DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Trabajo de Fin de Máster - curso 2018/2019

Desarrollo de dataset personalizado para entrenamiento de YOLO como sistema de detección de objetos en tiempo real, para entorno con brazo robot

Autor: Lisardo Fernández Cordeiro
Tutor: Dr. Roberto Paredes Palacios

10 de diciembre de 2018

Resumen

Nos encontramos en una etapa crucial de la evolución tecnológica; en los albores de una nueva y revolucionaria era donde el ser humano, muy posiblemente, se ayude y comparta mucho de su espacio habitual con máquinas multifuncionales, capaces de operar en su mismo entorno.

La robótica industrial lleva años formando parte de los sistemas productivos. Su principal motivación es la sustitución del hombre en aquellos trabajos que, por su repetitividad o especial peligro, salvaguardan tanto la salud del operario como el nivel de calidad requerido.

Además, hace relativamente poco tiempo que la robótica ha tomado otro cariz, otra dimensión: La robótica de servicios como ayuda al ser humano en su día a día y en labores ajenas al entorno puramente industrial y productivo.

En este escenario, la Inteligencia Artificial (IA), en su concepción más amplia, está teniendo un rol fundamental. La IA se ha convertido en el eje vertebrador de los avances tecnológicos globales por su capacidad para predecir, determinar tendencias, encontrar patrones, clasificar, aprender, etc. sobre los, cada vez más numerosos, datos disponibles. Esta capacidad está contribuyendo a que la simbiosis IA-robótica represente una nueva era tecnológica.

Este Trabajo Fin de Máster (TFM) pretende aprovechar el estado del arte en detección de objetos, la madurez de las tecnologías de control sobre accionadores y los conocimientos adquiridos en las diferentes asignaturas comprendidas en los estudios a punto de culminar, para implementar un *Sistema de detección de objetos personalizados* con el que dar respuesta a una necesidad real y acceder, de este modo, al nuevo mercado que se abre ante nosotros.

El origen del proyecto parte de una consulta al Dr. Roberto Paredes, director de este TFM, sobre la búsqueda de sistemas para detección y localización de objetos con el que informar a un brazo robot en entorno industrial. Visitada una empresa de mecanizado y soldadura de pequeñas piezas, estas labores las realizan operarios en condiciones de alta repetitividad, muy bajo estímulo y elevado riesgo, lo que representa un entorno laboral pernicioso y un buen punto de partida para desarrollar un trabajo que se pueda hacer extensivo a la robotica de servicio y asistencial.

Por ello, se han diseñado una serie de herramientas software con las que probar diferentes modos de preparación de la base de datos con la que entrenar un sistema de detección de objetos en tiempo real como es YOLO (You Only Looks Once), desarrollado por Joseph Redmon y Ali Farhadi de la Universidad de Washington. Este sistema termina comunicando a un brazo robot la localización y la clase de objeto detectado para que realice las labores correspondientes.

Resum

Ens trobem en una etapa crucial de l'evolució industrial; en les albors d'una nova i revolucionària era, en què l'ésser humà, molt possiblement, s'ajudarà de màquines multifuncionals capaces d'operar en el seu propi entorn amb les quals compartirà molt del seu espai habitual.

La robòtica industrial fa anys que forma part dels sistemes productius. La seua principal motivació és la substitució de l'home en aquells treballs que, per repetitivitat o especial perill salvaguarden tant la salut de l'operari com el nivell de qualitat requerit.

Tanmateix, no ha estat fins fa molt poc que la robòtica ha pres un altre caire, una altra dimensió: La robòtica de serveis, que ajuda l'ésser humà en el dia a dia i en tasques alienes a l'entorn purament industrial i productiu.

En aquest escenari, la Intel·ligència Artificial (IA) , en la seua concepció més àmplia, està tenint un rol fonamental. La IA s'ha convertit en l'eix vertebrador dels avanços tecnològics globals per la seua capacitat per a predir, determinar tendències, trobar patrons, classificar, aprendre, etc. sobre els, cada vegada més nombrosos, dades disponibles. Esta capacitat està contribuint que la simbiosi IA-robótica represente una nova dimensió tecnològica

Aquest Treball Fi de Màster (TFM) pretén aprofitar l'estat de l'art en detecció d'objectes, la maduresa de les tecnologies de control sobre accionadores i els coneixements adquirits en les diferents assignatures compreses en els estudis a punt de culminar, per a implementar un *Sistema de detecció d'objectes personalitzats* amb el que donar resposta a una necessitat real i accedir, d'esta manera, al nou mercat que s'obri davant de nosaltres..

L'origen del projecte part d'una consulta al Dr. Roberto Paredes, director d'este TFM, sobre la busca de sistemes per a detecció i localització d'objectes amb què informar un braç robot en entorn industrial. Visitada una empresa de mecanitzat i soldadura de xicotetes peces, estes labors les realitzen operaris en condicions d'alta repetitividad, molt baix estímul i elevat risc, la qual cosa representa un entorn laboral perniciós i un bon punt de partida per a desenrotllar un treball que es puga fer extensiu a la robotica de servici i assistència.

Per això, s'han dissenyat una sèrie de ferramentes software amb què provar diferents modes de preparació de la base de dades amb què entrenar un sistema de detecció d'objectes en temps real com és YOLO (You Only Looks Once), desenvolupat per Joseph Redmon i Ali Farhadi de la Universitat de Washington. Aquest sistema acaba comunicant a un braç robot la localització i la classe de l'objecte detectat perquè realitze les labors corresponents.

Abstract

We are at a crucial stage in the industrial evolution; at the dawn of a new revolutionary era, where human beings will possibly find help and share an important part of their personal space with multifunctional machines capable of operating in a common environment.

Industrial robotics have been a part of the productive systems in the ultimate years. Their main aim is the substitution of manpower, which due to repetitiveness or special hazard, safeguard both the health of the operator as well as the required quality level.

However, robotics has recently taken another turn, another dimension: Robotics Service, that help human beings in their everyday life and in tasks beyond a purely industrial and productive environment.

In this scenario, Artificial Intelligence (AI), in its broadest conception, is having a fundamental role. AI has become the backbone of global technological advances because of its ability to predict, determine trends, find patterns, classify, learn, etc. on the, increasingly numerous, data available. This capacity is contributing to the AI-robotics symbiosis representing a new technological dimension.

This Final Master's Project (TFM) aims to take advantage of the state of the art in the detection of objects, the maturity of control technologies over actuators and the knowledge acquired in the different subjects included in the studies that are about to be completed, to implement a *System of detection of personalized objects* with which to respond to a real need and access, in this way, the new market that opens before us.

The origin of the project is based on a consultation with Dr. Roberto Paredes, director of this TFM, on the search of systems for detection and location of objects with which to inform a robot arm in an industrial environment. Visited a company of machining and welding of small parts, these tasks are performed by operators in conditions of high repetitiveness, very low stimulus and high risk, which represents a pernicious work environment and a good starting point to develop a work that can be done extensive to service and assistance robotics.

For this reason, a series of software tools have been designed with which to test different ways of preparing the database with which to train an object detection system in real time, such as YOLO (You Only Looks Once), developed by Joseph Redmon and Ali Farhadi of the University of Washington. This system ends up communicating to a robot arm the location and the class of object detected so that it performs the corresponding tasks

Agradecimientos

Llegados a un hito tan relevante como la confección y defensa del Trabajo Final de Máster, a uno le asaltan todas las muestras de ánimo y confianza recibidas a lo largo de la vida. Incluso la carga de desánimo, desconfianza o descrédito asoman como forja sobre la que se ha fraguado parte del carácter. Pues no es vano el camino recorrido hasta el punto en el que nos encontramos.

Si estoy aquí es, desde luego, porque me parió una madre. Gallega y rebelde que me sigue suspirando y mirando con los mismos ojos de emoción y confianza ciega que al nacer. Para llegar a ese momento, se precisó la intervención de un padre. Humilde y luchador manchego, emigrante en la Alemania de posguerra y cuya primera frase al verme fue "este niño va a estudiar". Frase que revoloteaba en mi cabeza mientras lo despedía para siempre. Sí papá. En eso sigo.

Es difícil abstraerse del paso por la Escuela Universitaria de Ingeniería Técnica Industrial (EUITI-UPV), a la que debo el nacimiento de una gran pasión: Los robots y cómo hacer que aprendan. Razón fundamental de este trabajo. Y del paso por la Escuela Técnica Superior de Ingeniería (ETSE-UV) desde donde parte la oportunidad de realizar el presente Máster.

Sería insensato pasar por alto la encomiable dedicación de los profesionales que atesora el DSIC. Todos y cada uno de ellos se han ganado mi más profundo respeto y agradecimiento por sus desvelos en la labor docente. Si algo sé, es por ellos.

Pero poco o nada de lo que aparece en la presente memoria tendría sentido sin la inestimable ayuda, guía y sabiduría de Ricardo Paredes, director y alma de este proyecto. Su capacidad de insuflar ilusión permanente por el descubrimiento de las pequeñas cosas que conforman los grandes objetivos y su irreductible ánimo, han rejuvenecido mi forma de mirar la inteligencia artificial, la investigación y el mundo.

Por supuesto, a mis compañeros de clase. Insultántemente jóvenes, llenos de vida, capacidad e ilusiones que seguro llevarán a buen término. Sobre todo a Juan por su enorme talento y generosidad y a Zuska por su confianza.

Agradecer a la Universidad de Valencia, a través del subprograma "Atracció de Talent", en el marco de la convocatoria "Aprèn a investigar", la concesión de la beca que me empujó directamente a este Máster del DSIC-UPV que con este TFM se pretende cerrar.

A Héctor y Carla, mis hijos, que con sus risas, su vitalidad y sus sabias preguntas, me han reflotado en más de una ocasión de "desbordamiento de pila", al relativizar lo que aparentaban ser grandes problemas.

Y a Mluz. La mujer que me da la sensación de acompañar desde siempre. Vital y necesaria en todas y cada una de las etapas importantes de mi vida. Cuanto más, ésta que acaba. Confidente, fiel, perseverante y capaz donde las haya. Su siempre presente confianza y aliento han logrado conducirme sano y salvo a este puerto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

INTRODUCCIÓN

La robótica industrial, dedicada a la producción en serie, se ha manejado siempre en ambientes cerrados y controlados, con funciones de trabajo muy concretas y limitadas y con sistemas de control fijos y cerrados. Su implantación ha supuesto una mejora sustancial en la calidad de los productos. Además, el papel desempeñado por los operarios, con la incorporación de los robot a las líneas de producción más peligrosas y repetitivas, ha mejorado tanto en la salvaguarda de su seguridad física como en el nivel de preparación técnica que se le exige.

La investigación robótica, sin embargo, ha diseminado su alcance sobre infinidad de campos en los que la intervención de máquinas multifuncionales puede ayudar al ser humano a resolver situaciones complejas o labores peligrosas fuera del ambiente productivo.

Así, el desarrollo de algoritmos que conecten, con los sistemas inteligentes adecuados, la percepción y la acción ante los cambios del entorno en el que se mueve la máquina, abre el campo de aplicación de la robótica al mundo. Campos tan variados como la robótica protésica, de servicios, asistencial, de rescate, ... están en continuo avance en este terreno.

No solo se precisan sensores, actuadores, sistemas de control y redes de comunicaciones lo suficientemente maduros para vencer el escalón entre la investigación y la aplicación práctica que el mercado demanda. La percepción del entorno, la propiacepción, la toma de decisiones y el aprendizaje se convierten en la piedra angular sobre la que contruir sistemas robóticos capaces de interactuar en entornos con humanos.

Es precisamente ahora cuando los fabricantes de componentes, sensores y sistemas empotrados, ofrecen hardware económico con suficiente calidad y potencia. Y es precisamente ahora cuando la investigación en IA ofrece resultados con suficiente fiabilidad como para afrontar el reto de la producción y comercialización de robots de servicio.

Con este estado de la tecnología, el presente proyecto pretende apoyarse en dispositivos de control y sensorización comerciales, como son las cámaras USB, pa-

ra definir un sistema de desarrollo de un conjunto de entrenamiento personalizado para entrenar un detector de objetos en tiempo real como es YOLO (You Only Looks Once). Al ser YOLO el *estado del arte* en sistemas de detección, localización y clasificación de objetos en tiempo real, se considera un trabajo de interés adquirir conocimiento y aplicar esta tecnología en entornos específicos, con objetos personalizados.

En sí misma no se trata de una idea novedosa, puesto que el propio sistema ya ofrece la posibilidad de entrenarlo con conjuntos personalizados, pero se pretende desarrollar herramientas que puedan automatizar el proceso de localización y etiquetado del conjunto de objetos. Aspecto diferenciador de los productos existentes.

Se trata pues, de desarrollar un conjunto de entrenamiento para el sistema de detección en tiempo real YOLO, partiendo de una cabeza robotizada con dos cámaras. Una para la detección y localización de objetos a partir de su contorno y otra para capturar la imagen RGB del objeto. Con ello se entrena YOLO y se prueba con un conjunto de test compuesto por imagenes fijas y con una cámara USB conectada al sistema. La salida del sistema se utilizaría para informar a un brazo robot de la localización del objeto para que realice las labores adecuadas al objeto detectado.

El objetivo del proyecto no alcanza a ofrecer un sistema completo donde el brazo robot actúe, puesto que su consecución en tiempo y recursos es incompatible con la dimensión de este TFM. Sin embargo, sí será capaz de estructurar e implementar la arquitectura básica necesaria para su consecución futura.

Por lo tanto, este Trabajo Fin de Máster representa el análisis, diseño e implementación de todos los módulos necesarios para lograr una funcionalidad cercana a la que se acometerá en pasos sucesivos, sirviendo de base sólida para ello.

La estructura de la memoria es la siguiente:

- En este primer capítulo se ha descrito someramente el contenido de la memoria
- En el capítulo 2 se describe la motivación y los objetivos que se persiguen con este TFM
- El capítulo 3 analiza el estado de la investigación. Se muestra una visión general de los sistemas de detección en tiempo real, así como diferentes soluciones para el problema concreto de detección de objetos. También se repasan las diferentes alternativas en los componentes hardware necesarios en cada uno de los módulos en los que se divide la arquitectura del robot (GPUs, microcontroladores, SOC vs PC, cámaras, etc.). Por supuesto, se repasan las opciones software tanto a nivel de interface como de base de datos precisas para la implementación.
- En el capítulo 4 se detallan los requisitos, tanto funcionales como no funcionales, con los que elaborar la especificación del sistema adecuada al problema propuesto. A partir de esta especificación, se elabora la planificación del proyecto, valorando su alcance temporal. Adicionalmente, se realiza un estudio de viabilidad del trabajo con el que analizar las posibilidades de convertirlo en un proyecto empresarial.

- El capítulo 5 está íntegramente dedicado al desarrollo pormenorizado del proyecto. Se divide en tres apartados que definen la secuencia operativa lógica en todo proyecto apoyado en la buenas prácticas de Ingeniería Informática. Se inicia con las historias de usuario que definirán pormenorizadamente los objetivos que se pretenden de un modo racional. Una vez definido el primer paso, se detalla el backlog o estructura de implementación de la interfaz de operador, los procesos y datos de cada uno de los módulos integrantes y la interconexión entre ellos. Se cierra el capítulo con la pizarra Kanban para implementación del diseño definido.
- La implementación de cada uno de los hitos marcados en el capítulo anterior se pormenorizan en el capítulo 6, dedicado al trabajo directo con la *ConvNet* ***YOLOv3***.
- En el capítulo 7 se presenta una batería de pruebas para el conjunto del sistema. El objetivo es evaluar la consecución de la funcionalidad inicial y medir el rendimiento del sistema.
- El capítulo 8 arroja la conclusiones obtenidas del desarrollo del TFM así como los trabajos futuros derivados del mismo.

CAPÍTULO 2

MOTIVACIÓN Y OBJETIVOS

2.1. Motivación

Basar el Trabajo Fin de Máster en el desarrollo de un sistema de IA orientado a la robótica de servicio es una decisión tomada en el mismo momento que se depositaba la solicitud para el ingreso en el MIARFID en julio de 2015. A partir de aquí, la definición del tipo de trabajo que debía cubrir el sistema motivo del TFM ha supuesto un intenso debate entre campos por explorar y necesidades reales por resolver.

En este proceso, una consulta al director de este proyecto, sobre la posibilidad de detectar objetos en tiempo real con una alta tasa de fiabilidad para informar a un brazo robot, se convierte en una opción seria de investigación.

La robótica, tanto industrial como de servicio y asistencial, tiene un camino muy largo por recorrer. Ha demostrado sobradamente su utilidad y rendimiento en los sistemas automatizados de producción. Pero con la llegada de sistemas de control basados en IA, multiplicará su repercusión más allá del ámbito fabril, como ya está empezando a suceder.

Una parte muy importante en la relación de la robótica con los sistemas de control basados en IA será la correcta alimentación del sistema con datos del entorno, del propio robot y de los objetivos a alcanzar en cada momento. Para ello se precisan sensores adecuados y sistemas capaces de obtener conocimiento de la información suministrada por los mismos.

La detección, clasificación y localización de objetos en el campo de visión del sistema es una pieza importante. Esta pieza adquiere especial relevancia cuando se piensa en robots móviles, conducción autónoma, drones de vigilancia o asistencia, robots asistenciales con interacción en entornos con humanos y un largo etcétera.

También en el ámbito industrial cobra importancia la capacidad de visión para acceder a trabajos manuales repetitivos, como puede ser la alimentación de una máquina de soldadura por puntos de pequeñas piezas en series no lo suficientemen-

te grandes. La automatización básica de este tipo de procesos es extremadamente compleja y difícil, alejando su implantación por su baja o nula rentabilidad, creando una alta dependencia humana sobre un trabajo repetitivo.

Nos encontramos, pues, con un amplio abanico de utilidades y necesidades con soluciones actuales muy costosas e ineficientes y con trabajos pendientes de mejora por el elevado riesgo en seguridad y salud para los operarios.

Disponer de un sistema de visión inteligente y versátil, capaz de acomodarse con facilidad a conjuntos de objetos diferentes y que pueda alimentar un sistema robotizado con suficiente fiabilidad, supondría un importante avance en la relación de los robots con entornos humanos.

2.2. Objetivos

El objetivo principal de este TFM es el desarrollo de herramientas para generar conjuntos de datos de entrenamiento para un sistema de detección de objetos en tiempo real como es YOLO y probar su efectividad.

Estas herramientas ayudarán a testar el sistema con rapidez, proyectando, fuera del alcance de este TFM, su capacidad hacia la automatización de este proceso e incluso su derivacón hacia el autoaprendizaje sobre objetos que no pertenecen a conjuntos de datos generado.

La consecución de este objetivo lleva asociada una serie de objetivos secundarios que conforman, en sí mismos, módulos independientes con diseño y capacidades distintivas.

- Control de cabeza robotizada con dos cámaras USB con las que obtener imagenes de objetos detectados y la etiqueta de clase y localización en modo semiautomático.
- Conocer el diseño y las características del sistema de detección en tiempo real YOLO para generar los conjuntos de datos correctamente y operar eficientemente.
- Analizar y entender las herramientas existentes para el diseño de un interfaz adecuado para la operación con las cámaras.
- Diseñar herramienta que amplíe el dataset original generado con las cámaras.
- Desarrollar herramienta de ayuda al etiquetado y captura de imagen de objetos desde un archivo de vídeo.
- Establecer conexión entre sistema de detección y brazo robot para su manipulación.

CAPÍTULO 3

ESTADO DE LA INVESTIGACIÓN

La robótica de servicios se encuentra en proceso de despegue comercial. La variedad de trabajos que es capaz de abarcar es inmensa. De hecho, se espera que, en menos de 15 años, muchas labores realizadas por humanos en el terreno cotidiano, pasen a resolverlas robots.

Para ello, los sistemas de detección, localización y clasificación de objetos en tiempo real son cruciales. Esta parte de la IA, la visión por computador, está experimentando continuos y sorprendentes avances. Tal es así, que, si bien pasaron sus buenos 14 años desde la publicación en 1998 por LeCun et al [4], del uso las redes convolucionales "LeNet-5" para reconocimiento de documentos hasta la publicación, por parte de Alex Krizhersky et al [5] en 2012, de la primera red convolucional profunda para clasificación de imágenes, en estos últimos seis años la eclosión de sistemas, tecnologias, arquitecturas y modelos ha hecho posible que la visión por computador experimente un avance superlativo.

