprendizaje del conjunto paralelo.

b. Diseñar cada arquitectura paralela por separado.

Paso 2: Diseñar una capa de salida para ponderación de cada respuesta paralela en función de la profundidad, mediante:

a. Establecer una ecuación genérica de tipo algebraico que pondere la salida y emplee un elemento de saturación final.

b. Diseñar un sistema de inferencia difusa para ponderación de las capas establecidas.

Paso 3: Establecer la mejor arquitectura para reconocimiento en ambientes dinámicos.

end

Paso 3: Establecer un conjunto de algoritmos necesarios para entrenar un robot asistencial en ambientes dinámicos para agarre y entrega de herramientas, entre los que se tiene:

a. Planeación de trayectorias.

b. Evasión de posibles colisiones.

c. Agarre de herramientas.

d. Controlador para el efector de agarre.

Paso 4: Validar el desarrollo mediante un ambiente virtual de simulación.

end

1.8. Organización del documento

El capítulo 2 expone los conceptos generales y el estado del arte sobre agentes robóticos y el empleo de algoritmos de aprendizaje de máquina, que permiten dotar de cierto grado de inteligencia a dichos robots. De forma que, se busca exponer un escenario en el que: i) la robótica co-

bra un papel cada vez más relevante en la vida cotidiana, y ii) los algoritmos de aprendizaje de máquina aún están en desarrollo y son de interés investigativo. Esta revisión de los trabajos de investigación, en robótica asistencial y algoritmos de inteligencia de máquina, pretende validar el aporte aquí presentado, donde lo primero que destaca es la reciente incorporación de las técnicas de aprendizaje de máquina basadas en Deep Learning en el campo de la robótica y cómo el escenario multi-herramienta planteado no ha sido explorado, por lo cual los problemas y soluciones asociadas, están en incubación.

El capítulo 3 expone de forma puntual los algoritmos de Deep Learning, que actualmente están siendo trabajados, sus ventajas, frente a técnicas convencionales de aprendizaje de máquina, y sus desventajas, al operar en ambientes dinámicos, como lo es el requerido en la interacción hombre-máquina. Aquí, se evidencia claramente el problema de identificación de objetos cuando la distancia de captura de la imagen varía.

El capítulo 4 presenta el aporte principal del presente trabajo, donde se desarrolla una solución frente a la desventaja mencionada, mediante el diseño de una red multi-paralela, con una capa de ponderación final, que se valida mediante dos métodos: un sistema de inferencia difusa y el desarrollo de una ecuación que realiza de forma generalizada la misma tarea. Se expone un caso representativo que permite evidenciar el desempeño de la red y establecer así comparaciones frente a los resultados de ambas metodologías empleadas.

El capítulo 5 presenta algunos algoritmos necesarios para la implementación de un ambiente robótico asistencial en plataformas multi-herramientas. Primero, se presenta un algoritmo de optimización para planeación de trayectorias del brazo robótico. Posteriormente, se expone una aplicación adicional de la red propuesta como aporte al conocimiento, aplicada a la evasión de obstáculos en el desplazamiento robótico asistente. Finalmente, se expone un algoritmo de agarre, orientado a la captura de herramientas por medio de un efector tipo pinza.

El Capítulo 6 presenta el ambiente virtual de pruebas, contemplando todas las etapas descritas a lo largo del documento: i) detección de la herramienta, ii) generación de la trayectoria del punto inicial al punto donde se detectó la herramienta, iii) descripción de la trayectoria con evasión de colisiones, iv) selección del punto de agarre de la herramienta y proceso de entrega de la misma.

Finalmente, el Capítulo 7 presenta las conclusiones obtenidas en el desarrollo del trabajo de investigación en cada etapa y los trabajos futuros derivados que complementarían el aquí expuesto.

Capítulo 2

Desarrollo de Sistemas para Robótica Autónoma

Este capítulo permite contextualizar los desarrollos actuales de los algoritmos de inteligencia artificial y su empleo en sistemas robóticos, mediante un análisis del estado del arte. También se aborda cómo los recientes desarrollos de técnicas de Deep Learning, específicamente los basados en redes neuronales convolucionales, están generando avances en la frontera del conocimiento, que pueden ser orientados hacia nuevas aplicaciones para sistemas robóticos.

2.1. Generalidades

La capacidad para que un robot pueda realizar de forma autónoma una tarea, con algún grado de toma de decisiones, se basa en la integración de técnicas tanto de análisis cinemático como de inteligencia artificial, en los pasos iniciales e intermedios durante la ejecución de la tarea a llevar a cabo. Actualmente, en la industria, la mayor parte de los agentes robóticos cumplen papeles repetitivos. Por ejemplo, en líneas de ensamble (ver Figura **2-1**), donde existe una pre-programación de la tarea y el robot no toma ningún tipo de decisión, lo que los incapacita para acciones como la

interacción hombre-máquina. A la par, se suelen utilizar dispositivos de seguridad para protección, si una persona se aproxima al robot, deteniendo la operación que este ejecuta. La Figura **2-2** ilustra una celda infrarroja de seguridad, que si es atravesada detiene la operación del robot.

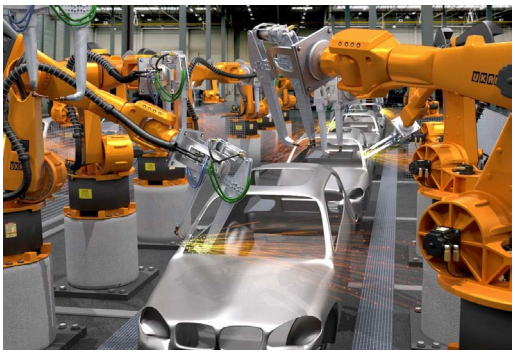


Figura 2-1: Línea de ensamble robótica[1].

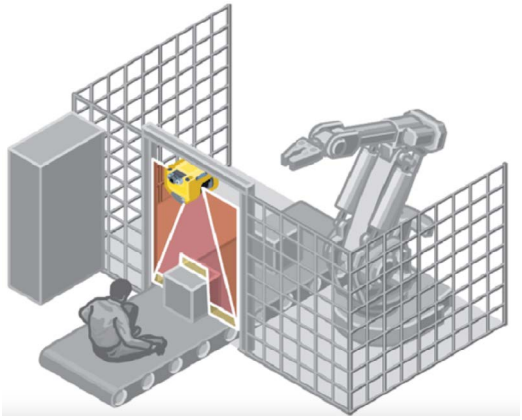


Figura 2-2: Celda infrarroja de seguridad[2].

El soportar las actividades robóticas por medio de algoritmos de inteligencia artificial se fundamenta en las técnicas de Machine Learning (ML). Estas técnicas permiten dar mayor autonomía a agentes robóticos, de cómputo y similares. Entre las múltiples aplicaciones que brindan se en-

1 Tomado de: https://www.motorpasion.com/industria/general-motors-esta-conectando-los-robots-de-sus-fabricas-a-internet-y-ya-comienza-a-cosechar-beneficios

2 Tomado de: https://www.interempresas.net/Robotica-industrial/Articulos/28378-La-proteccion-segura-de-celdas-robotizadas.html

cuentran: el reconocimiento de comandos naturales como el habla [Pinzón-Arenas *et al*., 2019], el reconocimiento de escritura [Pinzón-Arenas y Jiménez-Moreno, 2020] o de señas [Velandia *et al*., 2019], la interacción hombre-máquina [Pinzón *et al*., 2017] y la toma de decisiones [Pachon-Suescun *et al*., 2020].

Dentro de las más recientes técnicas de Machine Learning actualmente se destacan las referentes al Deep Learning [Perconti y Plebe, 2020, Koumakis, 2020, Zhang *et al*., 2020, Azarang y Kehtarnavaz, 2020, Salehi *et al*., 2020]. La Figura **2-3** ilustra la relación entre la inteligencia artificial, el Machine Learning y el Deep Learning. El Deep Learning ofrece algoritmos en la línea del Machine Learning para implementar sistemas de inteligencia artificial, que actualmente están siendo desarrollados y aplicados a diferentes áreas de la ingeniería y las ciencias básicas, con menos de una década de aplicación.

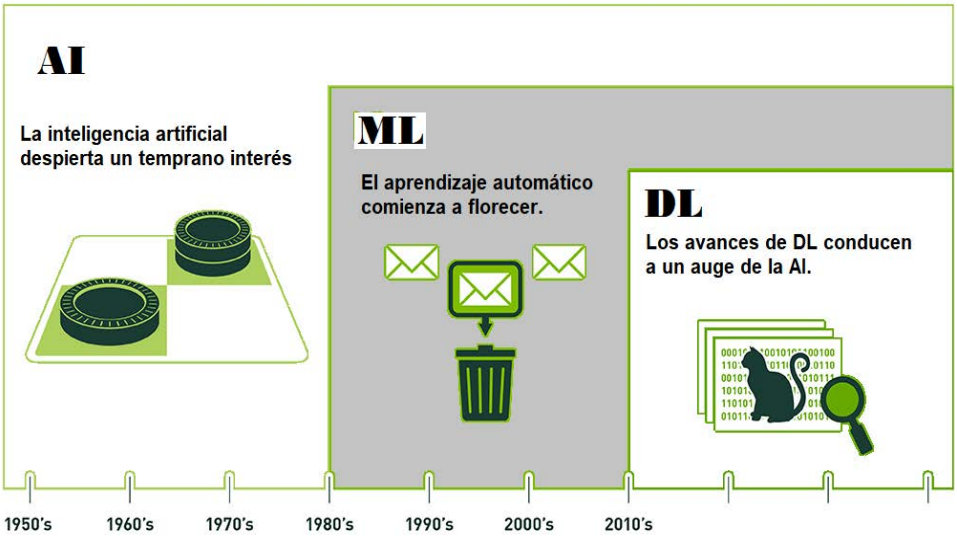


Figura 2-3: Relación entre inteligencia artificial, Machine Learnign (ML) y Deep Learning (DL).

El Deep Learning está orientado al entrenamiento multicapa, basado en los esquemas de aprendizaje del cerebro humano, permitiendo abarcar grandes cantidades de datos, de los cuales se extraen los patrones de interés. Por ejemplo, los que derivan de imágenes y el reconocimiento propio

de objetos dentro de esta. En la Figura **2-4** se puede observar la evolución de las técnicas de reconocimiento de patrones por la vía del entrenamiento neuronal, se observa que hacia 1979 se iniciaron los desarrollos con redes convolucionales, pero solo hasta 1999 se desarrollaron algoritmos de entrenamiento profundo basados en Máquinas Restrictivas de Bolztman (RBM, de sus siglas en inglés, Restrictive Boltzman Machine), que en 2006 derivaron en las primeras técnicas de Deep Learning con las redes de creencia profunda (DBN, de sus siglas en inglés, Deep Belief Networks).

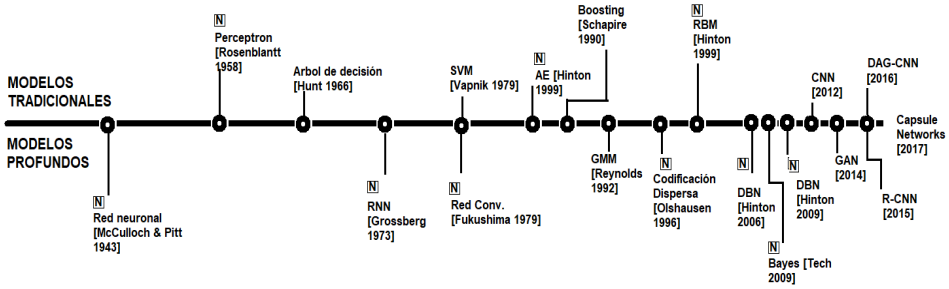


Figura 2-4: Línea de tiempo de Deep Learning.

La base de un sistema de inteligencia robótica, que permita la interacción hombre-máquina, debe integrar las técnicas propias del reconocimiento de patrones, que le faculten para la toma de decisiones y respectivas acciones. Los desarrollos en Deep Learning ofrecen soluciones en estos aspectos. Por ejemplo, para el caso de la presente investigación, en la búsqueda de la interacción de un robot como asistente, es requerida la capacidad de percepción del medio por parte de este, lo cual se logra con una cámara como elemento de captación del medio. En función de la información que se obtiene con la cámara, el robot debe discriminar qué hay en el medio y qué hacer con lo encontrado.

2.2. Estado del arte en sistemas de inteligencia robótica

La interacción hombre-máquina con agentes robóticos se ha presentado como un área importante de investigación y desarrollo en las dos últimas décadas [Daugherty y Wilson, 2018]. En este campo, se pueden encon-

trar agentes robóticos desempeñando diferentes tareas de la mano con algoritmos de Machine Learning. Por ejemplo, en medicina operan como asistentes para el tratamiento de patologías [Kiguchi y Hayashi, 2013]; en sistemas de automatización, operan en tareas de tipo industrial [Buchner *et al*., 2012]; en sistemas biológicos, permiten emular diferentes ambientes mediante redes neuronales [Kopman *et al*., 2015]. En [Jiménez-Moreno *et al*., 2012] se entrena un sistema de visión de máquina mediante redes neuronales a fin de detectar síntomas de somnolencia o distracción en un conductor.

Otra de las técnicas de Machine Learning son los sistemas difusos. Por ejemplo, en [Farooq *et al*., 2012, Guechi *et al*., 2012, Ansari *et al*., 2012] estos sistemas se utilizan para determinar la trayectoria de navegación de agentes robóticos. En [Moreno y Lopez, 2013] se presenta un sistema híbrido de Machine Learning, un algoritmo de clustering difuso que permite establecer la trayectoria de un móvil bajo un sistema de visión de máquina en 2 dimensiones. En [Moreno *et al*., 2013] se presentan varios casos de robots asistenciales que emplean sistemas difusos para su operación.

Los principales desarrollos alcanzados en Deep Learning cubren algunos casos puntuales como modelamiento de datos de series temporales [Längkvist *et al*., 2014]. Una aplicación bastante trabajada se encuentra en los sistemas de reconocimiento de habla [Cui *et al*., 2015], para la cual se vienen presentando diversos métodos complementarios a las técnicas de Deep Learning, como la inclusión del método de gradiente descendiente estocástico promedio [You *et al*., 2014], reduciendo el tiempo de entrenamiento de la red. Otra mejora de Deep Learning bajo esta aplicación se puede encontrar en [Ochiai *et al*., 2014], en donde un sistema de reconocimiento de palabras basado en modelos ocultos de markov (HMM, de sus siglas en inglés, Hiden Markov Models) es integrado con una red neuronal profunda (DNN, de sus siglas en inglés, Deep Neural Network), conformando un nuevo modelo denominado DNN-HMM, que mejora el reconocimiento de la técnica inicial con una reducción de error del 8.4%.

Los sistemas de reconocimiento de caracteres son otra aplicación importante de las técnicas de Deep Learning, que han originado modelos híbridos. Por ejemplo, en [Ji *et al*., 2014], los autores presentan una variación de las Deep Belief Networks (DBN), a fin de lograr reducir la redundancia en el entrenamiento. Esta nueva red denominada SR-DBN (SR, de sus siglas en inglés, Sparse Response), es capaz de extraer múltiples características a múltiples niveles de abstracción, validándola en caracteres numéricos a mano alzada, logrando mejorar significativamente el desempeño de técnicas, como el análisis de componentes principales (PCA, de sus siglas en inglés, Principal Component Analysis).

En relación al tratamiento de imágenes, se han comenzado a presentar desarrollos de métodos de identificación de objetos y extracción de las características presentes en la imagen . Por ejemplo, en [Chen *et al*., 2014], los autores proponen una variación al entrenamiento de una red neuronal convolucional profunda, otro tipo base de Deep Learning, con el objetivo de extraer de imágenes satelitales complejas, patrones deseados como lo es la detección vehicular, desarrollan una variante híbrida dividiendo los datos en varios bloques de diferentes escalas y empleando características Haar para ello, presentando así un mejor desempeño que las redes neuronales profundas convencionales y solucionando el problema de escala, visto desde el satélite, a la que se pueda encontrar el vehículo en entornos como una ciudad.

Otras aplicaciones se centran en la detección facial. Por ejemplo, en [Zhang y Zhang, 2014], los autores utilizan Deep Learning para identificar rostros con cambios de pose, de expresión y de iluminación, tomando una base de 117 mil rostros con dichos cambios a fin de reconocer 5 poses que, con variaciones de ángulo, determinan 15 subcategorias en las que encontrar un posible rostro, superando así los problemas de clasificadores convencionales como los presentados en el algoritmo de Viola-Jones [Viola y Jones, 2001]. Otras aplicaciones de detección de rostros mediante Deep Learning permiten identificar estados de adormecimiento en un conductor [Dwivedi *et al*., 2014], reconocimiento de expresiones [Song *et al*., 2014], y reconocimientos de características de belleza [Gan *et al*., 2014].

En [Hou *et al*., 2015] se presenta un caso particular de entrenamiento bajo Deep Learning. En este caso, se busca evaluar la calidad de una imagen, dada la posibilidad de que contenga ruido, variaciones de resolución, o pre-procesamiento, clasificándola en 5 grados o niveles de calidad y buscando emular la forma en que lo haría un humano. Otras aplicaciones de interés en procesamiento de imágenes se pueden encontrar en [Wang y Morel, 2014, Rioux-Maldague y Giguère, 2014, Wu *et al*., 2014].

En el campo de la automatización, son pocos los trabajos desarrollados mediante técnicas de Deep Learning. En [Tamilselvan y Wang, 2013], los autores proponen un método de validación de las condiciones de operación de sistemas eléctricos complejos, como son los motores de avión o transformadores eléctricos. Mediante entrenamiento multivariable empleando Deep Learning, se desarrolla un algoritmo orientado a reducir costos de mantenimiento y prevención de fallas. En [Shang *et al*., 2014] se presenta un entrenamiento mediante Deep Learning para un sensor industrial capaz de validar múltiples entradas (35 para este caso), que permite estimar el punto de destilación por unidad de diesel pesado, mostrando las ventajas que presenta sobre redes neuronales simples y máquinas de soporte vectorial.

Los desarrollos orientados en Deep Learning han generado complementos a los métodos convencionales de esta técnica, como lo son las redes convolucionales, auto-encoders y las redes de creencia profunda, mejoras que se han incrementado en los últimos años [Schmidhuber, 2015]. Algunas de estas mejoras están orientadas a aumentar la discriminación de las características en imágenes, como se presentan en [Zhang *et al*., 2015a, Dong *et al*., 2016, Guo *et al*., 2016]. Las variaciones presentadas involucran modelos híbridos de Deep Learning con técnicas convencionales como máquinas de soporte vectorial [Kim *et al*., 2015a, Kim *et al*., 2015b] y aprendizaje multi-escala [Bai *et al*., 2016], donde el desarrollo de nuevas técnicas de Deep Learning son emergentes de diversos trabajos de investigación [Liu *et al*., 2015, Zhang *et al*., 2015b].

En la Figura **2-5**, se puede observar una relación de las aplicaciones que recientemente (2016-2017) se han generado con el uso de las redes neuronales convolucionales en diversos campos, donde se evidencia que el mayor aporte se centra en el reconocimiento de patrones en imágenes.

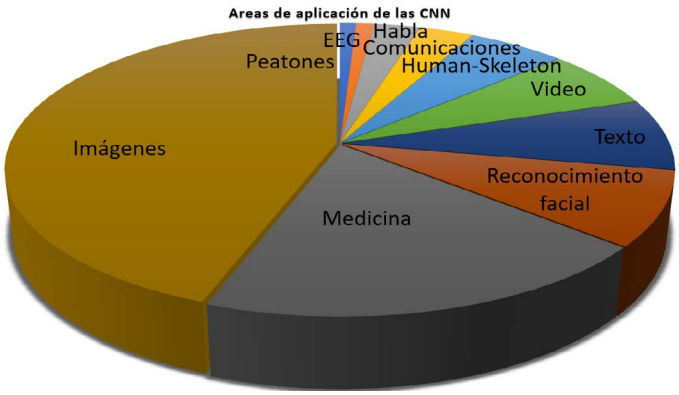


Figura 2-5: Áreas de trabajo en redes neuronales convolucionales 2016-2017.

En el campo de la robótica, igual que en el de la automatización, también son pocos los trabajos desarrollados mediante técnicas de Deep Learning, donde es de destacar que este es un campo de aplicación en el que se requiere trabajo investigativo y desarrollo de técnicas para entrenamiento robótico. Dentro de los trabajos más destacados se encuentran los mencionados a continuación, siendo evidente que las técnicas de Deep Learning permiten implementar sistemas de inteligencia computacional en robots [Neukart y Moraru, 2014].

Buchner, en [Buchner *et al.*, 2012], propone un método de interacción humano-robot en el que, mediante análisis de procesamiento de imágenes, reconoce señales de la mano para identificar las tareas a realizar por parte del robot. De las características extraídas, se emplea un sistema de Machine Learning empleando una técnica de Deep Learning, que deriva características espacio-temporales en una jerarquía en forma de árbol respecto a la imagen de entrada, combinado con técnicas conocidas de clasificación como lo son la máquinas de soporte vectorial y la técnica de los *K*-vecinos más cercanos.

Guo, en [Guo *et al.*, 2017], expone una métodologia de agarre de objetos mediante redes neuronales convolucionales basada en regiones de interés (RoI, de sus siglas en inglés, Regions of Interest). Este tipo de red tiende a incrementar el tiempo de ejecución, dependiendo de la cantidad de contornos que haya en la imagen y el tamaño de ésta, debido al algoritmo de la RoI, por lo que usan una variación de esta red, denominada Fast R-CNN que emplea cinco capas convolucionales, no siendo claro si se emplea información de profundidad en la imagen para el agarre del objeto.

Wang, en [Wang *et al.,* 2016], presenta el desarrollo de un algoritmo de agarre de objetos mediante un manipulador robótico, realizando reconocimiento del agarre mediante una red neuronal convolucional de cinco capas de profundidad, basada en el aprendizaje de aspectos como color, profundidad y superficie, de objetos etiquetados en bases de datos conocidas, no evidenciando qué ocurre con variaciones de distancia de la cámara o del objeto.

Lenz, en [Lenz *et al.*, 2015], presenta un desarrollo para agarre de objetos, orientado a robots manipuladores y basado en Deep Learning e información de profundidad, determinando superficies rectangulares como posibles candidatas a un agarre, empleando métodos de penalización que deriven en la optimización de este. Se emplean redes de dos capas de profundidad, entrenadas según tamaños, posiciones y orientación de agarres posibles. La información de profundidad permite conocer los aspectos base de contorno, para la generación de la superficie de la que resulta el agarre, no evidenciando qué ocurre con variaciones de distancia de la cámara o del objeto. Por esta misma vía, se encuentran otros desarrollos en Deep Learning orientados a manipulación robótica basada en la etapa de agarre. Dentro los más recientes, se encuentra el presentado en [Kalashnikov *et al*., 2018], donde llegan a evaluar hasta 580 mil intentos de agarre.

Hossain, en [Hossain *et al.*, 2017], propone un algoritmo de Deep Learning, una red de creencia profunda optimizada mediante algoritmos genéticos, para reconocimiento de objetos, con la que se estima la pose del

objeto y se encuentra el agarre adecuado. No se contempla información de profundidad y la vista desde la que se procesa la imagen corresponde a una distancia fija. Por la misma vía, en el área de la interacción humano-robot son pocos los desarrollos encontrados. En [Gutiérrez *et al*., 2017] se expone la aplicación de un robot social para interacción en un ambiente doméstico, donde se emplea información de profundidad para detectar y discriminar objetos, de forma tal que utilizando segmentación semántica se le informa a un usuario la posición de los objetos.

Chen, en [Chen *et al*., 2018], presenta el desarrollo de un algoritmo de reconocimiento de emociones, orientado a facultar a un robot social en el reconocimiento de dos estados de ánimo de la persona con que interactúe. Los estados considerados corresponden a felicidad o enojo. Para ello, emplean imágenes estáticas de rostros y basan los algoritmos en Deep Sparse Autoencoder Network (DSAN), una técnica de Deep Learning que permite aprender áreas como las cejas, los ojos y la boca, y facultan a un robot para discriminar hasta en un 89% alguno de estos dos estados.

Dairi, en [Dairi *et al*., 2018], expone un desarrollo utilizando sistemas de inferencia difusa e información de profundidad, orientando su trabajo a un sistema de evasión de obstáculos en sistemas de conducción vehicular autónoma. Los autores, en función a un mapa de disparidad (diferencia de dos imágenes), encuentran la información de profundidad de la escena detectando un posible obstáculo, para lo que emplean una máquina de Boltzman profunda como algoritmo de Deep Learning. La salida de esta es sometida a un sistema difuso para generar las alarmas de evasión. De forma tal que, se logra establecer una arquitectura totalmente diferente de Deep Learning, a la desarrollada en este trabajo, pero con los mismos componentes base de algoritmia. Lo cual ayuda a validar el camino optado como medio de solución de las falencias en aplicaciones de Deep Learning.

En conclusión, los trabajos previamente expuestos no contemplan las consideraciones propias del problema planteado, en el desarrollo y aporte del presente trabajo de investigación. Sin embargo, permiten evidenciar un

fuerte interés en el desarrollo de algoritmos de inteligencia artificial orientados a robótica, como se expone en [Jiménez-Moreno *et al*., 2012]-[Moreno *et al*., 2013] y [Neukart y Moraru, 2014], empleando técnicas como redes neuronales convencionales o sistemas difusos. En [Hou *et al*., 2015]-[Zhang *et al*., 2015b], se evidencia un creciente desarrollo de aplicaciones basadas en Deep Learning, donde técnicas como las redes neuronales convolucionales, que están actualmente en la frontera del conocimiento, están siendo implementadas y mejoradas. Es así como en [Guo *et al*., 2017]-[Hossain *et al*., 2017] se evidencia cómo en el campo de la robótica y, muy de cerca, de la robótica asistencial se comienzan a emplear estas técnicas para la solución de los problemas de interacción humano-máquina.

En [Lu *et al*., 2017], se encuentra una aproximación al escenario aquí planteado, en el cual se emplean imágenes de entrada con objetos que presentan variación de tamaño, según la distancia a la que se encuentren del foco de la cámara utilizada, que también esta orientado a aplicaciones robóticas. Los autores ilustran el caso hacia vehículos autónomos, cuya cámara de sensado del medio captura la imagen de otros vehículos, que varían su distancia respecto a si se acercan o alejan, lo cual también aplican a la identificación de peatones. La diferencia en las soluciones propuestas es que los autores realizan un modificación a las capas de convolución para que operen bajo regiones de histograma de los objetos encontrados y así discriminar su clasificación, pero desde un mismo enfoque visual dinámico, haciendo así divergir las metodologías y soluciones desarrolladas.

Capítulo 3

Identificación de Objetos Mediante Aprendizaje Profundo

El presente capítulo permite derivar las conclusiones en relación a la mejor arquitectura de red para discriminación de herramientas mediante Deep Learning, estableciendo claramente el problema planteado de variación de distancia mediante detección por redes neuronales convolucionales. Para ello se introduce el marco teórico referente a esta técnica, se compara su desempeño frente a resultados obtenidos mediante técnicas clásicas de redes neuronales y, finalmente, se establece una arquitectura de reconocimiento en plataformas multi-herramientas.

3.1. Redes Neuronales de aprendizaje profundo

El Machine Learning, desarrollado por medio de técnicas de inteligencia artificial, implica diferentes niveles de abstracción. Es decir, es dependiente del patrón a aprender, como se puede evidenciar en el reconocimiento de caracteres a mano alzada que se muestra en la Figura **3-1** [Walid y Lasfar, 2014]. Basados en este ejemplo, el entrenamiento para aprendizaje con caracteres uniformes, como los de la parte izquierda de la figura,

reduce la complejidad del patrón, lo que va de la mano con la reducción general del sistema de reconocimiento de patrones de caracteres numéricos. Sin embargo, un caso general debe implicar las posibles variaciones del patrón, como se ilustra en la parte derecha de la Figura **3-1**, con los caracteres a mano alzada. Un sistema como este requiere un aprendizaje más profundo. Es decir, un aprendizaje que permita mayor nivel de abstracción de los patrones, lo que implicará mayor complejidad del sistema de reconocimiento de patrones.

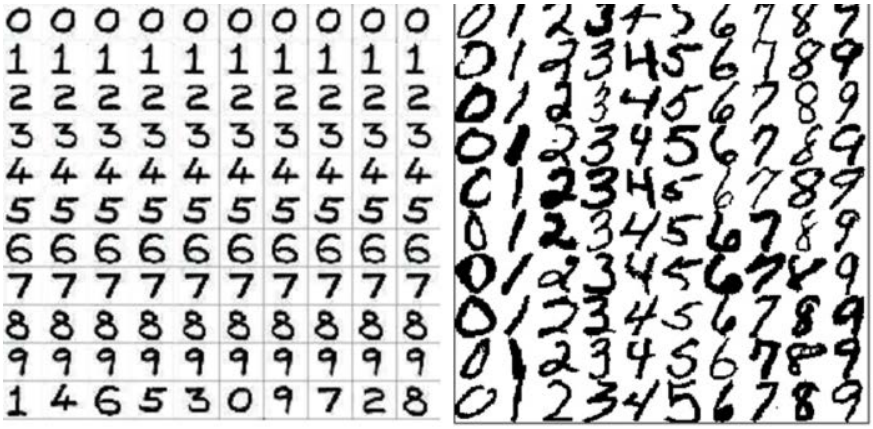


Figura 3-1: Aplicaciones de redes neuronales convolucionales en reconocimiento de texto a mano alzada.

Las técnicas de Deep Learning han surgido como solución al problema de entrenamiento multicapa de las redes neuronales convencionales. Estos problemas se centran en el desvanecimiento del gradiente asociado al error y al sobre-ajuste de los pesos de entrenamiento. Dichos problemas se evidencian en arquitecturas neuronales convencionales de más de una capa oculta, que no logran una mejoría en razón del aprendizaje, o una convergencia a un valor en los pesos de las neuronas de cada capa, generando un estancamiento en el aprendizaje y la profundidad de dichas redes.

Schmidhuber, en [Schmidhuber, 2015], hace referencia a lo descrito como: *"Una arquitectura profunda se refiere al número de niveles de composición de operaciones no lineales en la función aprendida. Mientras que la mayoría de los algoritmos de aprendizaje actuales corresponden a arquitecturas poco profundas (1, 2 o 3 niveles), el cerebro de los mamíferos está organizado*

en una arquitectura profunda, con una percepción de entrada dada, representada en múltiples niveles de abstracción, correspondiendo cada nivel a un área diferente de la corteza, de forma similar al de los seres humanos. El cerebro también parece procesar información a través de múltiples etapas de transformación y representación. Esto es particularmente claro en el sistema visual de primates, con su secuencia de etapas de procesamiento: detección de bordes, formas primitivas, y moviéndose hasta formas visuales gradualmente más complejas. Inspirados en la profundidad arquitectónica del cerebro, los investigadores de redes neuronales habían querido durante décadas entrenar profundas redes neuronales multicapa, pero no hubo intentos exitosos antes de 2006, cuando se obtuvieron resultados positivos con típicamente dos o tres niveles (es decir, una o dos capas ocultas), pero el entrenamiento de redes más profundas produjo consistentemente resultados más pobres."

Entre las técnicas más representativas de Deep Learning se encuentran las máquinas restrictivas de Boltzmann, las redes de creencia profunda y las redes neuronales convolucionales. Estas últimas han mostrado un alto desempeño en el reconocimiento de imágenes, por ejemplo, el reconocimiento de caracteres a mano alzada como el presentado en la Figura **3-1** [Walid y Lasfar, 2014]. Siendo actualmente la técnica más aplicada a diversos campos de las ciencias y la ingeniería, como se evidenció en el estado del arte del capítulo anterior.

3.2. Redes Neuronales Convolucionales

Como se menciona en [Bengio, 2009], las redes neuronales convolucionales fueron inspiradas por la estructura del sistema visual orientado al reconocimiento de objetos. En esencia, una red neuronal convolucional busca transformar gradualmente la imagen, o señal, de entrada, detectando elementos simples inicialmente, hasta aprender detalles específicos del objeto de interés. Es decir, basa el aprendizaje de conceptos complejos, mediante la descomposición de los mismos en elementos más simples, donde el entrenamiento viene dado por establecer gradualmente un grupo de filtros de convolución, los cuales son determinados en función a la

base de datos de entrenamiento y la estructura general de la red. Dicha estructura se compone de una serie de capas consecutivas de convolución - relu - pooling, o variaciones de las mismas, en una etapa denominada de extracción de características, seguida de otra de clasificación. Los filtros conforman un grupo de mapas de características, donde cada mapa se obtiene por la aplicación repetida de la función filtro a través de sub-regiones de toda la imagen de entrada, mediante [Gonzalez y Woods, 2008]

$$h_j^n = \max\left(0, \sum_{k=1}^{K} h_k^{n-1} * w_{kj}^n\right), \tag{3-1}$$

donde h_j determina el mapa de características de salida, h_k el de entrada, que para la etapa inicial responderá a la imagen, w_k corresponde al núcleo de convolución, objeto del aprendizaje, con $K = 1$ si la imagen está en escala de grises o $K = 3$ si es a color. Cada capa que determina la profundidad de la red neuronal convolucional, requiere un h_k para determinar un h_j, el filtro a aprender es operado en cada iteración del entrenamiento sobre la entrada h_k bajo el concepto de convolución, del cual recibe su nombre [Gonzalez y Woods, 2008].

El concepto de convolución en imágenes es asociado a la operación con matrices, donde la definición formal está dada por [Palomare *set al*., 2016], como se muestra a continuación.

Definición: Dada una matriz $\mathbf{A}_{m\times n}$ y una matriz $\mathbf{C}_{(2N+1)\times(2N+1)}$ con dimensiones $(2N+1),\ (2N+1) < m, n$ se define la convolución de las matrices $\mathbf{A}$ y $\mathbf{C}$ como una nueva matriz $\mathbf{D}=\mathbf{A}*\mathbf{C}$, donde en adelante el símbolo $*$ denotará la operación de convolución, definida a partir de

$$d_{(i,j)} = \frac{1}{p} \sum_{r=1}^{2F+1} \sum_{s=1}^{2F+1} a_{(i-N+r-1, j-N+s-1)}\, c_{(r,s)}, \tag{3-2}$$

donde los $a_{(i,j)}$ son los elementos de la matriz $\mathbf{A}$ y

$$p = \sum_{i,j=1}^{2N+1} c_{(i,j)}, \tag{3-3}$$

F corresponde al tamaño lateral del filtro, típicamente cuadrado. Si p calculada mediante (3-3) es igual a cero, se reemplaza por 1 para evitar la indeterminación de (3-2). Como ejemplo, la Figura **3-2** ilustra el proceso de convolución de un filtro de dimensiones 3×3 con una matriz de entrada h_k de dimensiones 7×7. El filtro se desplaza una cantidad de celdas determinada, hasta cubrir todas las celdas de la matriz de entrada, donde dicha cantidad se conoce como *stride (S)*.

