COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MASTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

TRABAJO DE FINAL DE MASTER

SISTEMA DE VISIÓN ARTIFICIAL BASADO EN UN DATASET SINTÉTICO PARA UNA APLICACIÓN DE PICK AND PLACE

Autor: Ignacio Ortiz de Zúñiga Mingot

Directores:

Álvaro Jesús López López

Ignacio de Rodrigo Tobías

Madrid

July de 2022

Este trabajo fue escrito con LaTeX y compilado en TeXmaker usando la distribución TeXLive -2013. Las familias de fuentes usadas son Bitstream Charter, Utopia, Bookman, and Computer Modern. A menos que se indique lo contrario, todas las figuras fueron creadas por el autor usando Lucidchart®, draw.io® y Python®.

Índice general

1. Introducción **1**

2. Estado de la cuestión **5**

2.1. Procesado de la imagen 5

2.1.1. Filtros, detección de borde y detección de formas 5

2.1.2. Redes neuronales convolucionales 10

2.2. Robótica industrial 12

3. Metodología de trabajo **15**

3.1. Motivación 15

3.2. Objetivos 15

3.3. Arquitectura del sistema 16

3.4. Herramientas 17

3.5. Cronograma 18

4. Generación de un dataset **21**

4.1. Estructura del dataset 22

4.2. Dataset real 23

4.3. Dataset sintético 23

4.3.1. Herramientas 23

4.3.2. Arquitectura del generador de datasets 24

4.4. Resultados 27

5. Objetivos de desarrollo sostenible **29**

Bibliografía **31**

Índice de figuras

Figura 2.1. Proceso de convolución en matrices 6
Figura 2.2. Comparativa de algoritmos de detección de borde 7
Figura 2.3. Detección de borde por algoritmo de Canny 8
Figura 2.4. Representación de la transforma de Hough 9
Figura 2.5. Estructura de AlexNet 11
Figura 2.6. Errores cometidos en ImageNet en los últimos años 12
Figura 2.7. Brazo robótico IRB 1400 YuMi 13
Figura 3.1. Esquema de la arquitectura del sistema 17
Figura 3.2. Diagrama de las etapas del sistema 17
Figura 3.3. Diagrama de gantt del proyecto 19
Figura 4.1. Esquema de la arquitectura del sistema 25
Figura 5.1. Objetivos de desarrollo sostenible aprobados en 2015 29

Índice de tablas

Tabla 3.1. Cronograma del proyecto 18

Siglas

API Application Programming Interface

DLR-RM German Aerospace Center (DLR) - Institute of Robotics and Mechatronics (RM)

Faster R-CNN Faster Region-proposal Convolutional Neuronal Networks

HSV Hue Saturation Value

PHT Probabilistic Hough Transform

PPHT Progressive Probabilistic Hough Transform

R-CNN Region-proposal Convolutional Neuronal Networks

R-RANSAC Recursive RANdom SAmple Consensus

RANSAC RANdom SAmple Consensus

RGB Red Green Blue

RGB-D RGB Depth

RHT Randomized Hough Transform

YOLO You Only Look Once

1

Introducción

Es siempre sabio mirar adelante, pero difícil mirar más allá de lo que puedes.
Winston Churchill (1874–1965)

Este primer capítulo introduce a la visión artificial y demuestra el gran interés otorgado por la comunidad científica en este área. También se muestra la motivación de este proyecto, así como sus objetivos y se desarrolla la estructura del mismo.

Los orígenes de la visión artificial surgieron en 1960 como sistemas para el reconocimiento de patrones y centrados en su posible implantación en el sector industrial debido a la gran cantidad de tareas repetitivas fácilmente automatizables [AT13]. A pesar de la carencia de los recursos en su momento, pero debido al gran interés del mercado estos siguieron siendo desarrollados y poco a poco implantados. Una de las primeras empresas en implantar estos sistemas fue Hitachi Labs en 1964 en Japón [AT13].

Desgraciadamente estos primeros desarrollos carecían de precisión y tenían una tasa de acierto del 95 % (no se consideró suficiente para su implementación en una línea de producción). Pero esto se vería resuelto en los años venideros de forma que en la década de los 70 estos sistemas pasaron a formar parte de un gran porcentaje del sector industrial. Un claro ejemplo fue la automatización de la producción de transistores con semiconductores en 1974 por Hitachi ya que, debido a su complejidad, esta tarea no podía ser llevada a cabo por humanos de forma segura [Lab].

En la actualidad se pueden distinguir dos tipos de visión artificial en función de sus capacidades de adaptación. En primer lugar, se encuentran los sistemas desarrollados a partir 1960 que se caracterizan por ser programas para cumplir un único objetivo bajo unas circunstancias dadas. Estos son los menos potentes ya que no se adaptan y no saben reaccionar ante un cambio, pero son los más fáciles de desarrollar. Estos sistemas se basan en características diferenciadoras del objeto de trabajo como puede ser la forma, color, un patrón... Además, en espacios controlados pueden llegar a dar mejores resultados que sistemas más complejos y avanzados [Cas+13]. Pero es debido a su falta de flexibilidad y a las limitaciones de características diferenciadoras de las piezas de trabajo que están siendo sustituidos.

Por otro lado, con el desarrollo de las redes neuronales y de inteligencias artificiales, se están desarrollando sistemas capaces de aprender y adaptarse a la situación a base de prueba y error. Se trata de sistemas muy modernos todavía en desarrollo que prometen traer avances como la conducción autónoma, detección de emociones en humanos, etc. Estos sistemas necesitan de grandes cantidades de bases de datos de las que aprender. Y sobre las que generar sus propias conclusiones y reglas de conducta. Su nivel de adaptabilidad y flexibilidad se ve definido por las capacidades de aprendizaje del sistema (neuronas y estructura de estas) así como de la riqueza de la base de datos. Pero gracias a los últimos avanzes, han conseguido superar la barrera de tecnología en desarrollo y se han empezado a implantar en sistemas reales.

El fin de este proyecto es la generación de un nuevo sistema de visión artificial para el reconocimiento de piezas de uso industrial. Este sistema se implantará dentro de una cadena de suministro del Grupo Antolín® que a su vez alimentará a una linea de montaje y ensamblaje. El sistema deberá de poder identificar múltiples piezas y determinar el punto de agarre óptimo de cada pieza. Este se ve definido por sus coordenadas así como el vector normal a la superficie. De esta forma un brazo robótico con un sistema de agarre por aspiración, ventosa o *soft-robotics* (varía en función de la pieza a coger) podrá recolectarlas. La multitud de herramientas de agarre así como de la necesidad de un sistema de determinación de puntos de agarre se debe a la gran variedad de piezas existentes con formas y tamaños de gran variedad (1-30 cm). Este problema a resolver se puede dividir en etapas:

1. Generación del pedido: en función de la demanda y de los requisitos del cliente se debe de generar una lista con todos los componentes necesarios para cada pedido.

2. Estructuración del pedido: Las piezas necesarias se encuentran distribuidas en diferentes secciones y es por ello que se debe de crear un que determine el orden de recolección de piezas optimo que reduzca el tiempo recolección.

3. Recolección de piezas: Se trata de un proceso iterativo que se debe de realizar para cada una de las piezas que constituyen el pedido.

    - Desplazamiento hasta la pieza. Dependiendo de la configuración del robot este sistema variará, pero el objetivo siempre será el mismo. Trasladar el robot hasta la región donde se encuentra la pieza a recolectar con el fin de poder capturarla y situarla dentro de la región de alcance.

    - Detección de la pieza: aplicando algoritmos de detección se identificará la pieza que se desea recolectar. Dentro de una misma zona se detectarán numerosas instancias de una misma pieza. Se debe de escoger la pieza/piezas mejor ubicadas y con más probabilidad de éxito.

    - Punto de agarre: tras detectar la pieza se debe de determinar como se debe de agarrar la pieza. Para ello se debe de determinar el punto de agarre óptimo y el vector normal a dicho.

    - Recolección de la pieza: se transfiere la información necesaria al robot para que este pueda recolectar la pieza a través del punto de agarre definido. El sistema de agarre a emplear dependerá del tpo de pieza.

    - Deposición y control de calidad: por último se debe de depositar la pieza dentro de la cesta que constituye el pedido. Y mediante el sistema de visión artificial se debe de comprobar que la pieza deseada ha sido correctamente depositada en la cesta.

4. Control de calidad y trazabilidad: antes de dar por finalizado en pedido se analiza por última vez para comprobar que todas las piezas necesarias se encuentran dentro de la cesta. Se registra en el sistema el pedido y toda la información necesaria para futura trazabilidad.

5. Traslado del pedido: una vez se da por finalizado el pedido este se debe de trasladar hasta la zona de ensamblaje para comenzar el proceso de montado.

Como se ha mencionado en el párrafo anterior, las capacidades de una red neuronal se ven limitadas tanto por la estructura de la red como por la riqueza de la base de datos. Y para esta aplicación concreta la obtención de dicha base de datos presenta un gran desafío debido a la gran cantidad de información que se requiere de cada imagen (identificación de las piezas, posibles puntos de agarre de cada pieza y el vector normal a dichos puntos). Es por lo que se ha optado por desarrollar a su vez una base de datos sintética que permita generar todas las imágenes e información necesarias. De esta forma se podrá automatizar el proceso de aprendizaje de la red para la introducción de nuevas piezas.