Este capítulo pretende repasar diferentes aspectos esenciales que sirvan para conocer el estado del arte tanto del plano donde se mueve el TFM, como los elementos esenciales que precisa para su consecución.

Así, el apartado 3.1 ofrece una visión normalizada de la robótica de servicios y su importancia creciente en el mundo, evidenciando la relevancia que la visión por computador, motivo del presente TFM, adquiere en el desarrollo de los robots. Las soluciones disponibles para la implementación de sistemas de visión por computador se pueden ver en el apartado 3.2. El detalle de YOLO como sistema de detección, clasificación y localización en tiempo real se desarrolla en el apartado 3.3.

3.1. Robots de servicio

Con objeto de clarificar y acotar convenientemente el ámbito en el que se pretende englobar el TFM, es conveniente conocer el estado actual de la robótica de servicios y el alcance comercial que representa.

La robótica de servicios[1] se caracteriza por englobar robots con multitud de formas y estructuras, así como gran diversidad de áreas de aplicación.

Desde plataformas móviles armadas con brazos robóticos, hasta plataformas estáticas sin brazos ni herramientas, se puede encontrar un gran abanico de combinaciones. Además, así como los robots industriales suelen ser mayoritariamente estáticos, los robots de servicios pueden prestar su capacidad de movimiento y/o maniobrabilidad para trabajos de teleoperación o telepresencia.

Con todo, la Comisión Económica de Naciones Unidas para Europa (UNECE), junto con la Federación Internacional de Robótica (IFR)[3], llevan trabajando desde 1995 en la definición de la robótica de servicios así como en un esquema de clasificación adecuado. Este esfuerzo se ha visto reflejado en el nuevo estándar ISO-8373, efectivo desde 2012.

3.1.1. Definición

La manera más sencilla de definir un robot de servicio es como todo robot no destinado a labores de manufactura o fabricación. Pero veamos a continuación un extracto de las definiciones más importantes que se encuentran en el estándar ISO-8373[1].

- Un robot es un mecanismo actuador programable en dos o más ejes, con un grado de autonomía y movimiento dentro de un entorno tal que le permite desarrollar tareas. La autonomía en este contexto, se refiere a la capacidad de realizar tareas en función de su estado actual y de la información percibida por los sensores, sin intervención humana.

- Un **robot de servicio** es un robot que realiza tareas útiles para los humanos o para otro equipamiento excluyendo aplicaciones de automatización industrial. Nota: La clasificación de un robot como industrial o de servicio se realiza de acuerdo con el destino de la aplicación.

- Un **robot personal de servicio o un robot de servicio para uso personal** es un robot de servicio destinado a tareas no comerciales, normalmente para personas no expertas. Ejemplos son robots de servicio doméstico, sillas de rueda automatizadas, robots de asistencia para movilidad de personas impedidas y asistentes para ejercicios.

- Un **robot de servicio profesional o robot de servicio para uso profesional** es un robot de servicio usado para tareas comerciales, normalmente operados por un técnico operador debidamente formado. Ejemplos son robots de limpieza en centros públicos, robots de mensajería en hospitales u oficinas, robots bomberos y robots de rehabilitación o cirugía en hospitales. En este contexto, un operador es la persona designada para poner en marcha, monitorizar y parar la operativa asignada al robot o el sistema robótico.

[1]Traducción libre del autor de este TFM

3.1.2. Clasificación

La robótica de servicio, tal y como se apuntaba en el Capítulo 1, se extiende por infinidad de aplicaciones y campos que se pueden englobar en las definiciones propuestas anteriormente. Con ellas, identificaremos los diferentes robots según su tipo de relación con las personas y su aplicación.

- Robots de servicio para uso personal.
  - Robots para aplicaciones domésticas: de ayuda, compañía, asistenciales, humanoides, de limpieza en hogar, corta césped.
  - Robots de entretenimiento: juguetes, de competición, educativos y de entrenamiento.
  - Robots asistenciales: sillas de ruedas robotizadas, de rehabilitación, otras labores asistenciales.
  - Robots para transporte de personas: exoesqueletos, vehículos autónomos
  - Robots de seguridad y vigilancia en el hogar
- Robots de servicio para uso profesional
  - Robots de campo: agricultura, ganadería, busca minas, espaciales, vigilancia de bosques.
  - Robots de limpieza profesional: friega-suelos, limpia-ventanas y muros, tuberías
  - Robots de inspección y mantenimiento: instalaciones y plantas, tanques y tuberías.
  - Robots para construcción y demolición: desmantelamiento nuclear, otros tipos de construcciones y demoliciones
  - Robots en sistemas logísticos: correo, transporte interior, transporte externo.
  - Robots médicos: sistemas de diagnosis, de asistencia o terapia, rehabilitación, cirugía.
  - Robots de defensa, rescate y seguridad: bomberos, vigilancia, aéreos, terrestres, anti minas.
  - Plataformas de uso general
  - Brazos robotizados de propósito general
  - Robots relaciones públicas: hotel, guías, marketing, bibliotecarios.
  - Robots especiales, ad-hoc, humanoides.

3.1.3. Estadísticas

Atendiendo al estudio anual que realiza la IFR sobre el estado de la robótica en el mundo, proporcionando cifras de venta y expectativas para los próximos años, el número de robots de servicio para uso profesional vendidos en el año 2016 ascendió a algo menos de 60.000 unidades. El valor en ventas se elevó a 4.700 millones de dólares.

Aproximadamente, el 50 % de las ventas se destinó a aplicaciones de logística. Un 4 % son robots médicos y representan el 40 % del total de ventas, siendo uno de los principales motores de la investigación robótica. También los robots destinados a defensa suponen un monto importante de robots de servicios, alrededor del 20 %, representando un campo muy importante en desarrollo de soluciones versátiles.

Por otro lado, las cifras de ventas de robots de servicios para uso personal y doméstico se elevaron hasta los 7 millones de unidades, representando un crecimiento superior al 24 % respecto al año anterior.

Las previsiones para el período 2018-2020 en robótica de servicios para uso profesional, según el estudio de la IFR, se eleva a 400.000 unidades vendidas, con una cifra de negocio cercana a los 26.000 millones de dólares de media anual.

La motivación de este apartado ha sido, fundamentalmente, ubicar el desarrollo de este proyecto en el estado actual del sector, como receptor fundamental de sistemas de visión por computador.

3.2. Sistemas y metodologías

El diseño e implementación de un sistema de visión por computador para un robot exige la confluencia de multitud de tecnologías y metodologías, con características muy definidas.

En esta sección se repasa el estado del arte con las principales arquitecturas y mejoras de implementación que han seguido los sistemas de visión por computador.

3.2.1. Redes Neuronales Convoluciones (ConvNets)

Hasta hace no demasiado tiempo, el trabajo con redes neuronales para visión artificial estaba prácticamente abandonado. La capacidad de procesamiento, los algoritmos existentes y el reducido tamaño de los conjuntos de datos hacían muy costoso su abordaje. Entre otras soluciones, Viola-Jones [6] entrenando un perceptrón con señales filtradas por una trasformada Haar2D pasando los resultados a un clasificador SVM, y la utilización de histogramas de gradientes orientados [7], algo más lento pero más efectivo que el anterior, representaban la mejor solución en sistemas de visión artificial. De hecho, a día de hoy se siguen encontrando en multitud de sistemas comerciales, pese a sus limitaciones de aprendizaje y generalización.

El aumento de la potencia de cómputo, la capacidad de procesamiento en GPUs y la disposición de grandes conjuntos de datos convenientemente etiquetados, impul-

sa el desarrollo del aprendizaje profundo, inspirado en la arquitectura del cerebro. Este paso supone un gran salto en la capacidad de las máquinas para abordar problemas de aprendizaje.

Las ConvNets, conceptualmente muy similares a la transformada wavelet utilizada en procesamiento de señales digitales, vienen a resolver un problema importante en el diseño de redes neuronales profundas. Entrenar redes neuronales clásicas más allá de cinco niveles y con nodos suficientes para modelar adecuadamente un problema, resulta prácticamente inviable en tiempo de cómputo y capacidad de procesamiento para utilizarlas en sistemas de tiempo real. El número de parámetros que precisa una ConvNet es varios órdenes de magnitud inferior y su rendimiento generalizando un problema es escasamente inferior a las NN, tal y como se demuestra en diferentes experimentos [5]. Por otro lado, la profundidad de diseño que se puede alcanzar con las ConvNets es muy alto, lo que redunda en beneficio del aprendizaje.

Veamos unas reseñas de las aportaciones primigenias en la aplicación de las ConvNets en el campo de la visión artficial.

LeNet-5: LeCun et al [4], presenta, en 1998, una primera aplicación de las ConvNets en el tratamiento de imágenes por computador. El trabajo se desarrolla sobre el reconocimiento de texto escrito (números).

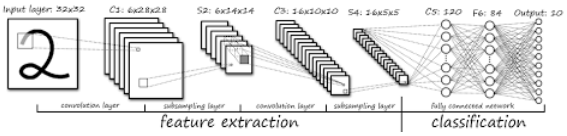


Figura 3.1: LeNet-5

Se trata de una pequeña red convolucional de cinco niveles. Dos convoluciones con filtros de 5x5 s=1, aplicando la tangente hiperbólica como función no lineal de activación, seguidos de una capa de *average pooling* cada una de ellas, para la extracción de características; una zona de clasificación formada por dos capas *fully connected (FC)* y una capa final de salida con 10 opciones evaluadas por una función *softmax* conforman la arquitectura de esta primera red con apenas 60000 parámetros y unos prometedores resultados.

AlexNet: Tuvo que esperarse hasta 2012, con el incremento de la capacidad de cómputo de las GPUs, para verificar las bondades que las ConvNets ofrecen en el campo de la visión por computador. Alex Krizhevsky et al [5] presenta, entonces, una arquitectua ConvNet más profunda.

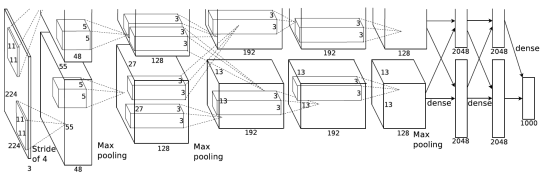


Figura 3.2: AlexNet

El diseño constaba de 5 capas convolucionales reduciendo la dimensión de los filtros conforme profundiza en la red. Existen tres partes de la red donde aplica *max pool* para reducir la dimensinalidad. La zona de clasificación final constaba de 2 capas *FC* y una última capa de salida con función *softmax* de 1000 nodos.

La función no lineal de activación elegida por su mayor velocidad de cómputo, entre otras cosas, fué *rectified linear unit (ReLU)*. Pese a sus bienvenidas características de no precisar entrada normalizada, se optó por aplicar *local response normalization (LRN)*, práctica abandonada más tarde por su escaso beneficio y su elevado coste computacional.

Para reducir el sobreentrenamiento (*overfitting*) se amplió el conjunto de entrenamiento artificialmente (*data augmentation*) así como el uso del recién llegado *dropout*, con la idea de robustecer el aprendizaje de características. Técnica ésta abandonada más tarde, al constatar la alta correlación entre nodos vecinos y la baja efectividad que supone aplicar *dropout* sobre los reducidos parámetros que maneja una ConvNet, frente a técnicas como *Batch Normalization*.

Encontramos, pues, una arquitectura referente en los sistemas posteriores de visión por computador, con alrededor de 60 millones de parámetros.

3.2.2. Evolución de las ConvNets

Debido al éxito de AlexNet [5] en el 2012, se abre la veda en el desarrollo de soluciones cada vez más potentes basadas en las ConvNets.

La evolución de las arquitecturas de las ConvNets busca resolver los diferentes problemas a los que se enfrentan los sistema de visión por computador en tiempo real. Analizar la mayor cantidad de *frames* por segundo (fps) detectando, localizando y clasificando con la mayor precisión posible no es baladí.

Así, el uso de filtros 1x1, la convolución de las ventanas deslizantes, el aumento paulatino de la profundidad de las redes, la concatenación o adición de capas a diferente nivel, el aumento de la anchura de las capas, etc. han supuesto pasos decisivos en el desarrollo de arquitecturas cada vez más complejas, cada vez más efectivas.

Repasamos los principales hitos en la investigación de ConvNets a lo largo de estos últimos años:

Network in Network (NIN): Lin et al [8] consideraron que las ConvNets, como modelos lineales de generalización, ofrecían un nivel de abstracción bajo.

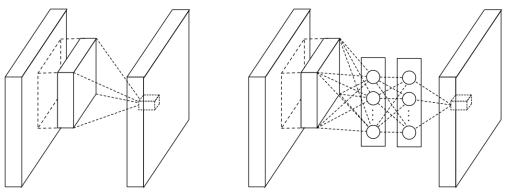


Figura 3.3: Network in Network

Para resolver esta minusvalía, propusieron sustituir los filtros de convolución por pequeñas redes neuronales, diseñadas como perceptrones multicapa, entre las capas de la ConvNet. Esta propuesta dotaría de más potencia de abstracción.

Esta propuesta lleva asociada una característica interesante. Se trata de la eliminación de la capa *maxpool*. NIN aporta mayor expresividad con el mismo resultado dimensional.

Desde este concepto se desarrollan los filtros convolucionales 1x1, a semejanza del comportamiento de una micro red neuronal entre capas, de gran utilidad en arquitecturas posteriores.

Overfeat: El análisis de las imágenes con ConvNets se llevaba a cabo con la estrategia de la *ventana deslizante multiescala*, a partir de la cual se van capturando partes de la imagen de manera ordenada. Se utilizan diferentes resoluciones de ventana en función de las características de la imagen o el tamaño de los objetos a detectar. Una solución que requiere de múltiples pasadas por la misma imagen redundando en un pobre rendimiento de la red.

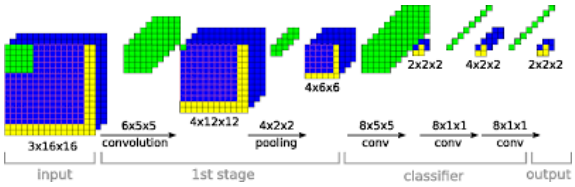


Figura 3.4: Overfeat

Sin abandonar este concepto, uno de los grandes avances en arquitecturas ConvNet lo propone Sermanet et al [9] como una solución a la detección, localización y clasificación que incluye una ventana deslizante convolucional. Con ella, el uso de la ventana deslizante se resuelve en una sola pasada de la imagen por la red, obteniendo el resultado como una matriz donde cada celda corresponde al análisis de la ventana oportuna.

VGG16: Otro de los grandes avances incorporados al campo de la visión por computador fué el trabajo presentado por Simonyan et al [10]. En él, se propone el crecimiento de las ConvNets en profundidad, aplicando el filtro mínimo con el que extraer características del contexto (3x3) en cada capa.

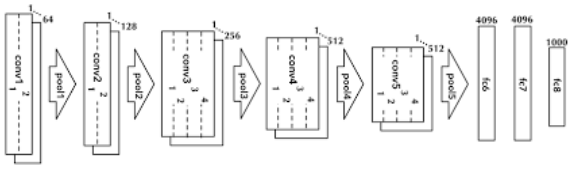


Figura 3.5: VGG16

Se trata de uno de los primeros trabajos sobre la profundidad de la arquitectura, con el que obtener menor error, en base a la extracción de características cada vez más complejas conforme crece la ConvNet, enriqueciendo su rendimiento.

ResNet: Los trabajos con arquitecturas cada vez más profundas se topan con un gran inconveniente: el desvanecimiento o explosión de los gradientes, obstaculizando la convergencia desde el principio del entrenamiento. Este problema se ha resuelto con capas de normalización intermedias, pero en redes muy profundas se produce un efecto de degradación que no se puede achacar al sobreentrenamiento.

He et al [11] proponen mitigar este problema introduciendo una novedosa forma de operar con las capas.

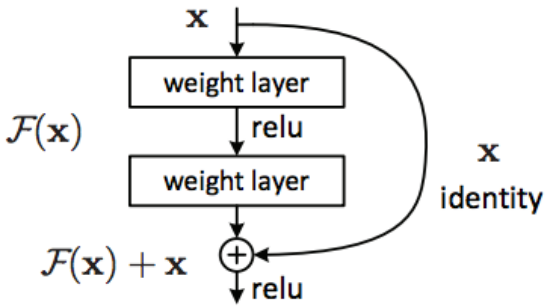
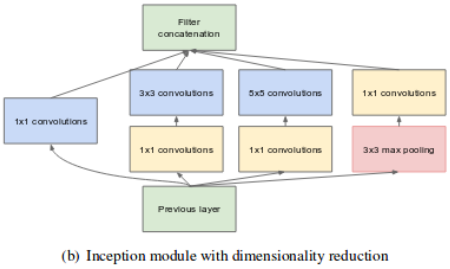


Figura 3.6: ResidualNet

En lugar de diseñar una red secuencialmente plana y difícil de entrenar, proponen la creación de capas profundas donde se entrene la diferencia entre la operación del bloque y la entrada del mismo. De este modo, el resultado hacia el siguiente bloque es una operación de suma entre la operación de la capa y la entrada de la capa anterior.

Según He et al [11], estos "atajos" (*shortcut)* posibilitan la adición controlada de nuevas capas, inicializadas a cero, sin efecto inicial en la original y garantizando la inexistencia de degradación por desvanecimiento.

Inception: Además de soluciones para hacer crecer las ConvNets con garantías a lo largo, C. Szegedy et al [13] proponen crecer también a lo ancho.



(b) Inception module with dimensionality reduction

Figura 3.7: Inception

Consideran que la defición de la forma de los filtros en cada una de las capas convolucionales no es trivial. Cada uno de ellos extrae unas características que otro no ofrece (5x5, 3x3, 1x1, *maxpool*, ...). De modo que proponen que sea la propia red la que decida, en la fase de entrenamiento, la contribución de cada uno de los filtros al problema.

Esta arquitectura es conocida con el sobrenombre de GoogLeNet en honor a su procedencia (Google) y al primer trabajo en visión artificial con ConvNets de LeCun et al [4] (3.2.1).

SSD300: Una de las principales tendencias, en las propuestas para detección, clasificación y localización de imagen, es el uso de una única red neuronal profunda por la que pasa la imagen una sola vez.

Como todo el proceso de detección se realiza en esa única pasada, Wei Liu et al [12] proponen una arquitectura modelada a partir de la VGG16 [10] (3.2.2), sin las capas *FC*, por su buen rendimiento en clasificación de imágenes de alta calidad y la facilidad que ofrece para tranferir aprendizaje debido a su popularidad.

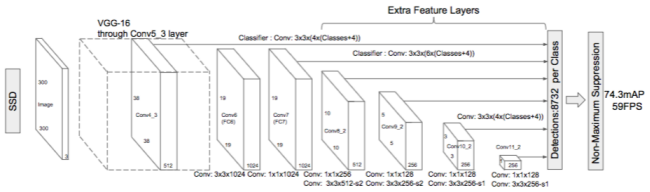


Figura 3.8: SSD300

Conforme realiza los trabajos de detección, valora el grado de confianza sobre diferentes formas y escalas de rectángulos con los que encapsular el objeto en una localización. Trabajo que se realiza en diferentes etapas de la red. Las ultimas capas se dedican a valorar las probabilidades asignadas a cada encapsulado eligiendo la de mayor valor por objeto como resultado válido.

RetinaNet: a contracorriente con la tendencia en diseño de modelos de detección, localización y clasificación de imágenes basados en el análisis monosescala, T. Lin et al [14] proponen el rediseño de una antigua técnica de análisis, consistente en trabajar con la misma imagen a diferenctes resoluciones.

