Universidad de Sevilla

Escuela Politécnica
Superior de Sevilla

Trabajo Fin de Grado en Ingeniería Electrónica Industrial

Prototipo de control brazo robótico industrial con detección de objetos basado en Deep-Learning

Autor: Roque Sánchez Ferrera

Tutores: Alejandro Linares Barranco y Enrique Piñero Fuentes

Fecha de Presentación: Julio 2023

Trabajo Fin de Grado

Grado en Ingeniería Electrónica Industrial

PROTOTIPO DE CONTROL BRAZO ROBÓTICO INDUSTRIAL CON DETECCIÓN DE OBJETOS BASADO EN DEEP-LEARNING

Autor:

Roque Sánchez Ferrera

Tutores:

Alejandro Linares Barranco

Enrique Piñero Fuentes

Departamento de Arquitectura y Tecnología de Computadores

Escuela Politécnica Superior

Universidad de Sevilla

Sevilla, 2023

RESUMEN

Este proyecto ha surgido con el fin de trabajar en el área de la visión artificial y, al mismo tiempo, emplear la tecnología Deep Learning. De esta manera se ha investigado para llegar a controlar un robot industrial.

Por lo tanto, se ha realizado un prototipo que mediante visión artificial sea capaz de detectar objetos que después serán clasificados por el robot Scorbot ER-4u.

PALABRAS CLAVES

Visión artificial, Deep Learning, Robótica Industrial, Robot, Calibración, Python.

ABSTRACT

This project has been developed with the aim of working in the field of computer vision and, at the same time, using Deep Learning technology. In this way, research has been carried out to control an industrial robot.

Therefore, a prototype has been created that by means of computer vision is capable of detecting objects which will later be classified by the Scorbot ER-4u robot.

KEYWORDS

Computer vision, Deep Learning, Industrial Robotics, Robot, Calibration, Python.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ECUACIONES

ÍNDICE DE TABLAS

1 INTRODUCCIÓN.

La evolución de la tecnología ha llevado consigo un importante crecimiento en el sector industrial. Desde 1801, con la invención de la que se conoce como la primera máquina programable, el telar; pasando por un acontecimiento tan importante como la revolución francesa; hasta la actualidad con el desarrollo de la tecnología 4.0 (Big Data, el internet de las cosas y la inteligencia artificial).

Este crecimiento ha provocado que las empresas del sector industrial tiendan a innovar su tecnología de uso. Es aquí donde aparecen los tres grandes temas que se pretenden plasmar en este documento: la visión artificial, el Deep Learning y la robótica industrial.

Cuando se emplea la robótica industrial, se trabaja en un campo que ofrece beneficios que otros no. Por un lado, aumenta la seguridad y protección de los trabajadores y disminuye el esfuerzo a aplicar. Además, la robótica industrial permite aumentar la producción y rentabilidad, reduciendo el tiempo de inactividad. También, evita realizar tareas repetitivas y previene el riesgo que se pudiera ocasionar a las piezas, productos o personas. Por otro lado, uno de los grandes beneficios que brinda la robótica industrial es la mejora de la calidad del producto a fabricar.

Por otra parte, la visión artificial ha adquirido actualmente una gran relevancia en el sector industrial. Pero no sólo en este campo sino también en otros como el sector de la automoción, en la electrónica y en el sector agrícola. En una última noticia de la actualidad se relata la revolución de la inteligencia artificial en este último sector. Conocer el número de frutos que hay en un árbol puede ser un dato muy importante y para ello la empresa Agerpix ha desarrollado un invento que, mediante visión artificial, inteligencia artificial y una serie de sensores es capaz de saber cuánta fruta hay en un campo o cuántos frutos en un árbol (El Mundo, 2023). Por lo tanto, la visión e inteligencia artificial ofrece soluciones a enigmas importantes que aparecen en nuestro día a día.

Pero no sólo estas dos tecnologías, sino que también el Deep Learning ha alcanzado una notable relevancia en diferentes áreas. Según un estudio publicado por The Lancet Digital Health, recientemente se ha desarrollado un modelo de aprendizaje profundo que detecta con precisión la función cardiaca y enfermedad valvular mediante radiografías de tórax.

Así pues, durante el desarrollo de este proyecto se ha considerado importante llevar a cabo el estudio de estas tres tecnologías implementando un prototipo capaz de detectar y clasificar objetos haciendo uso de la visión artificial, el Deep Learning y la robótica industrial.
Para poder capturar el entorno de trabajo se empleará la cámara *Webcam Pro-9000* de Logitech y para la clasificación de los objetos, el robot *Scorbot ER-4u* de Intelitek Inc. La detección de objetos será posible gracias a YOLOv3 que dispone de modelos pre-entrenados capaces de reconocer hasta 80 clases de objetos.

Mediante un programa realizado en Python se conseguirá obtener a través de la cámara una imagen totalmente calibrada (sin signos de distorsión). Se logrará detectar y recortar una determinada área de trabajo y sobre esta detectar los distintos objetos.

Sobre esta detección se investigará para obtener una relación entre los píxeles de la imagen y las coordenadas reales del objeto. Además, será necesario obtener la orientación (ángulo) que presente.

Una vez establecidos estos parámetros se hará uso de Matlab y una toolbox de control robótico que permitirá configurar las articulaciones del Scorbot ER-4u para llegar a una posición definida anteriormente y clasificar los objetos en su correspondiente depósito.

Realizado este proceso se habrá alcanzado el objetivo experimental del proyecto puesto que se dispondrá de un prototipo capaz de detectar y clasificar objetos. Además, se habrán consolidado conocimientos a nivel de programación en Python usando librerías como OpenCV y Numpy. E incluso en Matlab, donde se aprenderá el uso de librerías y poder utilizarlas desde un entorno de programación distinto. Igualmente, se habrá adquirido un gran conocimiento de las tres grandes tecnologías que definen el proyecto: visión artificial, robótica industrial y Deep Learning.

2 OBJETO, ALCANCE Y JUSTIFICACIÓN DEL PROYECTO.

2.1 OBJETO DEL PROYECTO.

El siguiente proyecto tiene como objetivo implementar un prototipo de control de brazo robótico con detección de objetos basado en Deep-Learning. Una cámara detectará objetos en una determinada área de trabajo, obtendrá el ángulo de ataque más adecuado para agarrarlo y un brazo robótico se encargará de clasificarlos en su correspondiente depósito.

2.2 ALCANCE DEL PROYECTO.

El proyecto consta fundamentalmente de dos partes:

- Proceso de visión artificial.

- Programación del robot Scorbot ER-4u.

Dentro del proceso de visión artificial se llevarán a cabo las siguientes tareas:

- Calibrado de la cámara: se desarrollará un programa capaz de obtener capturas con la cámara de un tablero de ajedrez con las que se conseguirán unos datos que permitirán eliminar la distorsión que presenta la cámara.

- Definición del área de trabajo: se definirá un área de trabajo que pueda ser capturado por la cámara y esté dentro del volumen de trabajo del robot. Para ello se desarrollará un programa de detección de esquinas que permita detectar cada esquina del área de trabajo y especificar las coordenadas x e y en píxeles de cada una para el posterior recorte de la imagen capturada.

- Relación píxeles/centímetros: para determinar en qué posición se encuentra cada objeto en el mundo real es necesario conocer la relación entre píxeles y centímetros que existe. Para ello será necesario conocer las distancias reales del área de trabajo y las distancias en píxeles de esta misma área.

- Detección de objetos: se llevará a cabo el desarrollo de un programa capaz de detectar objetos en una determinada área. Para ello se hará uso de Yolo-v3.

- Cálculo de la orientación más adecuada para agarrar el robot: puesto que el robot presenta como elemento terminal una pinza, para que este pueda coger de forma correcta el objeto, es necesario conocer la orientación de este dentro del área de trabajo.

En el módulo de programación del robot se llevarán a cabo las siguientes tareas:

- Inicialización del servidor: configurar la comunicación mediante puerto USB con la controladora del robot. Además, carga los DLLs y habilita el control del robot.

- Ejecutar Home del robot: se debe realizar el Home del robot para que se inicie desde su posición inicial.

- Cálculo de las variables necesarias para ejecutar los movimientos del robot: al igual que la cámara y el mundo real, el robot también debe tener una relación que permita pasar de coordenadas reales a coordenadas del robot. Estas coordenadas serán pasadas a una función que se encargará de mover el propio robot.

- Clasificación del objeto: conocidas las variables y la posición a la que debe ir el robot para coger o dejar el objeto, el último paso consistirá en clasificar los objetos, cada uno en su depósito cuya posición es conocida.

2.3 JUSTIFICACIÓN DEL PROYECTO.

Con este proyecto se pretende realizar un prototipo capaz de llevar a cabo un proceso de clasificación de objetos conociendo paso a paso la implementación que conlleva un proceso de visión artificial y la programación de robots en el ámbito industrial.

3 MARCO NORMATIVO LEGAL.

3.1 ÁMBITO DE LA ROBÓTICA INDUSTRIAL.

3.1.1 LEGISLACIÓN Y NORMAS EUROPEAS.

- Directiva de máquinas (2006/42/CE): establece los requisitos de seguridad y salud para las máquinas, incluyendo los robots industriales, que se comercializan en la Unión Europea. Define los deberes y responsabilidades de los fabricantes y distribuidores, y aborda aspectos como el diseño seguro, la información y marcado CE, la evaluación de conformidad y el uso seguro de las máquinas.

- Directiva de Baja Tensión (2014/35/UE): se aplica a los equipos eléctricos y aborda los requisitos de seguridad eléctrica.

- Reglamento REACH (1907/2006/CE): regula la producción, importación y uso de sustancias químicas en la Unión Europea. En el ámbito de la robótica industrial, puede ser relevante para los materiales y sustancias químicas utilizadas en la fabricación y operación de los robots.

- Reglamento CLP (1272/2008/CE): se ocupa de la clasificación, etiquetado y envasado de sustancias y mezclas químicas peligrosas. Si los robots industriales utilizan sustancias químicas peligrosas, el reglamento CLP establece los requisitos para su correcta clasificación y etiquetado.

- EN ISO 10218-1: establece los requisitos de seguridad para los robots industriales. Cubre aspectos como la integridad estructural, los sistemas de control, la interfaz humano-robot y las precauciones de seguridad.

- EN ISO 10218-2: complementa la EN ISO 10218-2 y proporciona directrices específicas para la integración de robots en sistemas de producción.

- EN ISO 13849-1: se centra en los requisitos y principios para el diseño y la validación de los sistemas de control relacionados con la seguridad de las máquinas, incluyendo los robots industriales. Establece criterios para evaluar el rendimiento de seguridad funcional.

- EN 62061: aborda la seguridad funcional de los sistemas de control eléctrico, electrónico y programable relacionados con la seguridad de las máquinas, incluyendo los robots industriales.

- EN 12415: establece los requisitos de seguridad de los robots utilizados en sistemas de manipulación y transporte de materiales.

- EN 15066: se centra en la seguridad en la interacción física entre humanos y robots, y proporciona directrices para la evaluación y reducción de riesgos.

3.1.2 LEGISLACIÓN Y NORMAS NACIONALES.

- Real Decreto 1215/1997, de 18 de julio, por el que se establecen las disposiciones mínimas de seguridad y salud para la utilización por los trabajadores de los equipos de trabajo: Este decreto establece las medidas de seguridad y salud que deben cumplir los equipos de trabajo, incluyendo los robots industriales. Se abordan aspectos como la instalación, uso, mantenimiento y reparación seguro de los equipos de trabajo, así como la formación y protección de los trabajadores.

- Real Decreto 486/1997, de 14 de abril, sobre disposiciones mínimas de seguridad y salud en los lugares de trabajo: En el contexto de la robótica industrial, establece requisitos generales para garantizar la seguridad de los trabajadores y la prevención de riesgos laborales.

- UNE-EN ISO 10218-1 y UNE-EN ISO 10218-2: versión nacional de las normas internacionales ISO 10218-1 e ISO 10218-2, que establecen los requisitos de seguridad para los robots industriales.

- UNE-EN ISO 113849-1 y UNE-EN ISO 13850: establecen los requisitos y principios para el diseño y la validación de los sistemas de control relacionados con la seguridad de las máquinas, incluidos los robots industriales.

3.2 ÁMBITO DE LA VISIÓN ARTIFICIAL.

3.2.1 LEGISLACIÓN Y NORMAS EUROPEAS.

- Reglamento General de Protección de Datos (RGPD): establece las normas para la protección de datos personales en la Unión Europea. Es aplicable a cualquier organización que procese datos personales, incluyendo aquellos obtenidos a través de sistemas de visión artificial. El RGPD establece principios de privacidad, derechos de los individuos y obligaciones para los responsables del tratamiento de datos.

- Directiva de Privacidad y Comunicaciones Electrónicas (ePrivacy): regula la privacidad en las comunicaciones electrónicas, lo cual puede incluir el procesamiento de imágenes o vídeos capturados mediante sistemas de visión artificial. Establece normas sobre el consentimiento, el tratamiento de datos de tráfico y la utilización de cookies y tecnologías similares.

3.2.2 LEGISLACIÓN Y NORMAS NACIONALES.

- Ley Orgánica de Protección de Datos Personales y Garantía de los Derechos Digitales (LOPDGDD): establece las reglas para la protección de datos personales en España y su tratamiento. Es aplicable a cualquier entidad que procese datos personales, incluyendo el uso de tecnologías de visión artificial. Establece los principios de protección de datos, derechos de los individuos y obligaciones para los responsables del tratamiento de datos.

- Reglamento General de Protección de Datos (RGPD): mencionado anteriormente, es también aplicable en España.

- Ley de Servicios de la Sociedad de la Información y de Comercio Electrónico (LSSI-CE): regula los servicios de la sociedad de la información y el comercio electrónico, incluyendo el uso de tecnologías de visión artificial en el contexto digital. Establece normas sobre el envío de comunicaciones comerciales, el uso de cookies y la responsabilidad de los intermediarios en Internet.

- Ley de Propiedad Intelectual (LPI): protege los derechos de propiedad intelectual en España, incluyendo los derechos de autor y las patentes. Puede ser relevante en el ámbito de la visión artificial cuando se trata de la protección de algoritmos, software y otros elementos relacionados.

3.3 ELABORACIÓN Y REDACCIÓN DE PROYECTOS

- UNE 157001:2014: se establecen los criterios generales para la elaboración de los documentos que constituyen un proyecto técnico.

- UNE-ISO 999:2014. Información y documentación: directrices sobre el contenido, la organización y presentación de índices.

- UNE-ISO 690:2013. Información y documentación: directrices para la redacción de referencias bibliográficas y de citas de recursos de información.

- UNE-EN 62023:2002. Estructuración de la información y documentación técnica.

- Normas de realización de los TFG: documento realizado por la Escuela Politécnica Superior de Sevilla para la elaboración de un Trabajo Fin de Grado (TFG).

4 DESCRIPCIÓN DE LA ESTRUCTURA DEL DOCUMENTO.

Una vez conocidos los objetivos del proyecto y el marco normativo legal del mismo, es necesario saber qué estructura presenta el documento.

Antes de comenzar con las tareas a realizar para el desarrollo del prototipo es necesario conocer y aprender los conceptos teóricos básicos sobre los que trata el proyecto. El apartado 5, Estado del Arte, pretende definir de la forma más adecuada las tres grandes tecnologías de este proyecto y conocer detalladamente la metodología necesaria para llevar a cabo el prototipo. También, se desarrolla brevemente la Inteligencia Artificial como introducción.

Tras haber comprendido la parte teórica del documento, el apartado 6 trata de especificar los materiales necesarios para llevar a cabo el proyecto tanto a nivel hardware como software, así como los conocimientos necesarios para su uso de forma correcta.

El apartado 7 detalla paso a paso las tareas a realizar para llevar a cabo el proyecto, desde calibrar la cámara hasta conseguir clasificar el objeto. Cada una explicada detenidamente y dando a conocer los inconvenientes encontrados conforme se avanza en el proyecto, así como el resultado que se obtiene después de realizar cada tarea.

Por otro lado, en el apartado 9, se realiza una breve conclusión y se exponen algunos trabajos futuros que se podrían implementar.

Seguidamente, en el apartado 10, se especifica la bibliografía sobre la que se ha basado el desarrollo del documento.

Por último, en los anexos se detalla como instalar los distintos paquetes para ejecutar el proyecto, el código de Python y las funciones de Matlab necesarias para el control del robot. Además, se muestra el presupuesto necesario para el desarrollo tanto del documento como del prototipo y unas normas básicas para el uso del brazo robótico Scorbot ER-4u y la controladora USB.

5 ESTADO DEL ARTE.

5.1 DEFINICIONES Y CONCEPTOS.

Para el desarrollo de este proyecto y como una breve introducción se deben de conocer los siguientes conceptos:

- Robot: Según el RIA (Robot Institute of America), “Un robot es un manipulador reprogramable y multifuncional, diseñado para mover cargas, piezas, herramientas o dispositivos especiales, según variadas trayectorias, programadas para realizar diferentes trabajos.”

- Robótica: La robótica es una disciplina científica que aborda la investigación y desarrollo de una clase particular de sistemas mecánicos, denominados robots manipuladores, diseñados para realizar una amplia variedad de aplicaciones industriales, científicas, domésticas y comerciales (ref. 18).

- Inteligencia Artificial: según Marvin Minsky, uno de los pioneros de la IA: “La Inteligencia Artificial es la ciencia de construir máquinas para que hagan cosas que, si las hicieran los humanos, requerirían inteligencia”.

- Deep Learning: es el subcampo de la inteligencia artificial que se enfoca en crear grandes modelos de redes neuronales que sean capaces de tomar decisiones precisas basadas en datos. El Deep Learning es particularmente adecuado para contextos donde los datos son complejos y donde hay grandes conjuntos de datos disponibles (ref. 12).

- Visión Artificial: es una disciplina que engloba todos los procesos y elementos que proporcionan ojos a una máquina (ref. 3). La visión artificial o compresión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de ese mundo (ref. 20).

5.2 INTELIGENCIA ARTIFICIAL.

5.2.1 ORIGEN Y EVOLUCIÓN DE LA INTELIGENCIA ARTIFICIAL.

La Inteligencia Artificial empieza a cobrar sentido durante la segunda mitad del siglo XX. En 1943, Wiener junto con Rosenblueth y Bigelow, establece las bases de la Cibernética (control, autorregulación). McCulloch y Pitts exponen la realización de operaciones lógicas mediante el inter conexionado neural y su control por realimentación, demostrando así que cualquier ley de entrada/salida podía implementarse con una red neuronal (ref. 2).

En 1949, Hebb sugerirá mecanismos de aprendizaje (acababa de nacer el paradigma conexionista). Un año más tarde, Turing plantea lo siguiente: *"¿Puede una máquina pensar?"*, para ello plantea una prueba denominada el test de Turing que se define de la siguiente forma: *"Disponemos a un humano y a una máquina en habitaciones diferentes. Un observador les hace una serie de preguntas a uno y a otro a través de la puerta. Si pasado un cierto tiempo el observador no es capaz de determinar quién es el humano y quién es la máquina, podemos concluir diciendo que la máquina posee inteligencia"* (ref. 2).

En 1956 aparece una nueva generación de investigadores entre los que destaca Minsky, McCarthy, Newell, Simon… los cuáles toman se proponen lo siguiente: *"todo aspecto de aprendizaje o cualquier otra característica de inteligencia puede ser definido de forma tan precisa que puede construirse una máquina para simularlo"*. Es por un grupo de estos investigadores por el que surge el Logic Theoristic (complejo sistema de manejo de información), el cuál en los años siguientes se mejoraría y se propondrían nuevos enfoques (ref. 2).

En 1958 surge el primer lenguaje de programación de IA, LISt Processing (LIST). La década de los 50 termina con la creación del programa Simbolic Automatic INTegrator (SAINT) de Slage, el primer programa de juegos de Samuel, y los logros en reconocimiento de patrones con el Perceptrón de Rosenblatt y el primer planteamiento de su regla de aprendizaje (ref. 2).

En la década de los 60, lo más destacable es el proyecto Blocks Micro-Worlds del MIT (Massachusetts Institute of Technology) cuyo objetivo era que, en un mundo repleto de formas geométricas, los robots supieran interpretar una escena, se movieran, manipularan bloques y explicaran sus experiencias. Es entonces donde en 1969 se desarrolla el programa SHRDLU de Winograd que permitía la comprensión del lenguaje natural y un año más tarde el programa ARCHES de Winston que permitía el aprendizaje estructural (ref. 2).

A pesar de todos estos avances, los investigadores no ven cumplidos sus objetivos en IA y a causa del informe ALPAC (disuade a las instituciones norteamericanas de continuar financiando los proyectos de traducción automática), los libros de Dreyfus y Weizenbaum y algunos proyectos con resultados nefastos, hacen que se produzca un desinterés por la IA (ref. 2).

En 1971 se completa el desarrollo de STRIPS, un nuevo planificador de SHAKEY. Entre 1972 y 1976 se financian investigaciones sobre el reconocimiento del habla. Y es en esta época donde se comienza con el desarrollo de la primera teoría sobre visión artificial. En 1975, ARPA financia un proyecto de compresión de imagen (ref. 2).

Es en la década de los 80 cuando se produce una gran expansión de la IA gracias al desarrollo de las redes neuronales que propiciará una nueva generación de sistemas inteligentes y arquitecturas de computación (ref. 2).

Durante los 90, comienza el desarrollo de los sistemas híbridos y surgen nuevas metodologías de adquisición de conocimientos. Se continúa con el desarrollo de la programación, así como con la interpretación del lenguaje natural (ref. 2).

No obstante, a pesar de todo este desarrollo en 50 años, la IA sigue estando en constante crecimiento y en busca de solucionar problemas clásicos como la percepción, el lenguaje natural, etc. Actualmente, el objetivo de los investigadores de IA es resolver estos problemas y poderlos transferir a la industria para que los sistemas comiencen a ser inteligentes y continue el desarrollo tecnológico en campos afines como la biología, psicología o neurofisiología (ref. 2).