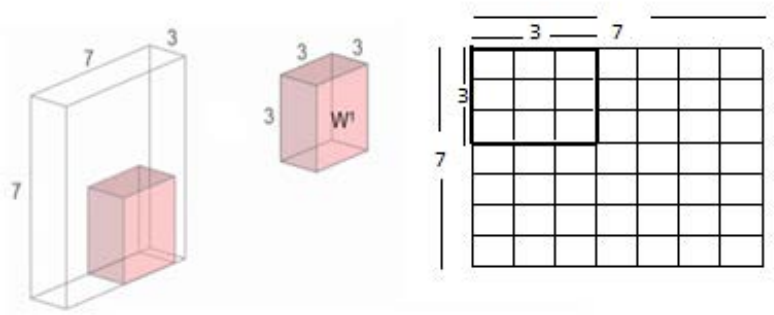


Figura 3-2: Capa de convolución de una red neuronal convolucional.

La Figura **3-3** muestra el resultado de aplicar un filtro de convolución conocido para detección de bordes en imágenes, denominado filtro Laplaciano, a manera de ejemplo de cómo lo realiza una red neuronal convolucional durante la etapa de entrenamiento y después de haber obtenido los filtros finales. Debido a que el filtro debe operar con cada celda de la matriz de entrada, para las celdas laterales se obtendrían valores negativos, por lo que se hace un relleno de bordes a la matriz de entrada denominado *padding (P)*, puede ser basado en relleno con ceros, unos o repitiendo el valor de los bordes.

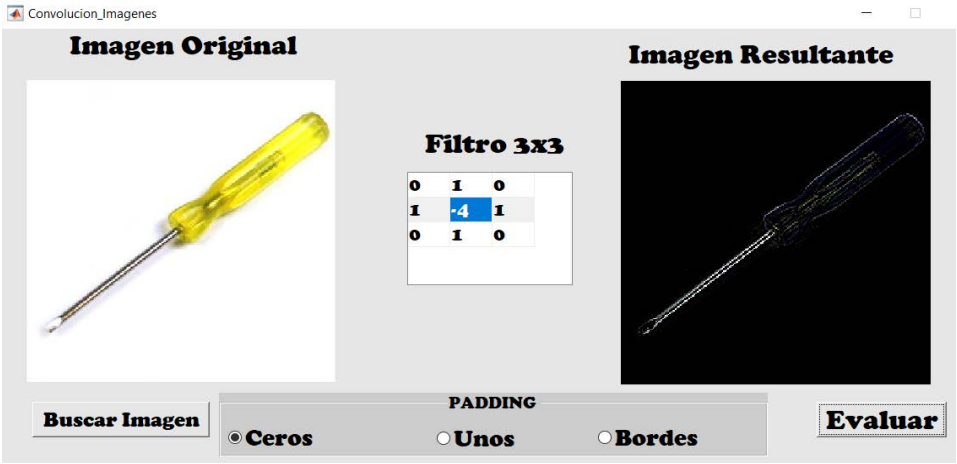


Figura 3-3: Operación de convolución en imágenes

La Figura **3-4** ilustra un ejemplo de la arquitectura de una red neuronal convolucional con la estructura base de este tipo de red.

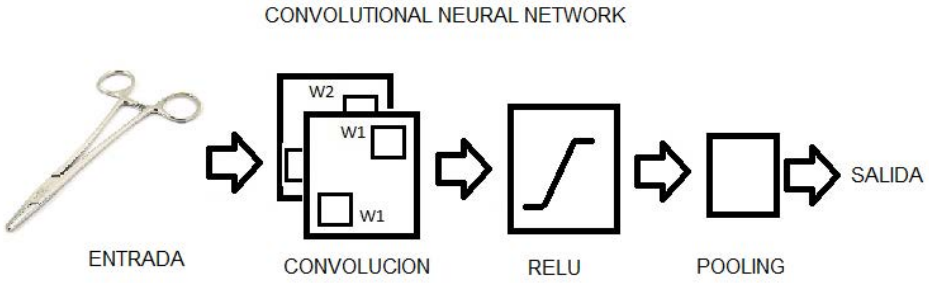


Figura 3-4: Estructura base de una red neuronal convolucional.

La estructura de una red neuronal convolucional cuenta con una entrada basada en imágenes en escala de grises, es decir de una dimensión, o a color referidas a sus componentes rojo, verde y azul, con tres dimensiones correspondientemente. Esta imagen inicia siendo el primer h_n, es decir h_1. Cada h_n cuenta con un alto (H) y ancho conocido (W) , así como una profundidad D_n que será 1, si la imagen es a escala de grises, o 3, si es a color. Dentro de los hiper-parámetros de entrenamiento iniciales para la convolución con cada filtro a aprender, se requiere determinar el tamaño del filtro $(F \times F)$, cantidad de filtros (Nf), un valor inicial de las posiciones de cada uno de los filtros (aleatorio generalmente), el stride y el padding (ver Figura **3-5**), que determinan el h_j de salida.

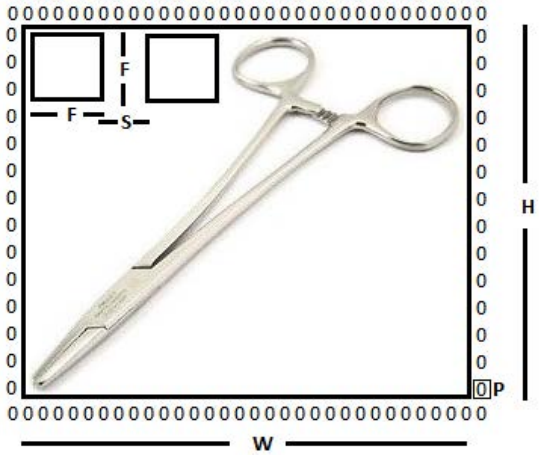


Figura 3-5: Hiper-parámetros de convolución.

El resultado de la convolución ingresa a la capa de RELU (de sus siglas en inglés, Rectified Lineal Unit, o unidad lineal rectificada), la cual es una capa de función de activación tipo rampa, que no presenta saturación por la parte superior, lo que evita saturación del gradiente y no cambia el tamaño del volumen de salida, pero si elimina valores negativos en el filtro.

Para efectuar las operaciones de la capa RELU, se debe determinar el número de capas (n) que ha de tener la red. Las dimensiones del volumen de salida de la capa de convolución a la de RELU se calculan mediante

$$W_{n+1} = ((W_n - F_n + 2P_n))/S_n + 1, \tag{3-4}$$

$$H_{n+1} = ((H_n - F_n + 2P_n))/S_n + 1 \tag{3-5}$$

y

$$D_{n+1} = Nf_n. \tag{3-6}$$

La Figura **3-6** ilustra los filtros de convolución aprendidos para el ejemplo de la Figura **3-4**, donde se han empleado dos bancos de filtros con 10 filtros cada uno.

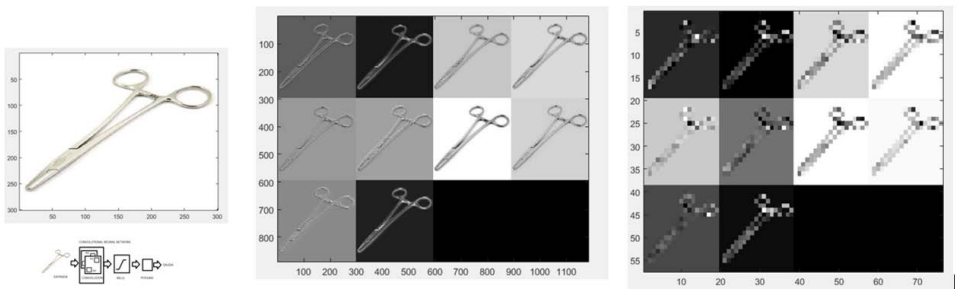


Figura 3-6: Resultado del banco de filtros.

La Figura **3-7** ilustra un ejemplo de la salida de la capa RELU para algunos de los filtros empleados en la red de la Figura **3-4**. La figura de entrada se muestra a la izquierda, dos resultados de convolución en el medio y dos resultados de la capa RELU a la derecha.

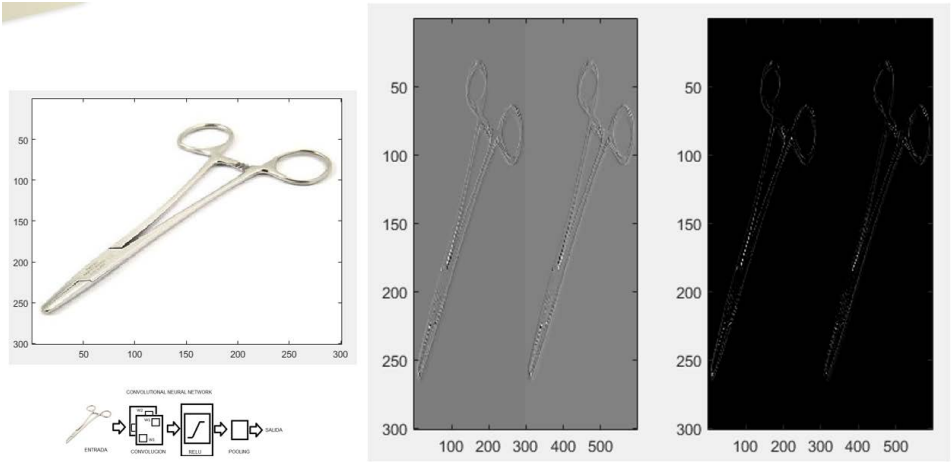


Figura 3-7: Resultado de la capa RELU.

La capa de pooling opera independiente y se encarga de reducir progresivamente el tamaño de las capas, mediante los métodos del máximo o del promedio, los cuales se establecen mediante

$$h_j^n(x,y) = \max_{\bar{x}\in N(x),\bar{y}\in N(y)} h_j^{n-1}(\bar{x},\bar{y}), \qquad (3\text{-}7)$$

y

$$h_j^n(x,y) = \frac{1}{K} \sum_{\bar{x}\in N(x),\bar{y}\in N(y)} h_j^{n-1}(\bar{x},\bar{y}), \tag{3-8}$$

donde h_j determina el nuevo mapa de características de esta etapa, basado en el anterior. La Figura **3-8** ilustra un ejemplo de la salida de la capa de pooling para uno de los filtros empleados en la red de la Figura **3-4**. Para este caso se emplea el método del máximo. La tabla **3-1** ilustra el resultado matemático de ambos tipos de pooling y evidencia la reducción del tamaño del volumen.

Tabla 3-1: Ejemplo pooling.

Entrada			Max		Prom	
4	10	3				
2	0	7	10	10	4	5
9	1	0	9	7	3	2

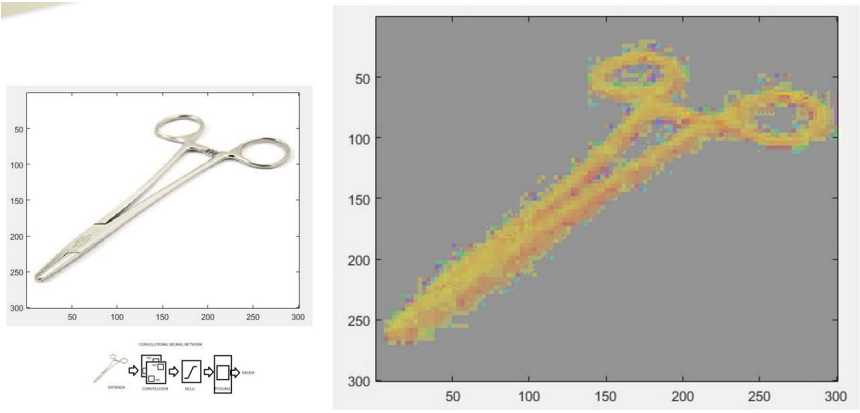


Figura 3-8: Resultado de la capa de pooling.

Otros hiper-parámetros de entrenamiento son:

- Tamaño del pooling.
- Tasa inicial de aprendizaje.
- Tamaño del Batch, que emplea un número normalizado de muestras de entrenamiento y prueba.

- Dropout, que establece aleatoriamente los elementos de en trada a cero con una probabilidad dada para evitar sobre ajuste de la red.

En [Zeiler y Fergus, 2014] se puede profundizar sobre la operación de las redes neuronales convolucionales en aplicaciones de reconocimiento de objetos en imágenes.

3.3. Comparación entre las redes neuronales convencionales y las convolucionales

Dentro de las desventajas de las redes neuronales convencionales, como las backpropagation, que permiten entrenamiento multicapa, la propagación hacia atrás propia del entrenamiento de este tipo de red hace que el error se diluya de forma exponencial desde las últimas capas hasta las iniciales, por lo que las primeras capas no tienen tendencia al cambio. A su vez, la complejidad de entrenamiento de una red neuronal para trabajo con imágenes es elevada. Por ejemplo, para una imagen de entrada de 128×128, es decir de 16.384 píxeles, se requiere una conexión completa entre todas las neuronas de una capa con la siguiente. Es decir, que cada píxel de la imagen de entrada se encuentra conectado con cada neurona de la primera capa de la red. Lo que implica al menos 16.384 neuronas, cada una con 16.384 conexiones y un total de $284'435.456$ pesos a entrenar solo en la primera capa. Por consiguiente, el incremento de capas en la red elevaría demasiado la cantidad de pesos a calcular, por lo que el entrenamiento se volvería demasiado extenso y demorado.

Para evidenciar más claramente la diferencia entre las redes neuronales convencionales y las convolucionales, se presenta un ejemplo de aprendizaje simple, como lo es la identificación de colores, para ello se establecen las arquitecturas de red de cada tipo y se evalúa el desempeño obtenido para cada caso.

3.3.1. Red Neuronal Backpropagation

El primer paso es establecer la base de datos de entrada que se empleará para el entrenamiento. En este caso, se presentan dos esquemas para comparación uno con 12 colores y otro con 18. La base de datos de entrenamiento se ilustra en Figura **3-9**, la cual es entregada a la red como un arreglo de 3 filas por M columnas, donde M es la cantidad de imágenes de entrenamiento, y cada fila representa las componentes R, G y B de cierto color. Cada columna del arreglo contiene el color promedio de cada imagen de entrada, y las 3 filas corresponden a los componentes RGB de dicho color [Pramparo y Moreno, 2017].

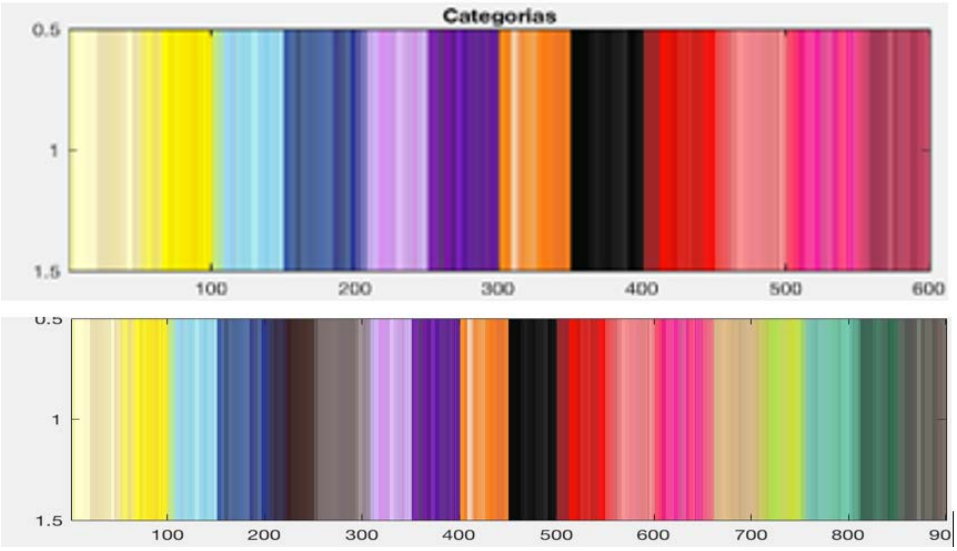


Figura 3-9: Base de datos de colores utilizados.

Debido a la respuesta típica de un red neuronal, en un rango numérico derivado de la función de activación empleada, a cada patrón de color se le asignó un código binario para clasificarlo, como se muestra en la Tabla **3-2**, generando así un arreglo objetivo Y, de 5 filas por M columnas, donde el 5 equivale a la cantidad de bits del código binario.

Tabla 3-2: Codificación de los patrones para la clasificación de 12 colores y 18 colores.

COLOR	CÓDIGO
Amarillo claro	00000
Amarillo intenso	00001
Azul claro	00010
Azul oscuro	00011
Lila	00100
Morado	00101
Naranja	00110
Negro	00111
Rojo	01000
Rosado claro	01001
Rosado intenso	01010
Vinotinto	01011
Amarillo claro	00000
Amarillo intenso	00001
Azul claro	00010
Azul oscuro	00011
Café oscuro	00100
Gris	00101
Lila	00110
Morado	00111
Naranja	01000
Negro	01001
Rojo	01010
Rosado claro	01011
Rosado intenso	01100
Arena	01101
Verde lima	01110
Verde marino	01111
Verde oscuro	10000
Vinotinto	10001

Las dos arquitecturas de red implementadas para cada base de datos se ilustra en la Figura **3-10**, donde la red de la Figura **3-10**(a) está orientada a la clasificación de 12 colores, y la red de la Figura **3-10**(b) a los 18 colores.

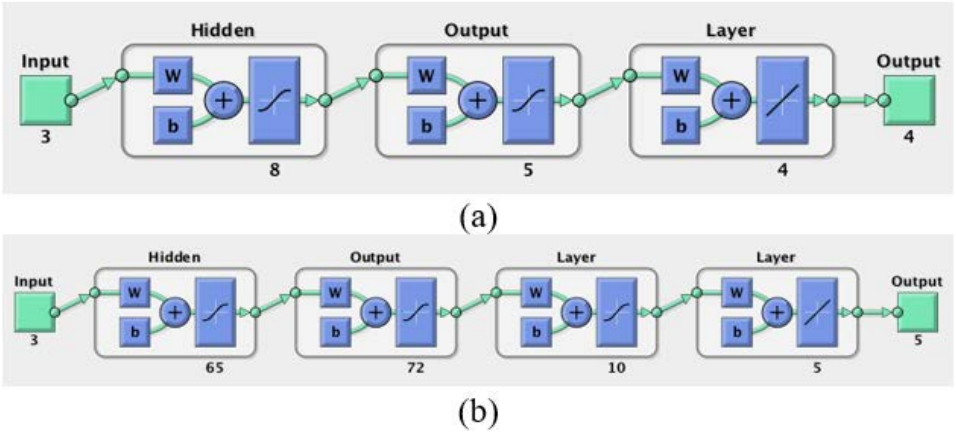


Figura 3-10: Estructura de la red neuronal multicapa backpropagation para clasificar 12 colores (a) y 18 colores (b).

En la Tabla **3-3** se presentan los porcentajes de exactitud obtenidos para cada red, y en la Figura **3-11** el reconocimiento de ambas redes sobre las imágenes de prueba, donde los colores de la gráfica superior corresponden a los colores de entrenamiento, y los colores de la gráfica inferior a los de prueba (arriba) y los de clasificación (abajo).

Tabla 3-3: Porcentajes de exactitud para la clasificación de 12 y 18 colores por medio de una FCNN.

	Cantidad de colores	Porcentaje de exactitud
Red 1	12 colores	93,35 %
	18 colores	35,23 %
Red 2	12 colores	33,33 %
	18 colores	22,23 %

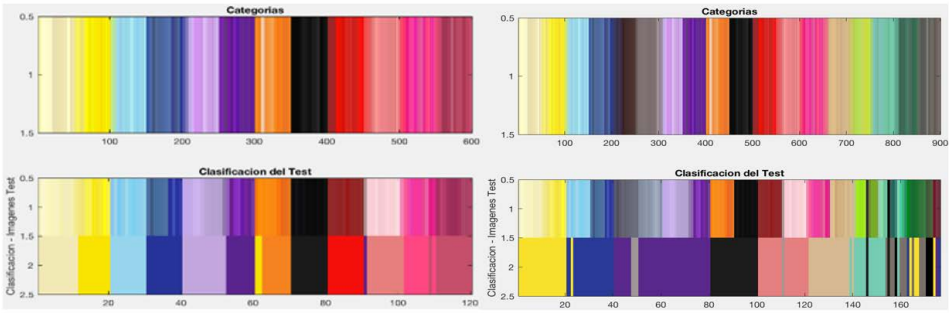


Figura 3-11: Clasificación para los colores de prueba, 12 colores (a) y 18 colores (b).

Como se puede observar, la red 1 logró reconocer casi todos los colores prueba, obteniendo un 93% de exactitud, pero al aumentar la cantidad de patrones a 18, la exactitud se redujo a un 35%, confundiendo casi todos los colores, y sin lograr reconocer casi ninguna de las tonalidades verdes. La red 2 generalizó los colores amarillo, azul y morado, y los demás no logró clasificarlos. Se variaron la cantidad de neuronas por capa para tratar de mejorar el reconocimiento, pero los porcentajes de exactitud rondaban el 22%.

3.3.2. Red Neuronal Convolucional

Para realizar el reconocimiento de la misma base de datos de colores, se implementó una arquitectura de red neuronal convolucional que se muestra en la Figura **3-4**, la cual no requiere ser muy profunda, ya que no requiere obtener patrones detallados, sino diferentes tonalidades de un mismo color, por lo que para el reconocimiento de 12 y 18 categorías, se utiliza la misma arquitectura. La Figura **3-12** ilustra los hiper-parámetros de entrenamiento empleados.

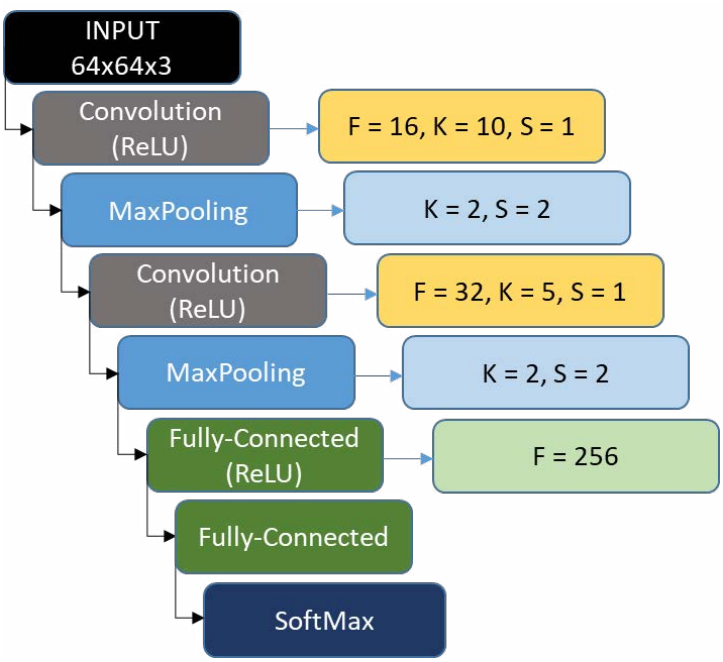


Figura 3-12: Arquitectura de la CCN empleada.

Una vez la red ha sido entrenada, se realiza la validación, obteniendo los resultados presentados en la Tabla **3-4**, en donde se encuentran los porcentajes de exactitud para la red entrenada con 12 colores como para 18. Teniendo en cuenta estos resultados, se puede observar que la arquitectura entrenada tanto para 12 como para 18 colores mantiene una exactitud por encima del 93%, lo cual representa una exactitud muy buena para aplicaciones de discriminación de color y evidentemente un desempeño superior a la de red neuronal convencional, sin necesidad de emplear una arquitectura profunda, determinada por más de tres capas de convolución. En este ejemplo, la principal dificultad para alcanzar el 100% en la predicción son las tonalidades semejantes de algunos colores, por ejemplo, las variaciones de amarillo.

Tabla 3-4: Porcentajes de exactitud para la clasificación de 12 y 18 colores por medio de una red neuronal convolucional.

Cantidad de colores	Porcentaje de exactitud
12 colores	$95.33\,\%$
18 colores	$93.67\,\%$

3.4. Problema de identificación de una CNN en ambientes 3D

Si bien en el estado del arte se planteó el gran auge de las redes neuronales convolucionales en aplicaciones de reconocimiento de imágenes, también se estableció que estas técnicas de Deep Learning siguen en desarrollo. Esto debido a que su aplicación particular para resolver un problema todavía presenta posibles mejoras, como en el caso de emplear una red neuronal convolucional para identificación de objetos en un ambiente dinámico, donde la variación de la distancia entre cámara y objeto, hace resaltar o reducir características de este.

Para evidenciar la falencia que las arquitecturas neuronales convolucionales presentan en ambientes dinámicos, en los que la cámara se mueve hacia o desde el objeto, se realiza el entrenamiento convencional de una red neuronal convolucional, estableciendo inicialmente la mejor arquitectura para la identificación de objetos en una plataforma multi-herra-

mientas, manteniendo una distancia fija de la cámara al objeto de 60 cm. El diseño de la arquitectura de la red neuronal convolucional se realiza por medio de iteraciones, en función a las capas de convolución y tamaño de sus respectivos filtros, que permiten determinar cuál es la arquitectura final que converge en una discriminación clara de dichas herramientas.

Para observar el comportamiento de las diferentes arquitecturas posibles, se establece inicialmente una base de datos con imágenes de cada una de las herramientas a identificar, sometidas a rotaciones y traslaciones leves. Posteriormente, se determinan arquitecturas posibles realizando cambios en el kernel de convolución y la cantidad de estos que se emplearán, como se expone a continuación.

3.4.1. Base de datos a distancia fija

Para determinar la arquitectura final de la red y poder evaluarla, se debe disponer de una base de datos de imágenes de entrenamiento y una para la validación, al igual que ocurre con las redes neuronales convencionales. La base de datos de entrenamiento permite a la red aprender sobre las características de cada clase, mientras que la de validación emplea imágenes que permiten evidenciar qué tan bien quedó aprendida cada clase, mediante imágenes que no se han presentado a la red. Para este caso, se construyó una base de datos de 800 imágenes con diferentes características de rotación y traslación, para cada una de las cuatro clases de herramientas a aprender: pinzas, destornillador, bisturí y tijeras, como se aprecia en la Figura **3-13**.

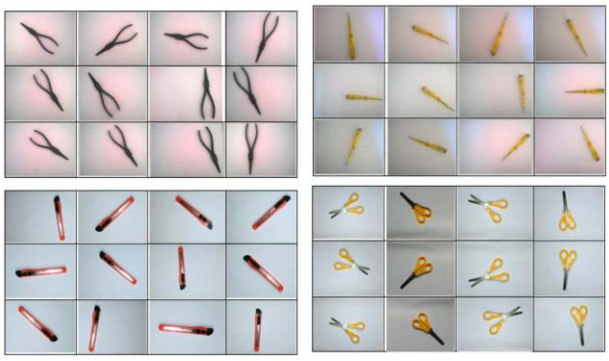


Figura 3-13: Base de datos de entrenamiento a distancia fija.

La Tabla **3-5** resume la cantidad de imágenes empleadas para el entrenamiento y validación del caso, a una distancia fija aproximada de 60 cm.

Tabla 3-5: Distribución de la base de datos.

Conjunto de imágenes		
Herramienta	Entrenamiento	Validación
Pinzas	125	75
Destornillador	125	75
Bisturí	125	75
Tijeras	125	75

3.4.2. Arquitecturas de red

A fin de determinar la red convolucional final que mejor desempeño ofrezca en el aprendizaje de las herramientas establecidas, se implementaron diferentes arquitecturas de redes neuronales convolucionales, basadas en combinaciones de la estructura básica mostrada en la Figura **3-4**, las cuales son evaluadas según la exactitud presentada en la discriminación de estas y calculadas mediante una matriz de confusión. Debido a que se cuenta con pocas clases y con herramientas aparentemente distinguibles unas de las otras, se inició con una arquitectura simple, dicha arquitectura se va robusteciendo para observar el desempeño de cada red, tanto por la variación de hiper-parámetros, como de la profundidad de la red, según se ilustra en la Tabla **3-6**. Para lograr determinar una arquitectura se deben realizar iteraciones que varíen los parámetros de la red en cuanto a capas, tamaño de filtros y cantidad de estos, según se indica en el Apéndice A. Para este caso se toman cinco arquitecturas representativas de un grupo de 30 combinaciones probadas.

Tabla 3-6: Arquitecturas de red evaluadas.

Nombre	Tipo	Kernel		Filtros
Arq 1	Convolution	10x10	S=2	10-10/50
	MaxPooling	8×8	S=2	20-20
	Convolution	5×5	S=2	30-30
	MaxPooling	10×10	S=2	40-40
	Full-Connected	-		
	Softmax	4		
Arq 2	Convolution	26×26	S=2	10-10/30
	MaxPooling	6×6	S=2	
	Convolution	6×6	S=2	
	MaxPooling	6×6	S=2	
	Full-Connected	-		
	Softmax	4		
Arq 3	Convolution	36×36	S=2	10-20
	MaxPooling	5×5	S=2	
	Convolution	7×7	S=3	
	Full-Connected	-		
	Softmax	4		
Arq 4	Convolution	4×4	S=1 / P=2	10-20-40-80-200 20-20-40-80-200
	Convolution	4×4	S=2	
	MaxPooling	2×2	S=2 / P=1	
	Convolution	5×5	S=2	
	Convolution	6×6	S=2	
	MaxPooling	2×2	S=2	
	Convolution	4×4	S=2	
	Full-Connected	-		
	Softmax	4		
Arq 5	Convolution	4×4	S=1 / P=2	20-20-50-50-200
	Convolution	4×4	S=2	
	MaxPooling	2×2	S=2 / P=1	
	Convolution	5×5	S=2	
	Convolution	5×5	S=2	
	MaxPooling	2×2	S=2	
	Convolution	4×4	S=2	
	MaxPooling	3×3	S=2	
	Full-Connected	-		
	Softmax	4		

En cada entrenamiento se lograron observar variaciones en la precisión de cada red, donde el comportamiento esperado es que el error descienda haciendo llegar la precisión al 100%. La Figura **3-14** permite apreciar el resultado del entrenamiento de cada una de las cinco arquitecturas

escogidas, que llegaron a un 100%. Se pueden observar variaciones en la forma como aprende la red en cada entrenamiento, mientras más compleja es la arquitectura, más le cuesta a la red aprender (mayor número de iteraciones).

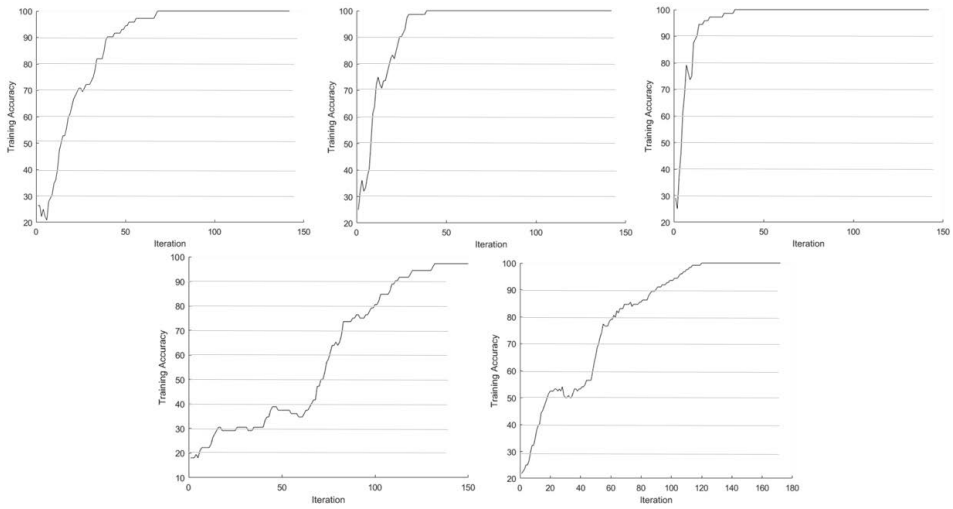


Figura 3-14: Precisión de entrenamiento.

La validación de las arquitecturas de las redes entrenadas se realizó mediante una matriz de confusión, que permite validar la exactitud de cada una en el reconocimiento de las diferentes clases: A, para las pinzas, B, para los destornilladores, C, para los bisturíes, y D, para las tijeras. La Tabla **3-7** muestra la nomenclatura empleada para la matriz de confusión, donde AA representa las imágenes de pinzas que fueron clasificadas como tal, AB representa pinzas que fueron clasificadas como destornilladores, AC, representa pinzas que fueron clasificadas como bisturíes, AD representa pinzas que fueron clasificadas como tijeras, y de manera similar se procede con las otras opciones de nomenclatura. A partir de la matriz de confusión, se puede calcular la exactitud (E) de la clasificación de la cada red, que está dada mediante el cálculo de los falsos y los verdaderos positivos (VP) de las clases pinzas, destornillador, bisturí y tijeras. En la Tabla **3-7**, los verdaderos positivos están representados mediante AA, BB, CC *y* DD, mientras que las demás representaciones están relacionadas con falsos positivos. Para lo que la exactitud puede ser calculada como

$$E(\%) = \frac{\sum VP}{\sum Tt} \times 100, \tag{3-9}$$

donde $\sum VP = AA + BB + CC + DD$, representa el total de verdaderos positivos y Tt representa el total de clasificaciones hechas, calculado como

$$\sum Tt = AA + AB + AC + AD + BA + BB + BC + BD + CA + CB + CC + CD + DA + DB + DC + DD.$$

Tabla 3-7: Matriz de confusión para la prueba a distancia fija de 60 cm.

		CLASIFICADA			
		Pinzas	Destornillador	Bisturí	Tijeras
REAL	Pinzas	AA	AB	AC	AD
	Destornillador	BA	BB	BC	BD
	Bisturí	CA	CB	CC	CD
	Tijeras	DA	DB	DC	DD

Los resultados obtenidos en el entrenamiento de las cinco arquitecturas, realizando diferentes variaciones que incluyen dropout y bacht-normalization, se ilustran en la Tabla **3-8**, incluyendo el máximo error obtenido (MáxErr), que se encuentra entre un 15% a un 66%.

Tabla 3-8: Mejores resultados por arquitectura para distancia fija.

Característica \ Arquitectura	Arq 1	Arq 2	Arq 3	Arq 4	Arq 5
MaxErr	66.7 %	66.5 %	44.7 %	24.7 %	15 %
Exactitud	37.3 %	34 %	53.7 %	87.3 %	93.8 %
AA	46	57	87	92	96
BB	60	65	96	112	116
CC	60	46	89	102	121
DD	16	47	78	92	119

Se logra evidenciar que una arquitectura simple, como Arq 1, no genera un buen aprendizaje para el caso particular de discriminación de las cuatro categorías de herramientas deseadas, ni siquiera con la variación de hiper-parámetros realizadas en las arquitecturas 2 y 3. Dada la multipli-

cidad de características presentadas por las cuatro clases a aprender, los requerimientos de la arquitectura de red se evidencian más exigentes, en comparación con el aprendizaje de múltiples clases de estructuras simples.

Un aumento en la profundidad de la red convolucional mejora significativamente el aprendizaje de características. Al igual que, el aumentar la cantidad de filtros en la capa de entrada. Lo que permite concluir que, la arquitectura 5 es la mejor opción para el caso de aprendizaje de herramientas con cuatro categorías. Finalmente, la arquitectura a emplear tiene la estructura que se muestra en la Figura **3-15**.