2

Estado de la cuestión

En este capítulo se desarrollará la evolución de la visión artificial hasta llegar a la situación actual. También se introducirá la componente robótica del proyecto así como otras aplicaciones *Pick and Place*.

Como ya se ha indicado en la Sección 3.3, este proyecto está claramente definido por dos elementos que deben de cooperar entre sí para poder cumplir el objetivo de identificar y recolectar las piezas. Se necesita de un buen sistema de reconocimiento de objetos a través de imágenes, así como de un brazo robótico para poder interactuar con el entorno. Por ello, se van a analizar cada uno de estos elementos por separado y después se desarrollará la interacción entre estos mismos. Primero se analizará la situación actual del procesado de imágenes, después se analizará los brazos robóticos y por último, se hablará de los sistemas *Pick and Place*.

2.1. Procesado de la imagen

Como ya se ha explicado en la introducción, los primeros sistemas de visión artificial surgieron en la década de los sesenta y desde entonces han avanzo notablemente hasta el nivel de convertirse en uno de los pilares fundamentales de la industria moderna. Actualmente los nuevos sistemas se han desviado drásticamente de los primeros intentos y emplean complejos sistemas neuronales y *machine learning* para obtener mejores resultados y mejores respuestas. En este proyecto se va a perfeccionar el actual sistema basado en filtros de color y detección de borde y también se va a desarrollar una alternativa basada en redes neuronales.

2.1.1. Filtros, detección de borde y detección de formas

La identificación de objetos no es un proceso directo, sino que está compuesto por varias etapas y dentro de estas existen multitud de herramientas. En esta sección se va a intentar segmentar este proceso y explicar de forma independiente cada proceso, así como las diferentes herramientas que se pueden usar.

Filtros

Los filtros son diseñados con el objetivo de modificar la imagen para facilitar la extracción de características. Estos se pueden dividir en dos grandes categorías:

- **Filtros lineales:** Consisten en aplicar a cada píxel un filtro con sus píxeles vecinos. Dependiendo de los pesos que se den a los píxeles vecinos se pueden obtener diferentes efectos [GSA16]. Se muestra la expresión matemática 2.1 que rige este proceso.

$$g(i,j) = \sum_{k,l} f(i+k, j+l) * h(k,l) \tag{2.1}$$

  Se trata de un proceso por convolución en el que dependiendo de la matriz a usar y el peso de sus componentes se pueden obtener diferentes resultados [GSA16]. Se puede ver su aplicación en la figura 2.1.

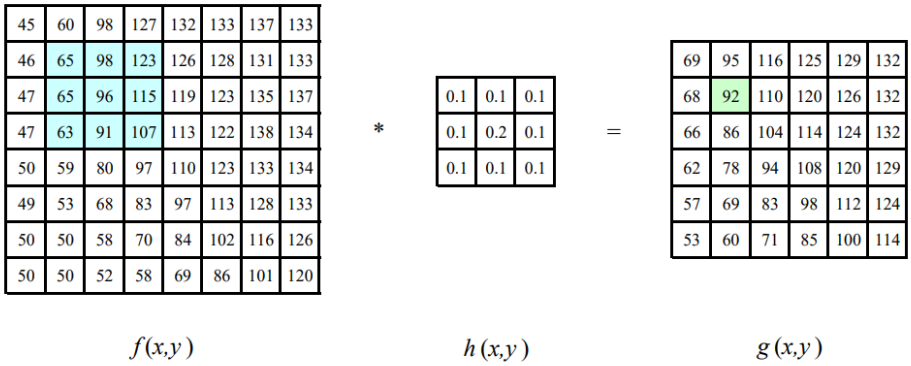

Figura 2.1. Ejemplo de un proceso de convolución aplicado a una matriz (Fuente: Computer Vision: Algorithms and Applications, Richard Szeliski figure 3.10 [Sze11])

  Un claro ejemplo de este tipo de filtros es el suavizado. Permite suavizar la imagen y hacerla más homogénea al combinar cada pixel con sus vecinos. Este filtro permite reducir los valores atípicos reduciendo así el efecto del ruido. Pero también se puede dar el caso contrario en el que nos interese resaltar los pequeños detalles. Este filtro es conocido como *Sharpening* y se basa en un filtro lineal de suavizado. A la imagen original se le resta las diferencias producidas por el suavizado y de esta forma se resaltan más los pequeños detalles de la imagen. Este proceso es conocido como *unsharp masking* [Sze11].

- **Filtros no lineales:** A pesar de que muchos de los filtros usados se pueden obtener de forma lineal, en muchas ocasiones se consigue un mejor rendimiento aplicando filtros no lineales. Tal y como su nombre indica, esta categoría engloba todos aquellos filtros que no se puedan expresar como una suma de diferentes argumentos.

  El beneficio de los filtros no lineales se puede ver al aplicar transformaciones que usen por ejemplo la mediana. El cálculo de la mediana puede suponer una alta carga computacional frente al resto de filtros. Por eso se han desarrollado alternativas para mejorar su cálculo. Tales como α-*trimmed mean* [LR90].

  Un buen ejemplo de filtro no lineal que emplee la mediana es la transformación a escala de grises. Se suele usar la mediana ya que es más robusta ante variaciones atípicas. En el reconocimiento de objetos es muy recomendable usar la escala de grises ya que se

consigue una mejor detección de borde [AMS18] y reduce la carga computacional al trabajar con una sola matriz (*Black*) en lugar de tres (RGB). Existen múltiples algoritmos para transformar a escala de grises, aunque uno de los más empleados y que mejores resultados da para el reconocimiento de borde es *Lightness color-to-greyscale conversión algorithm* [AMS18]. Sin embargo, gracias a los avances en redes neuronales y a la mejoras computacionales los sistemas actuales no se ven tan limitados y por ello son capaces de trabajar con imágenes de alta resolución y los tres canales RGB.

Detección de bordes

Los bordes son el pilar fundamental de la visión artificial. Representan la separación entre diferentes objetos y definen sus formas. En una imagen, un borde se define como un salto de la intensidad de los pixeles.

En la práctica, estos cambios no son tan bruscos, sino que se producen gradualmente a lo largo de varios pixeles. Por ello no son tan fáciles de detectar y suelen aparecer bordes residuales producidos por ruido y errores en la imagen. En la detección de borde existen dos claras vertientes, aquellos métodos que se basan en el gradiente y los que emplean el laplaciano.

- **Operadores basados en gradientes:** El gradiente de una función es la colección de todas las derivadas parciales en forma de vector. La dirección del vector es la de máximo crecimiento y su módulo es el valor de la pendiente [Aca]. Al aplicar este concepto a imágenes, podemos obtener para cada píxel un vector que indique la dirección en la que aumenta la intensidad y su módulo. Analizando los gradientes obtenidos con la ayuda de máscaras, podemos encontrar los bordes ya que se producen por un cambio brusco de intensidad.

  Existen varios algoritmos basados en el gradiente, aunque algunos de los más conocidos son el de Robert, Prewitt y el de Sobel. La diferencia entre estos son las máscaras a aplicar. Robert es un operador diferencial discreto con una matriz de 2x2, mientras que Prewitt y Sobel emplean dos matrices de 3x3. Se puede ver una comparativa de estos en la Figura 2.2.

  El principal inconveniente de estos operadores es su forma de representar los bordes ya que están definidas por una franja de pixeles y además se ven bastante afectados por el ruido. Por ello existen algoritmos basados en la segunda derivada (laplaciano) para delimitar más la región de borde y obtener una mayor precisión.

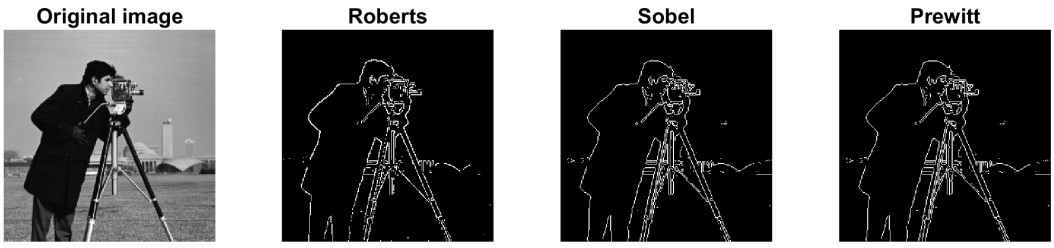

Figura 2.2. Comparativa de algoritmos de detección de borde (Fuente: Ejemplo de MATLAB con imágenes de stock)

- **Operadores basados en laplacianos:** Emplean la segunda derivada para obtener una mejor representación del borde y mayor robustez frente al ruido. Pero a cambio, requieren de una mayor carga computacional [BHM20].

Existen múltiples algoritmos basados en el laplaciano, pero el más importante y usado es Canny. Esto es debido a su alto nivel de precisión y eficiencia. Asimismo, debido a su popularidad, han surgido múltiples variantes del algoritmo de Canny para perfeccionar la detección de borde.

