GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE

Escuela Técnica Superior de Ingeniería de Telecomunicación

Curso académico 2021-2022

Trabajo fin de grado

Sistema de detección de emociones faciales para un robot de bajo coste basado en ROS

Autor: Javier Martínez Madruga

Tutor: Julio Vega Pérez

Documento de **Julio Vega**.

Agradecimientos

Unas bonitas palabras...

Quizás un segundo párrafo esté bien. No te olvides de nadie.

Un tercero tampoco viene mal para contar alguna anécdota...

¿Alguien más? Aunque sean *actores* secundarios.

Un quinto párrafo como colofón.

A alguien especial;
si no, tampoco pasa nada

Madrid, xx de xxxxxx de 20xx
Tu nombre

Resumen

Escribe aquí el resumen del trabajo. Un primer párrafo para dar contexto sobre la temática que rodea al trabajo.

Un segundo párrafo concretando el contexto del problema abordado.

En el tercer párrafo, comenta cómo has resuelto la problemática descrita en el anterior párrafo.

Por último, en este cuarto párrafo, describe cómo han ido los experimentos.

Acrónimos

RUR *Rossum's Universal Robots*

IFR *Federación Internacional de Robótica*

AGV *Vehículo Guiado Automático*

AMR *Robot Móvil Autónomo*

HRI *Interacción humano-robot*

SVM *Support Vector Machine*

KNN *K Nearest Neighbour*

ALU *Unidad Aritmética Lógica*

CPU *Unidad Central de Procesamiento*

SBC *Single Board Computer*

TFG *Trabajo de Fin de Grado*

Índice general

Índice de figuras

Listado de códigos

Listado de ecuaciones

Índice de cuadros

Capítulo 1
Introducción

Quizás algún fragmento de libro inspirador...

Autor, *Título*

La robótica se define como la intersección entre ciencia, ingeniería y tecnología con el objetivo de desarrollar máquinas que realicen tareas de forma automática. El término robot proviene de la palabra checa *robota* que significa *trabajo forzado*, se utilizó por primera vez en la obra de teatro RUR (Rossum's Universal Robots) del autor Karel Capek estrenada en 1921.

Se puede clasificar a los robots en dos grandes campos: robots industriales y robots de servicio. Según la norma internacional ISO 8373:2012 un robot industrial es un manipulador multifuncional, reprogramable y controlado automáticamente, programable en tres o más ejes que puede estar fijo en un área o móvil para su uso en aplicaciones de automatización industrial. Por otro lado, según la Federación Internacional de Robótica (IFR), un robot de servicio es un robot que opera de forma parcial o totalmente autónoma para realizar servicios útiles para el bienestar de los humanos y del equipamiento, excluyendo operaciones de manufactura.

En este primer capítulo se pretende dar un contexto amplio al lector sobre el presente trabajo. El autor comenzará presentando la robótica de servicio, campo para el cual está enfocado nuestro sistema, y continuará avanzando por los distintos conceptos generales hasta los más específicos.

1.1. Robótica de servicio

En los últimos años la robótica de servicio está obteniendo un auge exponencial, cada vez son más los campos en los que los robots ayudan a los seres humanos

intentando mejorar su calidad de vida ayudándoles a realizar tareas peligrosas o proporcionando una compañía agradable a aquellos que lo necesitan. Entre las aplicaciones más importantes encontramos:

- *Limpieza.* Aspiradoras domésticas como iRobot Roomba 980 o limpiadores de ventanas como WinBot 950 (Figura 1.1). Robots caracterizados por típicamente realizar tareas de navegación con el objetivo de recorrer completamente una zona mientras llevan a cabo las labores de limpieza.

Figura 1.1: iRobot Roomba 980 y WinBot 950.

- *Inspección.* Cartografías 3D, inspección de plantas petrolíferas, aerogeneradores o minas. Suelen ser lugares de difícil acceso para los humanos, por lo tanto son tareas que estarán al cargo de robots cuadrúpedos o drones. Los robots con cuatro patas son actualmente los autómatas terrestres más estables del mercado y tienen la capacidad de recorrer terrenos inestables o incluso subir y bajar escales. Uno de los más populares es Spot de Boston Dynamics (Figura 1.2).

Figura 1.2: Robot Spot de Boston Dynamics. Fuente [8]

- *Eduación.* Aquellos enfocados a la docencia tanto infantil como universitaria. Podemos destacar modelos como mBot con posibilidad de programación a través de bloques con mBlock IDE[1] o modelos como el TurtleBot con soporte ROS[2].

- *Conducción autónoma.* Una tecnología cada vez más madura y extendida. Empresas como Waymo (Figura 1.3) o AutoX ya están comercializando productos reales que llevan a cabo esta tarea proporcionando servicios de taxi autónomos. Otros fabricantes como Tesla comercializan también coches semi-automáticos.

Figura 1.3: Taxi autónomo de Waymo. Fuente [8]

- *Logística.* Robots destinados a agilizar el transporte de materiales o productos dentro de una fábrica o almacén. Existen dos grandes grupos: AGV (Vehículo Guiado Automático) y AMR (Robot Móvil Autónomo).

- *Ámbito sanitario.* Es uno de los campos más amplios dentro de la robótica de servicio, existen autómatas realizando múltiples tareas. Podemos encontrar exoesqueletos enfocados en la rehabilitación de la marcha humana como el Atlas de Marsi-Bionics o robots ayudantes en cirugía como Da Vinci o Mako (Figura 1.4).

  Además, en los últimos años está surgiendo y avanzando un nuevo tipo de robótica sanitaria, los robots de compañía o asistentes personales. Estos consiguen empatizar con los pacientes y hacerles pasar un rato más ameno o ayudarles a que no se sientan sólos. Para ello es indispensable realizar un buen HRI (Human Robot Interaction, en inglés), tema del cual trata la siguiente sección. Un ejemplo de robot de compañía sería Robin (Figura 1.5), el cual ayuda a los niños a superar el miedo de ir al médico.

[1] `https://ide.mblock.cc`
[2] `https://www.ros.org/`

Figura 1.4: Robot DaVinci y Robot Mako. Fuente [3] [2]

Figura 1.5: Robot Robin.

1.2. Interacción Humano Robot (HRI)

Podemos definir el HRI como el campo de estudio que intenta comprender, diseñar y evaluar la interacción entre los robots y los seres humanos. Debido a que los robots están cada vez más presentes en nuestras vidas, esta rama de investigación nace con la necesidad de que estos tengan la capacidad de colaborar y vivir con nosotros, los humanos.