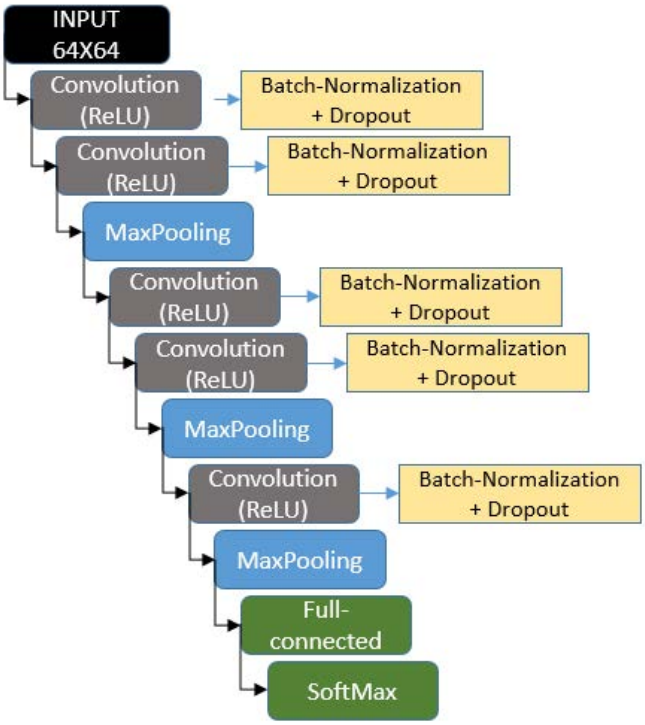


Figura 3-15: Arquitectura final establecida.

De esta forma, en función a las curvas de eficiencia de aprendizaje y matriz de confusión, se da cumplimiento al objetivo específico número uno, en donde se logra determinar una arquitectura eficiente de Deep Learning para la mejor discriminación de un grupo de cuatro herramientas.

3.4.3. Validación en profundidad

Para determinar el desempeño de la red convolucional entrenada en la discriminación de las categorías aprendidas, pero con variaciones de profundidad, se establece una base de datos de prueba según se relaciona en la Tabla **3-9**. En la Figura **3-16** se ilustra una muestra de la misma. El tamaño del volumen de entrada se evidencia significante en relación a la pixelación que recibe la imagen al ingresar a la red. Se observa que un redimensionamiento de 64×64 píxeles pierde las características relevantes de cada categoría (Figura **3-17**), mientras que uno de 128×128 permite mantener las características suficientes, compensando el mayor número de operaciones que debe generar la red y que implican aumento en el costo computacional. De forma tal que, el desempeño obtenido para una imagen de 128×128 píxeles se tabula por distancia, según el acierto en la categoría correspondiente, como se muestra en la Tabla **3-10**. Se puede evidenciar que, a medida que se acerca el objeto, o se aleja demasiado, se pierde precisión en la identificación, obteniendo una precisión promedio en el peor caso de 43.4% a una distancia de 20 cm, lo que implica una degradación de más del 50% del desempeño de la red.

Tabla 3-9: Base de datos de profundidad.

Herramientas / Profundidad	Pinzas	Destornillador	Bisturí	Tijeras
20 cm	100	100	100	100
40 cm	100	100	100	100
60 cm	100	100	100	100
80 cm	100	100	100	100

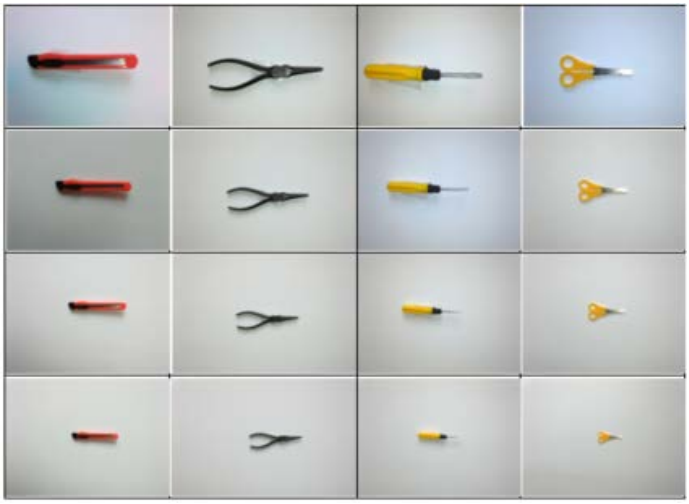


Figura 3-16: Base de datos en función a la distancia.

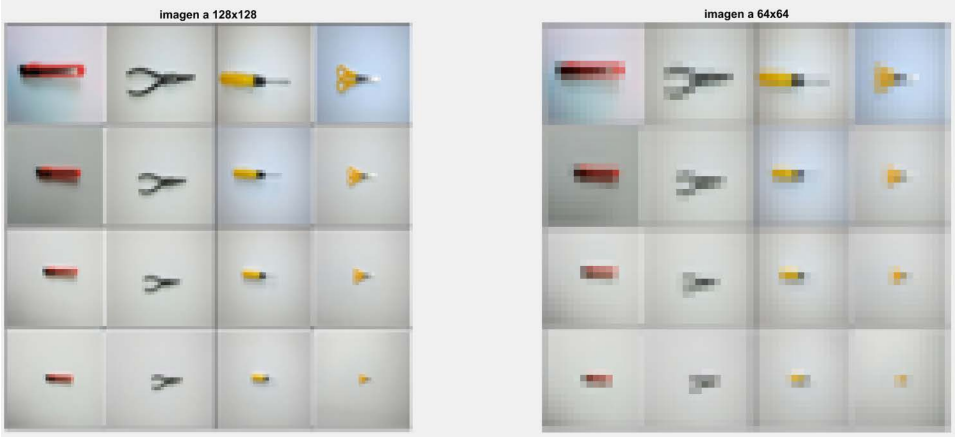


Figura 3-17: Redimensionamiento de entrada.

Tabla 3-10: Acierto en profundidad.

Herramientas / Profundidad	Pinzas	Destornillador	Bisturí	Tijeras
20 cm	40,4 %	51,1 %	48,4 %	33,1 %
40 cm	61,6 %	74,7 %	65,1 %	63,4 %
60 cm	91,5 %	94,2 %	95,8 %	93,2 %
80 cm	51,6 %	84,7 %	85,1 %	61,6 %

Como validación adicional se emplea la arquitectura de la Figura **3-15**. Pero, con la base de datos tomada a 40 cm de distancia entre objeto y cámara, donde el peor caso se presenta para una distancia de 80 cm, con una precisión promedio del $38{,}91\%$. Como era de esperarse, al alejarse la cámara, desapareen características. Lo que implica la pérdida del reconocimiento y, de ahí, la baja tasa de precisión. Con la nueva base de datos, los filtros cambian y esto genera las variaciones en la respuesta de la red, incluso empleando la misma arquitectura.

Al validar las activaciones de la red a 40 cm con un objeto determinado (tijeras para este caso), se presentan cerca de un 20% más de activaciones, que al alejar el objeto a 60 cm. Esto se evidencia en la Figura **3-18**, donde se observa que a 40 cm mejora el reconocimiento.

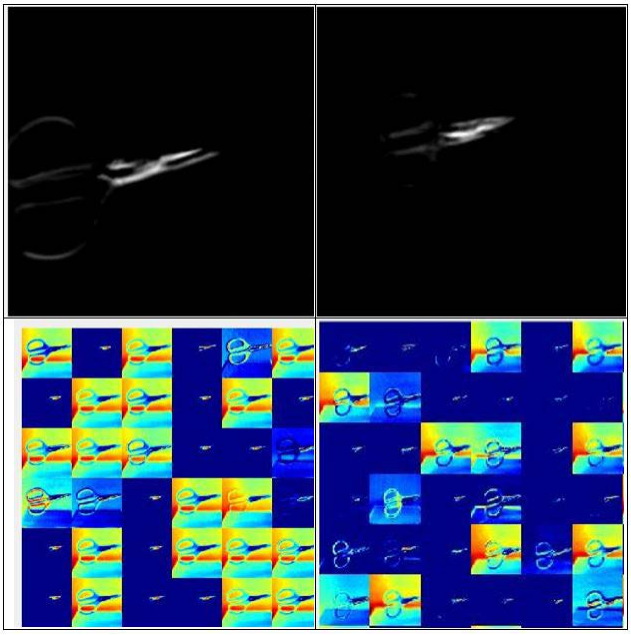


Figura 3-18: Activaciones a 40 y 60 cm.

Una alternativa de solución consiste en ampliar la base de datos de la red inicialmente entrenada, incluyendo las imágenes de los objetos a las diferentes distancias evaluadas (800 imágenes). Bajo este esquema, la Figura **3-19** ilustra el desempeño en el entrenamiento de la red basada en

la arquitectura escogida en la Sección 3.4.2, con la base de datos conjunta. Se observa que a la red le cuestan muchas más iteraciones el aprender las características, a la par que el desempeño baja al $79{,}71\%$ y le toma casi el doble de tiempo de entrenamiento con la nueva base de datos. Esto conlleva a una degradación general en la identificación de objetos bajo este aspecto. Lo cual era de esperarse, debido a que las características de aprendizaje ahora divergen con las nuevas imágenes. En la Tabla **3-11**, se observa la matriz de confusión obtenida, donde se aprecia que, con un aumento en la base de datos, las categorías se confunden entre ellas.

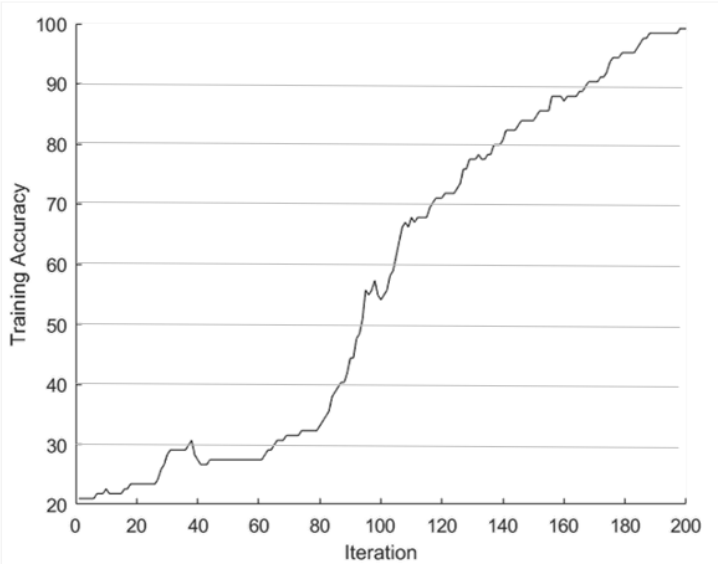


Figura 3-19: Entrenamiento de la red

Tabla 3-11: Matriz de confusión para la prueba a distancia variable.

		CLASIFICADA			
		Pinzas	Destornillador	Bisturí	Tijeras
REAL	Pinzas	130	0	52	34
	Destornillador	0	132	12	66
	Bisturí	68	0	142	0
	Tijeras	0	8	50	106

Principalmente, en la clase bisturí, donde 56 imágenes de otras categorías fueron interpretadas como esta, ya que por estar a diferentes distancias exhiben diferentes características y las más alejadas se aprecian como un objeto más uniforme, como lo es la estructura morfológica del bisturí.

Las pruebas de profundidad evidencian que la red neuronal convolucional, que tenía el mejor comportamiento para clasificar herramientas a distancia fija, no tiene un buen comportamiento cuando la distancia de toma de la imagen varía. Sin importar si se amplía la base de datos de entrenamiento, incluyendo imágenes tomadas a diferentes distancias. En el siguiente capítulo se abordará una solución a este problema.

Capítulo 4

Arquitecturas Neuronales Convolucionales Propuestas

Frente al problema expuesto en relación a la clasificación de los objetos al presentarse una variación espacial de captura de la imagen, se plantea como solución una arquitectura de red convolucional paralela especializada en las diferentes perspectivas del objeto a reconocer, en función a la profundidad en que se encuentre. Es decir, se realiza el entrenamiento de un conjunto de redes neuronales, donde cada una aprenderá a reconocer el objeto desde una distancia particular y se irá activando cada red a medida que se acerca o aleja del objeto, como se ilustra en Figura **4-1.** Esta solución se propone en función al trabajo expuesto en [Ciregan *et al.*, 2012], donde se presenta una arquitectura paralela basada en redes neuronales convolucionales, como una mejora al aprendizaje de características en el reconocimiento del mismo objeto, la cual presenta ventajas frente a la red neuronal convolucional convencional, pero en este trabajo la red no esta orientada a operar en profundidad y emplea la misma base da datos para todas las redes.

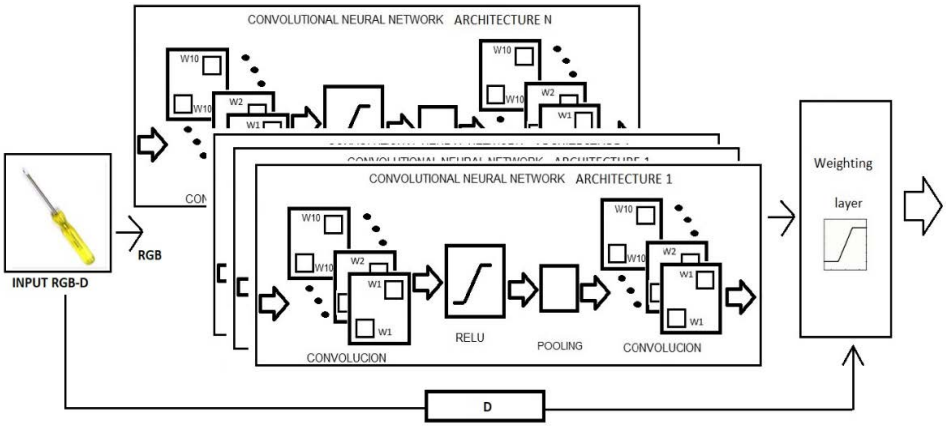


Figura 4-1: Arquitectura propuesta con ponderación aritmética.

El Algoritmo 2 expone los pasos establecidos para solucionar el problema de variación de características ante la clasificación de la red neuronal convolucional cuando varia la distancia de captura de la imagen.

Algoritmo 2: Diseño de arquitectura 3D

Begin

Paso 1: Establecer las bases de datos de entrenamiento en función a 4 distancias de prueba: 20 cm, 40cm, 60cm y 80 cm.

Paso 2: Establecer las arquitecturas neuro-convolucionales para cada distancia, evaluadas por transferencia de aprendizaje y diseño independiente.

Paso 3: Diseñar una capa de ponderación final que determine la salida de la red en función a las activaciones de cada red individual operando en paralelo, mediante la comparación de la solución por una ecuación matemática general y un sistema de inferencia difusa.

Paso 4: Establecer la mejor arquitectura final y evaluarla.

end

La entrada al tipo de red propuesto requiere de la imagen a color más un canal de distancia, este canal se puede establecer mediante la información suministrada por una cámara de captura RGB-D, como ejemplo se presenta la Blaster Senz3D Creative, cuyo rango de visión 3D se encuentra de los 0.2 a los 1.5 metros, la cual se puede apreciar en la Figura **4-2.**

Figura 4-2: Sensor de captura RGB-D

Se presentan dos opciones para el entrenamiento de la multi-red. La primera consiste en un pre-entrenamiento convolucional que, mediante transferencia de aprendizaje, permita emplear la red obtenida en la Sección 3.4.2 como arquitectura base ya conocida, donde se entrenen cuatro de estas redes, cada una con una base datos de profundidad diferente. La segunda opción consiste en determinar cada una de las cuatro arquitecturas de las redes convolucionales de forma independiente, para ser entrenadas con bases de datos respectivas.

4.1. Arquitectura basada en Transferencia de Aprendizaje

A partir de la red neuronal convolucional para discriminación de herramientas obtenida en la sección 3.4.2, que permite la discriminación de las cuatro categorías y cuya eficiencia fue del $93{,}8\%$ en la clasificación a una distancia fija, se emplea su arquitectura como base para realizar la tarea de transferencia de aprendizaje [Pan y Yang, 2010], donde para cada una de las cuatro redes re-entrenadas se utiliza la base de datos de imágenes ilustrada en la Tabla **4-1.**

Tabla 4-1: Base de imágenes en Profundidad.

Red/Profundidad	Pinzas	Destornillador	Bisturí	Tijeras
1 / 20 cm	100	100	100	100
2 / 40 cm	100	100	100	100
3 / 60 cm	200	200	200	200
4 / 80 cm	200	200	200	200

La diferencia en la cantidad de imágenes empleadas por cada red, según la profundidad, está determinada por la cantidad de características que deben aprender los filtros en el entrenamiento. Mientras más cerca, o más lejos, está la imagen, más variaciones se encuentran respecto a la red inicial. Una imagen cerca al objeto permite una mejor identificación, pues las características exhibidas son fácilmente extraíbles; requiriendo un menor uso de recursos computacionales, lo que implica menor tiempo en las tareas de entrenamiento y clasificación. En este caso, el tiempo promedio de entrenamiento es de 470 segundos y de 0.612 segundos en la clasificación. La Figura **4-3** ilustra el desempeño en el entrenamiento de las nuevas redes a $20,\ 40$ y 80 cm, para 60 cm se presentó en el capítulo anterior.

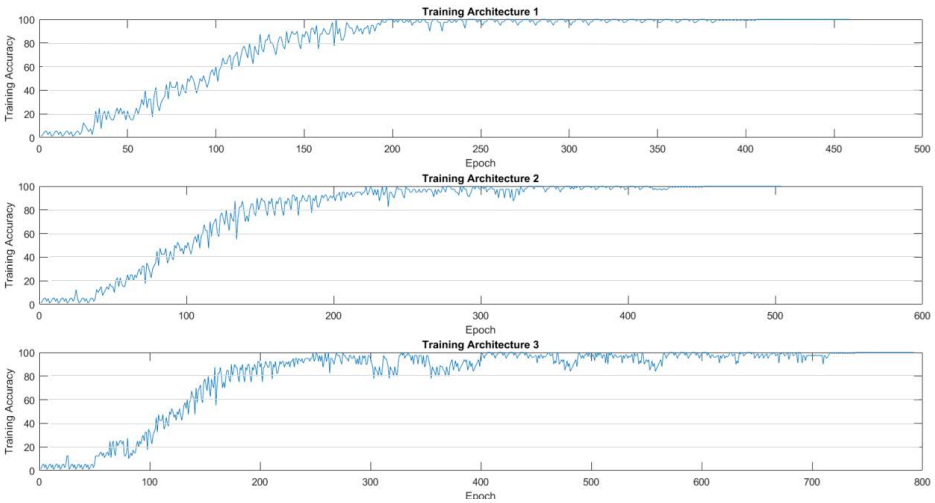


Figura 4-3: Desempeño por transferencia de aprendizaje para las redes a diferentes distancias: 20 cm (arriba), 40 cm (en el medio) y 80 cm (abajo).

En relación a la eficiencia en la clasificación se obtuvo un $86{,}25\%$, $83{,}49\%$ y un $82{,}37\%$ para cada red, según se observa en las Tablas **4-2** - **4-4**, a partir de las matrices de confusión respectivas. Se puede observar cómo a distancias más cercanas al objeto la clasificación mejora y que las características son más fácilmente reconocidas, como es el caso de las tijeras, que presentan muy pocos falsos positivos a distancias cortas, que aumentan considerablemente a distancias mayores, siendo confundidas principalmente con el bisturí.

Tabla 4-2: Matriz de confusión para la Red 1 - 20 cm. Eficiencia = 86.25%.

		CLASIFICADA			
		Pinzas	Destornillador	Bisturí	Tijeras
REAL	Pinzas	92	0	12	4
	Destornillador	0	85	20	0
	Bisturí	17	0	87	1
	Tijeras	0	0	1	81

Tabla 4-3: Matriz de confusión para la Red 2 - 40 cm. Eficiencia = 83.49%.

		CLASIFICADA			
		Pinzas	Destornillador	Bisturí	Tijeras
REAL	Pinzas	86	0	16	6
	Destornillador	0	79	26	0
	Bisturí	14	0	90	1
	Tijeras	2	0	1	79

Tabla 4-4: Matriz de confusión para la Red 4 - 80 cm. Eficiencia = 82.37%.

		CLASIFICADA			
		Pinzas	Destornillador	Bisturí	Tijeras
REAL	Pinzas	145	0	25	8
	Destornillador	0	165	12	33
	Bisturí	34	0	205	0
	Tijeras	0	8	21	144

4.2. Arquitectura basada en Aprendizaje Individual

Para la segunda opción, se implementaron cuatro arquitecturas diferentes para su entrenamiento individual, para la red orientada a reconocimiento de media distancia (60 cm) se mantiene la misma estructura de la Figura **3-15**. Para las otras se emplearon tres arquitecturas diferentes cuyas características son mostradas en la Tabla **4-5**.

Tabla 4-5: Arquitecturas de red basadas en profundidad.

Nombre	Tipo	Kernel		Filtros
Arq 1 - 20 cm	Convolution	10×10	S=2	10-10/50 20-20 30-30 40-40
	MaxPooling	8×8	S=2	
	Convolution	5×5	S=2	
	MaxPooling	10×10	S=2	
	Fully-Connected	1		
	Softmax	5		
Arq 2 - 40 cm	Convolution	26×26	S=2	10-10/30
	MaxPooling	6×6	S=2	
	Convolution	6×6	S=2	
	MaxPooling	6×6	S=2	
	Fully-Connected	1		
	Softmax	5		
	MaxPooling	5×5	S=2	
	Convolution	7×7	S=3	
	Fully-Connected	1		
	Softmax	5		
Arq 4 - 80 cm	Convolution	26×26	S=2	10-10/30
	MaxPooling	6×6	S=2	
	Convolution	6×6	S=2	
	MaxPooling	6×6	S=2	
	Fully-Connected	1		
	Softmax	5		
	MaxPooling	5×5	S=2	
	Convolution	7×7	S=3	
	Fully-Connected	1		
	Softmax	5		

Las diferencias en las arquitecturas se dan por la complejidad del aprendizaje que debe manejar cada una de las redes, acorde a la cantidad de información en la imagen. Mientras mayor sea la distancia de la captura de la imagen al objeto, mayor información ingresa a la red, la cual debe discrimi-

nar claramente el objeto del resto. Por ejemplo, gran parte corresponderá al fondo, el cual deberá ser discriminado del aprendizaje respecto al objeto.

Cada una de las arquitecturas implementadas fue entrenada con la base de datos de entrada que relaciona la Tabla **4-1**, donde el 80% de los datos data se usan para el entrenamiento y el 20% restante para validación. La Figura **4-4** ilustra el resultado del comportamiento en el entrenamiento de las arquitecturas 1, 2 y 4, respectivamente. Se puede observar que la arquitectura 1 presentó la curva de aprendizaje mas rápida. Pero, a su vez, asimila parte del fondo como del objeto en sí. Mientras que la arquitectura 4 reconoce patrones más complejos del mapa de características del objeto, lo cual explica el retardo de aprendizaje evidenciado en esta red, discriminando claramente el objeto del fondo. De igual forma, se evidencia que la complejidad de la arquitectura y el objeto de aprendizaje respecto a la imagen, afectan el tiempo de entrenamiento. Para este caso, la arquitectura1 requiere la mitad de épocas de entrenamiento que la arquitectura 4.

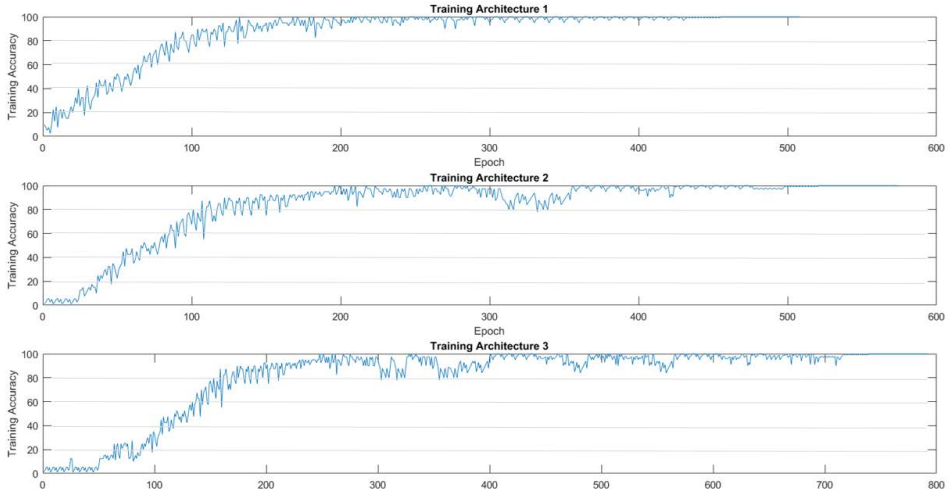


Figura 4-4: Desempeño por aprendizaje individual para las redes a diferentes distancias: 20cm (arriba), 40 cm (en el medio) y 80 cm (abajo).

4.3. Arquitectura final de aprendizaje en profundidad

Para determinar el desempeño de cada arquitectura, se evalúa la capacidad de predicción por escenario planteado, en función a la profundidad

de la captura de la imagen, caso A por transferencia de aprendizaje y caso B por diseño individual de cada red convolucional. En la Tabla **4-6** se puede apreciar la tasa de error obtenida que, de forma general, permite concluir que el caso B permite el mejor reconocimiento a profundidad dinámica.

Tabla 4-6: Error de clasificación.

Caso	Arquitectura 1	Arquitectura 2	Arquitectura 3	Promedio
A	12.67 %	10.72 %	6.6 %	9.99 %
B	10.56 %	7.87 %	6.5 %	8.31 %

Los resultados mostrados en la Tabla **4-6** son obtenidos de la validación individual de cada red. Debido a que la funcionalidad del desarrollo está orientada a cambios dinámicos de la profundidad, donde la clasificación se debe realizar en tiempo real, se emplea una arquitectura final conjunta con una capa de salida de ponderación basada en la profundidad de captura en la imagen de entrada.

La capa de ponderación adicional genera una suma ponderada de las respuestas individuales de cada salida de las redes de la arquitectura final, en función a la profundidad

$$P_c = \sum_{n=1}^{m} (1 + O_n)^{1/(n-d)}, \tag{4-1}$$

donde P_c corresponde a la ponderación que determina la categoría de salida, m representa el número de redes que posee la arquitectura final, la distancia normalizada (d) se toma como la distancia en centímetros de la cámara (do) sobre la distancia mínima que distingue (20 cm), es decir $d = do/20$. La función de saturación empleada, que se observa en la capa de salida en la Figura **4-1**, se hace necesaria debido a que el exponente en (4-1) tiende a infinito al activarse la red correspondiente. P_c es obtenida mediante el grupo de iteraciones ilustradas en la Tabla **4-7**. Dado que se requiere que, a las distancias de entrenamiento, la red correspondiente sea la predominante, se busca que su salida (O_n) posea una relación exponencial alta que demarque el resultado de la sumatoria. Para este caso,

la relación inversa $1/(n-d)$ genera un valor infinito en la dupla red-distancia, a diferencia de las combinaciones previas validadas. El parámetro SAT, denota dicho valor infinito al cual se le adiciona la saturación a un valor de 0 a 10, eliminando los valores negativos. Para el caso de las relaciones $1/(n+1-d)$ y $1/(d+1-n)$, la condición en el denominador de cero (DIV 0) elimina la ecuación como opción. Finalmente, la salida total de la red, O_f, corresponde al argumento máximo de P_c.

Tabla 4-7: Iteración para el establecimiento de Pc.

do	d	n	d+1-n	n+1-d	n/d	d/n	1/(d+1-n)	1/(n+1-d)	n-d	1/(n-d)
20	1	1	1	1	1	1	1	1	0	SAT
20	1	2	0	2	2	0,5	DIV 0	0,5	1	1
20	1	3	-1	3	3	0,33	-1	0,33	2	0,5
30	1,5	1	1,5	0,5	0,66	1,5	0,66	2	-0,5	-2
30	1,5	2	0,5	1,5	1,3	0,75	2	-0,66	0,5	2
30	1,5	3	-0,5	2,5	2	0,5	-2	0,4	1,5	0,6
40	2	1	2	0	0,5	2	0,5	DIV 0	-1	-1
40	2	2	1	1	1	1	1	1	0	SAT
40	2	3	0	2	1,5	0,6	DIV 0	0,5	1	1
50	2,5	1	2,5	-0,5	0,4	2,5	0,4	-2	-1,5	-0,6
50	2,5	2	1,5	0,5	0,8	1,25	0,6	2	-0,5	-2
50	2,5	3	0,5	1,5	1,2	0,8	2	0,6	0,5	2

En la Figura **4-5** se puede apreciar el resultado de la clasificación de la arquitectura neuro-convolucional diseñada, basada en profundidad. Se aprecia que las imágenes empleadas son reconocidas satisfactoriamente cuando la cámara es acercada hacia los objetos de interés. Para este caso el error se reduce al $8{,}31\%$. Note que, para cada columna de la Figura **4-5** se tiene la relación de captura ilustrada en la Tabla **4-8**. Se observa que para los valores de distancia del entrenamiento la salida se satura, mientras que para valores intermedios es ponderada por las activaciones respectivas. Por ejemplo, para una distancia de 50 cm, la ponderación resultante es 4.74, resultando de las activaciones de las arquitecturas 2 y 3 que, sin llegar a saturarse, da por encima del valor de las otras clases.

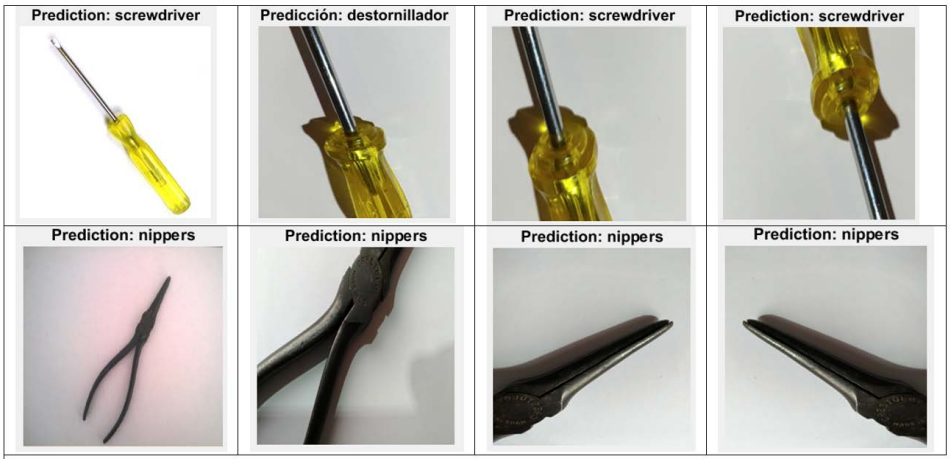


Figura 4-5: Respuesta de la red neuronal convolucional basada en profundidad.

Tabla 4-8: Resultados ponderación para bisturí en profundidad.

Profundidad	60 cm	30 cm	20 cm	10 cm
Arquitectura activada	3	2-3	1	1
Ponderación	10	5,66	10	10

4.4. Arquitectura híbrida difusa de aprendizaje en profundidad

Una alternativa a la solución planteada es cambiar la capa final de ponderación por una capa de inferencia difusa, como se ilustra en la Figura **4-6.** Obteniendo así una arquitectura híbrida en la que hay que diseñar el sistema de inferencia difusa que reemplace a (4-1).

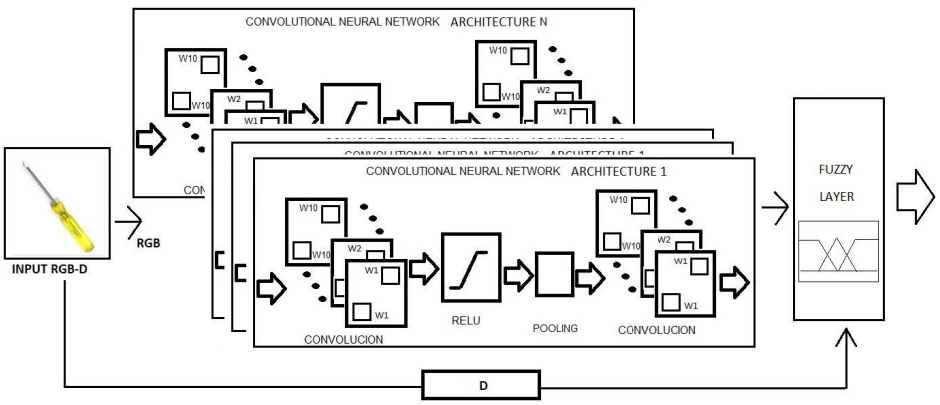


Figura 4-6: Arquitectura híbrida CNN-Difusa.

El primer aspecto a tener en cuenta es que se debe fusificar la salida de las redes paralelas, para su interpretación en el sistema de inferencia difusa. Cada una delas salidas de las redes, por categoría a reconocer, varía entre 0 y 1, ya que estas se componen de una función de activación en la etapa de clasificación que emplea una relación no lineal tipo tangencial, lo que orienta a usar funciones de pertenencia de tipo Gaussiano en los conjuntos difusos. A la salida de cada red (S_Ctg), se multiplica por 33.5 y se le ajusta un offset, a fin de cubrir un universo de discurso de 0 a 100, entre las cuatro categorías, como se ilustra en la Tabla **4-9**.

Tabla 4-9: Fusificación categorías.

CATEGORIA	Fusificación	Rango
Pinza	$33.5\,(1$ - $S\_Ctg1)$	0 - 33.5
Destornillador	$33.5\,(S\_Ctg2)$	0 - 67
Bisturí	$33.5\,(S\_Ctg3{+}1)$	33.5 - 100
Tijeras	$33.5\,(S\_Ctg4{+}2)$	67 - 100

La Figura **4-7** ilustra el sistema de inferencia difusa, donde se observan las entradas del sistema, relacionadas a cada una de las cuatro redes empleadas, y la entrada de distancia, las cuales determinarán finalmente la categoría de clasificación que se relacionan mediante la base de reglas que gobernará la respuesta del sistema.

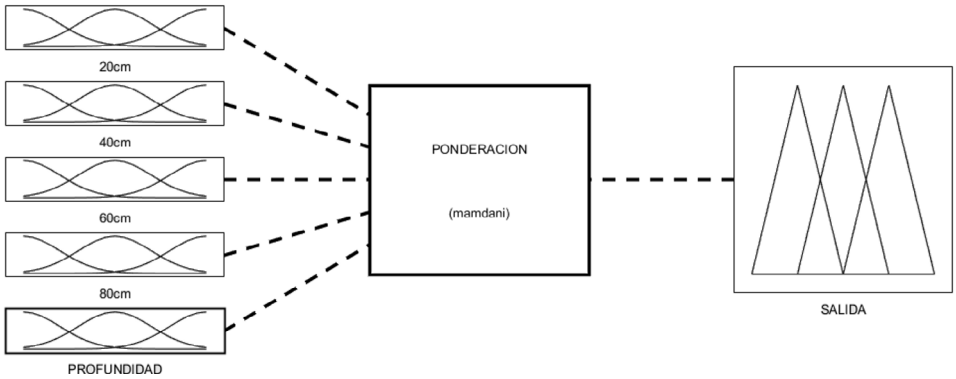


Figura 4-7: Sistema de inferencia Difusa para ponderación.

La base de reglas relaciona los diferentes conjuntos difusos de entrada, que corresponden a las salidas fusificadas de las redes convolucionales

entrenadas a cada distancia particular (20 cm, 40 cm, 60 cm y 80 cm). Cada red determina su categoría de salida entre pinzas, destornillador, bisturí y tijeras, las cuales entran a ser las etiquetas lingüísticas de cada conjunto, como se observa en la Figura **4-8**. Los conjuntos conforman el universo de discurso a operar, la figura a su vez permite apreciar gráficamente las relaciones de fusificación de la Tabla **4-9**.

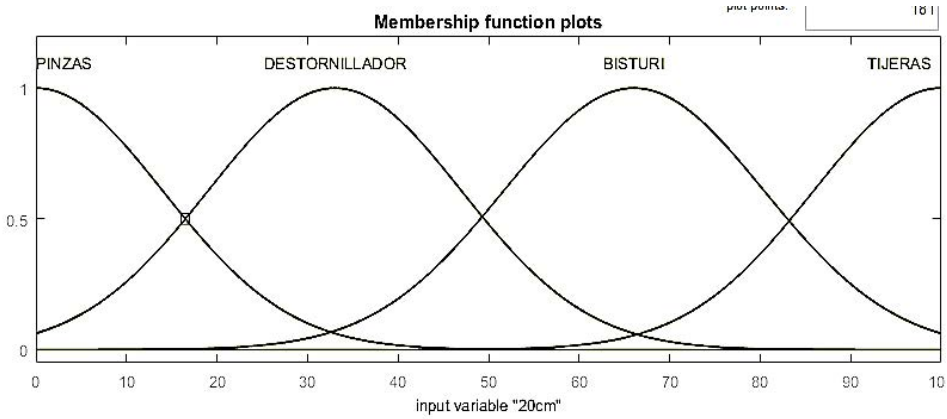


Figura 4-8: Funciones de pertenencia de entrada.

La información de distancia, obtenida del sensor RGB-D, es ingresada al conjunto difuso de profundidad, de forma tal que los universos de discurso no cubren más allá del rango de interés. Es decir, los 20 cm de rango mínimo y los 80 cm de entrenamiento máximo, como se ilustra en la Figura **4-9**.

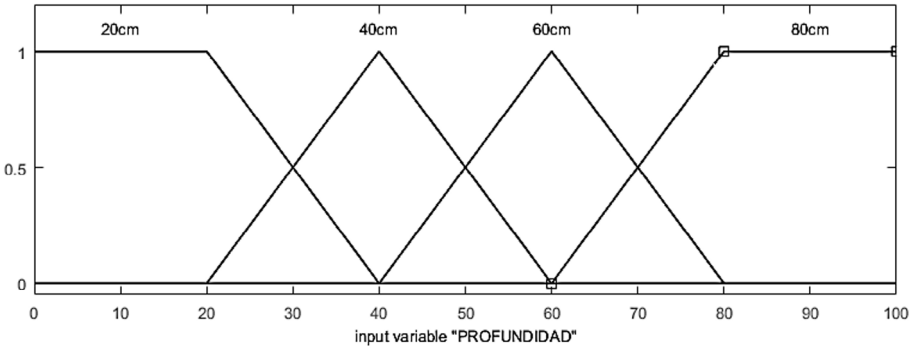


Figura 4-9: Funciones de pertenencia de profundidad.

El conjunto difuso de salida determinará finalmente la categoría de selección. Por lo tanto, se establece en el mismo rango y relación que los conjuntos difusos de entrada, dado que tiene la misma estructura respecto a la clasificación de las categorías. Pero en este caso, independiente de la profundidad, como se ilustra en la Figura **4-10**.

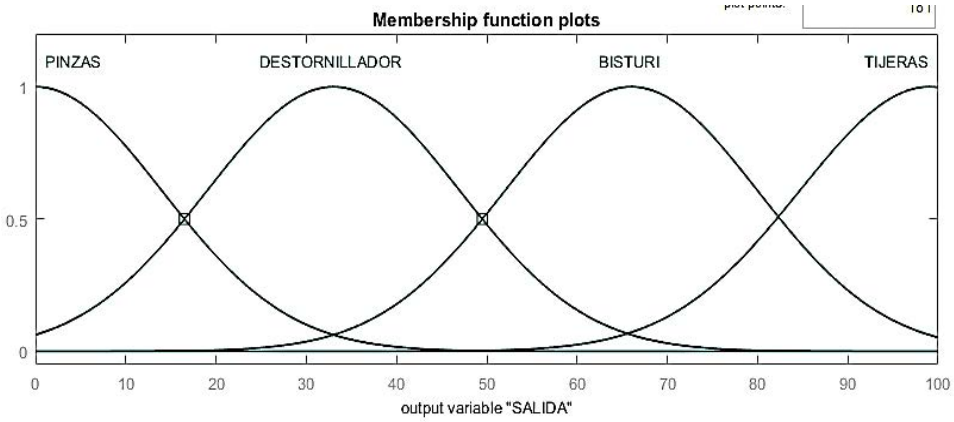


Figura 4-10: Funciones de pertenencia de salida.

En la Figura **4-11** se presenta una muestra de la base de reglas empleada. Es de resaltar que el problema se plantea con cuatro categorías, en cinco conjuntos difusos, lo que resulta en $4^5{=}1024$ combinaciones de reglas por determinar.

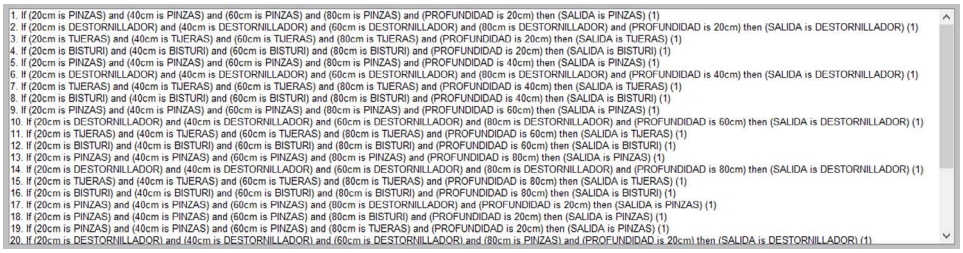


Figura 4-11: Base de reglas.

Para ilustrar el proceso de fusificación empleado, se expone el ejemplo de clasificación que se muestra en la Figura **4-12**, donde la entrada es la imagen de un destornillador a una distancia de 40 cm. La red 2, entrenada a 40 cm, evidencia un $94{,}99\%$ de acierto en la clasificación y las demás redes decrementan dicho acierto, como era de esperarse. En la Figura **4-12** se ilustran los resultados de las cuatro redes por cada categoría dada:

pinza (`label1`), destornillador (`label2`), bisturí (`label3`) y tijeras (`label4`). La fusificación de estos resultados se ilustra en la Tabla **4-10**, donde $\mu(x)$ corresponde al grado de pertenencia.

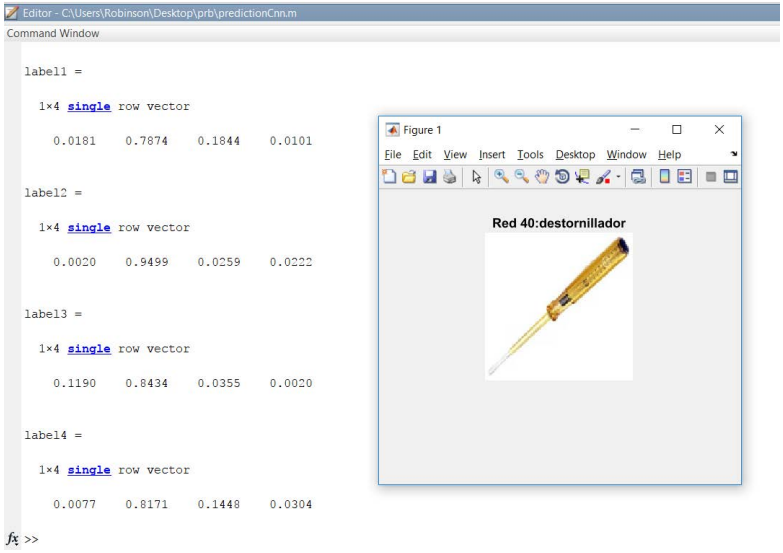


Figura 4-12: Respuesta del sistema sin ponderación.

Tabla 4-10: Resultados capa difusa.

RED	Categoría	Salida de la red (x)	Fusificación $(\mu(x))$
1- CNN 20 cm	Pinzas	0.0181	32.893
	Destornillador	0.7874	26.377
	Bisturí	0.1844	39.677
	Tijeras	0.0101	67.338
2- CNN 40 cm	Pinzas	0.002	33.43
	Destornillador	0.949	31.821
	Bisturí	0.0259	34.367
	Tijeras	0.022	67.743
3- CNN 60 cm	Pinzas	0.119	29.513
	Destornillador	0.8434	28.253
	Bisturí	0.0355	34.689
	Tijeras	0.002	67.06
4- CNN 80 cm	Pinzas	0.007	33.242
	Destornillador	0.817	27.372
	Bisturí	0.144	38.350
	Tijeras	0.0304	68.018

La Figura **4-13** permite observar gráficamente el resultado de la fusificación sobre el conjunto difuso para la red 2 (Arq 2 - 40 cm), evidenciando que corresponde a un destornillador. A su vez se pueden observar la fusificación de las otras tres redes.

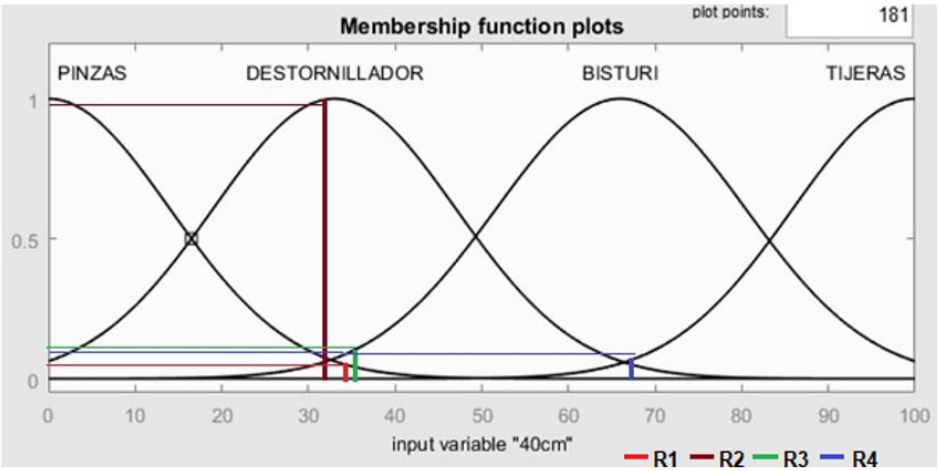


Figura 4-13: Salida difusa ponderada.

La salida del sistema de inferencia difusa se calcula empleando el algoritmo Mamdani, el cual implica los pasos descritos en el Algoritmo 3 [Babuska, 2001].

Algoritmo 3: Inferencia Mamdani

Begin

Paso 1: Evaluación del antecedente en cada regla.
$\beta_i = \underset{X}{max}[\mu A'(x) \wedge \mu A_i(x)],\ 1 \leq i \leq K.$

Paso 2: Obtener la conclusión en cada regla, para los conjuntos difusos de salida. $B'_i; \mu B'_i(y) = \beta_i \wedge \mu B_i(y),\ 1 \leq i \leq K$

Paso 3: Agregar la conclusión de cada regla. $B'_i : \mu B'(y) = \underset{1 \leq i \leq K}{max} \mu B'(y)$

end

En este caso, se tienen los conjuntos difusos de entrada, como antecedente difuso:

$$20cm = \{0{,}0181/32;\ \ 0{,}7874/26{,}37;\ \ 0{,}1844/39{,}67;\ \ 0{,}0101/67{,}33\},$$
$$40cm = \{0{,}002/33{,}43;\ 0{,}9499/31{,}82;\ \ 0{,}0259/34{,}36;\ \ 0{,}0222/67{,}74\},$$

$60cm = \{0{,}119/29{,}51;\ \ 0{,}8434/28{,}25;\ \ 0{,}0355/34{,}68;\ \ 0{,}002/67{,}06\},$
$80cm = \{0{,}0077/33{,}24;\ \ 0{,}8171/27{,}37;\ \ \ 0{,}1448/38{,}35;\ \ 0{,}0304/68{,}01\},$

que operaran con los conjuntos difusos relacionales:

$$\begin{aligned} pinzas &= \{1/0;\quad 0,\,03/35;\quad 0,\,001/67;\quad 0/100\},\\ destornillador &= \{0/0;\quad 1/35;\quad 0,\,01/67;\quad 0/100\},\\ bisturí &= \{0/0;\quad 0,\,001/35;\quad 1/67;\quad 0,\,01/100\},\\ tijeras &= \{0/0;\quad 0/35;\quad 0,\,01/67;\quad 1/100\}. \end{aligned}$$

Se tienen los conjuntos difusos de salida, como consecuente difuso:

$$\begin{aligned} pinzas &= \{1/0;\quad 0,\,03/35;\quad 0,\,001/67;\quad 0/100\},\\ destornillador &= \{0/0;\quad 1/35;\quad 0,\,01/67;\quad 0/100\},\\ bisturí &= \{0/0;\quad 0,\,001/35;\quad 1/67;\quad 0,\,01/100\},\\ tijeras &= \{0/0;\quad 0/35;\quad 0,\,01/67;\quad 1/100\}. \end{aligned}$$

A continuación, se ilustra el computo del algoritmo Mamdani, donde se requiere obtener cada β_n y B_n hasta β_{16} y B_{16} de las cuatro redes y operarlos posteriormente con el de profundidad ($\beta Depth$).

β_1= max({0,0181; 0,7874; 0,1844; 0,0101}^{1; 0,03; 0,001; 0})=0,03
β_2= max({0,0181; 0,7874; 0,1844; 0,0101}^{0; 1; 0,01; 0})=0,7874
β_3= max({0,0181; 0,7874; 0,1844; 0,0101}^{0; 0,001; 1; 0,01})=0,1844
β_4= max({0,0181; 0,7874; 0,1844; 0,0101}^{0; 0; 0,01; 1})=0,01
β_5= max({0,002; 0,9499; 0,0259; 0,0222}^{1; 0,03; 0,001; 0})=0,002
β_6= max({0,002; 0,9499; 0,0259; 0,0222}^{0; 1; 0,01; 0})=0,9499
β_7= max({0,002; 0,9499; 0,0259; 0,0222}^{0; 0,001; 1; 0,01})=0,0259
β_8= max({0,002; 0,9499; 0,0259; 0,0222}^{0; 0; 0,01; 1})=0,0222
β_9= max({0,119; 0,8434; 0,0355; 0,002}^{1; 0,03; 0,001; 0})=0,119
β_{10}= max({0,119; 0,8434; 0,0355; 0,002}^{0; 1; 0,01; 0})=0,8434
β_{11}= max({0,119; 0,8434; 0,0355; 0,002}^{0; 0,001; 1; 0,01})=0,0355
β_{12}= max({0,119; 0,8434; 0,0355; 0,002}^{0; 0; 0,01; 1})=0,0355
β_{13}= max({0,0077; 0,8171; 0,1448; 0,0304}^{1; 0,03; 0,001; 0})=0,03
β_{14}= max({0,0077; 0,8171; 0,1448; 0,0304}^{0; 1; 0,01; 0})=0,8171

β_{15}= max({0,0077; 0,8171; 0,1448; 0,0304}^{0; 0,001; 1; 0,01})=0,1448
β_{16}= max({0,0077; 0,8171; 0,1448; 0,0304}^{0; 0; 0,01; 1})=0,1448
B_1= 0,1844^{1; 0,03; 0,001; 0} ={0,1884; 0,03; 0,001; 0}
B_2= 0,7874^{0; 1; 0.01; 0} ={0; 0,7874;0,01;0}
B_3= 0,1844^{0; 0,001; 1; 0,01} ={0; 0,001; 0,1844; 0,01}
B_4= 0,01^{0; 0; 0,01; 1} ={0; 0; 0,01; 0,01}
B_5= 0,002^{1; 0,03; 0,001; 0} ={0,002; 0,002; 0,001; 0}
B_6= 0,9499^{0; 1; 0,01; 0} ={0; 0,9499;0,01;0}
B_7= 0,0259^{0; 0,001; 1; 0,01} ={0; 0,001; 0,0259; 0,01}
B_8= 0,0222^{0; 0; 0,01; 1} ={0; 0; 0,01; 0,0222}
B_9= 0,119^{1; 0,03; 0,001; 0} ={0,119; 0,03; 0,001; 0}
B_{10}= 0,8434^{0; 1; 0,01; 0} ={0; 0,8434;0,01;0}
B_{11}= 0,0355^{0; 0,001; 1; 0,01} ={0; 0,001; 0,0355; 0,01}
B_{12}= 0,0355^{0; 0; 0,01; 1} ={0; 0; 0,01; 0,0355}
B_{13}= 0,03^{1; 0,03; 0.001; 0} ={0,03; 0,03; 0,001; 0}
B_{14}= 0,8171^{0; 1; 0,01; 0} ={0; 0,8171;0,01;0}
B_{15}= 0,1448^{0; 0,001; 1; 0,01} ={0; 0,001; 0,1448; 0,01}
B_{16}= 0,1448^{0; 0; 0,01; 1} ={0; 0; 0,01; 0,1448}
B={0,1884; 0,9499;0,1844;0,1448}
$\beta Depth$= max({0; 1; 0; 0}^{0; 1; 0; 0}) =1
$BDepth$= 1^{0; 1; 0; 0} ={0;1;0; 0}
$Btotal$={0,1884; 1;0,1844;0,1448}

A este resultado se le aplica un método de fusificación. Por ejemplo, mediante centro de gravedad, como sigue

$$y' = \frac{\sum_{j=1}^{F} \mu_{B'}(y_j) \times y_j}{\sum_{j=1}^{F} \mu_{B'}(y_j)} = \frac{0{,}188 \times 1 + 1 \times 33 + 0{,}184 \times 66 + 0{,}144 \times 99}{0{,}188 + 1 + 0{,}184 + 0{,}144} = 39,3. \quad (4\text{-}2)$$

El valor y' corresponde al centro de gravedad del conjunto difuso de salida *Btotal* y para un valor de $39{,}3$ tiene un grado de pertenencia del $92{,}34\%$ en la categoría de destornillador (ver Figura **4-13**). De forma tal, que la diferencia entre la red difusa y el obtener el promedio aritmético de las salidas de la categoría destornillador entre cada red, que da un

$84{,}94\%$ de acierto en la clasificación, muestra un $7{,}39\%$ más acertada a la red difusa. Mientras que el resultado obtenido mediante (4-1), al ser una distancia de entrenamiento, satura la salida dando 100% en la clasificación.

La Tabla **4-11** relaciona los resultados obtenidos para ambos casos de ponderación propuestos, donde P_c corresponde a la respuesta de (4-1) y F_s a la respuesta del sistema difuso. Es evidente que, para los casos de distancia de entrenamiento, la salida P_c satura la respuesta, presentando mejor desempeño que la salida F_s. Sin embargo, para casos intermedios de distancia, las respuestas son más similares, siendo levemente mejor la salida difusa en algunos casos.

Tabla 4-11: Comparación capas de ponderación.

	Pinzas		Destornillador		Bisturí		Tijeras	
	P_c	F_s	P_c	F_s	P_c	F_s	P_c	F_s
20	100	91,56	100	93,97	100	90,21	100	94,01
30	87,01	84,61	92,5	89,33	91,11	89,96	92,83	91,34
40	100	92,34	100	89,62	100	91,68	100	94,83
50	91,3	91,87	90,8	91,65	92,12	92,43	93,15	90,68
60	100	92,71	100	91,06	100	90,67	100	92,35
80	100	94,79	100	90,66	100	89,32	100	89,47

Es de aclarar que, aunque una red híbrida convolucional difusa, como la diseñada, es una mezcla de técnicas de Machine Learning, que soluciona el problema del ambiente dinámico de reconocimiento planteado, la red por saturación mediante (4-1) es más simple y genérica, dado que el número de redes paralelas puede cambiar fácilmente, mientras que en el caso difuso se debe rediseñar el sistema de inferencia para ingresar un nuevo caso de profundidad.

4.5. Validación mediante DAG-CNN

Para validar el desempeño del sistema propuesto se implementa una red tipo DAG-CNN, que corresponde a un arquitectura paralela variante de las redes neuronales convolucionales (CNN). Pero que, a diferencia de la

arquitectura propuesta, no permite la inclusión de un canal de profundidad como el empleado y, a su vez, trabaja con una base de datos generalizada a todas las redes. Dentro del estado del arte, este tipo de red es el método de validación más cercano al propuesto.

Como se ha mencionado, en las redes neuronales convolucionales orientadas al reconocimiento de múltiples grupos de clases en imágenes, se necesita una mayor profundidad en la red para conservar una alta eficiencia en la clasificación. Para compensar esto, se diseñaron las estructuras de tipo Directed Acyclic Graph (DAG) Network [Thulasiraman y Swamy, 1992]. Este tipo de arquitectura permite tener una mayor cantidad de capas de aprendizaje trabajando en ramas paralelas, por lo que no la requiere hacer mas profunda, mejorando el tiempo de procesamiento e incrementando la cantidad de características que la red puede aprender, como se expone en [Ciregan *et al*., 2012].

Al tener la posibilidad de personalizar los parámetros de cada rama, estos se pueden determinar de forma que aprendan distintas características de los objetos a clasificar. Dicho comportamiento, es similar a la propuesta de red de arquitectura paralela expuesta, donde cada rama aprende las características a una profundidad determinada. Por la similitud de las arquitecturas se diseña una DAG-CNN para evaluar su comportamiento con la base de datos completa. Es decir, se incluye cada categoría en los diferentes niveles de profundidad que se establecieron.

Se implementó una primera DAG-CNN con la arquitectura de las redes mostradas en la Tabla **4-5**, como se aprecia en la Figura **4-14**. Esta primera arquitectura obtuvo una eficiencia en la clasificación del $81{,}3\%$. Para mejorar el desempeño de esta red, se realizan 10 iteraciones, variando los parámetros de cada rama, obteniendo un máximo de $86{,}64\%$ de eficiencia en la clasificación, donde la DAG-CNN obtenida se ilustra en la Figura **4-15**.

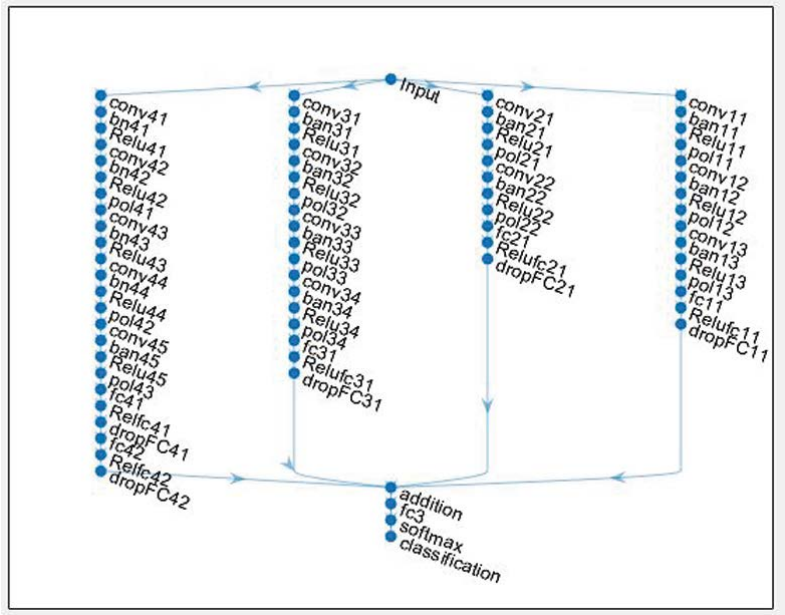


Figura 4-14: DAG-CNN de prueba inicial.

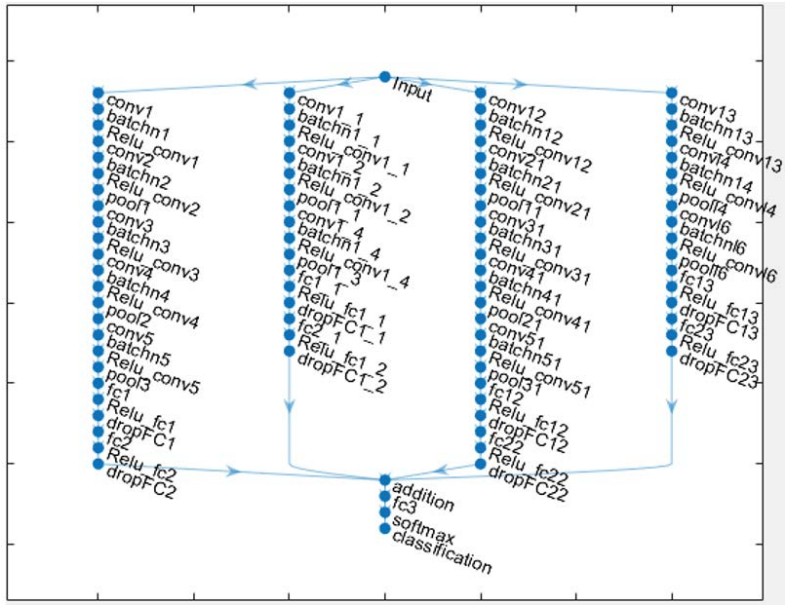


Figura 4-15: DAG-CNN de prueba final.

Aunque un desempeño cercano al 87% se puede considerar suficiente en la tarea de clasificación, más aún frente al $69{,}71\%$ alcanzado con la red neuronal convolucional de la Sección 3.4.2, que también incluye la base de datos completa. Se puede evidenciar que la red DAG-CNN adquiere mayor capacidad de discriminación frente a los problemas de variación de distancia y que una arquitectura paralela aporta un camino a solucionar dicho problema. Sin embargo, la arquitectura propuesta en este capítulo, con inclusión del canal de distancia, evidencia un mejor desempeño en un ambiente dinámico, reduciendo significativamente los falsos positivos.

La Figura **4-16** permite evidenciar las activaciones del mismo objeto a una distancia de 40 cm, con las redes cercanas (20 y 60 cm), donde se aprecia que a 40 cm se generan más y mejores activaciones que permiten un claro reconocimiento, el cual se degrada al acercar o alejar el objeto.

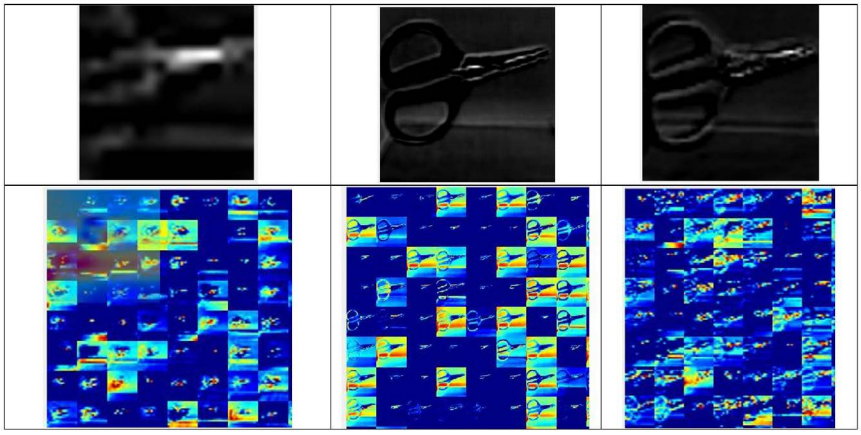


Figura 4-16: Activación 20, 40 y 60 cm.

Capítulo 5

Robótica Asistente Mediante Aprendizaje Profundo

Con el fin de entrenar un agente robótico como asistente, en una plataforma multi-herramientas, se deben considerar aspectos adicionales con relación a la identificación de la herramienta en un ambiente dinámico. Dentro de estos aspectos está la planeación de trayectorias que realizará el robot para ir de un punto en el espacio a otro. Dentro de este desplazamiento se puede presentar la necesidad de una posible evasión de obstáculos y, por ejemplo, al llegar a la herramienta se debe determinar cómo se realizará su agarre, acción que a su vez puede requerir de un algoritmo de control para compensación de fuerza, por las variaciones de peso presente entre herramientas y que afectan al ser tomadas por el efector final del brazo robótico.

A continuación se exponen algunos algoritmos que permiten dar una solución tentativa a tales aspectos.

5.1. Planeación de trayectoria

El desplazamiento de un robot está determinado por las relaciones geométricas de su estructura, las cuales determinan su cinemática básica,

y con un algoritmo, que le permita establecer por qué puntos en el espacio moverse. A continuación, se establece un posible esquema que permite a un brazo robótico asistencial, en función a su cinemática, desplazarse de un punto a otro, lo que se denomina *planeación de trayectoria*.

5.1.1. Análisis matemático

El análisis que permite conocer la posición del efector final del brazo robótico respecto a las coordenadas de la base (o del mundo), se conoce como análisis cinemático. De forma general, relacionar las coordenadas propias del robot respecto a cada uno de sus ángulos y longitudes, con la ubicación espacial (x,y,z) del efector final, se denomina cinemática directa. Esta se requiere para poder determinar, desde una posición inicial, la forma en que cada articulación del brazo se debe mover para lleva al efector a un punto deseado en el espacio.

El análisis cinemático se realiza empleando el método de Devanit-Hartenberg ($\mathrm{D\text{-}H}$), que relaciona el espacio de articulaciones de un brazo robótico con el espacio cartesiano [Weber, 2010], como una función $f.J(\theta_i) \rightarrow \mathbb{R}^3$, donde θ_i corresponde al ángulo de rotación de cada articulación. Se tienen n grados de libertad del brazo (GDL), que para el caso del brazo empleado es de 5 GDL, el quinto grado corresponde al efector final, que no cambia la distancia alcanzada, por lo cual no hace parte de análisis. Cada grado de libertad, genera un sistema coordenado ortogonal. De forma tal que, para relacionar el sistema ortogonal anterior (S_{i-1}) con el sistema actual (S_i), se utilizan transformaciones homogéneas que indican los rotaciones y traslaciones que se deben efectuar sobre el sistema S_{i-1} para llegar a la posición y orientación del sistema S_i . El método D-H propone que a partir de cuatro transformaciones básicas, asociadas a traslaciones y rotaciones sobre y alrededor de los ejes x y z, se puede relacionar la posición y orientación de dos sistemas consecutivos, previa selección adecuada de los sistemas coordenados ortogonales [Barrientos *et al*., 2007], como se muestra a continuación

$$^{\mathrm{i}-1}\mathbf{A}_i = \mathrm{Rot}_{z,\theta_i}\,\mathrm{Tras}_{z,d_i}\,\mathrm{Tras}_{x,a_i}\,\mathrm{Rot}_{z,\alpha_i},$$

donde $^{\mathrm{i}-1}\mathbf{A}_i$ representa la posición y orientación del sistema S_i en coordenadas del sistema S_{i-1}, θ_i y αi representan rotaciones al rededor de los ejes x y z (para que el sistema S_{i-1} quede con la misma orientación que S_i), respectivamente; mientras que d_i y a_i representan traslaciones sobre los ejes x y z (para que el sistema S_{i-1} quede en la posición de S_i), respectivamente. Note que la matriz de transformación genérica entre dos marcos de referencia se puede escribir como [Abdel-Malek y Othman, 1999]

$$^{i-1}\mathbf{A}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\,\cos\alpha_i & \sin\theta_i\,\sin\alpha_i & a_i\,\cos\theta_i \\ \sin\theta_i & \cos\theta_i\,\cos\alpha_i & -\cos\theta_i\,\sin\alpha_i & a_i\,\sin\theta_i \\ 0 & -\sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (5\text{-}1)$$

Por lo que, para relacionar cada uno de los sistemas de referencia con la base, se utiliza [Abdel-Malek y Othman, 1999]

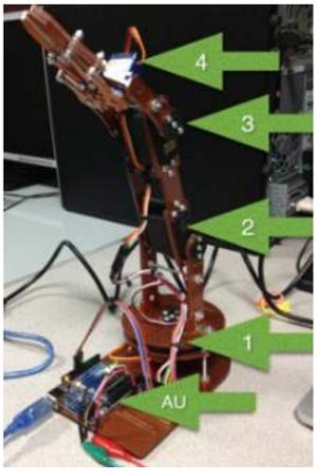
$$^{0}\mathbf{A}_n = \prod_{i=1}^{n} {}^{i-1}\mathbf{A}_i. \qquad (5\text{-}2)$$

Dado que $^{0}\mathbf{A}_n$ es una matriz de transformación homogénea, tiene la estructura

$$^{0}\mathbf{A}_n = \left[\begin{array}{ccc|c} & \mathbf{R}(\boldsymbol{\theta}) & & \begin{matrix} x(\boldsymbol{\theta}) \\ y(\boldsymbol{\theta}) \\ z(\boldsymbol{\theta}) \end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right], \qquad (5\text{-}3)$$

donde $\mathbf{R}(\boldsymbol{\theta})$ es una matriz de rotación, función de los ángulos de la articulaciones, $\boldsymbol{\theta} = [\theta_1\ \theta_2\ \theta_3\ \theta_4\ \theta_5]$, que indica la orientación del efector en el sistema coordenado de la base y ($x(\boldsymbol{\theta})$, $y(\boldsymbol{\theta})$, $z(\boldsymbol{\theta})$), también función de los ángulos de las articulaciones, representan la posición en $\mathbb{R}^3$ del efector.

La Figura **5-1** ilustra el brazo empleado y los parámetros de D-H asociados, los cuales se obtienen siguiendo el procedimiento en [Barrientos *et al*., 2007], y se condensan en la Tabla **5-1.**



Symbol	Small Manipulator
AU	Arduino One
1	Servomotor 1
2	Servomotor 2
3	Servomotor 3
4	Servomotor 4

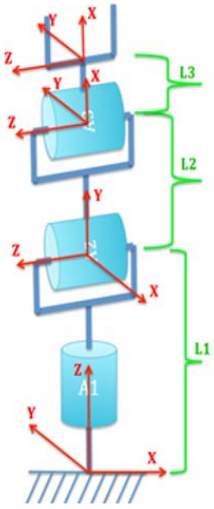


Figura 5-1: Robot utilizado. Izquierda: foto. Centro: notación. Derecha: esquemático del robot con la definición de sistemas coordenados por articulación y distancias entre ellos.

Tabla 5-1: Parámetros D-H para el robot de la Figura 5-1.

Parámetros D-H	θ	d	a	α
1	θ_1	l_1	0	-$\pi/2$
2	θ_2	0	l_2	0
3	θ_3	0	l_3	0
4	θ_4	0	0	$\pi/2$
5	0	0	0	θ_5

Del análisis cinemático del brazo se puede validar el desplazamiento del mismo en una trayectoria determinada en $\mathbb{R}^3$, como el brazo robótico corresponde a una estructura antropomórfica rotacional, sus movimientos están delimitados por las esferas de radio variable, según la longitud de cada articulación. Dado esto, se emplea la siguiente función de costo que se ajusta a las posibles trayectorias del brazo, como

$$J(\boldsymbol{\theta}) = (x(\boldsymbol{\theta}) - x_f)^2 + (y(\boldsymbol{\theta}) - y_f)^2 + (z(\boldsymbol{\theta}) - z_f)^2, \qquad (5\text{-}4)$$

donde (x_f, y_f, z_f) representan las coordenadas del punto deseado, a las que se desea que el manipulador se dirija, las cuales son obtenidas a partir a partir del resultado de la clasificación obtenida por la arquitectura de red CNN, que retorna la posición de la herramienta a agarrar en el espacio de trabajo.

Dado que el brazo utilizado existe físicamente, los ángulos θ_i tienen restricciones físicas, que se muestran en la Tabla **5-2**. Note que el quinto grado corresponde al efector que se encarga solamente del agarre. De igual forma, se tiene la restricción adicional del plano paralelo al eje de la base del robot $y \leq k$, que representa a la persona u obstáculo dinámico que ingresa al espacio de trabajo del robot. Por seguridad, se establece que el brazo robótico no podrá llegar hasta el plano sino hasta una distancia máxima de dicho plano, para el caso se toman 10 cm. Por lo que la ecuación del plano asignada como restricción es $y \leq p - 10$, donde p representa la distancia del plano al eje de la base del robot.

Tabla 5-2: Restricciones angulares.

Angulo	jmin	jmax
θ_1	-$3\pi/4$	$3\pi/4$
θ_2	-$2\pi/5$	$2\pi/5$
θ_3	-$2\pi/5$	$2\pi/5$
θ_4	-$2\pi/5$	$2\pi/5$
θ_5	-$\pi/4$	$\pi/4$

La Figura **5-2** ilustra la simulación del ambiente de trabajo, donde se aprecia el brazo robótico en la posición inicial y desde la punta del efector resaltada en negro se presenta el desplazamiento rotacional y traslacional de este, hasta aproximarse al plano representativo del obstáculo y alejarse nuevamente (véase la trayectoria del brazo en rojo y el punto final en azul).

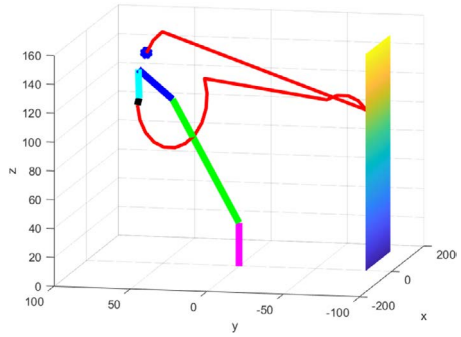


Figura 5-2: Simulación restricción de movimiento robótico.

5.1.2. Algoritmos de planeación de trayectorias

Determinadas las condiciones de operación del brazo, se establece la forma general de un problema de optimización, que permite encontrar los ángulos que cada articulación debe desplazarse para alcanzar la posición deseada, o un punto cercano a esta,

$$\begin{aligned} \min \quad & J(\boldsymbol{\theta}), \\ \text{s.a.} \quad & \boldsymbol{\theta}_{\text{mín}} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_{\text{máx}}, \\ & y(\boldsymbol{\theta}) \leq p - 10. \end{aligned} \tag{5-5}$$

Dado que el tipo de problema a resolver es de naturaleza no lineal, se emplea como algoritmo de optimización el método de penalización exterior, resuelto por sus variantes de valor absoluto y penalización cuadrática. Este método requiere un paso intermedio de búsqueda lineal, que para el caso se realiza mediante el método de gradiente conjugado, el cual a su vez requiere determinar el paso o nivel de cambio en la dirección de descenso, que conducirá al objetivo. Para este cálculo se emplea el método de búsqueda lineal Backtracking (Armijo's rule [Boyd y Vandenberghe, 2004]).

Con el fin de resolver el problema de optimización planteado en (5-5), es necesario replantearlo, de forma que se incluya en la función de coste las restricciones, con el fin de manejar un problema de optimización sin restricciones

$$\min \qquad J_r(\boldsymbol{\theta}, r_t) = J(\boldsymbol{\theta}) + r_t \left[\sum_{j=1}^{m} (g_j(\theta_i))^{\beta} \right], \tag{5-6}$$

donde J_r es la función de coste modificada que incluye la función de coste inicial, considerada en (5-5), r_t es la penalización de cada restricción de (5-5), j es el índice que barre las restricciones de (5-5) que son seis (cinco asociadas a los límites físicos de las articulaciones y una al plano que representa el obstáculo y la distancia de seguridad), las funciones $g_j(\cdot)$ representan las restricciones anteriores, que se consideran elevadas a la potencia β.

El algoritmo de penalización exterior implementado corresponde al Algoritmo 4.

Algoritmo 4: Algoritmo de penalización exterior.

Begin

Paso 1: Inicialización: inicialice con un punto inicial $\boldsymbol{\theta}_0$, posición de reposo del brazo. Establezca el parámetro de penalización inicial $r_1 > 0$, en este caso se utilizó $r_1 = 1$. También, se debe establecer la tolerancia $\varepsilon > 0$, para este caso dado que el posicionamiento del brazo no conlleva punto decimal que demarca la precisión o exactitud del punto final, se establece $\varepsilon > 1$.

Paso 2: Subproblema: Resuelva por un método descendente el problema de minimización, dado por (5-6), cuya solución son los valores angulares para las articulaciones del robot, dadas por $\boldsymbol{\theta}$. Se emplea el método de gradiente conjugado Fletcher-Reeves, donde el valor de $\boldsymbol{\theta}$ se actualiza en cada iteración a partir de

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \lambda_t \, \mathbf{d}_t,$$

donde la dirección de descenso se obtiene como

$$\mathbf{d}_t = -\nabla J_r(\boldsymbol{\theta}_t) + \beta_t * \mathbf{d}_{t-1},$$

donde $\nabla J_r(\boldsymbol{\theta}_t)$ representa el gradiente de la función de coste, y

$$\beta_t = \frac{\|\nabla J_r(\boldsymbol{\theta}_t)\|^2}{\|\nabla J_r(\boldsymbol{\theta}_{t-1})\|^2}.$$

El valor de paso para cada iteración se calcula mediante el método Backtracking de búsqueda lineal (Armijo's rule), que considera dos constantes $\alpha \in (0, 0.5)$ y $\beta_a \in (0, 1)$. Para cada iteración, el paso se inicializa como $\lambda = 1$ y debe ser actualizado como $\lambda = \beta_a \, \lambda$, siempre que se satisfaga que

$$J_r(\boldsymbol{\theta}_t + \lambda \, \mathbf{d}_t) > J_r(\boldsymbol{\theta}_t + \alpha \, \lambda \, \nabla J_r(\boldsymbol{\theta}_t)^T \, \mathbf{d}_t, \tag{5-7}$$

una vez la desigualdad planteada en (5-7) deja de satisfacerse, se obtiene el valor de $\lambda_t = \lambda$.

Paso 3: Criterio de parada, si $|J_r(\boldsymbol{\theta}_t - J_r(\boldsymbol{\theta}_{t-1}| < \epsilon$ en la función de penalización, el algoritmo finaliza, de otra forma vaya al paso 4.

Paso 4: Incremento de penalización, actualice el valor de penalización $r_{t+1} = \eta_t \, r_t$, con

$$\eta_t = r \, \left(\left| \frac{y(\boldsymbol{\theta}) - 10}{do} \right| \right)^b, \tag{5-8}$$

donde $y(\boldsymbol{\theta})$ corresponde al plano $y = k$, $b = 1$ para el método de valor absoluto y $b = 2$ para el cuadrático y do corresponde a la distancia desde la base del robot al operario.

La Figura **5-3** ilustra el diagrama de flujo del proceso completo del establecimiento de la trayectoria.

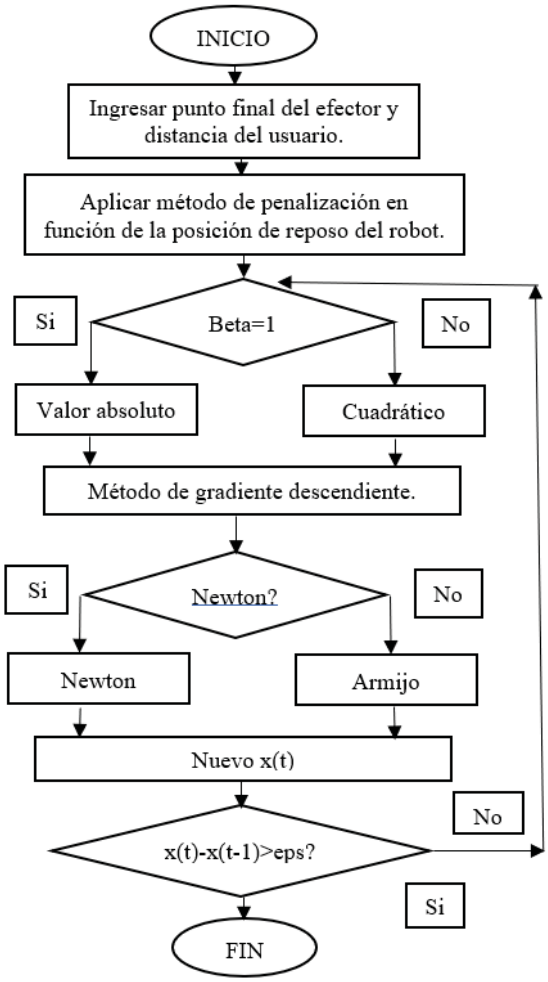


Figura 5-3: Diagrama de flujo algoritmo de optimización.

5.1.3. Análisis de Resultados

Las pruebas realizadas respecto a los diferentes métodos planteados permiten establecer criterios de desempeño de cada una, mediante su simulación en Matlab®. Para poder realizar un análisis comparativo, se establece un punto inicial para la posición del robot, este punto corresponde al

punto de partida respecto a cada trayectoria a realizar y el cual es tomado como posición de reposo con coordenadas $\boldsymbol{x}(0) = (0,\ 12,\ 76)$. Como punto objetivo, se escoge una posición aleatoria dentro del espacio de trabajo del robot. Para este caso, se toma $\boldsymbol{x}(t_f) = (50,\ 22,\ 89)$, y se establece de forma inicial una distancia del operario, o posible obstáculo, de 100 cm. En este caso, la restricción fuerte que corresponde a la cercanía del operario al robot no se activa, por lo que se espera que el resultado final corresponda al objetivo o punto final ($\boldsymbol{x}(t_f)$). A continuación, se muestran los resultados para cada método implementado.

Método valor absoluto, la Tabla **5-3** expone los resultados realizando la búsqueda descendente mediante el método de gradiente y calculando el paso por el algoritmo de Armijo con $\alpha = 0{,}5$ y $\beta_a = 0{,}1$. Se llegó a los siguientes ángulos por articulación: $\theta_1 = 0{,}12\pi$, $\theta_2 = 1{,}2\pi$, $\theta_3 = 0{,}69\pi$, $\theta_4 = -0{,}8\pi$, $\theta_5 = 0{,}5\pi$, con un valor de penalización final $r = 78125$.

Tabla 5-3: Resultados método valor absoluto $\alpha = 0{,}5$ y $\beta_a = 0{,}1$.

Iteración	X	Y	Z
1	0.2513	11.7877	76.314
2	20.1058	15.8626	81.388
3	32.0905	18.3076	84.433
4	39.2543	19.7745	86.260
5	43.5526	20.6547	87.356
6	46.1315	21.1828	88.0136
7	47.6789	21.499	88.408
8	48.607	21.689	88.645
9	49.164	21.8039	88.787

Método valor absoluto, la Tabla **5-4** expone los resultados realizando la búsqueda descendente mediante el método de gradiente y calculando el paso por el algoritmo de Armijo con $\alpha = 0{,}3$ y $\beta_a = 0{,}8$. Se llegó a los siguientes ángulos por articulación: $\theta_1 = 0{,}12\pi$, $\theta_2 = 1{,}2\pi$, $\theta_3 = 0{,}69\pi$, $\theta_4 = -0{,}8\pi$, $\theta_5 = 0{,}5\pi$, con un valor de penalización final $r = 125$.

Tabla 5-4: Resultados método valor absoluto $\alpha = 0{,}3$ y $\beta_a = 0{,}8$.

Iteración	X	Y	Z
1	0.2513	11.787	76.314
2	65.458	25.107	92.941
3	45.197	20.969	87.775
4	51.492	22.255	89.380
5	49.536	21.855	88.881

Método cuadrático, la Tabla **5-5** expone los resultados realizando la búsqueda descendente mediante el método de gradiente y calculando el paso por el algoritmo de Armijo con $\alpha = 0{,}5$ y $\beta a = 0{,}1$. Se llegó a los siguientes ángulos por articulación: θ_1=0,128π, θ_2=1,2π, θ_3=0,69π, θ_4=−0,8π, θ_5=0,51π**,** con un valor de penalización final $r = 390625$**.**

Tabla 5-5: Resultados método cuadrático $\alpha = 0{,}5$ y $\beta a = 0{,}1$.

Iteración	X	Y	Z
1	0.2513	11.787	76.314
2	20.105	15.891	81.388
3	32.090	18.277	84.433
4	39.254	19.730	86.260
5	43.552	20.598	87.356
6	46.131	21.118	88.013
7	47.678	21.428	88.408
8	48.607	21.614	88.645
9	49.164	21.725	88.786

Método cuadrático, la Tabla **5-6** expone los resultados realizando la búsqueda descendente mediante el método de gradiente y calculando el paso por el algoritmo de Armijo con $\alpha = 0{,}3$ y $\beta_a = 0{,}8$**.** Se llegó a los siguientes ángulos por articulación: $\theta_1 = 0$, 129π, $\theta_2 = 1$, 25π, $\theta_3 = 0$, 69π, $\theta_4 = -0$, 82π, $\theta_5 = 0$, 51π**,** con un valor de penalización final $r = 125$.

Tabla 5-6: Resultados método cuadrático $\alpha = 0{,}3$ y $\beta_a = 0{,}8$.

Iteración	X	Y	Z
1	0.2513	11.787	76.314
2	65.457	24.846	92.941
3	45.196	20.458	87.775
4	51.492	21.942	89.380
5	49.536	21.442	88.881

Por medio de los resultados expuestos, se puede observar la incidencia de los parámetros de paso del algoritmo de Armijo en el establecimiento de la trayectoria, donde es preferible iniciar con un valor bajo de α, e incrementarlo más rápidamente por medio de un β_a mayor. Se aprecia que para cada caso, tanto de penalización de valor absoluto como cuadrática, empleando el cálculo de paso por medio del algoritmo de Armijo, se requieren más iteraciones, implicando mayor costo computacional, siendo más larga la trayectoria y por consiguiente empleando mayor tiempo en lograr alcanzar el punto deseado. Como es de esperarse las variaciones en los ángulos finales son mínimas. Al activar la restricción de distancia del operario ubicándolo a 10 cm, se obtienen los resultados presentados a continuación.

Método cuadrático, la Tabla **5-7** expone los resultados realizando la búsqueda descendente mediante el método de gradiente y calculando el paso por el algoritmo de Armijo con α=0,5 y β_a=0,1. Donde θ_1=0,12π, θ_2=1,2π, θ_3=0,69π, θ_4=−0,8π, θ_5=0,5π con un valor de penalización final r = 78125.

Tabla 5-7: Resultados método cuadrático α=0,5 y β_a=0,1.

Iteración	X	Y	Z
1	0.251	11.787	76.314
2	20.105	13.515	81.388
3	32.090	14.206	84.433
4	39.254	14.206	84.433
5	43.552	14.593	87.356
6	46.131	14.637	88.013
7	47.678	14.654	88.408
8	48.607	14.662	88.645
9	49.164	14.664	88.787
10	49.498	14.665	88.872
11	49.699	14.665	88.923
12	49.819	14.666	88.95

Método cuadrático, la Tabla **5-8** expone los resultados realizando la búsqueda descendente mediante el método de gradiente y calculando el paso por el algoritmo de Armijo con α=0, 3 y β_a=0,8. Donde para cada

caso la variación de los ángulos no es significativa, obteniendo: θ_1=0,09π, θ_2=1,18π, θ_3=0,67π, θ_4=-0,83π, θ_5=0,5π.

Tabla 5-8: Resultados método cuadrático α=0,3 y β_a=0,8.

Iteración	X	Y	Z
1	0.2513	11.787	76.314
2	65.457	17.448	92.941
3	45.196	11.979	87.775
4	50.233	16.205	89.059
5	50.037	14.269	89.009
6	50.006	14.769	89.001
7	50.001	14.640	89.002
8	50.002	14.673	89.000
9	50.000	14.665	89.000
10	50.000	14.667	89.000
11	50.000	14.666	89.000

Se aprecia que en este caso la coordenada en $x_2 = y$ no se logra alcanzar, tal y como se esperaba. De igual forma, ambos métodos tardan más en converger. Nuevamente, resultando mejor el método de cuadrático con un valor α inicial pequeño.

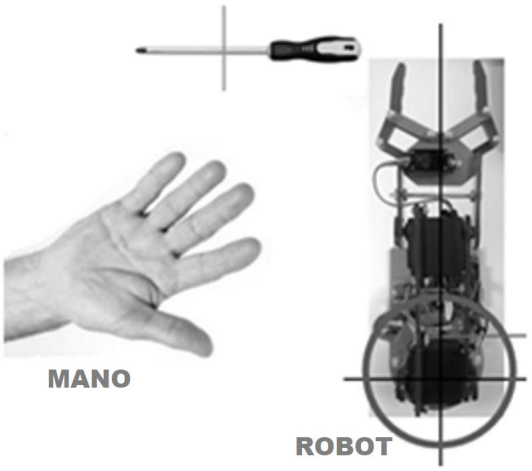
De forma general, los resultados obtenidos permiten establecer que los métodos de optimización aplicados logran alcanzar el punto objetivo, en el espacio de trabajo del robot, cuando este no es obstruido por alguna circunstancia. De forma tal que, al emplear el método de gradiente se asegura una trayectoria mínima, dado el seguimiento de esta directamente en la dirección del punto objetivo y, donde a su vez, el método de penalización cuadrática ofrece el mejor desempeño para el caso general.

5.2. Evasión de colisiones para asistencia robótica

Un método alternativo que permite realizar el desplazamiento de robot asistencial, teniendo en cuenta posibles colisiones, se plantea mediante la red neuronal convolucional desarrollada en la Sección 3.4.2, la cual opera en función a la distancia de reconocimiento del objeto. Para este

caso, se emplea el modelo cinemático inverso, debido a que el algoritmo de desplazamiento cambia en función de puntos en el espacio y no como en el caso anterior que el método de optimización entregaba directamente los ángulos deseados. Se emplea una cámara que visualiza el área de interacción, en la cual se identifica la herramienta y la mano del usuario, mediante el entrenamiento de redes neuronales convolucionales independientes. Para el caso considerado, se mantiene la red paralela, desarrollada para las herramientas, y se diseña la de la dirección la mano. En la Figura **5-4**, se puede apreciar un esquema del entorno de la aplicación, la herramienta, el brazo robótico y la mano del usuario, todo en una misma área.

Figura 5-4: Entorno del algoritmo anticolisión.



La Figura **5-5** ilustra un ejemplo de la base de datos empleada para la clasificación de la mano. Las redes empleadas solo tienen dos clases: mano o no mano. Para la clase mano, se emplean 100 imágenes de esta, tanto palma arriba como palma abajo, por cada uno de los cuatro niveles de profundidad. Para la clase no mano, se emplean 1000 imágenes que implican las herramientas y el fondo de la aplicación. Se obtienen las cuatro redes que se ilustran en la Tabla **5-9**.

Figura 5-5: Base de datos en profundidad para la mano.

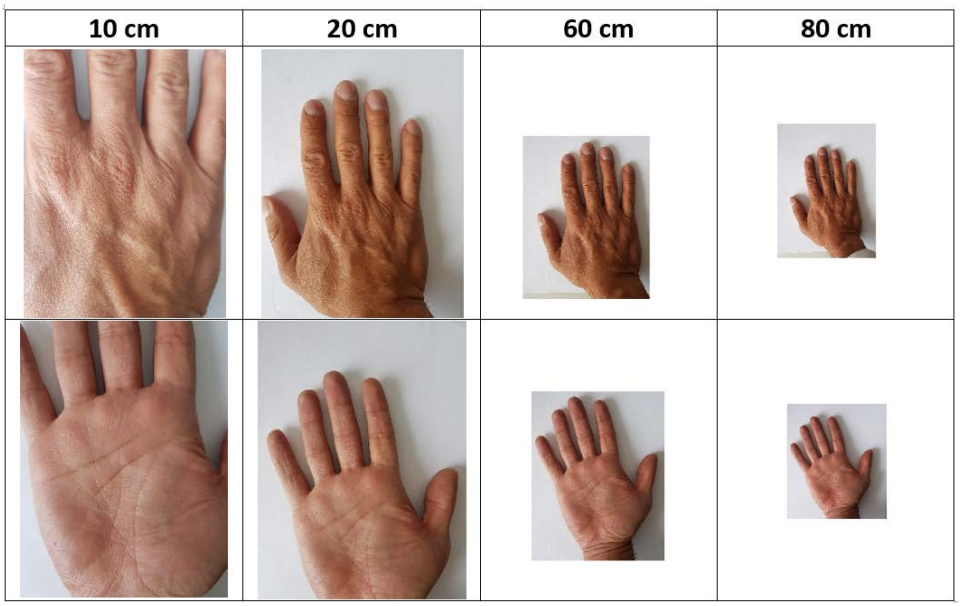


Tabla 5-9: Arquitecturas de red para la mano.

Capa	RED 20			RED 40			RED 60			RED 80		
	F	S	Nf	F	S	Nf	F	S	Nf	F	S	Nf
Convolución	8	1	20	4	1	20	4	1	30	4	1	40
Convolución	-	-	-	-	-	-	4	2	30	4	2	40
MaxPooling	2	2	-	2	2	-	2	2	-	2	2	-
Convolución	5	1	50	5	1	40	5	1	50	5	1	80
MaxPooling	2	2	-	2	2	-	2	2	-	2	2	-
Convolución	-	-	-	4	1	180	4	1	200	4	1	200
MaxPooling	-	-	-	10	2	-	3	2	-	3	2	-
Fully-connected	1		2	1		2	1		2	1		2
Softmax	2			2			2			2		

A continuación, se exponen las características del brazo robótico asistencial de prueba empleado, el cual se encarga de recoger la herramienta y entregarla en la mano del usuario, evitando colisionar con este. Los movimientos de desplazamiento se basan en la ubicación espacial de la mano, ubicación obtenida mediante el reconocimiento por redes neuronales convolucionales en función al centro del recuadro de detección y de la ubica-

ción de la herramienta por la misma técnica. De forma tal que, se establece un vector de orientación del movimiento de la mano, que pueda obstruir una trayectoria directa de desplazamiento del robot hacia la herramienta. En función a este vector, se determina el desplazamiento del brazo. Para lo cual, se requiere implementar la cinemática inversa del mismo. En la Figura **5-6**, se puede observar el modelo geométrico que permite inferir las ecuaciones mediante las cuales se logra establecer los movimientos angulares del robot [Useche *et. al*, 2018].

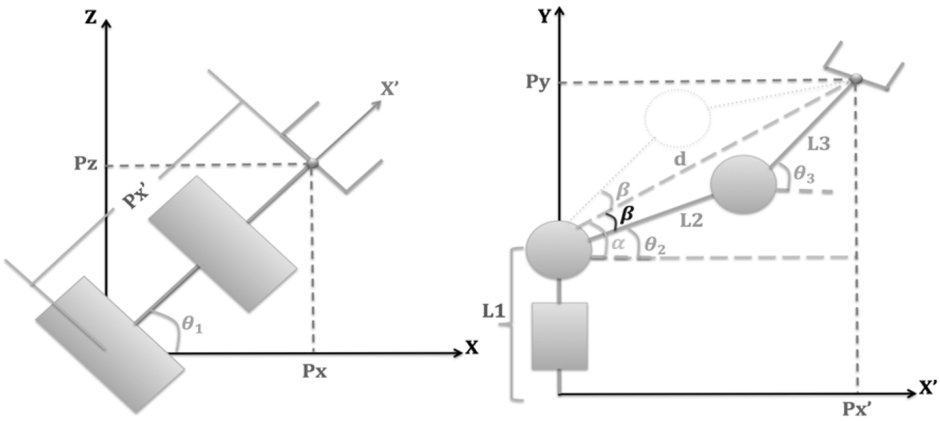


Figura 5-6: Cinemática inversa. Izquierda: Vista superior. Derecha: Vista lateral.

Con la vista superior de Figura **5-6**, se obtiene la componente en x' del punto P del efector final, como sigue

$$P_{x'} = \sqrt{P_z^2 + P_x^2}. \tag{5-9}$$

Por lo que, el ángulo de la articulación 1 se puede expresar como

$$\theta_1 = \tan^{-1}\left(\frac{P_z}{P_x}\right). \tag{5-10}$$

Mediante la vista frontal de Figura **5-6**, se obtiene la longitud alcanzada por el efector desde la segunda articulación, d, como

$$d = \sqrt{(P_y - L_1)^2 + P_x'^2}. \tag{5-11}$$

Conociendo d y las longitudes de los enlaces 2 y 3, L_2 y L_3, y aplicando el Teorema del coseno al triángulo con dichos vértices, tenemos

$$\cos(\pi - \theta_3) = \frac{d^2 - L_2^2 - L_3^2}{2L_2L_3}. \tag{5-12}$$

Sabiendo que

$$\sin(\pi - \theta_3) = \pm\sqrt{1 - \cos^2(\pi - \theta_3)}, \tag{5-13}$$

podemos encontrar el ángulo de la tercera articulación como

$$\theta_3 = \pi - \tan^{-1}\left(\frac{\pm\sqrt{4L_2^2L_3^2 - \left(d^2 - L_2^2 - L_3^2\right)^2}}{d^2 - L_2^2 - L_3^2}\right), \tag{5-14}$$

donde el signo positivo o negativo se seleccionará dependiendo dónde se desea la configuración codo abajo (θ_3 positivo) o codo arriba (θ_3 negativo).

De la vista frontal de la Figura **5-6**, también podemos obtener el ángulo

$$\alpha = \tan^{-1}\frac{P_y - L_1}{P_{x'}}. \tag{5-15}$$

Aplicando nuevamente el Teorema del coseno sobre el triángulo con lados d, L_2 y L_3, pero ahora respecto a β, tenemos

$$\beta = \cos^{-1}\left(\frac{L_2^2 + d^2 - 2L_3^2}{2L_2d}\right). \tag{5-16}$$

Por lo que

$$\theta_2 = \begin{cases} \alpha - |\beta|, & \text{codo arriba} \\ \alpha + |\beta|, & \text{codo abajo.} \end{cases} \tag{5-17}$$

Como se mencionó, se requiere implementar una nueva arquitectura neuronal del tipo convolucional, orientada a la detección de la mano del usuario. Para el caso, se presenta el entrenamiento para la detección de la mano abierta del usuario. La Tabla **5-10** resume la arquitectura final empleada para la red neuronal convolucional.

Tabla 5-10: Arquitectura CNN para detección de la mano.

CAPA	NUCLEO	
Input	$64 \times 64 \times 3$	
Convolution/RELU	4	20
Convolution/RELU	4	20
MaxPooling	2	
Convolution/RELU	5	50
Convolution/RELU	5	50
MaxPooling	2	
Convolution/RELU	4	200
MaxPooling	3	

La Figura **5-7** muestra el resultado de reconocimiento de la herramienta y de la mano, en función de las cuales se localiza su centroide para ser empleado como punto de referencia espacial en un entorno virtual de simulación mediante Matlab®.

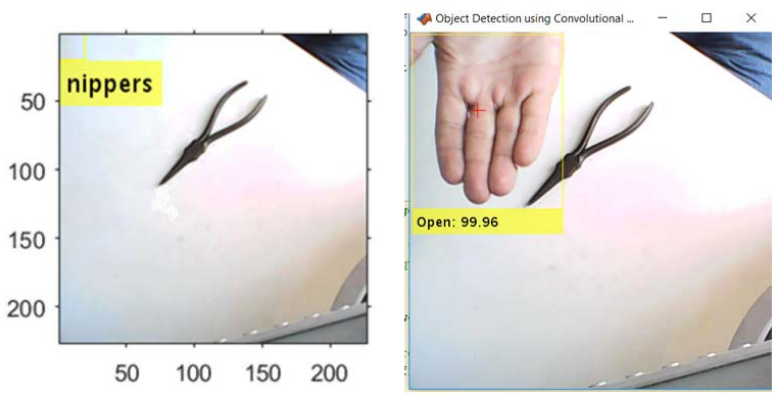


Figura 5-7: Resultado de la clasificación.

Para generar el desplazamiento del brazo robótico en función al punto objetivo (que es la herramienta) y buscando no colisionar con la mano de un usuario dentro del área de trabajo, se establecen como puntos característicos dos centros de referencia: el correspondiente a la herramienta, el cual es estático, y el correspondiente a la mano del usuario, el cual es dinámico. De esta forma, se implementa un vector direccional del desplazamiento de la mano en función de la herramienta (ver Figura **5-8**). Dado que el brazo robótico por defecto busca un desplazamiento espacial directo, mediante su cinemática desde un punto del espacio a otro, se genera una desviación de trayectoria en torno a la diferencia entre la distancia mano-herramienta.

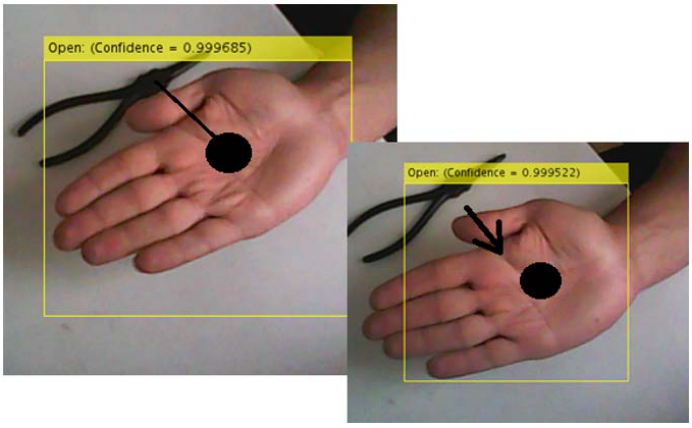


Figura 5-8: Vector direccional de la mano.

El algoritmo que evita la colisión opera en relación a determinar la posición de la herramienta, como punto de referencia. En función a este punto, se desplaza el efector final del brazo robot, con el objetivo de alcanzarlo en trayectoria directa.

Al detectar una mano, evalúa su vector direccional y lo suma al desplazamiento en el plano paralelo al de soporte de la herramienta. Desde este nuevo punto, se recalcula la ubicación del punto de referencia y se genera el nuevo desplazamiento por trayectoria directa hacia la herramienta.

La Figura **5-9** ilustra el entorno de simulación. La línea roja representa la trayectoria del efector del brazo robótico, la herramienta es simulada por el rectángulo azul, mientras que la mano del usuario es representada por el rectángulo rojo. Las ubicaciones de la mano y la herramienta son obtenidas desde la detección de cada una mediante las redes neuronales convolucionales. Como se mencionó, la mano se vuelve un obstáculo dinámico, cuya simulación varía la posición del plano que representa la mano, en función al centroide de la detección. La Figura **5-9** presenta el caso ideal, en el que el brazo robótico se mueve hacia la herramienta estando la mano estática sin obstruir el camino del brazo. El desplazamiento del brazo se realiza en función a la cinemática inversa descrita, buscando alcanzar el punto P, correspondiente a la referencia del centroide de la herramienta.

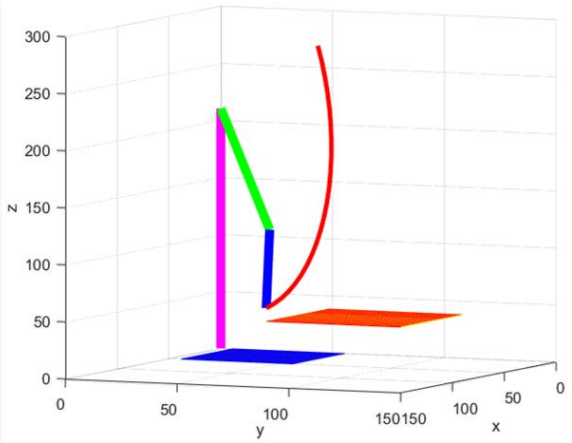


Figura 5-9: Simulación desplazamiento sin obstrucción de usuario.

La Figura **5-10** ilustra la simulación de una trayectoria, que evidencia el comportamiento del algoritmo para evitar la colisión. Al ser detectada la mano, a la izquierda del punto de referencia de la herramienta, la distancia de desplazamiento será incremental, si queda sobre la herramienta se detiene el movimiento del brazo y si es detectada a la derecha, el movimiento será decremental. El esquema simula un desplazamiento constante de la mano en z y en x, variando desde $y = -100$ hasta $y = 100$, punto en el que el brazo se mueve desde el mismo punto de inicio en y hacia la izquierda, tratando de buscar un punto de bajada hacia la herramienta, hasta que la

mano se detiene, baja y no encuentra ángulos que le permitan agarrar el objeto desde la posición final estática de la mano, se desplaza nuevamente a la izquierda y al no poder bajar sube a la distancia inicial en z para finalizar su desplazamiento.

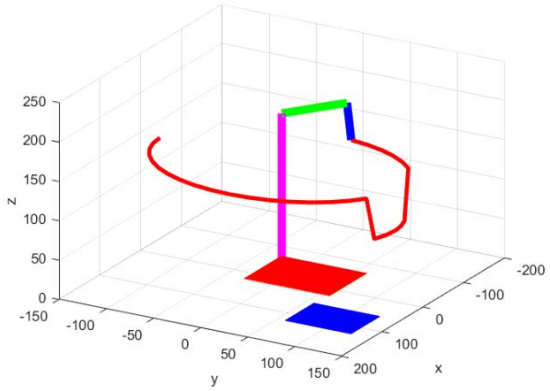


Figura 5-10: Simulación desplazamiento para evasión de usuario.

Al realizar diferentes simulaciones respecto a la incursión de la mano del usuario en el área de trabajo del robot, se observa la acción de evasión respecto al acercamiento del brazo robótico. La Tabla **5-11** ilustra el comportamiento de la variación de distancia frente al acercamiento del efector del robot. Se evidencia que el algoritmo trata de llevar el efector final hasta la herramienta (ítem 1 a 3), pero al encontrar un obstáculo (ítem 4 y 5) conserva la distancia en z y se desplaza lateralmente (en x) buscando una nueva perspectiva que le permita recalcular la trayectoria.

Tabla 5-11: Evasión de colisión.

	Dist Man-Herr	Dist z efect	Dist x efect
1	120	150	0
2	60	110	50
3	40	80	75
4	25	50	100
5	5	50	120

La Figura **5-11** ilustra un caso en el que la mano está estática junto a la herramienta. El brazo robótico inicia un desplazamiento pronunciado en el eje z, hasta ver la mano. En ese punto, gira hacia la herramienta en trayectoria recta con leves desplazamientos en z, para acercarse al punto objetivo. Nuevamente, supera la mano y ve la opción de bajar hasta la herramienta con un desplazamiento pronunciado en el eje z. Dado que al desplazarse en z hacia la herramienta viola la zona o espacio de seguridad, el algoritmo no genera más movimiento.

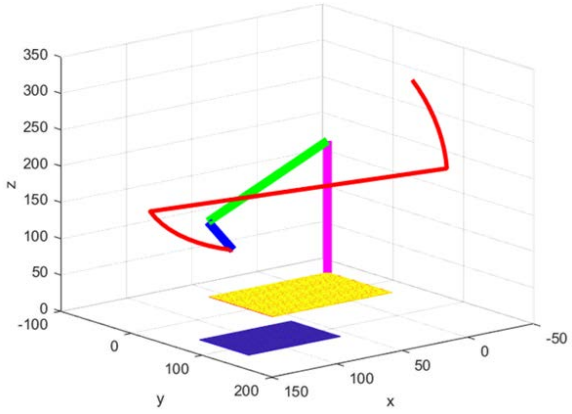


Figura 5-11: Trayectoria de evasión.

5.3. Algoritmo de agarre de herramientas

Un paso importante en la tarea de un robot asistencial, para plataformas multi-herramientas, es la toma de la herramienta, una vez identificada y ubicado el efector sobre la misma. Para esto, se requiere el desarrollo de un algoritmo específico que permita lograr la tarea. Para dar una solución a esta tarea, se desarrolla un algoritmo que permite establecer la posición de agarre de la herramienta, mediante el procesamiento de imagen de la escena del objeto discriminado, por diferencia del fondo y de tonalidad. Discriminado el objeto, se buscan los puntos de agarre más cercanos a su centroide, para obtener las posiciones x y y del punto de agarre y la orientación con que la pinza debe tomar el objeto [Moreno *et al.*, 2018].

En el Algoritmo 5, se muestra la estructura básica de los pasos seguidos para el agarre de herramientas desarrollado. Desde el momento en que se ingresa la imagen, hasta la salida de las coordenadas y la orientación del agarre para un efector tipo pinza, donde se establecen 19 pasos de ejecución del algoritmo.

Algoritmo 5: Algoritmo de agarre.

Begin

Paso 1: Inicialización de variables.
Paso 2: Captura de la imágen de entrada (Herramientas)
Paso 3: Cambio de espacio de color a blanco y negro, imágenes imHB, imHN.
Paso 4: Si $i \leq 180°$?. No, ir Paso 13.
Paso 5: Si, rotar $i = i + 1$. Paso 4.
Paso 6: Si [x,y]≤(fin de la imagen)? No, $i = i + 1$. Paso 4.
Paso 7:Si, recortar imagen, imágenes imcHB, imcHN.
Paso 8: Libre de bordes? No, $x = x + 10$ $y = y + 10$. Ir a paso 6.
Paso 9: Grosor aceptable? No, $x = x + 10$ $y = y + 10$. Ir a paso 6.
Paso 10: Inclinación $\leq 15°$? No, $x = x + 10$ $y = y + 10$. Ir a paso 6.
Paso 11: El efector esta alineado? No, $x = x + 10$ $y = y + 10$. Ir a paso 6.
Paso 12: Almacenar coordenadas de recorte y rotación, variables IgHB(:, :, n), IgX(n), IgY(n),
IgAr(n), IgAh(n), n=n+1.
Paso 13: n=0? No, ir a Paso 15.
Paso 14: NO hay zona de agarre posible.
Paso 15: Buscar el área más grande con la menor inclinación, variable iBest1.
Paso 16: Encontrar el mejor agarre al objeto, variable iBest2.
Paso 17: Mejor banlance del efector, variable iBest.
Paso 18: Recuperar coordenadas e inclinación, variables Xc,Yc y Ang.
Paso 19: Graficar resultados.

De forma general, los pasos 1 al 3, del Algoritmo 5, son los encargados de la inicialización del programa. En ellos, se realiza la captura de la imagen y la segmentación del objeto mediante un proceso de umbralización. En los pasos 4 al 7, se recortan secciones de la imagen de entrada, en diferen-

tes posiciones y orientaciones, para buscar posibles puntos de agarre en cada uno de ellos. En los pasos 8 al 12, se evalúa cada uno de los recortes y se guardan únicamente aquellos que cumplan con las condiciones de los pasos 8 al 11. Los pasos 13 y 14 se ejecutan únicamente cuando no se encuentra ningún agarre en toda la imagen. En los pasos 15 al 17, se evalúan todos los agarres guardados en el paso 12, para escoger uno de ellos como el punto de agarre ganador. Finalmente, en los pasos 18 y 19, se muestran los resultados obtenidos. A continuación, se explicará detalladamente cada uno de los pasos del Algoritmo 5.

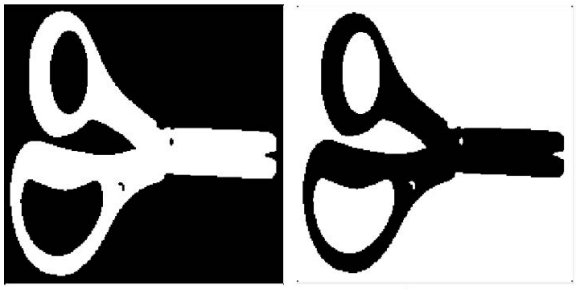
5.3.1. Inicialización del programa

En el paso 1, se inicializan las variables propias del algoritmo como las dimensiones de la pinza: apertura y ancho de las puntas. Se determina cada cuántos grados se debe rotar la imagen para evaluar posibles puntos de agarre en diferentes orientaciones. Se establece el grosor máximo y mínimo de la sección de agarre del objeto, y la cantidad de píxeles libres que debe haber entre la zona de agarre y las puntas. Se deben inicializar en el algoritmo las dimensiones de ancho y apertura de la pinza, con el fin de conocer el espacio que ocupa el efector dentro del área de trabajo y evitar choques entre los elementos cuando la pinza proceda a realizar el agarre, y se usan igualmente para definir la apertura máxima del efector al momento de seleccionar el punto de agarre.

Se deben determinar las dimensiones de la pinza en relación a sus tres características principales: largo, ancho y centro. La dimensión A representa la apertura de la pinza más el grosor de las puntas, la dimensión B representa el ancho del efector, y el punto C representa el centro de la pinza que corresponde a las posiciones Xc y Yc que determinan la posición final del agarre.

En el paso 2, se realiza la captura de la imagen del área de trabajo, la cuál es redimensionada a 220×220 píxeles. En el paso 3, se lleva a cabo el proceso de umbralización y binarización de la imagen, donde se realiza una conversión de espacio de color de la imagen original, en RGB, a escala de

grises y, finalmente, a blanco y negro. De tal manera que, el fondo de la imagen queda completamente negro y solo el objeto sobre este, queda de color blanco ($imHB$), como se observa en la Figura **5-12a**. Luego, se toma la imagen binarizada y se intercambia la intensidad de sus píxeles para obtener una segunda imagen ($imHN$), con un fondo completamente blanco y solo el objeto de color negro, como se aprecia en la Figura **5-12b**. El umbral se define en el algoritmo como 0.9, debido a que todas las imágenes que se probaron son de fondo blanco. Por lo que, no fue necesario considerar cambios de luz estableciendo un ambiente de pruebas controlado.



a. b.

Figura 5-12: Imagen binarizada.

5.3.2. Recorrido de la imagen

Sobre cada una de las imágenes binarizadas (variables $imHN$ e $imHB$), se desplazó un recuadro con las dimensiones A y B inicializadas en el paso 1, donde su posición inicial se encuentra en la esquina superior izquierda de la imagen, que coincide con la esquina superior izquierda del recuadro, y se desplaza hacia abajo una posición equivalente a 10 píxeles en busca de posibles agarres. Este recuadro representa el área que ocupa la pinza sobre el espacio cuando busca el objeto de trabajo. De tal manera que, a partir de dicho recuadro se puede estimar dónde hay un posible agarre y dónde la pinza no puede sujetar el objeto finalmente. En cada posición del recuadro, se recorta la sección de la imagen que se encuentra debajo de él, tanto para la imagen $imHN$ (donde el recorte se guarda en $imcHN$), como para $imHB$ (donde el recorte se guarda en $imcHB$). Cada recorte se evalúa de manera independiente, pasándolo por los condicionales de los pasos 8 al 11 del Algoritmo 5, para determinar si es un posi-

ble agarre o no. Cuando llega al paso 12 del algoritmo 5, o no cumple con alguno de los condicionales de los pasos 8 al 11, se desplaza el recuadro hacia abajo una posición para evaluar el siguiente recorte.

La Figura **5-13** muestra un posible agarre para la imagen $imHN$, donde se guarda la sección de imagen que se recorta del recuadro en la Figura **5-13a**. En la Sección de 5.3.3, se explica detalladamente cada uno de los condicionales de los pasos 8 al 11 del Algoritmo 5, para determinar un posible punto de agarre.

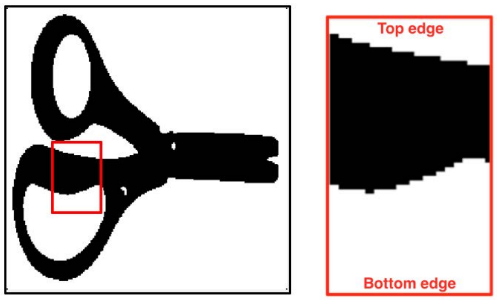


Figura 5-13: Recorte imcHN con un posible agarre.

Una vez finaliza el recorrido descendente y se alcanza la parte inferior de la imagen, el recuadro se ubica en su posición inicial, pero 10 píxeles desplazado hacia la derecha y vuelve a desplazarse hacia abajo para sacar nuevos recortes y, así sucesivamente, aumentando el desplazamiento horizontal cada 10 píxeles, hasta llegar a la esquina inferior derecha de la imagen, con todos los posibles agarres almacenados. La cantidad de píxeles que se desplaza el recuadro se seleccionó como 10 para acelerar el proceso de búsqueda de agarre, considerando las dimensiones de la imagen de entrada, y evitar guardar posibles agarres muy parecidos entre sí que aumenten la cantidad de datos a almacenar. Cuando el recuadro que recorre la imagen termina el recorrido, es decir su esquina inferior derecha coincide con la esquina inferior derecha de la imagen binarizada, vuelve a los pasos 4 y 5 del Algoritmo 5 para generar una rotación en la imagen y, de esta manera, volver a los pasos 6 y 7 para capturar nuevos recortes, pero con la imagen rotada un determinado ángulo. Este proceso permite buscar nuevos agarres con orientaciones diferentes de la pinza y se repite hasta

que la imagen se rota un total de 180^{o}, momento en que se le ha dado la vuelta completa y, por lo tanto, se han evaluado todos los posibles agarres que pueden haber sobre el objeto. Continuar con la rotación solo hará que se repitan los agarres encontrados pero en posiciones inversas, como se observa en la Figura **5-14**.

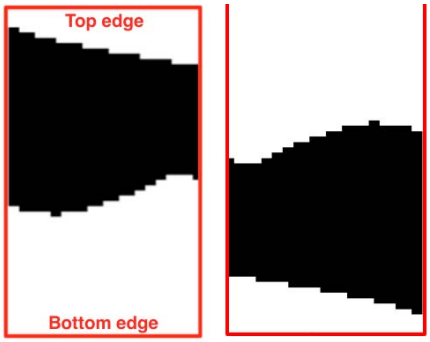


Figura 5-14: Posible agarre (a). Posible agarre invertido (b).

Se obtienen los recortes ($imcHN$ e $imcHB$) de ambas imágenes binarizadas, con el fin de aplicar todos los criterios de selección que se encuentran en los pasos 8 al 11. Para aplicar estos criterios, es necesario conocer aspectos de la imagen como la cantidad de píxeles blancos a los bordes superior e inferior del recorte $imcHN$ (marcados en la Figura **5-14**), y la cantidad de píxeles blancos en el recorte $imcHB$, para conocer el ancho del objeto capturado dentro del recorte.

5.3.3. Selección de posibles agarres

La selección de todos los posibles agarres, que pueden realizarse sobre el objeto, obedece a una serie de condiciones planteadas en los pasos 8 al 11 del Algoritmo 5. Cada condicional se aplica a cada uno de los recortes ($imcHB$ e $imcHN$) obtenidos en la Sección 5.3.2 y solo el recorte que supera todos los condicionales se considera como un posible agarre y se almacena en una variable que contiene cada posible agarre. Si el recorte no cumple con cualquiera de los condicionales se descarta y se vuelve a desplazar el recuadro para evaluar el agarre en una nueva posición.

A. Bordes libres para el agarre

El paso 8 plantea el primer condicional, el cuál busca únicamente aquellos recortes donde no haya posibilidad de choque entre las puntas de la pinza y el objeto. Para ello, realiza una sumatoria de la cantidad de píxeles blancos en la primera y última fila del recorte $imcHB$. Si la suma es igual a cero, significa que todos los píxeles son de color negro y, por lo tanto, corresponden al fondo de la imagen. Pero si la suma da diferente de cero, entonces existe algún píxel blanco que corresponde al objeto, e implica un posible choque entre la pinza y el elemento, razón por la que se elimina el recorte como posible agarre [Murillo *et al.*, 2018].

La suma de todos los píxeles de la primera y última fila del recorte $imcHB$ con fondo negro, donde $ReDim$ son las dimensiones de la imagen y S_1 el resultado de la suma, se calcula mediante

$$S_1 = \sum_{i=1}^{ReDim} imcHB(1,i) + imcHB(ReDim,i). \qquad (5\text{-}18)$$

La primera y última fila del recorte $imcHB$ se encuentran marcadas con un recuadro, que se observa en la Figura **5-15c**, y corresponde a las posiciones que ocuparían las puntas de la pinza en el espacio de trabajo. En la Figura **5-15a**, se encuentra la imagen de entrada binarizada con fondo negro $imHB$. En la Figura **5-15b**, se encuentra la imagen $imHB$ rotada con el recuadro que recorre la imagen ubicada sobre un posible agarre. Finalmente, en la Figura **5-15c**, se encuentra el posible agarre capturado por el recuadro.

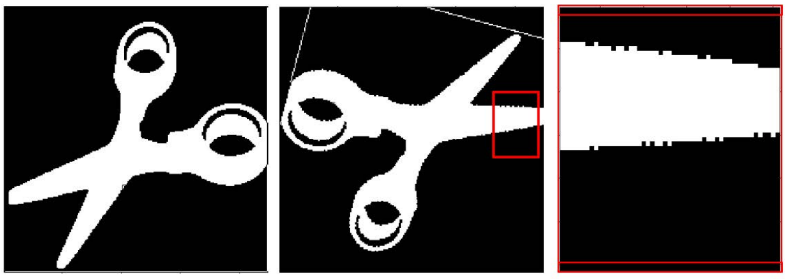


Figura 5-15: Imagen binarizada imHB (a). Imagen binarizada imHB rotada (b). Recorte imcHB como posible agarre (c).

B. Limite de grosor de la sección de agarre

El paso 9 del diagrama del Algoritmo 5 plantea el siguiente condicional, que se aplica únicamente si el recorte supera el paso 8. En este condicional, se evalúa el grosor del elemento donde se busca el agarre, con el fin evitar que se considere como posible agarre una imagen que contenga solo píxeles libres o ruido, o una superficie muy pequeña de agarre donde se corra el riesgo de que se caiga el objeto, o donde el objeto sea tan grueso que el ajuste entre la pinza y la sección del objeto sea demasiado estrecho. Para determinar qué recorte supera esta condición, se suman todos los píxeles de $imcHB$ mediante el cálculo de la variable S_2

$$S_2 = \sum_{i=1}^{ReDim} \sum_{j=1}^{ReDim} imcHB(i,j). \tag{5-19}$$

Posteriormente, se halla la cantidad total de píxeles en el recorte, a partir de la multiplicación de sus dimensiones, es decir $P_r = AB$. Acto seguido, se halla el máximo de píxeles blancos $MaxP$, mediante

$$MaxP = P_r\, P_{\text{máx}}, \tag{5-20}$$

donde $P_{\text{máx}}$ representa un porcentaje máximo de blanco. Así mismo, se calcula en mínimo número de píxeles blancos, a partir de

$$MinP = P_r\, P_{\text{mín}}, \tag{5-21}$$

donde $P_{\text{mín}}$ representa un porcentaje mínimo de blanco.

Si S_2 es mayor a $MinP$ y menor a $MaxP$, el recorte sigue al condicional del paso 10. En caso contrario, se descarta y se vuelve al paso 6 del algoritmo 5 para desplazar el recuadro una posición.

C. Inclinación máxima de la sección de agarre

El paso 10 del Algoritmo 5 evalúa el grado de inclinación, inc, de la sección del objeto en el recorte con respecto a la superficie de las pinzas. Para obtener el grado de inclinación se traza una elipse sobre el área de

píxeles blancos del recorte, como se muestra en la Figura **5-16a**, y se calcula la inclinación del eje más largo de la elipse con respecto al eje x, como se muestra en la Figura **5-16b**.

Figura 5-16: Cálculo de la inclinación de la sección del objeto.

La Figura **5-16a** muestra los cuadrados blancos, que representan los píxeles y, la línea curva, que representa la elipse que se traza sobre el área. En la Figura **5-16b**, la línea curva es la elipse trazada sobre el área de píxeles blancos, las líneas continuas son los ejes de la elipse, y la línea punteada es el eje x con respecto al cual se mide el grado de inclinación.

El valor de inclinación máxima, $IncMax$, del objeto con respecto al efector es determinado por el usuario e ingresado en grados. La sección del objeto capturada en los recortes debe tener una inclinación inferior o igual a la inclinación máxima $IncMax$ para superar la condición del paso 10. Este criterio se agregó con el fin de controlar la máxima inclinación admisible entre el objeto y la pinza, dado que grandes valores de inclinación pueden generar que el objeto se mueva de su ubicación y reduzca la precisión del agarre mientras el efector se está cerrando.

D. Espacio libre para el efector

El paso 11 del Algoritmo 5 muestra la última condición que debe superar el recorte para guardarse como un posible agarre. El criterio consiste en buscar aquellos recortes donde haya una cantidad mínima de píxeles blancos en los bordes superior e inferior del recorte, con el fin de aceptar únicamente aquellos que tengan el espacio suficiente para que encajen las puntas de la pinza en el agarre. Debido a que la dimensión A considera, no

solo la apertura de la pinza sino también el ancho de las puntas, se debe tener en cuenta que parte del área del recorte incluye el área que ocupa la pinza en el espacio.

Se debe tener en cuenta el grosor de las puntas de la pinza, que equivale a la cantidad mínima de píxeles blancos que debe haber al interior del recorte para asegurar que existe el espacio suficiente para que el efector encaje con el objeto.

Para determinar qué recortes superan el último condicional, se define el número de filas superiores e inferiores del recorte que deben contener píxeles blancos. De tal manera que, la cantidad de filas a evaluar corresponda al grosor real de las puntas de la pinza. Se obtiene un porcentaje, Pg, del área que ocupa cada punta de la pinza con respecto al área total del cuadro, obteniendo un Pg=10% para cada punta y un 80% del área libre para el agarre del objeto. Con el porcentaje, se calcula el número de filas de píxeles, fp, que deben sumarse para evaluar el condicional del paso 11. El calculo del número de filas de píxeles (fp) se realiza como

$$fp = A\,Pg, \tag{5-22}$$

y la suma de píxeles del recorte en las filas superior e inferior se calcula como

$$S_3 = \sum_{i=1}^{fp} \sum_{i=1}^{ReDim} imcHN(i,j) + imcHN(ReDim - i + 1, j). \tag{5-23}$$

Después de obtener la suma de píxeles S_3, se realiza una comparación entre S_3 y la cantidad mínima de píxeles blancos, Pw, que debe haber a los extremos del recorte multiplicado por un factor de ruido, wh, entre 0 y 1. Ese factor, permite considerar la existencia de áreas muy pequeñas de píxeles blancos en los bordes superior e inferior del recorte, donde dicho factor es determinado por el usuario según la calidad de la imagen binarizada que se obtenga tras la segmentación. Se requiere realizar el cálculo de la cantidad mínima de píxeles blancos que debe tener el recorte en los bordes superior e inferior, según el condicional del paso 11, a partir de

$$Pw = 2\,wh\,A\,B\,Pg. \tag{5-24}$$

E. Guardar posibles agarres

El paso 12 del Algoritmo 5 presenta el almacenamiento de todos los recortes que superan los condicionales de los pasos 8 al 11 como posibles agarres. Se almacena el recorte en una variable, $IgHB$; las coordenadas de su esquina superior izquierda ($x,\ y$) en los arreglos Igx e Igy respectivamente; el ángulo de inclinación de la sección de agarre con respecto a la superficie de las puntas de la pinza, en el arreglo $IgAr$; y el grado de rotación en el que se encuentra la imagen binarizada al momento de evaluar el recorte, se guarda en el arreglo $IgAh$.

En caso de que la imagen binarizada rote los 180^o y al terminar no guarde ningún recorte como posible agarre, el algoritmo entra en el condicional de los pasos 13 y 14, para indicar que no encontró ninguna zona de agarre y finaliza el proceso. En caso contrario, entra a los condicionales de los pasos 15 a 17, para escoger un solo agarre de todos los almacenados.

5.3.4. Selección del agarre final

Una vez almacenados todos los posibles agarres de la herramienta, se debe seleccionar la mejor opción entre ellos. A continuación se detalla este procedimiento.

A. Mayor área y menor inclinación

El primer criterio de selección del punto agarre se enfoca en buscar el recorte con la mayor área de agarre y el menor grado de inclinación de la herramienta con respecto al efector. Para ello, en el paso 15 del Algoritmo 5, se guardan los diez recortes con la mayor área de agarre y luego se escoge entre ellos aquel que tenga la menor inclinación. Para calcular el área de agarre de cada recorte, se suman todos los píxeles del recorte (aplicando (5-19)) y se escogen diez que tengan la mayor suma. Después, se comparan los grados de inclinación de los diez recortes ganadores (almacenados en la variable $IgAr$), y se escoge el que tenga el menor valor sin considerar

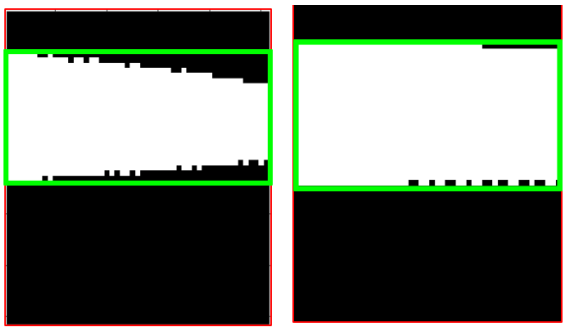
el signo. Finalmente, se guarda en la variable $iBest1$ la posición que ocupa el recorte ganador de los posibles agarres.

B. Mejor encaje

El segundo criterio de selección se enfoca en buscar el agarre que tenga la mayor superficie de contacto entre el objeto y la pinza, con el fin de obtener la mayor fricción entre ambos. Para ello, en el paso 16 del Algoritmo 5, se busca la sección del objeto que se asemeje más a un rectángulo cuyos bordes superior e inferior sean paralelos a la superficie de las puntas de la pinza. Para ello, se evalúa el encaje de la sección del objeto capturado en el recorte con respecto al borde inferior del recorte. Sobre los píxeles blancos que se encuentran conectados entre sí y representan al objeto en el recorte, se traza un rectángulo que los cubre por completo (como se observa en la Figura **5-17**), con coordenadas $(x,\ y)$ sobre la esquina superior izquierda, y el valor del ancho y el alto del mismo se almacena en las variables x_w, y_w, respectivamente.

Con las dimensiones obtenidas de ancho y alto del rectángulo, se calcula su área y se le resta la cantidad total de píxeles blancos (usando (5-19)) del recorte correspondiente guardado en $IgHB$. De tal manera que, si la resta es igual a cero, significa que la sección del objeto es completamente rectangular y, por lo tanto, tiene el mejor encaje con las puntas de la pinza. En caso contrario, se supone que la superficie del objeto está inclinada, o es irregular, o tiene una geometría curva y no hay suficiente superficie de contacto para realizar el agarre. De todos los recortes almacenados, se escoge como agarre ganador aquel cuya resta entre el área de la caja y la suma total de píxeles S_2 sea mínima, y se guarda en $iBest2$ su posición en el arreglo $IgHB$.

La Figura **5-17a** muestra el recuadro que cubre la sección del objeto y los espacios vacíos entre el objeto y la caja, mientras que en la Figura **5-17b** se muestra un encaje más exacto entre la sección del objeto y la caja. El recorte ganador entre ambos, según el criterio del paso 16, sería el de la Figura **5-17b**.



a. b.

Figura 5-17: Selección del mejor encaje.

C. Selección por centroide

Después de obtener los resultados $iBest1$ e $iBest2$, el algoritmo sigue al paso 17, donde se escoge entre los recortes ganadores el que se encuentra más cerca al centroide del objeto para asegurar un mayor equilibrio en el agarre.

Primero, se calcula la distancia entre el centro de los recortes ganadores y el centroide del objeto. Para luego, escoger la menor distancia de los dos y guardar el resultado en $iBest$. De tal manera que, si gana el recorte de mayor área, se guarda $iBest1$ en $iBest$, en caso contrario se guarda $iBest2$ en $iBest$.

5.3.5. Entrega de resultados

El paso 18 del Algoritmo 5 entrega la posición y el ángulo de inclinación del recorte que contiene el agarre final, utilizando la posición $iBest$ guardada en el paso anterior. Las coordenadas se buscan en las matrices generales $\mathbf{IgX}$, $\mathbf{IgY}$ e $\mathbf{IgAh}$, y se guardan como x_c, y_c y Ang, respectivamente, para generar las salidas. Después, en el paso 19, se grafican los resultados, como se muestra en la Figura **5-18**. Allí, el recuadro representa la posición (en píxeles) y el ángulo de inclinación (en grados) de la pinza sobre el objeto, el asterisco indica la posición (x_c,y_c) del agarre ubicado en el centro del recuadro, Ang es el ángulo de rotación que debe tener el

efector para realizar el agarre, y el círculo es el centroide del objeto con coordenadas (x_o,y_o).

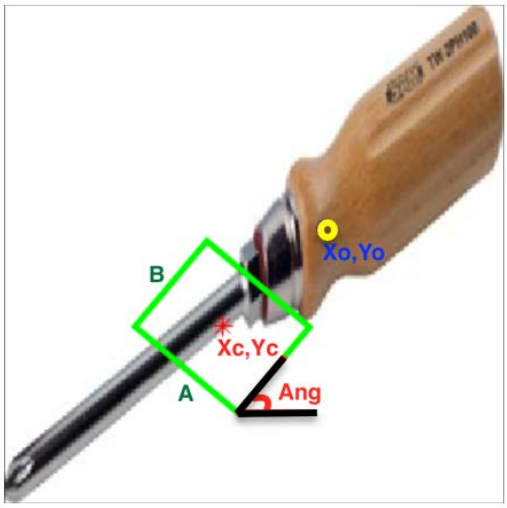


Figura 5-18: Posición y orientación de la pinza para el agarre escogido.

El lado A del rectángulo de la Figura **5-18** representa la apertura máxima de la pinza, o la dimensión A, inicializada en el paso 1 del Algoritmo 5. El lado B representa el grosor de las puntas de la pinza, o la dimensión B, inicializada en el mismo paso. Para el caso de lo mostrado en la figura, se tiene que $x_c = 99$, $y_c = 145$, $Ang = 50^{\circ}$, $t = 1{,}723758s$. Los valores de x_c se toman positivos desde el borde izquierdo de la imagen hacia la derecha, y y_c positivo desde el borde superior de la imagen hacia abajo. El grado de inclinación Ang se tomó con respecto a la horizontal de la imagen y la horizontal del recorte, tal y como se observa en la Figura **5-18**.

Se validó el algoritmo desarrollado con varios tipos de objetos, diferentes dimensiones A y B del efector, y con la inicialización de variables establecidas en la Tabla **5-12**.

Tabla 5-12: Inicialización de variables.

Variable	Valor
$P_{\text{máx}}$	$90\,\%$
$P_{\text{mín}}$	$20\,\%$
$Grado$	$5°$
$IncMax$	$15°$
Pg	$10\,\%$
wh	$98\,\%$

La variable $P_{\text{máx}}$ se escogió como 90%, para permitir secciones de agarre lo suficientemente amplias. Por su parte, $P_{\text{mín}}$ se seleccionó como 20%, tanto para evitar secciones de agarre muy delgadas como para filtrar falsos *"posibles agarres"* generados por píxeles blancos que no pertenecen al objeto (ruido). La variable Grado indica cada cuántos grados se rota la imagen binarizada, en los pasos 4 y 5 del Algoritmo 5, para la búsqueda de los posibles agarres y se escogió como 5^{o}, con el fin de reducir el tiempo de ejecución del algoritmo sin perder muchas posibilidades de agarre. Esta variable debe asignarse con un valor igual o superior a la resolución de rotación del efector, ya que este definirá el grado de rotación final de la pinza para realizar el agarre sobre el objeto. La variable $IncMax$ se escogió como 15^{o} tanto para reducir el tiempo de ejecución del programa, al filtrar todas las otras inclinaciones, como para evitar que el objeto se mueva demasiado cuando la pinza se ajuste sobre él, pues esto puede afectar el agarre, al generar un desplazamiento en el objeto. La variable Pg se inicializó como 10%, suponiendo que las puntas de la pinza no ocupan, cada una, más de un 10% del área total de agarre. El porcentaje de ruido aceptable wh se definió como el 2% debido a que las imágenes empleadas no tienen mucho ruido debido al fondo blanco.

5.3.6. Análisis y Resultados del agarre

La primera prueba, ilustrada en la Figura **5-19** en la izquierda, arrojó una sección de agarre ubicada sobre el filo de las tijeras para dimensiones $A = 60$ píxeles y $B = 50$ píxeles. Bajo estas medidas no se tuvo la apertura suficiente para realizar el agarre sobre el mango de la herramienta, por lo

que la única opción que encontró el algoritmo fue el filo, una posición complicada de manejar para un efector tanto por el grosor de la sección del objeto, como por su geometría, su distancia con respecto al centroide, y el cuidado necesario para el filo del elemento. El tiempo de cómputo (t) del algoritmo fue de $2{,}017$ segundos para calcular el agarre y graficar los resultados. La geometría de la sección de agarre es inclinada, por lo que puede generar deslizamiento entre las pinzas y el objeto y hacer que las tijeras se caigan, al igual que la distancia que existe entre el agarre y el centroide que puede generar que el peso de las tijeras afecte la estabilidad del agarre.

Figura 5-19: Punto de agarre tijeras abiertas.

Debido a los resultados obtenidos en la primera prueba, se cambiaron las dimensiones del recuadro por $A = 50$ píxeles y $B = 30$ píxeles, obteniendo los resultados de la Figura **5-19** derecha. En este caso, el agarre que seleccionó el algoritmo se encuentra más cerca del centroide de la herramienta y está ubicado sobre el mango de las tijeras. A comparación de la prueba anterior, las probabilidades de que se resbale el objeto debido a la geometría del agarre se reducen, ya que tanto el dedal como el tope de las tijeras tienen mayor área que el asta donde se realiza el agarre, y no hay riesgo de dañar la calidad del filo. El inconveniente con el agarre de la derecha es que la ubicación del efector sobre la herramienta debe ser muy preciso para que la pinza encaje en medio los dedales de la tijera sin tocarla.

En una segunda prueba, se buscaron los puntos de agarre sobre un bisturí quirúrgico obteniendo los resultados mostrados en la Figura **5-20** para dimensiones $A = 60$ píxeles y $B = 50$ píxeles. Debido a que la geome-

tría del bisturí no es tan irregular como la de las tijeras, cualquier posible agarre se ubica sobre su cuerpo y se diferencian entre sí por la distancia que existe entre el agarre y el centroide de la herramienta. Para el caso el agarre que escogió el algoritmo inicialmente (lado izquierdo), se encuentra alejado del centroide, pero en una zona donde el encaje entre el objeto y la pinza es elevado, cumpliendo con la condición de selección del paso 16 del Algoritmo 5 donde se obtiene $iBest2$. Como recorte ganador, el algoritmo escogió el de mejor encaje a pesar de no estar ubicado sobre el centroide, ya que la condición del centroide se evalúa después del condicional que selecciona los dos mejores agarres, y entre los agarres seleccionados el más cercano al centroide era el de mejor encaje con una distancia aproximada de 62 píxeles, mientras que el de mayor área se encontraba a 83 píxeles de distancia.

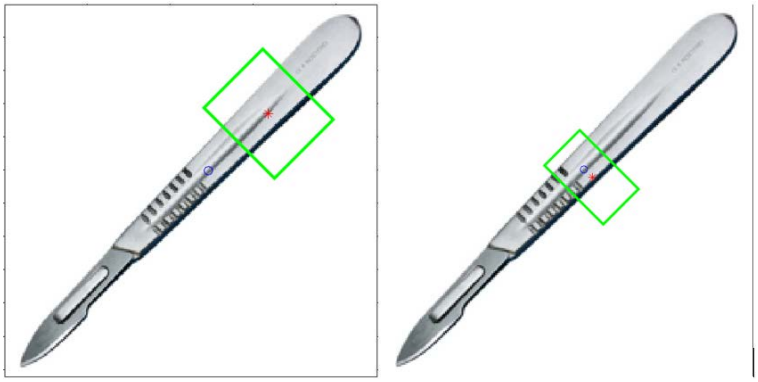


Figura 5-20: Puntos de agarre para un bisturí.

Para las dimensiones $A = 50$ píxeles y $B = 30$ píxeles, el agarre final se ubicó casi sobre el centroide del objeto como se observa a la derecha, en la Figura **5-20**. Al reducir las dimensiones del recuadro, se redujo el área de cada recorte, haciendo que la evaluación de cada condicional de los pasos 8 al 11 y los pasos 15 y 16 entregaran resultados diferentes a los obtenidos en el caso inicial, visto en la izquierda de la figura. Por lo tanto, el recorte ganador no podía contener áreas tan grandes del objeto, ni los mismos encajes que en el caso anterior, haciendo que el agarre se acercara más a una zona más angosta de la herramienta.

A partir de las pruebas realizadas es posible observar que, dependiendo de las dimensiones del efector que se maneje en la aplicación, se pueden obtener diferentes puntos de agarre para el mismo objeto. Por otro lado, la calidad del agarre depende de los condicionales de los pasos 15 y 16 del Algoritmo 5, ya que ellos definen cuáles son los dos mejores agarres entre los que se escoge el ganador.

La distancia entre el agarre y el centroide es importante para encontrar agarres estables. No obstante, para el objeto presentado, su relevancia apareció después de haber escogido dos posibles agarres. Esto se debió a que el centroide que se obtiene en el algoritmo está basado en la geometría del objeto más no en su peso, por lo que en casos donde el centro de masa y el centroide del objeto difieran considerablemente, un agarre sobre el centroide no puede asegurar un agarre estable, razón por la que se establecieron los condicionales de los pasos 15 y 16 para tratar de establecer agarres que consideren cambios de masa en diferentes áreas del objeto.

Capítulo 6

Ambiente Virtual de Prueba para Plataforma Multi-herramienta

Con el propósito de integrar la tarea de reconocimiento de un robot asistencial en una plataforma multi-herramienta, se emplea un ambiente virtual, asociado a una sala de cirugía, desarrollado en el lenguaje de modelización de realidad virtual (VRML, de sus siglas en inglés, Virtual Reality Modeling Language) de Matlab®, empleando el robot manipulador asistencial mostrado en la Figura **5-1**. El escenario consiste en el manipulador, una plataforma multi-herramientas y una mano lista para recibirlas. De forma tal que, sobre la plataforma virtual se establece un punto de agarre sobre la herramienta deseada y se mueve el manipulador hasta dicho punto, para tomar la herramienta y trasladarla directamente a la mano del usuario. La selección de la herramienta se ingresa de forma manual y se emplea una captura de datos externa por cámara para validar el algoritmo de la red neuronal convolucional de profundidad desarrollada en la Sección 4.2.

6.1. Ambiente virtual

El entorno virtual empleado consta de una mesa quirúrgica de aspecto metálico sobre la que se ubica el manipulador y las herramientas. La

mano virtual simula el punto final al que debe llevarse la herramienta seleccionada, dicha selección se realiza ingresando el nombre de la herramienta en el programa (*ver Apéndice B*).

El manipulador fue ensamblado pieza por pieza mediante el software SolidWorks, iniciando con una base fija que se agregó anidada a la mesa y terminando con el efector (pinza), este incluye pequeñas esferas metálicas en cada articulación, que simulan los motores del robot, para permitir los movimientos rotacionales de cada eslabón. Cada pieza adicional del manipulador se añadió anidada a las piezas anteriores, de tal manera que cualquier cambio de rotación en alguno de los motores de las articulaciones genera un movimiento en los eslabones siguientes hasta llegar al efector, tal y como opera el manipulador real.

Se adicionó una caja rectangular sobre la mesa metálica para poner sobre ella las herramientas, y así permitirle al manipulador un mayor rango de movimiento al que tendría cuando tiene que ir hasta un punto ubicado a la altura de su base. Sobre la caja se ubicaron tres herramientas, una tijera, un destornillador y un bisturí, cada uno de diferente color para reconocerlos con mayor facilidad visualmente. En la Figura **6-1** se muestra el ambiente virtual desarrollado [Moreno *et al.*, 2017].

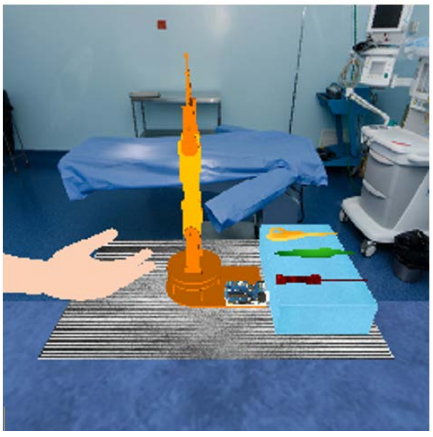


Figura 6-1: Ambiente virtual.

La captura externa se realiza mediante una webcam, empleando técnicas convencionales de procesamiento de imágenes basadas en binari-

zación, para segmentar cada herramienta y someterla a la red neuronal convolucional desarrollada en el Capítulo 4. La Figura **6-2** ilustra este procedimiento. Se observa mediante el recuadro verde las regiones de interés que serán ingresadas a la red neuronal convolucional para su clasificación. Debido a que se emplean imágenes no incluidas dentro de la base de datos del entrenamiento se validan las activaciones de cada herramienta, respecto a los filtros originales de la red neuronal convolucional entrenada. De forma tal que, al evaluar las activaciones de la red con los filtros aprendidos se tiene el resultado visto en la Figura **6-3**. Esto permite concluir el buen desempeño de la red.

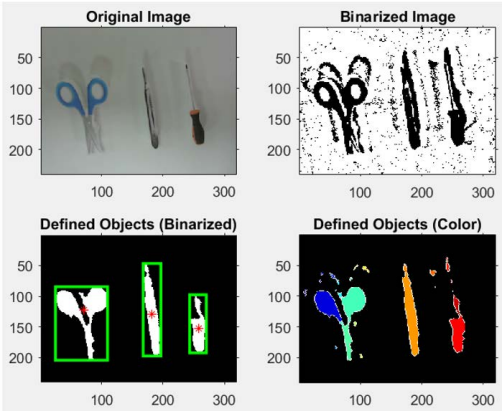


Figura 6-2: Entrada del ambiente virtual.

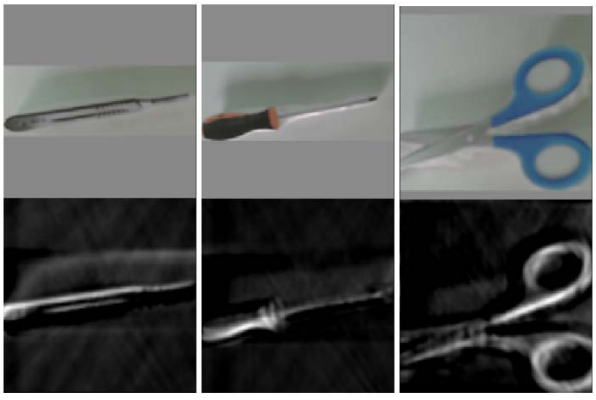


Figura 6-3: Activaciones de las redes neuronales convolucionales.

La validación de la red neuronal convolucional de profundidad se realiza moviendo manualmente la distancia de captura de la imagen respecto

las herramientas, obteniendo el resultado ilustrado en la Tabla **6-1.** Donde se observa que, el reconocimiento se realiza eficientemente con niveles de desempeño cercanos, haciendo el reconocimiento inmune a la variación de distancia. La Figura **6-4** ilustra el proceso para una distancia de 30 cm.

Tabla 6-1: Detección en profundidad para el ambiente virtual.

Distancia (cm)	Acierto (%)		
	Tijeras	Destornillador	Bisturí
20	100	99.82	98.34
30	96.81	97.79	96.85
40	99.43	99.23	99.12
50	96.93	97.62	96.98
60	98.33	97.93	97.45
70	96.12	96.34	97.02
80	96.56	96.95	97.12

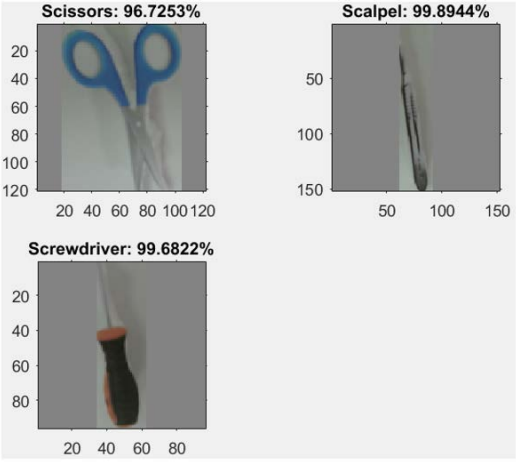


Figura 6-4: Desempeño CNN de profundidad para el ambiente virtual.

6.2. Pruebas de agarre de herramienta

Parte integral del ambiente de simulación y de las pruebas reales está asociada al algoritmo de agarre, que permite tomar la herramienta y entregarla al usuario, lo que demarca la función asistencial. Se establece una evaluación de desempeño del algoritmo operando en ambiente real. Ya

que, a diferencia de la simulación, el ambiente real presenta rozamientos y fricciones no simuladas, así como las imprecisiones propias del movimiento de los servo motores. La Tabla **6-2** relaciona el número de aciertos en la repetición de 25 agarres por herramienta. Se puede evidenciar un buen desempeño para las herramientas simples, como el bisturí y el destornillador, mientras que se le dificulta más al robot el agarre de herramientas con espacios entre parte de la misma como las pinzas y las tijeras, por tener mayor volumen.

Tabla 6-2: Precisión agarre.

	Número de errores	Precisión
Pinzas	5	80 %
Tijeras	3	88 %
Destornillador	2	92 %
Bisturí	2	92 %

A continuación se ilustrarán los agarres de diferentes herramientas. La Figura **6-5** ilustra uno los agarres obtenidos por el sistema para la herramienta tipo tijeras. Se puede observar cómo el efector del robot logra realizar un buen agarre cerca del centro de gravedad de la misma. La Figura **6-6** ilustra uno de los agarres obtenidos por el sistema para la herramienta tipo bisturí, el agarre da hacia uno de los extremos pero permite su transporte. La Figura **6-7** ilustra uno de los agarres obtenidos por el sistema para la herramienta tipo destornillador, a pesar de que el punto de agarre da hacia un extremo, el cierre de la pinza ajusta su posición. La Figura **6-8** ilustra uno de los casos de fallo. Para el ejemplo, el fallo se da con respecto a la herramienta bisturí, donde la cercanía con las tijeras reduce el campo de sujección y determina el punto de agarre muy al extremo, lo que ocasiona la caída de la herramienta.

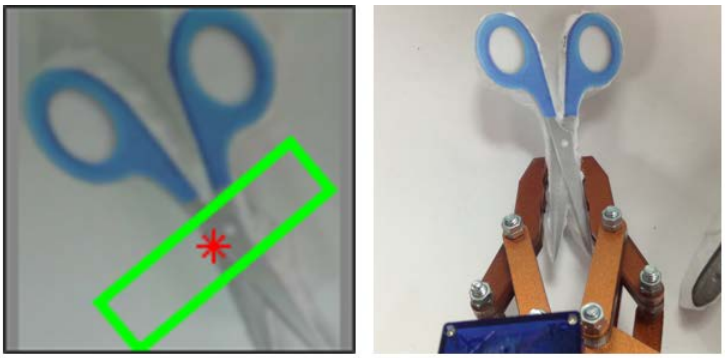


Figura 6-5: Punto de agarre tijeras.

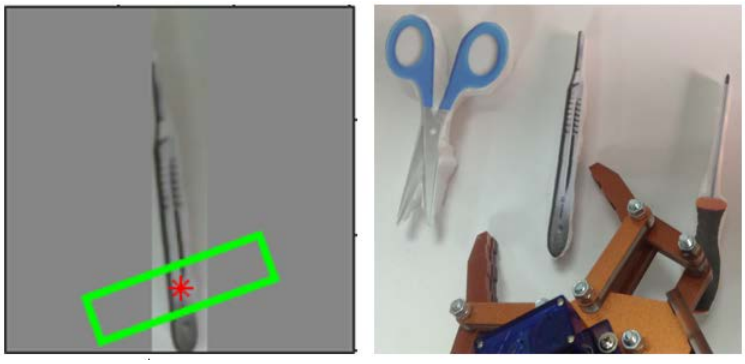


Figura 6-6: Punto de agarre bisturí.

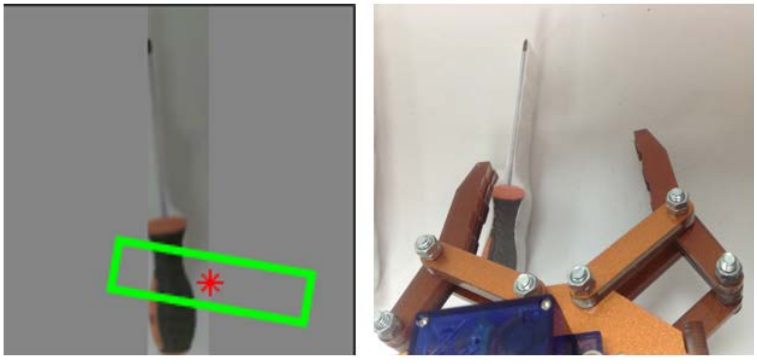


Figura 6-7: Punto de agarre destornillador.

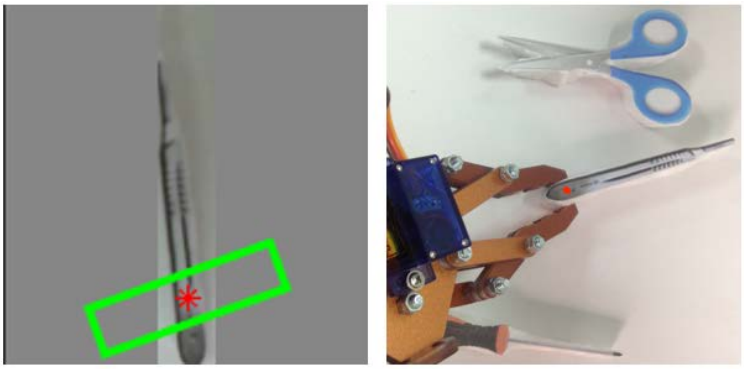


Figura 6-8: Agarre fallido.

6.3. Simulaciones de ubicación, agarre y entrega de herramienta

Por medio de la cámara web empleada, se logra obtener un ambiente híbrido real-virtual, lo que permite identificar las herramientas con la red neuronal convolucional diseñada y aplicar el algoritmo de agarre sobre la escena real. Dichos parámetros son replicados de forma ideal en el ambiente virtual y evidenciados posteriormente por la acción del robot, donde sí se evidencian fallos de funcionalidad. A continuación, se muestran algunas imágenes de pruebas realizadas.

La Figura **6-9** ilustra algunas escenas de la operación de agarre de la herramienta destornillador y posterior entrega al usuario final, en las etapas de ubicación y agarre, durante las que se evalúan las variaciones dinámicas de las características por cambios de distancia.

La Figura **6-10** ilustra el proceso de transporte y entrega en la mano al usuario final. La Figura **6-11** ilustra algunas escenas de la operación de agarre de la herramienta tijeras y, posterior, entrega al usuario final, en las etapas de ubicación, agarre, transporte y entrega. Como se indicó, el robot replica la acción prevista en la simulación, tomando la herramienta y llevándola a la mano del usuario. La Figura **6-12** ilustra el proceso de ubicación, desplazamiento y agarre para la herramienta tijeras en un ambiente real. La Figura **6-13** ilustra el proceso de entrega al usuario para la herramienta tijeras en un ambiente real.

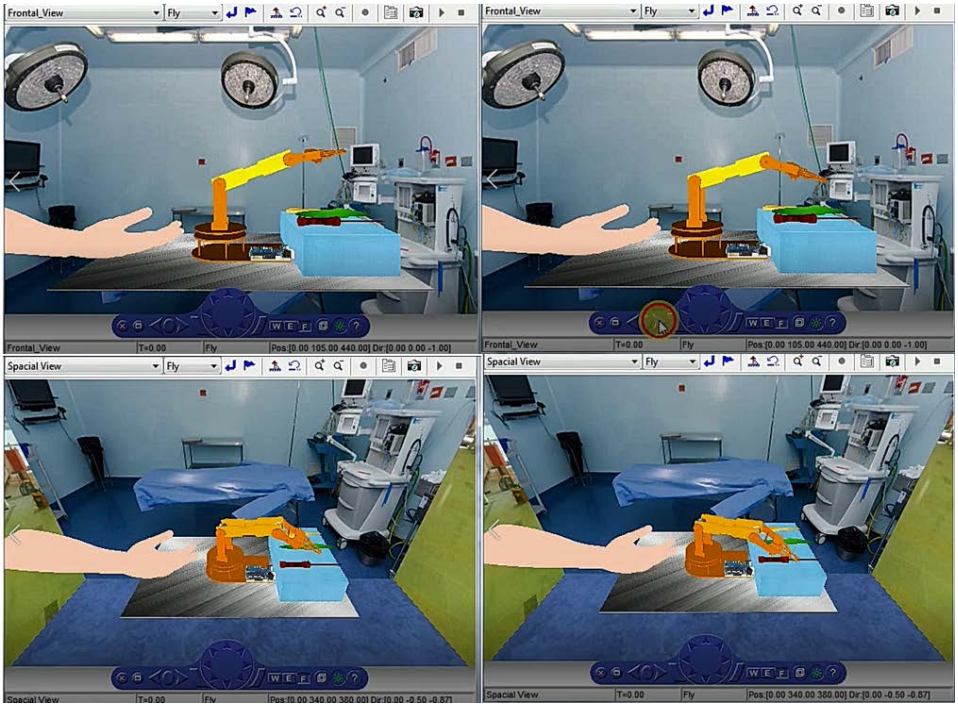


Figura 6-9: Pruebas de simulación para destornillador.

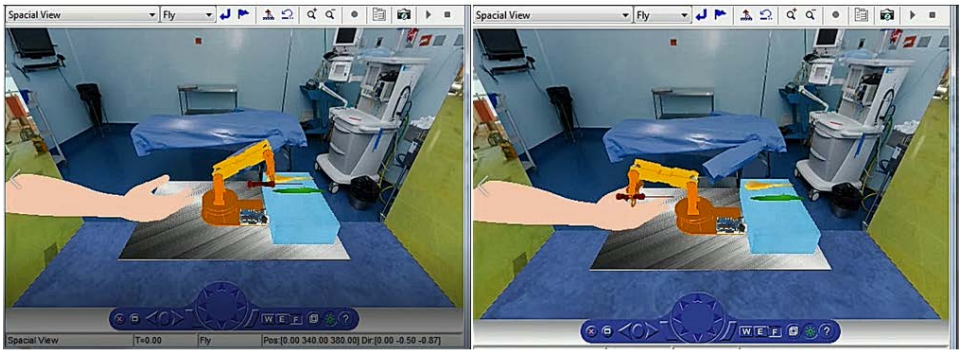


Figura 6-10: Simulación entrega destornillador.

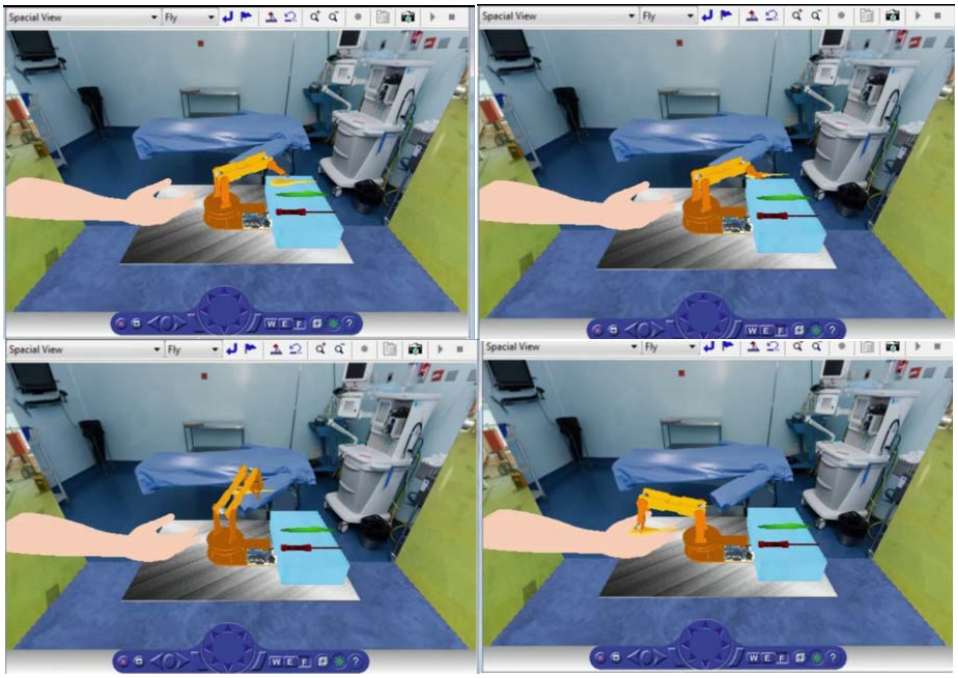


Figura 6-11: Pruebas de simulación para tijeras.

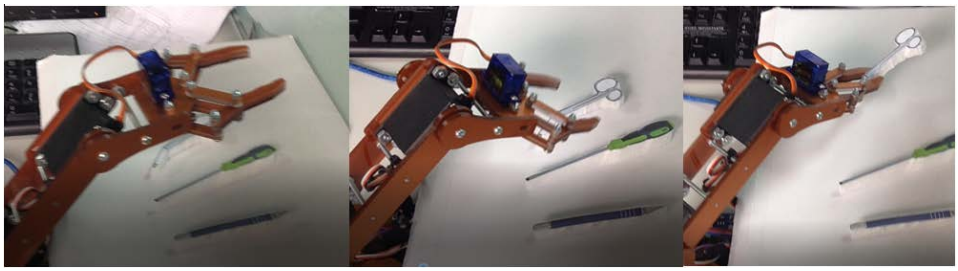


Figura 6-12: Ambiente real con tijeras.

Figura 6-13: Ambiente real entrega tijeras.

Conclusiones y Trabajo futuro

Conclusiones

Se logró verificar el alto desempeño que brindan las redes neuronales convolucionales en las aplicaciones de reconocimiento de patrones en imágenes y sus evidentes ventajas frente a los métodos tradicionales, que implican procesamiento de imagen y redes neuronales convencionales. Aún así, esta funcionalidad de las redes neuronales convolucionales, propicia para el trabajo con imágenes, tiene limitaciones inherentes que abren la posibilidad a mejoras en las arquitecturas establecidas, siendo dichas mejoras objeto de investigación y aportes al conocimiento, como es el caso aquí expuesto, en relación a la variación dinámica de la cámara y el desempeño de la red.

Al variar los diferentes hiper-parámetros propios de las redes convolucionales, se logró converger a una arquitectura de red, que permitiese obtener un alto desempeño en el reconocimiento de patrones en imágenes. Dicha arquitectura está orientada a discriminar herramientas a una distancia fija del foco de la cámara de captura. Sobre esta arquitectura se validó la necesidad de mejorar el desempeño de las redes convolucionales, al variar la distancia de reconocimiento de las herramientas, donde el des-

empeño se degrada con la perdida de características que la variación de distancia genera en las imágenes que ingresan a la red.

El generar un entorno de aplicación en robótica asistencial basado en reconocimiento de patrones, empleando redes neuronales convolucionales, permitió evidenciar una necesidad puntual de mejora en el desempeño del reconocimiento, frente a ambientes dinámicos que se puede alcanzar con variaciones de la arquitectura base de este tipo de red. El proponer una arquitectura paralela, que da la capacidad a la red de generar un reconocimiento en profundidad, ofreció una solución satisfactoria frente al reconocimiento en ambientes dinámicos, permitiendo no solo solucionar la pérdida o aparición de características de un objeto en una imagen, cuando la distancia de captura varía, sino un método de planeación de trayectorias para evasión de obstáculos dinámicos, como lo es la mano de un usuario en el área de trabajo del robot.

Dentro de las soluciones planteadas en la arquitectura paralela propuesta, se presentó el diseño de la capa final mediante dos opciones: una, basada en una ponderación aritmética con saturación y, la otra, mediante un sistema de inferencia difusa. Aunque ambas soluciones resuelven las necesidades de la capa final de ponderación, el uso de una ecuación aritmética ofrece una solución genérica para diferentes niveles de profundidad, a diferencia del sistema difuso, que debe ser rediseñado con cada agregación de nivel. La saturación permite delimitar la salida de la red obteniendo un 100% de acierto en la clasificación en los puntos de profundidad del entrenamiento que, a su vez, sirven de referencia espacial del desplazamiento del manipulador, lo que da la versatilidad a la red en la discriminación multi-distancia y de planeación de trayectoria en tres dimensiones.

Se evidencia que el solucionar el problema de reconocimiento dinámico de objetos, para un robot asistencial desplazándose en tres dimensiones, es solo una parte de la tarea de asistencia. Los algoritmos presentados para planeación de trayectoria y agarre, permiten esbozar el entorno completo de desarrollo, que requiere la implementación de un agente ro-

bótico de este tipo. Sin embargo, estos algoritmos son dependientes del tipo de efector final a utilizar en un ambiente real.

El ambiente híbrido real-virtual presentado, permitió validar la funcionalidad de los algoritmos diseñados y emular el ambiente integral de un robot asistencial capaz de entregar una herramienta a un usuario, con sus ventajas y desventajas.

Trabajo Futuro

El campo de la robótica asistencial está en desarrollo constante y algoritmos como los propuestos en el presente documento permiten tener una visión clara de los requerimientos mínimos que se deben satisfacer. Dentro de las aplicaciones desarrolladas como trabajo futuro se encuentra el análisis de obstáculos adicionales a la mano de un usuario, como puede ser el brazo, el cuerpo o incluso el rostro del mismo, que son susceptibles de verse involucrados dentro del área de trabajo del robot.

Las pruebas realizadas se aplicaron en función a que el efector final del brazo robótico posee la cámara de visión y que esta toma la imagen siempre hacia abajo, es decir, en orientación hacia el suelo, o más específicamente hacia el área de trabajo donde se encontrarían las herramientas. Lo cual abre la posibilidad a un trabajo futuro basado en una cámara de exploración omnidireccional, que valide el entorno del robot para generar trayectorias que puedan ser susceptibles de colisiones laterales y que en este trabajo no fueron consideradas. Otra alternativa es emplear un par de cámaras para detectar la profundidad y así obtener la información lateral.

El trabajo desarrollado fue delimitado a robots académicos con limitaciones en los actuadores y la dimensión de sus eslabones, lo cual hace que algoritmos de control para cada grado de libertad se deban considerar en un robot industrial o para un ambiente real. Esto a su vez ampliaría el área de trabajo del robot, lo cual podría implicar aumentar el nivel de profundidad por distancia que debe manejar la red.

Si bien el entorno multi-herramienta se desarrolló eficientemente, este tipo de aplicación puede ser orientada a otros escenarios diferentes a las herramientas propuestas. Por ejemplo, un entorno de sala de cirugía al manejar tijeras especializadas que varían levemente en su parte terminal de corte, siendo igual en la de sujeción, requeriría una estructura diferente de red convolucional por la limitada divergencia de patrones que exponen.

Bibliografía

[Abdel-Malek y Othman, 1999] Abdel-Malek, K. y Othman, S. (1999). Multiple sweeping using the Denavit-Hartenberg representation method. Computer-Aided Design, 31(9):567 – 583.

[Ansari *et al*., 2012] Ansari, U., Alam, S., y Jafri, S. M. U. N. (2012). Trajectory optimization and adaptive fuzzy based launch vehicle attitude control. En 2012 20th Mediterranean Conference on Control Automation (MED), pp. 457–462.

[Azarang y Kehtarnavaz, 2020] Azarang, A. y Kehtarnavaz, N. (2020). A review of multi-objective deep learning speech denoising methods. Speech Communication, 122:1 – 10.

[Babuska, 2001] Babuska, R. (2001). Fuzzy and neural control. DISC Course Lecture Notes, Delft University of Technology, Delft, The Netherlands.

[Bai *et al*., 2016] Bai, Y., Chen, Z., Xie, J., y Li, C. (2016). Daily reservoir inflow forecasting using multiscale deep feature learning with hybrid models. Journal of Hydrology, 532:193 – 206.

[Barrientos *et al*., 2007] Barrientos, A., Balaguer, L. F. P. C., y Aracil, R. (2007). Fundamentos de robótica. McGraw-Hill.

[Bengio, 2009] Bengio, Y. (2009). Learning deep architectures for AI. Found. Trends Mach. Learn., 2(1):1–127.

[Boyd y Vandenberghe, 2004] Boyd, S. y Vandenberghe, L. (2004). Convex Optimization. Cambridge University Press, USA.

[Buchner *et al*., 2012] Buchner, R., Wurhofer, D., Weiss, A., y Tscheligi, M. (2012). User experience of industrial robots over time. En 2012 7th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 115–116.

[Chen *et al*., 2018] Chen, L., Zhou, M., Su, W., Wu, M., She, J., y Hirota, K. (2018). Softmax regression based deep sparse autoencoder network for facial emotion recognition in human-robot interaction. Information Sciences, 428:49 – 61.

[Chen *et al*., 2014] Chen, X., Xiang, S., Liu, C., y Pan, C. (2014). Vehicle detection in satellite images by hybrid deep convolutional neural networks. IEEE Geoscience and Remote Sensing Letters, 11(10):1797–1801.

[Ciregan *et al*., 2012] Ciregan, D., Meier, U., y Schmidhuber, J. (2012). Multicolumn deep neural networks for image classification. En 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3642–3649.

[Cui *et al*., 2015] Cui, X., Goel, V., y Kingsbury, B. (2015). Data augmentation for deep neural network acoustic modeling. IEEE/ACM Transactions on Audio, Speech, and Language Processing, 23(9):1469–1477.

[Dairi *et al*., 2018] Dairi, A., Harrou, F., Senouci, M., y Sun, Y. (2018). Unsupervised obstacle detection in driving environments using deep-learningbased stereovision. Robotics and Autonomous Systems, 100:287 – 301.

[Daugherty y Wilson, 2018] Daugherty, P. R. y Wilson, H. J. (2018). Human + machine: reimagining work in the age of AI. Boston, Massachusetts: Harvard Business Review Press.

[Davis y Mermelstein, 1980] Davis, S. y Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4):357–366.

[Dong *et al*., 2016] Dong, Y., Liu, Y., y Lian, S. (2016). Automatic age estimation based on deep learning algorithm. Neurocomputing, 187:4 – 10. Recent Developments on Deep Big Vision.

[Dwivedi *et al*., 2014] Dwivedi, K., Biswaranjan, K., y Sethi, A. (2014). Drowsy driver detection using representation learning. En 2014 IEEE International Advance Computing Conference (IACC), pp. 995–999.

[Farooq *et al*., 2012] Farooq, U., Hasan, K. M., Asad, M. U., y Saleh, S. O. (2012). Fuzzy logic based wall tracking controller for mobile robot navigation. En 2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA), pp. 2102–2105.

[Gan *et al*., 2014] Gan, J., Li, L., Zhai, Y., y Liu, Y. (2014). Deep self-taught learning for facial beauty prediction. Neurocomputing, 144:295 – 303.

[Gonzalez y Woods, 2008] Gonzalez, R. y Woods, R. (2008). Procesamiento Digital de Imágenes. Prentice Hall.

[Guechi *et al*., 2012] Guechi, E., Abellard, A., y Franceschi, M. (2012). Experimental fuzzy visual control for trajectory tracking of a khepera II mobile robot. En 2012 IEEE International Conference on Industrial Technology, pp.25–30.

[Guo *et al*., 2017] Guo, D., Sun, F., Kong, T., y Liu, H. (2017). Deep vision networks for real-time robotic grasp detection. International Journal of Advanced Robotic Systems, 14(1).

[Guo *et al*., 2016] Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., y Lew, M. S. (2016). Deep learning for visual understanding: A review. Neurocomputing, 187:27 – 48. Recent Developments on Deep Big Vision.

[Gutiérrez *et al*., 2017] Gutiérrez, M. A., Manso, L. J., Pandya, H., y Núñez, P. (2017). A passive learning sensor architecture for multimodal image labeling: An application for social robots. Sensors (Basel), 17(2).

[Hossain *et al*., 2017] Hossain, D., Capi, G., y Jindai, M. (2017). Evolution of deep belief neural network parameters for robot object recognition and grasping. Procedia Computer Science, 105:153 – 158. 2016 IEEE International Symposium on Robotics and Intelligent Sensors.

[Hou *et al*., 2015] Hou, W., Gao, X., Tao, D., y Li, X. (2015). Blind image quality assessment via deep learning. IEEE Transactions on Neural Networks and Learning Systems, 26(6):1275–1286.

[Ji *et al*., 2014] Ji, N.-N., Zhang, J.-S., y Zhang, C.-X. (2014). A sparse-response deep belief network based on rate distortion theory. Pattern Recognition, 47(9):3179 – 3191.

[Jimenez Moreno, 2011] Jimenez Moreno, R. (2011). Sistema de detección de nivel de cansancio en conductores mediante técnicas de visión por computador. Tesis de máster, Universidad Nacional de Colombia.

[Jiménez Moreno et al., 2017] Jiménez Moreno, R., Avilés, O., y Ovalle, D. M. (2017). Evaluación de hiperparámetros en cnn para detección de patrones de imágenes. Visión electrónica, 11(2):140–145.

[Jiménez-Moreno *et al*., 2012] Jiménez-Moreno, R., Orjuela, S. A., Hese, P. V., Prieto, F. A., Grisales, V. H., y Philips, W. (2012). Video surveillance formonitoring driver's fatigue and distraction. En Optics, Photonics, and Digital Technologies for Multimedia Applications II, volumen 8436, pp. 263 – 270.

[Kalashnikov *et al*., 2018] Kalashnikov, D., Irpan, A., Pastor, P., Ibarz, J., Herzog, A., Jang, E., Quillen, D., Holly, E., Kalakrishnan, M., Vanhoucke, V., y Levine, S. (2018). Scalable deep reinforcement learning for vision-based robotic manipulation. En Proceedings of The 2nd Conference on Robot Learning, volumen 87, pp. 651–673. PMLR.

[Kiguchi y Hayashi, 2013] Kiguchi, K. y Hayashi, Y. (2013). Upper-limb tremor suppression with a 7DOF exoskeleton power-assist robot. En 2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 6679–6682.

[Kim *et al*., 2015a] Kim, S., Choi, Y., y Lee, M. (2015a). Deep learning with support vector data description. Neurocomputing, 165:111 – 117.

[Kim *et al*., 2015b] Kim, S., Yu, Z., Kil, R. M., y Lee, M. (2015b). Deep learning of support vector machines with class probability output networks. Neural Networks, 64:19 – 28. Special Issue on "Deep Learning of Representations".

[Kopman *et al*., 2015] Kopman, V., Laut, J., Acquaviva, F., Rizzo, A., y Porfiri, M. (2015). Dynamic modeling of a robotic fish propelled by a compliant tail. IEEE Journal of Oceanic Engineering, 40(1):209–221.

[Koumakis, 2020] Koumakis, L. (2020). Deep learning models in genomics; are we there yet? Computational and Structural Biotechnology Journal, 18:1466 – 1473.

[Lenz *et al*., 2015] Lenz, I., Lee, H., y Saxena, A. (2015). Deep learning for detecting robotic grasps. The International Journal of Robotics Research, 34(4- 5):705–724.

[Liu *et al*., 2015] Liu, H., Ma, B., Qin, L., Pang, J., Zhang, C., y Huang, Q. (2015). Set-label modeling and deep metric learning on person re-identification. Neurocomputing, 151:1283 – 1292.

[Lu *et al*., 2017] Lu, K., An, X., Li, J., y He, H. (2017). Efficient deep network for vision-based object detection in robotic applications. Neurocomputing, 245:31 – 45.

[Längkvist *et al*., 2014] Längkvist, M., Karlsson, L., y Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. Pattern Recognition Letters, 42:11 – 24.

[Moreno *et al*., 2013] Moreno, R. J., Espinosa, F. A., y Hurtado, D. A. (2013). Teleoperated systems: A perspective on telesurgery applications. Revista Ingeniería Biomédica, 7:30 – 41.

[Moreno y Lopez, 2013] Moreno, R. J. y Lopez, J. D. (2013). Trajectory planning for a robotic mobile using fuzzy c-means and machine vision. En Symposium of Signals, Images and Artificial Vision - 2013: STSIVA - 2013, pp. 1–4.

[Moreno et al., 2017] Moreno, R. J., Umaña, L. A. R., y Baquero, J. E. M. (2017). Virtual environment for robotic assistance. IJAER, 12(22):12315–12318.

[Moreno et al. , 2018] Moreno, R. J., Mauledoux, M., y Martinez, B. J. (2018). Algorithm for object grasp detection. Research Journal of Applied Sciences, 13:162–175.

[Murillo et al., 2018] Murillo, P. U., Jimenez, R., y Beleño, R. H. (2018). Algorithm for tool grasp detection. International Review of Mechanical Engineering (IREME), 12(1).

[Neukart y Moraru, 2014] Neukart, F. y Moraru, S.-A. (2014). A machine learning approach for abstraction based on the idea of deep belief artificial neural networks. Procedia Engineering, 69:1499 – 1508. 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2013.

[Ochiai *et al*., 2014] Ochiai, T., Matsuda, S., Lu, X., Hori, C., y Katagiri, S. (2014). Speaker adaptive training using deep neural networks. En 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6349–6353.

[Pachon-Suescun *et al*., 2020] Pachon-Suescun, C. G., Enciso-Aragon, C. J., y Jimenez-Moreno, R. (2020). Robotic navigation algorithm with machine vision. International Journal of Electrical and Computer Engineering (IJECE), 10.

[Palomares *et al*., 2016] Palomares, F. G., Serrá, J. A. M., y Martínez, E. A. (2016). Aplicación de la convolución de matrices al filtrado de imágenes. Modelling in Science Education and Learning, 9(1):97–108.

[Pan y Yang, 2010] Pan, S. J. y Yang, Q. (2010). A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359.

[Perconti y Plebe, 2020] Perconti, P. y Plebe, A. (2020). Deep learning and cognitive science. Cognition, 203:104365.

[Pinzon et al., 2018]. Pinzon, J. O., Jimenez-Moreno, R., Aviles, O., Nino, P., y Ovalle, D. (2018). Very deep convolutional neural network for speech recognition based on words. Journal of Engineering and Applied Sciences, 13:6680–6685.

[Pinzón *et al*., 2017] Pinzón, J., Jiménez, R., y Hernandez, R. (2017). Deep convolutional neural network for hand gesture recognition used for humanrobot interaction. Journal Of Engineering and Applied Sciences, 12:9278 – 9285.

[Pinzón-Arenas *et al*., 2019] Pinzón-Arenas, J., Jiménez-Moreno, R., y Pachón- Suescún, C. (2019). Handwritten word searching by means of speech commands using deep learning techniques. International Review on Modelling and Simulations (IREMOS), 12(4).

[Pinzón-Arenas y Jiménez-Moreno, 2020] Pinzón-Arenas, J. O. y Jiménez- Moreno, R. (2020). Comparison between handwritten word and speech record in real-time using CNN. International Journal of Electrical and Computer Engineering (IJECE), 10:4313–4321.

[Pramparo y Moreno, 2017] Pramparo, L. y Moreno, R. J. (2017). Colorimeter using artificial neural networks. Journal of Engineering and Applied Sciences, 12:5332–5337.

[Qian y Woodland, 2016] Qian, Y. y Woodland, P. C. (2016). Very deep convolutional neural networks for robust speech recognition. En 2016 IEEE Spoken Language Technology Workshop (SLT), pp. 481–488.

[Rioux-Maldague y Giguère, 2014] Rioux-Maldague, L. y Giguère, P. (2014). Sign language fingerspelling classification from depth and color images using a deep belief network. En 2014 Canadian Conference on Computer and Robot Vision, pp. 92–97.

[Salehi *et al*., 2020] Salehi, A. W., Baglat, p., y Gupta, G. (2020). Review on machine and deep learning models for the detection and prediction ofcoronavirus. Proceedings on Materials Today.

[Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61:85 – 117.

[Shang *et al*., 2014] Shang, C., Yang, F., Huang, D., y Lyu, W. (2014). Datadriven soft sensor development based on deep learning technique. Journal of Process Control, 24(3):223 – 233.

[Song *et al*., 2014] Song, I., Kim, H., y Jeon, P. B. (2014). Deep learning for real-time robust facial expression recognition on a smartphone. En 2014 IEEE International Conference on Consumer Electronics (ICCE), pp. 564–567.

[Tamilselvan y Wang, 2013] Tamilselvan, P. y Wang, P. (2013). Failure diagnosis using deep belief learning based health state classification. Reliability Engineering & System Safety, 115:124 – 135.

[Thulasiraman y Swamy, 1992] Thulasiraman, K. y Swamy, M. N. S. (1992). Graphs: Theory and Algorithms. John Wiley & Sons, Inc., USA.

[Useche et al., 2018] Useche, P. C., Beleno, R. D. H., y Moreno, R. J. (2018). Manipulation of tools by means of a robotic arm using artificial intelligence. Journal of Engineering and Applied Sciences, 13: 3479 – 3492.

[Velandia *et al*., 2019] Velandia, N. S., Moreno, R. J., y Rubiano, A. (2019). CNN architectures for hand gesture recognition using EMG signals throw wavelet feature extraction. Journal of Engineering and Applied Sciences, 14:3528–3537.

[Viola y Jones, 2001] Viola, P. y Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. En Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.

[Walid y Lasfar, 2014] Walid, R. y Lasfar, A. (2014). Handwritten digit recognition using sparse deep architectures. En 2014 9th International Conference on Intelligent Systems: Theories and Applications (SITA-14), pp.1–6.

[Wang y Morel, 2014] Wang, Y. y Morel, J. (2014). Can a single image denoising neural network handle all levels of gaussian noise? IEEE Signal Processing Letters, 21(9):1150–1153.

[Wang *et al*., 2016] Wang, Z., Li, Z., Wang, B., y Liu, H. (2016). Robot grasp detection using multimodal deep convolutional neural networks. Advances in Mechanical Engineering, 8(9).

[Weber, 2010] Weber, W. (2010). Automatic generation of the Denavit-Hartenberg convention. En ISR 2010 (41st International Symposium on Robotics) and ROBOTIK 2010 (6th German Conference on Robotics), pp. 1–7.

[Wu *et al*., 2014] Wu, K., Chen, X., y Ding, M. (2014). Deep learning based classification of focal liver lesions with contrast-enhanced ultrasound. Optik, 125(15):4057 – 4063.

[You *et al*., 2014] You, Z., Wang, X., y Xu, B. (2014). Exploring one pass learning for deep neural network training with averaged stochastic gradient descent. En 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6854–6858.

[Young *et al*., 2006] Young, S. J., Evermann, G., Gales, M., Hain, T., Kershaw, D., Moore, G. L., Odell, J. J., Ollason, D., Povey, D., Valtchev, y Woodland, P. C. (2006). The HTK book version 3.4.

[Zeiler y Fergus, 2014] Zeiler, M. D. y Fergus, R. (2014). Visualizing and understanding convolutional networks. En Computer Vision – ECCV 2014, pp. 818–833, Cham. Springer International Publishing.

[Zhang y Zhang, 2014] Zhang, C. y Zhang, Z. (2014). Improving multiview face detection with multi-task deep convolutional neural networks. En IEEE Winter Conference on Applications of Computer Vision, pp. 1036–1041.

[Zhang *et al*., 2020] Zhang, H., Hongyu, L., Nyayapathi, N., Wang, D., Le, A., Ying, L., y Xia, J. (2020). A new deep learning network for mitigating limitedview and under-sampling artifacts in ring-shaped photoacoustic tomography. Computerized Medical Imaging and Graphics.

[Zhang *et al*., 2015a] Zhang, Y., Li, X., Zhang, Z., Wu, F., y Zhao, L. (2015a). Deep learning driven blockwise moving object detection with binary scene modeling. Neurocomputing, 168:454 – 463.

[Zhang *et al*., 2015b] Zhang, Y., Li, X., Zhang, Z., Wu, F., y Zhao, L. (2015b). Deep learning driven blockwise moving object detection with binary scene modeling. Neurocomputing, 168:454 – 463.

Apéndice A

Ajuste de las Arquitecturas de las CNN

En este apéndice se presentan una serie de pruebas que permiten evidenciar las variaciones de hiper-parámetros para entrenamiento de redes neuronales convolucionales [Jiménez Moreno et al., 2017]. La operación de convolución implica un volumen fijo de entrada y un tamaño de filtro fijo. De igual manera, esto significa que para el entrenamiento de la red se debe establecer una base de datos de imágenes, que contengan el objeto de aprendizaje, u objetos de aprendizaje por categoría (cada objeto). Si bien cada categoría puede tener un diferente número de imágenes y cada imagen un diferente tamaño, a la entrada de la red deben ser redimensionadas de forma uniforme. Esta operación de redimensionamiento puede implicar variaciones de las características de aprendizaje. Por ejemplo, si la imagen es de un tamaño considerable en píxeles (superior a un Megapíxel), el redimensionamiento perderá resolución de la imagen, este efecto empeora si la imagen no es cuadrada. De forma que, las dimensiones de la imagen de entrada de la red, ya con el respectivo redimensionamiento, hacen parte de los parámetros a determinar y afectan el costo computacional.

Otro aspecto relevante de la base de datos es la cantidad de imágenes. Mientras mayor sea esta, mejor podrán establecerse los filtros de convolución. Esta base de datos debe repartirse en un grupo de entrenamiento y otro de validación, en relación promedio de $60/40$ a $80/20$. Donde el tiempo de entrenamiento tiene como uno de los parámetros de incremento el tamaño de la base de datos a emplear, de forma proporcional. Del conjunto de imágenes de entrenamiento, se emplea un subconjunto de imágenes que se utiliza para evaluar el gradiente de la función de pérdida y actualizar los pesos, denominado *mini-batch*, el tamaño de dicho subconjunto puede configurarse como uno de los parámetros de la red.

Debido a la multiplicidad de combinaciones que se pueden obtener de las diferentes variaciones de los parámetros de entrenamiento, solo se tomarán las más relevantes para poder apreciar su incidencia en la determinación de una arquitectura óptima de clasificación. A continuación, se exponen dichas variaciones sometidas al entrenamiento. En primera instancia, mediante dos equipos de computo de similares características. Pero, uno empleando procesamiento por CPU y el otro por GPU. Ambos equipos se caracterizan por poseer un procesador Intel core $\mathrm{i}7$ de séptima generación y 16 GB de memoria RAM, donde el equipo con GPU cuenta con una tarjeta NVIDIA 1050 de 4 GB de memoria. En las Tablas **A-1 - A-4**, se puede validar el efecto de emplear dos bases de datos de imágenes redimensionadas a escalas diferentes, sometidas a dos tipos de redes con variaciones de hiper-parámetros.

En las Tablas **A-1 y A-2**, se observan las mismas combinaciones de red, sometidas a las mimas bases de datos de entrada, pero validando las variaciones de entrenamiento basados en cambios del equipo de procesamiento. La diferencia fundamental se halla en el tiempo empleado, donde el consumo de la CPU se hace notorio ralentizando procesos adicionales al entrenamiento y, como se evidencia bajo tiempos prolongados, implicando un costo computacional elevado. Las pequeñas diferencias en la eficiencia de la red entrenada se dan por las variaciones propias de un entrenamiento particular bajo el random de imágenes que emplea en la separación de los grupos de entrenamiento y validación. La arquitectura de

la red 1 consta de 2 capas de convolución-relu-pooling, mientras que la red 2 consta de 3 de estas capas, donde la notación N-I en las tablas, alude a No Implementada, denotando la diferencia entre ambas redes. Se puede apreciar cómo el entrenamiento con imágenes de mayor tamaño entrega mayor información de aprendizaje a la red. Sin embargo, para el caso empleando un incremento en la profundidad de la red no resulta significativamente mejor.

Tabla A-1: Entrenamiento por CPU Learning Rate 0.001.

	BASE IMÁGENES 64X64				BASE IMÁGENES 128X128			
	RED 1		RED 2		RED 1		RED 2	
Type	Kernel	Filtros	Kernel	Filtros	Kernel	Filtros	Kernel	Filtros
Convolution/ReLu	5x5	30	5x5	30	5x5	30	5x5	30
MaxPooling	3x3	-	3x3	-	3x3	-	3x3	-
Convolution/ReLu	3x3	50	3x3	50	3x3	50	3x3	50
MaxPooling	2x2	-	2x2	-	2x2	-	2x2	-
Convolution/ReLu	N-I	N-I	2x2	10	N-I	N-I	2x2	10
MaxPooling	N-I	N-I	2x2	-	N-I	N-I	2x2	-
Full-Connected	4	-	4	-	4	-	4	-
Softmax	4	-	4	-	4	-	4	-
Tiempo de entrenamiento	6,15 Horas		14,5 Horas		9,45 Horas		16,12 Horas	
Precisión	78 %		76 %		81,2 %		82,6 %	

Tabla A-2: Entrenamiento por GPU Learning Rate 0.001.

	BASE IMÁGENES 64X64				BASE IMÁGENES 128X128			
	RED 1		RED 2		RED 1		RED 2	
Type	Kernel	Filtros	Kernel	Filtros	Kernel	Filtros	Kernel	Filtros
Convolution/ReLu	5x5	30	5x5	30	5x5	30	5x5	30
MaxPooling	3x3	-	3x3	-	3x3	-	3x3	-
Convolution/ReLu	3x3	50	3x3	50	3x3	50	3x3	50
MaxPooling	2x2	-	2x2	-	2x2	-	2x2	-
Convolution/ReLu	N-I	N-I	2x2	10	N-I	N-I	2x2	10
MaxPooling	N-I	N-I	2x2	-	N-I	N-I	2x2	-
Full-Connected	4	-	4	-	4	-	4	-
Softmax	4	-	4	-	4	-	4	-
Tiempo de entrenamiento	0,15 Horas		0,21 Horas		0,22 Horas		0,35 Horas	
Precisión	78,42 %		77,03 %		81,6 %		84,87 %	

La Tabla **A-3** se articula con la Tabla **A-2**, al evidenciar las variaciones de los hiper-parámetros y profundidad de la red, variando la razón de aprendizaje (learning rate) para observar su efecto. El primer cambio notorio es una reducción del tiempo de aprendizaje, pero la eficiencia de discriminación de categorías disminuye. Incluso una prueba con este valor en 0.005, buscando una media entre los dos casos, evidencia tiempos y eficiencias

intermedias entre las tabuladas. Donde, frente al costo computacional, es preferible dedicar más tiempo al entrenamiento bajo este parámetro.

Tabla A-3: Entrenamiento por GPU Learning Rate 0.01.

	BASE IMÁGENES 64X64				BASE IMÁGENES 128X128			
	RED 1		RED 2		RED 1		RED 2	
Type	Kernel	Filtros	Kernel	Filtros	Kernel	Filtros	Kernel	Filtros
Convolution/ReLu	5x5	30	5x5	30	5x5	30	5x5	30
MaxPooling	3x3	-	3x3	-	3x3	-	3x3	-
Convolution/ReLu	3x3	50	3x3	50	3x3	50	3x3	50
MaxPooling	2x2	-	2x2	-	2x2	-	2x2	-
Convolution/ReLu	N-I	N-I	2x2	10	N-I	N-I	2x2	10
MaxPooling	N-I	N-I	2x2	-	N-I	N-I	2x2	-
Full-Connected	4	-	4	-	4	-	4	-
Softmax	4	-	4	-	4	-	4	-
Tiempo de entrenamiento	0,134 Horas		0,202 Horas		0,2092 Horas		0,312 Horas	
Precisión	74,2 %		71,15 %		79,32 %		80,51 %	

De las pruebas realizadas, se establece como arquitectura base la red 2, con una entrada dimensionada a $128x128$ y eficiencia de $84{,}87\%$, según la Tabla **A-2**, debido a que es la que mejor desempeño presenta en relación al costo computacional y flexibiliza así mayor número de pruebas.

La Tabla **A-4** ilustra el cambio del número de filtros por cada una de las tres capas de convolución para esta arquitectura, evidenciando una mejora en la eficiencia hasta del $96{,}32\%$. Dicho cambio debe obedecer a un balance de la información a aprender de la capa anterior, se observa que un uso excesivo de filtros, degradan el desempeño final.

Tabla A-4: Variación filtros de la arquitectura CNN escogida.

	Prueba 1	Prueba 2	Prueba 3	Prueba 4
CNN 1	5x5/ 70 filtros	5x5/ 70 filtros	5x5/ 70 filtros	5x5/ 70 filtros
CNN 2	5x5/ 70 filtros	5x5/ 70 filtros	5x5/ 70 filtros	5x5/ 70 filtros
CNN 3	5x5/ 70 filtros	5x5/ 70 filtros	5x5/ 70 filtros	5x5/ 70 filtros
Precisión	78 %	87,6 %	89,16 %	96,32 %

Se observó que la incidencia del equipo de computo se hace relevante para el entrenamiento de la red y no para su uso en la labor de clasificación. Siendo así indiferente para una aplicación si se cuenta o no con una GPU. Sin embargo, se tiene que para poder iterar los parámetros de entrenamiento de la red a fin de optimizar la arquitectura a emplear, aún cuando

se parta de una base de datos adecuada, según las consideraciones aquí establecidas, los tiempos de convergencia demuestran la ventaja evidente del uso de la GPU. Se logró establecer que las variaciones en el tamaño de los filtros de convolución están relacionadas con las dimensiones del volumen de entrada. De forma que, el aprendizaje de características relevantes de una categoría dada, mejora mientras mas información se obtenga de esta. La razón de información esta determinada por el aumento del tamaño de los volúmenes en cada capa, que obedece a un aumento gradual del aprendizaje que cada filtro lleva a la siguiente capa. Para validar las pruebas presentadas es necesario realizar cambios paulatinos de un solo parámetro a la vez, que permitan el correcto análisis del efecto que causa su variación en el entrenamiento. Mediante iteraciones múltiples, se evidencian las relaciones finales que permiten converger más rápidamente en una red eficiente. Pero, se destaca la importancia de una base de datos adecuada, es decir, que genere patrones claros de aprendizaje.

Apéndice B

CNN para Reconocimiento de Comandos de Voz

Un complemento necesario para un robot asistencial, como el planteado, es el poder recibir la orden de entrega de una herramienta de forma natural, por ejemplo un comando de voz [Pinzon et al., 2018]. Para aplicaciones de reconocimiento de habla no es sencillo implementar una arquitectura tipo red neuronal convolucional. A diferencia de las arquitecturas creadas para reconocimiento de imágenes, en las que se puede tener una idea de la dimensión del objeto a reconocer, se puede iniciar con kernels de filtros que permitan identificar las características generales de dichos objetos, en el caso de reconocimiento de habla, los patrones que se quieren reconocer no son evidentes, cada combinación de palabras exhibe características particulares.

Con el fin de construir la arquitectura de la red y efectuar su entrenamiento, se construye una base de datos con las tres palabras de las que se quieren reconocer: bisturí, destornillador y tijeras. Dicha base de datos consta de 155 grabaciones por palabra de diferentes usuarios, con una duración de 2 segundos por grabación. De éstas, se toman 125 audios por categoría para entrenamiento y 30 para validación, obteniendo un total de

375 audios de entrenamiento y 90 de prueba. Cada audio es adquirido con una frecuencia de muestreo de 16000 Hz.

Debido a que las redes neuronales convolucionales son especializadas en reconocimiento de patrones, el encontrar una característica en una señal pura no se hace tan sencillo, ni evidente para la red. Por tal motivo, se requiere convertir la señal a una entrada más adecuada para guiar a la red en su aprendizaje. Para esto, se hace una extracción de características de cada señal de audio, con el fin de obtener un mapa que permite ver el comportamiento de la señal a través del tiempo en diferentes frecuencias. Esta extracción se realiza por medio de los MFCC (Mel-frequency cepstral coefficients) [Davis y Mermelstein, 1980], los cuales han tenido un amplio uso en los sistemas de análisis del habla.

Para efectuar la extracción de características, en primer lugar, se realiza un preénfasis de la señal, pasándola por un filtro de primer orden [Young *et al*., 2006], como

$$S_n' = S_n + \alpha\, S_{n-1}, \tag{B-1}$$

aplicando un coeficiente $\alpha = 0{,}97$, siendo S_n la señal original y S'_n la señal filtrada. Un ejemplo de este proceso, se muestra en la Figura **B-1**, para la palabra Bisturí.

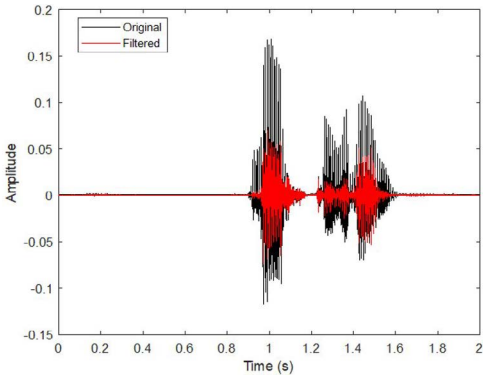


Figura B-1: Señal de voz para Bisturí.

Una vez hecho el filtrado, se realiza el *framing* y *windowing*, con el fin de arreglar las muestras en frames y atenuar las discontinuidades de la señal, por medio de

$$S'_{fn} = \left[0{,}54 - 0{,}46\,\cos\left(\frac{2\,\pi\,(n-1)}{N-1}\right)\right] S'_n. \qquad \text{(B-2)}$$

Cabe resaltar que cada frame tiene una duración de 20 ms y es muestreado cada 10 ms. La transformación se aplica a las muestras $\{S'_n\,,\,_n = 1{,}N_f\}$ [12], siendo N_f el número de frames. Seguido de esto, se obtiene el espectro de magnitudes, por medio de la Transformada Rápida de Fourier de cada frame, con una longitud de $N = 512$. Ya que cada frame contiene 320 muestras, se debe usar la siguiente potencia de 2 de la cantidad de muestras. Para encontrar las magnitudes, utilizamos

$$MAG = \left|\sum_{n=1}^{N} S'_{fn}(n) e^{-i\,2\,\pi\,n\,k/N}\right|, \qquad k = 0, \ldots, N-1. \qquad \text{(B-3)}$$

Con el fin de usar los MFCC, las frecuencias deben ser trabajadas en la escala de Mel, definida como

$$f_{mel} = 1127\,\ln(1 + f/700), \qquad \text{(B-4)}$$

donde f es la frecuencia en Hz. Con esta definición, se crea un banco de filtros triangulares separados uniformemente, con frecuencias de corte en escala de Mel dadas por

$$c_{mel}(m) = f_{low_{mel}}\, m\, \frac{f_{high_{mel}} - f_{low_{mel}}}{M+1}, \quad m = 0, \ldots, M+1, \qquad \text{(B-5)}$$

donde el límite inferior de frecuencia, $f_{low_{mel}}$, es de 300 Hz en escala Mel, y el límite superior de frecuencia, $f_{high_{mel}}$, es de 8000 Hz en escala Mel. Los rangos de frecuencias, se pueden encontrar como

$$F(k) = f_{\text{mín}} + k\,\frac{f_{\text{máx}} - f_{\text{mín}}}{K-1}, \quad k = 0, \ldots, K-1, \qquad \text{(B-6)}$$

donde $f_{\text{mín}}$ representa el rango inferior (0 Hz), $f_{\text{máx}}$ el rango superior (8000 Hz), M la cantidad de canales del filtro, cuyo valor es 20, K es la longitud de la respuesta en frecuencia, es decir $\mathrm{K} = (1 + \mathrm{N})/2$. Finalmente, el banco de filtros queda definido como

$$H(m,k) = \begin{cases} 0, & F(k) < c_{Hz}(m), \\ \dfrac{F(k) - c'(m)}{c_{Hz}(m+1) + c_{Hz}(m)}, & c_{Hz}(m) \leq F(k) \leq c_{Hz}(m+1), \\ \dfrac{c_{Hz}(m+2) - F(k)}{c_{Hz}(m+2) + c_{Hz}(m+1)}, & c_{Hz}(m+1) \leq F(k) \leq c_{Hz}(m+2), \\ 0, & F(k) > c_{Hz}(m+2). \end{cases} \tag{B-7}$$

Una vez diseñados los filtros triangulares, estos son aplicados a la sección única del espectro de magnitudes, acorde a

$$Fbe = H(m,k)\, MAG(k), \qquad k = 0, ..., K-1, \tag{B-8}$$

obteniendo como resultado las energías de los filtros, para luego, computarla con la transformada discreta del Coseno, mientras se le aplica el logaritmo a todos filtros, obteniendo los coeficientes cepstrales (Cc) [Young *et al.*, 2006]

$$Cc_i = \sqrt{\frac{2}{M}} \sum_{j_1}^{M} \log(Fbe_j)\, \cos\left(\frac{\pi i}{M}(j - 0{,}5)\right), \quad i = 0, ..., C. \tag{B-9}$$

Para este caso se tomó un numero de $C = 12$ coeficientes cepstrales por cada frame, obteniendo unas características generales dadas por la figura **B-2.**

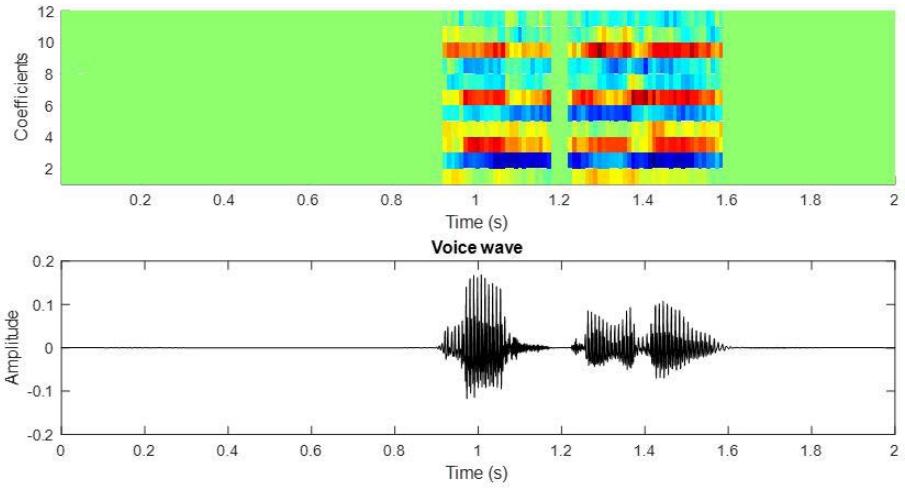


Figura B-2: Mapa de Características MFCC

Para el reconocimiento del habla, el uso de otros parámetros ayuda a incrementar el desempeño del sistema. Por lo cual, se adicionan la primera derivada (Cc') y segunda derivada (Cc') de los coeficientes con respecto al tiempo. Para esto, se aplica la primera derivada de Cc_i, dada por

$$Cc_i^{'} = \frac{\sum_{n=1}^{N} n(Cc'_{t+n} - Cc'_{t-n})}{2\sum_{n=1}^{N} n^2}, \quad t = 1, ..., N_f, \tag{B-10}$$

y la segunda derivada de Cc_i dada por

$$Cc^{''} = \frac{\sum_{n=1}^{N} n(Cc'_{t+n} - Cc'_{t-n})}{2\sum_{n=1}^{N}} n^2, \quad t = 1, ..., N_f \tag{B-11}$$

con un valor de análisis N=1 para este caso, obteniendo finalmente los mapas de características mostrados en la Figura **B-3.**

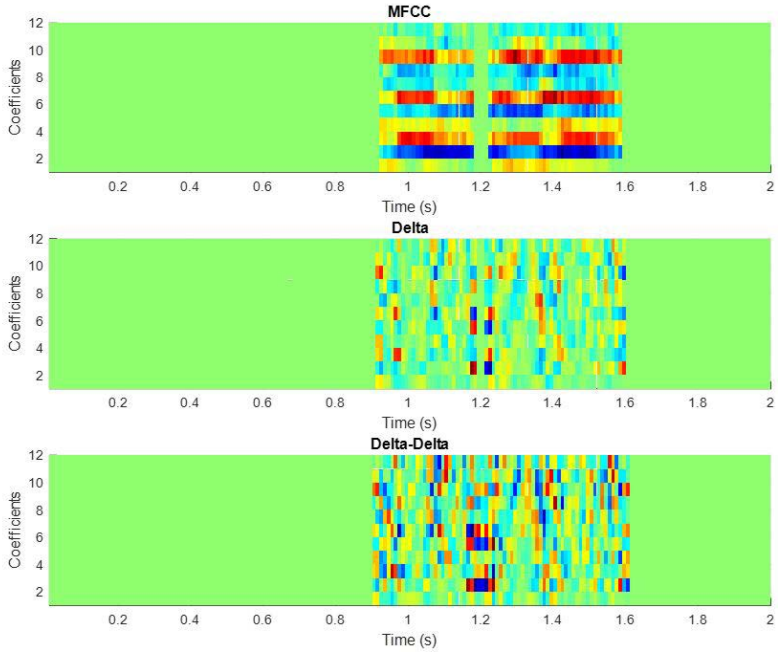


Figura B-3: Mapas de Características MFCC y derivadas.

Los mapas de características son ordenados en un arreglo matricial de $12 \times 199 \times 3$. Es decir, una matriz de 12 coeficientes adquiridos de 199 frames, con sus respectivas primera y segunda derivada. Esto se aplica a todas las grabaciones realizadas y, de esta forma, se construye la base de datos que se va a usar para ser ingresada en la red neuronal convolucional.

Se propone una arquitectura basada en filtros cuadrados (ver Tabla **B-1**). De forma tal que se puedan extraer características combinadas entre tiempo y coeficientes, permitiendo a la red aprender el comportamiento en las dos dimensiones. Adicionalmente, aunque en la mayoría de los trabajos no se agrega padding a las convoluciones [Qian y Woodland, 2016], en éste si se usa, con el fin de que el tamaño original del volumen de entrada se mantenga a la salida de cada capa.

Tabla B-1: Arquitectura CNN.

Vol. Entrada			Capa	Kernel				Filtros
H	W	D		M	N	S	P	
-	-	-	Entrada	12	199	-	-	-
12	199	3	Convolution	5	5	1	2	32
12	199	32	Convolution	5	5	1	2	32
12	199	32	MaxPooling	2	1	2	0	-
6	199	32	Convolution	3	3	1	1	64
6	199	64	Convolution	3	3	1	1	64
6	199	64	MaxPooling	2	3	2	0	-
3	99	64	Convolution	2	2	1	1	128
4	100	128	Convolution	3	3	1	1	128
4	100	128	MaxPooling	4	2	2	0	-
2	50	128	Fully-Connected	1	1	-	-	512
-			Dropout					-
512			Fully-Connected	1				2048
-			Dropout					-
2048			Fully-Connected	1	1	-	-	3
-			SoftMax	3		-	-	-

Es importante notar que, en el primer MaxPooling, se realiza el downsampling únicamente en el espacio de los MFCC. Principalmente, para que haya una reducción en los coeficientes. Pero, se mantengan los frames para la extracción. Mientras que en los siguientes, si se realiza tanto en el tiempo, como en los coeficientes. Cabe resaltar que, ya que se quiere hacer el reconocimiento completo de cada palabra, diferente a como normalmente se realiza (por medio de fonemas), la entrada de la red es el arreglo matricial completo obtenido de los mapas de características.

Se realiza el entrenamiento de la arquitectura propuesta usando el conjunto de datos construido por un total de 600 épocas. Según el comportamiento obtenido durante el entrenamiento (ver Figura **B-4**), la red tuvo un espacio de dificultad en su aprendizaje, debido a la dificultad de adquirir los datos del mapa de características. Sin embargo, aproximadamente en la época 170, la red empezó a tener una curva de aprendizaje, alcanzando una exactitud de entrenamiento de más del 90% en la época 238, hasta lograr una estabilización del 100% en la época 568. Con el fin de tener una mayor confiabilidad en el reconocimiento, se validaron las 100 últimas épocas, obteniendo los dos mejores desempeños en la época 500 y 511, con el 97.8% y 98.9% de exactitud de validación, respectivamente. Por consiguiente, para certificar que en épocas anteriores no haya un mejor desempeño, ya que cabe la posibilidad que la red haya entrado a overfitting cuando la red se estabiliza en 100% (empieza a memorizar mas no a aprender), se realizó una nueva validación de las épocas donde su precisión de entrenamiento fue superior al 95%, obteniendo que, en la época 308, se tiene una exactitud de validación del 100%, superando a las dos mejores épocas encontradas anteriormente, por lo cual se toma como red final dicha época.

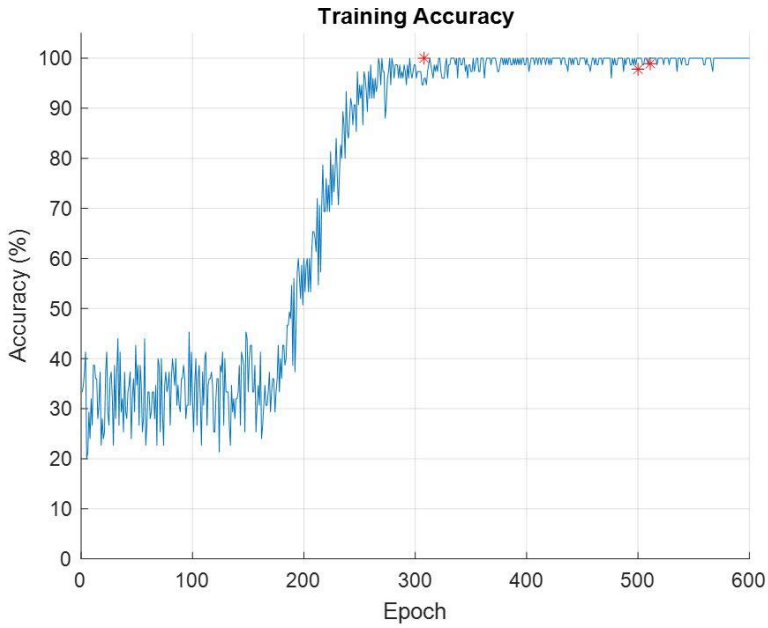


Figura B-4: Desempeño del entrenamiento de la red.

La nueva arquitectura de una red neuronal convolucional enfocada al reconocimiento de voz por medio de palabras, en función a la extracción de características de los audios obtenidos por medio de las MFCC, permitió obtener los datos de las palabras completas para ser ingresadas a la red, la cual logró un 100% de exactitud en la validación.

Impreso en papel bond 90 gr.
en familia tipográfica Candara a 11,5 pts.

Amadgraf Impresores Ltda.
Bogotá, D.C., Colombia
Octubre de 2020.

OTROS TÍTULOS DE ESTA COLECCIÓN

RADIACIÓN-MATERIA: GEANT4 HANDS ON!

REDES NEURONALES CONVOLUCIONALES USANDO KERAS Y ACELERANDO CON GPU

GESTIÓN DE LA ENERGÍA: EL USUARIO DE ENERGÍA COMO PARTE ACTIVA DEL SISTEMA

GESTIÓN Y CIBERSEGURIDAD PARA MICRORREDES ELÉCTRICAS RESIDENCIALES

DETECCIÓN Y CORRECCIÓN DE PROPAGACIONES ANÓMALAS EN RADARES METEREOLÓGICOS

INTRODUCCION A LA INVESTIGACIÓN SOBRE DESASTRES NATURALES Y CIUDADES INTELIGENTES

INVESTIGACIÓN EN INGENIERÍA FUNDAMENTADA EN LA GESTIÓN DEL CONOCIMIENTO

LOS RECURSOS DISTRIBUIDOS DE BIOENERGÍAEN COLOMBIA

Este libro aborda temáticas actuales pertinentes al desarrollo de la industria 4.0, donde las técnicas de inteligencia artificial y el uso de sistemas robóticos propenden por mejorar la calidad de vida de los seres humanos. Se presenta el desarrollo de arquitecturas paralelas de redes neuronales convolucionales que, en el marco del aprendizaje profundo, hoy día presentan amplios desarrollos dada la robustez que poseen en el reconocimiento de patrones. Dentro de las muchas posibilidades que se encuentran en las arquitecturas paralelas, se propone un aprendizaje independiente por rama, cada una con una arquitectura propia, donde la arquitectura final puede presentarse de forma híbrida. El problema abordado consiste en un robot asistencial que es capaz de reconocer una herramienta de entre un grupo, tomarla y entregarla a un humano, modificando su trayectoria para evitar colisionar con la mano del mismo. Para la aplicación presentada, se desarrolla una red neuronal convolucional paralela con integración difusa en la capa de salida, que es comparada con el uso de una ponderación aritmética basada en la distancia de captación del objeto, para determinar los beneficios de cada una. Además del problema de identificación de la herramienta a distancias variables, se expone un algoritmo de generación de trayectorias con evasión de obstáculos y un algoritmo orientado al agarre de la herramienta. Se plantea un entorno híbrido real-virtual, en el que se verifica la funcionalidad de la red diseñada, así como algunos aspectos por mejorar.

ISBN 978-958-787-237-8

9 789587 872378