El algoritmo de Canny se realiza en tres etapas. El primer paso es aplicar un filtro gaussiano para suavizar la imagen y evitar falsos bordes producidos por ruido en la imagen. A continuación, se calcula el gradiente de cada punto con su dirección y módulo. En este paso se intenta encontrar todos los bordes posibles de la imagen, pero con una diferencia respecto a los algoritmos anteriores, en este caso solo se queda con los máximos locales, convirtiendo así los bordes en líneas más nítidas y sin el efecto rampa. Por último, se hace un filtrado para eliminar los bordes innecesarios. Para ello se aplica un doble umbral sobre los gradientes y seguido por un rastreo de histéresis. Este elimina todos aquellos posibles puntos que no estén conectados a un borde. Se pueden observar los resultados en la Figura 2.3.

Figura 2.3. Detección de borde por algoritmo de Canny (Fuente: Ejemplo de MATLAB con imágenes de stock)

Detección de formas

Tras procesar la imagen y aplicar los algoritmos de detección de borde, se puede comenzar a analizar la escena. El objetivo de esta etapa es reconocer líneas y formas para poder detectar la posición y orientación del objeto. De nuevo, existen múltiples algoritmos para solventar el problema de localización y determinación a partir de imágenes. En este documento solo se van a analizar dos, la transformada de Hough y RANSAC ya que en múltiples ocasiones han demostrado su robustez y son dos de los algoritmos más extendidos.

- **Transformada de Hough:** Este algoritmo fue diseñado en 1972 por Richard Duda y Peter Hart [DH72] y se ha postulado como una de las mejores alternativas para el reconocimiento de formas geométricas primitivas. Entendiéndose estas como una curva o superficie que pueda ser expresada matemáticamente con tres parámetros [DH72]. Por ejemplo, una línea, circulo, elipse, etc.

  Para poder identificar curvas o superficies, el algoritmo analiza cada punto de la imagen y hace pasar por este todos los patrones posibles. Para cada punto se almacena ρ, que representa la distancia mínima entre la recta a analizar y el origen de coordenadas, esta viene dada por una perpendicular a la recta. Y θ que es el ángulo del vector director de esa recta [IK88]. Esto se puede ver gráficamente en la Figura 2.4.

La transformada de Hough ha demostrado numerosas veces su gran robustez para detectar patrones y líneas, pero a pesar de ello, presente un gran inconveniente que impide que sea usada tan extensamente. Requiere de una alta carga computacional ya que debe analizar cada punto de la imagen y almacenar todas las posibles rectas/formas. Por ello, se han desarrollado multitud de variantes del algoritmo original con el fin de o mejorarlo o reducir su carga. Algunas de las variantes más notables son:

- *Randomized Hough Transform (RHT)*: Se basa en el algoritmo original, pero con la peculiaridad de que no necesita analizar todos los puntos de la imagen. El algoritmo intenta seleccionar de forma aleatoria puntos que puedan formar parte de una posible curva. De esta forma, se reduce bastante la carga computacional.
- *Probabilistic Hough Transform (PHT)*: Parte del esquema de acumulación del algoritmo original, pero difiere en el método de selección del siguiente punto a analizar. Selecciona un subconjunto a analizar en el que se ha incluido puntos previamente detectados como borde. De esta forma se reduce notablemente la carga computacional.
- *Progressive Probabilistic Hough Transform (PPHT)*: Partiendo de la imagen original, se seleccionan puntos de forma aleatoria y se analizan antes de pasar al siguiente. Para cada punto se observa si puede formar parte de una línea o no y se decide en base a los resultados obtenidos por los anteriores puntos. La principal ventaja de este algoritmo es la capacidad de poder ser interrumpido en cualquier momento y aun así dar buenos resultados. Pero es bastante susceptible a dar falsos positivos producidos por ruido en la imagen.

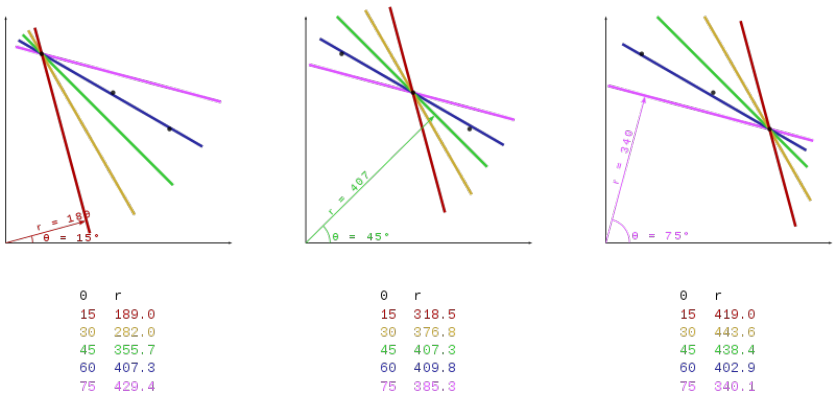

Figura 2.4. Representación de la transforma de Hough (Fuente: https://en.wikipedia.org/wiki/Hough_transform)

- ***RANdom SAmple Consensus*:** También conocido como RANSAC, se trata de un método matemático iterativo para encontrar los parámetros de un modelo matemático dentro de unos datos. Fue publicado por primera vez en 1981 por Fischler y Bolles y desde entonces se ha convertido en uno de los pilares de la visión artificial [FB81].

  El método iterativo se puede separar en varias etapas que se repiten consecutivamente, tantas veces como se desee. Al aumentar el número de iteraciones, aumenta la probabilidad de detectar el objeto. Las etapas de cada iteración son:

  - Selección de un subconjunto aleatorio sobre el que trabajar. Se conocen como *inliers hipotéticos*.
  - Basándose en el subconjunto, se monta un modelo sobre este.

- Se comprueba la robustez del modelo contrastando contra el resto de los datos.
- Si suficientes puntos se han clasificado como parte del modelo, este es válido.

En la siguiente iteración se partirá de este modelo y se intentará mejorar la detección de este.

Este algoritmo a pesar de ser muy preciso presenta una gran desventaja. Solo es capaz de contrastar la imagen frente a un modelo dado, por ello solo es capaz de detectar un tipo de objeto. Por ello se han desarrollado alternativas como Recursive RANdom SAmple Consensus (R-RANSAC).

2.1.2. Redes neuronales convolucionales

La idea del aprendizaje automatizado/aprendizaje profundo/inteligencia artificial no es novedosa, ya en 1955 IBM formó un grupo centrado en el reconocimiento de patrones supervisado por Nathaniel Rochester. Para ello, decidieron simular el funcionamiento de las neuronas con un ordenador IBM 704 [Vid19]. Y aunque estos conceptos no suenan novedosos, es importante repasar los conceptos principales para poder entender las técnicas de aprendizaje automatizado [Alo+18].

En función del tipo de aprendizaje que se vaya a realizar, los tipos de inteligencias artificiales se pueden separar en 4 categorías:

- Aprendizaje supervisado: la información de la que se va a aprender ha sido preparada y etiquetada. Es un trabajo largo y tedioso, pero le permite a la inteligencia saber qué es lo que debe aprenderse.
- Aprendizaje semi-supervisado: Tal y como su nombre indica, en este caso no toda la información ha sido preparada y etiquetada.
- Aprendizaje no supervisado: La información no ha sido preparada de ninguna forma y es el trabajo de la inteligencia el de entender la información y distinguir qué es lo que se desea aprender.
- Aprendizaje por refuerzo: es un área del aprendizaje automático inspirada en la psicología conductista. La inteligencia debe de decidir que acciones tomar en un entorno y en función de sus decisiones recibirá una recompensa.

Qué tipo de aprendizaje usar depende del tipo de problema que se intente aprender. En este proyecto se va a emplear el aprendizaje supervisado. Para ello se emplearán imágenes previamente etiquetadas para que una red neuronal pueda aprender de ellas.

Desde la década de los cincuenta han surgido múltiples algoritmos para intentar simular el comportamiento de las neuronas y así poder alcanzar el objetivo de aprendizaje automatizado/profundo. Pero no fue hasta 1989 que surgió el concepto de Red Neuronal Convolucional impulsado por Yann LeCun. Yann desarrolló una de las primeras redes capaces del auto aprendizaje diseñada para reconocer la escritura a mano [Lec+89]. LeCun destacó por el uso de la convolución y fue uno de los primeros en obtener unos resultados positivos con este método.

En 2012 un grupo de investigadores dirigido por Alex Krizhevsky consiguieron obtener el menor error de clasificación visto hasta la fecha sobre las 1.2 millones de fotos de ImageNet con 1000 clases diferentes [KSH17]. Fue gracias a esta hazaña que empezó a crecer un gran

interés por las redes neuronales convolucionales frente al resto de alternativas. Estas nuevas redes se basan en el principio de convolución, el cual ha sido explicado en la Sección 2.1.1. La convolución se caracteriza por ser un proceso completamente lineal, esto es lo que ha permitido el desarrollo de estas redes y que puedan aprender tan fácilmente. En la actualidad este tipo de redes son las más comunes para el análisis de imágenes y su uso está ampliamente extendido.