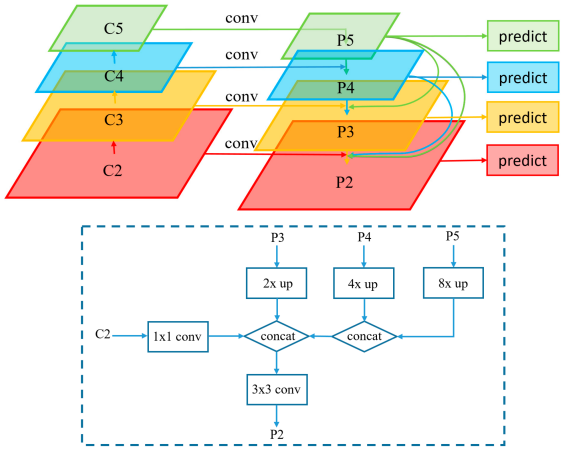


Figura 3.9: RetinaNet

Este modelo pareleliza una convolución estratificada sobre diferentes resoluciones de la imagen, realizando predicciones en cada uno de los estratos. El paso a la siguiente capa se produce previa concatenación de todos los estratos.

R-CNN: muy poco después de Overfeat [9] R. Girshick et al [16] presentan un modelo basado en selección de regiones formado por tres etapas: extracción de posibles objetos a través del método de proposición de regiones o segmentación, extracción de características de las regiones propuestas y clasificación de las regiones con una etapa SVM.

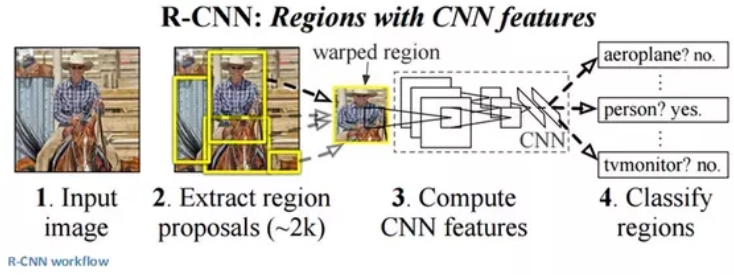


Figura 3.10: R-CNN

Pese a su incremento notable en detección, no resultaba especialmente adecuada para reconocimiento en tiempo real y dos años más tarde R. Girshick [17] propone *Fast R-CNN* una evolución del modelo sustituyendo la fase ConvNet sobre las regiones propuestas y la SVM para clasificar por la aplicación de la ConvNet sobre toda la imagen y sobre la región de interés (*RoI*) proyectada sobre la ConvNet global. A partir de este punto pasa a la zona de clasificación y regresión. Este modelo proporciona un flujo ConvNet hasta el final, facilitando su entrenamiento.

Poco tiempo después S. Ren [18] presenta una mejora del modelo añadiendo el concepto de *Region Proposal Network (RPN)*. Una de las mejoras más importantes aplicadas en muchos de los modelos posteriores. El modelo se diseña de principio a fin con una ConvNet.

3.3. YOLO - "You Only Looks Once"

Previa a la propuesta de W. Liu et al [12] con su modelo *Single Shot Detector "SSD"* (3.2.2), J.Redmon et al. [19] presentan el primer sistema de detección de objetos en tiempo real con una única ConvNet y en una sola pasada. De ahí su nombre.

Al ser el sistema de detección en tiempo real referente en la actualidad, se opta por estudiarlo para su aplicación en la detección de objetos personalizados que se pretende con este TFM.

Veamos las características y evolución de este sistema de visión en tiempo real, desde su presentación en junio de 2015 hasta la última revisión de abril de 2018.

3.3.1. YOLO: Detector unificado de objetos en tiempo real

La primera de las tres versiones que se han presentado de *YOLO* marca el concepto general del procesamiento de imagen atravesando una sola vez una única

ConvNet.

La idea detrás de este concepto nace en la convolución de la ventana deslizante que presentó P. Sermanet et al. [9] con su sistema *Overfeat*, pero con un desplazamiento (*stride*) suficiente para independizar las ventanas de sus adyacentes. Es decir, dividir el análisis de la imagen completa con una rejilla de S x S ventanas individuales. Este proceso permite al sistema analizar cada ventana en paralelo a las demás, ofreciendo el resultado final en una matrix donde cada celda frontal muestra el resultado de cada ventana. Estas celdas tienen la profundidad adecuada a los parámetros que identifican el objeto detectado. Se puede observar un pequeño ejemplo en la figura 3.11.

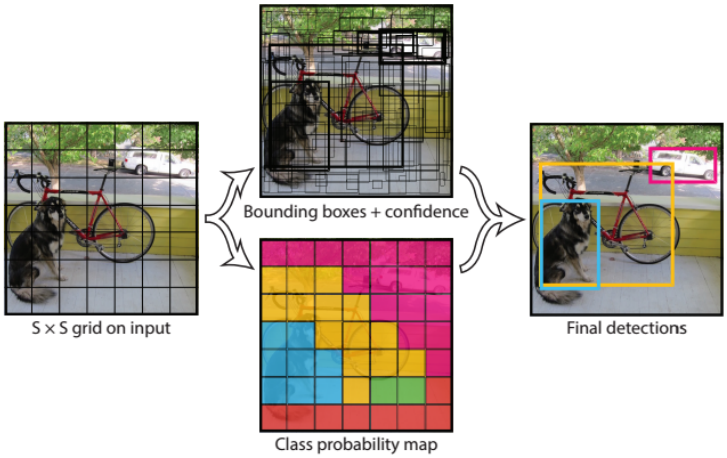


Figura 3.11: división de una imagen en una cuadrícula de ventanas (celdas) independientes. Cada celda identifica B *bounding boxes*. Cada *bounding box* identifica el grado de confianza de la detección, coordenadas y dimensiones del rectángulo y la probabilidad asociada a cada clase de objetos que es capaz de detectar. Fuente [19]

Las celdas de esta rejilla son responsables de identificar un único objeto/celda. Para ello, se ayudan de un conjunto de B contenedores rectangulares (B *bounding boxes*). Cada *bounding box* se responsabiliza de localizar y clasificar la detección de un objeto. Los datos que guarda son las coordenadas del centro, ancho y alto (4 datos) y el grado de confianza de que el *bounding box* contenga un objeto y lo adecuada que es para ese objeto (1 dato). Cada celda guarda la probabilidad condicional de que el objeto pertenezca a cada clase que es capaz de reconocer (C clases). Con ello, al final del proceso, se muestran las predicciones que han determinado en una matriz con dimensiones $(S * S * (B * (1 + 4) + C)$. Se puede ver la arquitectura de la red en la figura 3.12

Se trata, pues, de un sistema con una primera parte de la red dedicada a la extracción de características y reducción de la dimensionalidad a partir de una ConvNet. La ConvNet consta de 24 capas convolucionales. Una parte para resolver el problema de regresión lineal con una red clásica de dos capas *FC* y una zona de evaluación de resultados donde determinar las *bounding boxes* con mayor grado de confianza (*non-maximum suppression*).

La condición impuesta a cada celda de detectar un solo objeto limita la densidad

de objetos que puede detectar. Para detectar objetos diferentes, aunque pertenezcan a la misma clase, deben estar a una distancia mínima marcada por las dimensiones de la celda. Es una característica del sistema a tener en cuenta en la interpretación de los resultados.

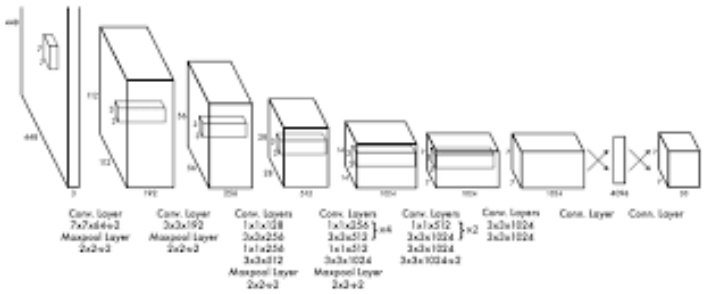


Figura 3.12: arquitectura de *YOLO* inspirada en GoogLeNet, pero utilizando únicamente la linea de reducción 1x1 seguida de la convolución 3x3 en lugar de todo el ancho por capa de la Inception [13]. Fuente [19]

3.3.1.1. Entrenamiento

La fase de entrenamiento consta de dos partes. Realiza un primer preentrenamiento con imágenes de 224x224 sobre 20 capas convolucionales, de las 24 que consta la red, y una capa *FC*. Con ello se entrena como clasificador.

Una vez terminada esta primera fase, la red se pone en modo detector añadiendo las cuatro capas restantes y el total de las capas *FC*. Se inicializan con pesos aleatorios. La resolución de las imágenes de entrada se incrementan a 448x448 para mejorar la calidad de la detección.

3.3.1.2. Estimación de probabilidades

Tal y como se ha comentado más arriba, la asignación de probabilidades se divide en dos partes. Por un lado se estima la probabilidad de detección y adecuación del contenedor rectangular para un objeto y se estiman las coordenadas y medidas del contenedor. Por otro, se calcula el grado de confianza de que el objeto detectado pertenezca a cada una de las clases.

Así, cada celda establece la estimación de la confianza para cada una de las clases según 3.1.

$$class\_confidence\_score = box\_confidence\_score * conditional\_class\_probability \tag{3.1}$$

Expresión que mide el grado de confianza tanto de la clasificación como de la localización de cada clase capaz de detectar.

Matemáticamente, cada una de la partes de la expresión 3.1 se formula tal y como se puede ver en 3.2

$$\begin{aligned} box\_confidence\_score &\equiv P_r(object) * IoU \\ conditional\_class\_probability &\equiv P_r(class_i \mid object) \\ class\_confidence\_score &\equiv P_r(class_i) * IoU \end{aligned} \tag{3.2}$$

donde:

$P_r(object)$: la probabilidad de que un contenedor albergue un objeto

IoU: *Intersection over Union* cociente entre la intesección de la superficie del contenedor rectangular estimado y la superficie del contenedor rectangular verdadero que contiene al objeto frente a la unión de las dos superficies.

$P_r(class_i \mid object$: la probabilidad a posteriori de una clase concreta dado un objeto.

$P_r(class_i)$: la probabilidad a priori asignada a la clase i.

3.3.1.3. Función de pérdida (*Loss function*)

Tal y como se ha detallado más arriba, *YOLO* maneja varios *bounding box* por celda. Para computar la función de pérdida en la fase de entrenamiento se precisa, únicamente, el valor de la mejor de las opciones de la celda. Para ello se busca la opción con la IoU más alta como responsable de la detección de un objeto en la celda. Esta estrategia facilita la especialización entre las diferentes predicciones de las *bounding boxes*. Cada una va adquiriendo mayor precisión para determinados tamaños y relaciones de aspecto.

La función elegida para calcular la pérdida es el error cuadrático entre la predicción y el valor real. *YOLO* utiliza una función de pérdida dividida en tres sumandos:

Pérdida en la clasificación: si se detecta un objeto, se computa el error cuadrático entre las probabilidades condicionales de cada clase.

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{3.3}$$

donde:

$\mathbb{1}_i^{obj} = 1$ si aparece un objeto en la celda i y es 0 en caso contrario.

$\hat{p}_i(c)$ es la probabilidad condicional de la clase c en la celda i.

Pérdida en la localización: el valor de pérdida de localización computa el error en la predicción de las coordenadas y tamaño de la *bounding box* responsable de la detección del objeto.

YOLO utiliza la raiz cuadrada de la anchura (*"w"eigth*) y altura (*"h"eight*) de los contenedores para minimizar la repercusión de la diferencia entre el error en

contenedores grandes y contenedores pequeños. Para dar mayor énfasis a esta parte de la función de pérdida, se aplica un factor de corrección λ_{coord} con valor mayor que uno (por defecto cinco).

$$\begin{aligned} &\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ +\lambda_{coord} &\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \end{aligned} \tag{3.4}$$

donde:

$\mathbb{1}_{ij}^{obj} = 1$ si la *bounding box* j de la celda i es la responsable de detectar el objeto y es 0 en caso contrario.

λ_{coord} coeficiente para incrementar el peso del error producido en las coordenadas de la *bounding box*.

Pérdida en la estimación de confianza: Uno de los grandes problemas en el entrenamiento de ConvNet dedicadas a visión artificial, radica en que no todas las celdas y por ende, no todas las *bounding boxes* detectan objetos. Es decir, gran parte del entrenamiento se dedica a evaluar el fondo de las imágenes del que, de momento, no aporta información útil. Para paliar este efecto de desbalanceo en la función de pérdida, *YOLO* computa el error en caso de que no detecte objeto y le aplica un factor de correción a la baja λ_{noobj} (por defecto 0.5).

Así, en caso de detectar un objeto, el error se computa como se refleja en 3.5

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{3.5}$$

donde:

$\mathbb{1}_{ij}^{obj} = 1$ si la *bounding box* j de la celda i es la responsable de detectar el objeto y es 0 en caso contrario.

$\hat{C}_i$ es la estimación de confianza en la *bounding box* j de la celda i

Para el caso de no detectar objeto en la celda, se computa el error según 3.6

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \tag{3.6}$$

donde:

$\mathbb{1}_{ij}^{noobj} = 1$ es el complementario de $\mathbb{1}_{ij}^{obj}$

$\hat{C}_i$ es la estimación de confianza en la *bounding box* j de la celda i

λ_{noobj} es el coeficiente para decrementar el peso del error ante una celda que no ha detectado objeto alguno.

3.3.1.4. Inferencia del mejor valor (*Non-Maximum Suppression*)

YOLO apuesta por la diversidad espacial para hacer predicciones, aumentando tanto la velocidad de procesamiento como la media de acierto. Para ello, se apoya en la división de la imagen en una rejilla de celdas con un solo objeto por celda.

Lo más habitual es que los objetos sean detectados por un solo contenedor de una sola celda. Sin embargo, cuando existen objetos muy grandes y/o el centro de coordenadas está muy cerca de los límites de varias celdas, puede haber múltiples detecciones para un solo objeto.

Para definir la estimación más probable entre todas las detecciones del mismo objeto en una zona muy cercana, se utiliza la técnica denominada *non-maximum suppression*. Esta técnica aporta un 2-3 % a la media de acierto. Consiste, básicamente, en el descarte de las detecciones con estimación más baja, manteniendo las detecciones con mayor *IoU*. Se itera sobre esta idea hasta que solo queda un contenedor para el objeto detectado en una zona concreta de la imagen.

3.3.2. YOLOv2 - YOLO9000: Mejor, Más rápido, Más robusto.

La solución basada en *SSD* (3.2.2) de W. Liu et al. [12] se presentó con mejores prestaciones que *YOLO*. Sin embargo, la segunda versión, *YOLOv2*, incorpora una serie de mejoras que lo volvieron a catalogar como el sistema de visión en tiempo real más rápido y preciso.

3.3.2.1. *Batch Normalization*

La regularización en la fase de entrenamiento se llevaba a cabo en la zona de regresión lineal, en las capas totalmente conectadas (*FC*). Para ello, se utilizaba *(*dropout) con valor 0.5. Como se detallará más adelante, la nueva arquitectura de la red sustituye esta zona *FC* por capas convolucionales. Aplicar dropout a estas nuevas capas convolucionales no es lo más adecuado por la alta correlación entre posiciones anexas, de modo que se opta por utilizar *Batch Normalization* [22]. Esta modificación eleva hasta un 2 % la media de acierto *mean Average Precision (mAP)*.

3.3.2.2. Clasificador de alta resolución

El cambio fundamental en la fase de entrenamiento como clasificador de *YOLOv2* es que utiliza imágenes de 228x228 para realizar un primer entrenamiento intensivo. Terminada esta primera parte, entrena con imágenes de 448x448 pero

muchas menos iteraciones. Esta nueva forma de entrenar, en la fase de clasificación, permite mejorar el trabajo con imágenes de mayor resolución y proporciona un incremento de *mAP* que puede llegar hasta el 4 %.

3.3.2.3. Contenedores de referencia (*Anchor boxes*)

Tal y como indica J. Redmon et al. [20], *YOLOv2* implementa la estrategia propuesta por S. Ren et al. [18] en la predicción y evaluación de los contenedores para los objetos detectados.

De este modo, pasa de predecir posiciones y dimensiones aleatorias, que no siempre funcionan bien para todos los objetos, a predecir offsets y probabilidades sobre una serie de contenedores de referencia o *anchor boxes*.

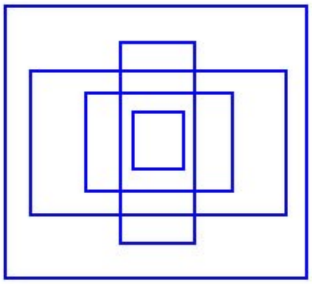


Figura 3.13: 5 *anchor boxes* fijos sobre los que se compara cada predicción. Fuente [25]

Y es que las predicciones aleatorias de contenedores no se adaptan bien a la detección de objetos. En gran parte de los dominios de trabajo, los objetos se pueden clasificar, con bastante facilidad, en una cantidad reducida de tamaños y relaciones de aspecto. Por ejemplo, los coches se pueden enmarcar, generalmente, en contenedores rectangulares apaisados con una relación de aspecto que varía muy poco entre unos y otros. Los peatones, por su parte, precisan de contenedores verticales muy altos y estrechos pero también con relaciones de aspecto muy similares entre ellos.

Así, *YOLOv2* sustituye la predicción aleatoria de *bounding boxes* por la de offsets respecto a 5 *anchor boxes* definidos con su tamaño y relación de aspecto.

Este movimiento lleva asociados unos cambios en la parte final de la arquitectura. Cada celda pasa a tener información sobre cinco *bounding boxes* correspondientes a cinco diferentes *anchor boxes*. Cada *bounding box* tiene información de la localización y tamaño, la probabilidad de haber detectado un objeto y la predicción sobre cada una de las clases que detecta el sistema.

La resolución de las imágenes se reduce de 448x448 a 416x416 para que al realizar la rejilla de celdas, se obtenga un número impar y así el centro de la imagen coincida con el centro de una celda.

La implementación de las *anchor boxes* reduce ligeramente el valor de *mAP*. Sin embargo, el factor de *recall* (la relación entre los positivos acertados y el total

de positivos) aumenta significativamente.

Definir exactamente 5 *anchor boxes* es el resultado de analizar el conjunto de entrenamiento con un proceso previo de clasificación. Para ello se aplica *k-means* con diferentes valores de k sobre la relación IoU entre los contenedores etiquetados y diferentes *anchor boxes* (tantos como k). En este punto, a partir de $k = 5$ tanto el valor de mAP como el *recall* no experimentan mejoras sustanciales.

3.3.2.4. Predicción

S. Ren et al. [18] predicen directamente el desplazamiento (*offset*) entre la detección estimada y el *anchor box* correspondiente. Esta solución añade mucha inestabilidad en la fase temprana de entrenamiento de *YOLOv2*, de modo que se opta por predecir las coordenadas y dimensiones de la *bounding box*, así como la estimación de detección, y se pasan por un filtro que controla sus valores para ajustarlos a la celda.

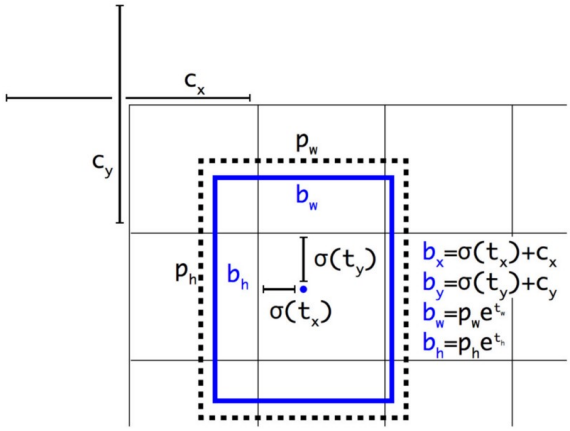


Figura 3.14: predicción de localización y estimación de detección. Fuente [20]

La formulación utilizada se puede ver en 3.7:

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ Pr(obj) * IoU(b, obj) &= \sigma(t_o) \end{aligned} \tag{3.7}$$

donde:

t_x, t_y, t_w, t_h, t_o son las predicciones reslizadas por *YOLOv2*

c_x, c_y es el desplazamiento de la celda respecto a la rejilla.

p_w, p_h son las medidas de la *anchor box* correspondientes a esa *bounding box*.

c_x, c_y, p_w, p_h se normalizan con las dimensiones de la imagen.

b_x, b_y, b_w, b_h son las dimensiones del contenedor asignadas a la *bounding box*.