5.2.2 PRINCIPALES RAMAS DE LA INTELIGENCIA ARTIFICIAL.

- Machine Learning: permite a un sistema aprender de los datos en lugar de aprender mediante la programación explícita (ref. 8). Son softwares desarrollados, entrenados a diario de forma automática de tal modo que,

conforme avanza el tiempo, son capaces de desempeñar ciertas tareas optimizando los procesos (ref. 19).

- Deep Learning: es un método específico de machine learning que incorpora las redes neuronales en capas sucesivas para aprender de los datos de manera iterativa. Es útil cuando se trata de aprender patrones de datos no estructurados (ref. 9).
- Computer vision: es la rama visual de la inteligencia artificial. Se trata de un conjunto de sistemas y algoritmos que analiza y extrae información de imágenes (ref. 19).

- Neural Networks: las redes neuronales son las encargadas de imitar el sistema de procesamiento de información que tiene un cerebro humano. Esta definición se le acuña porque estas redes tratan de aprender de sus errores y mejorar de forma continuada (ref. 19).

- Natural Language Processing: tecnología de machine learning que permite a cualquier tipo de ordenador comprender, interpretar y manipular el lenguaje humano (ref. 19).

- Natural Language Generation: es la capacidad que tiene un bot para dar una respuesta a un usuario. Lo que pretende la IA es parecerse lo máximo posible en funcionalidad y apariencia al servicio que prestaría una inteligencia humana (ref. 19).

- Virtual Digital Assistant: software que ayuda a los usuarios a realizar tareas de tal manera que la interacción entre el humano y la máquina sea mínima. Esta inteligencia ofrece servicios a los individuos tales como los asistentes de voz o el reconocimiento de la ubicación (ref. 19).

- Chatbot: asistente que se comunica a través de mensajes de texto con los usuarios (ref. 19).

- Recommender Systems: algoritmo que busca predecir el siguiente movimiento de un usuario (ref. 19).

- Predictive Analysis: encargados de analizar datos para poder tratar de predecir una acción que va a ocurrir en el futuro (ref. 19).

5.3 VISIÓN ARTIFICIAL.

5.3.1 HISTORIA Y ANTECEDENTES.

La historia de la visión artificial se remonta a los primeros intentos de los científicos por comprender y replicar la capacidad humana de ver y procesar visualmente el mundo que nos rodea. A lo largo de los años, ha habido importantes antecedentes y avances en esta área, sentando las bases para el desarrollo de la visión artificial moderna.

Los investigadores llevan desde mediados del siglo XX tratando de encontrar la forma de que las máquinas puedan ver y comprender datos visuales. La investigación comenzó en 1958 cuando un grupo de neurofisiólogos llevaron a cabo un experimento con un gato. En este experimento se mostraba una matriz de imágenes al felino, intentando correlacionar una respuesta en su cerebro. Descubrieron que respondía primero a las líneas o a los bordes más marcados, lo que significaba que el procesamiento de imágenes comienza por formas simples como pueden ser los bordes rectos (ref. 7).

Fue en 1960 cuando Larry Roberts (conocido también como el padre de la visión artificial) consiguió extraer de una imagen 2D información tridimensional. Muchos investigadores del MIT y del mundo de la IA siguieron con este trabajo desarrollando nuevos algoritmos (ref. 6). En 1963, se consiguió que los ordenadores transformaran imágenes en 2D a imágenes en 3D (ref. 9).

En 1982, el neurocientífico David Marr estableció que la visión artificial de forma jerárquica e introdujo algoritmos para que las máquinas pudieran detectar bordes, curvas y formas básicas parecidas. Años antes, en 1979, Kunihiko Fukushima desarrolló Neocognitron, una red de celdas capaz de reconocer patrones (ref. 9).

En el año 2000, todos los focos se centraron en el reconocimiento de objetos, consiguiendo en 2001 los primeros programas de reconocimiento facial (ref. 9).

En 2010, se hizo público ImageNet, un conjunto de datos que contenía millones de imágenes etiquetadas en miles de clases de objetos y proporcionaba una base para las CNN y los modelos de Deep Learning que se utilizan hoy en día (ref. 9).

En la actualidad, la visión artificial continúa evolucionando y desarrollándose en áreas más allá de la industria como son la medicina, la seguridad y muchos otros

campos, con el objetivo de mejorar la eficiencia y precisión de las tareas y aplicaciones.

5.3.2 ETAPAS DE UN PROCESO DE VISIÓN ARTIFICIAL.

En la Ilustración 1 se muestras las etapas que conlleva un proceso de visión artificial. En este apartado se describirán brevemente cada una de ellas.

La primera etapa en un proceso de visión artificial es la etapa de adquisición, o como se muestra en la Ilustración 1: Adquisición de Imágenes. En esta etapa el objetivo es adquirir la imagen digital de la forma más adecuada (ref. 3).

El siguiente paso es el preprocesamiento de la imagen digitalizada cuyo objetivo es mejorar la calidad de la imagen eliminando la degradación que pueda sufrir la imagen en forma de ruido, pérdida de definición o fidelidad de la imagen (ref. 3).

La siguiente etapa es la conocida como Segmentación de la cuál obtendremos los objetos que hay en la imagen y así extraer las características de cada uno de ellos (ref. 3).

Como cuarta etapa tenemos la de parametrización o selección de rasgos que se dedica a extraer rasgos que den alguna información cuantitativa de interés o rasgos que permiten diferenciar distintas clases de objetos (ref. 3).

Por último, se encuentra la etapa de reconocimiento y la de interpretación. En la etapa de reconocimiento se clasifican los objetos de forma que se le asignan una etiqueta en la que se proporciona información. La etapa de interpretación consiste en actuar según los resultados obtenidos (ref. 3).

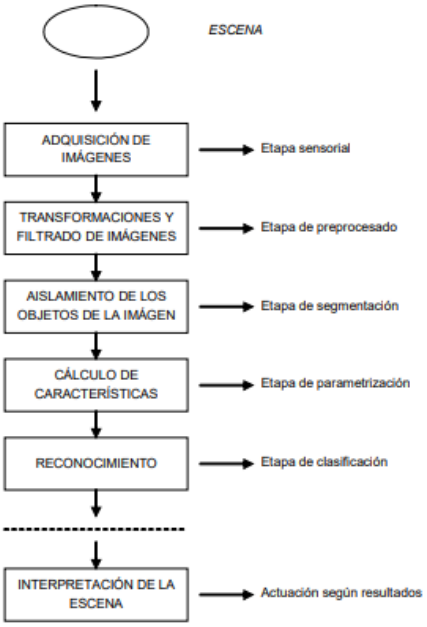


Ilustración 1. Diagrama de bloques de las etapas de un proceso de visión artificial

5.3.3 COMPONENTES DE UN SISTEMA DE VISIÓN ARTIFICIAL.

Para que un sistema de visión artificial se lleve a cabo, es necesario un conjunto de elementos hardware:

- Sensor óptico: se encarga de producir una imagen completa de la escena. Puede ser una cámara color o monocromo, cámara scanner, etc. (ref. 3).

- Tarjeta de Adquisición de imagen: su objetico es digitalizar la señal recibida por el sensor óptico (ref. 3).

- Computador: almacena en la memoria la imagen digitalizada para posteriormente ser procesada y manipulada por un programa (ref. 3).

- Monitor de vídeo: permite visualizar las imágenes o videos ya procesados (ref. 3).

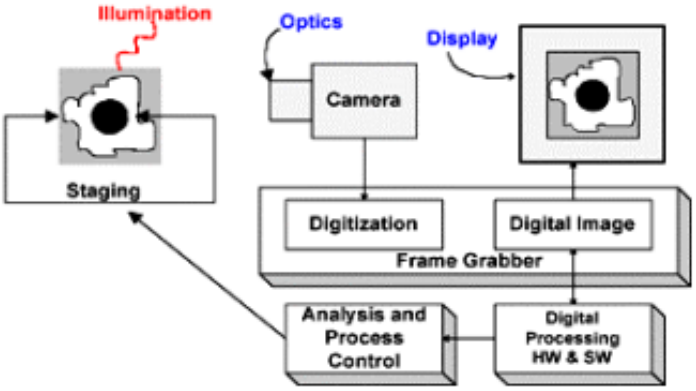


Ilustración 2. Diagrama de bloques de un sistema de Visión Artificial

5.3.4 ADQUISICIÓN DE DATOS.

En esta sección se aborda el proceso de captura de imágenes en el contexto de visión artificial.

La adquisición de datos trata de conseguir que la imagen sea lo más adecuada posible. Una correcta adecuación de la imagen supone un mayor éxito a la hora del reconocimiento de objetos (ref. 3).

En esta etapa existen múltiples factores que influyen en el proceso de captura de la imagen como son la cámara, la óptica, la tarjeta de adquisición, el ordenador o el software, además del entorno (iluminación y fondo) cómo el posicionamiento de los elementos (ref. 3).

5.3.4.1 LA CÁMARA.

La cámara es un dispositivo que desempeña un papel fundamental en el proceso de captura de imágenes y permite la obtención de datos visuales para su posterior análisis y procesamiento. La elección de la cámara a utilizar va a depender de múltiples factores:

- Cámara de color o monocromática: la primera elección se basa es si vamos a utilizar una cámara de color o de escala de grises. Esta elección dependerá del presupuesto, pero fundamentalmente del proceso que se vaya a realizar. Generalmente en los sistemas de visión artificial la adquisición se realiza mediante cámaras monocromáticas, siendo las ventajas principales

económicas y de cómputo. También se debe de tener en cuenta de que en una cámara de color el tiempo de procesado de la imagen es mayor (ref. 3).

- Cámara digital o analógica: actualmente se tiende a utilizar cámaras digitales por sus ventajas con respecto a las analógicas (velocidad y calidad). Son dispositivos comunes y versátiles que utilizan sensores de imagen para capturar imágenes o vídeos. Existen diferentes tipos de cámaras digitales, como cámaras compactas, cámaras réflex digitales (DSLR) y cámaras sin espejo (mirroless) (ref. 3).

- Tipo de salida de vídeo: existen tres tipos de salida de vídeo: vídeo compuesto, vídeo digital y salida RGB (para cámaras de color) (ref. 3).

- Formato PAL o NTSC: son dos formatos de vídeo. El formato europeo PAL (50 Hz) o el americano NTSC (60 Hz) (ref. 3).

- Salida entrelazada o no entrelazada: la salida entrelazada es ideal para aplicaciones donde los objetos se encuentran quietos o que se desplazan de forma lenta, mientras que la salida no entrelazada es ideal para imágenes en movimiento. Lo ideal sería una cámara que permitiera los dos tipos de salida (ref. 3).

- Sincronismos Vertical y Horizontal: permite una completa iteración entre la tarjeta y la cámara, pudiéndose realizar la captura de los frames en el momento que deseamos (ref. 3).

- Velocidad del obturador: generalmente existen cámaras con velocidad de su obturador de 1/60, 1/125, 1/250, 1/500, 1/1000, 1/2000, 1/4000, 1/10000 segundos de obturación (ref. 3).

- Características del CCD: el CCD es una de las partes más importantes de la cámara y entre sus características fundamentales se encuentra la resolución y la relación calidad-ruido (ref. 3).

- Programación vía RS232: una programación software puede permitir realizar el calibrado de la cámara permitiendo una automatización de este proceso de calibrado (ref. 3).

En resumen, la elección del dispositivo de adquisición de imágenes dependerá de factores como la aplicación específica, la calidad requerida, el presupuesto y las limitaciones técnicas.

5.3.4.2 LA ÓPTICA.

La óptica depende de las condiciones ambientales y de la distancia de medición. En la selección de la óptica, es necesario evitar aquellas lentes que deformen la imagen (lentes que amplían o reducen el campo de visión, etc.). Además, se aconseja estudiar las distancias y tamaño de los objetos a medir para elegir una óptica adecuada. Por último, es conveniente proveerse de unos anillos de extensión que den más posibilidades de aumentar o disminuir las distancias mediante el uso de lentes fijas (ref. 3).

5.3.4.3 LA TARJETA DE ADQUISICIÓN DE DATOS.

Las características que se deben tener en cuenta a la hora de elegir la tarjeta de adquisición de datos son las siguientes:

- Tipo de entrada de vídeo: la entrada de vídeo debe admitir la salida de vídeo de la cámara; sea vídeo compuesto, RGB o digital. Por otro lado, se debe considerar el uso simultáneo de varias cámaras, por ejemplo, en sistemas de visión tridimensional (ref. 3).

- Memoria en la tarjeta y controlador propio de DMA: se debe escoger una tarjeta que disponga de memoria en sí misma y de un sistema de control DMA eficiente que permita llevarla hasta la memoria del ordenador. Las tarjetas que usan la memoria del ordenador presentan varios inconvenientes. Uno de ellos es que puede producir conflictos con el sistema operativo. Por otro lado, estas tarjetas al tener que gestionar la memoria propia del sistema tienen que acceder al chipset del ordenador, siendo muchas veces incompatibles (ref. 3).

- Entradas y salidas de sincronismo: la tarjeta de vídeo debe permitir el control de las señales de sincronismo de la cámara, sobre todo cuando se utilizan varias cámaras a la vez. Existen tarjetas que tienen una salida de sincronismo en vídeo compuesto que permite el uso de varias cámaras. Este proceso denominado sync-locking permite sincronizar las señales cuando se realiza la conmutación de canales o cuando se capturan imágenes de más de una cámara a la vez (ref. 3).

- Posibilidad del uso de triggers externos, entradas y salidas digitales: en toda tarjeta de adquisición de datos es fundamental la presencia de entradas de trigger que permitan realizar un disparo cuando se produce un evento externo. También se valora la presencia de entradas y salidas digitales de forma que no se necesite otra tarjeta adicional (ref. 3).

- Uso de DSP: el uso de un procesador digital de señal es fundamental puesto que en un entorno industrial las aplicaciones que se realizan generalmente son en tiempo real. El DSP puede ser programado para realizar las tareas de preprocesado, reduciendo el tiempo de proceso que va a tener que realizar la CPU del host (ref. 3).

- Uso de procesadores especiales para proceso pipeline: actualmente existen tarjetas formadas por procesadores capaces de realizar el procesado en pipeline aumentando la velocidad de cálculo (ref. 3).

- Bus PCI: se recomienda el uso de estos buses por su gran capacidad de transferencia de hasta 132 MB/s en burst mode (ref. 3).

- Software de adquisición y librerías: el software de adquisición y las librerías son fundamentales a la hora de un buen uso de la tarjeta. Por ello es necesario que se ajuste a nuestras necesidades (ref. 3).

Es importante tener en cuenta que las características específicas de una tarjeta de adquisición de datos pueden variar según el fabricante y modelo. La elección de la tarjeta adecuada dependerá de los requisitos y necesidades específicas del sistema de visión artificial implementado.

5.3.4.4 EL HOST Y SOFTWARE UTILIZADO.

Otro aspecto importante es el software utilizado y el ordenador que va a realizar el proceso de cálculo.

Dentro de la gama de sistemas que se puede encontrar en el mercado, aparecen máquinas con procesadores de la familia INTEL, ALPHA, MIPS, etc. Aunque si realmente el ordenador va a trabajar en un ambiente industrial donde hay polvo, humedad, cambios bruscos de temperatura, vibraciones, etc.; esto lleva a utilizar ordenadores industriales (ref. 3).

Uno de los aspectos más importantes a la hora de escoger un buen ordenador industrial es su procesador. Actualmente los procesadores más avanzados y que más se encuentran en el mercado son los de la familia INTEL. Aunque también es importante el uso de una gran cantidad de memoria, de un buen disco y de una tarjeta de visualización mediante BUS AGP que permita reducir los tiempos de transferencia en la visualización de las imágenes capturadas (ref. 3).

Además del ordenador, el software es otro de los aspectos para tener en cuenta. Los programas de visión artificial profesionales están formados por un entorno visual de pruebas, procesado y estudio; lenguajes de programación y librerías de desarrollo (ref. 3).

5.3.4.5 TÉCNICAS DE ILUMINACIÓN.

5.3.4.5.1 SISTEMAS DE ILUMINACIÓN.

La iluminación del área de trabajo debe realizarse de una forma correcta, dada su importancia. Existen dos formas de iluminación:

- Iluminación frontal: la luz incide directamente sobre el objeto, ya sea de forma vertical, horizontal, oblicua o difusa (ref. 3).

- Iluminación trasera o Retroiluminación: se ilumina una pantalla de forma que lo que se busca es el contorno del objeto (ref. 3).

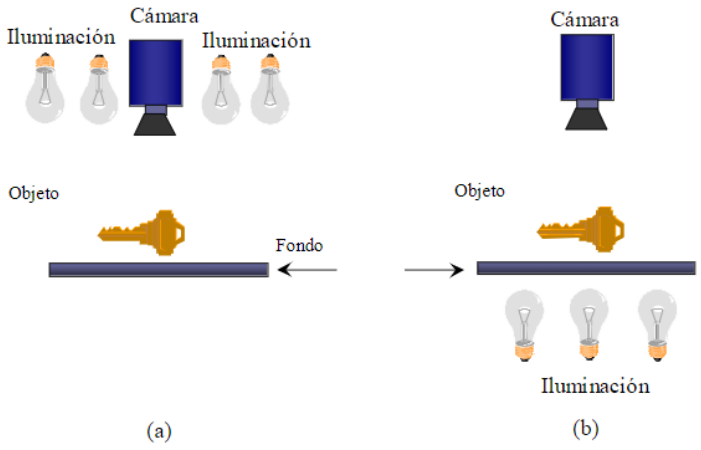


Ilustración 3. Tipos de iluminación: a) frontal y b) retroiluminación

La iluminación frontal permite distinguir los detalles de los objetos, así como su forma, permitiendo extraer más parámetros de cada objeto (color, detalles internos, etc.) aunque presenta dos inconvenientes: la creación de sombras y los reflejos. Para evitar estos inconvenientes se hace uso de una luz difusa, de lámparas circulares o de una luz indirecta (ref. 3).

La retroiluminación sólo permite la detección de contornos del objeto y se puede hacer de dos formas:

- Proyección sobre una pantalla: se sitúa el objeto entre la pantalla y los focos que lo iluminan de forma que lo que captura la cámara es la sombra proyectada sobre la pantalla (Ilustración 4.a) (ref. 3).

- Iluminación del fondo de la escena: se sitúa el objeto entre la cámara y la pantalla de forma que lo que se ilumina es el fondo de la escena (Ilustración 4.b) (ref. 3).

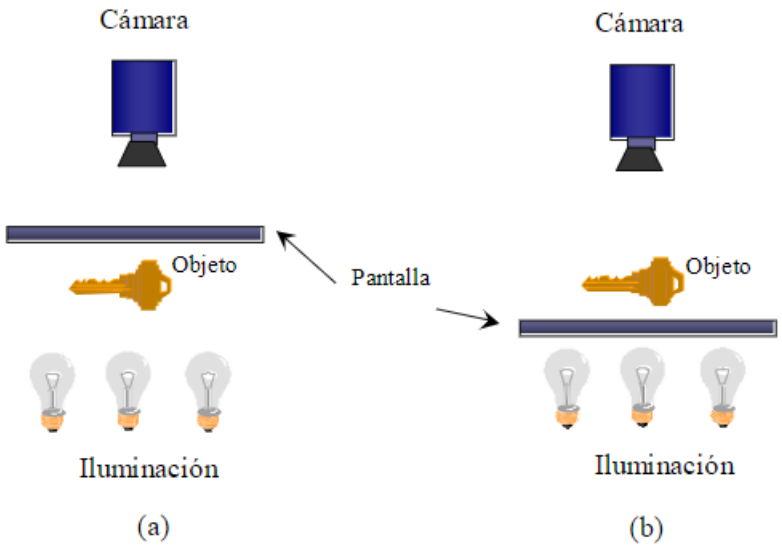


Ilustración 4. Retroiluminación: a) proyección sobre una pantalla, b) iluminación del fondo de la escena.

La proyección del objeto sobre la pantalla presenta el inconveniente de que, si existen objetos montados uno encima de otro, el reconocimiento de ambos es difícil. Otro de los inconvenientes es que, si la iluminación no está completamente perpendicular a la pantalla, la sombra del objeto se deforma. Por otro lado, presenta la ventaja de que el procesado y segmentación se realiza en pocos pasos (ref. 3).

El segundo método es más ventajoso. Su objetivo es iluminar el fondo de forma que sea más fácil diferenciarlo de los objetos. Este fondo suele ser una pantalla blanca que difumina la luz trasera produciendo un fuerte fondo blanco iluminado. Al ser más fuerte esta iluminación, las sombras se eliminan consiguiendo un fuerte contraste entre el objeto y el fondo (ref. 3).

La desventaja que presenta la retroiluminación es su uso en un sistema industrial automático. La detección de objetos en cintas transportadoras, si utilizamos este sistema de iluminación, la luz tendría que estar debajo de la cinta por lo que sería necesario una cinta translúcida, que actúe como pantalla, transportando los objetos a través del sistema de visión (ref. 3).

5.3.4.5.2 APLICACIONES DE LOS SISTEMAS DE ILUMINACIÓN.

La selección de la iluminación va a depender de varios factores como son las características de la superficie, las necesidades del sistema, etc. De forma general, los sistemas de iluminación más utilizados son:

- Sistemas anulares: proporcionan una gran intensidad de luz uniforme, sin sombras, por lo que son adecuados para iluminar objetos de reducidas dimensiones (ref. 3).

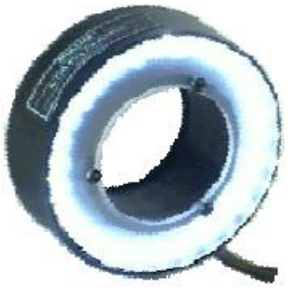


Ilustración 5. Iluminador anular.

- Sistemas de iluminación lineal: apropiados para iluminar zonas estrechas . (ref. 3).

Ilustración 6. Iluminador lineal.

- Sistemas de panel: se utiliza para formar una silueta de un objeto, depositándolo sobre la fuente de luz, cuando las estructuras superficiales del mismo no son importantes (ref. 3).

Ilustración 7. Iluminador de panel.

- Sistemas puntuales: están diseñados para iluminar intensamente desde distintas posiciones. Estos pueden ser flexibles o semirrígidos (ref. 3).

Ilustración 8. Iluminadores puntuales flexibles y semirrígidos.

- Luz difusa: se utilizan en superficies brillantes y lisas donde cualquier reflexión puede confundirse con un defecto (ref. 3).

5.3.4.5.3 EL FONDO.

El fondo debe ser lo más homogéneo posible y de un color que permita distinguirlo fácilmente de los objetos. Cuando se utiliza iluminación frontal, el fondo debe ser lo más opaco posible evitando todo reflejo. Si la iluminación es trasera, se busca que la pantalla difumine lo más posible la luz de forma que se obtenga un fuerte fondo blanco que lo distinga fácilmente de los objetos (ref. 3).

5.3.4.6 CALIBRACIÓN DE LA CÁMARA.

El objetivo de calibrar una cámara es determinar las características internas, geométricas y ópticas de la cámara (Parámetros intrínsecos) y la posición en 3D del marco de la cámara con respecto a un sistema de coordenadas (Parámetros extrínsecos) (ref. 5).

Gracias a la calibración se obtienen los parámetros que intervienen en el proceso de formación de imágenes. Entonces, la calibración se entiende como el proceso por el que se establece una relación entre coordenadas tridimensionales de los objetos en el entorno con sus correspondientes proyecciones bidimensionales en la imagen (ref. 5).

5.3.4.6.1 PARÁMETROS EXTRÍNSECOS E INTRÍNSECOS.

- Parámetros Intrínsecos: describen la geometría y óptica de la cámara y tarjeta de adquisición de imágenes (ref. 5).

- Parámetros Extrínsecos: definen la orientación y posición de la cámara respecto a un sistema de coordenadas denominado sistema del mundo. Se definen tres grados de libertad para el desplazamiento y otros tres para la orientación (ref. 5).

5.3.4.6.2 MODELO PIN-HOLE.

El modelo básico de la cámara o también conocido como modelo Pin-Hole representa la transformación de las coordenadas de los puntos de la escena en las coordenadas de la imagen (ref. 5).

En este modelo cada punto de un objeto situado en el espacio de trabajo 3D se proyecta en un punto de un plano denominado plano imagen (ref. 5).

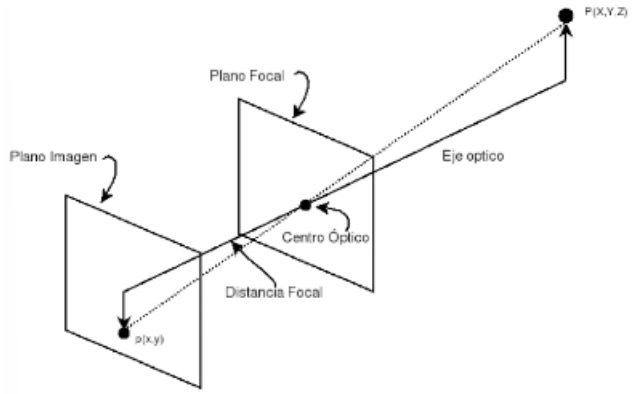


Ilustración 9. Proyección de un punto del espacio en la imagen con el modelo Pin-Hole.

Los parámetros intrínsecos se agrupan en la matriz de calibración, propuesta por el modelo de cámara, que se ve modificada por la variación, en condiciones reales, de los siguientes parámetros: desplazamiento del punto principal, escalado, distorsión óptica y distorsión focal (ref. 5).

Los parámetros extrínsecos se agrupan en la matriz de rotación y el vector de traslación que relacionan las coordenadas tridimensionales del objeto con las coordenadas tridimensionales de la cámara (ref. 5).

5.3.4.6.3 TÉCNICAS SEGÚN EL OBJETIVO DE CALIBRACIÓN.

Dependiendo del objeto utilizado para realizar la calibración, podemos distinguir entre calibración coplanar y no coplanar.

- Coplanar: el objeto utilizado es plano. Este objeto es un patrón impreso al que se le llama rejilla de calibración. Generalmente se utiliza un tablero de ajedrez (ref. 5).

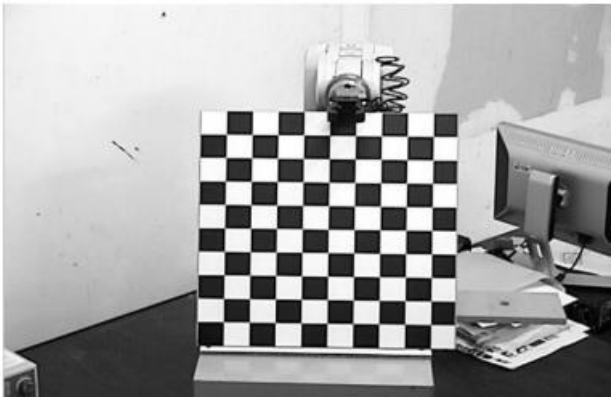


Ilustración 10. Técnica de calibración coplanar.

- No coplanar: el objeto utilizar para la calibración es un cubo en el que cada cara aparece el mismo patrón. También se suele utilizar una rejilla que consta de dos planos con el mismo patrón (ref. 5).

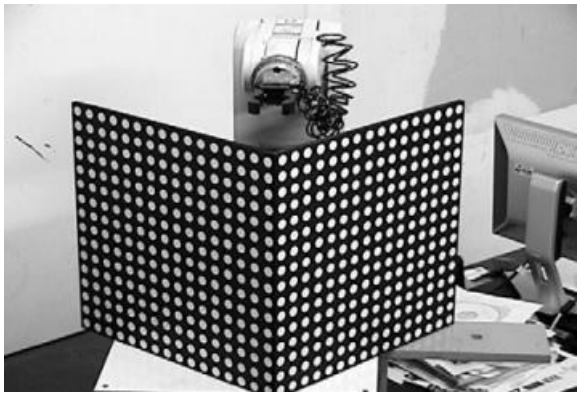


Ilustración 11. Técnica de calibración no coplanar.

5.3.4.6.4 MODELO MATEMÁTICO.

Cualquier punto tridimensional de la escena está definido por tres coordenadas (X_W, Y_W, Z_W). Este punto tridimensional puede llevarse a un sistema de coordenadas de la cámara definido como (X_C, Y_C, Z_C). Esto se consigue con dos matrices, una de rotación *R* y otra de translación *T* (ref. 5):

$$\begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix} = R \begin{pmatrix} X_W \\ Y_W \\ Z_W \end{pmatrix} + T = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix} \begin{pmatrix} X_W \\ Y_W \\ Z_W \end{pmatrix} + \begin{pmatrix} T_Z \\ T_Z \\ T_Z \end{pmatrix}$$

Ecuación 1. Sistema de coordenadas tridimensional de la cámara.

Los parámetros *R* y *T* son los conocidos como parámetros extrínsecos.

Este sistema de coordenadas tridimensional se debe llevar a un espacio bidimensional a la hora de trabajar con una imagen (ref. 5). Las coordenadas de la imagen (x_b, y_b) se calculan de la siguiente manera:

$$\begin{pmatrix} x_b \\ y_b \\ 1 \end{pmatrix} = k \begin{pmatrix} f_{cx} & s & C_x \\ 0 & f_{cy} & C_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix}$$

Ecuación 2. Sistema de coordenadas bidimensional de la imagen.

Donde f_{cx} y f_{cy} son las distancias focales, C_x y C_y son el centro óptico de la imagen y *s* se denomina encuadre y generalmente corresponde a un ángulo de 90º. El valor *k* es un factor de escalado.

Todos estos parámetros son los denominados parámetros intrínsecos.

5.3.4.6.5 LA DISTORSIÓN.

Existen dos efectos que distorsionan la imagen, conocidos como distorsión radial y distorsión tangencial. Estos efectos se modelan generalmente con 5 coeficientes. La distorsión radial se modela con los coeficientes k_{C1}, k_{C2} y k_{C5}. La distorsión tangencial con los coeficientes k_{C3}, k_{C14} (ref. 5).

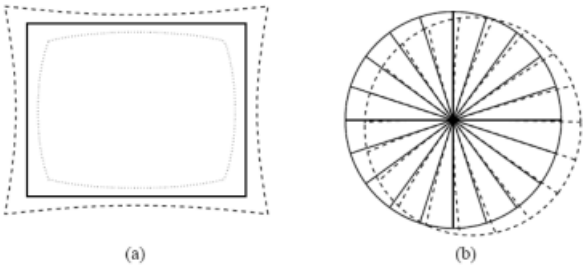


Ilustración 12. Efecto de la distorsión: a) radial, b) tangencial.

Para corregir estos efectos se hace uso de una ecuación con términos no lineales como, por ejemplo:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = (1 + kc_1(x_b^2 + y_b^2) + kc_2(x_b^2 + y_b^2)^2 + kc_5(x_b^2 + y_b^2)^3) \begin{pmatrix} x_b \\ y_b \end{pmatrix}$$

Ecuación 3. Ecuación no lineal para la distorsión.

5.3.4.6.6 OBTENCIÓN DE LOS VALORES DE LOS PARÁMETROS.

Mediante tres ángulos se pueden calcular los valores de la matriz de rotación. Si a estos tres les sumamos los tres de translación son seis valores desconocidos para los parámetros extrínsecos. Por otro lado, en los parámetros intrínsecos conocemos el valor *k* y *s* y son desconocidos las distancias focales y el centro óptico. En total son 10 valores que desconocemos (ref. 5).

Generalmente para determinar estos valores, se toman muchos puntos de la imagen con el patrón de calibración. Para detectar estos puntos se hace uso de alguna técnica de detección de esquinas (ref. 5).

Ilustración 13. Detección de esquinas en el patrón de calibración.

Se asume un punto 3D como origen de la escena y se coloca el eje Z orientado hacia la cámara para facilitar el cálculo de los puntos 3D. Una vez obtenidos los puntos de varias imágenes (es aconsejable realizar el mayor número de imágenes posible para garantizar una buena calibración) se arma un sistema de

ecuaciones utilizando las ecuaciones (1) y (2) pero con los valores conocidos de la cámara (x_{bi}, y_{bi}) y de la escena (X_{Wi}, Y_{Wi}, Z_{Wi}). Este sistema de ecuaciones de resuelve utilizando técnicas de pseudoinversa o mediante el planteamiento de ecuaciones como el método de Tsai. (ref. 5).

5.3.4.6.7 CALIBRACIÓN DE HEIKKILÄ.

Esta técnica es una derivación del modelo Pin-Hole en el que se realiza la corrección de los efectos de la distorsión. Para ello se le añade tres pasos adicionales:

- Estimación de los parámetros lineales: se realiza una transformación lineal directa para transformar las coordenadas del objeto a coordenadas en la imagen (basado en el modelo Pin-Hole), ignorando la distorsión radial y tangencial. El objetivo es resolver los parámetros de la matriz DLT normalizando para evitar una solución trivial y resolviendo la ecuación con una técnica pseudoinversa. Mediante los valores de la matriz DLT se extraen los parámetros de calibración haciendo una descomposición lineal (ref. 5).

- Estimación no lineal: al incorporar los valores de distorsión ya no se puede utilizar la DLT, por lo que se hace uso de un método de mínimos cuadrados (ref. 5).

- Corrección de la imagen: se corrige la imagen con un algoritmo que usa los valores de distorsión (ref. 5).

5.3.5 PREPROCESADO.

Toda imagen capturada por medios ópticos, electroópticos o electrónicos sufre los efectos de la degradación que se manifiesta en forma de ruido, pérdida de definición y de fidelidad de la imagen. Esta degradación puede ser producida por el ruido de los sensores de captura, imprecisiones en el enfoque de la cámara o movimiento de esta u otras perturbaciones (ref. 3).

El objetivo del preprocesado es reparar la imagen eliminando así los desperfectos producidos (ref. 3).

A parte de la degradación existen otras características de la imagen que también son convenientes mejorar cómo el contraste, el brillo, niveles de grises, etc. A

estas técnicas de corrección de la imagen se les conoce como operaciones de mejora de la imagen (ref. 3).

Algunas de las operaciones básicas utilizadas en visión artificial en el proceso de preprocesado son: inversión, operaciones aritméticas, operaciones lógicas, transformaciones no lineales, slicing, clipping, umbralización y binarización (ref. 3).

Para la reconstrucción de imágenes deformadas, giradas o no ajustadas se hace uso de técnicas de transformación geométrica de la imagen. Algunas de las técnicas más utilizadas son: escalado, transformación, giro y espejo. Éstas se basan en realizar una nueva distribución de los píxeles según se pretenda. Para calcular los valores de los nuevos píxeles se utiliza la técnica de interpolación (la más utilizada es la interpolación lineal) (ref. 3).

También existen técnicas generadoras de ruido las cuales permiten introducir un ruido en una imagen para, por ejemplo, probar la respuesta del sistema de visión a diferentes interferencias que se puedan introducir. Para eliminar este ruido se hace uso de filtros (ref. 3).

La mayoría de las implementaciones de filtros se realizan en el dominio espacial o dominio frecuencial (ref. 3).

Los métodos basados en el dominio espacial manipulan de forma directa la luminancia de los píxeles mientras que en el dominio frecuencial modifica indirectamente la luminancia de cada píxel utilizando como factores de ponderación los valores de los otros píxeles de la imagen o del entorno del punto y las relaciones numéricas entre ellos (ref. 3).

Para determinar el nivel de gris de la imagen se hace uso del histograma. El histograma de una imagen es una curva donde se representa la frecuencia con la que aparece cada nivel de gris en la imagen. El histograma puede ser útil para saber dónde está la mayor parte de la información en la imagen y así poder realizar una mejor comprensión de ella (ref. 3).

5.3.6 SEGMENTACIÓN.

La segmentación consiste en dividir una imagen en zonas disjuntas e individualizadas, es decir, consiste en diferenciar los diversos objetos y dónde se encuentran (ref. 3).

El objetivo final de la segmentación es conocer los objetos que hay en la imagen para extraer las características de cada uno de ellos (ref. 3).

En el campo de la segmentación existen números técnicas desarrolladas por investigadores. En este apartado se explica brevemente algunas de ellas.

5.3.6.1 SEGMENTACIÓN BASADA EN UMBRALIZACIÓN.

Este método consiste en convertir una imagen en escala de grises a una imagen binaria, con solo dos niveles: claro y oscuro. La segmentación por umbralización se aplica cuando existe una clara diferencia entre el objeto y el fondo. Para clasificar cada píxel como claro u oscuro, se compara su intensidad con una intensidad de referencia denominada umbral (ref. 3).

Para determinar este umbral existen diferentes métodos como:

- Método P-Cuantil.
- Umbralización basada en la búsqueda de mínimos.
- Umbralización basada en la minimización de la varianza de cada grupo de valores.
- Umbralización basada en técnicas de reconocimiento de formas.
- Umbralización en estructuras de datos jerárquicos.

5.3.6.2 SEGMENTACIÓN BASADA EN BORDES.

Las técnicas de segmentación basadas en bordes consisten en detectar los bordes que existen en la imagen. El objetivo es agrupar bordes locales en una imagen donde sólo cadenas de bordes con una correspondencia con objetos en la imagen o partes de la imagen están presentes (ref. 3).

Algunas de las técnicas utilizadas son:

- Relajación de bordes.
- Extracción de la frontera.

- Transformada de Hough.
- Transformada Radon.
- Transformada del rellenado creciente.

5.3.6.3 SEGMENTACIÓN ORIENTADA A REGIONES.

Sí en el anterior caso se hablaba de la segmentación basada en bordes, ahora directamente se examinan las regiones de la imagen (ref. 3).

Algunos de los métodos utilizados son:

- Unión de regiones.
- Crecimiento de regiones por agregación de píxeles.
- Algoritmos de crecimiento por mezclado.
- División y fusión de regiones.
- Segmentación mediante extracción y análisis de texturas.

5.3.6.4 TÉCNICAS DE CLUSTERIZADO.

El objetivo de las técnicas de clusterizado es definir un conjunto de características asociadas a cada píxel generando un vector multidimensional por cada punto de la pantalla (ref. 3).

Esta técnica se encarga agrupar dichos vectores según la semejanza entre cada uno de ellos y así se obtienen N grupos de píxeles con características semejantes entre ellos (ref. 3).

Una de las técnicas de clusterizado más usada es la que se conoce cómo clusterizado substractivo. Esta técnica calcula el potencial (valor directamente proporcional al número de puntos vecinos e inversamente proporcional a la distancia entre ellos) de cada punto con respecto a los demás. Si un punto tiene un potencial elevado querrá decir que tiene muchos puntos cercanos a él. El objetivo final de esta técnica es encontrar el número de grupos y los centros de

gravedad de cada uno y mediante el punto de mayor potencial se obtiene el primer centro, anulando todos los puntos que estén a una distancia prefijada de él. Se busca el siguiente punto con potencial mayor y se realiza el mismo proceso. Este proceso se repite hasta que el potencial sea menor que el estipulado (ref. 3).

5.3.7 ANÁLISIS DE DATOS.

El último paso consiste en obtener parámetros que definan las características de cada objeto: forma, textura, color, orientación…
Entre todos los parámetros que se pueden obtener tenemos que seleccionar aquellos que tengan las siguientes características:

- Ser discriminantes: diferencien lo mejor posible los objetos de una familia o clase con las demás (ref. 3).

- Ser independientes entre sí: los descriptores que definen cada objeto deben ser independientes, es decir, uno no puede depender de otro. Así cuando un descriptor varie, los demás no lo hagan (ref. 3).

- Ser suficientes: tienen que delimitar de forma suficiente la pertenencia de un objeto a una clase determinada (ref. 3).

Dentro de los descriptores se definen tres categorías: descriptores de frontera, de región y descriptores adecuados para representar estructuras tridimensionales (ref. 3).

Los descriptores de frontera son parámetros que se obtienen del estudio del contorno de los objetos (ref. 3).

Los descriptores de región se pueden obtener del propio análisis del contorno del objeto o por características internas del mismo (ref. 3).

5.4 DEEP LEARNING.

El Deep Learning es una técnica informática para extraer y transformar datos, en casos que van desde el reconocimiento del habla hasta la detección de objetos, movimientos, animales en imágenes mediante el uso de múltiples capas de redes neuronales. Cada una de estas capas actúa como entrada de la capa

anterior y son entrenadas con algoritmos que minimizan los errores y mejoran la precisión (ref. 4).

Esta tecnología puede emplearse en aplicaciones como la medicina, la biología, en el área de la robótica o en la generación de imágenes. Además, el Deep Learning puede emplearse en la visión artificial para la interpretación de imágenes, reconocimiento facial, lectura de señales, localización de vehículos, etc. (ref. 4).
Tal y como se vio en el apartado 5.2.2, el Deep Learning es un campo de la inteligencia artificial. En el área de la IA hay diferentes vertientes y una de ellas en el Aprendizaje Automático o Machine Learning. Como subconjunto del Machine Learning aparecen las redes neuronales artificiales. Este conjunto más los mecanismos que permiten entrenar las redes neuronales artificiales constituyen el Deep Learning (ref. 14).

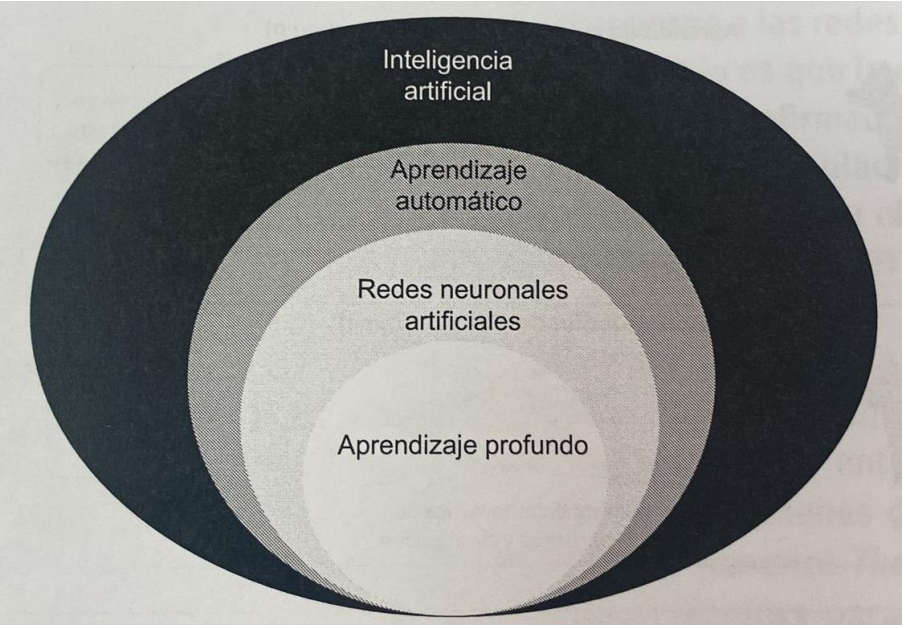


Ilustración 14. El Deep Learning como un campo de la IA.

5.4.1 BREVE RESEÑA HISTÓRICA.

El concepto de neurona surge a finales del siglo XIX cuando el científico español Santiago Ramón y Cajal describe los diferentes tipos de neuronas. Al mismo tiempo, plantea la que se conoce como la teoría de la neurona: el sistema nervioso estaba constituido por neuronas individuales que se comunicaban entre sí por medio de contactos funcionales denominados sinapsis (ref. 14).

Los primeros intentos de desarrollar modelos de redes neuronales llegan en 1943 de mano de McCulloch y Pitts. Los resultados de estos modelos fueron funciones lógicas como “or” o “and” (ref. 14). Cinco años más tarde, en 1949, Donald Hebb presenta el principio del aprendizaje no supervisado en su libro The Organization of Behavior (ref. 14).

Pero el desarrollo de las redes neuronales no sólo destacaba en el área de la neurociencia, sino que comenzó a progresar en la física y la ingeniería. En 1958, Rosenblatt desarrolló el modelo neuronal Perceptron (ref. 14).