Dentro del procesado de imágenes por aprendizaje profundo se distinguen dos categorías de redes en función de su objetivo:

- Clasificadores: son capaces de identificar un objeto en una imagen. Pueden ser entrenados para identificar una gran cantidad de objetos, pero tienen la desventaja de que solo pueden detectar un objeto por imagen. Además, solo indican su presencia, no su posición. Uno de los clasificadores más conocidos es AlexNet, la red neuronal diseñada por Alex Krizhevsky y mencionada anteriormente. Se muestra su estructura en Figura 2.5.

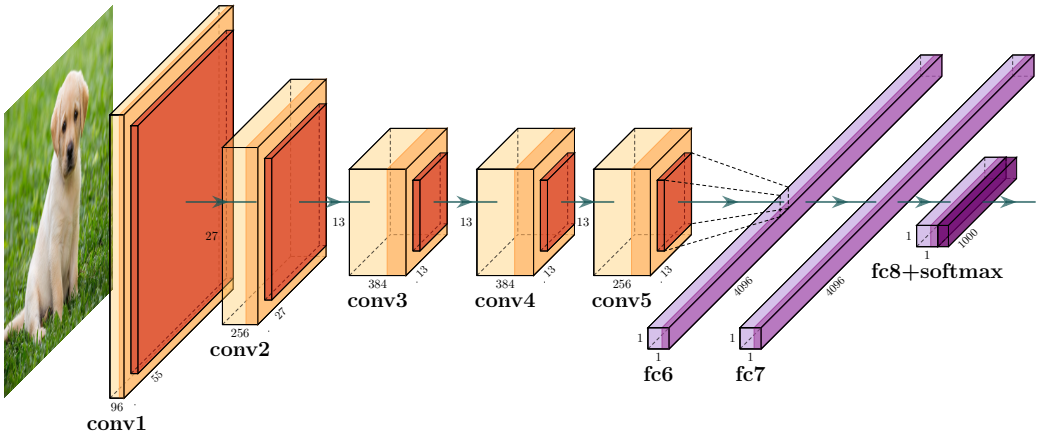

Figura 2.5. Representación de la estructura de AlexNet [KSH17]

En los últimos años los clasificadores han vivido una gran evolución y desarrollo y en parte esto es debido a la competición de ImageNet. Todos los años, investigadores de todo el mundo compiten por intentar obtener el mejor resultado al intentar clasificar millones de imágenes y mil clases. En la actualidad ImageNet cuenta con más de 14 millones de imágenes de alta resolución para ser clasificadas en mil clases. A continuación, se muestra en la Figura 2.6 el avance de las redes en este área analizando el error cometido por estas en ImageNet. Como referencia también se ha añadido el error medio cometido por los humanos. Se puede observar que algunas de las redes más modernas ya son capaces de superar al ojo humano en estas pruebas.

- Detectores de objetos: se basan en los principios de los clasificadores pero por medio de diferentes herramientas son también capaces de localizar el objeto en la imagen. Además, pueden identificar y detectar múltiples objetos en una misma imagen. En la actualidad existe una gran variedad de estructura y métodos para la detección de objetos, pero en este proyecto solo se van a analizar y emplear algunos de los más populares:

  - *Region-proposal Convolutional Neuronal Networks (R-CNN)*:Se basan en el principio de propuesta de regiones frente a sistemas empleados previamente conocidos como ventana flotante. Este sistema consiste en correr un clasificador barriendo todas las posibles secciones de una imagen y con diferentes tamaños. Como este proceso es

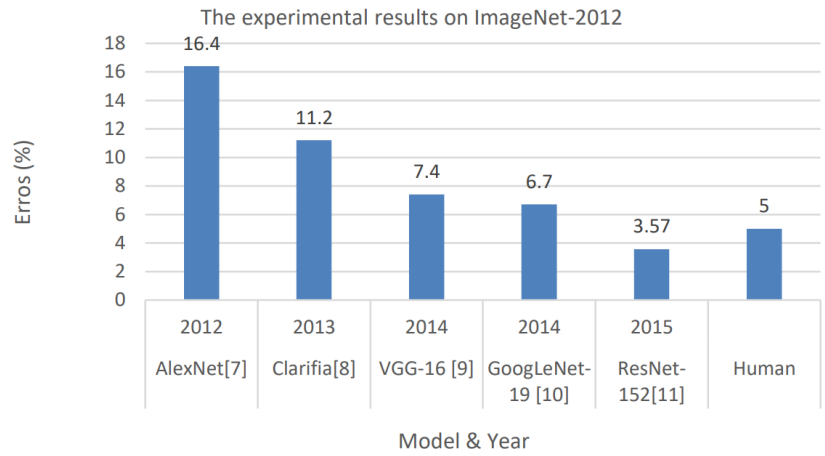

Figura 2.6. Errores cometidos en ImageNet en los últimos años (Fuente: The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches [Alo+18]

muy costoso a nivel computacional y lento, R-CNN propone usar un sistema que proponga un máximo de 2000 regiones a estudiar basándose en colores y formas fácilmente reconocibles. De esta forma se reduce bastante la carga, pero manteniendo un buen nivel de confianza y precisión.

- *Faster Region-proposal Convolutional Neuronal Networks (Faster R-CNN)*: Al igual que R-CNN se basan en el principio de propuesta de regiones pero difiere de este en el método para proponer dichas regiones de interés. En este método, la propia red neuronal se encarga de analizar la escena y decidir qué regiones debe considerar de interés. Esto resulta en una red bastante más rápida, aunque puede ser menos precisa que R-CNN.

- *You Only Look Once (YOLO)*: Es una de las técnicas más actuales y prósperas. Como su nombre indica, se caracteriza porque la imagen solo se pasa una vez por el clasificador y una capa convolucional se encarga de predecir la clase de objeto detectado y su posición. Este tipo de redes se caracterizan por ser extremadamente rápidas, aunque son incapaces de igualar la precisión de R-CNN.

2.2. Robótica industrial

El término robot se remonta hasta 1921 donde por primera vez se introdujo en la obra de teatro R.U.R (Rossum's Universal Robots) obra del novelista y autor checo Karel Čapek. La palabra deriva de *robota* que en checo significa fuerza del trabajo o servidumbre [Oll05]. En la actualidad es un término aceptado y conocido por toda la sociedad, empleado para designar a cualquier "máquina o ingenio electrónico programable que es capaz de manipular objetos y realizar diversas operaciones" (definición de robot según la RAE).

Actualmente los robots representan una gran parte de la fuerza de trabajo industrial. Se presentan en una gran variedad de configuraciones diferentes para cumplir diferentes objetivos, aunque una de las configuraciones más destacadas son los brazos robóticos. Tal y como su nombre indica, se asemejan a brazos de forma que tienen múltiples articulaciones sobre las que rotar para poder moverse en todas las direcciones y orientaciones posibles.

Estos robots se pueden clasificar en función de diversas categorías:

- Por su función. Para ello se han desarrollado diferentes efectores que les permiten realizar tareas tan simples como *Pick and Place* hasta tareas más exigentes como soldar.
- Por su forma. Esta muchas veces se ve determinada por la función que debe desarrollar el brazo, así como el entorno en el que debe trabajar. Aunque a grandes rasgos se pueden distinguir dos tipos, fijos y móviles. Como sus nombres indican, esto depende de la capacidad del robot para moverse libremente.
- Por el sistema de control. Los robots se pueden controlar por control manual, a distancia, mediante programación o con un sistema de aprendizaje por refuerzo.
- Por el modelo de control de la planta. Una planta industrial dotada de numerosos robots tiene dos formas de ser controlada. Se puede tener un sistema de control independiente para cada robot, esto simplifica la creación de la planta, pero dificulta su manejo. O puede contar con un complejo sistema de interconexión entre las diferentes etapas de la planta. Un buen ejemplo de este tipo de sistemas de control es SCADA (Supervisory Control And Data Acquisition) [Pér15].

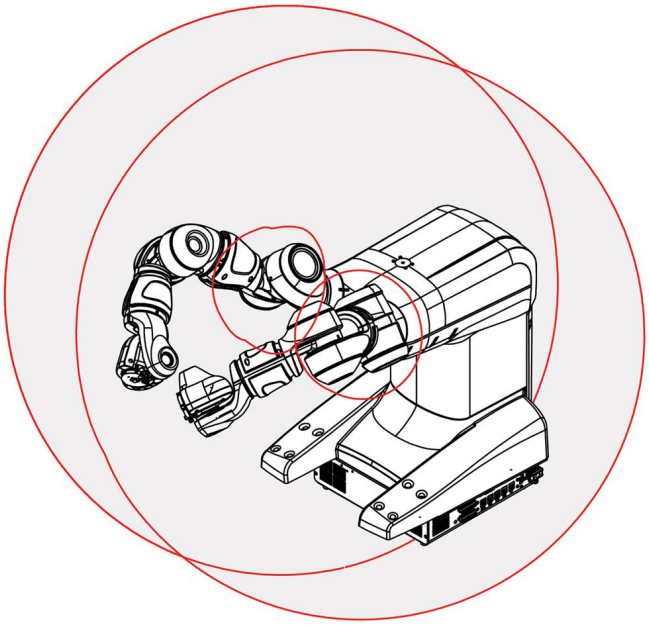

Figura 2.7. Brazo robótico IRB 1400 YuMi empleado en el proyecto con su rango de acción [ABB]

Como ya se ha explicado previamente, el objetivo de este proyecto es el desarrollo de un sistema capaz de localizar y capturar piezas para la preparación de un pedido. Este tipo de sistemas son conocidos como configuraciones *Pick and Place* y son ampliamente usadas en la industria. Estos sistemas suelen estar constituidos por un brazo robot, una cámara y una unidad de procesamiento para analizar las imágenes. En función de los objetos que deben transportar, estos varían su forma de coger los objetos y sus cabezales.