$\sigma(t_o)$ es la estimación de confianza del contenedor.

La mejora experimentada por *YOLOv2* con el uso de esta variante de las *anchor boxes*, definida como *detection clusters*, se eleva hasta el 5 %.

3.3.2.5. Rango de detección y resolución de entrada ampliado

La detección de objetos relativamente grandes, respecto al tamaño de la imagen, se realiza con suficiente precisión y velocidad. Los objetos más pequeños que puedan formar parte de la imagen no siempre ofrecen la suficiente resolución como para ser detectados. Las ConvNet van reduciendo su dimensionalidad (ancho por alto) conforme avanzar en su arquitectura, de modo que la resolución disminuye con ella.

Soluciones como *Faster-RCNN* [18] o *SSD* [12] van acumulando detecciones a lo largo de su arquitectura para analizarlas al final en una serie de capas de diferente resolución. *YOLOv2* resuelve este problema desde otra aproximación basada en el paso directo de una capa temprana, de mayor resolución, hacia adelante en la ConvNet para concatenarla, tras su adaptación dimensional, con las capas finales de estimación de los datos de detección *passthrough*, tal y como se puede ver en la figura 3.15

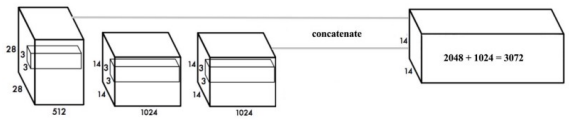


Figura 3.15: concatenación de diferentes niveles para aumentar resolución o capacidad de detección de pequeños objetos. Fuente [25]

El nuevo diseño de *YOLOv2* sustituyendo las capas *FC* por capas convolucionales y de *pooling*, facilita el trabajo con diferentes resoluciones de la imagen de entrada. Así, la fase de entrenamiento utiliza, cada 10 iteraciones, una resolución de imagen diferente logrando una detección más robusta al forzar a la red a aprender sobre diferentes tamaños de imagen.

3.3.2.6. Clasificador jerárquico

Mejorar el sistema de detección en tiempo real incluye aumentar la capacidad de detectar y clasificar una cantidad cada vez mayor de clases. Para ello, J. Redmon et al. [20] se proponen entrenar *YOLOv2* con dos de los conjuntos de datos más importantes en el dominio de los sistemas de visión por computador. Se trata de *ImageNet* [23] especialmente concebido para clasificación y *COCO* [24] especialmente concebido para detección.

En este punto aparecen algunos retos. Por un lado, se pretenden unir diferentes conjuntos de datos para detección y conjuntos de datos con diferentes sistemas de clasificación, usando diferentes etiquetas. Por otro lado, las etiquetas de un conjunto

y otro no son excluyentes. En un conjunto de datos encontramos, por ejemplo, una raza de gato, y en otro solo existe la clase "gato", perteneciendo aquella a ésta.

La solución que se plantea organiza las etiquetas de ambos conjuntos de datos en un árbol jerárquico de clases. De este modo, partiendo de la detección de un *physical object*, se desgrana en nodos hasta las hojas que definen la clase en su grado más detallado tal y como se puede ver en 3.16.

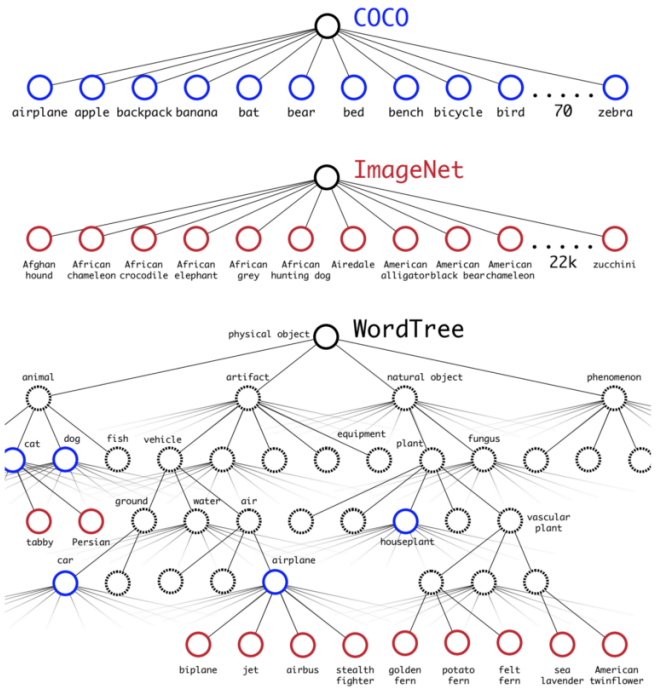


Figura 3.16: Wordtree, árbol jerárquico de clasificación. Fuente [20]

Cada uno de los nodos padre actúa como un clasificador esclusivo con sus nodos hijos. La probabilidad de ocurrencia de una clase hija, entonces, se vincula a que se dé la clase padre.

$$P_r(clase_{hijo} \mid clase_{padre}) \tag{3.8}$$

dado que tenemos 3.9

$$\begin{aligned} P_r(clase_{hijo1} \mid clase_{padre1}) + P_r(clase_{hijo2} \mid clase_{padre1}) + \\ ... + P_r(clase_{hijoN} \mid clase_{padre1}) = 1 \end{aligned} \tag{3.9}$$

Es decir, se aplica la función *softmax* para computar la probabilidad sobre cada nodo y el resto que comparten nodo padre. Con ello, *YOLOv2* combina múltiples estimaciones *softmax* desde las hojas hacia arriba para computar la probabilidad sobre cada clase-hoja, tal y como se muestra en la expresión 3.10 y en la figura 3.17.

$$
\begin{aligned}
P_r(hijo-hoja) = P_r(hijo-hoja \mid padre_{hijo-hoja}) \\
\cdot P_r(padre_{hijo-hoja} \mid padre_{padreHH}) \\
\cdot ... \\
\cdot P_r(padre_{...} \mid physical\text{-}object) \\
\cdot P_r(physical-object)
\end{aligned}
\tag{3.10}
$$

En la fase de clasificación se presupone que un objeto a sido detectado, de modo que $P_r(physical\text{-}object) = 1$.

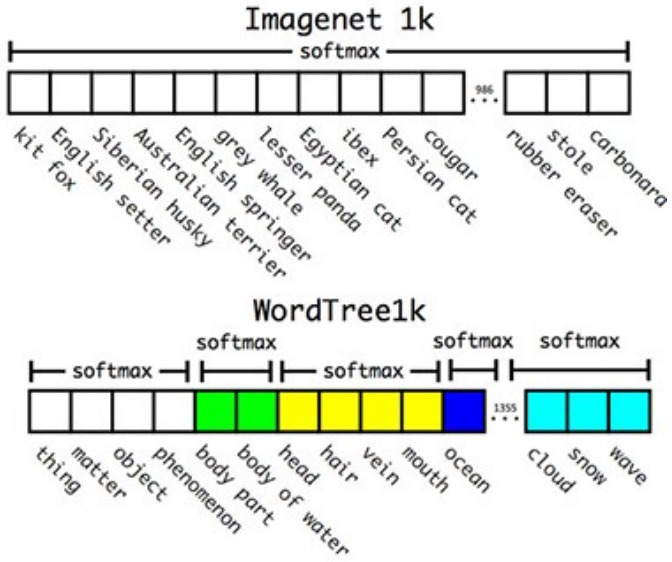


Figura 3.17: redistribución de la estimación de probabilidad sobre clases jerarquizadas frente a la aplicación de la función *softmax* sobre el total de las clases. Fuente [20]

Detectado un objeto, el proceso de clasificación recorre el árbol jerárquico desde el nodo principal *physical-object* hacia abajo. La ruta se define por el nodo hijo con la máxima estimación sobre el nodo padre que se computa con la función *softmax*. Así, el proceso culmina cuando se alcanza cierto umbral que impide seguir avanzando o se acaba en un nodo hoja.

Este proceso aporta robustez al sistema de detección de objetos en tiempo real. Por un lado, si no es capaz de llegar a un nodo hoja, clasifica por el último nodo padre con mayor estimación de probabilidad. Esta característica proporciona mayor capacidad de generalización en la extracción de características comunes entre subclases. Aspecto muy interesante ante la clasificación de objetos para los que no ha sido entrenado. Por otro lado, cada objeto detectado acaba asociado a todas las subclases con mayor estimación de probabilidad por las que ha pasado el proceso en su descenso por el árbol jerárquico.

3.3.3. YOLOv3: Una mejora incremental

J. Redmon et al. [21] presentan *YOLOv3* dentro de un proceso de mejora técnica, más que un modelo sustancialmente diferente a *YOLOv2*.

Los principales cambios que contempla el renovado sistema de detección en tiempo real *YOLOv3*, situándolo de nuevo como referencia del estado de la investigación en visión por computador, se detallan en los siguientes subapartados.

3.3.3.1. Predicción de *bounding boxes* y contribución a la función de pérdida

La estimación de probabilidad en la detección de un objeto pasa a computarse por regresión logística. Esto implica que se asigna 1 a la *bounding box* que mejor se superponga a su modelo representativo *anchor box*. El resto de *bounding boxes*, si están dentro de un umbral, no computarán en la función de pérdida en la fase de entrenamiento.

Cada objeto etiquetado se asocia con una sola *bounding box* de referencia (*anchor box*). De este modo, las *anchor box* no asignadas computan en la función de pérdida solo con su estimación de detección, sin tener en cuenta sus estimaciones de localización ni clasificación.

La expresión de cálculo es la misma que utiliza *YOLOv2* detallada en 3.7. Para el cómputo en la función de pérdida se utilizan los parámetros estimados t_x y t_y.

3.3.3.2. Clasificación

Tal y como se resolvía en la versión anterior, cada contenedor puede clasificar con múltiples etiquetas, según el modelo jerárquico visto en la figura 3.16. La diferencia radica en que se sustituye la función *softmax* por una función de regresión logística sobre cada clase de manera independiente. Para llegar a ello, el proceso de entrenamiento utiliza *binary cross-entropy* como función de pérdida para la estimación de clase.

Este modelo de clasificación disminuye la carga de cómputo y prepara el sistema de detección para conjuntos de datos más complejos, con objetos pertenecientes a múltiles clases no excluyentes. Por ejemplo, mujer y persona.

3.3.3.3. Predicción multiescala

En la figura 3.18 se puede observar que existen tres puntos de detección de objetos dentro de la nueva arquitectura. Con ellos se pretende detectar objetos a diferentes escalas.

La primera detección en la capa 82 es para los objetos más grandes, mientras que la última en la capa 106 se responsabiliza de detectar objetos más pequeños. Para ello realiza procesos de aumento de resolución y concatenación de capas previas para obtener mayor información de la imagen a una resolución más alta.

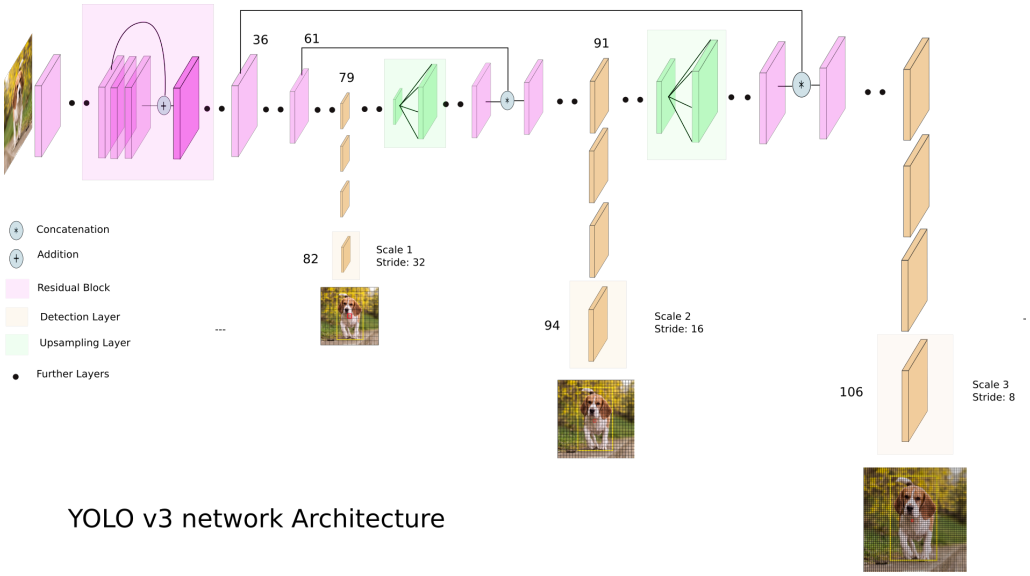


Figura 3.18: Arquitectura de *YOLOv3*. Fuente `https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b`

La organización de la matriz de resultados varía respecto a la versión anterior. Así, se utilizan 9 *bounding boxes*. 3 por celda para cada una de las escalas de detección. Estas 3 *bounding boxes* por celda responden a tres *anchor boxes* utilizadas como referencia para la estimación.

3.3.3.4. Extracción de características

YOLOv3 utiliza una nueva red neuronal de 106 capas. Utiliza una primera parte de extracción de característica formada por 53 capas entrenadas con ImageNet [23]. La segunda parte, formada por las restantes 53 capas, se destinan a la detección de objetos con las tres diferentes escalas detalladas en el apartado anterior.

CAPÍTULO 4

PLANIFICACIÓN Y ESPECIFICACIÓN

Conocidos los objetivos del proyecto, como paso previo a su desarrollo, se debe realizar un análisis de aquello que se espera del mismo. Para ello, se procede a detallar al máximo las características del sistema. Estos requisitos se verán en el apartado 4.1. Con el detalle de los requisitos vistos, se estará en condiciones de establecer la especificación del sistema en el apartado 4.2. Validados los apartados anteriores, se estará en condiciones de pasar al desarrollo de los trabajos necesarios para su consecución.

4.1. Análisis de requisitos

Tras diferentes reuniones con el Dr. Roberto Paredes, director del presente TFM, donde se analiza el alcance del sistema a entrenar, se definieron una serie de requisitos mínimos. Funcionalidades que lo hicieran suficientemente interesante y útil para el trabajo que se espera del mismo.

4.1.1. Requisitos funcionales

Qué se espera.

- F1..La localización con las coordenadas de posición, el ancho y el alto del contenedor de cada objeto para su etiquetado debe ser automática.
- F2..Las imágenes podrán tomarse con varias resoluciones.
- F3..Se almacenarán dos versiones de cada toma. Una en color y otra en blanco y negro.
- F4..Alternativamente al sistema de etiquetado automático, existirá una opción de etiquetado manual.
- F5..El etiquetado manual permitirá trabajar imágenes con más de un objeto.

- F6..El sistema de localización automática debe teleoperarse.
- F7..Se debe tener la posibilidad de ajustar el sistema de localización automática.
- F8..El conjunto de datos extraido con los sistemas de localización automática o etiquetado manual, se debe poder extender para aumentar su cantidad.
- F9..El aumento de los conjuntos de datos contemplará, al menos, escalado, traslación y rotación.
- F10..El sistema entrenado informará de la localización del objeto y la clase a la que pertenece.

4.1.2. Requisitos no funcionales

Cómo debe ser.

- NF1..Será posible su instalación en ordenadores con los sistemas operativos más comunes.
- NF2..La fase de entrenamiento se realizará en el servidor más potente disponible.
- NF3..Al tratarse de un sistema con eficacia por debajo del 100 %, se precisa la supervisión de un operario debidamente instruido.
- NF4..La operatividad del sistema debe ser sencilla e intuitiva.
- NF5..El entrenamiento debe poder realizarlo personal con titulación técnica media.

4.2. Especificación del sistema

Definidos y validados los requisitos del sistema, se procede a continuar con el análisis del problema planteado, aunando tanto lo reflejado en el punto anterior como la información facilitada respecto a los procedimientos empleados en la actualidad. Con ello, se podrá proponer una solución al problema planteado.

4.2.1. Información del dominio

La relación e interacción de los robots con los humanos en los mismos entornos se va transformando conforme la tecnología y la investigación en sistemas inteligentes avanza.

El entorno profesional e industrial avanza inexorable hacia una mayor intervención robótica y un enriquecimiento profesional del operario con trabajos de supervisión y control, menos peligrosos, monótonos y repetitivos. El entorno doméstico incorpora pequeños elementos robotizados de ayuda a tareas poco gratificantes con resultado cada vez mejores.

Disponer de sensores cada vez más potentes y económicos, así como de sistemas de detección, clasificación, localización, regresión, etc. rápidos y eficientes, facilita la interacción de máquinas y humanos en cualquier tipo de entorno, rebajando la distancia de seguridad hasta el trabajo cooperativo debidamente organizado.

En este dominio, los sistemas de visión por computador en tiempo real juegan un papel primordial.

4.2.2. Descripción de la situación actual

Las empresas dedicadas al desarrollo de sistemas robotizados, tanto industriales como de servicios, utilizan componentes y desarrollos convenientemente probados para su uso industrial o comercial. Debido a ello, el estado del arte en sistemas de visión por computador tarda en llegar al usuario.

Mantener la ventaja empresarial que aporta el desarrollo de sistemas exclusivos y cerrados, dificulta la introducción de los resultados de la investigación institucional en soluciones abiertas al mercado de consumo tradicional.

Por otro lado, la implementación y uso del estado del arte en sistemas de visión por computador, es más factible encontrarla en la industria de servicios. Empresas dedicadas a ofrecer análisis de imagen en publicidad, seguimiento de marca en eventos, facilitadores de localización para realidad aumentada y tantas otras aplicaciones ajenas al entorno robótico, trabajan con mayor cercanía a lo que la investigación ofrece, reduciendo significativamente el *gap* entre investigación institucional y empresa.

4.2.2.1. Fortalezas de la situación actual

Salvando los sistemas de visión utilizados en el desarrollo de vehículos autónomos, todos ellos con tecnología en plena expansión, existen robots con cámaras de visión complementando su diseño.

Los sistemas dedicados a entornos industriales se diseñan específicamente para una tarea dada, cuya complejidad está perfectamente delimitada, es conocida y resoluble. De este modo, se garantiza la robustez exigible a este tipo de soluciones. Si bien, los modelos que utiliza suelen estar adaptados a sistemas empotrados o específicos del robot.

En cuanto a la robótica de servicios, encontramos que, a nivel profesional, existen sistemas de visión sobre robots teleoperados para labores de inspección, vigilancia, salvamento, atención comercial, etc. pero carentes de sistemas de detección automática especialmente complejos. Por otro lado, la robótica de entretenimiento incorpora sistemas de reconocimiento facial sencillo y acorde a las limitadas prestaciones de computación de un juguete.

4.2.2.2. Debilidades de la situación actual

Los sistemas de visión por computador para aplicación en robótica adolecen de la capacidad que se puede encontrar en cualquiera de los sistemas descritos en 3.2.

Para la interacción en tiempo real con el entorno de un robot de cualquier tipo, manteniendo las garantías de seguridad adecuadas para cualquier usuario o persona dentro del campo de acción del robot, se precisan sistemas de detección rápidos y fiables. Para ello, se precisan soportes *hardware* adecuados a esta labor de computación intesiva. Aspecto éste, difícil de encontrar en la robótica actual por su alto coste.

4.2.3. Catálogo de especificaciones

Analizando el conjunto de conocimientos adquiridos hasta el momento, tanto en requisitos como en la descripción de la situación actual, se propone un sistema que sea facilmente entrenable, robusto y fiable.

4.2.3.1. Especificaciones generales

El sistema de visión artificial a entrenar se basará en el estado de la investigación de sistemas de detección, localización y clasificación en tiempo real, como es YOLOv3.

Se diseñarán una serie de herramientas para que se pueda generar el conjunto de entrenamiento y test de modo semiautomático independiente de la plataforma software utilizada. Además, se crearán herramientas para aumentar artificialmente el número de elementos del conjunto de entrenamiento así como la modificación del fondo.

Con objeto de probar su eficiencia, se creará una herramienta para etiquetado manual desde una fuente videograbada.