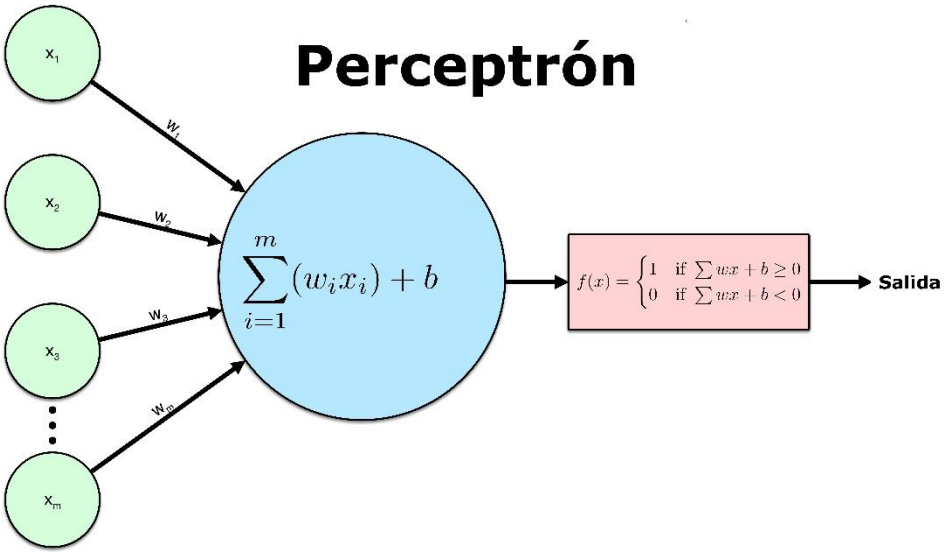


Ilustración 15. Modelo neuronal Perceptron.

El Perceptron posee una sola capa de procesamiento que se encarga de manipular los datos entregados por las neuronas de entrada, generando así la salida de la red neuronal (ref. 14).

En 1960 se inventa un nuevo modelo denominado Adaline desarrollado por Widrow y Hoff basado en el concepto de gradiente descendiente.

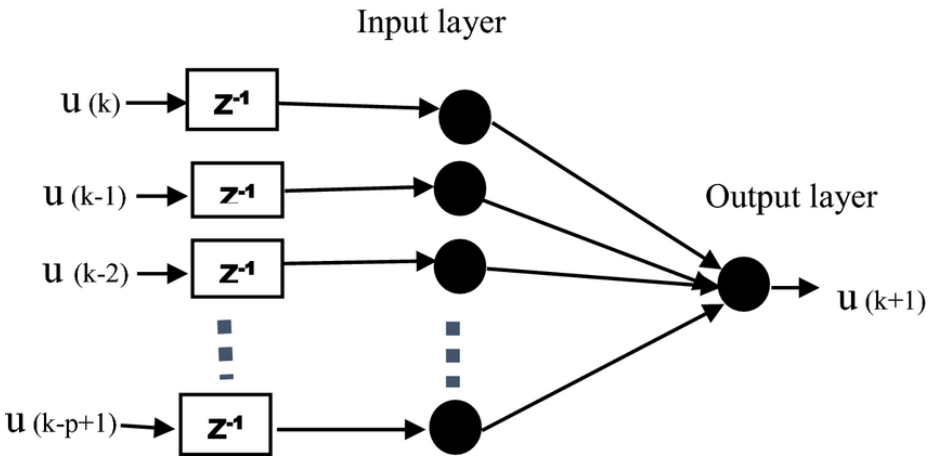


Ilustración 16. Modelo neuronal Adaline.

En 1969, Minsky y Papert presentaron un trabajo basado en las limitaciones del Perceptron a la hora de resolver problemas complejos. El resultado de este trabajo llevó consigo un gran desinterés de los investigadores en esta área que se vio estancada durante años (ref. 14).

No fue hasta el año 2006 donde comienza la era del aprendizaje profundo. Se comenzaron a resolver las dificultades que existían para entrenar redes neuronales de varias capas ocultas dando origen al aprendizaje profundo o Deep Learning. En el año 2012, una red basada en Deep Learning ganó la competencia de reconocimiento de imágenes de Imagenet. A partir de esa se ha convertido en el gran foco de atención y desarrollo. Actualmente, empresas como Microsoft, Facebook o Google utilizan esta tecnología para el reconocimiento de objetos en una imagen, la traducción simultánea o el etiquetado de imágenes (ref. 14).

5.4.2 REDES NEURONALES ARTIFICIALES.

Para comprender el Deep Learning, se debe comenzar por un entender los conceptos básicos sobre las redes neuronales artificiales (RNA).

Las RNA son una técnica de inteligencia artificial clasificada en el campo del Machine Learning. Surgen por la motivación de los investigadores para conectar neuronas entre sí, debido a que una sola neurona artificial posee una baja capacidad de procesamiento y su nivel de aplicabilidad es bajo (ref. 14).

La distribución de las neuronas dentro de una red neuronal artificial se realiza formando niveles de un número de neuronas determinado. Si un conjunto de

neuronas artificiales recibe simultáneamente el mismo tipo de información, se denominará capa (ref. 14). En una red se puede distinguir tres niveles:

- Entrada: conjunto de neuronas que recibe la información externa de la red (ref. 14).

- Oculto: conjunto de neuronas internas de la red y que no tienen contacto con el exterior (ref. 14).

- Salida: conjunto de neuronas que transfiera la información que la red ha procesado (ref. 14).

La forma como se organizan las neuronas dentro de una red neuronal se conoce como su arquitectura y se puede clasificar dependiendo del número de capas (monocapa, multicapa superficial y multicapa profunda) o dependiendo de la forma en la que fluye la información (redes Feedforward y recurrentes) (ref. 14).

5.4.3 MACHINE LEARNING VS DEEP LEARNING.

El aprendizaje automático o Machine Learning son una serie de técnicas capaces de aprender las relaciones intrínsecas que hay en un conjunto de datos. Se hace una extracción manual de las características de entrada y con las misma se entrena un clasificador. Por otro lado, el aprendizaje profundo o Deep Learning permite realizar de manera automática una extracción de las características de la información que se le suministra como entrada (ref. 14).

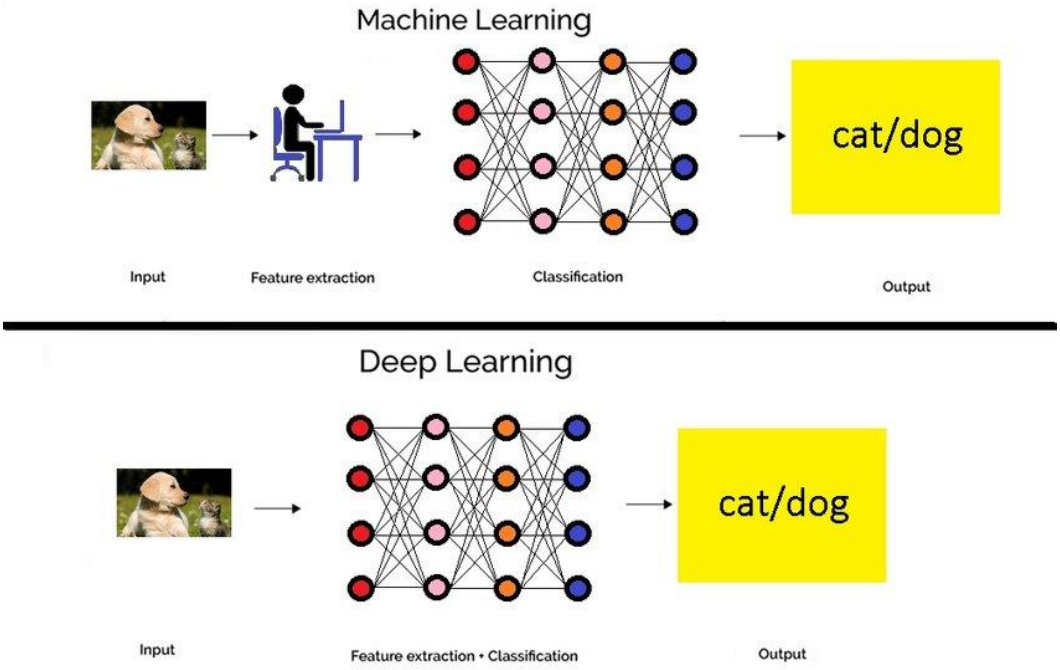


Ilustración 17. Machine Learning vs Deep Learning.

5.4.4 ARQUITECTURAS DE DEEP LEARNING.

Las tres arquitecturas que se utilizan en el Deep Learning son los autocodificadores apilados, las redes neuronales convencionales y las redes recurrentes tipo LSTM (Long Short Term Memory) (ref. 14).

Los autocodificadores apilados se usan para procesamiento de señales de una dimensión (audio) y de dos dimensiones (imágenes). El método consiste en obtener a la salida el mismo dato que a la entrada, para ello la capa oculta de la red deberá generar una codificación que constituirá las características de la información de entrada (ref. 14).

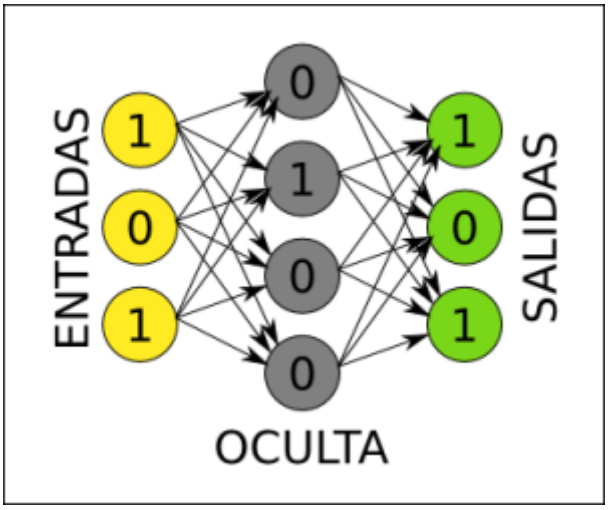


Ilustración 18. Autocodificadores apilados.

Las redes neuronales convencionales son la arquitectura más utilizada en el Deep Learning. Su arquitectura se basa en realizar repetitivamente una operación denominada convolución (ref. 14). En el siguiente apartado (5.4.5) se estudiará con más detenimiento.

Las redes recurrentes tipo LSTM se utilizan en aprendizaje de secuencias para tareas como la predicción del siguiente valor de la secuencia, clasificación de la secuencia de entrada, generación de unas secuencias en función de la entrada o para predecir una nueva secuencia dependiendo de la secuencia de entrada.

5.4.5 REDES NEURONALES CONVENCIONALES.

El desarrollo actual del Deep Learning y de las redes neuronales es gracias a las CNN (Convolutional Neural Networks). Hoy en día no existe ningún sector de la visión artificial donde no se hayan aplicado las redes neuronales convencionales.

El funcionamiento de una CNN consiste en realizar una serie de tareas como la convolución, el zero-padding, el max pooling, entre otros (ref. 14).

La convolución se trata de una de las operaciones más famosas en el procesamiento de señales. En el área de las CNN, esta operación se emplea para señales de dos dimensiones o imágenes (ref. 14).

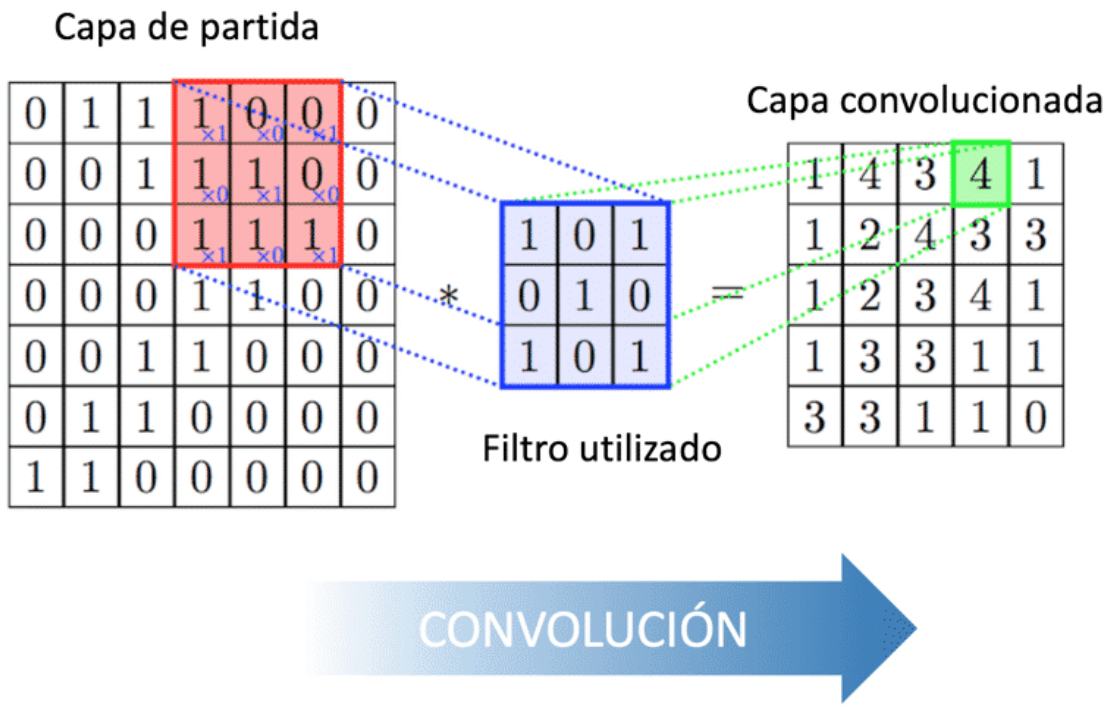


Ilustración 19. Representación gráfica de la convolución.

Se tiene como matriz de entrada una imagen, el filtro con el que se realiza la convolución y la matriz de salida de la nueva imagen. El filtro trabaja sobre una porción de la imagen multiplicando componente a componente los píxeles de la porción de la imagen a procesar con el filtro y haciendo una suma de dichas multiplicaciones (ref. 14).

El zero-padding permite conservar el tamaño original de la imagen sin importar el filtro de convolución aplicada. Para ello, se añaden ceros alrededor de la imagen (ref. 14).

El max pooling tiene como objetivo submuestrear una representación de entrada reduciendo su tamaño. La imagen se divide en regiones del mismo tamaño y para cada región se extrae el valor máximo que corresponderá a un píxel en la imagen resultante (ref. 14).

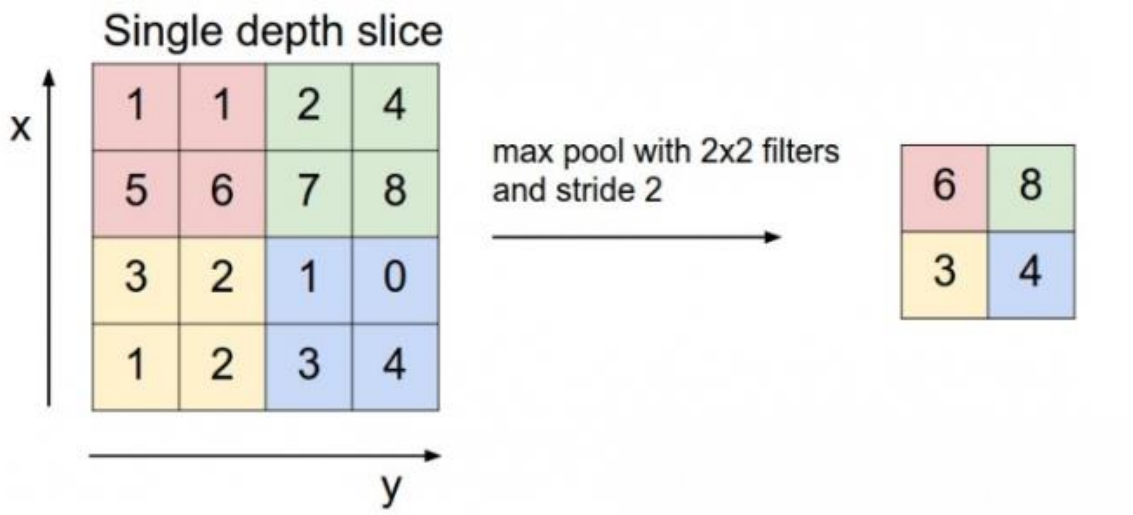


Ilustración 20. Representación gráfica del max pooling.

Por tanto, una red neuronal convolucional presenta la arquitectura que se ilustra en la Ilustración 21.

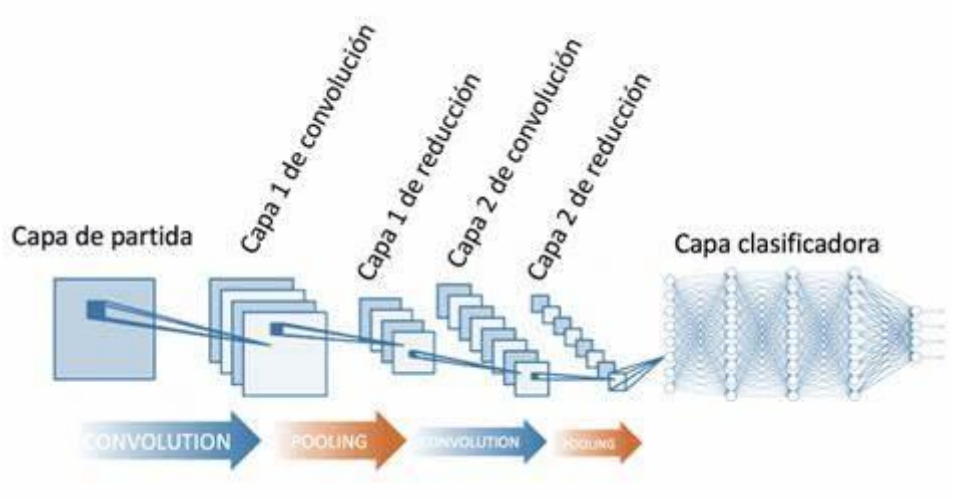


Ilustración 21. Red neuronal convencional.

5.5 ROBÓTICA INDUSTRIAL.

5.5.1 HISTORIA DE LA ROBÓTICA INDUSTRIAL.

En 1920, el término ROBOTICA fue introducido al vocabulario por el escritor Karel Capek (1889-1938) en su obra Rosum's Universal Robots. Esta palabra proviene de "Robotat" que significa trabajar (ref. 16).

Mucho antes, en 1801 se inventó la que se conoce como la primera máquina programable. Se trata del telar y fue inventada por J. Jaquard (ref. 1).

Ilustración 22. Telar de J. Jaquard.

En 1954, George Devol, un inventor estadounidense, desarrolló un brazo primitivo que mediante programación podía realizar unas tareas específicas (ref. 1).

En 1956, Joseph F. Engelberger y George Devol fundan la Consolidates Controls Corporation, que en 1960 se convierte en Unimation, la primera firma fabricante de robots en EE. UU. (ref. 1).

Antes, en 1959, se realiza el primer prototipo de robot industrial. Se introdujo en el comercio el primer robot por Planet Corporation (ref. 1).

En 1961, comienzan las primeras experiencias con robots en el área industrial, utilizando el robot Unimate (desarrollado por George Devol) en la Ford Motors Company para atender una máquina de fundición de troquel (ref. 1).

Ilustración 23. Robot Unimate.

En 1968, comienza la robótica en Japón que rápidamente aventaja a EE. UU. gracias a la marca Nissan. En 1972 se formó la primera asociación robótica del mundo, la Asociación Robótica Industrial de Japón (JIRA) y dos años después el Instituto de Robótica de América (RIA) que en 1984 cambió su nombre por el de Asociación de Industrias Robóticas (ref. 1).

En 1971, en la universidad Standford University se desarrolló un pequeño brazo robótico con accionamiento eléctrico llamado Standford Arm (ref. 1).

Ilustración 24. Brazo robótico Standford Arm.

Dos años más tarde, en 1973, ASEA desarrolla el primer robot completamente eléctrico conocido como el IRB6 (ref. 1).

Ilustración 25. Robot IRB6, primer robot completamente eléctrico.

En 1978, se desarrolló el robot PUMA (Programmable Universal Machine for Assambly) por Unimate. Este robot era capaz de mover un objeto y colocarlo en cualquier lugar y orientación que estuviera a su alcance (ref. 1).

Ilustración 26. Robot PUMA.

Un año más tarde, en 1979, se desarrolló en la Universidad de Yamanashi en Japón el robot SCARA (Selective Compliance Arm for Robotic Assembly) (ref. 1).

Ilustración 27. Robot SCARA.

En 1981, se desarrolló en la Universidad de Carnegle-Mellon un robot de impulsión directa. Este utilizaba motores eléctricos situados en las articulaciones en vez de las transmisiones mecánicas empleadas hasta ese momento (ref. 1).

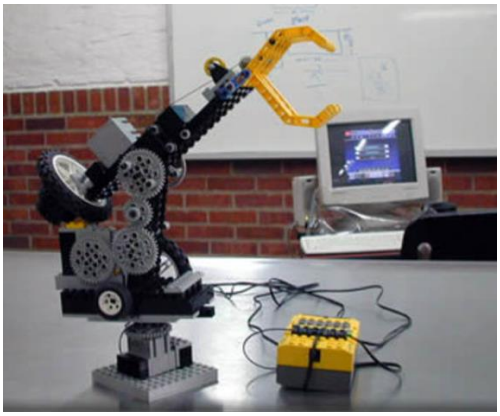


Ilustración 28. Robot de impulsión directa.

A partir de la década de los 90, se comenzaron a desarrollar robots colaborativos o “cobots”. Estos robots estaban diseñados para trabajar junto a los humanos, compartiendo el espacio de trabajo de forma segura y colaborando en tareas específicas (ref. 1).

La robótica industrial ha evolucionado constantemente a lo largo de los años, incorporando nuevas tecnologías y mejorando la eficiencia y seguridad en los procesos de fabricación. A medida que avanza la tecnología, se espera que los robots industriales sigan empleando un papel fundamental en la automatización de la producción y la mejora de la productividad en la industria (ref. 1).

5.5.2 CLASIFICACIÓN DE ROBOTS.

Atendiendo al tipo de actuador, los robots pueden ser:

- Robot Neumático.
- Robot Hidráulico.
- Robot Eléctrico.

Dependiendo de la configuración, se pueden clasificar como:

- Robot Cartesiano.
- Robot Cilíndrico.

- Robot Polar o esférico.
- Robot Articular o antropomórfico.
- Robot SCARA.
- Robot Paralelo.

Atendiendo al tipo de control:

- Robot secuencial.
- Robot controlado por trayectorias.

Atendiendo a la aplicación del robot:

- Robots para manipulación.
- Robots para soldadura.
- Robots para aplicación de materiales como pintura, adhesivos, secantes, etc.
- Mecanización.
- Robots para montaje.
- Paletización.
- Etc.

5.5.3 MORFOLOGÍA DEL ROBOT.

Un robot manipulador está formado por una serie de elementos (brazos, enlaces, segmentos, eslabones o links) unidos mediante articulaciones o ejes (joints) que permiten un movimiento relativo entre cada dos eslabones consecutivos. Este movimiento es producido por los actuadores del robot (motor).

El movimiento de la articulación puede ser:

- De desplazamiento.

- De giro.
- Combinación de ambos.

5.5.3.1 TIPOS DE ARTICULACIONES.

Los principales tipos de articulaciones utilizados en la robótica son:

- Esférica o rótula: permiten la rotación en tres direcciones.

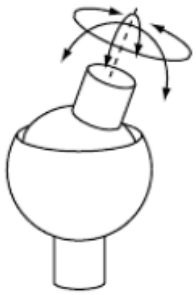


Ilustración 29. Articulación esférica.

- Tornillo: proporciona una rotación y una traslación que es función de la rotación.

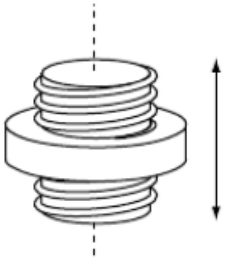


Ilustración 30. Articulación de tornillo.

- Prismática: permite la traslación entre dos eslabones.

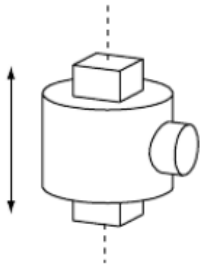


Ilustración 31. Articulación prismática.

- Rotación: permite el giro relativo entre dos eslabones.

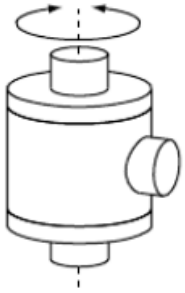


Ilustración 32. Articulación de rotación.

- Cilíndrica: permite un giro y una traslación.

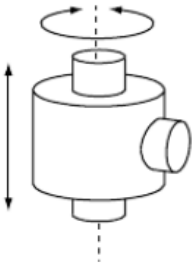


Ilustración 33. Articulación cilíndrica.

En la robótica industrial, las articulaciones más utilizadas son la prismática y la de rotación.

5.5.3.2 CONFIGURACIÓN CINEMÁTICA DEL ROBOT.

Las configuraciones más usuales son:

- Configuración cartesiana: se caracteriza porque sus tres primeras articulaciones son prismáticas. El número de rotaciones posible es cero.

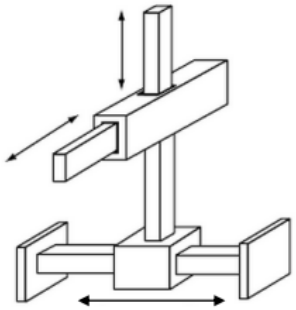


Ilustración 34. Configuración cartesiana.

- Configuración cilíndrica: la primera de las articulaciones es de rotación, mientras que las otras dos son prismáticas. El número de rotaciones posible es uno.

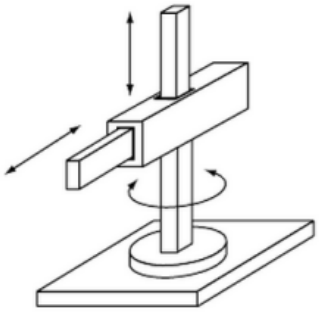


Ilustración 35. Configuración cilíndrica.

- Configuración esférica: las dos primeras articulaciones son de rotación y la tercera prismática. El número de rotaciones posibles son dos.

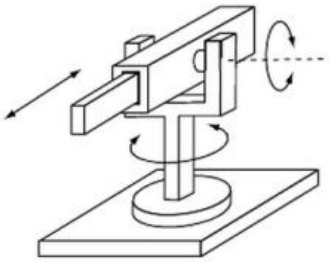


Ilustración 36. Configuración esférica.

- Configuración SCARA: al igual que en la configuración esférica, las dos primeras articulaciones son de rotación y la tercera prismática. El número de rotaciones posible es dos. Aunque tiene la misma configuración que la esférica, es diferente tanto en apariencia como en rango de aplicaciones.

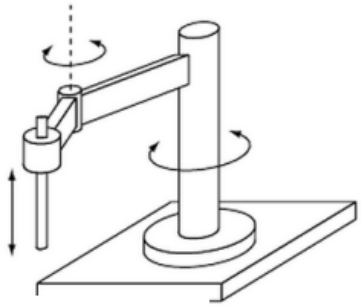


Ilustración 37. Configuración SCARA.

- Configuración antropomórfica: las tres articulaciones son de rotación, siendo el número de rotaciones tres.

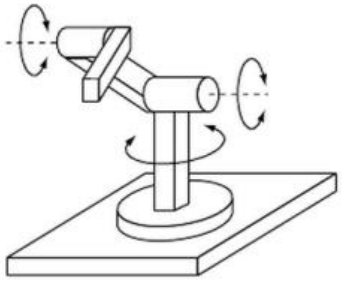


Ilustración 38. Configuración antropomórfica.

5.5.3.3 ELEMENTOS TERMINALES DEL ROBOT.

Los elementos terminales del robot permiten interaccionar directamente con el entorno. Estos normalmente son diseñados específicamente para cada tipo de aplicación lo que suele suponer hasta un 30% del coste total del robot. Pueden ser tanto elementos de aprehensión como herramientas y proporcionan una mayor versatilidad al robot.

Entre los principales tipos de elementos terminales se encuentran:

- Sujeción: pinzas, ventosas, adhesivo, ganchos.

- Operación: pistola de pintura, soldadura, corte, mecanizado.

- Montaje y manipulación: ensamblado, medición, verificación.

5.5.3.4 SENSORES Y ACTUADORES DEL ROBOT.

Los sensores y actuadores del robot forman un papel fundamental en el correcto funcionamiento de este.

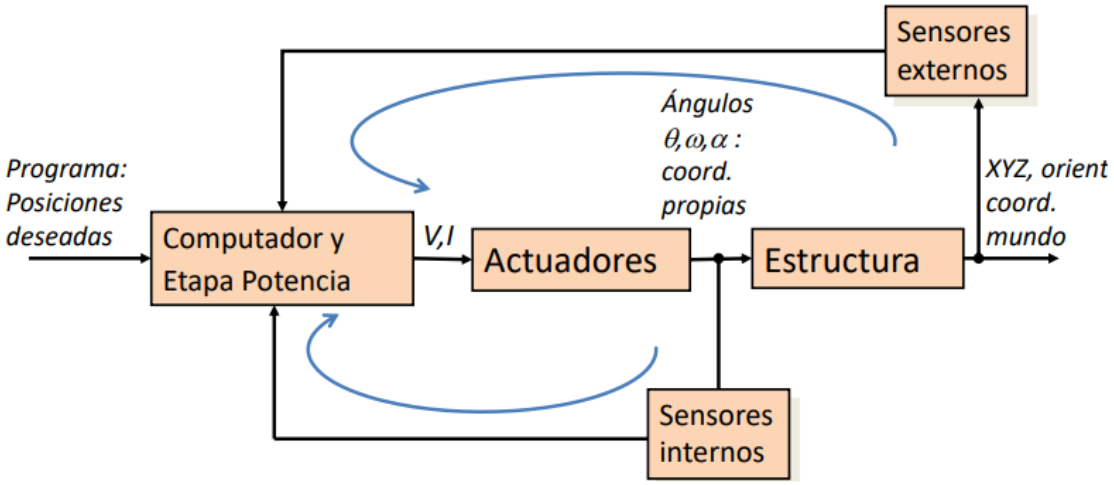


Ilustración 39. Esquema de un robot.

Los actuadores son los encargados de generar el movimiento de los elementos del robot. Las principales características que se deben tener en cuenta son:

- Potencia.
- Controlabilidad.
- Peso y volumen.
- Precisión.
- Velocidad.
- Mantenimiento.
- Coste.

Los actuadores se pueden clasificar en: neumáticos, hidráulicos y eléctricos. Actualmente los más utilizados son los eléctricos (motores DC, AC o motores paso a paso).

Los sensores son los dispositivos que permiten a un robot percibir su entorno y su propio estado. Se pueden clasificar en:

- Sensores internos: dan información sobre el estado del propio robot. Ejemplos: potenciómetros, encoders, resolvers.

- Sensores externos: dan información sobe el entorno del robot. Ejemplos: ultrasonidos, sensores infrarrojos, telémetros láser, cámaras, sensores de fuerza, etc.

5.5.4 CINEMÁTICA Y DINÁMICA DEL ROBOT.

La cinemática y dinámica del robot tienen como objetico gobernar los movimientos del robot necesarios para llevar a cabo una tarea concreta. Esta tarea puede consistir en un movimiento aislado o en un programa de movimientos almacenado en la memoria del computador.

La cinemática y dinámica forman el sistema de control del robot tal y como se ilustra en la Ilustración 32.

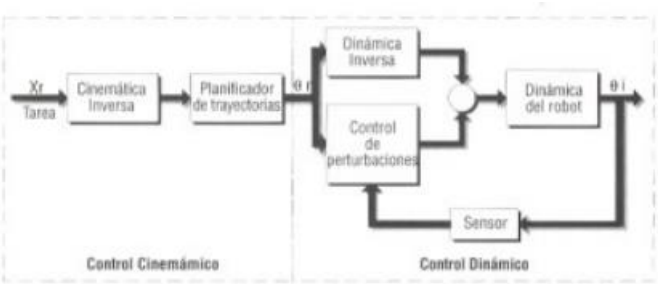


Ilustración 40. Sistema de control de un robot industrial.

5.5.4.1 CINEMÁTICA.

La cinemática es el estudio del movimiento del robot con respecto a un sistema de referencia. Tiene como objetivos:

- Realizar una descripción analítica del movimiento espacial del robot como una función del tiempo.

- Establecer relaciones entre la posición y orientación del extremo del robot en coordenadas del mundo y los valores de sus coordenadas propias (coordenadas articulares).

Para resolver la cinemática del robot, se debe resolver:

- Problema cinemático directo.

- Problema cinemático inverso.

- Matriz Jacobiana.

La relación que existe entre la cinemática directa y la inversa se presenta en la Ilustración 41.

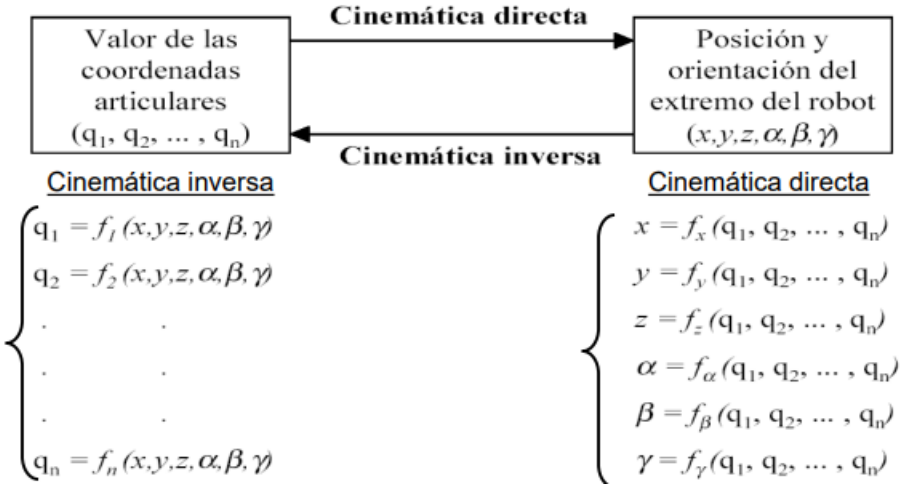


Ilustración 41. Relación entre problema cinemático directo e inverso.

5.5.4.1.1 PROBLEMA CINEMÁTICO DIRECTO.

El objetivo de resolver el problema cinemático directo es determinar la posición y orientación del extremo del robot, con respecto a un sistema de coordenadas de referencia, conocidos los valores de las articulaciones y los parámetros geométricos de los elementos del robot.

Para resolverlo se utilizan dos tipos de métodos:

- Métodos geométricos: se obtienen la posición y orientación del extremo del robot basándose en las relaciones geométricas.

- Método de Denavit-Hartenberg (D-H): método cuyo objetivo es resolver el problema cinemático directo mediante matrices de transformación homogéneas.

5.5.4.1.2 PROBLEMA CINEMÁTICO INVERSO.

El problema cinemático inverso tiene como misión encontrar los valores que deben adoptar las coordenadas articulares del robot para que su extremo se posicione y oriente según una determinada localización espacial.

La resolución de este problema depende de la configuración del robot y pueden existir soluciones múltiples (redundancia) o inexistencia de solución (punto inaccesible).

Para resolverlo se utilizan varios métodos:

- Métodos geométricos.

- Matrices de transformación homogéneas.

- Desacoplo cinemático parcial (sólo para determinados robots de 6 GDL): se intenta obtener una resolución independiente de los grados de libertad que posicionan y de los que orientan y luego se resuelven por separado los 3 GDL de posición y los tres de orientación.

5.5.4.1.3 MATRIZ JACOBIANA.

La matriz Jacobiana es un modelo diferencial que forma parte de la cinemática y cuyo objetico es obtener las relaciones entre las velocidades del movimiento de las articulaciones y las del extremo del robot.

Al igual que en la cinemática del robot, se puede distinguir entre jacobiana directa e inversa y la relación entre ambas se muestra en la Ilustración 42.

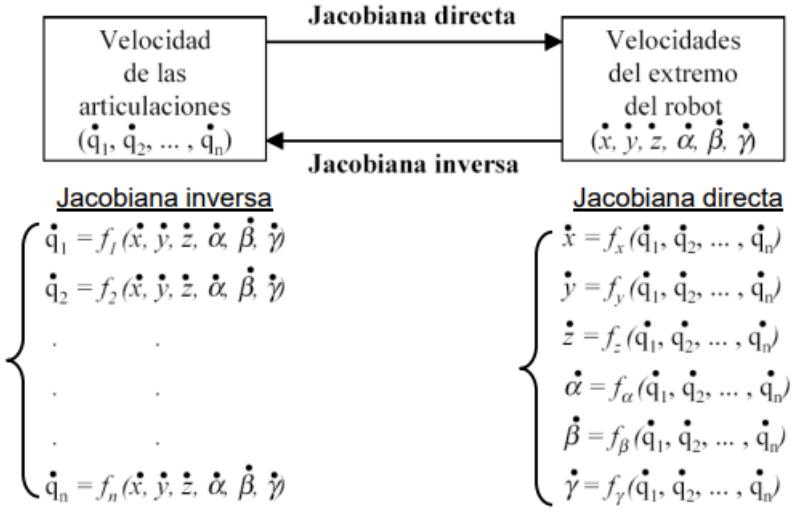


Ilustración 42. Relación entre Jacobiana directa e inversa.

5.5.4.2 DINÁMICA.

La dinámica del robot se ocupa de la relación entre las fuerzas que actúan sobre un cuerpo y el movimiento que en él se origina. Esta relación se obtiene mediante el modelo dinámico que relaciona matemáticamente:

- La localización del robot definida por sus variables articulares o por las coordenadas de localización de su extremo, y sus derivadas: velocidad y aceleración.

- Las fuerzas y pares aplicados en las articulaciones.

- Los parámetros dimensionales del robot, como longitud, masas e inercias de sus elementos.

Se puede distinguir entre:

- Modelo dinámico directo: calcula la evolución temporal de las coordenadas articulares y sus derivadas, en función de las fuerzas y pares que intervienen.

- Modelo dinámico inverso: calcula las fuerzas y pares que intervienen, en función de la evolución temporal de las coordenadas articulares y sus derivadas.

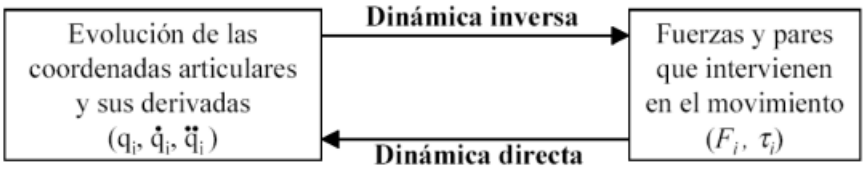


Ilustración 43. Relación entre modelo dinámico directo e inverso.

6 MATERIALES Y HERRAMIENTAS.

6.1 MATERIALES.

6.1.1 LOGITECH WEBCAM PRO-9000.

Para poder realizar el proceso de visión artificial, uno de los materiales importantes es la cámara. En este proyecto se hará uso de la Webcam Pro-9000 de la empresa Logitech.

Ilustración 44. Webcam Pro-9000.

La cámara web ofrece conexión Plug and Play y para conectarla al ordenador dispone de un cable USB.

Físicamente consta de:

- Componentes ópticos Carl Zeiss con enfoque automático.
- Botón de captura de instantáneas.
- Diodo de actividad.
- Micrófono.
- Clip universale.

Entre las características más importantes de este producto se encuentran:

- Espacio mínimo de almacenamiento del disco: 200MB.
- Interfaz: USB.
- Longitud del cable: 1,8m.
- Procesador mínimo: Pentium P4 1,4 GHz.
- RAM mínima: 128MB.
- Requisitos mínimos del sistema: CD-ROM.
- Sistema de lentes: Carl Zeiss.
- Velocidad máxima de cuadro: 30 pps.
- Resolución de captura de vídeo: 1200p.

6.1.2 SOPORTE PARA LA CÁMARA.

Para sostener la cámara a una altura determinada y así poder capturar imágenes de buena calidad, se hace uso de un soporte proporcionado por la empresa Intelitek Inc.

Ilustración 45. Soporte para la cámara.

6.1.3 SCORBOT ER-4U.

Para el desarrollo de este proyecto, y conseguir clasificar cada uno de los objetos, se hace uso del robot Scorbot ER-4U. Se trata de un robot manipulador cuyo diseño está orientado a un entorno académico.

Ilustración 46. Robot Scorbot ER-4U.

Este robot diseñado por la empresa Intelitek Inc. se controla por defecto mediante el programa Scorbase que es un paquete software de control robótico que proporciona una herramienta, de sencillo uso, para la programación y operación de robots. En este proyecto no se hará uso de Scorbase, sino que se utilizará una toolbox de Matlab para el control del robot.

Además del Scorbot ER-4U, Intelitek Inc. proporciona una controladora USB.

Ilustración 47. Controladora USB.

Estos dos equipos se comunican entre sí mediante un cable propiedad del fabricante. Para completar el sistema robótico se hace uso de un ordenador conectado mediante cable USB a la controladora de forma de que el diagrama de bloques sería el de la Ilustración 48.

Ilustración 48. Diagrama de bloques sistema robótico.

6.1.3.1 ESTRUCTURA DEL SCORBOT ER-4U.

El Scorbot ER-4U es un robot manipulador antropomórfico formado por 5 ejes más la pinza. Posee 6 motores de corriente continua y con transmisión mecánica mediante correas y engranajes (ref. 11).

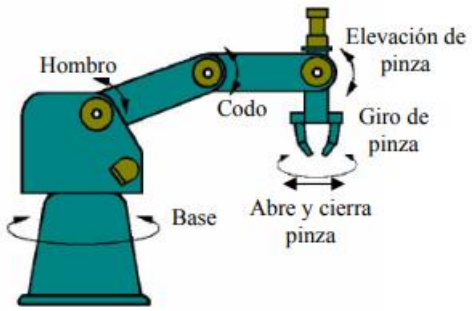


Ilustración 49. Ejes y articulaciones Scorbot ER-4U.

Cada una de estas articulaciones tiene su propia función tal y como se especifica en la Tabla 1.

Tabla 1. Ejes y Articulaciones Scorbot ER-4U. (ref. 11)

Eje	Articulación	Movimiento
1	Base	Rota el cuerpo
2	Hombro	Levanta y baja el ante brazo
3	Codo	Levanta y baja el brazo
4	Elevación de pinza	Levanta y baja la pinza
5	Giro de pinza	Gira la pinza
6	Pinza	Abre y cierra la pinza

Además, menos la pinza, todas tienen un rango de movimiento específico. Estos rangos de movimiento se recogen en la Tabla 2 junto con las velocidades máximas de cada articulación.

Tabla 2. Rangos y velocidades ejes Scorbot ER-4U. (ref. 11)

Eje	Rango	Velocidad
1	310º	20º/sec
2	158º	26,3º/sec
3	260º	26,3º/sec
4	260º	83º/sec
5	Sin límite (mecánicamente); ±570º (eléctricamente)	106º/sec

Así el brazo robótico tiene un área de operación tal y como se muestra en la Ilustración 50.

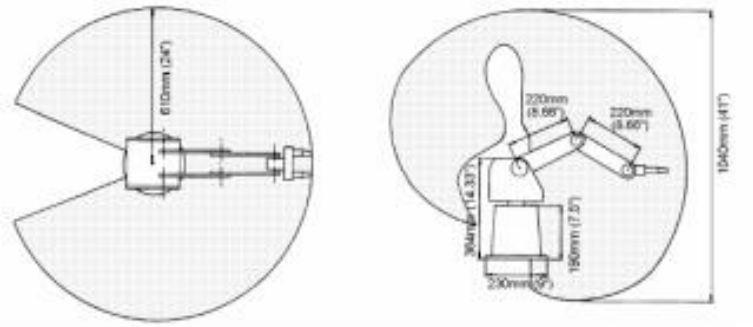
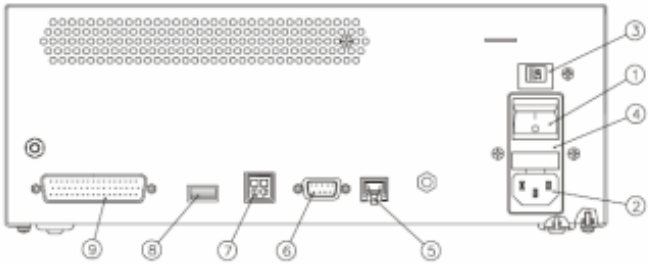


Ilustración 50. Volumen de trabajo Scorbot ER-4U. (ref. 11)

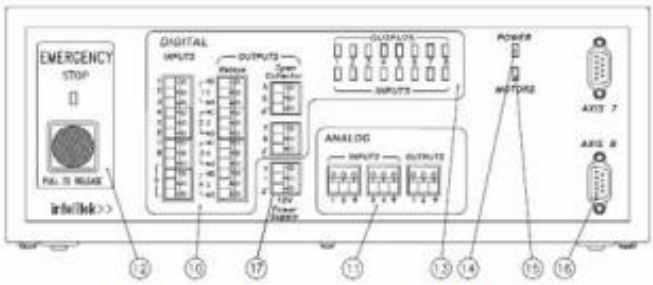
6.1.3.2 CONTROLADORA USB.

La controladora USB (Ilustración 47) posee una fuente de alimentación que suministra 24V a los seis motores y a dos accesorios (ejes 7 y 8). Además, posee entradas y salidas tanto analógicas como digitales de forma de que se pueden conectar dispositivos como sensores, actuadores, etc (ref. 10).



1. Interruptor de energía
2. Alimentador 110/220 VAC
3. Selector del Voltaje de línea
4. Fusible de entrada AC (110V, 2A; 220V, 1A)
5. Conexión «Teach pendant»
6. Puerto RS232, para uso futuro
7. Conector de emergencia remota
8. Conector USB para el PC
9. Conector DB62 para el brazo robot

Ilustración 51. Parte trasera de la controladora USB.



10. Terminales de Entrada/Salida digital
11. Terminales de Entrada/Salida análoga
12. Botón de Emergencia y LED indicador
13. LED indicadores de las Entradas/Salidas digital
14. LED indicador de la potencia
15. LED indicador de motor
16. Conectores DB9 para los ejes 7 y 8
17. Voltaje de 12V auxiliar. 0.1 A máx.

Ilustración 52. Parte delantera de la controladora USB

6.2 HERRAMIENTAS SOFTWARE.

6.2.1 PYTHON.

Python es un lenguaje de programación de alto nivel que permite trabajar de forma más rápida e integrar sistemas de forma más eficiente (ref. 23).

Ilustración 53. Logo Python.

Entre las principales características de Python (ref. 23), destacan:

- Es un lenguaje de programación orientado a objetos.
- Se trata de un lenguaje de programación de código abierto.
- Es fácil de aprender.
- Es un lenguaje de programación integrado, permite ejecutar el código línea a línea.

En este proyecto Python forma un papel fundamental, ya que toda la programación de detección de objetos y la clasificación de ellos se realiza con este lenguaje de programación.

6.2.2 VISUAL STUDIO CODE.

Visual Studio Code se trata de un editor de código fuente desarrollado por Microsoft para Windows, Linux, macOS y Web. Es gratuito y de código abierto e incluye control integrado de Git, soporte para la depuración, resaltado de sintaxis, finalización inteligente de código, fragmentación y refactorización de código (ref. 24).

Ilustración 54. Logo de Visual Studio Code.

Visual Studio Code permite la instalación de entornos de programación como Python, Matlab, C/C++, etc. Por lo que para nuestro proyecto será de gran utilidad a la hora de desarrollar el código a ejecutar.

También se debe destacar que permite seleccionar el intérprete que se desee, es decir, se puede tener varias versiones, por ejemplo, de Python instaladas y utilizar la que mejor se ajuste a nuestras necesidades en cada momento.

6.2.3 MATLAB.

Matlab es una plataforma de programación y cálculo numérico utilizada por millones de ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos (ref. 15).

Ilustración 55. Logo Matlab.

Matlab combina un entorno de escritorio perfeccionado para el análisis iterativo y los procesos de diseño con un lenguaje de programación que expresa las matemáticas de matrices y arrays directamente (ref. 15).

En este proyecto Matlab es necesario para el uso de una toolbox capaz de controlar el robot ScorBot-er 4u. Esta toolbox permite también adquirir información de los sensores del robot y visualizar los movimientos de este.

6.2.4 LIBRERÍAS UTILIZADAS.

6.2.4.1 OPENCV.

OpenCV (Open Computer Vision) es la biblioteca libre más popular de visión artificial. Se trata de una biblioteca desarrollada por Intel, de libre uso, multiplataforma, documentada y explicada (ref. 22).

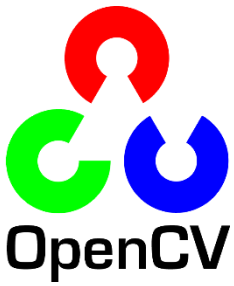


Ilustración 56. Logo de OpenCV.

Entre las principales áreas de aplicación de OpenCV (ref. 22), se encuentran:

- Detección de objetos.
- Reconocimiento facial.
- Reconocimiento de gestos.
- Características 2D y 2D.
- Segmentación.
- Robótica móvil.

En este proyecto, OpenCV se trata de la librería más importante ya que a través de ella se realiza todo el proceso de visión artificial con funciones que permiten desde el calibrado de la cámara hasta la propia detección de los objetos.

6.2.4.2 NUMPY.

NumPy es una biblioteca exclusiva para el lenguaje de programación Python. Esta permite crear vectores y matrices multidimensionales, además de poseer funciones matemáticas de alto nivel (ref. 21).

Ilustración 57. Logo de NumPy.

6.2.4.3 SCORBOT TOOLBOX.

Esta toolbox de Matlab permite la simulación del Intelitek ScorBot-ER 4u. Su interfaz permite controlar el robot ScorBot, adquirir información de los sensores y visualizar los movimientos de este (ref. 12).

Es compatible con cualquier versión de Matlab y además es multiplataforma, permitiendo su uso en Windows, macOS y Linux (ref. 12).

Esta librería de libre acceso posee funciones básicas para la inicialización del robot (conexión con el servidor, realizar el Home del robot), funciones para mover el robot a un punto determinado, etc. (ref. 12).

6.2.4.4 USO DE MATLAB CON PYTHON.

Como se ha comentado anteriormente, el código principal va a ser desarrollado con el entorno de programación Python, y para ello utilizaremos como editor de código fuente Visual Studio Code. Pero la toolbox que permite el control del robot está desarrollada en Matlab por lo que es necesaria la búsqueda de una solución que permita llamar desde Python a funciones creadas en Matlab.

Para utilizar funciones de Matlab en Python se hace uso de la API del motor Matlab para Python. Esta proporciona un paquete que permite que Python llame a Matlab como motor de cálculo.

Las condiciones para utilizar este motor de Matlab son:

- Tener instalado Matlab.

- Compatibilidad entre la versión de Python y la versión de Matlab.

6.2.5 YOLOV3.

YOLO (You Only Look Once, o en español, “sólo miras una vez”) es un algoritmo de detección de objetos en tiempo real (ref. 17).

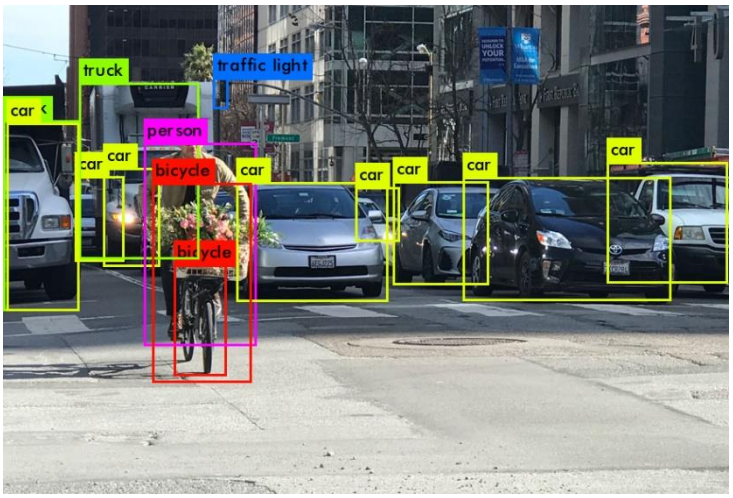


Ilustración 58. Ejemplo de uso de Yolov3.

Se trata de una red neuronal convolucional (CNN) que permite la detección de objetos. Las CNN son sistemas que pueden procesar imágenes de entrada como matrices estructuradas de datos y reconocer patrones entre ellas. YOLO posee la ventaja de que es mucho más rápido y preciso que otras redes. A continuación, se muestra una comparación con otros detectores (ref. 17).

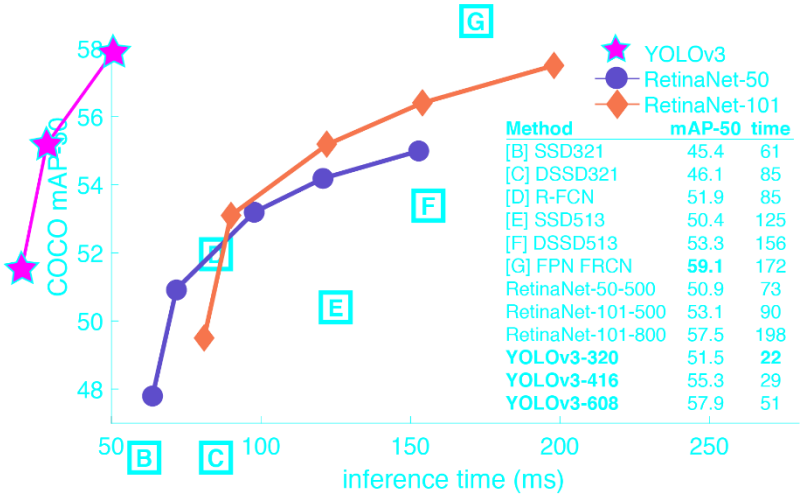


Ilustración 59. Comparación de YOLOv3 con otros detectores.

En este proyecto se utiliza YOLOv3-416.

Los sistemas de detección previa reutilizan clasificadores o localizadores para realizar la detección. Aplican el modelo a una imagen en varias ubicaciones y escalas. Las regiones de puntuación más alta de la imagen se consideran detecciones (ref. 17).

En YOLOv3 se aplica una sola red neuronal a la imagen completa. Esta red divide la imagen en regiones y predice cuadros delimitadores y probabilidades para cada región. Estos cuadros delimitadores se ponderan por las probabilidades predichas (ref. 17).

7 DESARROLLO, IMPLEMENTACIÓN Y RESULTADOS.

El desarrollo del proyecto puede ser dividido en dos partes:

- Visión artificial.

- Robótica industrial.

Dentro de la parte de visión artificial, las tareas a realizar son las que se muestran en el siguiente diagrama de bloques (Ilustración 60).

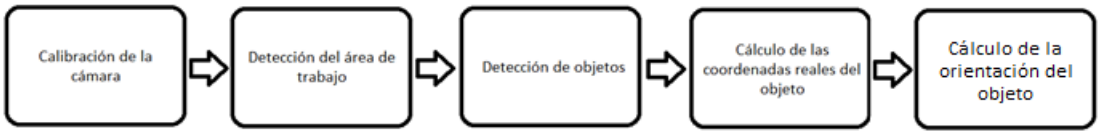


Ilustración 60. Diagrama de bloques Visión Artificial.

En el área de la robótica industrial las tareas a realizar se ilustran en la Ilustración 61.

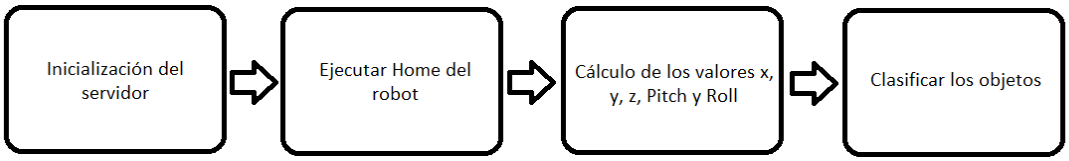


Ilustración 61. Diagrama de bloques Robótica Industrial.

7.1 VISIÓN ARTIFICIAL.

Dentro del área de visión artificial, el objetivo fundamental es conseguir detectar un determinado objeto en un área de trabajo específico y poder hallar las coordenadas y orientación real de este objeto.

Para realizar estas tareas se hará uso de la biblioteca de OpenCV, la cual proporciona funciones básicas para controlar la cámara y realizar operaciones sobre una imagen capturada.

7.1.1 CALIBRACIÓN DE LA CÁMARA.

Como se ha estudiado anteriormente en el apartado teórico de calibración de la cámara, el objetivo es determinar las características internas, geométricas y ópticas de la cámara y la posición en 3D del marco de la cámara con respecto a un sistema de coordenadas, obteniendo así una relación entre coordenadas tridimensionales de los objetos con sus correspondientes coordenadas bidimensionales en la imagen.

Para la calibración de la cámara se vio que existen dos técnicas dependiendo del patrón utilizado, en este proyecto se hace uso de la calibración coplanar, siendo así el patrón de calibración utilizado el que se muestra la Ilustración 62.

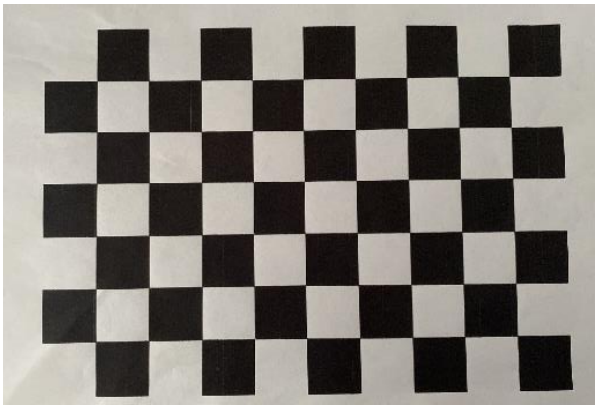


Ilustración 62. Patrón de calibración de la cámara.

Se trata de una cuadrícula donde los puntos de interés son las esquinas internas. Cada uno de los cuadrados tienen las mismas dimensiones.

OpenCV facilita la operación de calibración de la cámara ya que nos proporciona dos funciones importantes:

- *cv2.calibrateCamara()*: función que permite obtener tanto la matriz de la cámara como los coeficientes de distorsión.

- *cv2.undistort()*: permite eliminar la distorsión de la cámara.

Para realizar la calibración se creará un código dónde el primer paso será obtener los puntos 3D del mundo real y 2D de la imagen y almacenarlos en una lista.

La obtención de estos puntos se lleva a cabo detectando las esquinas del patrón de calibración. Para ello se utiliza la función *cv2.findChessboardCorners(frame, calibration_board_size)*, donde *frame* se trata de la imagen capturada y *calibration_board_size* las dimensiones del patrón que en este caso es 9x6 (destacar que estas dimensiones son las esquinas internas). Esta función nos devuelve dos valores: *found* y *corners.* La variable *found* es de tipo booleano e indica si se ha encontrado alguna esquina o no. La variable *corners* indica las esquinas que se han detectado.

Mediante la función *cv2.drawChessboardCorners(frame, calibration_board_size, corners, found)* se puede visualizar los puntos obtenidos y de los cuales se van a hacer uso en la calibración.

Ilustración 63. Puntos de interés sobre patrón de calibración.

Se debe realizar captura de las imágenes para la obtención de los puntos. Cuantas más capturas se realicen y con el patrón de calibración en diferentes posiciones, mayores datos se tendrán de la imagen y mejores resultados se obtendrán de la calibración.

Una vez obtenidas todas las capturas y almacenados todos los puntos y datos de interés, estos se pasan a la función *cv2.calibrateCamera(object_points_list, image_points_list, frame.shape[:2], None, None)*, donde *object_points_list* son los puntos 3D del objeto e *image_points_list* son los puntos 2D de la imagen. Esta función devuelve cuatro parámetros: *camera_matrix* (matriz de la cámara), *dist_coeffs* (coeficientes de distorsión), *rvec* y *tvec* (vectores de rotación y traslación).

```
Camera matrix:
[[2.86120728e+03 0.00000000e+00 2.50073580e+02]
 [0.00000000e+00 2.80382267e+03 2.97862494e+02]
 [0.00000000e+00 0.00000000e+00 1.00000000e+00]]

Distortion coefficient:
[[ 2.05076484e+00 -9.24640708e+01 -8.57320086e-03 -1.02507075e-01
   2.69555458e+03]]
```

Ilustración 64. Ejemplo valores de la matriz de la cámara y coeficientes de distorsión.

De estos cuatro valores se hace uso tanto de la matriz de la cámara como de los coeficientes de distorsión. Estos dos parámetros se pasan a la función *cv2.undistort(frame, camera_matrix, dist_coeffs)* la cuál realiza una ecuación con términos no lineales que permite eliminar el efecto de la distorsión no deseada. A continuación, se muestra la diferencia existente entre una imagen calibrada y una no calibrada.

Ilustración 65. Imagen calibrada.

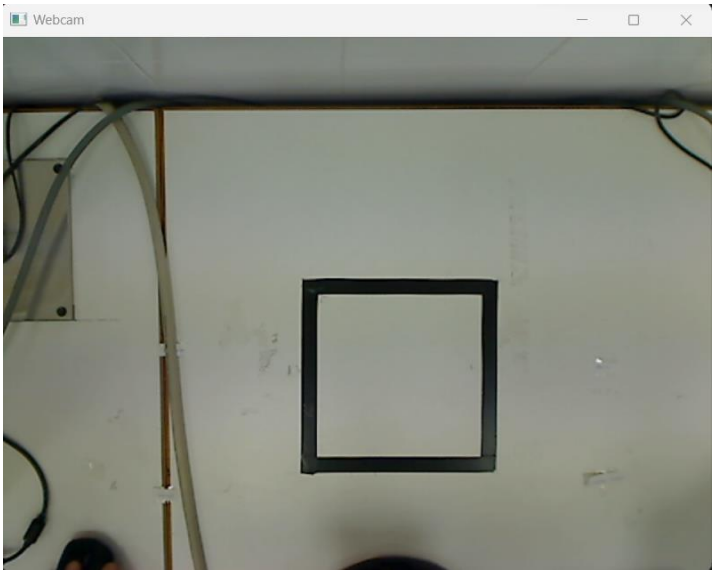


Ilustración 66. Imagen sin calibrar.

Se puede observar como en la Ilustración 65 aparece en los laterales de la imagen una pequeña curva que tal y como se vio en el apartado teórico y comparando con la Ilustración 12, se trata de la distorsión radial.

Cada vez que la cámara sea movida de sitio o el entorno del área de trabajo sea distinto, se debe de realizar una nueva calibración y obtener nuevos parámetros.

También hay que destacar que para no tener que estar obteniendo los parámetros de calibración cada vez que se ejecute el código sin haber

desplazado la cámara (es decir, la cámara se encuentra en la misma posición), estos se pueden almacenar, por ejemplo, en un archivo de extensión txt.

Generalmente la calibración en un área de trabajo adecuado resulta 100% efectivo, pero en el caso de este proyecto se encuentra la limitación de que la pared es capturada por la cámara y no permite la calibración por esa zona, además de que hace que no todo lo que es grabado esté a la misma distancia con respecto a la cámara, haciendo que el resultado de la calibración no sea muy bueno, pero sigue siendo efectivo para conseguir el objetivo que se pretende alcanzar.

7.1.2 DETECCIÓN DEL ÁREA DE TRABAJO.

Antes de la detección del área de trabajo se debe determinar la posición real de esta con respecto a la cámara y el robot.

El área de trabajo debe estar en una posición que cumpla las siguientes dos condiciones:

- Debe ser capturada por la cámara.

- Debe estar dentro del volumen de trabajo del robot, es decir, el robot debe ser capaz de trabajar en cualquier punto de esta área de trabajo.

Una vez obtenida una posición que cumpla las dos anteriores condiciones, se debe dibujar esta área sobre el tablero. Puesto que el tablero es de color blanco se utilizará un color negro para delimitar el área de trabajo y así la cámara pueda distinguir bien las líneas delimitadoras.

Esta área puede ser de diferentes formas: cuadrada, rectangular, triangular, etc. Por simplicidad se tomará como área de trabajo un cuadrado. En el caso de utilizar otra forma geométrica se debe tener en cuenta varios aspectos a la hora de recortar la imagen con OpenCV o cuando se vaya a realizar el cálculo de la relación entre píxeles y centímetros (se estudiará más adelante).

Ilustración 67. Área de trabajo vista general.

Una vez está definido el área de trabajo, debe ser detectado por la cámara para conocer en qué píxel se encuentra cada esquina y así poder recortar la imagen capturada.

Hay que destacar que esta área debe ser obtenida sobre una imagen ya calibrada y libre de distorsión.

Para detectar cada esquina se hace uso de la función *cv2.goodFeaturesToTrack(gris, 8, 0.01,10)* donde *gris* es la imagen capturada, pero en escala de grises convertida por la función *cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)*, *8* es el número máximo de esquinas detectable, *0.01* el nivel de calidad y *10* la mínima distancia. Esta función devuelve las coordenadas x e y en píxeles donde se encuentra cada esquina. Dibujando estos valores sobre la imagen capturada se podría visualizar donde se encuentra cada esquina.

Ilustración 68. Detección de esquinas.

Pero aparece un inconveniente y es que el área que graba la cámara posee esquinas que no interesan y entorpecen una buena detección del área de trabajo. Incluso, en ciertas ocasiones, puesto que la calibración no llega a ser del todo buena, la propia corrección de la imagen distorsionada hace que aparezca esquinas que no existen. Un ejemplo de ello, puede ser la Ilustración 68, donde en la esquina superior izquierda aparece una esquina a causa de la calibración.