Un buen ejemplo de la importancia de estos sistemas es *Amazon Robotics Challenge*. Con el fin de mejorar los sistemas actuales y de buscar a los mejores en el sector, Amazon ha creado una competición entre alumnos universitarios para desarrollar el mejor sistema *Pick and Place* [Mor+17]. Estos sistemas se basan en redes neuronales muy avanzadas y desarrolladas tales como RefineNet [Mor+17].

3

Metodología de trabajo

Para poder desarrollar w innovar es necesario primero definir una estructura base sobre la que edificar. Y se debe determinar un plan a seguir que sirva de guía.

3.1. Motivación

La industria avanza a un ritmo constante y cada día es más necesario la implantación de sistemas robóticos para poder llevar a cabo tareas que los humanos no podemos desarrollar o que presentan una elevada tasa de errores humanos o tienen un elevado nivel de peligrosidad. Al dotar a estos sistemas de inteligencia y un sistema de visión artificial, se consigue que se pueda adaptar mejor al entorno y se evite tener que re-programar los robots con cada cambio de las condiciones de operación. Avanzamos hacia una sociedad en la que los robots serán la principal mano de obra y para llegar a ese objetivo es necesario invertir y desarrollar más los sistemas actuales. Por ello se ha propuesto como proyecto la integración de un sistema de visión artificial en un robot industrial. Con este proyecto se podrá modernizar las instalaciones del Grupo Antolín® y aumentar las capacidades de la línea de montaje y ensamblaje actual.

3.2. Objetivos

Este trabajo parte de un proyecto actualmente en desarrollo con un sistema de visión artificial ya desarrollado e implantado. Desgraciadamente la implantación actual presenta limitaciones así como una tasa de error superior a los requisitos planteados por el Grupo Antolín.

Para poder superar las limitaciones actuales se ha planteado el desarrollo de un nuevo sistema que debe de mejorar las capacidades de identificación y detección de las piezas así como poder determinar el punto de agarre óptimo. El sistema debe de ser modular de forma que se pueda adaptar fácilmente a nuevas piezas. Y con el fin de que el robot tenga la mayor posibilidad de coger la pieza, también se debe de determinar el vector normal al punto de agarre. Por último, se pide que sea independiente del resto del proyecto de forma que pueda ser fácilmente manejado, modificado y actualizado.

Con el fin de cumplir dichos requisitos se deben plantear unos objetivos de corto, medio y largo alcance a seguir durante todo el desarrollo del trabajo:

- Desarrollo de una base de datos sintética:
  - Investigación y prueba de diferentes sistemas/herramientas (bpi, zpi Zumo labs y Blenderproc).
  - Obtención de primeras imágenes RGB y de profundidad.
  - Introducción de aleatoriedad y repetibilidad al sistema.
  - Generación de un *Pipeline* para la automatización del proceso.
  - Introducción de ruido en las imágenes para mejorar la robustez.
- Desarrollo de una base de datos real complementaria tanto para la fase de entrenamiento como evaluación del sistema.
- Desarrollo de las nuevas redes neuronales:
  - Investigación y pruebas de diferentes sistemas/herramientas (YOLO, MobilNet, redes regresoras, etc).
  - Planteamiento y definición de la estructura modular a desarrollar. Una primera red para la detección de piezas, seguida de una red para la detección de zonas de intereses con probabilidad de ser el punto de agarre óptimo. Y finalmente un una red neuronal del tipo regresor para la determinación del punto de agarre.
  - Primeros desarrollos independientes de las partes modulares del sistema
  - Desarrollo y corroboración del conjunto modular
  - Creación del sistema de evaluación tanto del conjunto como de las partes modulares del sistema
  - Entrenamiento y evaluación de todo el sistema con diferentes configuraciones y bases de datos.
  - Comparativa y determinación del sistema óptimo.
  - Implantación del nuevo sistema de visión artificial

3.3. Arquitectura del sistema

El sistema se compone de tres elementos claves: un robot industrial para poder interactuar con las piezas, una cámara RGB-D para la captura de imágenes y Python para el procesamiento de las imágenes y la toma de decisiones. Es necesario que estos tres elementos funcionen correctamente y estén comunicados entre sí. La arquitectura básica se puede ver en Figura 3.1

A continuación, se va a detallar todo el proceso desde el arranque del sistema hasta el final del mismo y se analizarán cada una de las etapas que constituyen el proceso. Este se puede ver gráficamente en Figura 3.2. El sistema debe compenetrase con los sistemas actuales los cuales diseñan los pedidos que se deben de preparar y ensamblar. Estos pedidos se reciben y se procesan para establecer qué piezas se desean y en que orden deben de ser recogidas por medio de un proceso cíclico.

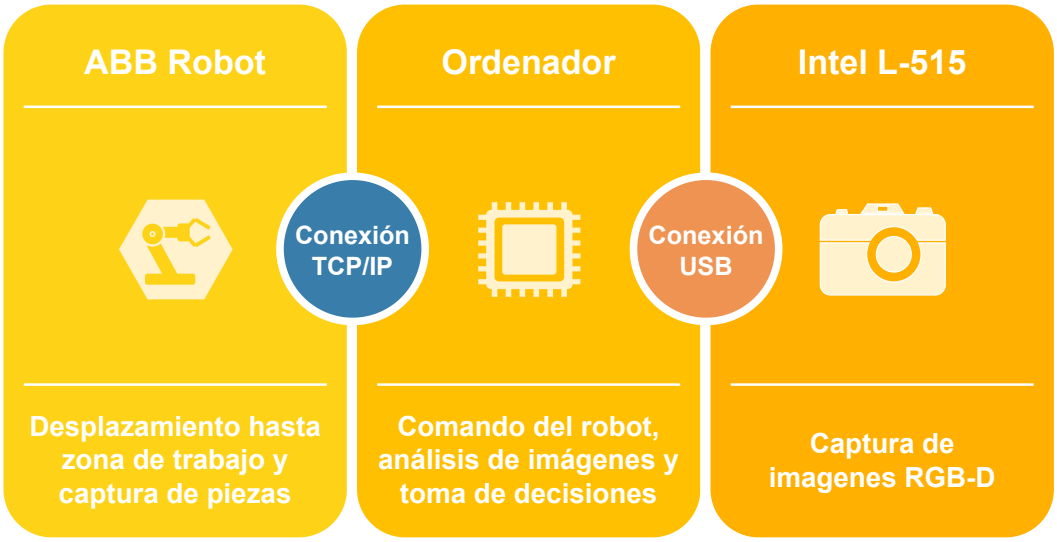

Figura 3.1. Esquema de la arquitectura del sistema

Una vez procesado el pedido y establecida la primera pieza a recoger se debe de desplazar el robot a la zona de trabajo y prepararse para la captura de dicha pieza. Una vez reubicado el robot, un sistema de cámaras capturará la escena para determinar la posición de la pieza/piezas que se desean capturar frente a la posición del robot. Esta imagen es analizada por el sistema de visión artificial que se encargará de .[en] primera instanciaïdentificar las piezas para entre ellas poder centrarse solo en las que se desea capturar. A continuación, si se trata de una pieza de dimensiones grandes se analizará más en profundidad para determinar los posibles puntos de agarre. Y una vez determinados estos puntos de agarre un regresor determinará el centro exacto de dichos puntos de agarre así como el vector normal a dicho punto (dirección que debe emplear la muñeca del robot para capturar la pieza). Si por el contrario se trata de una pieza de dimensiones reducida se empleará el centro de la pieza como punto de agarre.

Esta información es transmitida al robot permitiéndole así poder capturar la pieza y depositarla en la cesta del pedido. Para ello este se deberá aproximar a la pieza seleccionada siguiendo en la medida de lo posible la trayectoria normal determinada. Al aproximarse a la pieza también deberá reducir la velocidad de operación para evitar colisionar.

Figura 3.2. Diagrama de las etapas del sistema

3.4. Herramientas

- Cámara Intel RealSense L515 [Inta] y Wrapper de Python para Intel Realsense [Intb].
- Ordenador con los siguientes SO/paquetes/programas:
  - Ubuntu 20.04 LTS

- Blender 2.93.5
- BlenderProc 2.0 (Blender API)
- PyTorch 1.10.1
- Tensorflow 2.7.0
- CUDAToolkit
- Python 3.8 o superior
- Conda ó miniconda

3.5. Cronograma

La planificación del proyecto permite establecer un orden a seguir. No solo da estructura al proyecto, también ayuda a los desarrolladores a estructurar sus ideas. Es por ello que es vital y necesario establecer guías, objetivos e hitos a seguir.