4.2.3.2. Restricciones técnicas

Analizando las especificaciones del sistema de detección en tiempo real elegido, podemos esperar una respuesta muy pobre con ordenadores sin soporte de procesamiento GPU con CUDA. Así, testearemos el sistema, una vez entrenado, con un ordenador sin capadidad de procesamiento GPU, uno con cierta capacidad de procesamiento GPU y otro con una tarjeta nVidia GTX-1080 buscando el mejor equilibrio coste-funcional en el entorno con brazo robot.

Al tratarse de una primera aproximación al desarrollo de herramientas para entrenamiento de un sistema de detección en tiempo real de un conjunto de objetos personalizados, se trabajará con algunos objetos comunes que puedan formar parte fácilmente de un conjunto de datos como ImageNet [23] y otros objetos específicos obtenidos de un proceso industrial, totalmente manual, que se pretende investigar como trabajo futuro.

El sistema de visión por computador seleccionado para realizar el TFM dispone de dos versiones entrenables. Por un lado la versión completa de 106 capas y por otro lado una versión, *Tiny YOLO*, con muchas menos capas (trece convolucionales), ideal para especificaciones más limitadas de equipos, tales como algunos *SOCs* populares ("raspberry", "intel galileo", "udoo", etc.). La elección del sistema completo radica en la idea de obtener resultados de un sistema complejo para tareas de detección complejas, como puede ser una caja llena de la misma pieza en la que detectar para coger con un brazo robot.

CAPÍTULO 5

METODOLOGÍA DE DESARROLLO DEL PROYECTO

Una vez considerada la viabilidad del sistema, se debe poner en marcha el proceso de desarrollo del mismo. Para ello, en virtud de las buenas prácticas clásicas en el modelado de software con el que razonar sobre la implementación del sistema, se debería dividir el trabajo siguiente en tres pasos fundamentales. El primero de ellos, el análisis, analizaría los requisitos establecidos en el apartado 4.1. Con ello, se dispondría de lo que se espera del sistema. A partir de estos datos, estaríamos en condiciones de establecer cómo se alcanza el objetivo con el proceso de diseño. La implementación o codificación del sistema, a partir del diseño definido, representaría una mera traducción del trabajo de modelado anterior.

Pero nos encontramos en una tesitura importante. El escenario de predictibilidad en el que se está trabajando hasta ahora, no parece ser la mejor práctica de ingeniería para el sistema que se viene desarrollando. No acoplan bien. Esto es así al estar realizando el trabajo de construcción conforme se va avanzando en el diseño del sistema. No se espera la conclusión de la fase de diseño. Es más, se tira del diseño en lugar de esperar a su finalización. El diseño, pues, no cumple con la función de empuje a la construcción, típica de la metodología clásica.

Mantener la metodología de gestión de proyectos según el desarrollo tradicional, resta capacidad de adaptación al presente TFM. Este trabajo, por su destacada vertiente investigadora, se halla sujeto a continuas modificaciones en diseño y ajuste de funcionalidad.

Ante esta situación, se decide adaptar un formato mixto. Así, a partir del trabajo de planificación realizado hasta el momento, guiados por la metodología tradicional de gestión de proyectos, se dispone de suficiente documentación para afrontar la implementación y pruebas modulares mediante un proceso iterativo e incremental. Estas características constituyen la base de la metodología ágil.

Los principios que caracterizan un proceso ágil, derivados del "Manifiesto Ágil", describen la actividad y necesidades de este trabajo con suficiente cercanía. Sirvan como ejemplo algunos de los doce principios del manifiesto:

- Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo.

Los procesos ágiles se doblegan al cambio como ventaja competitiva.

- Entregar frecuentemente software que funcione, con el menor intervalo de tiempo posible entre entregas.
- La atención continua a la calidad técnica y al buen diseño.
- La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.

Se adapta, pues, el framework que ofrece SCRUM a la sección de implementación. Para ello, se parte del ajuste de las funcionalidades y documentación desarrollada en la planificación. Con ello, se compondrán las Historias de Usuario en la sección 5.1. A partir de la evaluación de estas historias se creará el Product Backlog o lista de historias de usuario, con el que mostrar la evolución continua a la que se ve sujeto el TFM durante todo el proyecto. Las historias de usuario, aprobadas por su aporte de valor al proyecto y pasadas al Product Backlog, se irán incorporando al software a partir de incrementos sucesivos. En la sección 5.2 se observa su construcción y la confección de Sprints Backlog, o metas intermedias con resultados funcionales, a partir del Product Backlog.

Periodicamente, se evalúan los avances analizando el trabajo realizado, el pendiente y los problemas aparecidos aplicando, igual que para el resto del trabajo, los principios de la estrategia "lean".

- Construir solo lo necesario.
- Eliminar todo aquello que no añade valor.
- Parar si algo no va bien.

Kanban es una técnica que proporciona herramientas útiles para controlar el avance del trabajo. Las tres reglas en que se sustenta este método son:

- Visualizar los trabajos.
- Limitar el trabajo en proceso (WIP)
- Medir el flujo de tareas

Para ello, utiliza la división de las Historias de Usuario para conformar una pizarra, virtual o real, donde visualizar la evolución del trabajo de cada uno de los Sprints Backlog.

Además, se incorpora a esta sección, bajo el mismo principio ágil, el desarrollo del prototipo con el que ayudarse a realizar la batería de pruebas correspondiente a los test modulares y de integración.

5.1. Historias de usuario

Con las historias de usuario tendremos un conjunto de descripciones breves de funcionalidades software derivadas de la sección 4.1.

Estarán compuestas por un identificador de la historia de usuario, un título descriptivo, una descripción sintetizada incluyendo el quién-qué-porqué de la misma, una estimación del tiempo de implementación, un valor que determine su importancia dentro del proyecto, las dependencias con otras historias de usuario y una serie de pruebas que debe vencer para dar por válida y finalizada la implementación.

Así, las historias de usuario en las que se basa la implementación de este TFM, proporcionarán módulos funcionales que, en sucesivas iteraciones incrementales de integración entre ellos, darán respuesta a las funcionalizadades establecidas en 4.1.

El conjunto de Historias de usuario se presentan a continuación:

Tabla 5.1: Control de cabeza robotizada

id:1	Control de posición de los servomotores
descripción	**quien:** desarrollo **qué:** la posición de los servomotores que componen la cabeza robotizada donde se ubican las cámaras se controlará desde el interfaz de usuario. **porqué:** se debe tener la capacidad remota de ajustar el campo de visión de las cámaras montadas en la cabeza robotizada.
estimación	días: 3 - personas: 1
valor	20
dependencias	ninguna
pruebas de aceptación	..el control tiene la precisión suficiente para centrar el campo de visión deseado. ..se puede controlar cada uno de los motores en los dos sentidos. ..la intensidad de giro no variará la velocidad lineal del robot.

Tabla 5.2: Vibración de cámara localizadora

id:2	Control de vibración del soporte para cámara de localización
descripción	**quien:** desarrollo **qué:** el soporte de la cámara de localización de objetos por cambio de intensidad de los pixels vibra en el eje vertical y horizontal. **porqué:** la localización de elementos estáticos por cambio de intensiad de pixels necesita que se la cámara. Que vibre.
estimación	días: 1 - personas: 1
valor	20
dependencias	1
pruebas de aceptación	..el movimiento de vibración es suficiente para conseguir una imagen con suficientes pixels para definir su contorno.

Tabla 5.3: Adquisición de datos por cambio de intensidad de los pixels

id:3	Adquisición de imagen por cambio de intensidad de los pixels
descripción	**quien:** desarrollo **qué:** capturar imagen por cambio de intensidad de los pixels para cálculo de coordenadas del contenedor rectangular. **porqué:** el objeto detectado proporciona información automática de localización y dimensión para su etiquetado.
estimación	días: 5 - personas: 1
valor	10
dependencias	2
pruebas de aceptación	..la frecuencia de muestreo es, al menos, de 24 fps. ..la detección del objeto por cambio de intensidad es clara. ..la localización y dimensiones del contenedor rectangular del objeto detectado es precisa. ..el umbral de cambio de intensidad para detección se puede regular.

Tabla 5.4: Captura automática de imagen y etiquetado

id:4	Captura automática de imagen y etiquetado
descripción	**quien:** desarrollo **qué:** una vez detectada la imagen por cambio de intensidad de pixels, se debe calcular la localización y tamaño del contenedor rectangular del objeto y guardar en un fichero de texto con la información de clase, localización y dimensión en una sola línea. El resultado de localización y dimensión debe ser relativo al tamaño de la imagen. La imagen de pixels, la imagen en b/n de la cámara de detección y la imagen de la cámara color se guardarán con nombres de raiz común y sufijo referente a la procedencia de la imagen. **porqué:** el formato de etiqueta lo define el sistena de detección en tiempo real *YOLOv3*. Disponer de diferentes representaciones de cada objeto se estima buena idea para enriquecer el entrenamiento del sistema.
estimación	días: 8 - personas: 1
valor	10
dependencias	1,2 y 3
pruebas de aceptación	..las etiquetas se guardan en formato .txt con una sola línea correspondiente a la clase y las coordenadas de posición y dimensión relativas al tamaño de la imagen. ..las imágenes se capturan y guardan correctamente con el formato JPG. ..se guardan los datos recibidos en ficheros separados. ..la clase del objeto se puede seleccionar con facilidad.

Tabla 5.5: Interfaz de control para etiquetado semiautomático

id:5	Desarrollo de interfaz de control para etiquetado semiautomático
descripción	**quien:** desarrollo **qué:** desarrollar la interfaz de teleoperación con la cabeza robotizada para enviar órdenes de movimiento, recibir datos de las cámaras, regular el umbral de intensidad y seleccionar la clase de objeto en proceso de detección. **porqué:** el operador del sistema debe disponer de un interfaz con el que facilitar la interacción con las funcionalidades esperadas bajo el principio de usabilidad. Se debe procurar una experiencia de usuario, a partir de un conjunto de factores y elementos de interacción del usuario con el dispositivo, que genere una percepción positiva y enriquecedora del sistema.
estimación	días: 10 - personas: 1
valor	15
dependencias	1, 2, 3 y 4
pruebas de aceptación	..se deben reflejar todas y cada una de las imágenes recibidads de las cámaras. ..debe controlar la cabeza robotizada con facilidad. ..debe brindar la posibilidad de cambiar la resolución de la imagen. ..la operatividad para seleccionar la clase, el umbral de cambio de intensidad de pixels, el control de motores, la selección de resolución de imagen y ejecutar el proceso de captura, será intuitiva e inmediata.

Tabla 5.6: Interfaz de etiquetado manual

id:6	Interfaz para el etiquetado manual desde fuente de vídeo
descripción	**quien:** desarrollo **qué:** El operador capturará imágenes relevantes de los objetos a clasificar desde una fuente de vídeo. Determinará las dimensiones del contenedor rectangular que defina al objeto con el ratón. **porqué:** por un lado,en aquellas imágenes donde se ubiquen más de un objeto, el sistema semiautomático no funciona. Por otro lado, se puede comparar o enriquecer los dos sistemas de etiquetado.
estimación	días: 5 - personas: 1
valor	20
dependencias	4 y 5
pruebas de aceptación	..desde el interfaz se controlará el avance y pausa del vídeo del que extraer imágenes para etiquetado.

continúa en la página siguiente

Tabla 5.6: (continuación)

	..con el ratón sobre la imagen se podrá marcar el tamaño del contenedor rectangular y su localización. ..la captura de la imagen genera un fichero de etiquetado y uno de imagen con los formatos definidos anteriormente.

Tabla 5.7: Entrenamiento y test

id:7	Proceso de entrenamiento del sistema de detección en tiempo real *YOLOv3*.
descripción	**quien:** desarrollo **qué:** crear y dividir los conjuntos de etiquetado e imagen generados. Entrenar *YOLOv3*.. **porqué:** el proceso de entrenamiento debe realizarse conforme con las directrices marcadas por los autores [21].
estimación	días: 20 - personas: 1
valor	10
dependencias	5 y 6
pruebas de aceptación	..el proceso de entrenamiento guarda pesos cada, al menos, 500 iteraciones. ..el proceso de entrenamiento ha realizado, al menos, 40000 iteraciones.

5.2. Backlog

Una vez se han definido las historias de usuario con los tiempos estimados para completarlas y el valor percibido (a menor valor → mayor importancia), se puede confeccionar el *Product Backlog* global del proyecto. Se dispone, pues, de una pila de producto, implementada a partir del listado de requisitos visto en 4.1, con el orden de prioridad basado en el valor que aporta al proyecto la implementación de cada historia.

Esta visión del proyecto aporta cualidades importantes a su desarrollo. Al estar detallado adecuadamente, con una prioridad conocida y los tiempos estimados, se consigue un escenario emergente o preparado para la incorporación de nuevos requisitos, su modificación e incluso su eliminación, sin que estos cambios representen problemas graves para el objetivo perseguido en tiempo y coste. Son cambios que se estudian al finalizar cada iteración y se resuelven en ciclos muy cortos de trabajo, pues afectan a aspectos muy concretos del proyecto.

La tabla 5.8 representa el product backlog del proyecto.

Tabla 5.8: Product Backlog

id	**requisito**	**valor percibido**	**duración**
1	Control de cabeza robotizada	20	3 días
2	Vibración de cámara de detección	20	1 días
3	Detección de imágen por cambio de intensidad de pixels	10	5 días
4	Captura automática de imagen y etiquetado	10	8 días
5	Interfaz de control	15	10 días
6	Interfaz de etiquetado manual	20	5 días
7	Entrenamiento	10	20 días

Resulta beneficiosa la aplicación de la metodología ágil, en tanto en cuanto se trabaja con objetivos a corto plazo que precisan una intensidad de trabajo diario muy estable.

Para obtener mayor control sobre los tiempos de desarrollo, se agruparán las historias de usuario de modo que se compongan tareas de duración similar. De este modo, el desarrollador trabaja sobre tramos temporales similares en cada tarea, ayudando a crear una dinámica de entrega de funcionalidades regular.

Así, se procede a establecer los diferentes sprints con los que componer cada uno de los *Sprints Backlog* en la pizarra Kanban. Estos sprints constituirán los objetivos de funcionalidad iterativa/incremental que se persiguen con la metodología SCRUM.

La división de los bloques en sprint backlog se puede observar en la tabla 5.9

Tabla 5.9: listado de Sprints Backlogs

sprint	**tareas**
1	1, 2
2	3
3	4
4	5
5	6, 7

Finalmente, se observa que la división se centra entre una y dos semanas por sprint, lo que garantiza un control adecuado de la actividad de desarrollo a la vez que la entrega regular de funcionalidad de producto.

5.3. Pizarra Kanban

Se ha optado por el uso de una pizarra sencilla para cada uno de los sprints.

Esta pizarra consta de tres columnas. La primera columna refleja las tareas

pendientes de comenzar; la columna central alberga el trabajo en progreso y la última columna recoge las tareas finalizadas.

La columna "en progreso" (WIP), está limitada a una sola tarea cada vez. Esta limitación viene dada por dos motivos fundamentales: por un lado, solo se dispone de un recurso para la implementación de cada una de las tareas; por otro lado, se estima contraproducente que el único recurso mantenga más de una tarea en proceso de desarrollo.

La estimación de tiempo transcurrido y restante del sprint en curso, a partir de la pizarra utilizada, es inmediato. Así, se pueden detectar posibles problemas con los plazos y acometer las soluciones más adecuadas tan pronto como aparecen.

CAPÍTULO 6

CONVNET ***YOLOV3*** - GENERANDO EL DATA-SET

Conforme se han ido sucediendo los sprints, las diferentes pruebas modulares de conformidad y las iteraciones incrementales en la funcionalidad del sistema, la referencia a la documentación generada en la fase de planificación ha sido continua. El diseño ha resultado enriquecido al disponer de herramientas adecuadas para su revisión y mejora.

Tras ello, las partes de implementación que, por su importancia o esfuerzo, merecen una atención especial, corresponden a la elección de los objetos a detectar, la detección de imagen por variación de intensidad de pixels, la cabeza robotizada, la interfaz de control, la interfaz de etiquetado manual y el entrenamiento.

6.1. Conjunto Base de objetos personalizado

Los objetos elegidos para el proceso de entrenamiento y detección se dividen en dos grupos diferenciados. Por un lado, se encuentran una serie de objetos comunes fácilmente identificables y con alta probabilidad de encontrarse en el conjunto de datos con el que se ha preentrenado el sistema de detección en tiempo real *YOLOv3*. Por otro lado, presentamos objetos personalizados de una fábrica de mecanizado. Se pueden ver en la figura 6.1

Dentro de los objetos exclusivos de la fábrica de mecanizado, existen dos grupos de objetos simétricos entre ellos. Con esta particularidad se pretende comprobar el nivel de profundidad en la extracción de características del sistema.

La identificacion de los diferentes objetos dentro del sistema se refleja en la tabla 7.1.

Figura 6.1: objetos que conforman el conjunto de datos para entrenamiento

Tabla 6.1: identificador de objetos

id	**objeto**
0	cucaracha
1	jarra
2	peonza
3	coche
4	cubito
5	robot
6	pelota
7	diapasón
8	candado
9	llave
10	grapadora
11	jugador
12	C83FA
13	48-17
14	17W45

continúa en la página siguiente

Tabla 6.1: (continuación)

15	17W35
16	18-17I
17	18-17D
18	37-17
19	40-17

6.2. Generación del conjunto de objetos y etiquetado por cambio de intensidad en los pixels

La generación de un conjunto de objetos personalizado implica una tediosa y árdua labor de captura fotográfica de cada uno de los objetos desde diferentes perspectivas y el posterior etiquetado de cada una de las imágenes.

Para reducir la carga de trabajo inicial, se proyecta un sistema de detección de imagen semiautomático. Este sistema constaría de una cámara a color para capturar la imagen, un brazo robotizado para cambiar la perspectiva de los objetos a etiquetar según un patrón preestablecido y una cámara ultrarrápida de detección de variación en la intensidad de los pixels desarrollada por F. Pardo et al. [26].

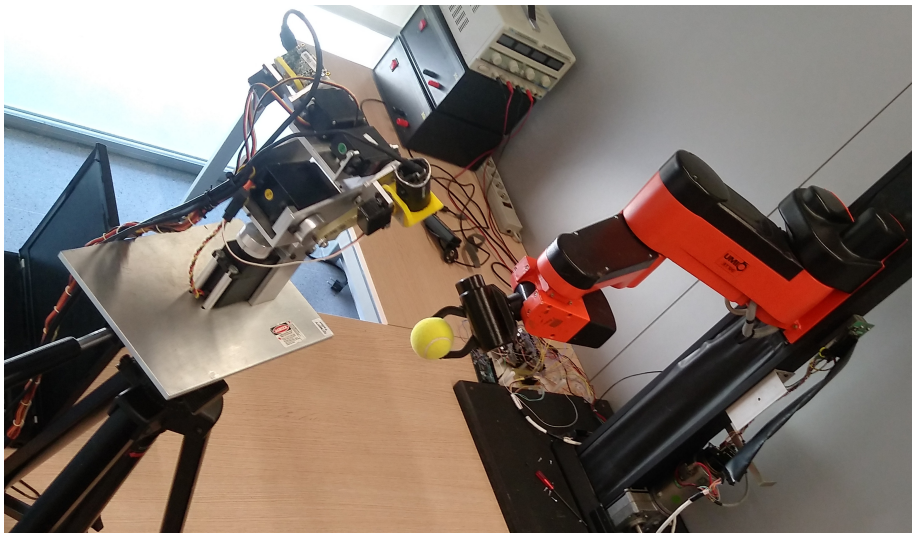


Figura 6.2: sistema inicial de captura y etiquetado semiautomático de objetos

La cámara de F. Pardo et al. [26], en modo *SCD*, envía un vector con las coordenadas de los pixels que han detectado cambios de iluminación por ciclo de reloj. Esta característica lo convierte en un rapidísimo detector de movimiento. Para que la cámara detecte un objeto, éste debe estar en movimiento. El brazo robot sería el encargado de hacer oscilar el objeto en cada una de las perspectivas que, el mismo brazo robot, lo situase. Conocidas las coordenadas del objeto y las dimensiones del contenedor rectangular que lo encuadra de manera automática y seleccionando manualmente la clase de objeto que se está detectando, una segunda cámara se

encargaría de capturar la imagen, mientras el sistema guarda tanto la imagen como la etiqueta correspondiente.