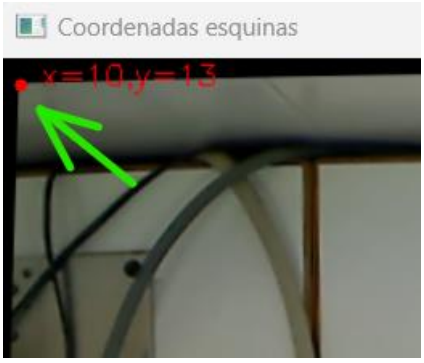


Ilustración 69. Detección de esquinas no deseadas.

Para solucionar este problema se puede plantear dos alternativas:

- La primera sería aumentar el número de esquinas a detectar, pero resulta ser un método no muy efectivo, aunque es perfectamente funcional.

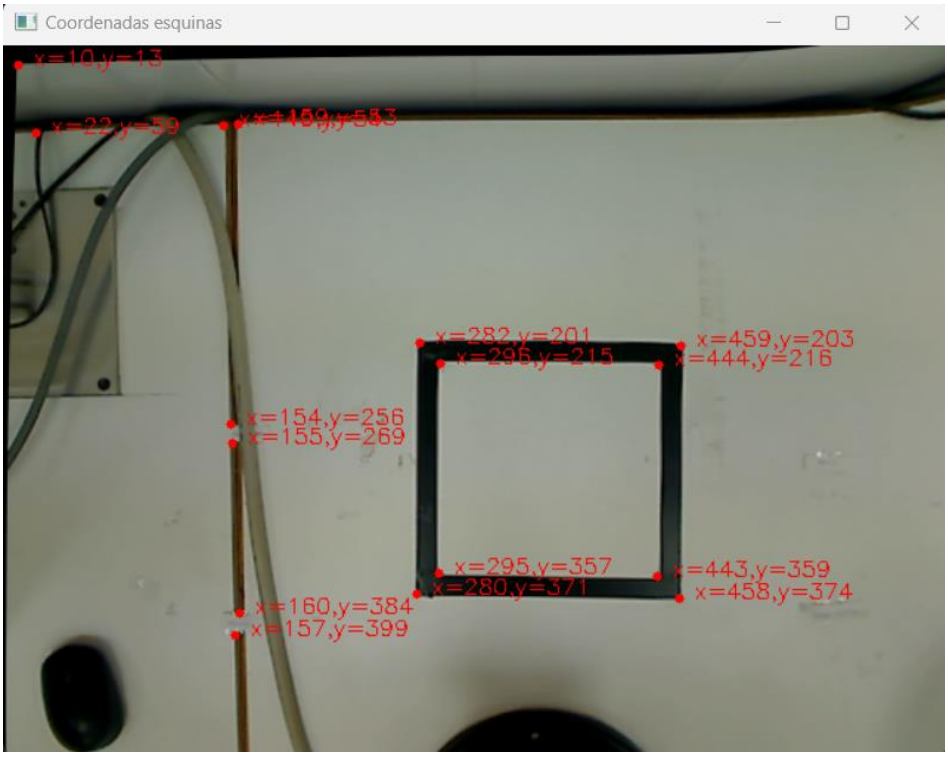


Ilustración 70. Detección de esquinas alternativa 1.

Como se puede apreciar en la Ilustración 70, aparece cada una de las coordenadas correspondientes a las esquinas del área de trabajo. Sin embargo, siguen apareciendo esquinas indeseadas que no son de interés.

- El segundo método es el utilizado en este proyecto. Se trata de realizar un recorte previo de la zona de interés eliminando así zonas capturadas por imagen que no son de uso. Para llevar a cabo este método se abre una ventana de la imagen capturada y sobre ella se desplaza el ratón determinando así el área de interés. La ventana se crea mediante la función *cv2.namedWindows("Recorte de imagen")* y a esta se la asocia una función de retrollamada del ratón *cv2.setMouseCallback("Recorte de imagen", seleccionar_roi)*, donde *seleccinar_roi* se define como una función que espera un evento en la ventana creada y guarda las coordenadas de inicio y fin de arrastre del ratón. Hay que destacar que estas coordenadas deben de ser tenidas en cuenta más tarde a la hora de calcular las posiciones reales por lo que es aconsejable que sean guardadas en un archivo txt.

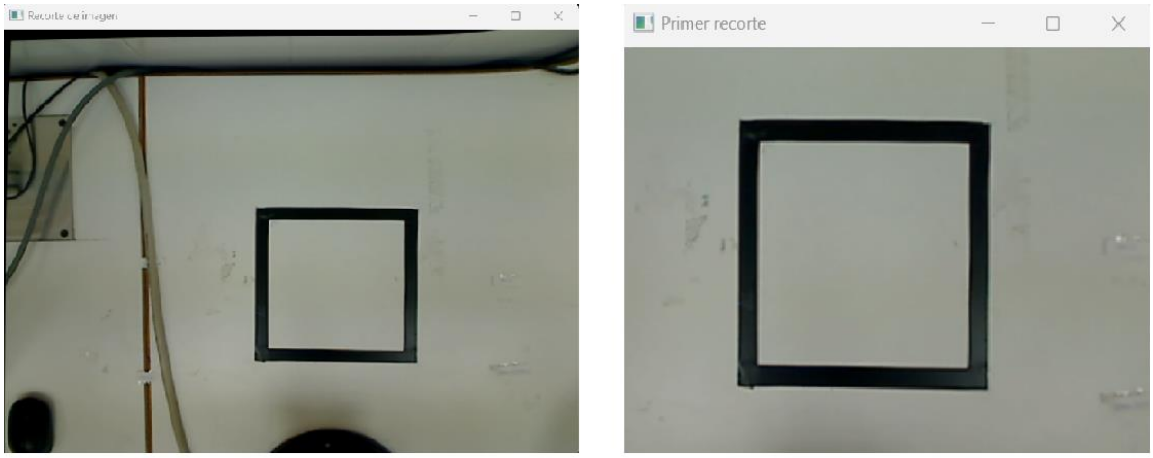


Ilustración 71. Primer recorte de imagen: a) Antes del recorte b) Después del recorte.

Una vez recortada la imagen capturada, sobre esta se lanza el programa de detección de esquinas obteniendo como resultado el que se muestra en la Ilustración 72.

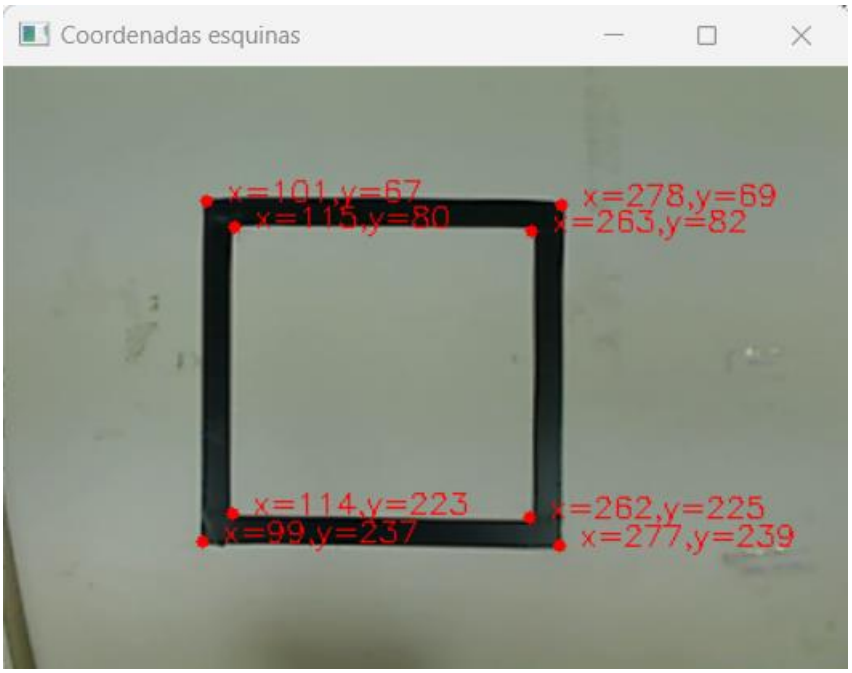


Ilustración 72. Detección de esquinas alternativa 2.

Como se puede observar solamente se detectan los puntos de interés por lo que se presenta como una solución favorable y que no incrementa mucho la dificultad de la programación presentada.

Estas coordenadas resultantes, al igual que en el primer recorte que se realizó deben ser guardadas en un archivo txt.

Como resultado final se obtiene el área de trabajo ya recortado y sobre el que se puede comenzar a detectar objetos.

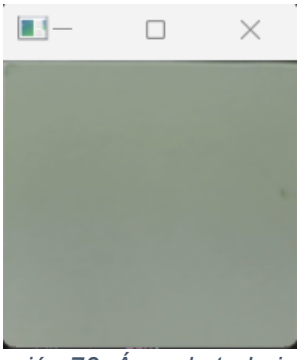


Ilustración 73. Área de trabajo final.

7.1.3 DETECCIÓN DE OBJETOS.

Para la detección de objetos se empleará un modelo pre-entrenado de YOLOv3 y el módulo DNN (Deep Neural Networks) de OpenCV que permitirá reconocer hasta 80 clases de objetos diferentes.

Antes de comenzar con la programación, se debe instalar previamente los tres archivos necesarios especificados en el Anexo 1: *yolov3.weights*, *yolov3.cfg* y *coco.names*.

Hay que destacar que el texto que se desarrolla a continuación está basado en el código que se recoge en el Anexo 2: Código del Proyecto.

Una vez instalados los tres archivos se deben importar al código especificando la ruta de cada uno de ellos.

La variable *labels* almacenará una lista de todas las etiquetas leídas en el archivo coco.names cuyo contenido se observa en la Ilustración 74.

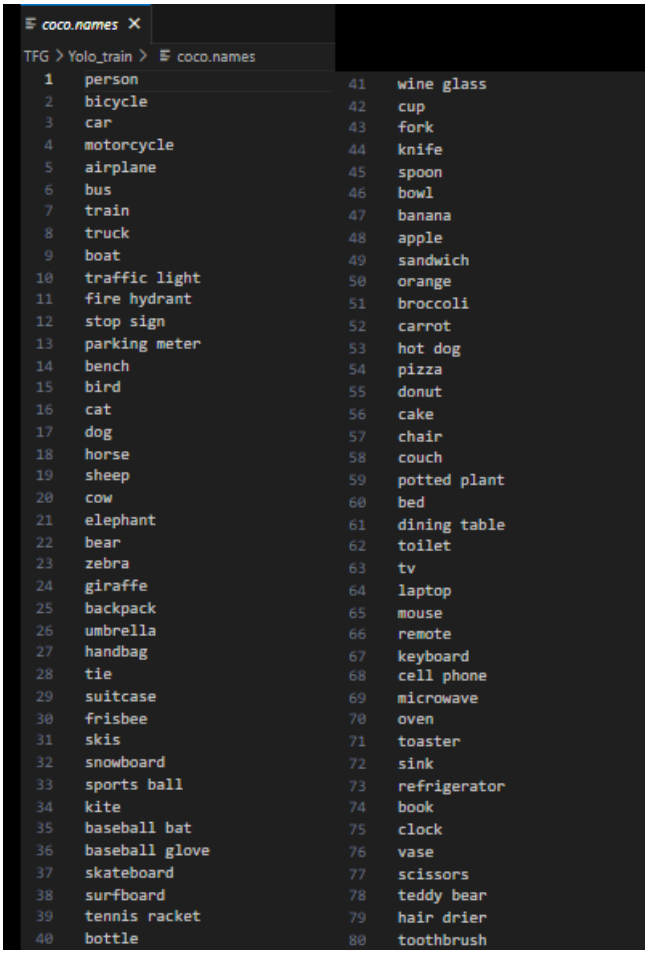


Ilustración 74. Contenido coco.names.

Para cargar el modelo se utiliza la función *cv2.dnn.readNetFromDarknet(modelConfiguration, modelWeights)* donde, *modelConfiguration* es la variable que guarda la ruta del archivo *yolov3.cfg* y *modelWeights* la ruta de *yolov3.weights*. Esta función será llamada por la variable *net*.

Mediante *net.getnamesLayers()* se obtendrán los nombres de todas las capas de la red y de todas ellas se extraen tres, debido a que YOLOv3 presenta una estructura de tres escalas para la detección.

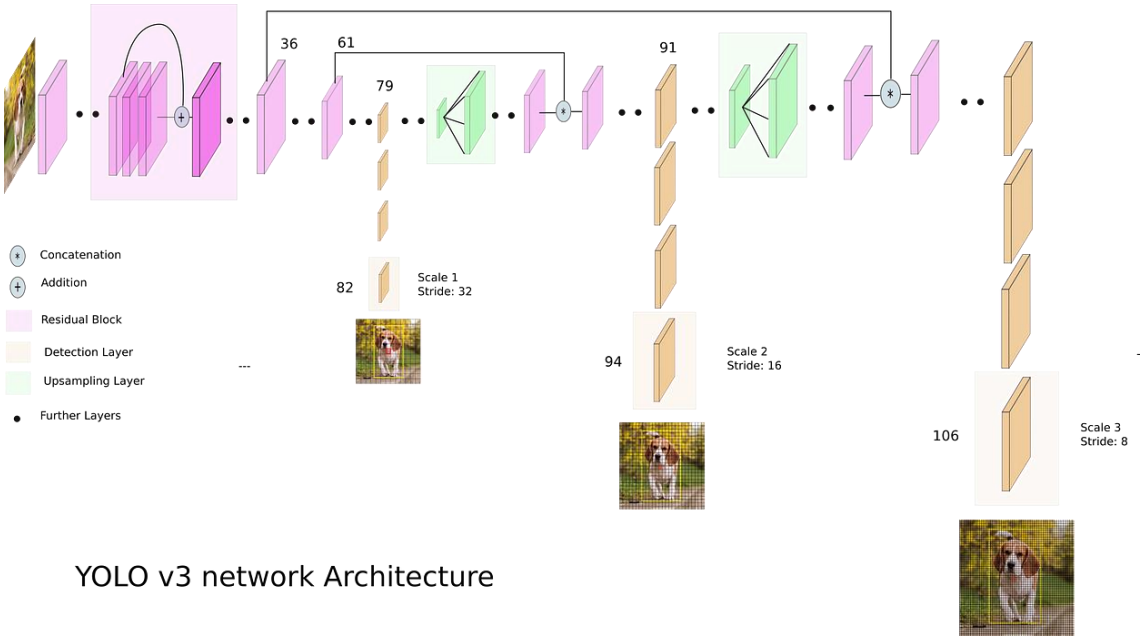


Ilustración 75. Arquitectura YOLOv3.

```
('conv_0', 'bn_0', 'leaky_1', 'conv_1', 'bn_1', 'leaky_2', 'conv_2', 'bn_2', 'leaky_3', 'conv_3', 'bn_3', 'leaky_4', 'shortcut
_4', 'conv_5', 'bn_5', 'leaky_6', 'conv_6', 'bn_6', 'leaky_7', 'conv_7', 'bn_7', 'leaky_8', 'shortcut_8', 'conv_9', 'bn_9', 'l
eaky_10', 'conv_10', 'bn_10', 'leaky_11', 'shortcut_11', 'conv_12', 'bn_12', 'leaky_13', 'conv_13', 'bn_13', 'leaky_14', 'conv
_14', 'bn_14', 'leaky_15', 'shortcut_15', 'conv_16', 'bn_16', 'leaky_17', 'conv_17', 'bn_17', 'leaky_18', 'shortcut_18', 'conv
_19', 'bn_19', 'leaky_20', 'conv_20', 'bn_20', 'leaky_21', 'shortcut_21', 'conv_22', 'bn_22', 'leaky_23', 'conv_23', 'bn_23',
'leaky_24', 'shortcut_24', 'conv_25', 'bn_25', 'leaky_26', 'conv_26', 'bn_26', 'leaky_27', 'shortcut_27', 'conv_28', 'bn_28',
'leaky_29', 'conv_29', 'bn_29', 'leaky_30', 'shortcut_30', 'conv_31', 'bn_31', 'leaky_32', 'conv_32', 'bn_32', 'leaky_33', 'sh
ortcut_33', 'conv_34', 'bn_34', 'leaky_35', 'conv_35', 'bn_35', 'leaky_36', 'shortcut_36', 'conv_37', 'bn_37', 'leaky_38', 'co
nv_38', 'bn_38', 'leaky_39', 'conv_39', 'bn_39', 'leaky_40', 'shortcut_40', 'conv_41', 'bn_41', 'leaky_42', 'conv_42', 'bn_42'
, 'leaky_43', 'shortcut_43', 'conv_44', 'bn_44', 'leaky_45', 'conv_45', 'bn_45', 'leaky_46', 'shortcut_46', 'conv_47', 'bn_47'
, 'leaky_48', 'conv_48', 'bn_48', 'leaky_49', 'shortcut_49', 'conv_50', 'bn_50', 'leaky_51', 'conv_51', 'bn_51', 'leaky_52', '
shortcut_52', 'conv_53', 'bn_53', 'leaky_54', 'conv_54', 'bn_54', 'leaky_55', 'shortcut_55', 'conv_56', 'bn_56', 'leaky_57', '
conv_57', 'bn_57', 'leaky_58', 'shortcut_58', 'conv_59', 'bn_59', 'leaky_60', 'conv_60', 'bn_60', 'leaky_61', 'shortcut_61', '
conv_62', 'bn_62', 'leaky_63', 'conv_63', 'bn_63', 'leaky_64', 'conv_64', 'bn_64', 'leaky_65', 'shortcut_65', 'conv_66', 'bn_6
6', 'leaky_67', 'conv_67', 'bn_67', 'leaky_68', 'shortcut_68', 'conv_69', 'bn_69', 'leaky_70', 'conv_70', 'bn_70', 'leaky_71',
 'shortcut_71', 'conv_72', 'bn_72', 'leaky_73', 'conv_73', 'bn_73', 'leaky_74', 'shortcut_74', 'conv_75', 'bn_75', 'leaky_76',
 'conv_76', 'bn_76', 'leaky_77', 'conv_77', 'bn_77', 'leaky_78', 'conv_78', 'bn_78', 'leaky_79', 'conv_79', 'bn_79', 'leaky_80
', 'conv_80', 'bn_80', 'leaky_81', 'conv_81', 'permute_82', 'yolo_82', 'identity_83', 'conv_84', 'bn_84', 'leaky_85', 'upsampl
e_85', 'concat_86', 'conv_87', 'bn_87', 'leaky_88', 'conv_88', 'bn_88', 'leaky_89', 'conv_89', 'bn_89', 'leaky_90', 'conv_90',
 'bn_90', 'leaky_91', 'conv_91', 'bn_91', 'leaky_92', 'conv_92', 'bn_92', 'leaky_93', 'conv_93', 'permute_94', 'yolo_94', 'ide
ntity_95', 'conv_96', 'bn_96', 'leaky_97', 'upsample_97', 'concat_98', 'conv_99', 'bn_99', 'leaky_100', 'conv_100', 'bn_100',
'leaky_101', 'conv_101', 'bn_101', 'leaky_102', 'conv_102', 'bn_102', 'leaky_103', 'conv_103', 'bn_103', 'leaky_104', 'conv_10
4', 'bn_104', 'leaky_105', 'conv_105', 'permute_106', 'yolo_106')
```

Ilustración 76. Capas de la red.

Para encontrar estas tres capas de salida se emplea net.getUnconnectedOutLayers() y se itera hasta conseguirlas.

```
['yolo_82', 'yolo_94', 'yolo_106']
```

Ilustración 77. Capas de salida YOLOv3.

Una vez realizado estos pasos, será necesario leer la imagen de entrada que previamente debe ser calibrada y recortada tal y como se detalla en los apartados anteriores. Se utilizará la función *cv2.dnn.blobFromImage(frame, 1/255.0, (416,416), swap=True, crop=False)* donde se especificará la imagen de entrada (*frame*), el scale factor (*1/255.0*), el tamaño de la imagen que la red espera (*416,416*), el valor de *swap* que debe ser *True* debido a que OpenCV por

defecto lee imágenes en BGR y no RGB por lo que se tiene que cambiar el orden de los canales R y B, y la variable *crop* que indica si la imagen se recortará después de cambiar el tamaño. Esta función llevará a obtener una colección de 4 dimensiones (el número de imágenes, el número de canales, y el ancho y alto de la imagen) que será la que se deba pasar a la red.

El resultado de esta función será guardado en la variable *blob* y será la entrada de la red. Para establecerlo como entrada se utiliza *net.setInput(blob)*.

Se empleará *net.forward(outputLayer)* donde *outputLayer,* guarda las 3 capas de salida, para que *blob* se propague hacia delante en la red. Esta función la guardaremos en la variable *output* y cada vez que se realice una detección devolverá un vector de 85 elementos donde los cuatro primeros son las coordenadas x e y del centro del objeto y el ancho y alto del mismo. El quinto elemento se trata de la confianza asociada al cuadro delimitador, es decir, asociada a los cuatro valores anteriores y los 80 elementos restantes son la confianza de las clases.

Bastará que por cada detección realizada se escoja la confianza mayor y sobre esta guardar las variables x e y para determinar el centro del objeto.

Hay que destacar que estos valores x e y son coordenadas en píxeles y será necesario realizar una conversión a centímetros para determinar la posición del objeto en el mundo real.

7.1.4 CÁLCULO DE LAS COORDENADAS X E Y REALES.

El procedimiento para el cálculo de las coordenadas x e y reales es bastante sencillo. Simplemente se debe conocer la relación que existe entre píxeles en la imagen y centímetros en el mundo real.

Para ello si se recapitula y se vuelve al apartado 7.1.2, en él se obtenían las coordenadas x e y de cada esquina. Si se conoce la distancia que existe entre cada esquina tanto para el eje x como para el eje y, y se conoce la distancia real entre esquinas, entonces se puede obtener una relación píxeles/centímetros.

Los pasos que se deben seguir para obtener esta relación son:

- Obtener la distancia real entre esquinas: con una simple regla medimos la distancia que existe tanto en el eje x como en el eje y. Ejemplo: $x = 18{,}8\ cm$ e $y = 18{,}8\ cm$.

- Obtener la distancia en píxeles entre esquinas: se obtienen cuatro distancias, una para cada lado del área de trabajo como muestra la Ilustración 66.

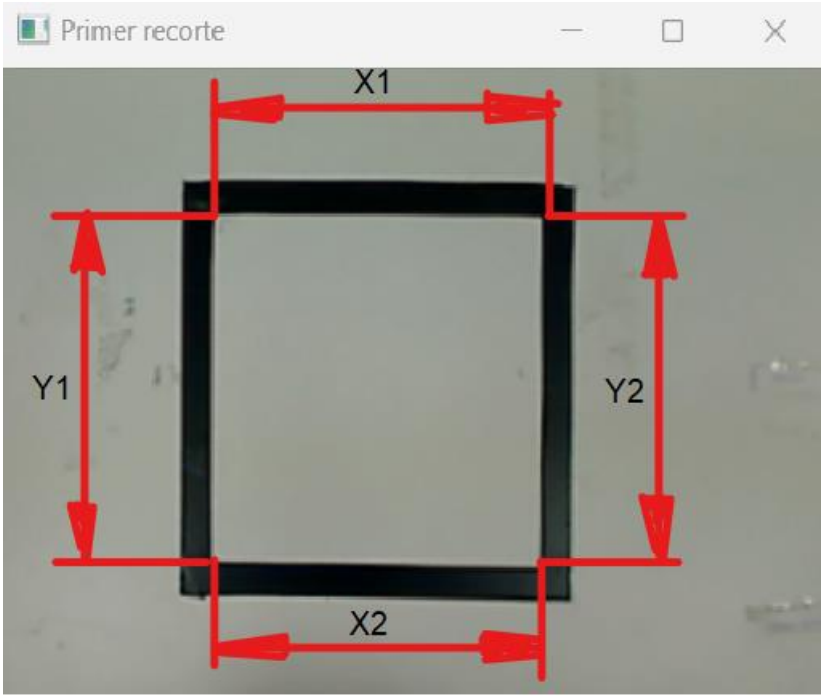


Ilustración 78. Distancias X1, X2, Y1 e Y2.

Para obtener estas distancias, simplemente hay que restar las diferentes coordenadas (recordar que están guardadas en el segundo archivo txt creado, por lo que solo habría que leer este archivo para poder obtener cada coordenada), es decir:

$$distancia\_pixel\_x1 = coordenada\_2 - coordenada\_0$$

Ecuación 4. Distancia en píxeles X1.

$$distancia\_pixel\_x2 = coordenada\_6 - coordenada\_4$$

Ecuación 5. Distancia en píxeles X2.

$$distancia\_pixel\_y1 = coordenada\_5 - coordenada\_1$$

Ecuación 6. Distancia en píxeles Y1.

$$distancia\_pixel\_y2 = coordenada\_7 - coordenada\_3$$

Ecuación 7. Distancia en píxeles Y2.

Estos valores de coordenadas corresponden a:

- coordenada_0: coordenada x esquina superior izquierda.
- coordenada_1: coordenada y esquina superior izquierda.
- coordenada_2: coordenada x esquina superior derecha.
- coordenada_3: coordenada y esquina superior derecha.
- coordenada_4: coordenada x esquina inferior izquierda.
- coordenada_5: coordenada y esquina inferior izquierda.
- coordenada_6: coordenada x esquina inferior derecha.
- coordenada_7: coordenada y esquina inferior derecha.

- Una vez obtenidas estas distancias, se puede obtener la relación entre píxeles y centímetros dividiendo cada distancia en píxeles entre la distancia real.

$$relacion\_x1 = \frac{distancia\_pixel\_x1}{x}$$

Ecuación 8. Relación píxeles/cm X1.

$$relacion\_x2 = \frac{distancia\_pixel\_x2}{x}$$

Ecuación 9. Relación píxeles/cm X2.

$$relacion\_y1 = \frac{distancia\_pixel\_y1}{y}$$

Ecuación 10. Relación píxeles/cm Y1.

$$relacion\_y2 = \frac{distancia\_pixel\_y2}{y}$$

Ecuación 11. Relación píxeles/cm Y2.

- Calculadas estas relaciones y puesto que el área de trabajo se trata de un cuadrado, se podría realizar una media de todas estas relaciones para obtener un sólo valor.

$$relacion\_pixelcm = \frac{(relacion\_x1+relacion\_x2+relacion\_y1+relacion\_y2)}{4}$$

Ecuación 12. Relación píxeles/cm final.

- Con esta relación, detectado un objeto en cualquier posición con coordenadas en píxeles x e y, podemos obtener las coordenadas reales dividiendo la distancia x o y entre la relación píxel/centímetro.

Las operaciones anteriores solamente son de aplicación para áreas de trabajo con forma de cuadrado. Si el área fuera un rectángulo se debe calcular dos relaciones píxel/centímetros, una para el eje x y otra para el eje y, de forma de que cuando se obtenga las coordenadas x e y en píxeles del objeto, a la coordenada x se le aplique su correspondiente relación y a la coordenada y la suya, siendo ambas relaciones diferentes.

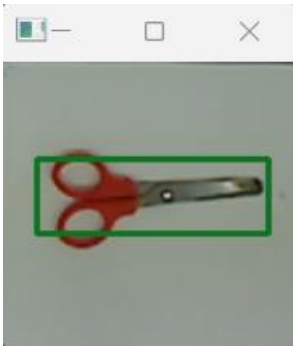


Ilustración 79. Objeto en unas coordenadas determinadas.

Para un objeto colocado en la posición determinada por la Ilustración 79, el resultado sería el que se observa en la Ilustración 80.

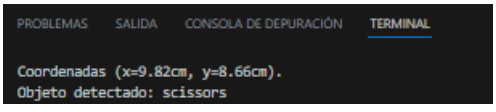


Ilustración 80. Resultado de la obtención de las coordenadas x e y reales.

Se debe destacar la gran precisión, oscilando los valores de las coordenadas en un rango menor de 0,1 cm, lo que es una precisión muy grande para un proceso como el llevado a cabo en este proyecto.

7.1.5 CÁLCULO DE LA ORIENTACIÓN DEL OBJETO.

Para realizar el cálculo de la orientación del objeto, es decir, el ángulo que forma este objeto con respecto a la horizontal de la imagen capturada por la cámara, se hace uso de una función proporcionada por OpenCV conocida como *cv2.minAreaRect()*. El objetivo de esta función es determinar los contornos de un objeto posicionado en el área de trabajo. A partir de la detección del contorno devuelve las coordenadas en píxeles del centro del contorno, el ancho y alto del contorno y además el ángulo de orientación. Puesto que las coordenadas x e y ya han sido obtenidas antes, esta función solamente se utilizará para obtener el ángulo.

Hay que destacar que, para el cálculo del ángulo la imagen que se debe pasar a la función *cv2.minAreaRect()* no debe contener más de un objeto ni cualquier cosa que pueda ser detectada como contorno. Si ocurriera eso el contorno del objeto no se detectaría con claridad y daría lugar a un error en el cálculo del ángulo.

Para evitar este fenómeno, el área de trabajo a la hora de calcular el ángulo de orientación ha sido reducido en 5 píxeles por cada esquina para que en ningún caso pueda llegar a capturarse la delimitación del área de trabajo. Esta distancia no afecta a la detección de objetos ya que se utiliza un nuevo frame capturado desde la cámara y además la distancia de 5 píxeles es casi inexistente en distancia real debido a que el área de trabajo elegido es menor de 20cm, siendo 5 píxeles equivalente a 0,646 cm.

Ilustración 81. Objeto en una orientación determinada.

Colocado un objeto en la posición determinada por la Ilustración 81, el resultado sería el que se ilustra en la Ilustración 82.

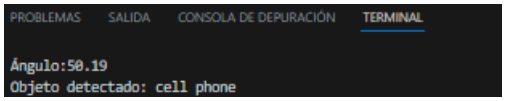


Ilustración 82. Resultado del cálculo de la orientación.

7.2 ROBÓTICA INDUSTRIAL.

El objetivo de uso del robot ScorBot ER-4u es el de clasificar los objetos. Dado un ángulo de orientación y unas coordenada x e y del objeto reales, el robot debe ser capaz de coger este objeto y clasificarlo en su correspondiente depósito.

En este apartado se hace uso de la toolbox de Matlab para el control del robot y la API del motor de Matlab para poder llamar funciones propias de Matlab desde Python.

7.2.1 INICIALIZACIÓN DEL SERVIDOR.

Para inicializar el robot, previamente se debe ejecutar ScorbotServer.exe y Matlab.

La configuración de la comunicación con el robot se realiza con la función *ScorInit()*, la cual carga los DLLs, inicializa la comunicación USB con la controladora del robot y habilita el control de este.

7.2.2 EJECUTAR HOME DEL ROBOT.

El home del robot tiene como objetivo encontrar la posición inicial del mismo. Para ello cada una de sus articulaciones, una a una, van desplazándose por todo su recorrido hasta dar con la posición inicial marcada por unos pequeños interruptores.

Para ejecutar el home se hace uso de la función *ScorHome()*.

Hay que comentar que la toolbox tiene otra función denominada *ScorGoHome()* que permite lleva directamente al robot a su posición de inicio sin necesidad de buscar el home del robot cada vez que se quiera que esté en su posición inicial. Aunque previamente se deberá de haber ejecutado la función *ScorHome()* al menos una vez.

7.2.3 CÁLCULO DE LOS VALORES X, Y, Z, PITCH Y ROLL.

La coordenada x indica la posición final del efector en el eje x, la coordenada y en el eje y, y la coordenada z en el eje z. El Roll indica la rotación del efector son el eje x, y el Pitch la rotación con respecto al eje y.

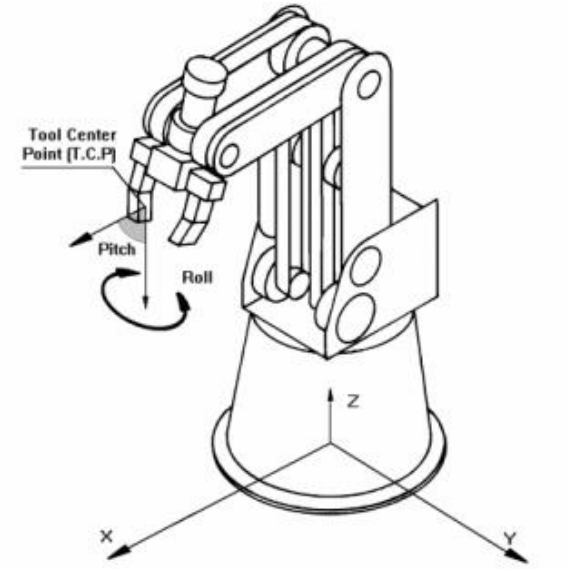


Ilustración 83. Ejes x, y, z. Pitch y Roll

La determinación de estos valores se realiza de forma similar al cálculo de las coordenadas reales del objeto conocidas estas en píxeles tal y como se veía en el apartado 7.1.4.

Previamente se debe conocer cuál es la posición del robot en cada extremo del área de trabajo, es decir, que valores tiene las variables x, y, z, Pitch y Roll (aunque tanto el Pitch como el Roll no son importantes en este aspecto) en cada esquina.

Manualmente se desplaza el robot por estas posiciones y se obtienen los datos que se observan en la ilustración 84.

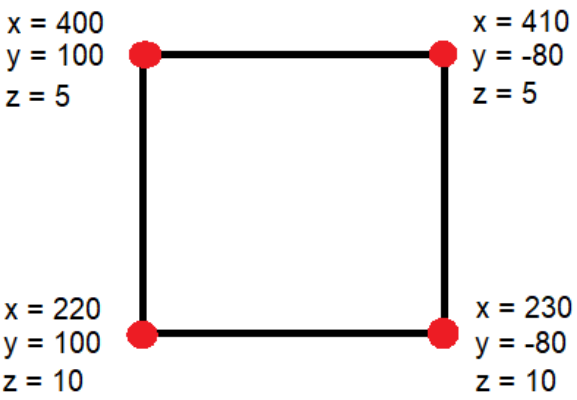


Ilustración 84. Coordenadas del robot en cada esquina del área de trabajo.

Conociendo las distancias del área de trabajo y la distancia que existe entre esquinas para el robot se puede calcular una relación que relacione ambos parámetros.

Esta relación es aplicable a las variables x e y. Destacar que el eje x en el mundo real (coincidente con el eje x en píxeles de la cámara) es el eje y del robot, y el eje y en el mundo real (coincidente con el eje y en píxeles de la cámara) es el eje x del robot.

La variable z dependerá de la altura de cada objeto y se deberá configurar dependiendo de las características de este. Aun así, también se le deberá aplicar una relación puesto que el valor aumenta cuanto más cerca se encuentra la pinza de la base del robot.

En el caso del Pitch siempre tomará el valor -1.5 rad que corresponde a que la pinza del robot esté de forma perpendicular al objeto.

El valor del Roll dependerá del ángulo devuelto por la detección de objetos. Este valor en radianes indicará la orientación que deberá tener la pinza. Es importante tener en cuenta de que el ángulo devuelto no es el mismo que la orientación de la pinza, sino que se debe restar π/2 para que la orientación de la pinza sea perpendicular al objeto

Calculados los cinco valores, la posición final del efector quedará definida y solamente quedará realizar el problema cinemático inverso para poder alcanzar esa posición.

7.2.4 CLASIFICACIÓN DE LOS OBJETOS.

Conocidos los parámetros que debe tomar el robot para llegar a una posición determinada, solamente se necesita funciones que permitan este movimiento.

Algunas funciones básicas que aporta la toolbox para el control del robot y que tiene bastante utilidad en este proyecto son:

- *ScorSetGripper()*: establece la configuración de la pinza. Open (pinza abierta) o Close (pinza cerrada).

- *ScorSetXYZPR()*: realiza el movimiento del robot a una posición determinada por las variables x, y, z, Pitch y Roll. Se encarga de resolver el problema cinemático directo. Dada una posición real, obtiene la configuración de las articulaciones del robot para alcanzar dicha posición.

- *ScorWaitForMove()*: se ejecuta una espera hasta que se completa el movimiento del robot.

- *ScorSetSpeed()*: establece la velocidad del robot.

Con estas funciones se puede ejecutar cualquier movimiento del robot necesario para coger o dejar un objeto en una determina posición.

Colocado un objeto en una cierta posición dentro del área de trabajo, el robot alcanzará fácilmente dicha posición como se muestra en la Ilustración 85.

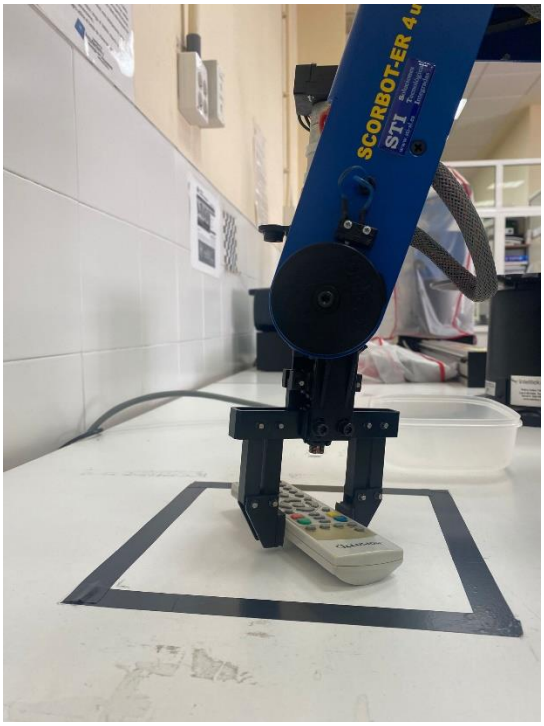


Ilustración 85. Robot cogiendo objeto.

El último paso sería especificar en qué posición se encuentra el depósito donde se va a almacenar cada objeto para llevar el robot hacia esa posición.

Ilustración 86. Robot clasificando objeto.

8 CONCLUSIÓN.

El proyecto ha demostrado la viabilidad y eficacia del uso del Deep Learning, la visión artificial y la robótica industrial para la detección y clasificación de objetos.

Se ha logrado implementar un algoritmo capaz de reconocer diferentes objetos en tiempo real y manipularlos mediante el control de un brazo robótico.

La combinación de estas tres grandes tecnologías presenta un enfoque prometedor para la automatización y la interacción hombre-máquina en un sector como es el industrial que se encuentra en continuo desarrollo.

Al tener la capacidad de reconocer y clasificar objetos de forma autónoma el brazo robótico puede realizar tareas específicas según las necesidades del entorno. Por lo tanto, este suceso ofrece posibilidades en aplicaciones industriales, logísticas y domésticas.

Sería relevante destacar que, aunque se haya conseguido alcanzar los objetivos que se plantearon, aún hay margen de mejora y oportunidades de expansión del proyecto. Por un lado, se puede seguir optimizando el modelo de Deep Learning para conseguir una mayor precisión y velocidad de detección de objetos. Esto implicaría recopilar más datos de entrenamiento y explorar arquitecturas de redes neuronales más avanzadas. Por otro lado, se podría incorporar más sensores, de proximidad o cámaras en 3D que permitan una mejora de la percepción y entendimiento del entorno del robot e incluso considerar la ampliación de las capacidades del brazo robótico realizando movimientos más complejos.

Concluyendo, el proyecto ha demostrado una aplicación prometedora que debe seguir siendo desarrollada e investigada para conseguir grandes avances en la automatización e interacción de robots en diferentes entornos.

9 BIBLIOGRAFÍA.

(ref. 1) ALMEIDA, G. (2009). *UNIDAD I. FUNDAMENTOS GENERALES DE LA ROBÓTICA.*

(ref. 2) ESCOLANO RUIZ, F., CAZORLA QUEVEDO, M. Á., ALFONSO GALIPIENSO, M. I., COLOMINA PARDO, O., & LOZANO ORTEGA, M. Á. (n.d.). *Inteligencia Artificial. Modelos, Técnicas y Áreas de Aplicación* (Thomson).

(ref. 3) GONZÁLEZ MARCOS, A., MARTÍNEZ DE PISÓN ASCACÍBAR, F. J., PERNÍA ESPINOZA, A. V., ALBA ELÍAS, F., CASTEJÓN LIMAS, M., ORDIERES MERÉ, J., & VERGARA GONZÁLEZ, E. (2006). *Técnicas y algoritmos básicos de Visión Artificial*. Universidad de La Rioja, Servicio de Publicaciones.

(ref. 4) HOWARD, J., & GUGGER, S. (2020). *Deep Learning for Coders with fastai & PyTorch* (First edition). O'Reilly Media, Inc.

(ref. 5) HOYOS GUTIÉRREZ, J. G., CARDONA ARISTIZÁBAL, J. E., CAPACHO VALBUENA, L. M., & OROZCO, L. F. (n.d.). Técnicas de Calibración de Cámaras para visión estéreo y reconstrucción. *XV SIMPOSIO DE TRATAMIENTO DE SEÑALES, IMÁGENES Y VISIÓN ARTIFICIAL-STSIVA 2010*.

(ref. 6) HUANG, T. S. (1996). *Computer Vision: Evolution and Promise*.

(ref. 7) HUBEL, D. H., & WIESEL, T. N. (1959). RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX. In *J. Physiol. (I959)* (Vol. 48).

(ref. 8) IBM. (n.d.-a). *¿Qué es el Machine Learning?* Retrieved June 26, 2023, from https://www.ibm.com/mx-es/analytics/machine-learning

(ref. 9) IBM. (n.d.-b). *¿Qué es la visión artificial?* Retrieved May 22, 2023, from https://www.ibm.com/es-es/topics/computer-vision#citation2

(ref. 10) INTELITEK. (n.d.-a). *Manual de usuario Controladora USB*. Retrieved July 19, 2023, from https://downloads.intelitek.com/Manuals/Robotics/ER-4u/Controller-USB-H.pdf

(ref. 11) INTELITEK. (n.d.-b). *Manual de usuario Scorbot ER-4u*. Retrieved July 19, 2023, from https://downloads.intelitek.com/Manuals/Robotics/ER-4u/ER_4u_B.pdf

(ref. 12) KELLEHER, J. D. (2019). *Deep Learning*.

(ref. 13) KUTZER, M. (2023). *Kutzer/ScorbotToolbox*. https://github.com/kutzer/ScorBotToolbox

(ref. 14) LÓPEZ SOTELO, J. A. (2021). *Deep Learning. Teoría y aplicaciones.* (Primera edición). Alpha Editorial S.A.

(ref. 15) MATHWORKS. (n.d.). *Matlab*. Retrieved July 19, 2023, from https://es.mathworks.com/products/matlab.html

(ref. 16) OLIER CAPARROSO, I., Avilés, O., & Hernández Bello, J. (1999). Una introducción a la Robótica Industrial. *Revista de La Facultad de Ingeniería*.

(ref. 17) REDMON, J., & FARHADI, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv*.

(ref. 18) REYES CORTÉS, F. (2011). *Robótica. Control de Robots Manipuladores* (Primera edición).

(ref. 19) UNIE. (2023, May 8). *Principales ramas de la Inteligencia Artificial*. https://www.universidadunie.com/blog/ramas-inteligencia-artificial

(ref. 20) VISHVJIT S, N. (1994). A guided tour of computer vision. *Addison-Wesley*.

(ref. 21) WIKIPEDIA. (n.d.-a). *Numpy*. Retrieved July 19, 2023, from https://es.wikipedia.org/wiki/NumPy

(ref. 22) WIKIPEDIA. (n.d.-b). *OpenCV*. Retrieved July 19, 2023, from https://es.wikipedia.org/wiki/OpenCV

(ref. 23) WIKIPEDIA. (n.d.-c). *Python*. Retrieved July 19, 2023, from https://es.wikipedia.org/wiki/Python

(ref. 24) WIKIPEDIA. (n.d.-d). *Visual Studio Code*. Retrieved July 19, 2023, from https://es.wikipedia.org/wiki/Visual_Studio_Code