Tabla 3.1. Cronograma del proyecto

Descripción del hito	Categoría	Progreso	Inicio	Días
MVP				
Estado de la Cuestión	Hito	100 %	01/06/2021	14
Base de datos sintéticas v0.1	Objetivo	100 %	15/06/2021	14
Detección puntos de agarre	Hito	100 %	29/06/2021	7
Driver Intel RealSense v0.1	Objetivo	100 %	06/07/2021	7
Regresor - Tensorflow v0.1	Objetivo	100 %	13/07/2021	7
Evaluación regresor	Objetivo	100 %	20/07/2021	3
Base de datos sintética v1.0				
Introducción de Arucos	Objetivo	100 %	01/09/2021	21
Extracción y lectura de normales	Hito	100 %	22/09/2021	7
Preparación de bases de datos	Objetivo	100 %	29/09/2021	21
Actualización a Blenderproc 2.0	Hito	100 %	20/10/2021	7
Replicabilidad	Objetivo	100 %	27/10/2021	7
Riqueza de escenarios	Hito	15 %	03/11/2021	14
Generación de base de datos real	Objetivo	0 %	17/11/2021	7
Visión artificial v1.0				
Estado de la Cuestión	Hito	100 %	01/12/2021	10
Arquitectura del sistema	Objetivo	100 %	11/12/2021	14
YOLO	Objetivo	100 %	25/12/2021	7
TINY YOLO	Objetivo	100 %	01/01/2022	7
Regresor	Objetivo	15 %	08/01/2022	30
Evaluación				
Tamaño del dataset sintético	Objetivo	0 %	14/02/2022	30
Comparativa con dataset real	Objetivo	0 %	16/03/2022	10
Redacción				
Anexo B	Objetivo	100 %	01/12/2021	20
Memoria	Objetivo	15 %	01/04/2022	50

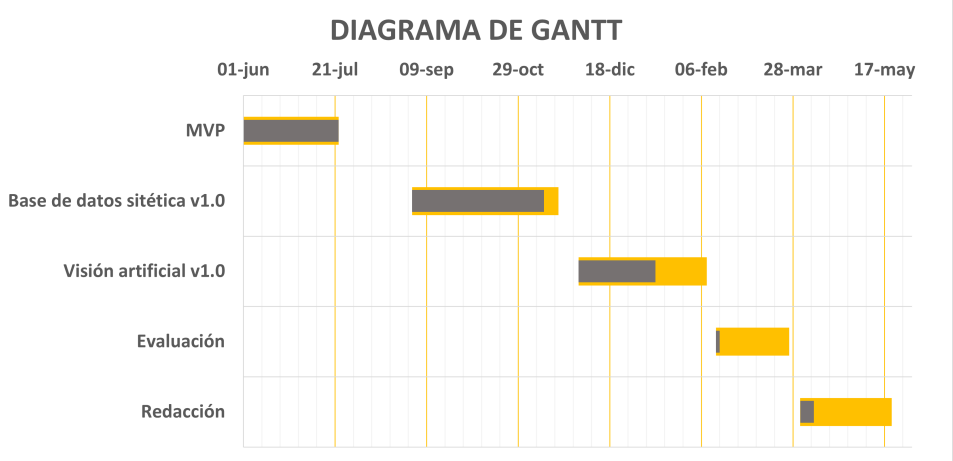

Figura 3.3. Diagrama de gantt del proyecto

4

Generación de un dataset

Un buen sistema de visión artificial requiere de un buen *dataset*. En este capítulo se desarrollará y justificará la creación de un *dataset* real así como un *dataset* sintético que permita expandir las capacidades del sistema de visión artificial.

Se entiende por *dataset* a un conjunto de datos normalmente tabulados/organizados empleados en la ejecución de programas o algoritmos. El término *dataset* es muy amplio y su funcionalidad y formato varía dependiendo del campo de desarrollo pero en este trabajo nos centraremos en una de las áreas donde mas se ha desarrollado el concepto, *Machine Learning*. La capacidad de aprender de un modelo depende en primera instancia de la calidad del *dataset*. Si este no tiene un formato correcto o no es capaz de asegurar la integridad de los datos entonces independientemente del modelo empleado el resultado sera desfavorable. Por lo tanto, la creación del *dataset* es de vital importancia y debe de ser uno de los primeros pasos en el desarrollo de todo sistema basado en *Machine Learning*.

Uno de los métodos mas comunes es la generación de forma manual del *dataset*. Esto implica que un ser humano debe de categorizar, estructurar y definir el valor del dato. Se caracteriza por ser sencillo y rápido para *datasets* pequeños pero presenta numerosos problemas cuando se requiere de datos complejos o de un tamaño elevado. Este proyecto entra dentro de esta categoría y es por ello que se ha creado un sistema de generación de datos que ha permitido automatizar y agilizar el proceso. A este tipo de *datasets* creados por ordenador se les define como sintéticos. Sin embargo no se recomienda depender solo de *datasets* sintéticos ya que estos no reflejan con total precisión la realidad. Es recomendable desarrollar un sistema que mezcle datos reales y sintéticos durante la fase de entrenamientos. Y emplear datos reales para la fase de validación ya que solo de esta forma se puede determinar la verdadera capacidad del modelo.

Pero antes de analizar los datos y las herramientas creadas para obtenerlos, es necesario entender que datos se desean obtener. Para ello se debe de entender el objetivo/problema del proyecto (3.2), el agarre de piezas industriales de diferentes formas y tamaños. Este problema se puede dividir en etapas:

1. Generación del pedido: en función de la demanda y de los requisitos del cliente se debe de generar una lista con todos los componentes necesarios para cada pedido.
2. Estructuración del pedido: Las piezas necesarias se encuentran distribuidas en diferentes secciones y es por ello que se debe de crear un que determine el orden de recolección de piezas optimo que reduzca el tiempo recolección.
3. Recolección de piezas: Se trata de un proceso iterativo que se debe de realizar para cada una de las piezas que constituyen el pedido.
    - Desplazamiento hasta la pieza. Dependiendo de la configuración del robot este sistema variará, pero el objetivo siempre será el mismo. Trasladar el robot hasta la región donde se encuentra la pieza a recolectar con el fin de poder capturarla y situarla dentro de la región de alcance.
    - Detección de la pieza: aplicando algoritmos de detección se identificará la pieza que se desea recolectar. Dentro de una misma zona se detectarán numerosas instancias de una misma pieza. Se debe de escoger la pieza/piezas mejor ubicadas y con más probabilidad de éxito.
    - Punto de agarre: tras detectar la pieza se debe de determinar como se debe de agarrar la pieza. Para ello se debe de determinar el punto de agarre óptimo y el vector normal a dicho. Para las piezas pequeñas se puede emplear el centro de la pieza como punto de agarre. Sin embargo, para piezas más grandes este método no funciona ya que se trata de piezas irregulares que debido a su gran tamaño requieren de un buen agarre capaz de levantar el peso de la pieza. Por ello se debe de buscar zonas lisas sobre las que se pueda emplear una ventosa o superficies más complejas que permitan el uso de un cabezal *soft-robotics*. Esto implica que la salida final del sistema debe de ser un punto de agarre y el vector normal a dicho punto que debe de seguir el brazo robótico.
    - Recolección de la pieza: se transfiere la información necesaria al robot para que este pueda recolectar la pieza a través del punto de agarre definido. El sistema de agarre a emplear dependerá del tpo de pieza.
    - Deposición y control de calidad: por último se debe de depositar la pieza dentro de la cesta que constituye el pedido. Y mediante el sistema de visión artificial se debe de comprobar que la pieza deseada ha sido correctamente depositada en la cesta.
4. Control de calidad y trazabilidad: antes de dar por finalizado en pedido se analiza por última vez para comprobar que todas las piezas necesarias se encuentran dentro de la cesta. Se registra en el sistema el pedido y toda la información necesaria para futura trazabilidad.
5. Traslado del pedido: una vez se da por finalizado el pedido este se debe de trasladar hasta la zona de ensamblaje para comenzar el proceso de montado.

4.1. Estructura del dataset

Una vez entendido el problema se puede determinar el tipo de datos que se necesitan y para este problema se puede observar que dependen en gran medida de la pieza. Por ello se debe de distinguir entre las piezas grandes y las piezas pequeñas. A continuación, se muestra la estructura estándar que deben de seguir las piezas pequeñas y grandes:

Piezas pequeñas

- Imagen en formato ".png"de toda la escena.
- Archivo ".txt"para identificar y detectar las piezas presentes en la imagen. Se emplea una fila para cada pieza y en este se debe de mostrar:
  - Categoría: identifica el tipo de pieza.
  - Centro x: coordenada horizontal en píxeles (normalizados) del centro del rectángulo que engloba la pieza en la imagen.
  - Centro y: coordenada vertical en píxeles (normalizados) del centro del rectángulo que engloba la pieza en la imagen.
  - Ancho: dimensiones en píxeles (normalizados) del ancho del rectángulo.
  - Largo: dimensiones en píxeles (normalizados) del largo del rectángulo.