Desafortunadamente, se han encontrado dos problemas. La resolución del prototipo disponible es de 128x128 pixels (desestimado por baja resolución) con una estimación de recepción de una cámara con doble resolución que no se podía asumir. Además, el brazo robot se encontraba en proceso de modernización y, pese a los esfuerzos dedicados, no ha podido estar en disposición de formar parte del sistema inicialmente proyectado.

Pese a ello, no se abandona el concepto y se desarrolla la idea desde otro punto de vista. Sobre una cabeza robotizada, se montan dos cámaras con especificaciones y propósitos diferenciados. La cámara elegida para la detección por cambio de intensidad de los pixels es una FireFly USB 2.0 FMVU-03MTM-CS de Flir Systems. Se trata de una cámara monocromática de 0.3 M de resolución (752x480) y 60 fps. Configurada a una resolucíon de 256x256 pixels, el tráfico de datos y el tiempo necesario para simular el comportamiento de la cámara inicial, proporciona información a suficiente velocidad como para trabajar en tiempo real.

Figura 6.3: distribución de cámaras para detección y captura

La simulación del modo *SDC* se divide en dos procesos:

- ante la falta de brazo robotizado que mueva y haga oscilar el objeto a detectar, se opta por que sea la cámara la que vibre. Para ello, se programan adecuadamente los servos de su soporte.

- al tratarse de una cámara monocromática con resolución fijada en 255x255 pixels, el tamaño de la imagen es suficientemente reducido para un tratamiento eficiente entre capturas con el que obtener un *framerate* no inferior a 24 fps. A partir de la captura actual y la anterior, convenientemente guardada, se comparan pixel a pixel para conocer la diferencia de intensidad entre ellos. Se forma una imagen binaria (blanco/negro), marcando como blanco todos los pixels cuya diferencia supere un umbral determinado.

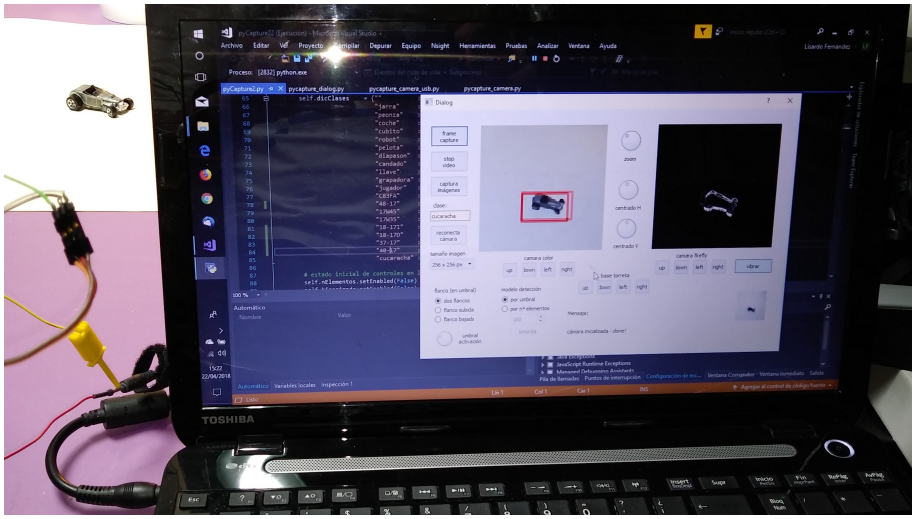


Figura 6.4: detección de objeto por contorno y cálculo de contenedor rectangular

De la imagen binarizada se buscan las posiciones de los pixels blancos ubicados en los extremos superior-izquierdo e inferior-derecho. Con estas posiciones se determina el centro del contenedor rectangular que alberga el objeto detectado y sus dimensiones.

Una vez se dispone de la imagen binaria que nos permite etiquetar la imagen, con el centro de coordenadas del objeto y las dimensiones del contenedor rectangular, se guardan tres imágenes y la misma etiqueta personalizada con el nombre de cada una de las imágenes. Las tres imágenes a guardar son:

- imagen de cámara color
- imagen de cámara b/n en escala de grises
- imagen de cámara b/n binaria resultado del procesamiento de detección de variación en la intensidad de los pixels

Con todas las imágenes y etiquetas capturadas con este proceso semiautomático se conforma el cojunto base de datos de entrenamiento.

6.3. Cabeza robotizada para etiquetado del Conjunto Base

Para albergar, orientar y controlar el movimiento de las dos cámaras responsables de la detección y la captura de imágenes para el entrenamiento, se desarrolla un sistema formado por un microcontrolador, una cabeza robotizada como se puede ver en la figura 6.5 y los programas de control adecuados para el microcontrolador y su control desde el PC.

Un microcontrolador Arduino DUE, programado en lenguaje "C" desde el IDE que suministra su fabricante, se encarga de recibir y trasladar las órdenes de movimiento de motores. Desde el PC, un sistema de control basado en "Python 3.5" y

con la librería de comunicación serie "pySerial" se comunica con el microcontrolador por el puerto USB.

A través de unos pulsadores en el interfaz de control se podrá mover cada motor de manera independiente en cualquiera de los dos sentidos.

La vibración necesaria para detectar el objeto a detectar y localizar, se realizará desde el microcontrolador una vez recibe la orden de marcha o paro. Los motores responsables de mover de forma independiente la cámara de detección realizan un movimiento cíclico de vaivén alrededor de la posición actual. La distancia de vaivén se determinará a partir de pruebas visuales sobre el mejor resultado en la detección.

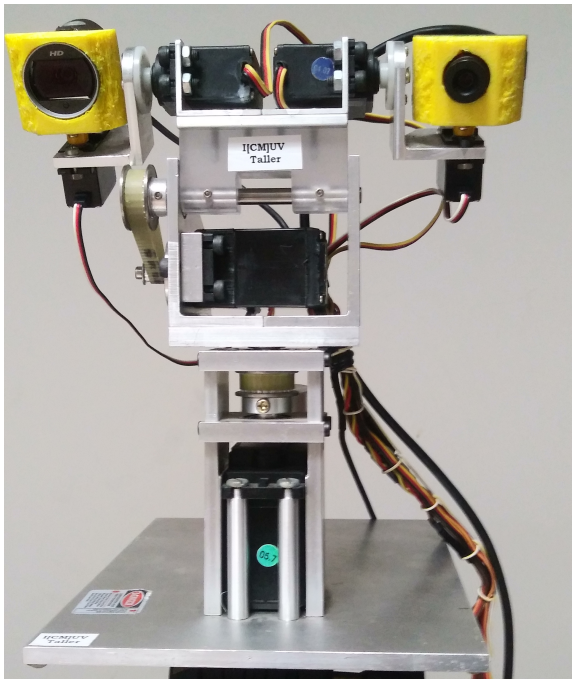


Figura 6.5: cabeza robotizada con 6 grados de libertad

6.4. Interfaz para etiquetado semiautomático

El desarrollo de herramientas software debe estar ligado, necesariamente, con su usabilidad. Por ello, para obtener una interacción adecuada con el sistema de etiquetado semiautomático desarrollado, se implementa un interfaz de usuario sencillo e intuitivo.

El interfaz de control del sistema de etiquetado semiautomático consta de varias secciones fácilmente identificables.

- imagen grande de cámara color
- imagen pequeña de cámara b/n
- imagen grande de imagen binaria resultado de la comparación entre la captura actual y la anterior
- control de posición de cada una de las cámaras

- control de posición de la cabeza robotizada
- control de la imagen de la cámara a color para hacerla coincidir con la imagen de la cámara b/n.
- control de vibración de la cámara b/n
- control de captura de imagen
- selección de resolución de la imagen de las cámaras
- control de umbral para generación de imagen binaria

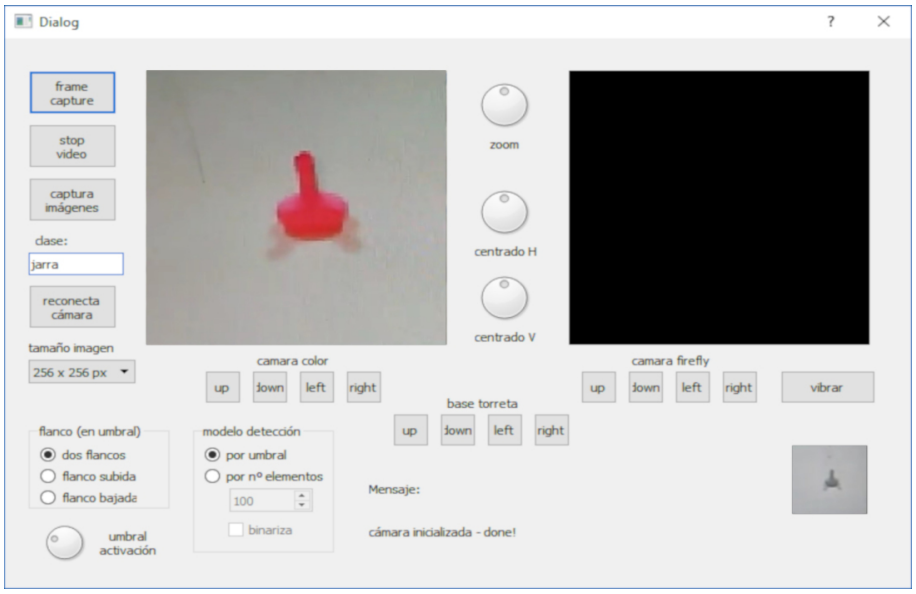


Figura 6.6: interfaz de control con vibración desactivada

Una vez se lanza el programa, comprueba si están conectadas las cámaras, notificando de su estado en la propia interfaz. Con el pulsador de "start/stop vídeo" se conecta con la cámara color para que se visualice en la ventana correspondiente.

Con la imagen de la cámara color en la interfaz, se procede a modificarla para que el objeto que aparece se encuentre en tamaño y posición lo más similar al objeto que muestra la cámara b/n en la ventana pequeña.

A partir de ese momento, se puede activar la vibración de la cámara b/n y se comprueba si el umbral es el adecuado.

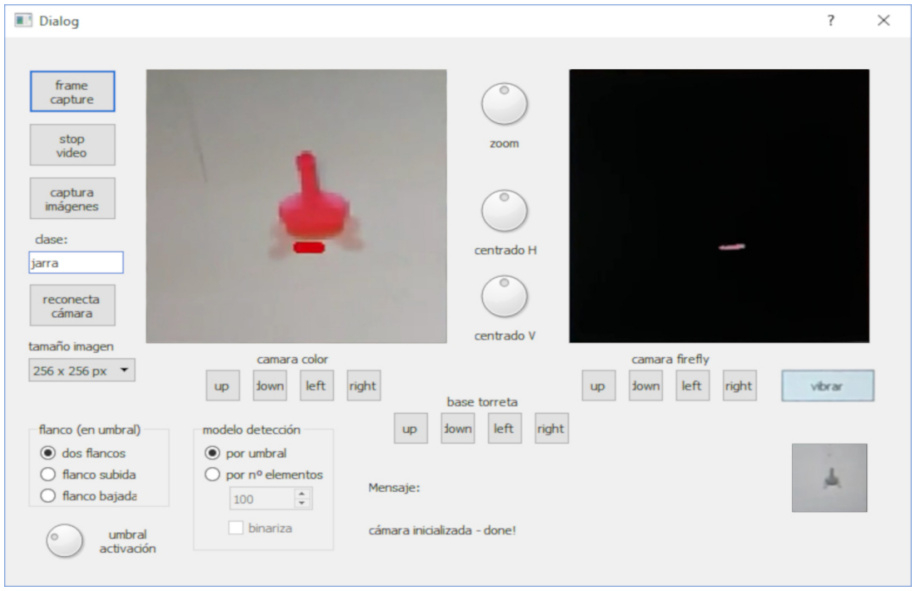


Figura 6.7: interfaz de control con vibración activada y umbral alto

En caso de no detectar correctamente el objeto, se disminuye el umbral de diferencia entre pixels para que la imagen binaria ofrezca más información.

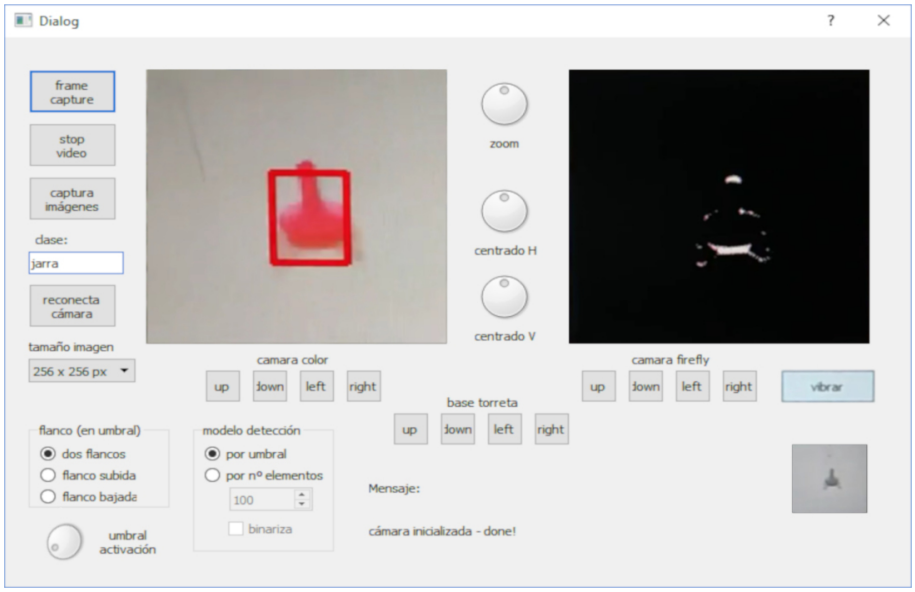


Figura 6.8: interfaz de control con vibracicón activada, umbral bajo y objeto de cámara a color desubicado

Ajustado el umbral de detección, se procede al refinado en la posición del objeto que muestra la ventana de la cámara a color. De este modo, el contenedor rectangular calculado coincide lo máximo posible con el objeto de la cámara color.

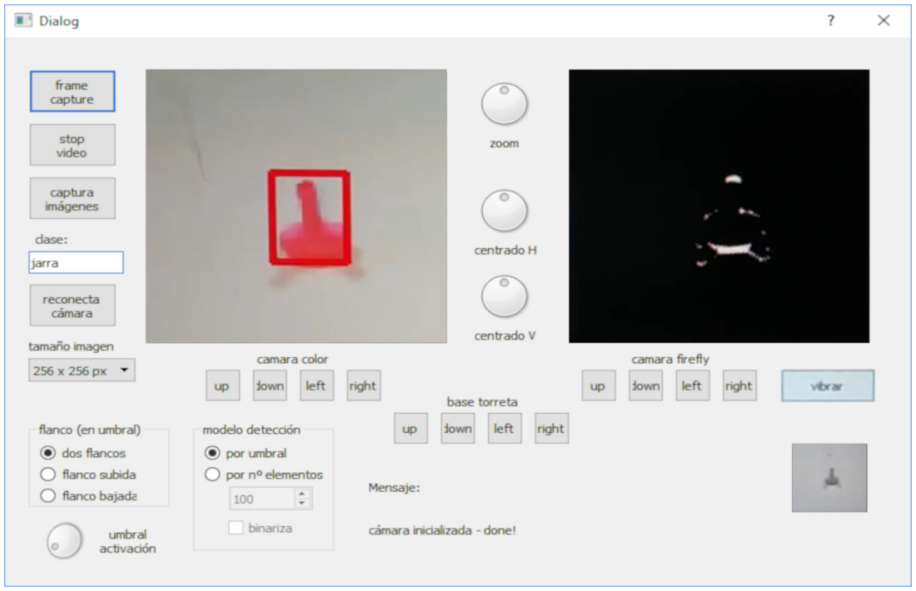


Figura 6.9: interfaz de control con vibracicón activada, umbral bajo y objeto de cámara a color desubicado

Terminado todo el proceso de ajuste, se está en disposición de comenzar la captura y etiquetado de los objetos personalizados. Para ello,

- 1. se sitúa un objeto delante de la cabeza robotizada
- 2. se asigna un nombre al objeto y que constituirá la clase con la que se clasificará en el sistema de detección en tiempo real.
- 3. se pulsa sobre "captura imágenes" y el sistema guarda 3 juegos de imágenes y etiquetas. Esto es así por el efecto de la cámara b/n que varía la intensidad de los pixels ligeramente entre capturas, modificando la imagen binaria resultante. Este efecto ofrece coordenadas ligeramente diferentes para la misma imagen, lo que, a priori, puede resultar interesante.

  Cada juego se compone de las imágenes correspondientes a la captura de la cámara color, la captura de la cámara b/n y la imagen binaria.
- 4. se cambia la perspectiva del objeto por giro, diferente apoyo o cualquier otra alternativa.
- 5. se repite desde el paso 3 tantas veces como sea necesario para identificar al objeto desde la mayor cantidad de ángulos y perspectivas posible.
- 6. se repite desde el paso 1 tantas veces como objetos a clasificar.

Figura 6.10: diferentes capturas del mismo objeto

Es prudente ir verificado que el ajuste previo entre las imágenes de las dos cámaras es correcto. En caso contrario, realizar los ajustes precisos para ello.

6.5. Interfaz para etiquetado manual

En el proceso de pruebas, se grabó parte de las acciones que se estaban realizando. Al visionar los ficheros grabados, se estimó interesante crear una herramienta capaz de facilitar el etiquetado manual partiendo de la visualización del fichero de vídeo.

De este modo se podrían crear dos conjuntos de datos diferenciados por la precisión de la detección y el etiquetado. Con ellos se podrían realizar pruebas comparativas para validar el sistema de etiquetado semiautomático.

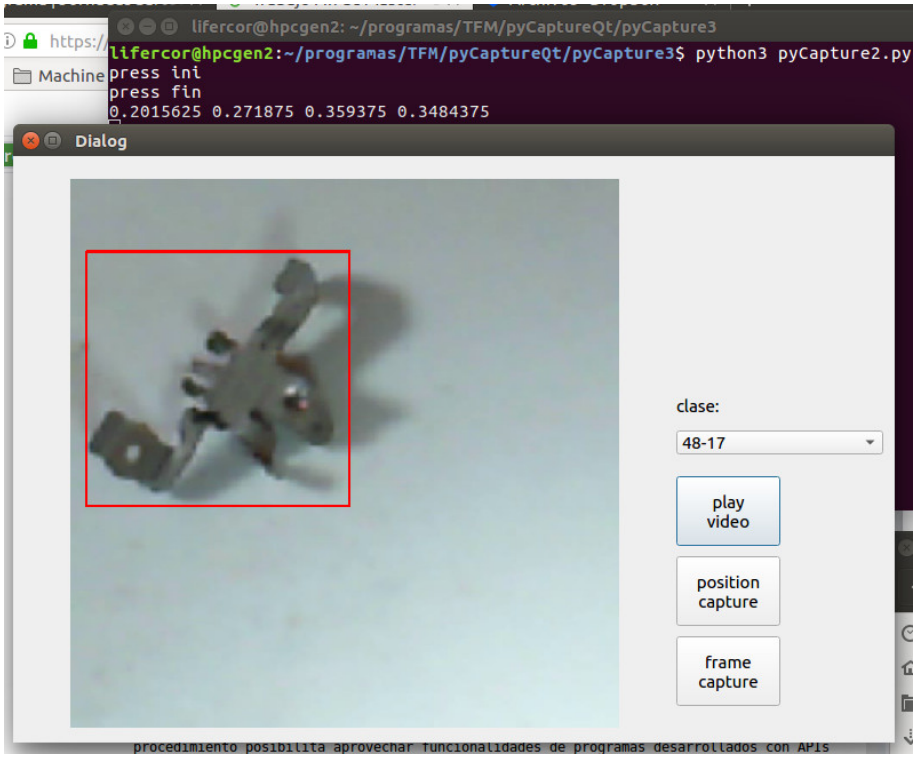


Figura 6.11: Interfaz de etiquetado manual

El principio de usabilidad se aplica igualmente a esta herramienta. Se puede observar en la figura 6.11 que solo existe una ventana en la que se visualiza el vídeo. A través de los diferentes pulsadores, se puede parar y reanudar el vídeo en cualquier momento. Una vez seleccionada la imagen a etiquetar, se selecciona la clase a la que pertenece el objeto a la vista y se enmarca con el ratón. A partir de este punto, se realiza la captura. Automáticamente, la imagen recibe un nombre derivado de la clase y la etiqueta guarda las coordenadas marcadas con el ratón.

6.6. Aumento artificial de datos

Terminado el proceso de generación semiautomática de imágenes y etiquetas, se dispone de 2271 imágenes etiquetadas conforme el proceso establecido. Estas imágenes forman el conjunto base de entrenamiento para el sistema de visión en tiempo real *YOLOv3*.

Terminado el proceso de etiquetado manual de los vídeos grabados, se dispone de 1477 imágenes con sus correspondientes etiquetas.

Estos conjuntos son claramente insuficientes para afrontar el entrenamiento de un sistema basado en *deep ConvNet*. El conjunto de datos debe contener miles, decenas o centenares de miles de imágenes etiquetadas para garantizar una precisión aceptable.

Con este principio, se desarrolla un programa que se encargue de ampliar el número de imágenes etiquetadas. Se desarrolla con el lenguaje interpretado *Python 3.5* dada su velocidad de implementación y versatilidad.

Elegimos incrementar el conjunto de datos pasando cada uno de los conjuntos generados por un proceso de **escalado**. Para cada escala se realiza un conjunto de **traslaciones** y sobre cada traslación se realizan **rotaciones**.

Al desconocer a priori cual es la mejor combinación de número de escalas, número de traslaciones y número de rotaciones, se generan varios conjuntos con lo que realizar pruebas de precisión.

Por un lado, se aumenta el total de imágenes del conjunto generado con la herramienta de etiquetado semiautomático y del conjunto etiquetado manualmente. Se definen, para ello, los siguientes parámetros: 5 escalados, una rejilla con 25 traslaciones sobre cada escalado y 8 rotaciones sobre cada posición trasladada, que identificaremos como "E5-T25-R8-C".

Esta operación aumenta el conjunto generado por las cámaras, desde las 2271 imágenes hasta 1271760. El conjunto generado manualmente pasa de las 1477 imágenes a 627200.

Revisando algunas imágenes resultantes de ambos conjuntos, se observa que, tras las modificaciones en escala, posición y giro, muchas de ellas muestran zonas negras del fondo sin información alguna.

Con esta situación se decide preparar uno de los grupos realizando un ***chroma*** sobre el negro con imágenes coloridas y llenas de objetos. Así, dispondremos de dos

conjuntos claramente diferenciados con los que entrenar el sistema de detección en tiempo real.

Figura 6.12: imagen escalada, trasladada, girada y tratada con chroma sobre fondos diferentes

Como el número de elementos de los conjuntos es muy elevado para realizar un entrenamiento en tiempo limitado a pocos días. Así, se decide realizar una elección random de un tercio de los elementos del primer conjunto (408780 imágenes) y mantener la totalidad de los elementos del segundo conjunto de datos.

Por otro lado, se separa el subconjunto de las imágenes tomadas con la cámara de color para preparar diferentes conjuntos de entrenamiento generados con diferentes parámetros de escalado, traslación y rotación. Con ello se pretende testar la respuesta del sistema entrenado con diferente número de imágenes. Cada conjunto de datos se testará con un subconjunto separado del de entrenamiento y con un conjunto común. El conjunto común será de imágenes a las que se les ha añadido el efecto chroma.

Se define la configuración de parámetros que se puede observar en la tabla 6.2

Tabla 6.2: identificador de parámetros para el data$_a ugmentation$

id	**escalados**	**traslaciones**	**rotaciones**	**imágenes**
E5-T20-R8	5	20	8	377280
E5-T20-R8-C	5	20	8 chroma	377280
E5-T20-R4	5	20	4	188640
E5-T12-R4	5	12	4	141480
E3-T12-R4	3	12	4	84888

Como es de esperar, la etiqueta también se modifica para que su información corresponda con cada una de las transformaciones a las que se somete la imagen.

6.7. Entrenamiento

Sobre los conjuntos de datos seleccionados, se realiza un reparto aleatorio de imágenes para formar el conjunto de entrenamiento y el conjunto de test de cada

uno. Se selecciona un 10 % de los elementos de cada conjunto para realizar el test sobre los resultados del entrenamiento y el 90 % restante para entrenar.

Para realizar el entrenamiento se utiliza una rama del sistema de detección en tiempo real *Yolo v3* disponible en *GitHub*. La versión corre a cargo de AlexeyAB (`https://github.com/AlexeyAB/darknet`). La principal razón para elegir esta rama del sistema desarrollado por J. Redmon et al. [21] es la posibilidad de guardar los pesos cada 100 iteraciones, manteniendo su entorno de desarrollo en *C* y Darknet como *framework* de red neuronal desarrollado por J. Redmon et al. [21]. De este modo, se pueden comparar los resultados de test con diferentes pesos.

La inicialización de los pesos para comenzar el proceso de entrenamiento con cualquiera de los dos conjuntos de datos se puede hacer aleatoriamente o a partir de los pesos que ofrece J: Redmon et al. [21] en su *GitHub*. Los pesos son producto del entrenamiento con el conjunto de datos ImageNet [23]. Al disponer de diferentes objetos similares a parte del conjunto de datos personalizado, tales como "pelota", se estima adecuado inicializar con este último archivo de pesos para entrenar a partir de él.

Como el número de objetos a detectar es diferente de la configuración por defecto del fichero de control del *framework*, se realizan las modificaciones oportunas para adecuarlo a los 20 objetos de nuestro conjunto de datos personalizado.

CAPÍTULO 7

EXPERIMENTOS Y RESULTADOS

Tras la conclusión de las diferentes pruebas de aceptación de cada una de las historias de usuario, se puede entender que la funcionalidad del sistema se ha testado correctamente, cubriendo el total de las expectativas iniciales.

De modo que, este apartado, en lugar de ser un último paso, posterior y ajeno al capítulo de implementación, se ha incorporado dentro de cada una de las historias de usuario. Siendo más preciso, las historias de usuarios 1 a 6 incluían pruebas modulares de cada una de las partes del sistema. No así la historia de usuario 7. El propósito de esta última historia ha sido, precisamente, realizar el entrenamiento del sistema de detección en tiempo real *YOLO v3* para testar sus resultados.

Así, a continuación se describen las pruebas planificadas y realizadas al sistema en la historia de usuario referida.

7.1. Descripción de Experimentos

7.1.1. Pruebas funcionales

Las pruebas que nos proporcionarán información sobre el correcto funcionamiento del sistema se realizarán sobre dos máquinas con configuraciones *hardware* diferentes. Por un lado, un ordenador portátil con procesador Intel(R) i7-4700MQ, 4 cores hiperthreading, 8Gb de RAM y gráfica GT740M. Por otro lado, un PC con procesador Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz de 24 núcleos en dos sockets de 12 núcleos cada uno, con 64 Gb de memoria RAM y una tarjeta gráfica GeForce GTX 1080 con 2560 cores y 8 Gb de memoria GDDR5X.

Una vez se dispone de los resultados en forma de archivos de pesos para diferente número de iteraciones en el proceso de entrenamiento, se realizarán evaluaciones sobre los conjuntos de test de cada uno de los dos conjuntos de datos. Con ello se evaluará la precisión en la detección del conjunto de test.

Además, se comprobará la capacidad de detección de los objetos conectando una cámara color.

7.1.2. Pruebas de rendimiento

Las limitaciones del sistema se comprobarán evaluando el tiempo de entrenamiento entre iteraciones.

7.2. Resultados y discusión

7.2.1. Pruebas funcionales

El entrenamiento sobre cualquiera de las plataformas *hardware* ha sido el esperado. El diseño de los conjunto de datos, conforme con las directrices de J. Redmon et al [**?**] y AlexeyAB (`https://github.com/AlexeyAB/darknet`), no ha revestido dificultad ni error alguno.

El total de los conjuntos de datos generados tanto con el proceso semiautomático como con el manual y, porteriormente, aumentados artificialmente hasta conseguir una dimensión adecuada para el entrenamiento de esta *ConvNet*, se separan en dos subconjuntos. Ayudados por un pequeño programa en *PYthon 3.5* se automatiza la separación aleatoria del 90 % de las imágenes para realizar el entrenamiento, dejando el 10 % restante para realizar las pruebas de test.

Agrupados los ficheros de imagen con los ficheros que guardan el etiquetado correspondiente para cada imagen en un mismo directorio y generados los ficheros de configuración correspondientes para el entrenamiento, se procede.

Concluido el entrenamiento se pasa la batería de imágenes de test.

Los resultados obtenidos se dividen en los dos bloques señalados en 6.6.

El primer bloque, con los dos conjuntos de datos grandes, el resultado ha sido absolutamente diferente entre los dos conjuntos. La figura 7.1 muestra los datos de precisión para diferente número de iteraciones. Se puede observar que, con relativamente pocas iteraciones, se pueden obtener números muy interesantes y esperanzadores en el proceso de detección. Sin embargo, el conjunto de datos resultado del etiquetado manual y sin las imágenes de fondo, ofrece unos resultado extremadamente pobres. La figura 7.2 refleja una muestra de ello.

	precision	recall	F1-score	TP	FP	FN	average IoU	mAP
5000	0,78	0,41	0,54	16896	4819	23982	0,5317	0,5017
10000	0,83	0,60	0,69	24355	5090	16523	0,5731	0,6488
15000	0,85	0,69	0,76	28356	4938	12522	0,6114	0,7193
20000	0,85	0,65	0,74	26640	4861	14238	0,6047	0,7006
25000	0,79	0,53	0,63	21592	5857	19286	0,5674	0,5663
30000	0,83	0,65	0,73	26541	5602	14337	0,5794	0,7288
33000	0,77	0,37	0,50	15220	4570	25658	0,5632	0,5062
33500	0,83	0,53	0,65	21838	4452	19040	0,7079	0,5793
34000	0,80	0,48	0,60	19677	5057	21201	0,5771	0,5858
34500	0,84	0,66	0,74	26909	4978	13969	0,6215	0,7237
34600	0,80	0,67	0,73	27549	7082	13329	0,5733	0,6878
34700	0,88	0,72	0,79	29409	3969	11469	0,6526	0,8012
34800	0,89	0,74	0,81	30446	3625	10432	0,6680	0,7979
34900	0,79	0,46	0,58	18886	4926	21992	0,5577	0,5489
35000	0,88	0,80	0,84	32708	4617	8170	0,6419	0,8106
35100	0,85	0,68	0,76	27955	4814	12923	0,5944	0,7517
35200	0,79	0,53	0,64	21802	5674	19076	0,5774	0,5693
35300	0,76	0,53	0,63	21791	6944	19087	0,5560	0,5638
35400	0,85	0,67	0,75	27258	4944	13620	0,6157	0,7354
35500	0,84	0,71	0,77	29064	5599	11814	0,6189	0,7409
36000	0,79	0,67	0,73	27402	7145	13476	0,5472	0,6777
36500	0,76	0,47	0,58	19268	6202	21610	0,5316	0,5082
37000	0,72	0,50	0,59	20516	8026	20362	0,5287	0,5843
37500	0,81	0,54	0,65	22218	5050	18660	0,6079	0,6625
37600	0,87	0,76	0,81	31104	4627	9774	0,6318	0,7829
37700	0,68	0,48	0,56	19699	9250	21179	0,4553	0,4983
37800	0,91	0,80	0,86	32891	3104	7987	0,6724	0,8200
37900	0,89	0,76	0,82	30817	3973	10007	0,6382	0,7859
38000	0,86	0,77	0,82	31618	4962	9260	0,6361	0,7654
38100	0,88	0,75	0,81	30748	4141	10130	0,6450	0,7820
38200	0,80	0,66	0,72	26787	6645	14091	0,5941	0,6960
38300	0,79	0,60	0,68	24438	6401	16440	0,5786	0,6740
38400	0,86	0,66	0,75	27107	4564	13771	0,6332	0,7147
38500	0,81	0,49	0,61	19831	4722	21047	0,5823	0,6423
38600	0,90	0,83	0,86	33872	3567	7006	0,6583	0,8337
38700	0,90	0,80	0,84	32500	3737	8378	0,6755	0,7874
38800	0,64	0,36	0,46	14696	8231	26182	0,4582	0,4987
38900	0,82	0,50	0,62	20605	4453	20273	0,6138	0,5976
39000	0,85	0,66	0,74	27039	4671	13839	0,6336	0,7153
39100	0,86	0,73	0,79	29791	5018	11087	0,6103	0,7648
39200	0,73	0,39	0,51	15817	5830	25061	0,5333	0,5338
39300	0,72	0,46	0,56	18774	7216	22104	0,5270	0,5724
39400	0,89	0,81	0,84	32908	4165	7970	0,6508	0,8131
39500	0,86	0,65	0,74	26561	4364	14317	0,6298	0,7435
39600	0,82	0,49	0,61	19863	4425	21015	0,5863	0,5960
39700	0,90	0,81	0,86	33241	3560	7637	0,6624	0,8023
39800	0,89	0,75	0,82	30759	3804	10119	0,6613	0,7781
39900	0,90	0,81	0,85	33125	3679	7753	0,6605	0,8136
40000	0,81	0,41	0,55	16895	3940	23983	0,5826	0,5571

Figura 7.1: tabla de resultados con pesos de diferentes iteraciones sobre conjuntos de datos con etiquetado semiautomático y *chroma*

	precision	recall	f1-score	iou	map
5000	0,000	0,000	0,000	0,000	0,000
7500	0,000	0,000	0,000	0,000	0,000
10000	0,320	0,000	0,000	0,230	0,008
12500	0,050	0,000	0,000	0,033	0,004
15000	0,250	0,000	0,010	0,161	0,021
17500	0,060	0,000	0,000	0,038	0,003
20000	0,230	0,000	0,000	0,155	0,026
22500	0,140	0,000	0,000	0,091	0,007
25000	0,300	0,000	0,010	0,196	0,027
27500	0,270	0,000	0,010	0,173	0,035
30000	0,370	0,010	0,010	0,244	0,042
32500	0,320	0,010	0,010	0,213	0,039
35000	0,320	0,000	0,010	0,207	0,037
37500	0,260	0,000	0,010	0,174	0,030
40000	0,320	0,010	0,010	0,212	0,053

Figura 7.2: tabla de resultados con pesos de diferentes iteraciones sobre conjuntos de datos con etiquetado manual sin *chroma*

La gran diferencia de resultados entre el conjunto de datos con etiquetado semiautomático y el de etiquetado manual puede deberse a varios factores. Entre ellos, estimamos que trabajar con diferentes representaciones del objeto, ofrecidas por las dos cámaras y la imagen binaria, junto al etiquetado automático, no extremadamente preciso, por detección de la cámara b/n y el enriquecimiento de la imagen con el *chroma*, enriquece la extracción de características. Esta situación no se da en el conjunto de datos con etiquetado manual. Es más preciso pero más plano en cuanto a características globales de la imagen.

En las gráficas 7.3, 7.4 y 7.5, derivadas de las tablas anteriores, se observa un incremento progresivo pero leve de la precisión conforme aumenta el número de iteraciones de entrenamiento. Se estima que puede deberse al estar entrenando sobre una red preentrenada, de modo que el entrenamiento sobre ella con el conjunto de datos personalizado solo aporta pequeñas variaciones.

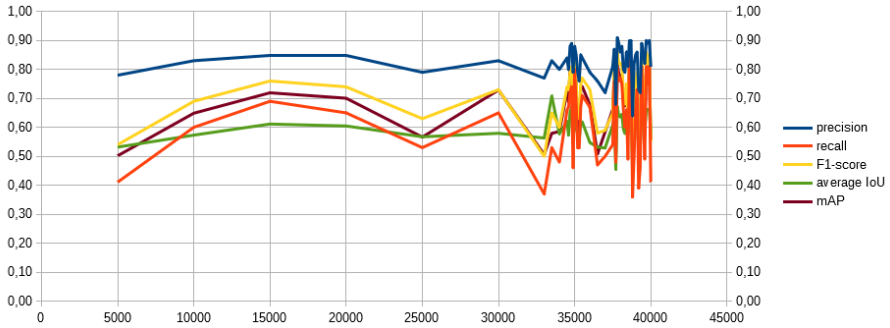


Figura 7.3: gráfica de resultados con pesos de diferentes iteraciones sobre conjuntos de datos con etiquetado semiautomático con *chroma*

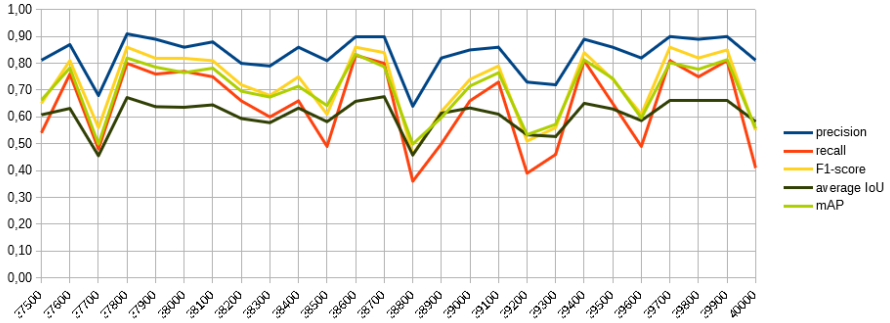


Figura 7.4: gráfica de resultados a partir de las 37000 iteraciones, con pesos de diferentes iteraciones sobre conjuntos de datos con etiquetado semiautomático con *chroma*

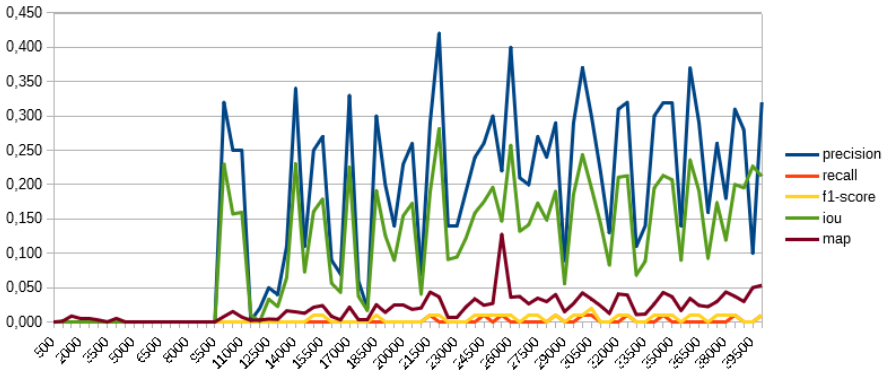


Figura 7.5: gráfica de resultados con pesos de diferentes iteraciones sobre conjuntos de datos con etiquetado manual sin *chroma*

Por otro lado, la precisión individual por objeto del conjunto de datos, centrando el análisis en el conjunto de datos correspondiente al etiquetado semiautomático, se puede ver en la tabla 7.1.

Tabla 7.1: mAP % resultado test conjunto 1

id	**clase**	**E5-T25-R8-C**
0	cucaracha	69.47
1	jarra	68.12
2	peonza	90.17
3	coche	89.28
4	cubito	69.98
5	robot	86.00
6	pelota	85.51
7	diapasón	86.68
8	candado	75.03
9	llave	52.14
10	grapadora	88.69
11	jugador	69.91
12	C83FA	89.94
13	48-17	90.82
14	17W45	89.34
15	17W35	87.88
16	18-17I	85.60
17	18-17D	86.70
18	37-17	89.02
19	40-17	89.91

Resulta interesante observar los valores de las piezas especiales del taller de mecanizado frente al conjunto de objetos comunes. En general obtienen mejores resultados que un "cubo", un "candado" o una "llave". Es posible que no existan este tipo de objetos en el conjunto utilizado para el preentrenamiento o que no sean

muy comunes. De ahí, se podría concluir que le cuesta generalizar sobre ellos. Sin embargo, choca contra los buenos resultados sobre los objetos mecanizados. Una posible explicación es que, por su tamaño o gran cobertura sobre el total de la imagen, sea capaz de obtener o generalizar mejor por asociación con otros objetos.

En la figura 7.6 y 7.7 se puede ver algunos ejemplos de detección sobre el conjunto de test.

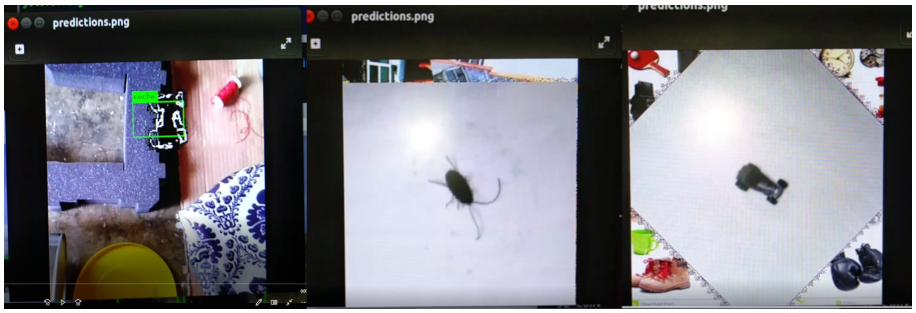


Figura 7.6: imágenes del conjunto de test pasadas por el sistema de detección en tiempo real

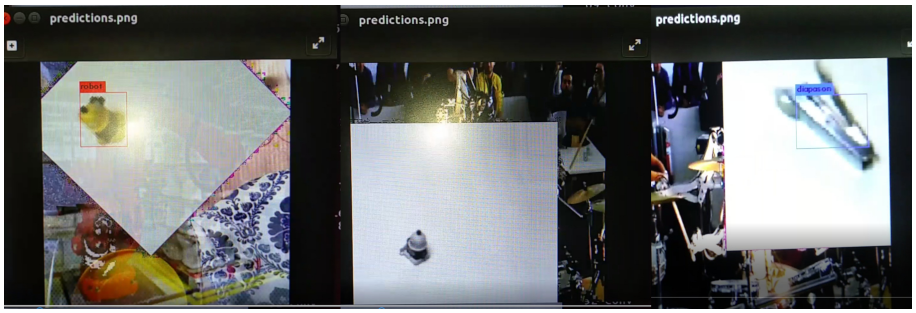


Figura 7.7: imágenes del conjunto de test pasadas por el sistema de detección en tiempo real

Con las pruebas de cámara color, los resultado no han sido todo lo buenos que se esperaba. Quizá no se había establecido un modo riguroso de medición. Sin embargo, el sistema detecta, bajo condiciones de iluminación, entorno y zoom adecuado, la práctica totalidad de los objetos del conjunto. Las figuras 7.8 y 7.9 muestran algunos ejemplos.

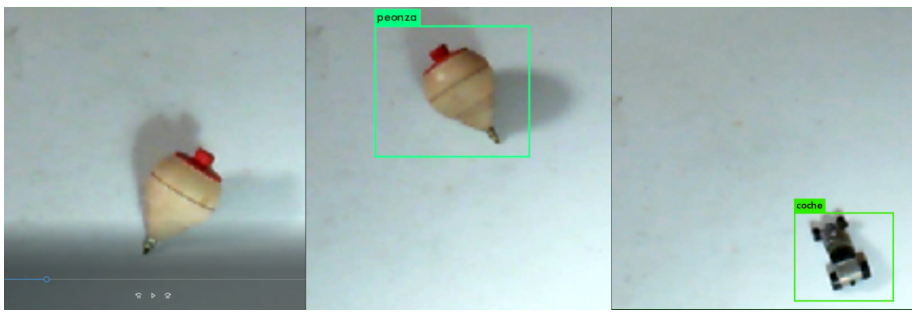


Figura 7.8: imágenes de cámara del sistema de detección en tiempo real

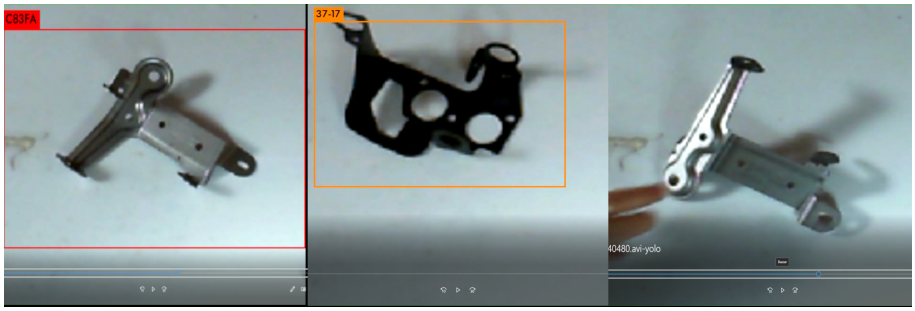


Figura 7.9: imágenes de cámara del sistema de detección en tiempo real

En el siguiente bloque, correspondiente al estudio de precisión de los diferentes conjuntos generados según la tabla 6.2, fruto de la variación de los parámetros para el aumento de datos, la mAP evoluciona tal y como se observa en la tabla 7.2 en cada número de iteraciones.

Tabla 7.2: identificador de objetos

iteraciones	**E5T20R8**	**E5T20R8C**	**E5T20R4**	**E5T12R4**	**E3T12R4**
10000	74.57	56.14	54.81	33.75	50.99
20000	82.98	72.53	88.75	51.02	52.23
30000	87.33	77.78	90.01	58.64	57.52
40000	88.57	80.99	82.16	56.65	59.46
50000	85.46	79.42	93.69	57.54	60.49

La figura 7.10 muestr que, en general, a partir de las 30000 iteraciones, la mejora en la precisión se estabiliza bastante y no hay un avance en las prestaciones sensible. Asemejando el comportamiento al observado en el bloque anterior.

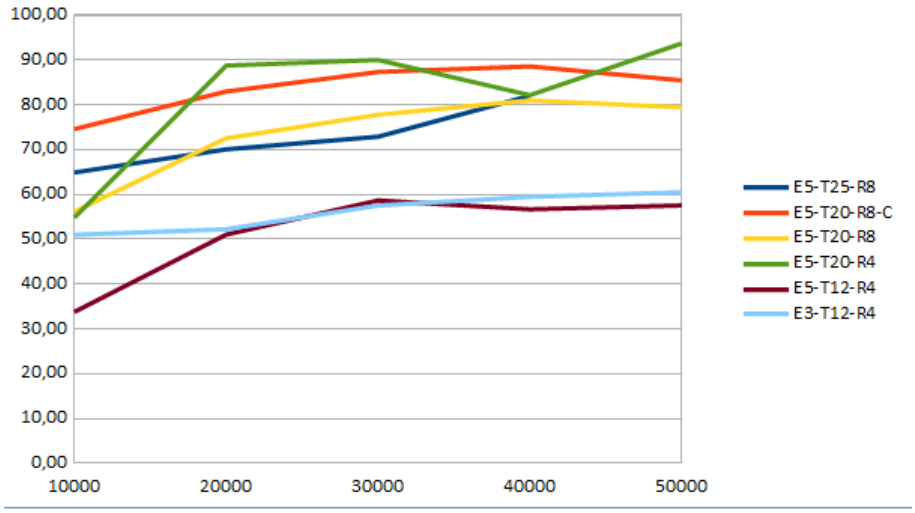


Figura 7.10: gráfica de evolución de mAP por conjunto entrenado e iteraciónes.

En cuanto a la precisión en función de los parámetros de cada uno de los conjuntos, se aprecia un incremento importante en el identificado con la referencia

E5T20R4. Prácticamente con la mitad de las imágenes para entrenar, obtiene una precisión muy por encima del resto de los conjuntos, lo que resulta francamente llamativo. Se podría pensar en un sobreentrenamiento si no fuera porque la división entre conjunto de entrenamiento y conjunto de test se ha realizado de la misma forma para todos los conjuntos. En la tabla 7.3 se puede observar el comportamiento para cada uno de los objetos, destacando la referencia citada.

Tabla 7.3: mAP % mejores resultados test conjuntos

id	**objeto**	**E5T20R8**	**E5T20R8C**	**E5T20R4**	**E5T12R4**	**E3T12R4**
0	cucaracha	88.90	87.75	90.75	59.53	67.56
1	jarra	77.13	70.67	84.05	47.17	58.73
2	peonza	89.71	86.59	99.98	63.62	61.49
3	coche	90.44	90.65	90.91	60.55	62.93
4	cubito	88.78	74.31	90.70	53.60	55.45
5	robot	90.73	86.59	90.66	54.49	69.46
6	pelota	90.91	73.99	100.00	40.75	47.40
7	diapasón	89.28	58.08	88.07	41.75	54.15
8	candado	88.68	76.39	90.40	55.23	59.18
9	llave	79.06	34.65	86.51	59.02	53.47
10	grapadora	90.21	90.40	90.87	58.87	60.53
11	jugador	89.01	73.06	89.16	55.95	59.49
12	C83FA	88.18	89.32	100.00	63.44	63.45
13	48-17	90.61	90.79	100.00	61.59	70.34
14	17W45	90.02	88.93	100.00	61.52	57.51
15	17W35	90.44	88.62	100.00	61.38	57.40
16	18-17I	90.72	90.39	100.00	62.52	61.87
17	18-17D	90.37	90.83	100.00	69.47	69.26
18	37-17	89.42	87.22	90.91	57.53	57.75
19	40-17	89.25	90.51	90.90	62.84	62.43

Tal y como ocurría en el bloque de dos conjuntos previo (figura 7.1), se observa que el grupo de objetos correspondiente a las piezas de fábrica, obtienen mejor precisión media que el grupo con el resto de objetos cotidianos y más comunes. No se puede aventurar una idea concreta, salvo, quizá, que su tamaño medio es superior al de objetos comunes, como elemento diferenciador que pueda influir en la mayor precisión.

Con objeto de sacar de la zona de confort a cada uno de los conjuntos entrenados en este bloque, se realiza un test de los pesos que mejor puntuación ha obtenido cada uno de los conjuntos, sobre el conjunto de test de los objetos con chroma.

La figura 7.4 muestra el resultado de cada uno de los test realizados. El comportamiento de cada uno de los conjuntos es sensiblemente diferente. El conjunto perteneciente al primer bloque tiene un comportamiento muy pobre, alejado de su 82.00 % debido, supuestamente, a su entrenamiento con muchos objetos en blanco y negro y binarizados. Contrasta con lo observado en el bloque anterior, pero hay que tener en cuenta que el conjunto de test es exclusivamente de fotos en color, con

lo que es posible que no pueda compensar la precisión que puede obtener con otro tipo de imágenes.

Tabla 7.4: precisión frente a conjunto con chroma

id	mAP %
E5-T25-R8-C	48.82
E5-T20-R8	84.30
E5-T20-R8-C	80.99
E5-T20-R4	52.32
E5-T12-R4	50.12
E3-T12-R4	49.10

Por otro lado, el conjunto E5-T20-R8 ha tenido un comportamiento cercano al de su mismo conjunto de test y, curiosamente, supera la precisión del conjunto entrenado con imágenes del mismo conjunto de test de esta prueba. Para encontrar una explicación razonable se deberían realizar más pruebas. En la figura 7.11 también se observa como el conjunto E5T20R4 que tan bien se había comportado con su conjunto de test, se muestra bastante débil, puntuando casi un 40 % peor. No así los conjuntos más humildes, cuyo comportamiento es algo peor que con su propio conjunto de test. Pero esta circunstancia solo refuerza el hecho de haber sido entrenados con un conjunto pobre de imágenes.

Observamos que el comportamiento de los conjuntos entrenados con diferentes valores en los parámetros, ha respondido a lo esperado frente a la teoría sobre la necesidad de entrenar las redes profundas con grandes conjuntos de imágenes si se pretende obtener buenos niveles de precisión.

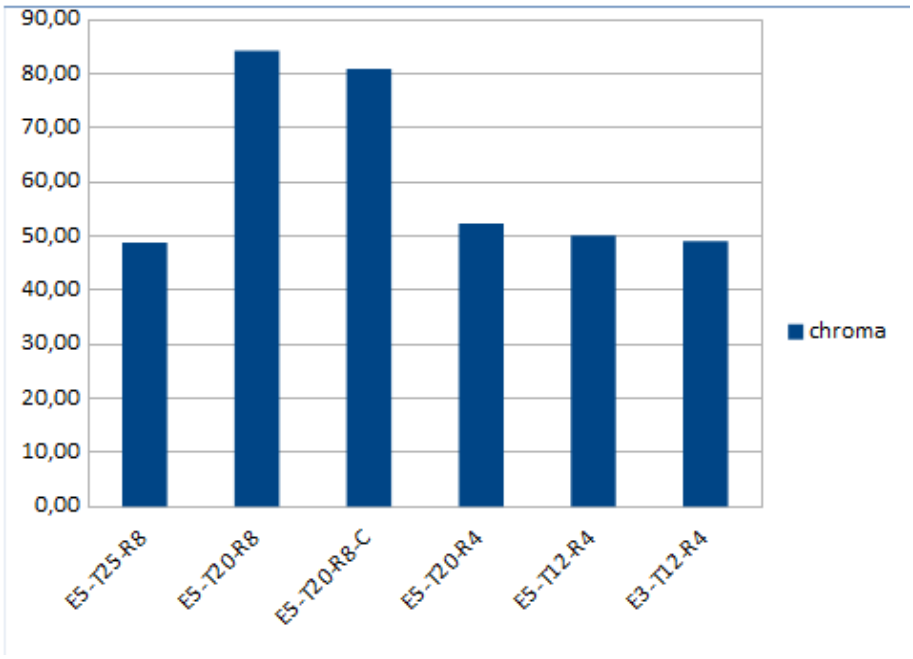


Figura 7.11: comparativa test de conjuntos entrenados frente a conjunto de test con chroma

7.2.2. Pruebas de rendimiento

Los tiempos entre iteraciones decantan rápidamente la elección hacia el PC. No se deja terminar el proceso de entrenamiento completo en el portátil por su exagerada lentitud. Aún así, el proceso de entrenamiento, de 40000 iteraciones, para un solo conjunto de datos sobre la plataforma PC, ha precisado de 5 días de trabajo continuado con el primer conjunto de datos. El segundo bloque de pruebas con conjuntos más pequeños, precisaba de 3 días de entrenamiento.

CAPÍTULO 8

CONCLUSIONES Y TRABAJO FUTURO

8.1. Conclusiones

Una vez se da por finalizado el trabajo planificado, se entiende que se ha alcanzado prácticamente el total del objetivo principal. Los objetivos secundarios, necesarios para la consecución del principal, se entienden cumplidos en su totalidad.

Así, el conocimiento adquirido en:

- Sistemas de visión artificial para la detección de objetos en tiempo real.
- Redes convolucionales profundas.
- Desarrollo de conjuntos de datos para el entrenamiento de redes ConvNet.
- Trabajo de control con dispositivos hardware de diferente tipo.
- Planificación, evaluación, metodologías ágiles y gestión de proyectos en su conjunto.

ha supuesto un enriquecimiento importante a nivel académico. A la par, ha acercado el objetivo más importante que se halla en lo más profundo de este trabajo: desarrollar la base de un sistema que pueda covertirse en un proyecto comercial.

8.2. Trabajo futuro

Tal y como se ha adelantado en el apartado anterior, el presente TFM ha representado una importante introducción en el mundo de la inteligencia artificial, desde el punto de vista de los sistemas de visión por computador con ConvNets.

Supone el inicio de un proceso de estudio importante para el desarrollo de versiones mejoradas de los conjuntos de datos personalizados y sistemas de etiquetado más eficientes y usables, con el fin de abrir los sistemas de detección a entornos personalizados.

Sobre la base del desarrollo y entrenamiento de estos sistemas de visión en tiempo real, se enlazarán directamente los sistemas robotizados de diferente índole para aumentar su capacidad de relación en un entorno, profesional o doméstico, donde interactuar con las personas.

BIBLIOGRAFÍA

[1] R. Aracil, C. Balaguer y M. Armada, *Robots de servicio*, 2008, http://recyt.fecyt.es/index.php/RIAII/article/view/794

[2] European Robotics Research Network, http://www.euron.org/

[3] IFR - International Federation of Robotics, http://www.ifr.org/

[4] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, *Gradient-based learning applied to document recognition*, in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998. doi: 10.1109/5.726791. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=726791&isnumber=15641

[5] Alex Krizhevsky, Ilya Sutskever y Geoffrey E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, January 2012. Advances in neural information processing systems 25(2) DOI: 10.1145/3065386. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[6] Paul Viola y Michael J. Jones, *Robust Real-time Object Detection*, Compaq Computer Corporation, Cambridge Research Laboratory. CRL 2001/01, February 2001 http://www.hpl.hp.com/techreports/Compaq-DEC/CRL-2001-1.pdf

[7] N. Dalal y B. Triggs, *Histograms of oriented gradients for human detection*, 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 2005, pp. 886-893 vol. 1. doi: 10.1109/CVPR.2005.177 http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1467360&isnumber=31472

[8] Min Lin, Qiang Chen y Shuicheng Yan, *Network In Network*, CoRR, December 2013 https://arxiv.org/abs/1312.4400

[9] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus y Yann LeCun, *OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks*, CoRR, February 2014 https://arxiv.org/abs/1312.6229

[10] Karen Simonyan y Andrew Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*, CoRR, September 2014 https://arxiv.org/abs/1409.1556

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren y Jian Sun, *Deep Residual Learning for Image Recognition*, CoRR, December 2015 https://arxiv.org/abs/1512.03385

[12] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu y Alexander C. Berg, *SSD: Single Shot MultiBox Detector*, CoRR, December 2015 https://arxiv.org/abs/1512.02325

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke y Andrew Rabinovich, *Going Deeper with Convolutions*, CoRR, September 2014 https://arxiv.org/abs/1409.4842

[14] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan y Serge Belongie, *Feature Pyramid Networks for Object Detection*, CoRR, December 2016 https://arxiv.org/abs/1612.03144

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He y Piotr Dollár, *Focal Loss for Dense Object Detections*, CoRR, August 2017 https://arxiv.org/abs/1708.02002

[16] Ross Girshick, Jeff Donahue, Trevor Darrell y Jitendra Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, CoRR, November 2013 https://arxiv.org/abs/1311.2524

[17] Ross Girshick, *Fast R-CNN*, CoRR, April 2015 https://arxiv.org/abs/1504.08083

[18] Shaoqing Ren, Kaiming He, Ross Girshick y Jian Sun, *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*, CoRR, June 2015 https://arxiv.org/abs/1506.01497

[19] Joseph Redmon, Santosh Divvala, Ross Girshick y Ali Farhadi, *You Only Look Once: Unified, Real-Time Object Detection*, CoRR, June 2015 https://arxiv.org/abs/1506.02640

[20] Joseph Redmon y Ali Farhadi, *YOLO9000: Better, Faster, Stronger*, CoRR, December 2016 https://arxiv.org/abs/1612.08242

[21] Joseph Redmon y Ali Farhadi, *YOLOv3: An Incremental Improvement*, CoRR, April 2018 https://arxiv.org/abs/1804.02767

[22] Sergey Ioffe y Christian Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*, CoRR, February 2015 https://arxiv.org/abs/1502.03167

[23] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg y Li Fei-Fei, (* = equal contribution) *ImageNet Large Scale Visual*

Recognition Challenge, International Journal of Computer Vision, April 2015 https://link.springer.com/article/10.1007/s11263-015-0816-y?sa_campaign=email/event/articleAuthor/onlineFirst#

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick y Piotr Dollár, *Microsoft COCO: Common Objects in Context*, CoRR, May 2014 https://arxiv.org/abs/1405.0312

[25] Jonathan Hui, *Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3*, Medium, March 2018 https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

[26] F. Pardo, J. A. Boluda y F. Vegara, *Selective Change Driven Vision Sensor With Continuous-Time Logarithmic Photoreceptor and Winner-Take-All Circuit for Pixel Selection*, IEEE Journal of Solid-State Circuits (Volume: 50, Issue: 3, March 2015) https://ieeexplore.ieee.org/document/7015637/