La definición de HRI se remonta al año 1941, donde Isaac Asimov habla por primera vez de ello en su novela *Yo, Robot*. Además escribe unas leyes, las que se conocen actualmente como *Las Tres Leyes de la Robótica*, que promueven una interacción segura entre humanos y robots. Las tres leyes son las siguientes:

- *Primera Ley.* Un robot no hará daño a un ser humano ni, por inacción, permitirá que un ser humano sufra daño.
- *Segunda Ley.* Un robot debe cumplir las órdenes dadas por los seres humanos, a excepción de aquellas que entren en conflicto con la primera ley.

- *Tercera Ley.* Un robot debe proteger su propia existencia en la medida en que esta protección no entre en conflicto con la primera o la segunda ley.

Actualmente esas leyes representan el código moral y han sido modificadas por Isaac Asimov y otros autores para conseguir mayor perfección. Además Asimov agregó una cuarta ley por encima de esas tres que viene a ser una generalización de la primera, *La Ley Cero*:

- *Ley Cero.* Un robot no puede dañar a la humanidad o, por inacción, permitir que la humanidad sufra daños.

En el artículo [7] se realiza un estudio enfocado en como el trato proporcionado por un robot a un humano influía en la distancia física de ambos. Para realizar los experimentos usaron un robot que era capaz de expresar emociones a través de gestos y los resultados demostraron que si el robot ejercía un trato poco social y nada empático, la distancia con el humano aumentaba. En cambio, si el robot expresaba un comportamiento amigable y comprensivo, la distancia con el humano disminuía ya que el nivel de confianza con el robot aumentaba. Por lo tanto si lo que queremos es que los robots sean capaces de colaborar y vivir con nosotros, debemos ser capaces de que estos actúen con empatía, comprensión y amabilidad hacia los seres humanos. Esto no es una tarea trivial, es más bien complicada, y surgen muchos baches por el camino.

1.2.1. Problemática. Paradoja de Moravec

Según Dautenhahn el robot debe adaptarse a nuestra forma de expresión y nos debe comprender tal como somos y actuamos. Esto es algo realmente complicado para una máquina ya que no tienen la capacidad de razonar, únicamente se limitan a ejecutar órdenes que previamente han sido programadas, y las reglas de un entorno social humano pueden ser muy variadas y poco esperadas. Este problema lo presenta la *Paradoja de Moravec*.

Según Moravec los actos voluntarios de un humano requieren de poca computación para una máquina, mientras que los actos no conscientes e involuntarios requieren de grandes esfuerzos computacionales. Moravec afirmó «comparativamente es fácil conseguir que las computadoras muestren capacidades similares a las de un humano adulto en tests de inteligencia, y difícil o imposible lograr que posean las habilidades perceptivas y motrices de un bebé de un año» [6]. Esto según él, es debido a la evolución biológica humana.

Todas nuestras habilidades han sido perfeccionadas a lo largo de millones de años por el proceso de selección natural y por lo tanto, sería lógico pensar que si intentamos replicar dichas habilidades en una máquina nos tomaría como mínimo el mismo tiempo proporcionalmente. Muchas de nuestras acciones más valiosas (coger objetos, reconocer voces, prestar atención, habilidades sociales...) son involuntarias y es eso lo que provoca que aplicarles ingeniería inversa sea muy complicado. Sin embargo habilidades como las matemáticas resultan complejas para nosotros, ya que nuestro cuerpo y cerebro no está preparado para ello, y muy triviales para las máquinas.

Conociendo toda esta problemática parece casi imposible que un robot sea capaz de interaccionar con un humano, pero existen últimos avances en la ingeniería que aportan un poco de claridad y optimismo al HRI.

1.2.2. Soluciones robóticas de interacción

Una interacción completa de humano-humano está regida por la vista, el oído y el tacto. Esos tres sentidos proporcionan toda la información que posteriormente nuestro cerebro procesará y razonará. Podemos concluir diciendo entonces que la interacción entre un humano y un robot estará compuesta por dos fases: *percepción* y *razonamiento*. Además después de haber razonado habría que actuar adecuadamente para que la interacción prosiga, pero no es un tema relevante en el contexto de este trabajo.

Percepción

Lo que para nosotros serían los sentidos, en los robots lo podemos sustituir por sensores. Cámaras para la vista, micrófonos para el oído o sensores de presión para el tacto. Además de estos existen múltiples variantes más y con mayor o menor precisión en su tarea. Podríamos decir que esta fase de la interacción está bastante bien cubierta y de algún modo es muy semejante a la humana.

Estos sensores por si sólos no tienen ningún valor sino que detrás de todos esos datos capturados deben existir algoritmos que saquen conclusiones de todos ellos. Por ejemplo, en los fotogramas capturados por una cámara se puede realizar detección de objetos o personas a través de la Visión Artificial y el Machine Learning. O por ejemplo, extraer palabras o frases del audio capturado por un micrófono usando también Machine Learning.

Podríamos pensar que esto ya nos resuelve todos los problemas, pero no es así.

Hasta ahora sólo tenemos muchos datos sin ningún contexto, cosa que los humanos hacemos a través de la razón.

Razonamiento

Sin lugar a dudas es la más compleja y la que más investigación necesita. A día de hoy no se ha conseguido implementar un real razonamiento humano dentro de un robot, pero si que se utilizan diversos trucos que simulan ese *razonamiento*:

- *Contexto de una conversación.* La frase «No he visto ninguno» puede tener múltiples significados dependiendo del tema el cual se esté tratando en la conversación. Se podría estar expresando que no se ha visto ningún error en la carta que se escribiendo o que no se ha visto llegar el taxi se había pedido. Entonces los robots no se pueden dedicar a detectar únicamente frases sueltas y actuar ante ellas. Para ello existen modelos de lenguaje como GPT-3 entrenados a través de Machine Learning que consiguen simular que están entendiendo el diálogo, pero en realidad sólo están repitiendo conversaciones con las que han sido entrenados.

- *Atención.* Mediante reconocimiento facial (usando Visión Artificial y Machine Learning) el robot puede realizar un seguimiento con la mirada a la cara del sujeto con el que está interactuando. Esto por ejemplo, simularía que el robot está prestando atención a una conversación.

- *Compresión de la situación emocional.* A través de la detección de emociones o expresiones faciales (otra vez usando Visión Artificial y Machine Learning) del sujeto con el que se está interactuando, el robot puede actuar de una manera u otra simulando que está comprendiendo situación emocional. El sistema de detección de emociones desarrollado en este trabajo estaría presente en este apartado y el anterior de *Atención*.

Se ha comprobado que la Visión Artificial y el Machine Learning son temas clave en el desarrollo del sistema del presente trabajo. Ambos son introducidos en las próximas dos secciones.

1.3. Visión Artificial

Los seres humanos utilizamos nuestros ojos para, de alguna manera, comprender todo aquello que nos rodea. El objetivo de la visión artificial es trasladar esa misma

habilidad a una máquina, osea, que sea capaz de percibir información visual del entorno y actuar según la situación. Para ello se realiza lo que se conoce como procesamiento de imagen, este se puede dividir en las siguientes fases o etapas:

- *Digitalización.* Proceso de transformación que sufre una imagen analógica a otra digital para que pueda ser manipulada por un ordenador. Una máquina sólo sabe comprender números, por lo tanto la imagen estará representada como una matriz de números (píxeles).

- *Preprocesamiento.* En la etapa anterior es muy probable que las imágenes sufran degradaciones como pérdida de definición o aparición de ruido. Esta etapa intenta subsanarlas con técnicas como la reducción de ruido o realce del contraste.

- *Segmentación.* Extracción de información contenida en la imagen mediante la descomposición de la misma en regiones significativas. Por ejemplo determinar en una imagen que píxeles pertenecen a los objetos y cuáles al fondo.

- *Representación.* Tras realizar la segmentación se poseen píxeles en bruto. Se deberá elegir si se desean representar esos datos como el contorno de una región o como los puntos de dicha región. En eso consiste esta etapa.

- *Descripción.* Selección de características o descriptores de la representación elegida para permitir la posterior clasificación de los objetos. Por ejemplo la cantidad de huecos o el perímetro del contorno.

- *Reconocimiento.* Clasificación de los objetos de la imagen usando las características o descriptores obtenidos en la etapa anterior. A cada objeto se le asigna una etiqueta como *Persona* o *Planta*.

- *Interpretación.* Etapa final, la cual se encarga de dar significado a los objetos reconocidos. Por ejemplo localizar que objetos son dinámicos o estáticos, o detectar la posición en la que se encuentra un cuerpo.

Lo explicado y desarrollado anteriormente se puede enmarcar dentro de lo que se conoce como la *Visión Artificial Clásica*, enfocada en la utilización de algoritmos específicos para procesar imágenes y reconocer en ellas características básicas.

El auge de la Inteligencia Artificial (IA) y sobre todo el Machine Learning, está expandiendo exponencialmente las capacidades de la Visión Artificial. Sobre todo las Redes Neuronales Profundas o Deep Learning están aportado mucho valor en este

campo. Son técnicas muy potentes que permiten resultados mucho mejores que los ofrecidos por la visión clásica y además mucho más fáciles de implementar. En la siguiente sección se introducirá dicho campo, el Machine Learning.

1.4. Machine Learning

El Machine Learning o Aprendizaje Automático es una disciplina del campo de la Inteligencia Artificial que permite a un ordenador realizar tareas de manera automática sin previamente haber sido programados explícitamente para dichos casos. Según el tipo de aprendizaje que realicen los algoritmos podemos clasificar estos en tres grandes grupos:

- *Aprendizaje supervisado*
- *Aprendizaje no supervisado*
- *Aprendizaje por refuerzo*

Cada uno de los tipos de aprendizaje tiene determinadas características y diferentes aplicaciones finales que estudiaremos en la siguientes secciones.

1.4.1. Aprendizaje supervisado

Usado para resolver problemas conocidos. Se le proporciona al algoritmo un conjunto de datos de entrada y sus salidas correspondientes, entonces el algoritmo se dedica a *aprender* la relación entre las salidas y las entradas. Generará unos patrones a partir de los cuales realizará predicciones.

Utilizando un ejemplo más familiar, si queremos que nuestro algoritmo aprenda a detectar gatos lo que debemos hacer es proporcionarle imágenes de ejemplo con gatos debidamente etiquetados. Una vez que el algoritmo haya recibido toda esa información y la haya procesado adecuadamente, la próxima vez que vea datos similares sabrá clasificarlos como gatos.

Dentro del aprendizaje supervisado se diferencian dos grandes tipos:

- ***Regresión.*** Tiene como objetivo predecir la salida mediante una función que proporciona valores continuos. Por ejemplo predecir el precio de una vivienda a partir de su tamaño en metros cuadrados usando regresión lineal (Figura 1.6).

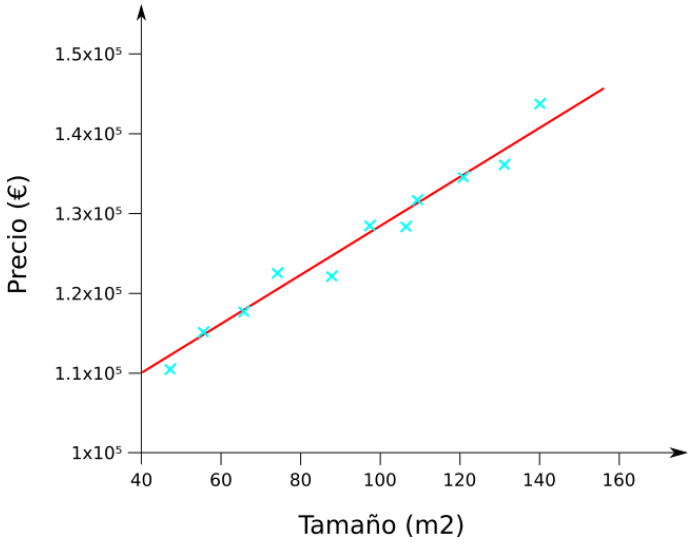

Figura 1.6: Ejemplo de regresión lineal.
Predicción del precio de la vivienda.

En el ejemplo de la Figura 1.6 los datos usados como entrada serían los tamaños de las viviendas y los datos de salida los precios. Las cruces azules representan estos valores de entrada y salida y la línea roja representa la relación obtenida entre ellos. A partir de esa función lineal, el algoritmo ya sería capaz de realizar cualquier predicción para una pareja de datos de tamaño y precio.

Además de la regresión lineal, que es el ejemplo más simple, existen otros tipos como la regresión logística o la regresión polinomial.

- ***Clasificación.*** Las salidas toman valores discretos en función de los valores de entrada. Si la salida únicamente posee dos valores discretos, entonces estamos ante una clasificación binaria. Si la salida puede tomar más de dos valores discretos, la clasificación será multiclase.

  Un ejemplo de clasificación binaria sería la predicción del tipo de casa (piso o chalet) en función de la distancia al centro de la ciudad y el tamaño del jardín (Figura 1.7).

  En el ejemplo de la Figura 1.7 los datos usados como entrada serían el tamaño del jardín y la distancia al centro y los datos de salida el tipo de casa. Se puede observar como el algoritmo de clasificación ha separado los datos en dos conjuntos diferenciados, cualquier dato nuevo que caiga en uno de esos conjuntos será clasificado con su etiqueta correspondiente. Es a eso a lo que se le llama predicción.

  Los algoritmos más utilizados para realizar clasificación son SVM (Support Vector

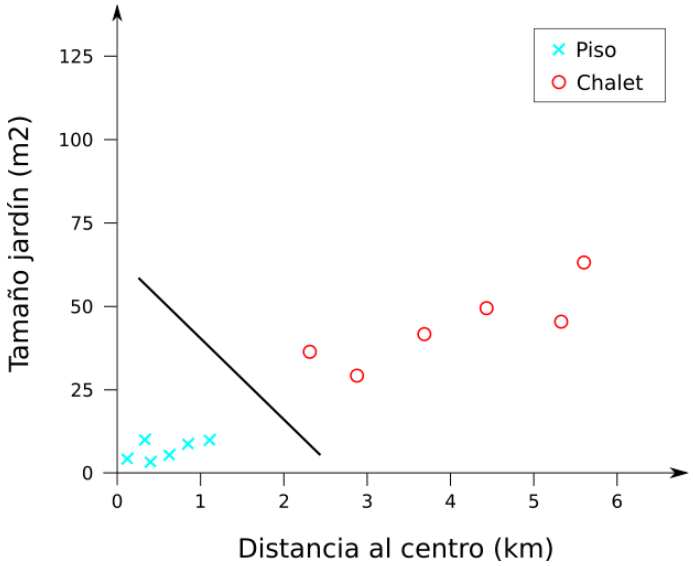

Figura 1.7: Ejemplo de clasificación binaria.
Predicción del tipo de casa.

Machine), KNN (K Nearest Neighbour), Árboles de decisión y Redes Neuronales (Convolucionales, Recurrentes, etc.).

1.4.2. Aprendizaje no supervisado

Únicamente se le proporciona al algoritmo un conjunto de datos de entrada y el propio algoritmo será el encargado de detectar patrones dentro de ese conjunto. A diferencia del aprendizaje supervisado, aquí no existe ningún etiquetado de los datos, por lo tanto la máquina únicamente separara los datos por patrones pero no tendrá el concepto de que son gatos o perros.

Un ejemplo, siguiendo con la temática de las viviendas, sería agrupar las casas en función de la distancia al centro y del tamaño del jardín (Figura 1.8). Esto se conoce como clustering o segmentación.

En el ejemplo de la Figura 1.8 los datos de entrada serían el tamaño del jardín y la distancia al centro. Además al algoritmo, en este tipo de aprendizaje, se le suele indicar en cuántas clases se desea que se clasifiquen los datos. También se puede no indicar esta información y dejarle total libertad a la máquina. En este último caso los científicos de datos tiene la posibilidad de aprender más sobre estos y puede encontrar patrones interesantes u ocultos que antes no eran visibles.

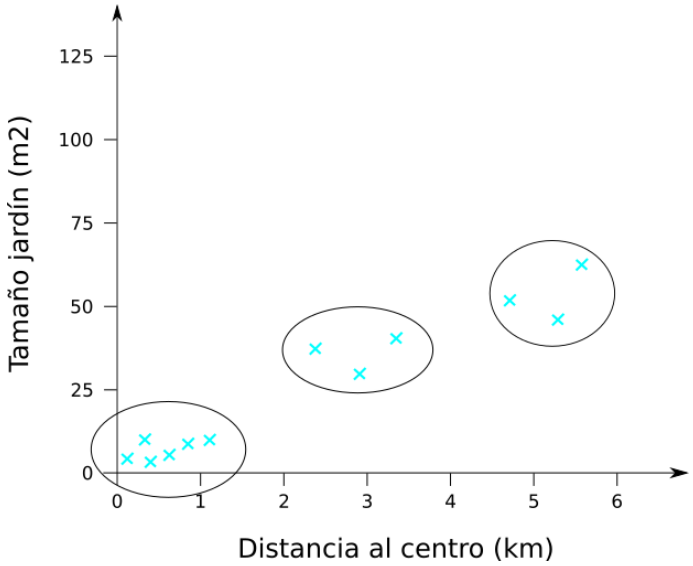

Figura 1.8: Ejemplo de clustering.
Agrupación de datos de casas.

1.4.3. Aprendizaje por refuerzo

No se le proporcionan datos de entrada ni de salida. El algoritmo aprende a desarrollar una tarea a partir de un esquema de recompensas y penalizaciones ante las decisiones que toma en cada una de las iteraciones. Ya no sólo se trata de clasificar unos datos en unas determinadas clases sino que tendremos muchos factores a la vez a los que prestar atención y actuar según la situación. Por eso este tipo de aprendizaje es sobre todo usado en robótica o videojuegos, ambos son máquinas o personajes actuando en un entorno cambiante.

A diferencia de los otros tipos de aprendizaje que se intenta reducir el error, aquí se intenta maximizar la recompensa. Los componentes del Aprendizaje Reforzado son:

- ***Agente.*** Modelo que se pretende entrenar para que aprendar a tomar decisiones.
- ***Ambiente.*** Entorno donde interactúa el agente, el cual contiene las reglas posibles en cada momento.

Además la relación que se realimenta entre los componentes mencionados anteriormente cuenta con los siguientes nexos:

- ***Acción.*** Posibles acciones que puede tomar el Agente en un momento determinado.

- ***Estado (del ambiente).*** Indicadores del ambiente de cómo están los elementos que lo componen en ese momento.
- ***Recompensas o penalizaciones.*** Por cada acción tomada por el Agente, se obtendrá un premio o una penalización que orientará al Agente en si lo está haciendo bien o mal.

Se podría concluir afirmando que es una forma de entrenamiento basada en la fuerza bruta. Si el objetivo es que un robot recorra una habitación esquivando obstáculos, se le deberá someter a choques, acelerones, frenazos... para hacerle aprender lo que está bien y lo que está mal. El algoritmo más usado es Q-Learning.

1.5. Sistemas embebidos

Un sistema embebido (también conocido como *empotrado*) es un sistema de computación diseñado para realizar funciones específicas, y cuyos componentes se encuentran integrados en una placa base. El procesamiento central del sistema se lleva a cabo gracias a un microcontrolador, es decir, un microprocesador que incluye además interfaces de entrada/salida, así como una memoria de tamaño reducido en el mismo chip [5].

Dependiendo del tipo de sistema embebido algunos pueden ser programados directamente en lenguaje ensamblador del microcontrolador o microprocesador u otros en lenguajes de más alto nivel como C++ o Python para desarrollo de aplicaciones.

Se comenzará definiendo lo que es un microprocesador y un microcontrolador. Es esencial entender ambos significados si se quiere saber en que consiste un sistema embebido.

1.5.1. Microprocesador y microcontrolador

Denominamos microprocesador al conjunto de elementos fusionados en un mismo circuito. Algunos de esos elementos la ALU (Unidad Aritmética Lógica) la cual se encarga de realizar las operaciones matemáticas o los registros que guardan los datos temporalmente. Normalmente conocemos a estos microprocesadores o procesadores como CPU (Unidad Central de Procesamiento) siendo el núcleo de nuestros ordenadores, esto es así ya que por sí sólos no tienen ningún uso, se integran junto a otros componentes que le dan el uso concreto final. Por ejemplo en un ordenador

se integra junto con el disco duro, la memoria RAM, la tarjeta gráfica, y demás componentes.

Aquí es donde entra en juego el término de microcontrolador, podemos entenderlo como un pequeño ordenador con capacidad limitada. Si antes hablábamos del ordenador convencional esto lo podríamos entender igual pero todo integrado en un único chip. Es por ello que tendrán capacidad de cómputo limitada y no se les podrá someter a tareas muy exigentes. Sin embargo esto les proporciona una gran ventaja, su bajo coste económico. Por esto último es por lo que son tan populares y está en auge la investigación dedicada a intentar transportar tareas (que en principio necesitan de gran cómputo) a estos sistemas embebidos realizando las simplificaciones necesarias.

1.5.2. Sistemas embebidos populares

Existen múltiples fabricantes de sistemas empotrados, en este capítulo de introducción se nombraran los dos más grandes actualmente: Arduino[3] y Raspberry[4]. Ambos proporcionan microcontroladores aunque Raspberry es más conocida por sus SBC (Single Board Computer, en inglés).

Arduino. Fabricante especializado en la venta de microcontroladores. Posee modelos como los Arduino UNO R3 (Figura 1.9) o Arduino Nano. Son microcontroladores integrados en el mismo chip con todos los componentes necesarios para su correcto funcionamiento (resistencias, condensadores, pines para conectar elementos, etc). Además de esto mencionado, la ventaja que nos proporciona este tipo de placas y Arduino es que a través de su entorno de desarrollo (Arduino IDE[5]) tenemos la oportunidad de cargar código en los microcontroladores sin realizar métodos de "flasheado" y compilación tediosos que si requieren otro tipo de microcontroladores.

Raspberry. Tiene a la venta también microcontroladores como la Raspberry Pi Pico pero su producto principal son los SBC, su último modelo la Raspberry Pi 4b (Figura 1.10). El concepto es muy parecido al de Arduino pero con características más robustas, no sólo se trata de un microcontrolador simple, es un ordenador completo con su propio sistema operativo en una sola placa. Este mencionado sistema operativo es Raspberry Pi OS basado en Debian.

[3] `https://www.arduino.cc/`
[4] `https://www.raspberrypi.org/`
[5] `https://www.arduino.cc/en/software`

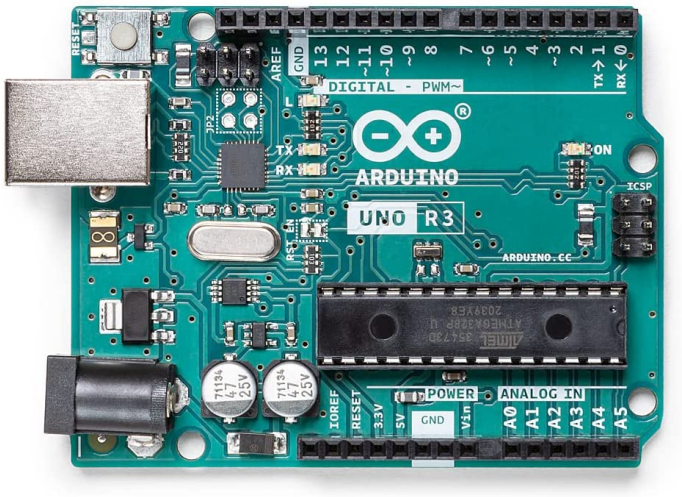

Figura 1.9: Microcontrolador Arduino UNO R3.

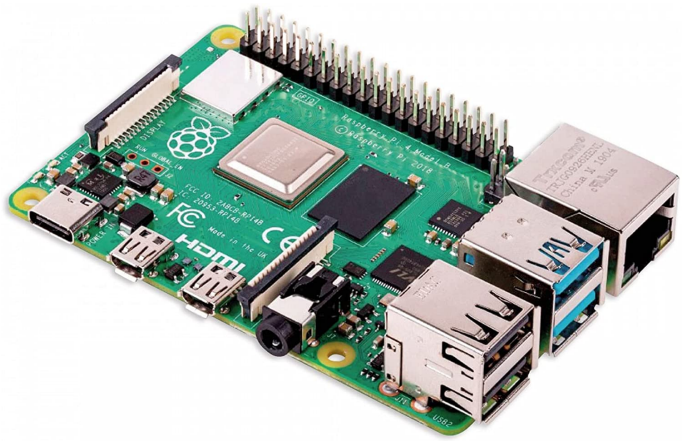

Figura 1.10: Raspberry Pi 4b.

En el siguiente capítulo (número 2) se realizará una descripción del problema a desarrollar y la metodología y el plan de trabajo que se ha llevado a cabo. Prosiguiendo con el capítulo 3, el autor expondrá las herramientas hardware y software utilizadas. En el capítulo 4 se mostrará todo el desarrollo del trabajo y los experimentos realizados. Por último, en el capítulo 5 se podrán encontrar las conclusiones finales.

Capítulo 2
Objetivos

Quizás algún fragmento de libro inspirador...

Autor, *Título*

Una vez presentado el contexto general en el cual se enmarca nuestro trabajo de fin de grado, se procederá a realizar una descripción del problema planteando los objetivos finales de este, requisitos, metodología y plan de trabajo para llegar a resolverlo.

2.1. Descripción del problema

El objetivo principal de este TFG es desarrollar una herramienta de reconocimiento de emociones que sea capaz de funcionar en tiempo real en un sistema robótico de bajo coste. Para lograr dicha meta, se ha dividido el problema en estos subobjetivos:

1. Estudiar cuál será la técnica más óptima en cuanto al reconocimiento de emociones. Debe ser una técnica liviana que no consuma muchos recursos para conseguir un alto valor de FPS (fotogramas por segundo) final en nuestro sistema.

2. Tras decidir que la extracción de puntos característicos faciales y el posterior entrenamiento con el tratamiento de ellos era la técnica más óptima y liviana para nuestro hardware, el siguiente paso es estudiar qué método para extraer esos puntos faciales es el más rápido y preciso.

3. Crear un dataset de valor con los puntos característicos faciales. Se debe hacer un correcto tratamiento de los datos para conseguir un resultado preciso en el posterior entrenamiento.

4. Realizar el entrenamiento con varios algoritmos de Machine Learning de clasificación. Estudiar el rendimiento y precisión de cada uno de ellos.

5. Integrar nuestro sistema en ROS para facilitar su uso en un sistema robótico.

2.2. Requisitos

El requisito principal del proyecto es que el sistema funcione a una tasa de FPS que permitan usarlo en tiempo real. La herramienta está enfocada en ayudar en la interacción entre personas y robots, por lo tanto debe poder ofrecer información lo más rápido posible para actuar en el momento preciso.

Otro requisito es que todo el software debe correr en la Raspberry Pi 4b, ya que es el sistema de bajo coste escogido para realizar el trabajo. Además todo deberá funcionar bajo el sistema operativo Raspberry Pi OS porque es el más optimizado actualmente para nuestro hardware y el que por lo tanto nos ofrecerá mayor rendimiento (que es nuestro requisito principal).

Por último, al ser una herramienta para un sistema robótico, es muy importante conseguir la mayor robustez posible.

2.3. Metodología

Se ha seguido un protocolo de reuniones semanales con el tutor de TFG a través de la plataforma Teams para comentar los avances y recibir realimentación, además de proponer cada semana las actividades a realizar.

Se ha usado un repositorio[1] de Github en el cual se ha ido subiendo todo el código del desarrollo del sistema de este trabajo. Además, este repositorio posee una Wiki[2] que contiene explicaciones semanales de todo lo llevado a cabo durante estos meses de trabajo.

La herramienta final de ROS se puede encontrar en un repositorio[3] de GitHub a parte. Se ha hecho de esta manera para que esté disponible para toda la comunidad de ROS y se la puedan descargar e instalar directamente.

2.4. Plan de trabajo

El desarrollo del TFG ha comprendido nueve meses de trabajo. Se comenzó en Octubre de 2021 y se ha terminado en Junio de 2022. Durante estos meses la planificación ha sido la siguiente:

[1]Repositorio TFG: `https://github.com/jmvega/tfg-jmartinez`
[2]Wiki: `https://github.com/jmvega/tfg-jmartinez/wiki`
[3]Sistema final en ROS: `https://github.com/jmrtzma/emotion_detection_ros`

1. *Etapa de investigación y entrenamiento.* Fase inicial en la que se realizaron diferentes lecturas y pruebas con pequeños scripts de código para descubrir cual sería el tema de TFG a desarrollar. Una vez escogido el tema de TFG se realizaron lecturas sobre otros proyectos similares.

2. *Estudio de técnicas de reconocimiento de emociones.* Investigación para descubrir cuáles eran las técnicas más usadas para realizar esta labor. Se estudió cuál podía ser la más liviana y precisa para nuestra plataforma (Raspberry Pi 4b + Raspberry Pi OS + Raspberry Pi Camera Module V2).

3. *Creación del dataset.* Tratamiento de los datos para generar un dataset que nos proporcione entrenamientos precisos. Se realizaron diversos estudios hasta encontrar el dataset que mejores resultados nos proporcionaba.

4. *Entrenamiento de los modelos.* Fase de entrenamiento usando los algoritmos SVM, KNN y una red neuronal multicapa. Se buscó cual era la técnica más óptima para llevar a cabo los entrenamientos y además se realizó un estudio del rendimiento y precisión de los algoritmos.

5. *Búsqueda de robustez.* El sistema ya estaba desarrollado pero no poseía la suficiente robustez como para ser usado en un robot. Esta fase se encargó de aumentar dicha robustez a nuestro sistema consiguiendo más fiabilidad en la detección de las emociones en un entorno real y práctico.

6. *Integración del sistema en ROS.* Se buscó la forma de instalar una versión de ROS en Raspberry Pi OS y se creó el paquete que porta la herramienta desarrollada en este TFG.

Capítulo 3
Plataforma de desarrollo

Quizás algún fragmento de libro inspirador...

Autor, *Título*

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo. En este capítulo, explica qué has usado a nivel hardware y software para poder desarrollar tu trabajo: librerías, sistemas operativos, plataformas, entornos de desarrollo, etc.

3.1. Raspberry Pi 4 Model B

La Raspberry Pi 4 Model B (Figura 3.1) es la plataforma hardware de bajo coste escogida para este proyecto. Debido a la gran comunidad de desarrolladores y usuarios que posee, además de las especificaciones ofrecidas (Cuadro 3.1) por su escaso precio (65,44 €[1]), es la plataforma más usada por la mayoría del público.

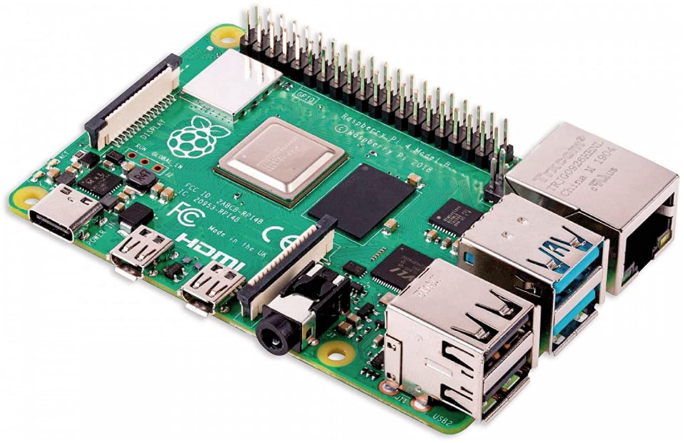

Figura 3.1: Raspberry Pi 4b.

[1]Distribuidor oficial de Raspberry: `https://www.kubii.es/40-raspberry-pi-3-2-b`

Sus características más atractivas —entre otras— son su bajo consumo energético, su pequeño tamaño y por lo tanto mínimo peso, su alta conectividad y puertos (red WIFI, Bluetooth, Ethernet, USB2 y USB3, HDMI, etc.) y la gran fluidez que posee su sistema operativo (Sección 3.1.1). Todo esto la convierte en una auténtica joya y son cada vez más usuarios los que la utilizan en diversos proyectos. Podemos encontrarla —por ejemplo— como centro doméstico inteligente para controlar la domótica de una casa o incluso en sectores más profesionales formando parte de la arquitectura de algunos robots. Esto último es lo más interesante, para nosotros, dentro de los múltiples usos que tiene una Raspberry.

Procesador	Broadcom BCM2711 (4 núcleos Cortex-A72 (ARM v8), 64-bit, 1.5Ghz)
Tarjeta gráfica	Broadcom VideoCore VI (integrada en el procesador)
Memoria RAM	4 GB LPDDR4-3200 SDRAM
Conexión	WIFI 2.4 GHz y 5 GHz Bluetooth 5.0/BLE Gigabit Ethernet
Puertos	2 x micro-HDMI (4K 60 Hz) MIPI Display Serial Interface MIPI Camera Serial Interface Jack Audio/Video Slot para micro-SD
Alimentación	5V por USB-C (3A mínimo) 5V por GPIO (3A mínimo)

Cuadro 3.1: Especificaciones técnicas de la Raspberry Pi 4 Model B.

Muchos de los robots son de tamaño reducido y no tienen el espacio suficiente como para acoplar una gran estación de procesamiento, por lo tanto en esos casos es muy común usar algún modelo de Raspberry como unidad central. Incluso en robots grandes se suelen utilizar también para realizar el control de zonas concretas, por ejemplo de los ojos de un humanoide. Por lo tanto, nuestro sistema de detección de emociones corriendo en la Raspberry Pi 4 Model B puede servir de gran ayuda a la hora de construir uno de estos robots comentados anteriormente que sólo tienen la capacidad de albergar una placa de tamaño reducido, o que simplemente los desarrolladores de dicho robot quieren ahorrar dinero en costes.

3.1.1. Raspberry Pi OS

Raspberry Pi OS (Figura 3.2) es el sistema operativo oficial para Raspberry. Es un sistema operativo gratuito basado en Debian optimizado específicamente para el hardware de la Raspberry Pi, por lo tanto es el que mayor rendimiento nos ofrecerá frente a otros como Ubuntu (que también puede ser instalado). Además, está en constante desarrollo y continuamente se está mejorando su estabilidad y funcionalidad. Es por todo ello que será el sistema operativo elegido en nuestro proyecto, sobre todo por su alto rendimiento (algo esencial para impulsar el desempeño de nuestro sistema de detección de emociones).

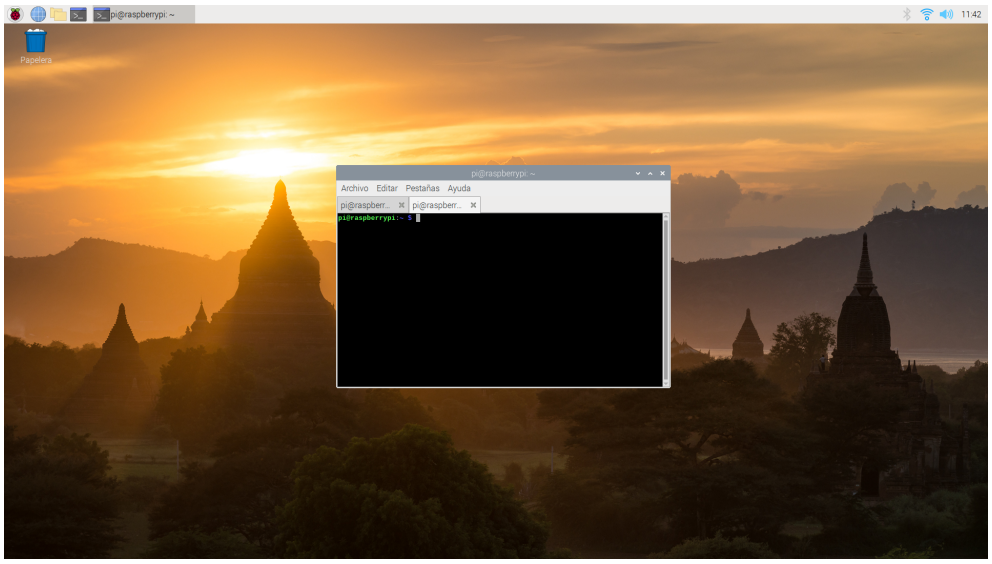

Figura 3.2: Captura de pantalla de Raspberry Pi OS.

La versión de Raspberry Pi OS escogida ha sido Raspberry Pi OS Legacy (Cuadro 3.2). Se ha elegido dicha versión porque, de todas las disponibles, ha sido la única en la que se ha conseguido instalar una versión de ROS/ROS2 (en concreto, ROS Noetic). Además, aunque las versiones de Raspberry Pi OS de 64-bit ofrecían más rendimiento, todavía no estaban maduras y no ofrecían total compatibilidad con todas las librerías usadas en el presente trabajo.

Fecha de lanzamiento	4 de Abril de 2022
Sistema	32-bit
Versión del Kernel	5.10
Versión de Debian	10 (buster)

Cuadro 3.2: Especificaciones de Raspberry Pi OS Legacy.

3.1.2. Raspberry Pi Camera Module V2.1

La Raspberry Pi Camera es la cámara oficial desarrollada por Raspberry para ser utilizada en sus placas. Para este trabajo, se ha hecho uso de la versión 2.1 (Cuadro 3.3). Es una cámara de alta definición (3280x2464) que se conecta a cualquier Raspberry Pi compatible a través de una interfaz de bus CSI-2. Además de vídeo de alta calidad, ofrece una reducción de la contaminación de la imagen (ruido o manchas).

Sensor de imagen	Sony IMX 219 PQ CMOS
Resolución de imagen	3280x2464 (8-megapixeles)
Resolución de vídeo	1080p 30fps 720p 60fps
Conexión	Cable plano de 15 pines, MIPI Camera Serial Interfaze (CSI-2)
Peso	3g
Dimensiones	23.86 x 25 x 9 mm

Cuadro 3.3: Especificaciones de Raspberry Pi Camera Module V2.1.

Se ha decidido escoger la Raspberry Pi Camera Module V2.1 en vez de una versión convencional de WebCam (USB) debido a las siguientes ventajas:

- *Mayor framerate.* Gracias a que la Raspberry Pi 4 Model B tiene un puerto dedicado para conectar la Raspberry Pi Camera, es posible conseguir una gran velocidad de fotogramas. Esto es debido a que esa conexión especial permite que la codificación vaya dirigida directamente a la GPU y sólo haya un pequeño impacto en la CPU, dejándola libre para otros usos. En cambio, una WebCam conectada por USB utiliza directamente la CPU, y mover datos a través de un USB es bastante costoso para un sistema de recursos limitados.

- *Mayor calidad de imagen.* Es cierto que también existen WebCam con una calidad de imagen muy buena, pero su precio es elevado. Por lo tanto, la Raspberry Pi Camera acaba ganando en cuanto a calidad de imagen si realizamos la comparativa en el mismo rango de precio (29,14 €[2]). Sin embargo, este no es un apartado muy relevante porque finalmente en el sistema de detección de emociones no hacemos uso de la máxima resolución para aumentar el rendimiento.

- *Menor tamaño.* El tamaño tan compacto de la Raspberry Pi Camera es uno de sus mayores atractivos y es por eso que es también una gran ventaja frente a las

[2]Distribuidor oficial de Raspberry: `https://www.kubii.es/318-camaras-sensores`

WebCam. De cara a instalar estos pequeños ordenadores con cámara en un robot es esencial que ocupen el menor espacio posible, además de que su peso sea muy reducido.

3.2. Python

Python es un lenguaje de programación interpretado (se ejecuta sin necesidad de ser compilado) y de tipado dinámico (las variables se comprueban en tiempo de ejecución). Es de licencia totalmente libre y soporta programación orientada a objetos. Se caracteriza por hacer uso de una sintaxis muy legible en la que es obligatoria una correcta tabulación.

Se ha escogido Python (versión 3.7.3) como lenguaje de programación para este proyecto debido a los dos siguientes motivos:

1. El sistema desarrollado usa algoritmos de Machine Learning para detectar las emociones faciales y Python es el lenguaje de programación rey en ese campo, posee múltiples librerías como Pandas, TensorFlow, Keras, Scikit-learn... que brindan un soporte excepcional para cualquier tarea relacionada con el aprendizaje automático.

2. La librería que da soporte oficial a la Raspberry Pi Camera (Sección 3.1.2) únicamente se puede usar en Python. Se trata del paquete *picamera*.

Capítulo 4
Diseño

Quizás algún fragmento de libro inspirador...

Autor, *Título*

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo. En este capítulo (y quizás alguno más) es donde, por fin, describes detalladamente qué has hecho y qué experimentos has llevado a cabo para validar tus desarrollos.

4.1. Snippets

Puede resultar interesante, para clarificar la descripción, mostrar fragmentos de código (o *snippets*) ilustrativos. En el Código 4.1 vemos un ejemplo escrito en `C++`.

```
void Memory::hypothesizeParallelograms () {
  for(it1 = this->controller->segmentMemory.begin(); it1++) {
    squareFound = false; it2 = it1; it2++;
    while ((it2 != this->controller->segmentMemory.end()) && (!squareFound))
        {
      if (geometry::haveACommonVertex((*it1),(*it2),&square)) {
        dist1 = geometry::distanceBetweenPoints3D ((*it1).start, (*it1).end);
        dist2 = geometry::distanceBetweenPoints3D ((*it2).start, (*it2).end);
      }
    // [...]
```

Código 4.1: Función para buscar elementos 3D en la imagen

En el Código 4.2 vemos un ejemplo escrito en `Python`.

4.2. Verbatim

Para mencionar identificadores usados en el código —como nombres de funciones o variables— en el texto, usa el entorno literal o verbatim

```
def mostrarValores():
    print (w1.get(), w2.get())

master = Tk()
w1 = Scale(master, from_=0, to=42)
w1.pack()
w2 = Scale(master, from_=0, to=200, orient=HORIZONTAL)
w2.pack()
Button(master, text='Show', command=mostrarValores).pack()

mainloop()
```

Código 4.2: Cómo usar un Slider

`hypothesizeParallelograms()`. También se puede usar este entorno para varias líneas, como se ve a continuación:

```
void Memory::hypothesizeParallelograms () {
  // add your code here
}
```

4.3. Ecuaciones

Si necesitas insertar alguna ecuación, puedes hacerlo. Al igual que las figuras, no te olvides de referenciarlas. A continuación se exponen algunas ecuaciones de ejemplo: Ecuación 4.1 y Ecuación 4.2.

$$H = 1 - \frac{\sum_{i=0}^{N} \frac{(\frac{d_{js}+d_{je}}{2})}{N}}{M} \tag{4.1}$$

Ecuación 4.1: Ejemplo de ecuación con fracciones

$$v(entrada) = \begin{cases} 0 & \text{if} \quad \epsilon_t < 0{,}1 \\ K_p \cdot (T_t - T) & \text{if} \quad 0{,}1 \leq \epsilon_t < M_t \\ K_p \cdot M_t & \text{if} \quad M_t < \epsilon_t \end{cases} \tag{4.2}$$

Ecuación 4.2: Ejemplo de ecuación con array y letras y símbolos especiales

4.4. Tablas o cuadros

Si necesitas insertar una tabla, hazlo dígnamente usando las propias tablas de LaTeX, no usando pantallazos e insertándolas como figuras... En el Cuadro 4.1 vemos

un ejemplo.

Parámetros	**Valores**
Tipo de sensor	Sony IMX219PQ[7] CMOS 8-Mpx
Tamaño del sensor	3.674 x 2.760 mm (1/4"format)
Número de pixels	3280 x 2464 (active pixels)
Tamaño de pixel	1.12 x 1.12 um
Lente	f=3.04 mm, f/2.0
Ángulo de visión	62.2 x 48.8 degrees
Lente SLR equivalente	29 mm

Cuadro 4.1: Parámetros intrínsecos de la cámara

Capítulo 5
Conclusiones

Quizás algún fragmento de libro inspirador...

Autor, *Título*

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo, que básicamente será una recapitulación de los problemas que has abordado, las soluciones que has prouesto, así como los experimentos llevados a cabo para validarlos. Y con esto, cierras la memoria.

5.1. Conclusiones

Enumera los objetivos y cómo los has cumplido.

Enumera también los requisitos implícitos en la consecución de esos objetivos, y cómo se han satisfecho.

No olvides dedicar un par de párrafos para hacer un balance global de qué has conseguido, y por qué es un avance respecto a lo que tenías inicialmente. Haz mención expresa de alguna limitación o peculiaridad de tu sistema y por qué es así. Y también, qué has aprendido desarrollando este trabajo.

Por último, añade otro par de párrafos de líneas futuras; esto es, cómo se puede continuar tu trabajo para abarcar una solución más amplia, o qué otras ramas de la investigación podrían seguirse partiendo de este trabajo, o cómo se podría mejorar para conseguir una aplicación real de este desarrollo (si es que no se ha llegado a conseguir).

5.2. Corrector ortográfico

Una vez tengas todo, no olvides pasar el corrector ortográfico de LaTeXa todos tus ficheros *.tex*. En `Windows`, el propio editor `TeXworks` incluye el corrector. En `Linux`, usa `aspell` ejecutando el siguiente comando en tu terminal:

```
aspell --lang=es --mode=tex check capitulo1.tex
```

Bibliografía

[1] Neha Baranwal. On human robot interaction using multiple modes. 11 2018.

[2] Medical Expo. Sistema de resuperficialización de rodilla rio. https://www.medicalexpo.es/prod/mako-surgical-corp/product-81512-517354.html.

[3] HM Hospitales. Cirugía robótica. https://www.hmsanchinarro.com/especialidades/programas-medicos/robot-da-vinci.

[4] Iberdrola. Descubre los principales beneficios del 'machine learning'. https://www.iberdrola.com/innovacion/machine-learning-aprendizaje-automatico.

[5] Sebastián Luchetti. Sistemas embebidos y sus características — conceptos fundamentales, 2021. https://tech.tribalyte.eu/blog-sistema-embebido-caracteristicas.

[6] Hans Moravec. *The future of robot and human intelligence*. Harvard University Press, 1988.

[7] Jonathan Mumm and Bilge Mutlu. Human-robot proxemics: Physical and psychological distancing in human-robot interaction. In *Proceedings of the 6th International Conference on Human-Robot Interaction*, HRI '11, page 331–338, New York, NY, USA, 2011. Association for Computing Machinery.

[8] Xataka. https://www.xataka.com.