Piezas grandes

- Imagen en formato ".png"de toda la escena.
- Archivo ".txt"para identificar y detectar las piezas presentes en la imagen. Idéntico al archivo ".txt"de las piezas pequeñas.
- Imágenes recortadas de cada una de las piezas presentes en la imagen global.
- Archivo ".txt"para identificar y detectar en cada una las imágenes recortadas las regiones de interés donde hay presentes zonas de agarre. Idéntico al archivo ".txt.ªnterior pero en este caso en lugar de piezas, lo que se detecta son regiones. Y todas las dimensiones se referencian a la imagen recortada.
- Archivo ".txt"para determina el punto de agarre de cada una de las regiones de detectadas.
  - Nombre: identifica el tipo de región.
  - Centro x: coordenada horizontal en píxeles (normalizados) del centro del punto de agarre respecto a la imagen recortada.
  - Centro y: coordenada vertical en píxeles (normalizados) del centro del punto de agarre respecto a la imagen recortada.
  - u: coordenada normalizada respecto al eje x del vector normal.
  - v: coordenada normalizada respecto al eje y del vector normal.
  - w: coordenada normalizada respecto al eje z del vector normal.

4.2. Dataset real

En desarrollo... Hablar con juan para ver que tal va.

4.3. Dataset sintético

4.3.1. Herramientas

La generación sintética de imágenes es una tarea compleja que requiere de avanzados motores y simuladores que permitan representar un entorno realista. Actualmente se trata de un campo todavía en desarrollo y por lo tanto no se dispone de herramientas específicamente diseñadas para cumplir ese objetivo. Sin embargo, si que existen sistemas/*plugins* que permiten adaptar sistemas/herramientas ya existentes para permitir la generación de *datasets* sintéticos. Para el desarrollo de este proyecto se ha escogido un sistema de esta categoría, BlenderProc. Es una *Application Programming Interface (API)* que permite controlar el programa Blender para la generación de *datasets*.

Blender

Blender es una aplicación gratuita y *open source* bajo una licencia GNU GLP que permite desarrollar proyectos 3D por completo. Cuenta con sistemas todos los sistemas necesarios para proyectos 3D: modelado, *rigging*, animación, simulación, renderizado, composición, *motion tracking*, edición de video y desarrollo de videojuegos. También permite un uso avanzado gracias a la existencia de una API basada en Python llamada *BPy*.

Es gracias a la existencia de esta API que la comunidad puede desarrollar extensiones para Blender y así expandir las capacidades de este. Esto es de vital importancia para el desarrollo de este proyecto ya que para el desarrollo del *dataset* sintético se ha empleado un *pipeline* que facilita y automatiza la creación de *datasets*.

BlenderProc

BlenderProc es un *pipeline* desarrollado por German Aerospace Center (DLR) - Institute of Robotics and Mechatronics (RM) (DLR-RM) con el fin simplificar el proceso de generación de *datasets* centrados en imágenes para entrenar redes convolucionales [Den+19]. Se caracteriza por centrarse en la modularidad al dotar a blender de un sistema con las herramientas necesarias para generar imágenes foto realistas pero que manteniendo una estructura modular que permite adaptarse al problema.

Algunas de las utilidades de este *pipeline* son: cargar, modificar, eliminar escenas y objetos, variar la iluminación, la composición de la escena, aplicar simuladores de físicas para obtener escenas realistas, renderizar imágenes de color, profundidad, distancia y normales. Y todo esto se realiza mediante *scripts* que llaman a la API de BlenderProc y esta se encarga de cargar y controlar Blender.

Aruco

Empleando Blender se puede renderizar imágenes foto realistas pero se sigue requiriendo del punto de agarre y del vector normal a dicho punto. Para ello se empleara un sistema de posprocesamiento basado en Aruco [AVA20] que permite la extracción de la información restante. Aruco es una librearía basada en OpenCV diseñada para aplicaciones de realidad aumentada cuya principal ventaja y objetivo es la simplicidad. Permite detectar QR con un simple linea de código, el altamente eficiente y permite trabajar con múltiples diccionarios.

4.3.2. Arquitectura del generador de datasets

El objetivo del generador de imágenes es obtener un número elevado de muestras con las que poder entrenar varios modelos de redes neuronales. Y tal como se ha definido en la Sección 4.1, las muestras deben de estar constituidas por varias imagen así como información respecta a la posición de las piezas en la imagen e información respectos a los puntos de agarre. Desgraciadamente, no se ha podido determinar un único sistema que soporte la generación de las múltiples imágenes así como la definición de los puntos de agarre y por ello se ha tenido que crear una capa de posprocesamiento que permita extraer la información adicional necesaria.

Generación de imágenes

Como se puede observar en la Figura 4.1, se trata de un proceso iterativo constituido por varias etapas que se ejecutan en serie para la generación de cada nueva imagen. A excepción de

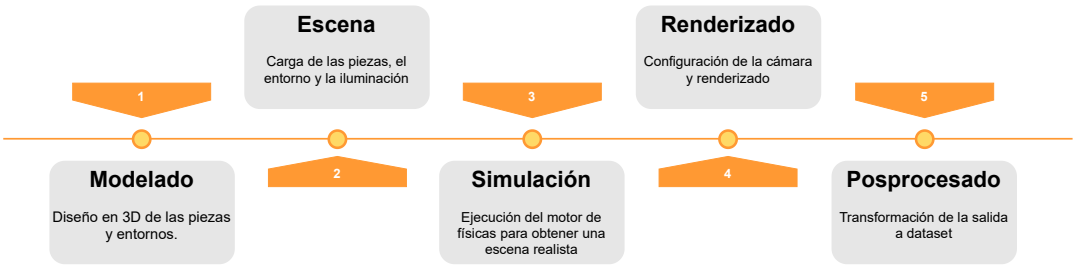

Figura 4.1. Esquema de la arquitectura del sistema

la primera etapa que consiste en la generación de los modelos 3D, esta etapa no forma parte del generador pero este depende de esta.

En esta etapa se han diseñado múltiples piezas de diferentes tamaños con la ayuda de Blender así como múltiples entornos sobre los que más adelante se dispondrán las piezas. También se ha generado las texturas necesarias para las piezas las cuales incluyen defectos y patrones que se varían para cada instancia de estas añadiendo así aleatoriedad. Para los entornos emplearán una textura aleatoria para cada imagen de una librearía constituida por más de mil seiscientas texturas.

Partiendo de los modelos y texturas necesarios, el sistema desarrollado puede empezar a generar imágenes:

1. Escena: Carga del entorno, de las piezas, la iluminación y la cámara

    - Entorno: Se escoge de forma aleatoria uno de los entornos disponibles y se carga dentro de blender. A continuación, se escoge de forma aleatoria una de las texturas disponibles y se aplica en la totalidad del entorno. Por último, se centra el entorno y se deshabilita el cuerpo dentro del motor de físicas. De esta forma se fija la posición del entorno y este pasa a ser inalterable. Pero el resto de objetos si que se verán afectados por el entorno.

    - Piezas: Se escoge de forma aleatoria el número de piezas que estarán presentes en la escena (se establecen mínimos y máximos dependiendo del tamaño de la pieza) y se cargan junto con sus texturas dentro de blender. Por último, se define una propiedad *category_id* que más adelante será usada para identificar las distintas piezas.

    - Iluminación: Se parte de la base de que el entorno de trabajo real estará bien iluminado. Es por ello que se han definido unas condiciones de iluminación fijas y óptimas que permitan reducir las sombras. Originalmente se planteo usar condiciones de iluminación aleatorias pero estas daban lugar a imágenes oscuras en las que apenas se podían distinguir las piezas. El efecto de las sombras se ve incrementado debido a la gran cantidad de piezas que presentas texturas oscuras.

    - Cámara: Para asemejarse a la realidad se ha fijado una cámara cenital a la escena y se ha implementado las parámetros intrínsecos de la cámara real que se va a emplear dentro de blender.

2. Simulación: activación de las piezas, posicionamiento de las piezas y simulación física.

    - Activación de las piezas: se activa el motor de físicas para todas las piezas de forma que se vea afectadas por la gravedad y el entorno.

- Posicionamiento de las piezas: con la ayuda de la API de blenderproc se posicionan todas las piezas de forma aleatoria dentro de un volumen de trabajo. El volumen de trabajo se fija en base a las dimensiones del entorno y la posición y orientación de la pieza se determina de forma aleatoria con el motor de aleatoriedad de numpy.
- Simulación física: se ejecuta una simulación física en la que se deja caer las piezas sobre el entorno/zona de trabajo. Se ejecuta el motor hasta que todas las piezas se vuelvan estáticas o se alcance el máximo tiempo de simulación de cuatro segundos.

3. Renderizado: configuración del motor de renderizado, renderizado y escritura de la salida de blender.
   - Configuración: dependiendo del tamaño de la pieza se fija la configuración necesaria. Las piezas grandes requieren de una mayor resolución y detalle para la capa de posprocesado que se desarrollará en la **??**.
     - Piezas grandes: resolución 3840x2160, 50 muestras y se activa el mapa de normales y de profundidad.
     - Piezas pequeñas: resolución 1920x1080, 25 muestras y no se requiere de una mapa de normales ni profundidad.
   - Renderizado: se activa el proceso o procesos de renderizado dependiendo del tamaño de la pieza.
   - Salida: se transfiere el resultado de los renderizados del archivo temporal de trabajo de blender al lugar de salida deseado. Para las piezas grandes se guardan también la posición de las piezas respecto a la imagen en un archivo ".txt".

Uno de los factores más importantes durante el desarrollo de este sistema es el sesgado. Se debe de evitar que el sistema genere un *dataset* sesgado lo cual implica comprobar y asegurar la aleatoriedad de los resultados obtenidos. Para ello se empleará siempre que sea posible el motor de aleatoriedad de numpy. Este se caracteriza por ser capaz de obtener distribuciones uniformes (la mayor expresión de la aleatoriedad ya que todos los resultados son equitativamente probables).

Posprocesado

Desarrollar y explicar en detalle como funciona el sistema de posprocesado.

Pasos par explicar el posprocesado:

1. Procesado para YOLO: e parte de la base de que el entorno de trabajo real estará bien iluminado. Es por ello que se han definido unas condiciones de iluminación fijas y óptimas que permitan reducir las sombras. Originalmente se planteo usar condiciones de iluminación aleatorias pero estas daban lugar a imágenes oscuras en las que apenas se podían distinguir las piezas. El efecto de las sombras se ve incrementado debido a la gran cantidad de piezas que presentas texturas oscuras. Se escoge de forma aleatoria el número de piezas que estarán presentes en la escena (se establecen mínimos y máximos dependiendo del tamaño de la pieza) y se cargan junto con sus texturas dentro de blender. Por último, se define una propiedad *category_id* que más adelante será usada para identificar las distintas piezas.
2. Procesado para tiny YOLO

Figura 4.2. Filtrado por color

3. Procesado para Regresor

4.4. Resultados

Mostrar múltiples resultados así como la riqueza del dataset

5

Objetivos de desarrollo sostenible

Análisis comparativo del proyecto frente a los objetivos de desarrollo sostenible. Desde un punto de visto social, ecológico y económico

El 25 de septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos como parte de una nueva agenda de desarrollo sostenible. Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años.

Figura 5.1. Objetivos de desarrollo sostenible aprobados en 2015

En total se han planteado 17 objetivos para llevar a cabo, este proyecto se enmarca dentro de tres de ellos y a continuación se van a desarrollar de forma individual.

- Objetivo 8 - Trabajo decente y crecimiento económico: El proceso de automatización de la industria supone la creación de nuevos puestos de trabajo más especializados y con mejores condiciones laborales. Implica un crecimiento de la producción sin un aumento notable de costes y por lo tanto implica un desarrollo económico. Pero par ello es importante controlar este desarrollo y evitar el sobre uso de estas máquinas y con ello el reemplazo del ser humano en el sector industrial.
- Objetivo 9 - Industria, innovación e infraestructura: Gracias al empleo de las nuevas tecnologías para la automatización de los procesos industriales se ha logrado un crecimiento y desarrollo económico en la industria. Una de las metas de este objetivo es la globalización de estas tecnologías y el facilitar el acceso a estas para los países en vías de desarrollo. Con este proyecto se ha desmostado y creado un sistema reentrenable y usable en el proceso de automatización y esto sin suponer un gran coste de desarrollo. La tecnología desarrollada en este proyecto es de acceso público y modificable para implantar en diferentes sistemas.
- Objetivo 12 - Producción y consumo responsable: Con la implantación de los robots en la industria no solo se obtiene un incremento en la producción. También, se obtiene una reducción en el número de errores cometidos durante el proceso de fabricación. Esto implica que menos recursos deben de ser usados en la producción. Además, los robots son capaces de desarrollar tareas imposibles para un ser humano y con diferentes materiales. Gracias a sus capacidades, se pueden desarrollar productos más complejos o innovadores centrados en mejorar la sostenibilidad y con un empleo más ecológico de los recursos.

Es por todo esto que se considera que este proyecto puede conllevar mejoras para la sociedad a nivel económico, ecológico y social. Siempre y cuando la implantación de estos sistemas se realice correctamente y teniendo en cuenta a la mano de obra humana ya existente. Estos sistemas no deben de reemplazar sino ayudar a este sector.

Bibliografía

[ABB] ABB. IRB 1400 YuMi, dirección: https://new.abb.com/products/robotics/es/robots-colaborativos/yumi (visitado 16-11-2021).

[Aca] K. Academy. El gradiente, dirección: https://es.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/partial-derivative-and-gradient-articles/a/the-gradient (visitado 18-06-2020).

[Alo+18] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. V. Esesn, A. A. S. Awwal y V. K. Asari, *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*, 2018. arXiv: 1803.01164 [cs.CV].

[AMS18] I. Ahmad, I. Moon y S. J. Shin, «Color-to-grayscale algorithms effect on edge detection — A comparative study,» en *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, 2018, págs. 1-4.

[AT13] A. Andreopoulos y J. Tsotsos, «50 Years of object recognition: Directions forward,» *Computer Vision and Image Understanding*, vol. 117, págs. 827-891, ago. de 2013.

[AVA20] AVA. (2020). ArUco: a minimal library for Augmented Reality applications based on OpenCV, dirección: https://www.uco.es/investiga/grupos/ava/node/26.

[BHM20] J. Burnham, J. Hardy y K. Meadors, «Comparison of the Roberts, Sobel, Robinson, Canny, and Hough Image Detection Algorithms,» jun. de 2020.

[Cas+13] R. Casanelles, R. Pons, Xiaolong Feng, R. Patel, D. Wäppling, J. Weström y H. Andersson, «Towards high performance robotic solutions in press automation - An ABB view,» en *IEEE ISR 2013*, 2013, págs. 1-3.

[Den+19] M. Denninger, M. Sundermeyer, D. Winkelbauer, Y. Zidan, D. Olefir, M. Elbadrawy, A. Lodhi y H. Katam, «BlenderProc,» *arXiv preprint arXiv:1911.01911*, 2019.

[DH72] R. O. Duda y P. E. Hart, «Use of the Hough Transformation to Detect Lines and Curves in Pictures,» *Commun. ACM*, vol. 15, n.º 1, págs. 11-15, ene. de 1972. dirección: https://doi.org/10.1145/361237.361242.

[FB81] M. A. Fischler y R. C. Bolles, «Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,» *Commun. ACM*, vol. 24, págs. 381-395, 1981.

[GSA16] F. Giménez Palomares, J. Serrá y E. Alemany, «Aplicación de la convolución de matrices al filtrado de imágenes,» *Modelling in Science Education and Learning*, vol. 9, pág. 97, ene. de 2016.

[IK88] J. Illingworth y J. Kittler, «A Survey of the Hough Transform,» *Comput. Vision Graph. Image Process.*, vol. 44, n.º 1, págs. 87-116, ago. de 1988. dirección: https://doi.org/10.1016/S0734-189X(88)80033-1.

[Inta] Intel. Realsense L515, dirección: https://www.intelrealsense.com/lidar-camera-l515/ (visitado 10-11-2021).

[Intb] ——, SDK for Intel Realsense, dirección: https://github.com/IntelRealSense/librealsense/releases (visitado 10-11-2021).

[KSH17] A. Krizhevsky, I. Sutskever y G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks,» *Commun. ACM*, vol. 60, n.º 6, págs. 84-90, mayo de 2017. dirección: https://doi.org/10.1145/3065386.

[Lab] H. Labs. History (1961-1980). Castellano.

[Lec+89] Y. Lecun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard y L. Jackel, «Backpropagation Applied to Handwritten Zip Code Recognition,» *Neural Computation*, vol. 1, págs. 541-551, dic. de 1989.

[LR90] M. E. Lee y R. A. Redner, «A note on the use of nonlinear filtering in computer graphics,» *IEEE Computer Graphics and Applications*, vol. 10, n.º 3, págs. 23-29, 1990.

[Mor+17] D. Morrison, A. W. Tow, M. McTaggart, R. Smith, N. Kelly-Boxall, S. Wade-McCue, J. Erskine, R. Grinover, A. Gurman, T. Hunn, D. Lee, A. Milan, T. Pham, G. Rallos, A. Razjigaev, T. Rowntree, K. Vijay, Z. Zhuang, C. Lehnert, I. Reid, P. Corke y J. Leitner, *Cartman: The low-cost Cartesian Manipulator that won the Amazon Robotics Challenge*, 2017. arXiv: 1709.06283 [cs.RO].

[Oll05] A. Ollero Baturone, *ROBOTICA. Manipuladores y Robots Móviles*, Castellano, S. MARCOMBO, ed. 2005, 464 págs.

[Pér15] E. Pérez-López, «Los sistemas SCADA en la automatización industrial,» *Revista Tecnología en Marcha*, vol. 28, págs. 3-14, dic. de 2015.

[Sze11] R. Szeliski. (2011). Computer vision algorithms and applications, dirección: http://dx.doi.org/10.1007/978-1-84882-935-0.

[Vid19] A. Vidhya. (2019). Brief History of Neural Networks, dirección: https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec.