

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR



TRABAJO DE FIN DE GRADO
GRADO EN INGENIERÍA DE SISTEMAS DE COMUNICACIONES
CONTROL CENTRALIZADO DE FLOTAS DE ROBOTS

AUTOR: ADRIÁN JIMÉNEZ CÁMARA

TUTOR: JAVIER V. GÓMEZ GONZÁLEZ

SEPTIEMBRE DE 2012

Índice

1.	Introducción.....	1
1.1	Motivación y objetivos	1
1.2	Estructura del documento	1
2.	Planteamiento del problema: análisis del estado del arte, requisitos y restricciones.....	2
2.1	Introducción.....	2
2.2	Análisis del estado del arte	3
2.3	Requisitos y restricciones del sistema.....	6
3.	Base teórica en la que se basa el trabajo	7
3.1	Introducción.....	7
3.2	Visión	7
3.2.1	¿Qué es la visión artificial?.....	7
3.2.2	La imagen digital.....	8
3.2.3	El color	10
3.2.4	Segmentación y etiquetado	13
3.2.5	Modelo Pinhole	14
3.3	Robótica.....	15
3.3.1	La estructura	16
3.3.2	Electrónica.....	16
3.3.3	Servomotores	17
4.	Diseño de la solución técnica	19
4.1	Introducción	19
4.2	Visión	21
4.2.1	Herramientas.....	21
4.2.1.1	Hardware	21
4.2.1.2	Software	23
4.2.2	El algoritmo	23
4.2.2.1	Calibración	23
4.2.2.2	Segmentación y etiquetado.....	26
4.2.2.3	Mapa del entorno de trabajo	28
4.3	Robótica.....	29
4.3.1	Diseño	29

4.3.2 Electrónica.....	32
4.3.2.1 Placa de Control.....	33
4.3.2.2 Servos.....	34
4.3.2.3 Módulos de radiofrecuencia XBEE.....	36
4.3.3 El programa.....	38
4.3.4 El robot final.....	42
5. Resultados y evaluación.....	43
5.1 Introducción.....	43
5.2 Condiciones y entorno de trabajo.....	43
5.3 Calibración, segmentación y etiquetado.....	49
6. Presupuesto y planificación del proyecto.....	56
7. Conclusiones y aplicaciones futuras.....	57
7.1 Introducción.....	57
7.2 Objetivos conseguidos.....	57
7.3 Conclusiones.....	57
7.4 Aplicaciones futuras.....	58
Referencias.....	59
Anexo I (Código).....	60
Anexo II (Fuentes Adicionales).....	60

Índice de figuras

Figura 2.1: Recreación de una flota de robots en misión de exploración planetaria	2
Figura 2.2: Robots con funciones cooperativas para jugar al fútbol	3
Figura 2.3: Robot humanoide Rh-1	5
Figura 3.1: Proceso de adquisición de imágenes del ojo humano.....	8
Figura 3.2: Representación de una imagen digital	8
Figura 3.3: Imagen binaria, en escala de grises y en color	9
Figura 3.4: Espectro electromagnético	10
Figura 3.5: Colores primarios y su combinación.....	10
Figura 3.6: Cubo RGB	11
Figura 3.7: Cono HSV.....	12
Figura 3.8: Etiquetado y cuenta de objetos	14
Figura 3.9: Modelo Pinhole	15
Figura 3.10: Servomotor.....	18
Figura 3.11: Control básico de un servo.....	18
Figura 4.1: Logitech Webcam C210	21
Figura 4.2: Lámpara LED apagada.....	22
Figura 4.3: Lámpara LED encendida.....	22
Figura 4.4: Calibración para el color azul	25
Figura 4.5: Calibración para el color verde.....	26
Figura 4.6: Imagen original con resultado sobreimpresionado de la segmentación y etiquetado	27
Figura 4.7: Segmentación del color verde individualmente.....	27
Figura 4.8: Segmentación del color azul individualmente	27
Figura 4.9: Simulación de etiquetas correspondientes a dos robots.....	28
Figura 4.10: Mapa del entorno de trabajo para ver la dirección y sentido según los cuales están orientados los robots.	28
Figura 4.11: Representación del espacio ocupado por los robots en el entorno de trabajo.	28
Figura 4.12: Chasis del robot	30
Figura 4.13: Rueda del robot	30
Figura 4.14: Impresora MADRE	31
Figura 4.15: Impresora PADRE.....	32
Figura 4.16: Arduino UNO (Parte delantera).....	34
Figura 4.17: Arduino UNO (Parte trasera).....	34
Figura 4.18: Partes de un servo	35
Figura 4.19: Módulo XBee de MaxStream	36
Figura 4.20: XBee Shield montado sobre la placa Arduino	37
Figura 4.21: Interfaz de programación Arduino	38
Figura 4.22: Robots	42
Figura 4.23: Vista lateral	42
Figura 4.24 Vista desde abajo.....	42
Figura 5.1: Etiquetas de distintos colores	43

Figura 5.2: Detección de etiquetas iniciales.....	44
Figura 5.3: Mapa virtual representando dirección y sentido de los robots con las etiquetas iniciales.....	44
Figura 5.4: Mapa virtual representando el espacio ocupado por los robots con las etiquetas iniciales.....	44
Figura 5.5: Etiquetas finales	45
Figura 5.6 Mapas resultado de aplicar el algoritmo de visión.....	46
Figura 5.7: Coordenadas del centroide de cada robot.....	47
Figura 5.8: Mapa con centroide para la comprobación	47
Figura 5.9: Comprobación distancia horizontal	48
Figura 5.10: Comprobación distancia vertical	48
Figura 5.11: Calibración con baja iluminación.....	49
Figura 5.12: Calibración con iluminación natural (12 am)	50
Figura 5.13: Calibración con iluminación natural (12 am) + iluminación artificial.....	50
Figura 5.14: Entorno de trabajo con baja iluminación y resultados del etiquetado sobreimpresionados.	51
Figura 5.15: Segmentación del color azul	52
Figura 5.16: Segmentación del color verde.....	52
Figura 5.17: Frame 1 de secuencia de movimiento y resultados	52
Figura 5.18: Frame 2 de secuencia de movimiento y resultados	52
Figura 5.19: Frame 3 de secuencia de movimiento y resultados	53
Figura 5.20: Tiempo de procesado para un entorno experimental con dos etiquetas estáticas	53
Figura 5.21: Tiempo de procesado para un entorno experimental con una etiqueta dinámica y otra estática.....	54
Figura 5.22: Tiempo de procesado para un entorno experimental con dos etiquetas dinámicas	54

1. Introducción

1.1 Motivación y objetivos

Actualmente, en el mundo de la robótica existen algoritmos de planificación de trayectorias muy robustos y potentes. Sin embargo, la aplicación experimental de éstos no es trivial y conlleva un gran esfuerzo. Sistemas de robots con infinidad de sensores hacen que la complejidad sea aún mayor ya que los datos proporcionados por estos sensores influyen en la toma de decisiones y en la planificación. Esto deriva en que la experimentación requiera muchas horas de trabajo y muchas veces lidiar con gran cantidad de problemas que no corresponden al campo de investigación.

Para simplificar el problema de la experimentación, se considera la arquitectura de un sistema de control centralizado, capaz de analizar en tiempo real el entorno de trabajo y gestionar las órdenes de los múltiples robots que operan en él. El problema inicial se verá reducido a un sistema en el que el robot ve el resto de objetos como obstáculos físicos ya sean de naturaleza pasiva como puede ser una pared o de naturaleza activa y móvil como pueden ser otros robots en el entorno de trabajo. Conseguir resultados en este entorno cerrado y simplificado permitirá la futura aplicación en sistemas de mayor complejidad.

1.2 Estructura del documento

La presente memoria presenta los diferentes aspectos que conforman el proyecto. El documento se compone de las siguientes secciones:

El capítulo 2 muestra una ligera descripción del Estado del Arte en materia de gestión de flotas de robots para realizar trabajos cooperativos. Esto es, en qué consisten las flotas de robots, por qué se usan y cuáles han sido las diferentes soluciones que se han llevado a cabo en la historia para la gestión de las mismas así como los pros y contras de cada una de estas soluciones. Asimismo se detallan los requisitos que impondremos al sistema a desarrollar así como las limitaciones que tendrá el mismo. En el capítulo 3 se detallan las bases teóricas en las que se fundamenta el trabajo y que creemos necesarias explicar para la correcta comprensión del mismo. El capítulo 4 recoge el diseño de la solución técnica, esto es, una descripción detallada del sistema así como información relevante acerca de diferentes aspectos del mismo tanto en la parte de visión artificial como en la parte de robótica. Los apartados 5 y 6 muestran como se ha llevado a cabo el proyecto y las pruebas efectuadas (así como resultados y consideraciones). Finalmente en el capítulo 7 se exponen las conclusiones extraídas, las líneas futuras de trabajo y las posibles repercusiones que puede tener el sistema implementado en el campo de la robótica y la visión artificial.

2. Planteamiento del problema: análisis del estado del arte, requisitos y restricciones.

2.1 Introducción

La robótica móvil surgió como una herramienta para explorar entornos inaccesibles al ser humano o bien por su lejanía o bien por su coste o peligrosidad así como la realización de tareas en estos entornos de trabajo. En ocasiones las tareas a realizar son muy laboriosas y únicamente se pueden llevar a cabo a través de estos dispositivos.

A la hora de la experimentación y la aplicación en el área de trabajo raramente nos encontramos con un único robot. Agrupaciones o flotas de robots son las distribuciones más usadas para la exploración espacial, conflictos bélicos, misiones de rescate o tomas de datos en entornos inexplorados. En la figura 2.1 podemos ver un ejemplo de ello, tradicionalmente las misiones de exploración espacial eran llevadas a cabo por robots teleoperados directamente por ingenieros en la Tierra. En la figura podemos ver una recreación de cómo sería la exploración del mismo terreno por un equipo de robots autónomos que además de cubrir un mayor terreno se comunican únicamente con un satélite que será el encargado de mandar la información al centro de control en la Tierra.

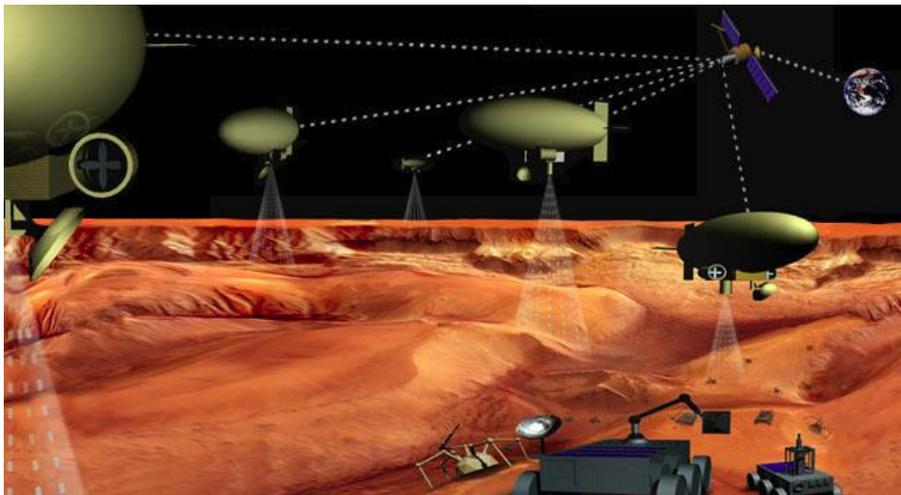


Figura 2.1: Recreación de una flota de robots en misión de exploración planetaria

Un ejemplo de la aplicación de flotas de robots para exploración de terreno lo podemos encontrar en el trabajo de Bouraqadi [1].

Uno de los objetivos primarios al construir equipos o flotas de robots móviles es sintetizar el comportamiento cooperativo de los mismos. Con comportamientos cooperativos de robots móviles se logran misiones que a priori pueden ser muy difíciles o imposibles de realizar por un solo robot. Además se intenta mejorar la robustez y la eficacia en la ejecución de las tareas.

2.2 Análisis del estado del arte

Históricamente dos soluciones son propuestas para el control de flotas de robots que tienen que trabajar en un entorno común a todos ellos.

- Control centralizado. Una unidad central, en este caso un ordenador es el encargado de controlar, supervisar y establecer las acciones de los robots de la flota. Todas las funciones de planificación y decisión se realizan en un solo centro de control. Los distintos robots móviles que componen la flota incluirán, aparte de sus propios sensores, los medios de comunicación necesarios para intercambiar los datos con el centro de control.

La ventaja principal de un control centralizado es que todos los movimientos y conflictos entre los distintos robots móviles son resueltos fácilmente en el ordenador central. La función del mismo es la asignación de tareas del sistema y la transmisión del flujo de información a los robots. De la misma manera, esto supone también un inconveniente, ya que si hay algún fallo en el centro de control el sistema entero se detiene y queda inutilizado.

En la figura 2.2 podemos ver la utilidad de un control centralizado en el que un ordenador central es capaz de organizar un equipo de robots que juegan al fútbol. Al ser un sistema en tiempo real el ordenador puede diseñar estrategias cooperativas para mejorar el juego en equipo, de vital importancia en los deportes de grupo [2].



Figura 2.2: Robots con funciones cooperativas para jugar al fútbol

- Robots autónomos: los robots autónomos son entidades físicas capaces de percibir características de un entorno y que posteriormente actúan sobre el mismo en base a dichas percepciones, sin supervisión directa de otros agentes. Por lo general se trata de robots grandes, con múltiples sensores y con sistemas de localización y planificación propios.

Uno de los mayores retos en la investigación actual en robótica es el desarrollo de técnicas para la navegación autónoma en entornos reales. La gran cantidad de incertidumbre inherente a los entornos reales hace que el problema se complique. Generalmente los sistemas de navegación de robots autónomos están basados en lógica difusa o heurística donde los algoritmos de planificación de trayectorias se basan en aproximar estadísticamente en vez de buscar la exactitud [3].

Esta solución por lo general conlleva una gran cantidad de toma de decisiones ya que cada robot como ente autónomo tiene que procesar toda la información proveniente de sus sensores para luego planificar los movimientos, las acciones, etc. También se tienen que ocupar de la intercomunicación entre ellos mismos y el resto de robots operativos para saber las posiciones de los mismos e intercambiarse información relevante de cara a las acciones o tareas a realizar. Muchas variables y mucha información hacen que el sistema formado por robots autónomos no sea trivial y aumente su complejidad exponencialmente a medida que aumenten el número de robots, funcionalidades, etc.

La principal desventaja reside en la complejidad del sistema y ser capaces de experimentar en un entorno de trabajo con múltiples robots autónomos conlleva largos periodos de toma de datos e interpretación de los mismos. Hay que asegurarse de que los protocolos de comunicación entre ellos funcionan correctamente y que la realización de tareas se lleve a cabo sin obstaculizar al resto de la flota. La ventaja de este tipo de sistemas compuesto por robots autónomos es que si un robot de la flota queda inutilizado, no supone un mayor problema para el resto. Será tratado como un obstáculo más del entorno incapaz de interactuar con la flota.

En la figura 2.3 podemos ver un ejemplo de robot autónomo. El Robot Humanoide Rh-1 desarrollado por la Universidad Carlos III de Madrid es capaz de realizar misiones cooperativas tanto con humanos como con otros robots en entornos de trabajo reales.



Figura 2.3: Robot humanoide Rh-1

Actualmente, se tiende a que las flotas de robots cada vez dependan menos de un control centralizado con el fin de hacerlos más autónomos de cara a lidiar con eventos u obstáculos inesperados. Esta autonomía se conseguirá usando capacidades avanzadas basadas en múltiples sensores (para la localización, detección de obstáculos y modelado del entorno) así como la planificación deliberada entre los robots a través de la intercomunicación y coordinación entre ellos como en el Martha Project [4].

Al tender a la descentralización se flexibilizará la estimación de los recursos necesarios y el tiempo que les llevará realizar una tarea a los propios entes autónomos. De esta manera se evita al sistema central la carga que supone elaborar planes eficientes y fiables de movimiento. A medio y largo plazo esta tarea puede ser complicada debido a la naturaleza dinámica del entorno.

Ya que el guiado de un robot móvil involucra su localización en un entorno, y por estar este entorno generalmente acotado, se decide que para la realización de este trabajo se utilizará un control centralizado que con ayuda de una cámara digital permita al centro de control saber en todo momento el estado del sistema y de todos los robots que lo componen. El centro de control analizará la información proveniente de la cámara en tiempo real y será capaz de situar y posicionar en un entorno virtual los robots móviles del sistema. Posteriormente, y una vez analizado el entorno de trabajo así como el número de robots integrantes de la flota, ser capaz de interactuar con ellos y transmitir mediante protocolos de comunicación órdenes básicas de movimiento.

2.3 Requisitos y restricciones del sistema

Dado que el objetivo del trabajo es crear una plataforma experimental y no comercial los requisitos que vamos a imponer en nuestro sistema estarán ligados con la obtención de resultados fiables. Las características que exigiremos a nuestro sistema es que sea:

- Robusto. El sistema tiene que poder contemplar todas las posibles facetas del problema que queremos resolver. Si logramos construir un sistema robusto, cualquier giro inesperado del problema será controlado por el sistema y sus algoritmos. Si se produce un fallo en el sistema, ser capaz de identificar el error y que el propio sistema sea capaz de autorecuperarse.
- Fácil e intuitivo de cara al usuario final. Que se pueda manejar sin entrenamientos previos, que no presente una interfaz compleja que dificulte la utilización. La simplicidad es la máxima sofisticación. La posible complejidad residente en los algoritmos no tiene que ser relevante para el usuario y no tiene que ser exteriorizada a través de la interfaz si no al contrario.
- Bajo coste. Queremos obtener resultados aplicables en sistemas de mayor complejidad con un sistema que no tenga un coste excesivo.
- Capacidad de procesado cercana a la velocidad de refresco de las cámaras comerciales (24 Hz). Se busca que cada iteración del algoritmo esté dentro de un límite de tiempo para garantizar de que el procesado de cada frame sea transparente para el usuario. Que el usuario pueda ver los resultados sin sufrir ralentizaciones y que se produzcan de manera fluida los resultados.
- Capacidad de detectar varios robots. El sistema tiene que ser capaz de identificar varios robots y situarlos de una manera precisa en el entorno virtual con el fin de crear un mapa del entorno de trabajo con los robots de la flota identificados correctamente.
- Falsos positivos muy bajos. De vital importancia para garantizar la fiabilidad del sistema.

Las restricciones que nos encontraremos serán que al ser un entorno de trabajo cerrado y limitado la experimentación será también limitada ya que solo se podrán testear sistemas de robots simples que no requieran mucha complejidad y sean de reducido tamaño.

3. Base teórica en la que se basa el trabajo

3.1 Introducción

En este capítulo se presentará la base y los fundamentos teóricos sobre los que estará basado el trabajo. Para una correcta comprensión del mismo creemos necesario detallar los principios básicos tanto de visión artificial como de robótica implicados en el trabajo.

3.2 Visión

Actualmente, uno de los mayores retos con los que se encuentra la robótica es el tratamiento de imágenes en tiempo real. Aspectos que aparentemente pueden resultar sencillos para nosotros como puede ser discriminar un objeto, no son tan triviales a la hora de tratarlo con un ordenador. El tratamiento digital de la imagen nos permitirá a partir de una imagen origen, llegar a otra en la que se mejoren ciertas características de la misma haciendo que el resultado sea más adecuado para una aplicación específica.

Para la correcta comprensión de los futuros capítulos nos centraremos en explicar en qué consiste la visión artificial y cómo es el proceso de formación y representación de imágenes en los dispositivos digitales.

3.2.1 ¿Qué es la visión artificial?

La visión artificial es el conjunto de técnicas y algoritmos que permiten la obtención, procesamiento y análisis de cualquier tipo de información en imágenes digitales.

Los objetivos típicos de la visión artificial incluyen:

- La detección, segmentación, localización y reconocimiento de ciertos objetos en imágenes (por ejemplo, caras humanas).
- La evaluación de los resultados (por ejemplo, segmentación, registro).
- Registro de diferentes imágenes de una misma escena u objeto, es decir, hacer concordar un mismo objeto en diversas imágenes.
- Seguimiento de un objeto en una secuencia de imágenes.
- Mapeo de una escena para generar un modelo tridimensional de la misma.
- Estimación de las posturas tridimensionales de humanos.
- Búsqueda de imágenes digitales por su contenido.

Estos objetivos se consiguen por medio de reconocimiento de patrones, aprendizaje

estadístico, geometría de proyección, procesamiento de imágenes, teoría de grafos y otros campos. Las herramientas usadas en la visión artificial nos permitirán establecer la relación existente entre el mundo real y sus vistas bidimensionales.

3.2.2 La imagen digital

La formación de imágenes es un proceso mediante el cual una información luminosa 3D (la escena) es proyectada en un plano 2D (la imagen). Las cámaras intentan imitar el proceso que tiene lugar en el ojo humano. El esquema de la figura 3.1 explica de manera simplificada este proceso.

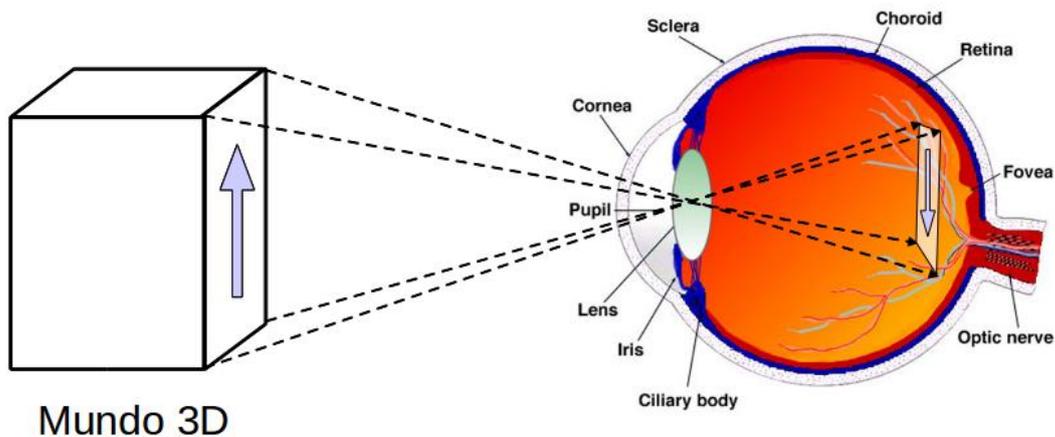


Figura 3.1: Proceso de adquisición de imágenes del ojo humano

En el tratamiento digital de la imagen, esta se ve como una matriz, o array bidimensional de números donde cada celda de la matriz es un píxel. Cada píxel representa el valor de una magnitud física como puede ser la cantidad de luz en un punto de una escena, el valor de color (cantidad de radiación en la frecuencia del rojo, verde y azul), profundidad, etc. En la figura 3.2 podemos ver como es esta representación bidimensional.

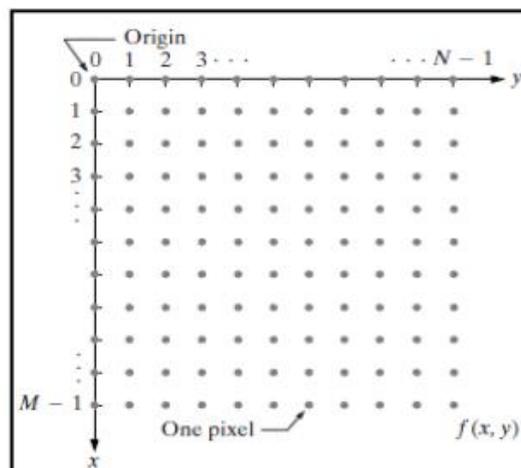


Figura 3.2: Representación de una imagen digital

Una de las caracterizaciones posibles de las imágenes es por el tipo de datos que es cada celda de la matriz así:

- 1- Imagen binaria donde cada píxel se corresponde con un bit (0 = negro; 1 = blanco).
- 2- Imagen en escala de grises donde cada píxel se corresponde con un byte que corresponde con el valor de gris asociado (256 niveles de gris).
- 3- Imagen en color donde cada píxel se corresponde con tres bytes (3 canales). Generalmente las cámaras digitales trabajan con el modelo RGB, esto es, un byte por cada color (Rojo, Verde y Azul). Otros modelos de representación de colores mediante tres canales son posibles trabajando en otros espacios de color como HSV, YcrCb, etc.

En la siguiente figura podemos ver las tres caracterizaciones de una misma imagen según los puntos explicados anteriormente.

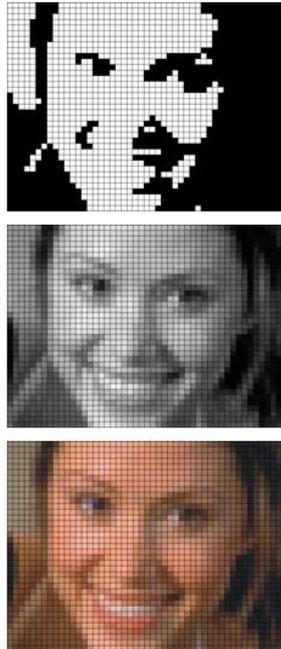


Figura 3.3: Imagen binaria, en escala de grises y en color

En nuestro caso vamos a trabajar con una cámara de visión Logitech por lo que las imágenes que vamos a tratar serán las imágenes en color que nos proporcione la propia cámara. Nuestro objetivo será tratar estas imágenes digitales (recordemos de 3 canales) para resaltar las características de la imagen en base a los criterios de identificación que se detallan en el este y en el siguiente capítulo. De esta manera, podremos identificar posteriormente con mayor facilidad los robots en el entorno de trabajo.

3.2.3 El color

La luz visible forma parte del espectro de radiación electromagnética. Los seres humanos sólo somos capaces de distinguir una parte muy pequeña dentro de todo el espectro electromagnético. Concretamente, la radiación comprendida entre los 400nm y los 750 nm como podemos ver en la siguiente figura.

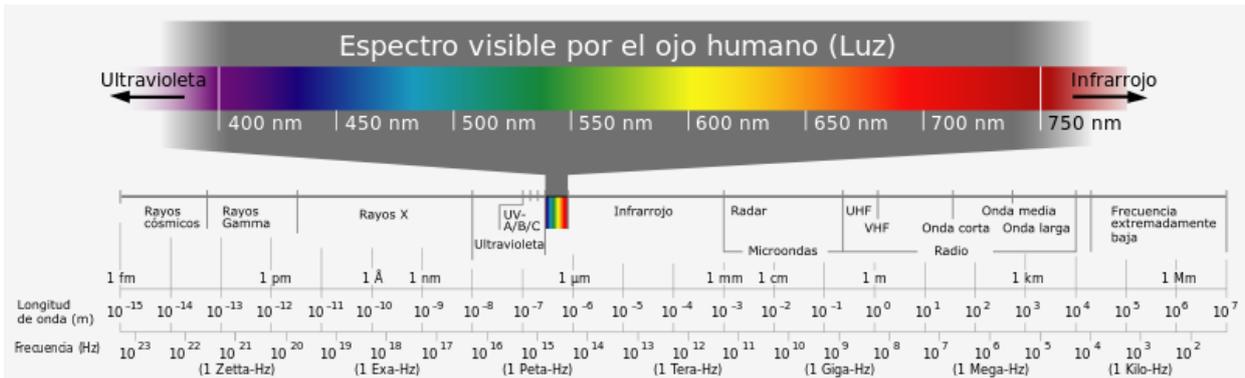


Figura 3.4: Espectro electromagnético

Como el color es una impresión sensorial del observador la Teoría de Color se basa en los principios de funcionamiento del órgano de la vista. Thomas Young (1773 - 1829) determinó que los colores del espectro pueden obtenerse por la suma de tres colores básicos: el rojo, el verde y el azul intenso (síntesis aditiva del color).

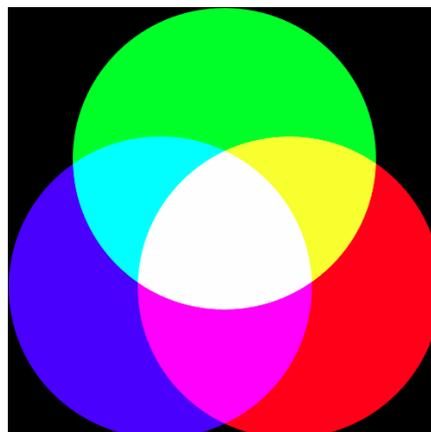


Figura 3.5: Colores primarios y su combinación

Estos colores, comúnmente llamados primarios, son conceptos idealizados utilizados en modelos de color matemáticos que no representan las sensaciones de color reales o incluso los impulsos nerviosos. El espacio de color formado por los tres colores primarios, esto es el espacio RGB, es el más extendido y el que usan la mayoría de las cámaras fotográficas y de vídeo para construir una imagen a color. La importancia de este espacio de color dentro de la visión artificial reside en que trabajando con el mismo espacio de color con el que trabaja la cámara que captura las imágenes se evita la alteración de las propiedades del color durante el procesado de la imagen.

En el espacio RGB las imágenes digitales a color se caracterizan con 3 canales, esto es, tres matrices distintas en la que cada una hace referencia a la intensidad de color rojo, verde y azul respectivamente de un determinado píxel. Por tanto, este modelo se puede caracterizar matemáticamente utilizando el espacio cartesiano de forma que los vectores ortogonales (siendo estos los colores rojo, verde y azul) forman un subespacio vectorial capaz de caracterizar cualquier color como podemos ver en la figura 3.6.

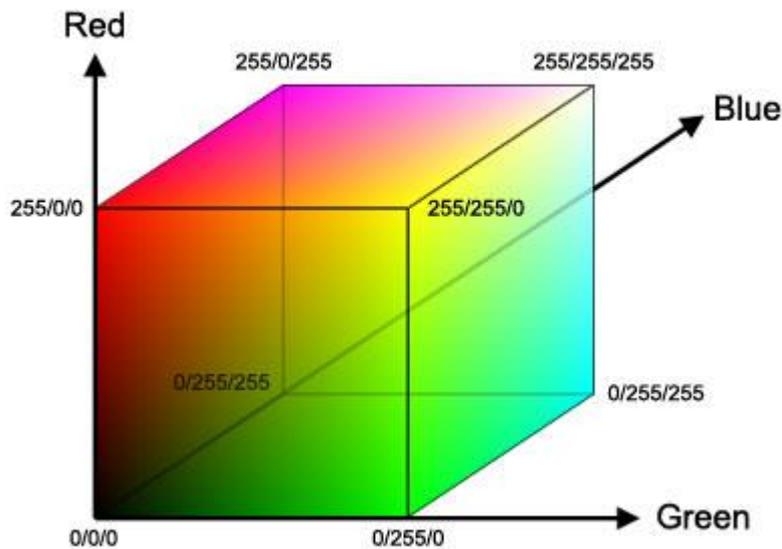


Figura 3.6: Cubo RGB

Este modelo, sirve para entender la síntesis aditiva del color ya que hace uso de una de las principales características del ojo humano, que es la detección de los colores primarios. Sin embargo, a la hora de trabajar con él no resulta demasiado útil ya que no tiene en cuenta la separación de intensidad y cromaticidad entre los colores, propiedades importantes a la hora de diferenciar y discriminar los colores.

El modelo HSV resuelve los problemas encontrados en el modelo RGB y se deriva de él aplicando una serie de transformaciones no lineales. La caracterización matemática de este modelo pasa de ser un cubo a un cono de color, donde el tono (hue) corresponde con el ángulo desde un eje de referencia, dado en coordenadas polares, la saturación (saturation) depende del grado en que el color tenga luz blanca, y viene dado por la distancia al centro y la intensidad (value) es una medida de la cantidad de luz asociada al máximo de los tres valores RGB y corresponde a la altura del eje perpendicular al sistema de coordenadas polares. Podemos verlo gráficamente en la figura 3.7 que representa el cono de color correspondiente al espacio HSV.

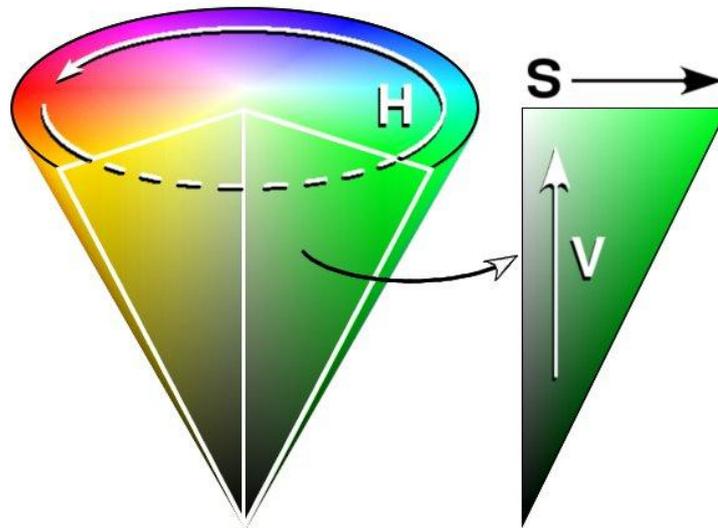


Figura 3.7: Cono HSV

Las transformaciones entre el espacio RGB y el HSV se hacen según las fórmulas presentes a continuación. Cabe destacar que el resultado de aplicar las ecuaciones siguientes deriva en tener un valor de matiz (value) entre 0° y 360° , un valor de saturación (saturation) entre 0 y 1 y un valor de intensidad entre 0 y 255. En la práctica, en la programación, estos valores estarán representados por 1 byte, por tanto se representarán todos con valores de entre 0 y 255.

$$H = \begin{cases} \frac{G-B}{V-\min\{R,G,B\}} \cdot 60^\circ, & \text{if } V = R \text{ and } G \geq B; \\ \left(\frac{B-R}{V-\min\{R,G,B\}} + 2 \right) \cdot 60^\circ, & \text{if } G = V; \\ \left(\frac{R-G}{V-\min\{R,G,B\}} + 4 \right) \cdot 60^\circ, & \text{if } B = V; \\ \left(\frac{R-B}{V-\min\{R,G,B\}} + 5 \right) \cdot 60^\circ, & \text{if } V = R \text{ and } G < B \end{cases} \quad H \in [0^\circ, 360^\circ]$$

$$S = \frac{V - \min\{R, G, B\}}{V} \quad S \in [0, 1]$$

$$V = \max\{R, G, B\} \quad V \in [0, 255]$$

3.2.4 Segmentación y etiquetado

La segmentación es el proceso por el cual la imagen se descompone en regiones o elementos que pueden corresponder a objetos o partes de objetos. Se evalúa en cada píxel si este pertenece o no al objeto que se está buscando. Esta técnica de procesamiento de imágenes genera una imagen binaria, donde se representa con 1 los píxeles que pertenecen a dicho objeto y 0 a los que no.

En el tratamiento de la imagen digital, la segmentación se puede hacer en base a múltiples criterios. En nuestro caso hemos decidido hacer la segmentación por color, porque además de ser más visual se consiguen mejores resultados. La primera parte del algoritmo segmentará por tanto por color dejando en una imagen binarizada únicamente los colores que nos interesan para posteriormente proceder al etiquetado de cada uno de los robots.

Cómo hemos visto anteriormente se decide que la segmentación por color se realizará en el espacio HSV por ser el más idóneo para nuestra aplicación. La segmentación dentro del programa se realiza segmentando individualmente el matiz, la saturación y la luminancia. Para ello primero tendremos que pasar la imagen del espacio RGB (el que nos proporciona la cámara de visión) al espacio HSV y posteriormente separar las tres componentes de color.

Una vez que tenemos en una imagen segmentada por color se procede a hacer un etiquetado teniendo en cuenta la vecindad de los píxeles. Este etiquetado permite:

- Distinguir los objetos respecto del fondo
- Distinguir un objeto respecto al resto
- Conocer el número de objetos de una imagen

En la figura 3.8 podemos ver una imagen a color, a continuación la segmentación del color rojo que da lugar a una imagen binarizada y finalmente el etiquetado y numeración de las etiquetas.

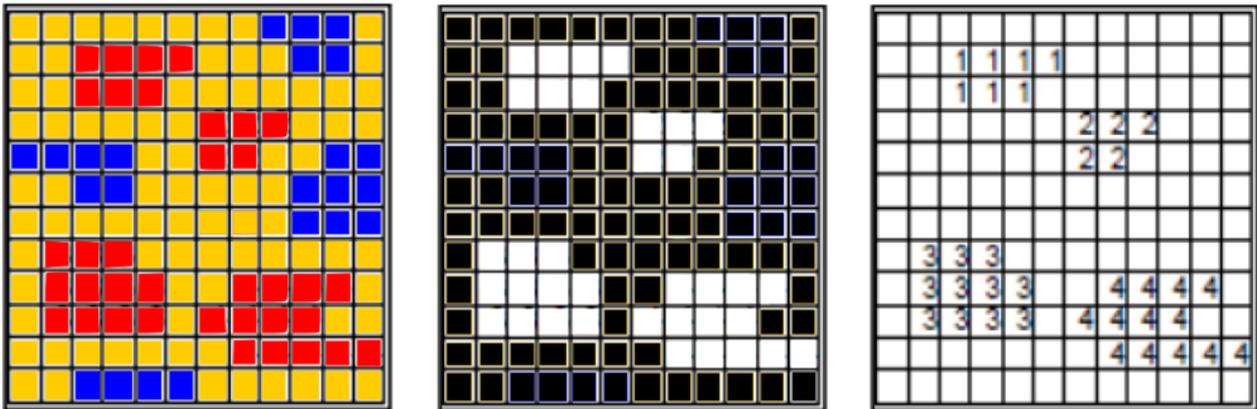


Figura 3.8: Etiquetado y cuenta de objetos

3.2.5 Modelo Pinhole

El modelo pinhole describe la relación matemática entre las coordenadas de un punto tridimensional y su proyección en una imagen plana de una cámara estenopeica ideal (*ideal pinhole camera*). El modelo pinhole por tanto es usado para hacer una aproximación de primer orden para mapear una escena 3D en una imagen 2D. Su validez depende en la calidad de la cámara y en general decrece desde el centro de la imagen hacia los bordes a medida que la distorsión de las lentes aumenta.

La explicación matemática la podemos encontrar en la siguiente figura. Supongamos un sistema de coordenadas ortogonal tridimensional con su origen en 0. Denotamos los ejes del sistema de coordenadas como X1, X2 y X3 estando el eje X3 apuntando en la dirección de la cámara y siendo conocido este como eje óptico o principal.

La imagen bidimensional donde el mundo 3D es proyectado es paralela a los ejes X1 y X2 y está situada a una distancia f del origen 0 en la dirección negativa del eje X3. Esta distancia f es conocida como distancia focal. El punto R es la intersección del eje X3 y el plano de la imagen y es conocido como el centro de la imagen. Desde este punto deriva un nuevo sistema de coordenadas, ahora bidimensional, compuesto por los ejes Y1 e Y2 (paralelos a X1 y X2). Un punto P del espacio con coordenadas (x_1, x_2, x_3) es proyectado a través de la cámara en el punto Q de la imagen bidimensional dando lugar a unas nuevas coordenadas (y_1, y_2) en el nuevo sistema de referencia bidimensional.

$$y_1 = f \frac{x_1}{x_3} \quad y_2 = f \frac{x_2}{x_3}$$

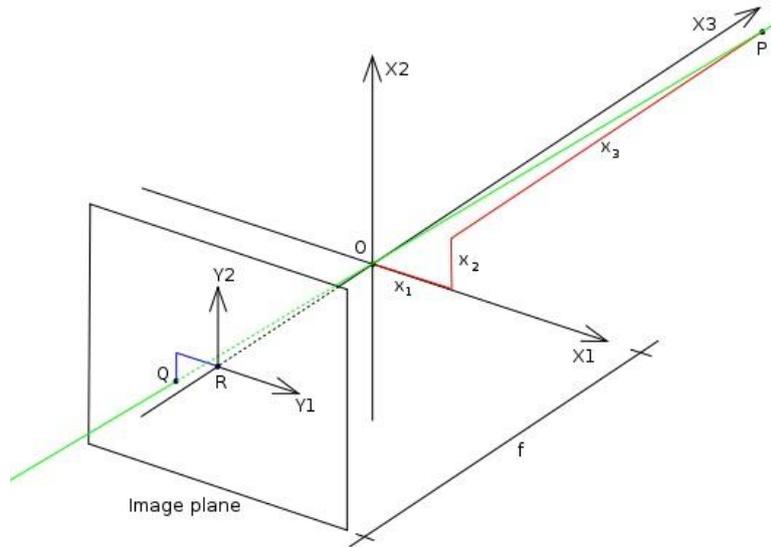


Figura 3.9: Modelo Pinhole

3.3 Robótica

La robótica es la rama de la tecnología dedicada al diseño, construcción, operación, disposición estructural, fabricación y aplicación de los robots. Un robot, es un agente artificial mecánico o virtual aunque para nuestro trabajo consideraremos únicamente el primer caso.

En general, un robot, para ser considerado como tal, debería presentar alguna de las siguientes propiedades:

- Ha sido creado artificialmente.
- Puede sentir su entorno.
- Puede manipular cosas de su entorno.
- Tiene cierta inteligencia o habilidad para tomar decisiones basadas en el ambiente o en una secuencia preprogramada automática.
- Es programable.
- Puede moverse en uno o más ejes de rotación o traslación.
- Puede realizar movimientos coordinados.

Existen diferentes tipos y clases de robots dependiendo de sus capacidades:

- Androides: robots con forma humana. Intentan reproducir de forma total o parcial la forma y el comportamiento del ser humano. Su utilidad en la actualidad es de solo experimentación. La principal limitante de este modelo es la implementación del equilibrio en el desplazamiento. Últimamente se han obtenido grandes logros en el campo de la locomoción bípeda con este tipo de robots.

- **Móviles:** cuentan con una gran capacidad de desplazamiento. Dotados de un sistema locomotor de tipo rodante se guían en el entorno usando sensores con los que caracterizan el entorno que les rodea.
- **Industriales:** formados por dispositivos mecánicos y electrónicos destinados a realizar de forma automática determinados procesos de fabricación o manipulación.

Los robots de nuestro trabajo son robots móviles y a continuación explicaremos los elementos básicos que componen un robot de este tipo.

3.3.1 La estructura

La estructura es el esqueleto del robot. La rigidez estructural del robot va a ser determinante en la realización de tareas y dependiendo el campo de trabajo del propio robot el diseño variará para adecuarse correctamente a la labor que va a realizar.

3.3.2 Electrónica

La parte de la electrónica en un robot compone la serie de elementos que proporcionarán capacidad de procesamiento y comunicación al mismo para la realización de tareas. El elemento principal es el microprocesador, que integrado en una placa de control hará las funciones de cerebro del robot.

El microprocesador (o concretamente el microcontrolador en el mundo de la robótica) está presente hoy en día en todos los aparatos electrónicos que requieren un grado mínimo de control. La principal característica que los hace imprescindibles en estos sistemas es que proporcionan inteligencia y además son programables. Con la ayuda de los sensores, dispositivos capaces de detectar magnitudes físicas o químicas dentro de un rango de medida, serán capaces de caracterizar el entorno para poder tomar decisiones futuras tanto de movimiento como de comunicación. También son importantes los actuadores, dispositivos inherentemente mecánicos cuya función es proporcionar fuerza para mover o actuar otro dispositivo mecánico. En los actuadores eléctricos la fuerza que provoca el actuador proviene de la fuerza motriz. Son ideales cuando es necesario generar fuerza rápidamente o cuando se necesita un posicionamiento preciso.

3.3.3 Servomotores

Un servomotor (o servo) es un motor de corriente continua que tiene la capacidad de ser controlado en posición. También existe la versión síncrona de corriente alterna pero estos son usados para máquinas con grandes requisitos dinámicos como robots industriales.

En función de una señal de control es capaz de posicionarse dentro de un rango de operación (generalmente 180°) y mantenerse estable en dicha posición. Los servos se suelen utilizar en robótica, automática y modelismo debido a que ofrecen grandes ventajas:

- Generalmente tienen un tamaño reducido.
- Tienen un gran momento en la fuerza (torque), fuerza con respecto al eje, comparado con su tamaño.
- Operan en lazo cerrado, esto es, mediante un circuito de control interno pueden medir la posición exacta de la posición, por tanto, son muy precisos.
- Son eléctricamente eficientes, la corriente requerida es proporcional al peso que tienen que desplazar.

Control básico de un servo

Los servos, generalmente, disponen de tres cables: dos cables de alimentación (positivo y negativo/masa) que suministran un voltaje 4.8-6V y un cable de control que indica la posición deseada al circuito de control mediante señales PWM (“Pulse Width Modulation”). En la figura 3.10 podemos apreciar la presencia de los tres cables donde el cable blanco corresponde a la señal de control, el cable rojo a la alimentación y el cable negro a tierra. La señal digital que se manda por el cable de control debe ser cuadrada y dependiendo la longitud del pulso variará el grado de giro del servo como podemos ver en la figura 3.11.



Figura 3.10: Servomotor

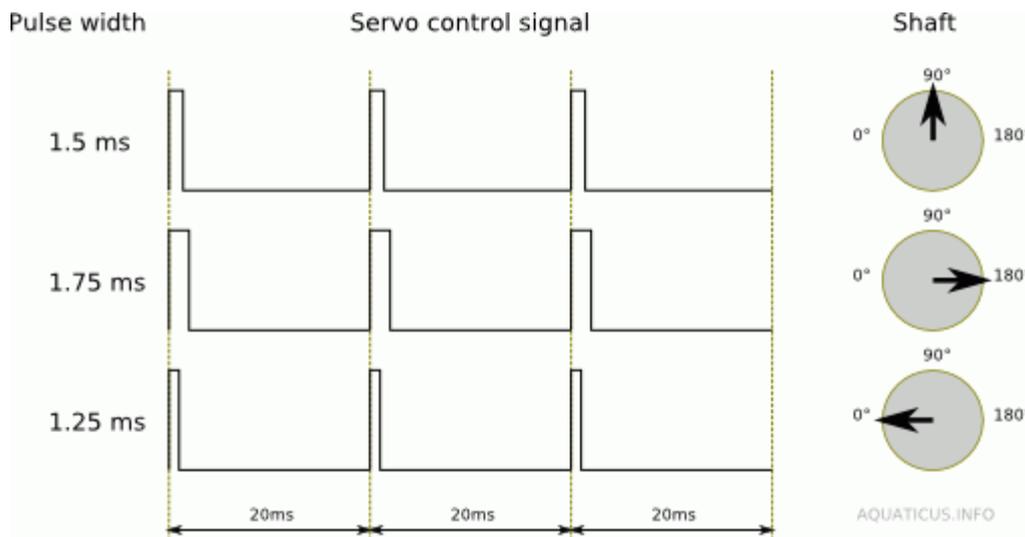


Figura 3.11: Control básico de un servo

4. Diseño de la solución técnica

4.1 Introducción

Esta parte se divide en dos grandes bloques; la parte de visión y la parte de robótica. Cabe destacar que ambas partes están basadas en herramientas Open Source Hardware (OSHW) y Free and Open Source Software (FOSS). Esta decisión la hemos tomado ya que creemos necesario el desarrollo de plataformas de acceso público que potencien la libertad de conocimiento. Las grandes industrias del sector tecnológico, con el fin de afianzar su monopolio, limitan la información accesible por medio de la implementación de patentes y restrictivos derechos de autor. Esto dificulta enormemente la difusión del conocimiento ya que únicamente lo hace accesible a una entidad muy reducida. El objetivo principal de las iniciativas de Open H / S es cultivar el conocimiento y tratar de ofrecer productos y servicios tecnológicos de calidad hechos y mejorados por todo el mundo.

Creemos en la necesidad de código abierto no solo en el terreno educativo. El código abierto es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista orientado a los beneficios prácticos ya que es software que podemos leer, modificar y redistribuir gratuitamente dando libertad total para los usuarios. Esta libertad repercute en que usuarios de todo el mundo pueden mejorar los programas y redistribuirlos nuevamente. Esto beneficia a todos ya que el progreso es instantáneo pudiéndose beneficiar de él el resto de la comunidad.

Estas comunidades de desarrolladores no están formadas únicamente por usuarios anónimos si no que existen también organizaciones que creen que la propiedad intelectual y el copyright asfixian la creatividad, favorecen la industria de contenidos y limitan la difusión de conocimiento ya que están basados en sistemas beneficiarios con ánimo de lucro.

Creative Commons (CC) es una organización no gubernamental sin ánimo de lucro que desarrolla planes para ayudar a reducir las barreras legales de la creatividad, por medio de nueva legislación y nuevas tecnologías [5]. La iniciativa de Creative Commons se inspira claramente en la filosofía del software libre y es el resultado de un año de esfuerzos por parte de un grupo de especialistas estadounidenses en ciberderecho, entre ellos Lawrence Lessig, profesor de derecho en la Universidad de Stanford y autor de *El código y otras leyes del ciberespacio*.

La idea central de Creative Commons es ofrecer un modelo legal y una serie de aplicaciones informáticas que faciliten la distribución y uso de contenidos dentro del dominio público. Creative Commons proporciona un sistema que automatiza la

búsqueda de contenidos "comunes". Al licenciar su obra, el creador establece condiciones generales que quedan incorporadas digitalmente a la obra, de manera que un motor de búsqueda puede identificarlas. El usuario con unas ciertas necesidades puede hallar una serie de obras que le permiten satisfacerlas y escoger la que más le convenga [6].

El objetivo último de Creative Commons es modular y refinar el concepto de propiedad intelectual, y crear un espacio en el que creadores de todo el mundo colaboren productivamente "a ciegas" y con una mayor libertad que la que el modelo actual permite. Cabe destacar que las licencias CC están inspiradas en la licencia GPL. La licencia GNU GPL (GNU General Public License) es una licencia creada por la Free Software Foundation en 1989 (la primera versión, escrita por Richard Stallman), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Hoy en día numerosos proyectos están basados en esta filosofía y son distribuidos bajo este tipo de licencias. Nuevos proyectos surgen con la idea de que todo el mundo se pueda beneficiar de ellos y colaborar para mejorarlos como el SGPS Project [7]. Quizá, el paradigma por excelencia del software libre está representado por Linux, sistema operativo GNU basado en Unix y desarrollado por colaboradores de todo el mundo [8]. Para este proyecto se han utilizado plataformas de Open H / S como OpenCV, Arduino e impresoras 3D que se detallarán conjuntamente en el desarrollo del presente capítulo.

4.2 Visión

En esta parte del trabajo nos centraremos en realizar un algoritmo de visión que nos permita discriminar la información que consideremos relevante para situar los robots en un entorno virtual que se asemeje a la situación real de los mismos en el entorno operativo real.

4.2.1 Herramientas

En este apartado se describirán las herramientas usadas tanto en hardware como en software para la implementación de la parte de visión del trabajo. No solo se describen las características técnicas si no también la información que hemos considerado relevante para justificar la funcionalidad de las mismas.

4.2.1.1 Hardware

Logitech Webcam C210

Formato de video soportado	480p
Megapíxeles	1.3 MP
Máxima resolución de video	640x480 píxeles



Figura 4.1: Logitech Webcam C210

Lámpara de tecnología LED

Lámpara de diseño y fabricación propia con 4 LEDs de alta iluminación y ópticas difuminadas para garantizar una iluminación homogénea en el entorno de trabajo. Necesita un voltaje de alimentación de 3.3 V para funcionar. En la figura 4.2 podemos ver la lámpara montada sobre la placa de diseño y sin encender. A continuación en la figura 4.3 vemos la misma lámpara funcionando.

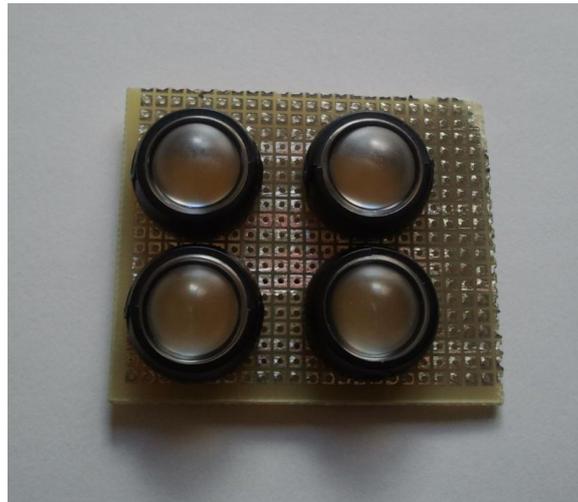


Figura 4.2: Lámpara LED apagada

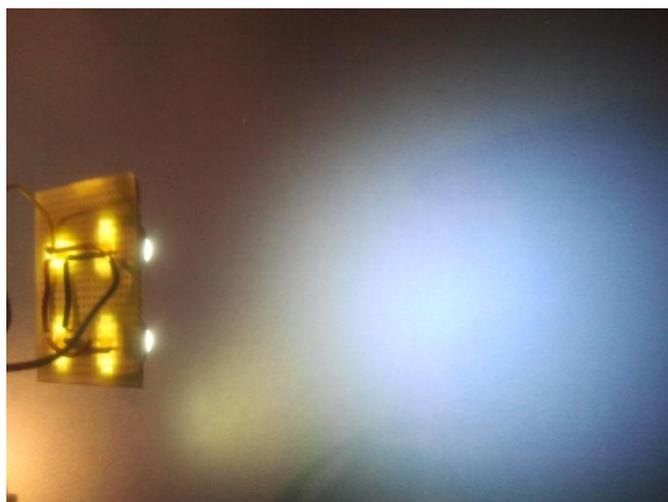


Figura 4.3: Lámpara LED encendida

4.2.1.2 Software

La programación se realizará en C++ usando las siguientes librerías de visión.

OpenCV



OpenCV (Open Source Computer Vision Library) es una biblioteca libre de visión artificial originalmente desarrollada por Intel. Su primera versión alfa data de enero de 1999. Esta distribuida bajo una licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación con las condiciones en ella expresadas.

Open CV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS X y Windows. Contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión, como reconocimiento de objetos, reconocimiento facial, calibración de cámaras, visión estéreo y visión robótica.

CvBlob



CvBlob es una librería específica de visión artificial para detectar regiones conectadas en imágenes digitales binarizadas. Utiliza análisis de componentes conectados (también conocido como etiquetado) para la detección de características.

4.2.2 El algoritmo

En este apartado vamos a proceder a analizar el algoritmo de visión por partes. El algoritmo tiene 3 partes claramente diferenciadas que ahora procederemos a analizar en detalle.

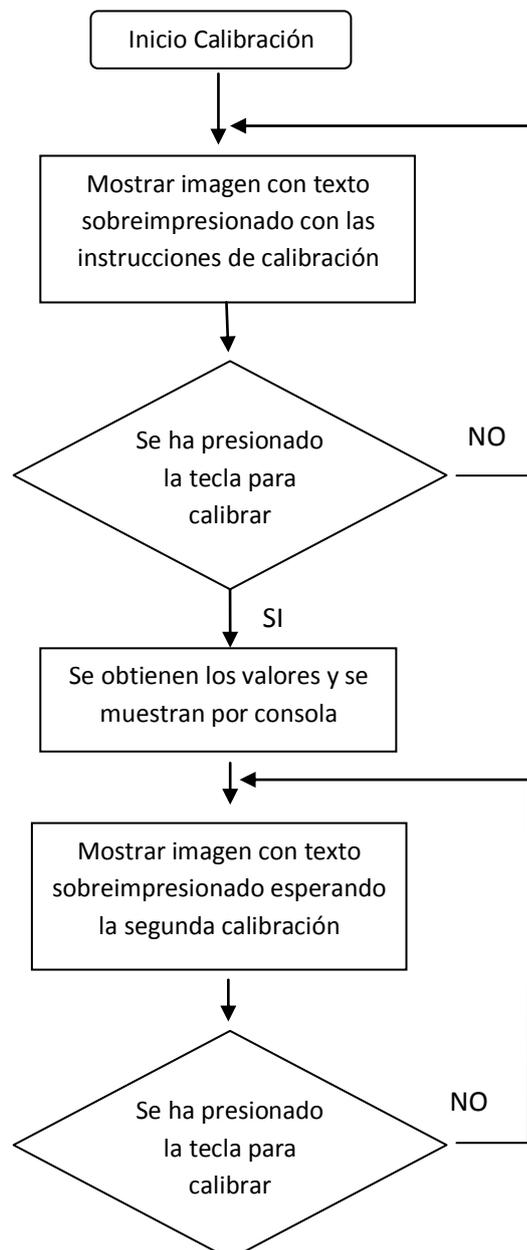
4.2.2.1 Calibración

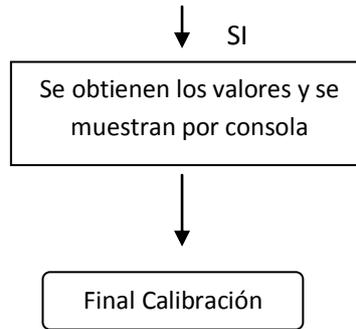
Uno de los requisitos del sistema que vamos a diseñar es que sea fiable. La fiabilidad en visión artificial es difícil de conseguir sin una fase previa de calibración. El ajuste para el futuro proceso de segmentación de la imagen estará basado en esta primera fase. Midiendo los valores de una magnitud conocida y que posteriormente

estará presente en el entorno de trabajo ayudará a segmentar con una mayor fiabilidad y precisión.

Por tanto, en esta parte inicial el programa pide al usuario que calibre el programa para posteriormente proceder a la segmentación por colores de una forma precisa. La calibración se realiza mostrando en una ventana por pantalla una imagen de la webcam con un rectángulo blanco sobreimpresionado así como las instrucciones para la calibración. El usuario deberá colocar en el rectángulo anteriormente mencionado una cartulina del color que va a identificar al primer robot y presionar la letra c (“calibration”) para que el propio programa obtenga los valores HSV que caracterizarán al primer robot. Posteriormente se pide lo mismo para el segundo robot, que estará identificado con etiquetas de otro color. Cabe destacar que los valores de HSV obtenidos en la calibración se muestran por consola.

El proceso de calibración se desarrolla según el siguiente diagrama de flujo.





Como podemos ver en la figura 4.4, se pide al usuario que coloque una cartulina del color que caracterizará al primer robot en el rectángulo blanco sobreimpresionado en la imagen. Al presionar la tecla c se pasa a la segunda calibración donde se pide ahora el color que identificará al segundo robot, figura 4.5. Nótese que en la figura 4.4 aparecen por consola los valores obtenidos para la primera calibración correspondiente al color azul.

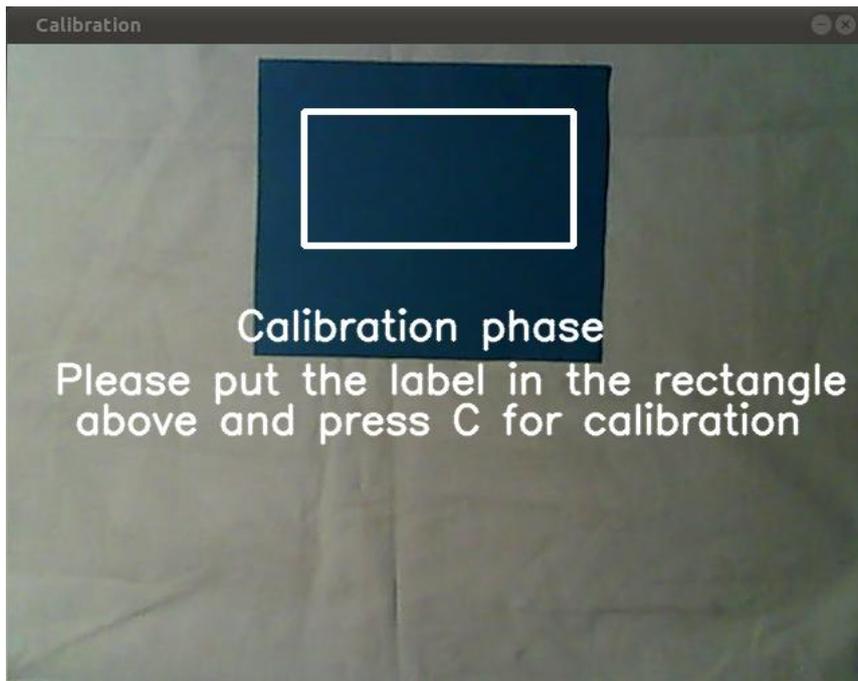


Figura 4.4: Calibración para el color azul

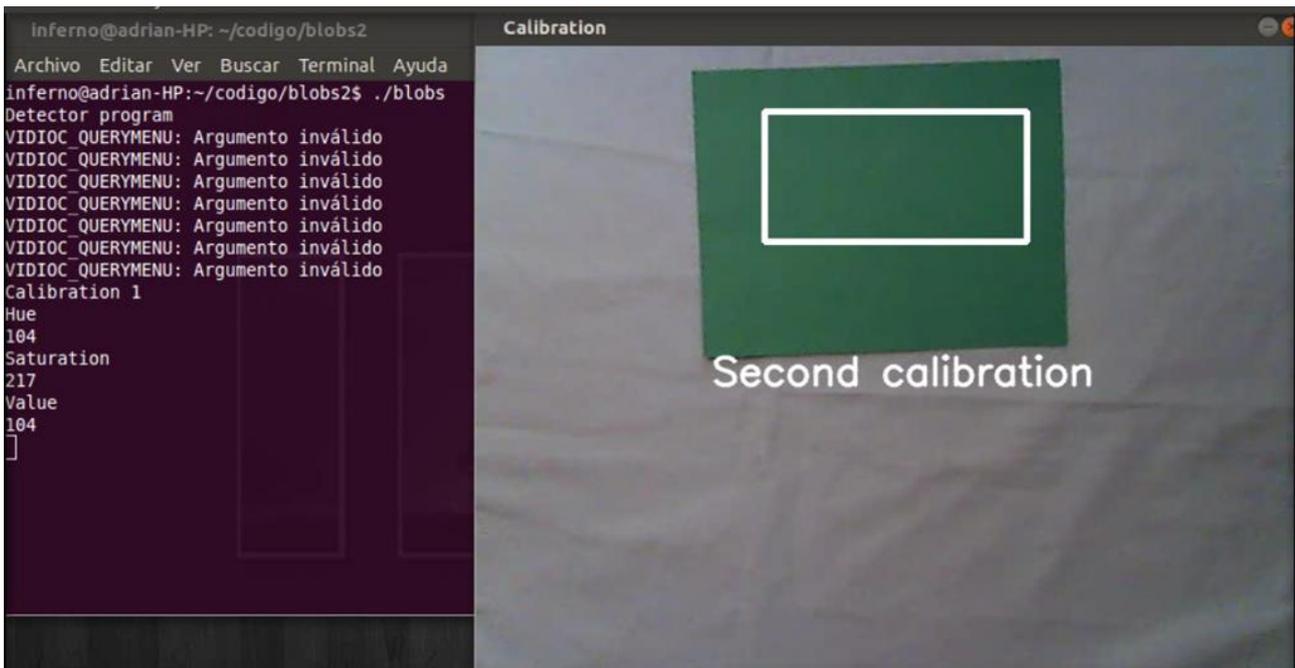


Figura 4.5: Calibración para el color verde

Una vez obtenidos los colores que caracterizarán a cada uno de los robots en su representación HSV se pasa a la siguiente parte del algoritmo.

4.2.2.2 Segmentación y etiquetado

El primer paso, después de la calibración, en la mayoría de sistemas de visión es la reducción de información que nos proporciona la cámara. Deshacernos de la información no relevante constituirá la base para el etiquetado posterior. El principio de la reducción de información es la localización de áreas de píxeles del mismo color. En la práctica encontrar píxeles con valores iguales a los obtenidos en la calibración es difícil por lo que se amplían estos valores dentro de unos márgenes ajustables para así adecuarse al caso real que dista del teórico debido a las condiciones inherentes del entorno como la iluminación.

En esta parte del algoritmo se segmenta la imagen que proporciona la webcam para ver pixel por pixel si el valor de color del mismo, en valor HSV, coincide con el de alguno de los robots. De esta manera se pasa a dos imágenes binarizadas donde los píxeles blancos (valor 1) indican la presencia del color correspondiente (con unos límites de tolerancia) y los píxeles negros (valor 0) indican que el valor del color del equivalente en la imagen original está fuera de los márgenes de color que estamos buscando.

En la figura 4.6 se puede observar la imagen original que está compuesta por etiquetas de color azul y verde sobre un fondo blanco. En las figuras 4.7 y 4.8 se pueden apreciar la segmentación de ambos colores individualmente.

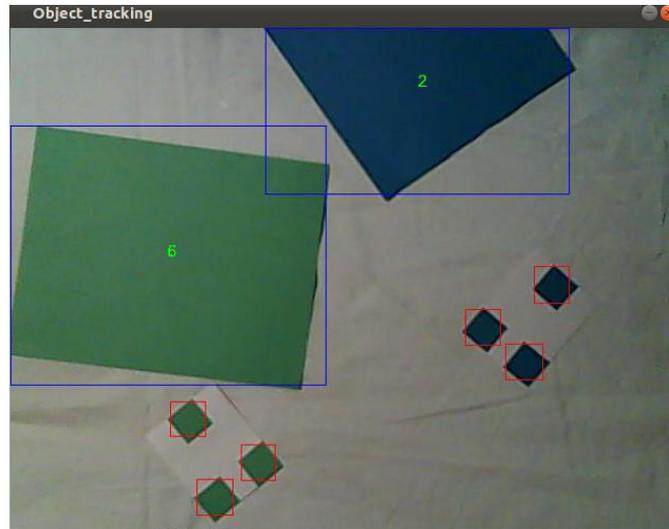


Figura 4.6: Imagen original con resultado sobreimpresionado de la segmentación y etiquetado

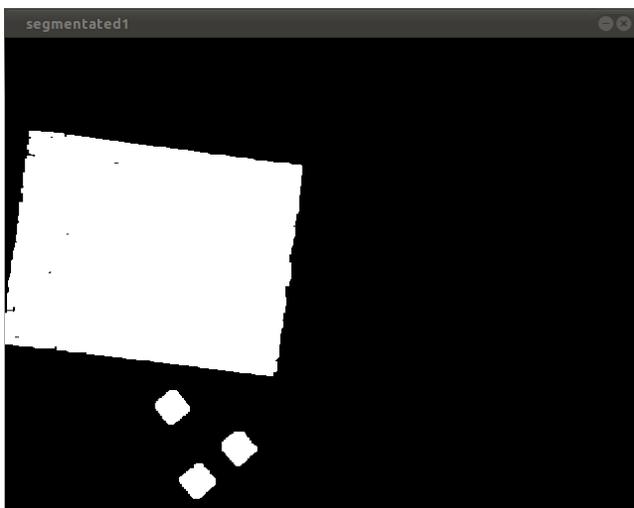


Figura 4.7: Segmentación del color verde individualmente

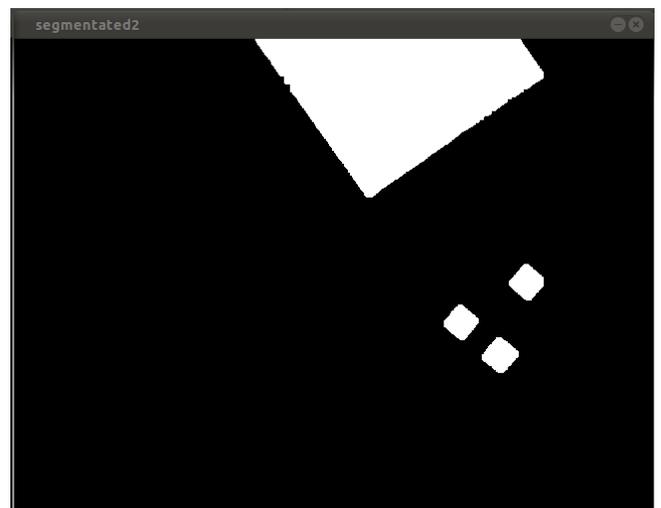


Figura 4.8: Segmentación del color azul individualmente

Una vez que tenemos la imagen binarizada representando la presencia de los colores buscados se procede al etiquetado. El etiquetado nos permite agrupar píxeles conectados para considerarlos como una región. La etiqueta que identifica cada robot no está compuesta por una lámina homogénea de color si no que está compuesta por varias formas separadas con el fin de permitir averiguar la dirección en la que está orientado el robot. Esto indica que vamos a tener que buscar regiones inconexas en la imagen que tengan el mismo color; estas supondrán la presencia de un robot y analizando las distancias entre las regiones se podrá determinar la dirección y el sentido del mismo.

4.2.2.3 Mapa del entorno de trabajo

Una vez identificado el número de robots presentes en el entorno de trabajo se procede a representar en un mapa virtual la situación en tiempo real. Se busca que en la representación virtual del entorno se puedan identificar las áreas ocupadas por los diferentes robots y la dirección y sentido según están orientados los robots. En la figura 4.9 podemos ver dos etiquetas que corresponderían a cada uno de los robots presentes en el entorno de trabajo. Posteriormente en la figura 4.10 vemos en un mapa virtual un círculo representando a cada una de las subetiquetas del color característico de cada robot y conectados entre sí los correspondientes al mismo robot. También se representa un punto indicando el centroide del robot y una recta desde el mismo hasta la etiqueta más alejada para representar la dirección sobre la cual está el robot así como su sentido. Finalmente en la figura 4.11 se representa un mapa virtual sobre el cual las áreas de color negro representan espacio ocupado dentro del entorno de trabajo así como los límites perimetrales del mismo.

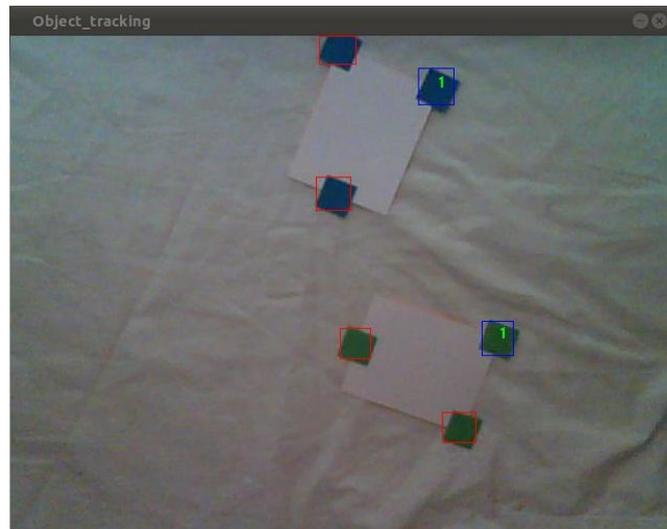


Figura 4.9: Simulación de etiquetas correspondientes a dos robots

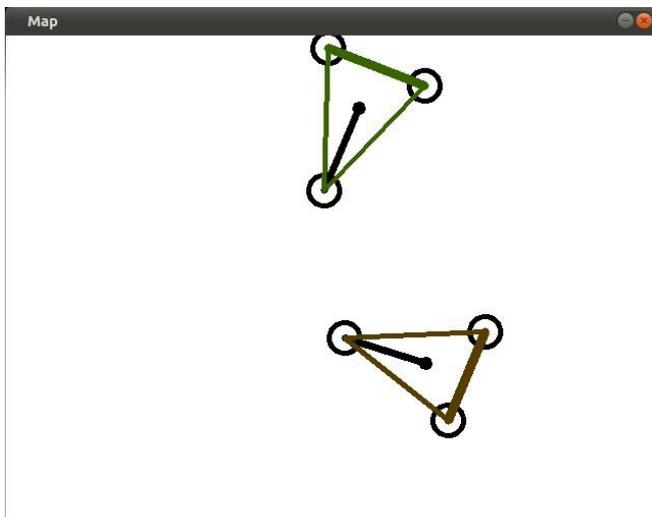


Figura 4.10: Mapa del entorno de trabajo para ver la dirección y sentido según los cuales están orientados los robots.

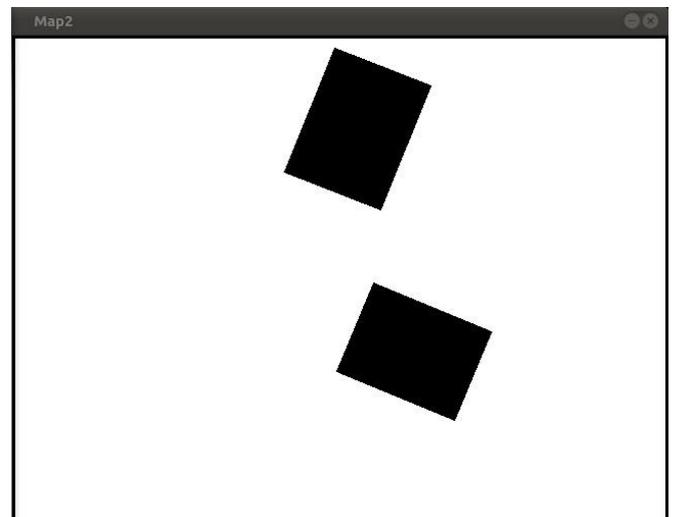


Figura 4.11: Representación del espacio ocupado por los robots en el entorno de trabajo.

Para sacar la dirección y el sentido del robot lo que se hace es unir mediante una recta el centroide con la subetiqueta más alejada de este, de esta manera obtenemos la dirección del robot. El sentido según el cual está orientado el robot corresponde, sobre la recta anterior, a la etiqueta más alejada del centroide ya que esta indica la parte delantera del mismo.

4.3 Robótica

La parte de robótica se ha centrado en el diseño, la construcción y la programación de los robots que estarán en el entorno de trabajo. Los requisitos que imponemos al diseño del robot tienen que ser acordes a las especificaciones que detallamos en el primer capítulo del trabajo, económicamente viable y sencillo. Estas especificaciones no están reñidas con otras importantes que impondremos como son:

- Robustez: El robot tiene que ser lo suficientemente robusto para soportar el peso de los componentes electrónicos y ser capaz de soportar pequeños impactos producidos por posibles colisiones en sus desplazamientos.
- Ligereza: El robot tiene que ser lo suficientemente liviano para que los componentes motrices del robot no sufran un esfuerzo excesivo y no haya un desgaste energético elevado.

Teniendo en cuenta que necesitamos para la construcción un material ligero y robusto la mejor solución es el plástico. Fácilmente moldeable y con un bajo coste de producción posee una baja densidad que nos permitirá reducir al máximo el peso de nuestros robots.

Una vez que sabemos con qué material diseñaremos nuestro robot en el siguiente apartado veremos cómo lo construiremos.

4.3.1 Diseño

Tras sopesar distintos posibles modelos de diseño para el robot decidimos basarnos en el diseño del profesor Alberto Valero para el seminario de máster de robótica ya que además de ser un diseño simple, consta de un chasis de una sola pieza que aumenta la robustez. El diseño completo está en Thingiverse [9], página dedicada a compartir piezas de diseño digitales bajo la licencia GNU.

El chasis como sabemos es estructuralmente una de las partes más importantes del robot ya que compone el esqueleto del mismo. El siguiente diseño del chasis es idóneo para nuestro trabajo ya que está especialmente pensado para un robot de dos ruedas y un punto de apoyo móvil como podemos ver en la figura 4.12.

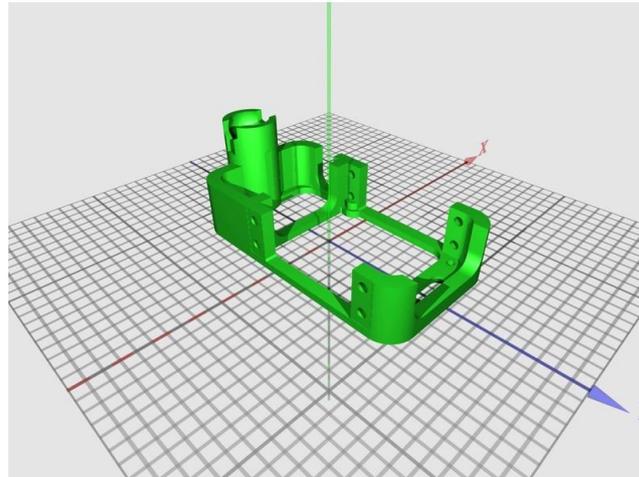


Figura 4.12: Chasis del robot

El diseño de las ruedas está pensado conjuntamente con el chasis anterior para que la altura sea la adecuada. Cabe destacar la existencia de un canal en el canto de la rueda donde posteriormente se colocará una junta tórica para facilitar la tracción del robot como podemos ver en la figura 4.13.

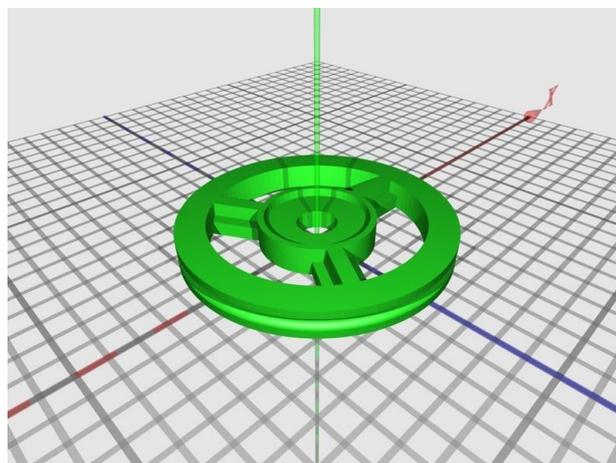


Figura 4.13: Rueda del robot

Necesitaremos para cada robot que construyamos un chasis y dos ruedas. Para crear estas piezas recurriremos a las impresoras 3D. Una impresora 3D es una máquina capaz de realizar impresiones de diseños tridimensionales, creando piezas volumétricas a partir de un diseño hecho por ordenador. Resultan extremadamente útiles porque permiten prototipar piezas reales a partir de diseños virtuales (archivos CAD).

Existe un grupo en la Universidad Carlos III de Madrid para desarrollar impresoras 3D Open-Source [10]. El objetivo de este grupo es el desarrollo de robots avanzados mediante la tecnología de impresión 3D Open-Source y uno de los proyectos con más éxito es el proyecto Clone Wars [11]. Este proyecto consiste en replicar impresoras 3D construyéndolas con piezas previamente diseñadas e impresas por otras impresoras 3D.

La creación de tanto el chasis como las ruedas de los robots que se han usado para este trabajo están impresas en las impresoras 3D propiedad de la Universidad Carlos III de Madrid.



Figura 4.14: Impresora MADRE

Nombre: MADRE

Modelo: Thing-o-Matic de la empresa Makerbot

Propiedad de: Asociación de Robótica de la UC3M



Figura 4.15: Impresora PADRE

Nombre: PADRE (a.k.a UC3-PO)

Modelo: Thing-o-Matic de la empresa Makerbot

Propiedad de: Departamento de Ingeniería de Sistemas y Automática

4.3.2 Electrónica

La parte de la electrónica de los robots estará desarrollada sobre la plataforma Arduino y módulos compatibles que veremos a continuación.



Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar [12].

En el mercado hay infinidad de microcontroladores sobre los cuales se pueden desarrollar los programas necesarios para el control de un robot simple. Muchos de ellos ofrecen funcionalidades similares pero el problema reside en que no todos son igual de fáciles de controlar. Arduino, además de simplificar el proceso de trabajar con microcontroladores, ya que organiza el trabajo de programar en paquetes fáciles de usar, ofrece otras ventajas con respecto al resto del mercado:

- Asequible: Las placas Arduino son más asequibles comparadas con otras plataformas de microcontroladores. Recordemos que uno de los objetivos de

nuestro trabajo es que sea económicamente viable.

- **Multi-plataforma:** Mientras que la mayoría de los entornos para microcontroladores están limitados a Windows el software de Arduino funciona en los sistemas operativos Windows, Mac OSX y Linux.
- **Entorno de programación simple y directo:** Basado en el entorno de programación Processing es fácil de usar para principiantes y lo suficientemente flexible para usuarios avanzados.
- **Software ampliable y de código abierto:** El software Arduino esta publicado bajo una licencia libre y preparado para ser ampliado por programadores experimentados. El lenguaje puede ampliarse a través de librerías de C++, y si se está interesado en profundizar en los detalles técnicos, se puede dar el salto a la programación en el lenguaje AVR C en el que está basado. De igual modo se puede añadir directamente código en AVR C en tus programas si así lo deseas.
- **Hardware ampliable y de código abierto:** Arduino está basado en los microcontroladores ATMEGA168, ATMEGA328 y ATMEGA1280. Los planos de los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores de circuitos con experiencia pueden hacer su propia versión del módulo, ampliándolo u optimizándolo. Incluso usuarios relativamente inexpertos pueden construir la versión para placa de desarrollo para entender cómo funciona y ahorrar algo de dinero.

4.3.2.1 Placa de Control

La placa de control constituye el cerebro de nuestro robot ya que es la encargada de interpretar toda la información proveniente de los sensores del robot y es capaz de ejecutar las órdenes transmitidas por el programa de una forma rápida y eficiente. Para la elección de la placa necesitamos una que nos proporcione el número suficiente de entradas y salidas, tanto analógicas como digitales para poder conectar todos los sensores y componentes motrices de los que consta el robot.

La unidad de control Arduino UNO, cuyo microcontrolador es el Atmega328P, resultaba perfecta, a nivel de comunicaciones y entradas/salidas, para los requisitos que queremos imponer a nuestro robot.



Figura 4.16: Arduino UNO (Parte delantera)

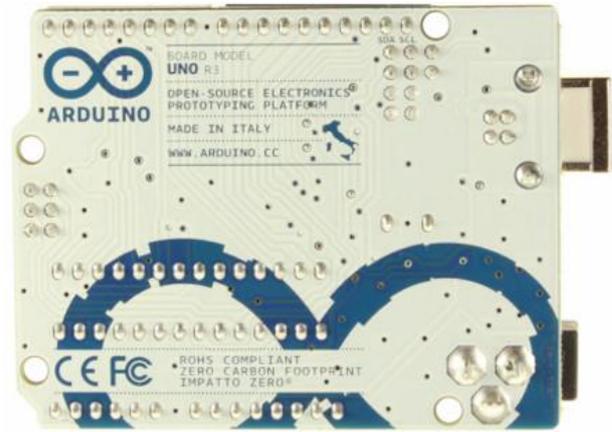


Figura 4.17: Arduino UNO (Parte trasera)

4.3.2.2 Servos

En el capítulo anterior se explicó el funcionamiento básico de un servo según el cual, mediante una señal de control, se podía determinar el grado de giro del eje. Sin embargo, este no es el funcionamiento que deseamos para los servos que van a ir instalados en nuestro robot. Nosotros vamos a querer que el giro no esté limitado a 180° si no que el giro sea completo para poder permitir que el robot avance sin problemas.

Es inevitable que surja la cuestión de por qué no se usan motores de corriente continua estándar para nuestra aplicación. Como sabemos los motores de corriente continua (Motores CC) son muy fáciles de controlar ya que disponen únicamente de dos cables donde se aplica un voltaje. A mayor voltaje mayor velocidad y la polaridad aplicada determina el sentido de giro. Sin embargo el problema que presentan estos motores es que giran demasiado rápido y tienen poca fuerza, lo que los hace poco adecuados para la construcción de robots móviles. Es necesaria la presencia de una caja reductora con un eje de salida que gire más lentamente pero con mayor fuerza. Ya que la construcción de una caja reductora es complicada la solución más utilizada en el mundo de la robótica es recurrir a los servomotores. Los servos incorporan un motor CC, una caja reductora y un circuito de control. La problemática de la limitación en el giro se resuelve “trucando” el servo, eliminando el circuito de control y cortando los toques mecánicos de manera que se comporten como motores CC normales pero con más fuerza.

El truco de los servos consiste en una parte hardware y una parte software [13]. En la parte hardware se inutiliza el potenciómetro existente dentro del servo bloqueándolo de tal manera que el circuito de control vea que está en la posición intermedia. El circuito de control sabe cuál es la posición del eje gracias a la lectura que

efectúa de este potenciómetro, estabilizándolo en la posición intermedia hacemos que el circuito crea que el cursor del potenciómetro está en el medio. Una vez hecho esto quitamos los topes mecánicos de los engranajes que limitan el movimiento de 0° a 180° . En la parte software diremos al servo que queremos que vaya desde la posición inicial, correspondiente a 90° , hasta la posición de 0° o 180° . Como el valor del potenciómetro no variará, el motor intentará ir en una u otra dirección y al no alcanzar el tope seguirá sin pararse como si fuera un motor de continua.

En la siguiente figura podemos ver las partes de las que se compone un servo:

a) carcasa; b) motor DC; c) potenciómetro; d) circuito de control; e) tren reductor; f) brazo (elemento terminal en el eje).

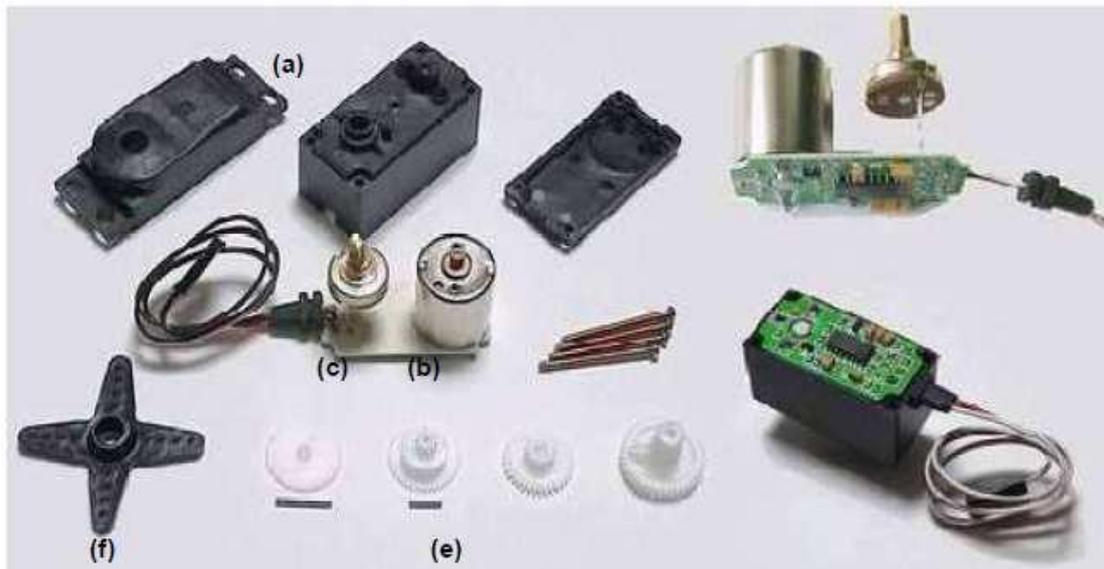


Figura 4.18: Partes de un servo

4.3.2.3 Módulos de radiofrecuencia XBEE

Los módulos Xbee de MaxStream permiten a una placa Arduino comunicarse de forma inalámbrica creando enlaces seriales de señales TTL en distancias de 30 metros en interiores, 100 metros en exteriores con línea de vista y hasta 1.5 Km con los módulos Pro.

Los módulos XBee utilizan el protocolo IEEE 802.15.4 mejor conocido como ZigBee. Este protocolo se creó pensando en implementar redes de sensores. El objetivo es crear redes tipo mesh que tengan las propiedades de auto-recuperación y bajo consumo de energía. Las áreas de aplicación son múltiples:

- Entretenimiento en casa y control: Seguridad, iluminación inteligente, control de temperatura, contenidos multimedia.
- Hogar inteligente: Sensores de agua, sensores de potencia, electrodomésticos inteligentes, sensores de acceso.
- Servicios Móviles: pagos móviles, monitorización y control móvil, seguridad y control de acceso, teleasistencia.
- Edificios inteligentes: monitorización y control de energía, HVAC, iluminación y control de acceso.
- Plantas industriales: control de procesos, gestión ambiental, gestión de energía y control de dispositivos industriales.

En nuestro caso particular vamos a utilizar los módulos de XBee para crear una comunicación serial inalámbrica entre un ordenador y un Arduino. Cabe destacar que los módulos Xbee no se conectan directamente al Arduino si no que la conexión se hace a través de un shield que hace una adaptación entre el módulo y el Arduino.

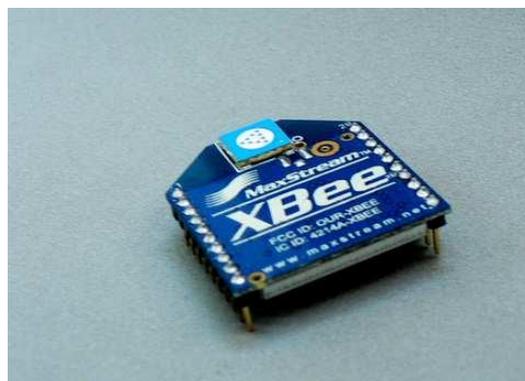


Figura 4.19: Módulo XBee de MaxStream

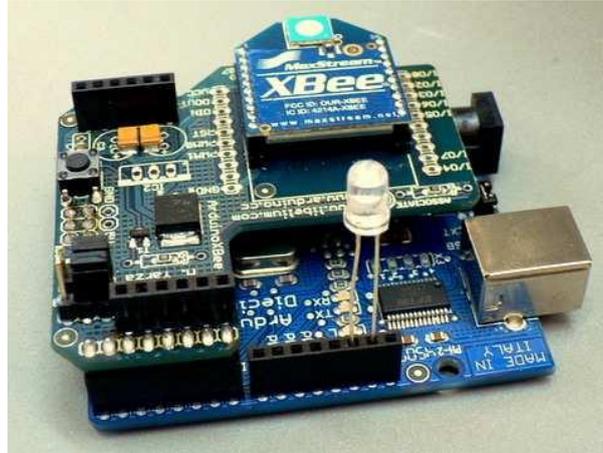
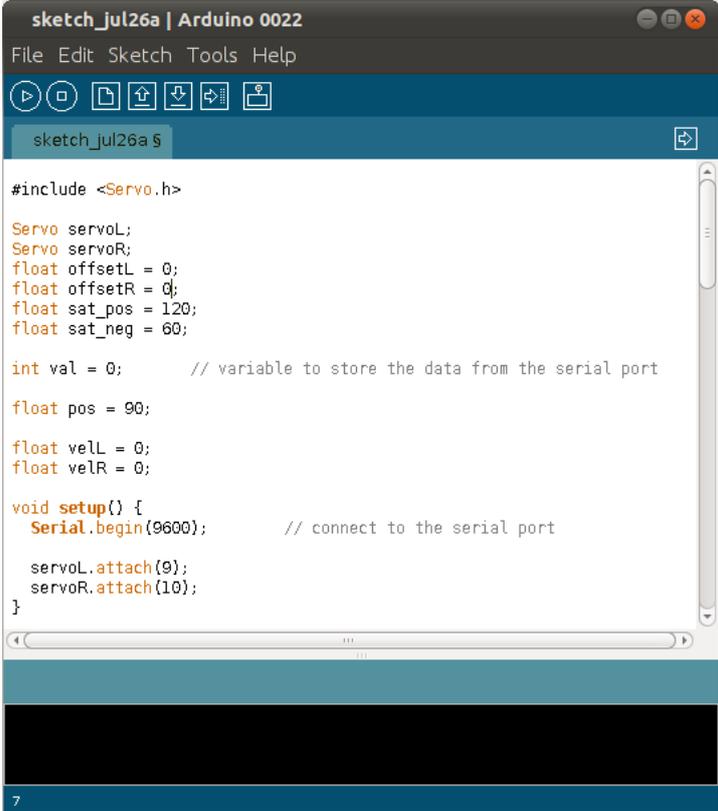


Figura 4.20: XBee Shield montado sobre la placa Arduino

Para el trabajo utilizaremos tres módulos XBEE y dos XBEE Shield. Un módulo XBEE estará conectado al ordenador directamente mediante un adaptador USB y desde este será desde el cual se mandarán las instrucciones a la flota. En la configuración del módulo lo programaremos para que envíe en la dirección broadcast, esto es, para todas las direcciones. Será en el propio robot donde discriminaremos si las instrucciones tenemos que interpretarlas o no porque corresponden a otro robot. Cada robot llevará un módulo XBEE montado sobre el XBEE Shield y sobre la placa Arduino UNO. A continuación explicaremos cómo programaremos cada robot para que responda individualmente a las instrucciones.

4.3.3 El programa

El programa desarrollado en lenguaje Arduino y que será cargado en la placa de control Arduino UNO incorporada en cada uno de los robots servirá para teleoperarlo desde el ordenador por radiofrecuencia. Haciendo uso del puerto serie, se mandarán comandos básicos de movimiento para controlar en tiempo real los robots individualmente. El Arduino al recibir las órdenes a través del módulo Xbee las interpretará directamente como si vinieran del puerto serie del ordenador.



```

sketch_jul26a | Arduino 0022
File Edit Sketch Tools Help
sketch_jul26a S
#include <Servo.h>

Servo servoL;
Servo servoR;
float offsetL = 0;
float offsetR = 0;
float sat_pos = 120;
float sat_neg = 60;

int val = 0; // variable to store the data from the serial port

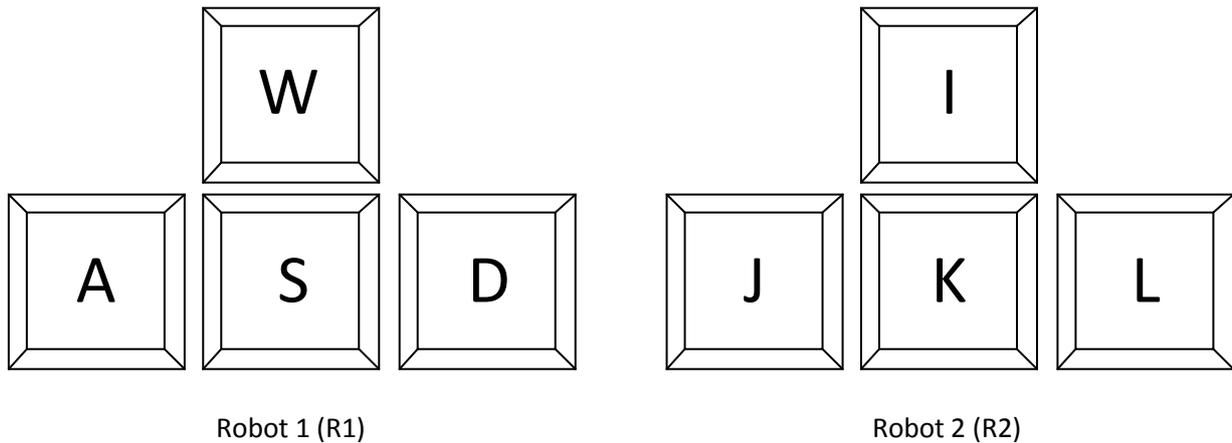
float pos = 90;

float velL = 0;
float velR = 0;

void setup() {
  Serial.begin(9600); // connect to the serial port
  servoL.attach(9);
  servoR.attach(10);
}
  
```

Figura 4.21: Interfaz de programación Arduino

Al haber dos robots es necesario poder controlarlos individualmente. Por ellos se establecen unos controles individuales y unos comandos comunes para la flota. Los comandos de movimiento son los siguientes.



Tecla	Acción
W	R1 adelante
A	R1 gira a la izquierda
S	R1 atrás
D	R1 gira a la derecha

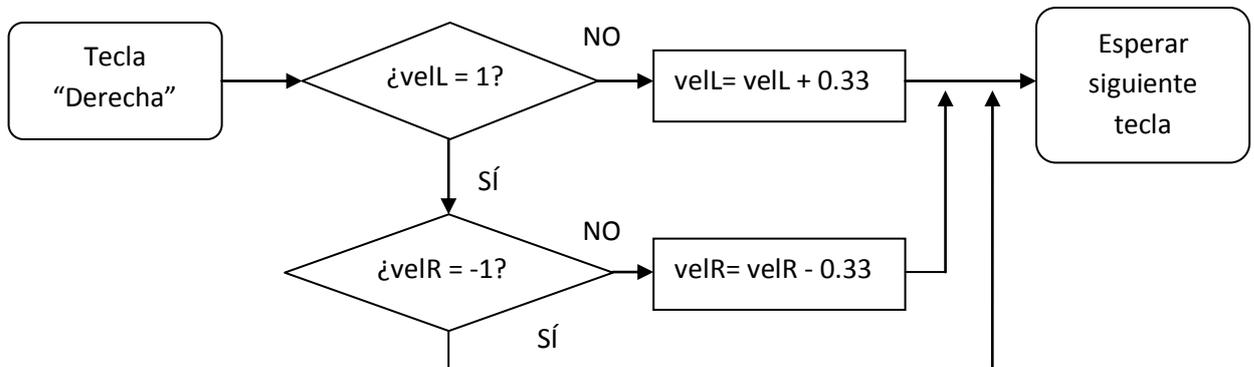
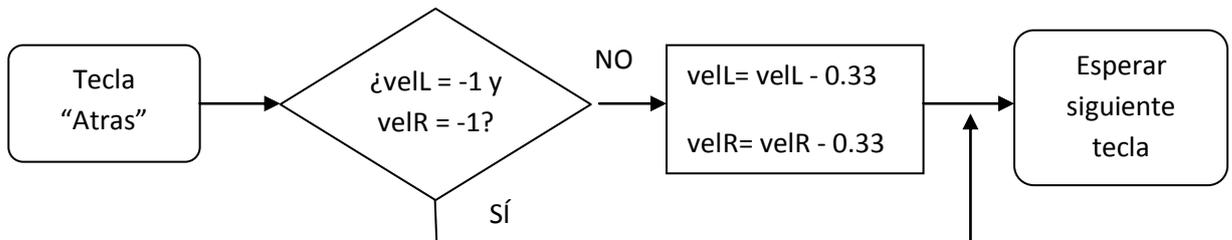
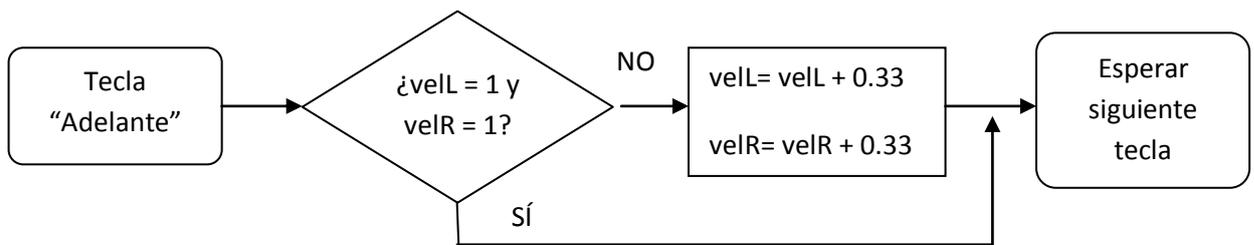
Tecla	Acción
I	R2 adelante
J	R2 gira a la izquierda
K	R2 atrás
L	R2 gira a la derecha

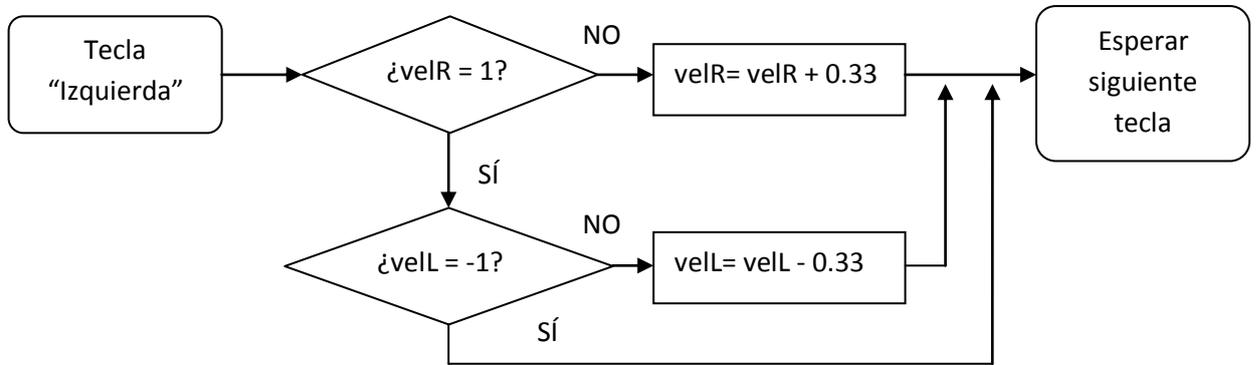
Controles comunes	
Tecla	Acción
X	R1 y R2 adelante (máxima velocidad)
C	Parar R1 y R2
V	R1 y R2 atrás (máxima velocidad)

Cabe destacar que los controles individuales son incrementales con 3 grados, esto es, suponiendo que el robot 1 está parado y presionamos la tecla “W” el robot irá para adelante a una determinada velocidad. Si volvemos a presionar la misma tecla el robot incrementará su velocidad y si repetimos la acción alcanzará el máximo de velocidad. Si queremos hacerlo parar tendremos que ir decrementando la velocidad en tres pasos con la tecla “S”. Al tener únicamente dos ruedas los comandos de giro se traducen en que un servomotor gira en una dirección y el otro servomotor está quieto o gira en la dirección contraria. Podemos ver como se traducen entonces estos comandos básicos de movimiento en las velocidades de cada uno de los servos en los siguientes esquemas.

Cada uno de los servos L (“Left”) y R (“Right”) tiene por tanto 7 posible estados con respecto a su velocidad. Denotaremos con $velL$ y $velR$ la velocidad de cada uno de los servos respectivamente pudiendo tomar cada uno de ellos los siguientes valores.

Valor	Acción
0	Servo parado
0.33	Velocidad mínima avanzando
0.66	Velocidad media avanzando
1	Velocidad máxima avanzando
-0.33	Velocidad mínima retrocediendo
-0.66	Velocidad media retrocediendo
-1	Velocidad máxima retrocediendo





4.3.4 El robot final

A continuación podemos ver el resultado final después del montaje completo de los robots. En la primera imagen se pueden ver los dos robots montados. En la figura 4.23 podemos ver una vista lateral y finalmente en la figura 4.24 se puede ver la parte inferior del robot donde se pueden identificar los dos servomotores, la pila de alimentación y la canica, que encajada en la estructura proporciona el tercer punto de apoyo al mismo.

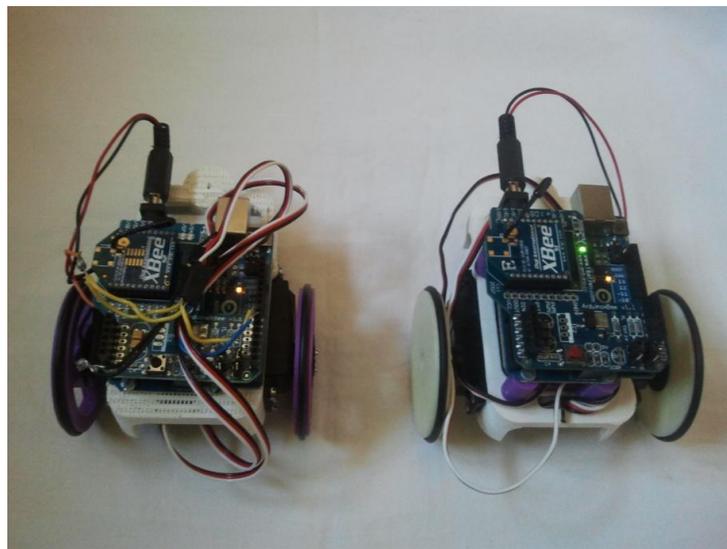


Figura 4.22: Robots

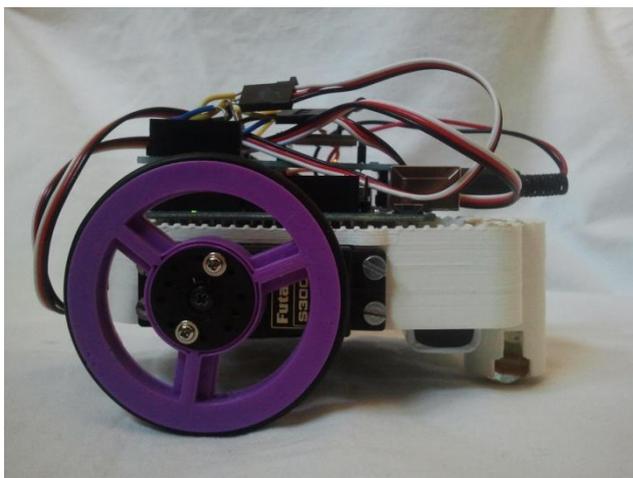


Figura 4.23: Vista lateral

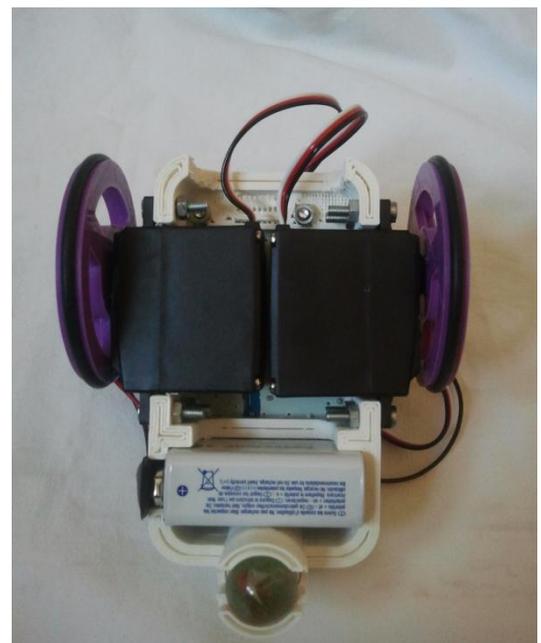


Figura 4.24 Vista desde abajo

5. Resultados y evaluación

5.1 Introducción

En este capítulo se detallará la plataforma creada y se presentarán los resultados obtenidos de la experimentación tanto en la parte de visión como en la parte de robótica. Con el fin de desarrollar una plataforma de investigación se intentarán simular unas condiciones de naturaleza estática para que los resultados sean lo más fiables posibles.

5.2 Condiciones y entorno de trabajo

Las etiquetas usadas para la identificación de cada robot son rectangulares, para adecuarse a la vista de planta del mismo y que se superpongan sobre el espacio real que ocupa el mismo. Las propias etiquetas están compuestas por tres etiquetas secundarias de color, que sobre la principal de fondo blanco servirán para identificar tanto cuál es el robot así como la dirección y sentido sobre el que está orientado. Hemos elegido este tipo de etiquetas por su simplicidad y porque nos valen para obtener las características de modelado del entorno que buscamos.

Los colores de las subetiquetas se han elegido de tal manera que dentro del cono HSV (espacio de color en el que trabajaremos) estén lo más distantes posible entre sí. De esta manera nos aseguramos de que exista una equidistancia entre los colores que favorecerá luego en la segmentación minimizando los errores por confusión de etiquetas.

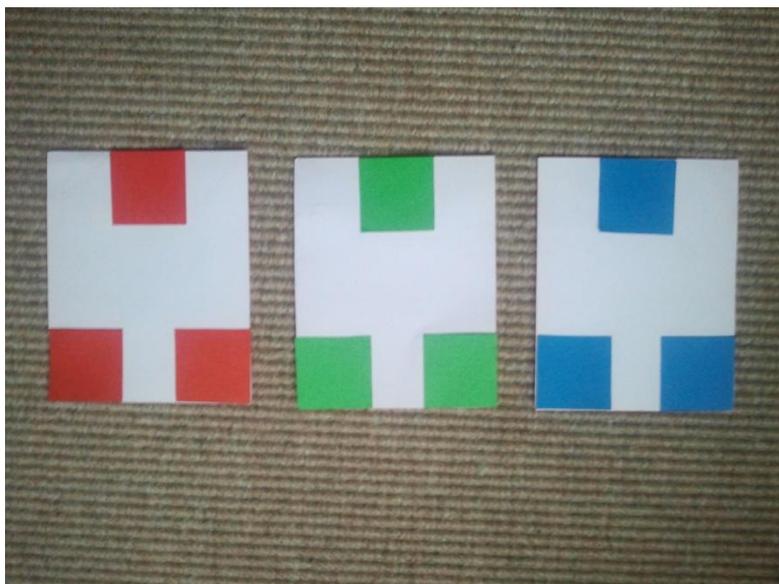


Figura 5.1: Etiquetas de distintos colores

Surgieron problemas trabajando con las etiquetas anteriores ya que la representación en el mapa virtual después de tratar las imágenes no reflejaba correctamente el espacio ocupado por los robots como podemos ver en la figura 5.4. Esto es porque el algoritmo usa el centro de la subetiqueta de color para determinar el límite del mismo cuando en realidad no es así, por tanto el rectángulo negro que debería representar el espacio ocupado por el robot era más pequeño de lo que en realidad debería ser. Esto tendría consecuencias ya que en aplicaciones futuras donde se use esta información habría parte del robot en un área en el que a priori parece libre.



Figura 5.2: Detección de etiquetas iniciales



Figura 5.3: Mapa virtual representando dirección y sentido de los robots con las etiquetas iniciales

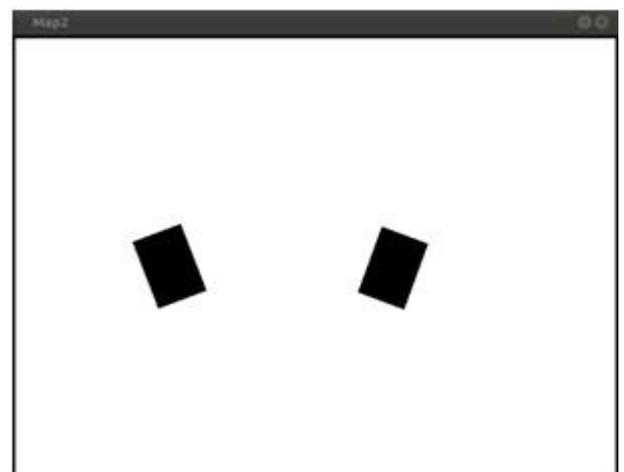


Figura 5.4: Mapa virtual representando el espacio ocupado por los robots con las etiquetas iniciales

Para solventar este problema se decide el diseño de unas etiquetas nuevas donde el centro de la subetiqueta de color sí que se corresponde con el límite real del robot como podemos ver en la siguiente figura.

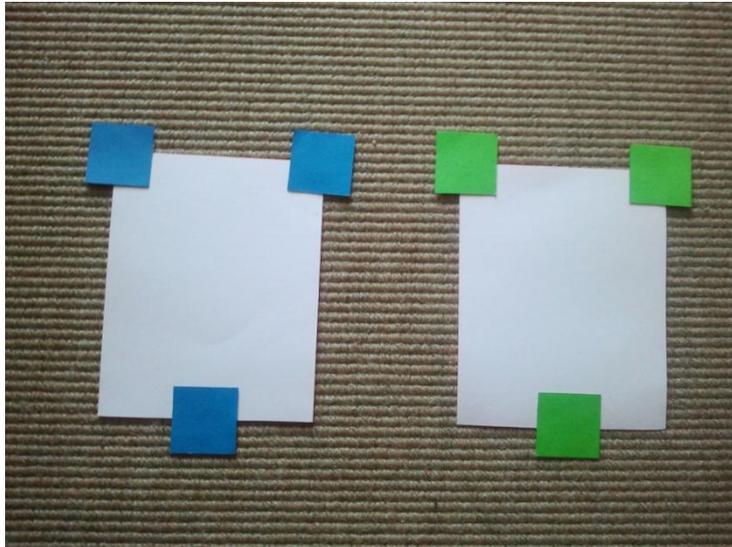


Figura 5.5: Etiquetas finales

Las pruebas realizadas se han hecho situando la cámara a una altura de 80 cm e intentando que esté en una línea perpendicular a la línea de suelo. Se cubre de esta manera un área de $60 \times 50\text{ cm}$.

Basándonos en el modelo “Pin Hole” y dado que desconocemos el tamaño de un pixel en el plano de la imagen de la cámara, esto es, el tamaño de un pixel en la célula fotodetectora situada dentro de la cámara, vamos a obtener matemáticamente el tamaño del pixel para poder hacer la conversión lineal de las coordenadas de la imagen a coordenadas reales en el área de trabajo. Aplicando las siguientes ecuaciones, y sabiendo que a una altura $x_3 = 0.80\text{ m}$, se cubre un área de $60 \times 50\text{ cm}$ y la resolución de la cámara es de 640×480 píxeles, averiguamos el tamaño de un pixel en el plano de la imagen. La distancia focal $f = 4\text{ mm}$ la obtenemos de las especificaciones de la webcam.

$$y_1 = f \frac{x_1}{x_3} \quad y_2 = f \frac{x_2}{x_3}$$

Tamaño horizontal

$$y_1 = 4\text{ mm} \frac{60\text{ mm}}{80\text{ mm}} = 3\text{ mm} ; y_{1\text{ pixel}} = \frac{3\text{ mm}}{640\text{ píxeles}} = 4.69\text{ um/pixel}$$

Tamaño vertical

$$y_2 = 4mm \frac{50 mm}{80mm} = 2.5mm ; y_2_{pixel} = \frac{2.5 mm}{480 píxeles} = 5.2 um/pixel$$

Con estos datos podemos averiguar fácilmente en que coordenada real está cada uno de los robots. Como dato tenemos que el centroide del robot tiene las coordenadas de pixel (y_1, y_2) por tanto podemos sacar las coordenadas en el sensor:

$$y_{1\ sensor} = y_1 \cdot 4.69 um/pixel$$

$$y_{2\ sensor} = y_2 \cdot 5.2 um/pixel$$

Una vez que tenemos las coordenadas en el sensor se puede obtener directamente la posición real en el entorno experimental aplicando las fórmulas iniciales y despejando x_1 y x_2 .

En la figura 5.6 podemos ver los mapas resultado de aplicar el algoritmo de visión en un entorno experimental como el mostrado en la figura 5.7. En esta última figura podemos ver, aparte del resultado sobreimpresionado del etiquetado sobre la imagen real, las coordenadas del centroide tanto en píxeles como en distancia real. Recordemos que el centro de coordenadas está en el extremo superior izquierdo.

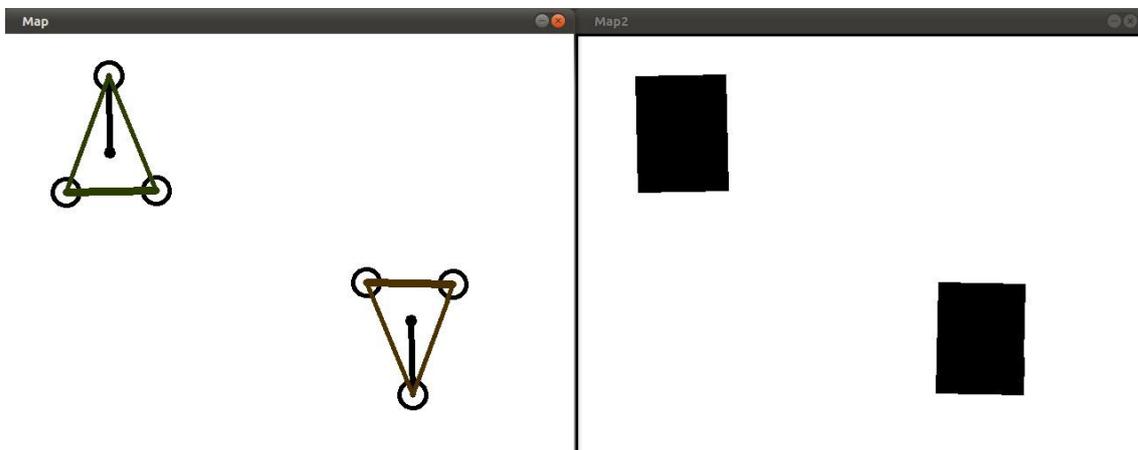


Figura 5.6 Mapas resultado de aplicar el algoritmo de visión

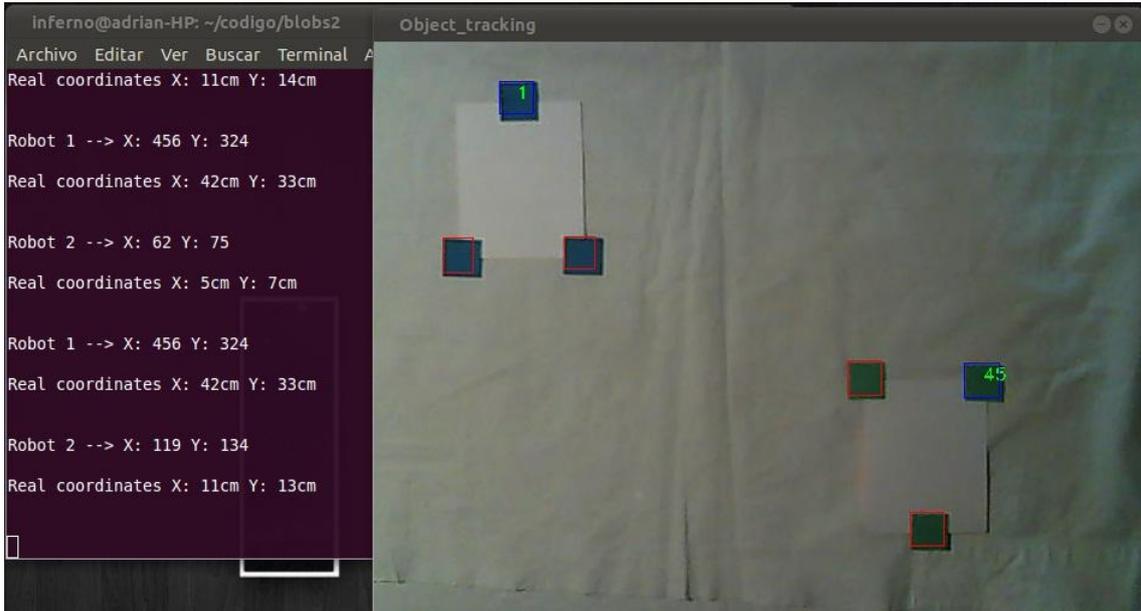


Figura 5.7: Coordenadas del centroide de cada robot

Los datos obtenidos mediante el cálculo teórico son aplicados directamente sobre el algoritmo para hacer el mapeado a coordenadas reales en el entorno experimental. Creímos necesario realizar pruebas para ver si se correspondían estos datos mostrados con las coordenadas reales. En la figura 5.8 podemos ver el mapa virtual de un entorno experimental con una etiqueta donde se puede apreciar el centroide, que es el punto del cual el algoritmo obtendrá las coordenadas.

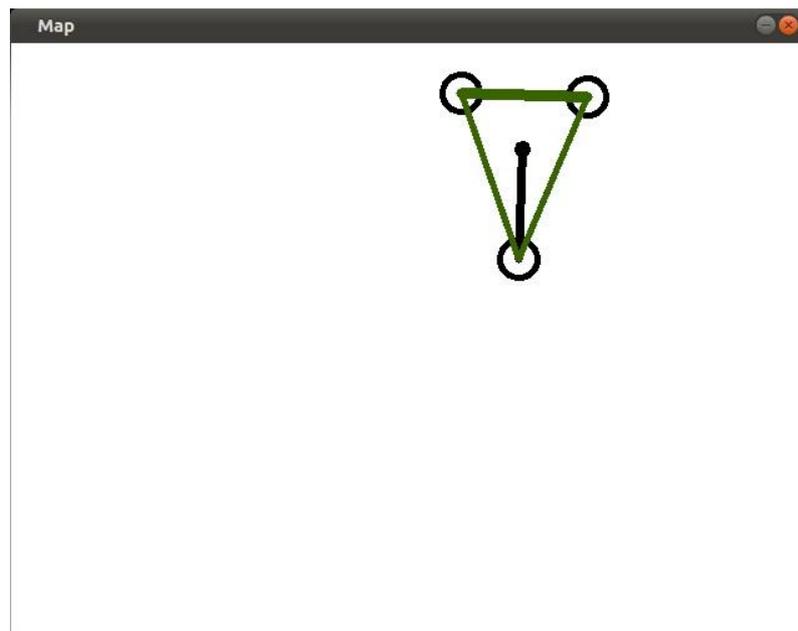


Figura 5.8: Mapa con centroide para la comprobación

Como podemos ver en las figuras 5.9 y 5.10 las coordenadas mostradas por consola y obtenidas a partir de los datos obtenidos anteriormente aplicados al algoritmo se aproximan bastante a las coordenadas reales.

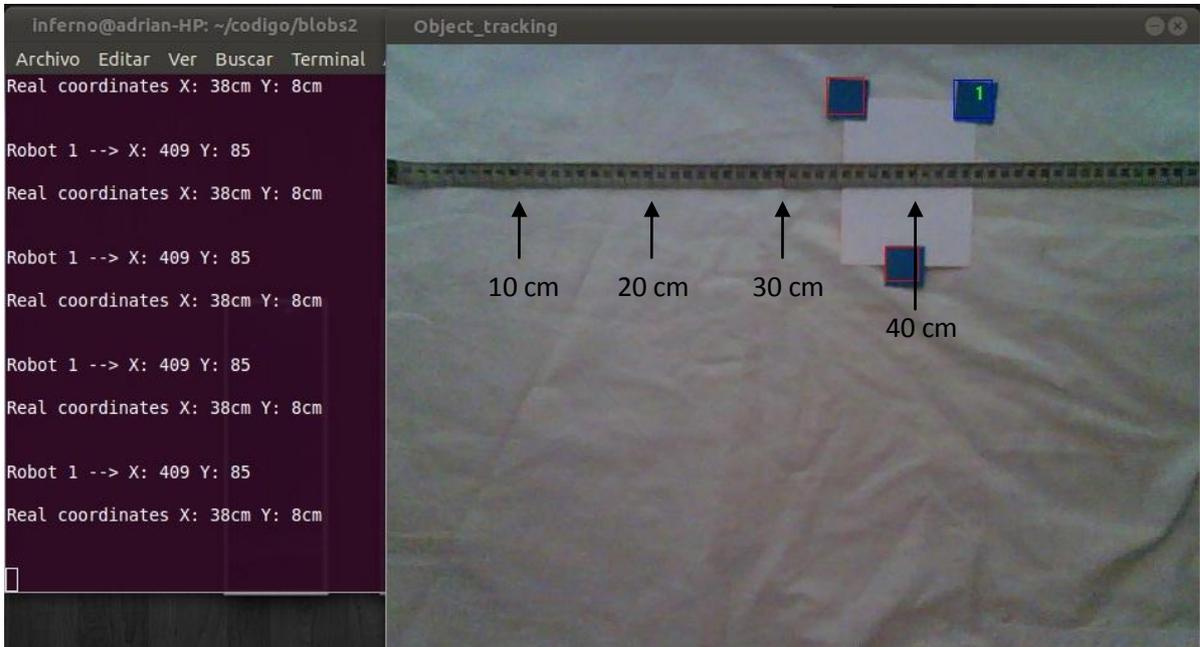


Figura 5.9: Comprobación distancia horizontal

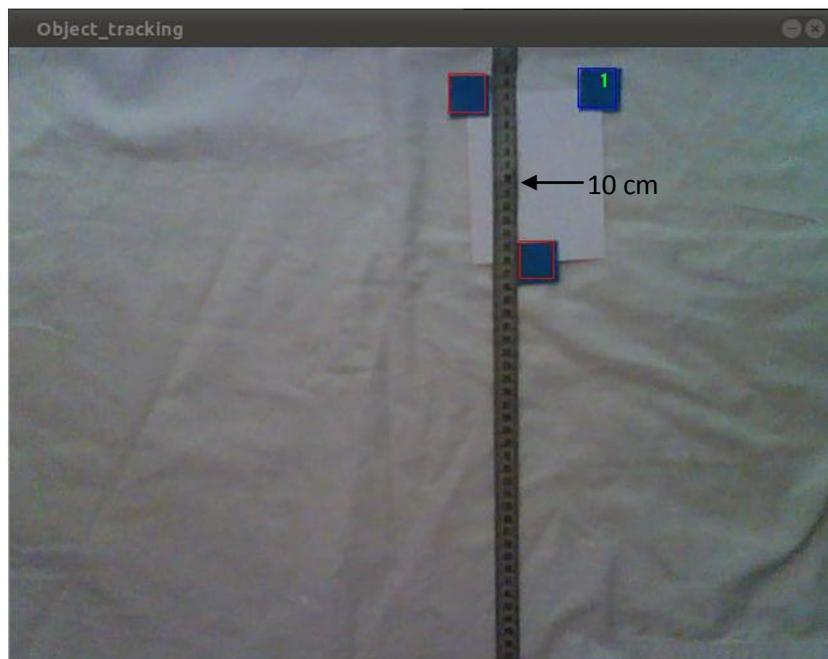


Figura 5.10: Comprobación distancia vertical

5.3 Calibración, segmentación y etiquetado.

Como hemos mencionado con anterioridad la iluminación es un factor muy importante en el proceso de visión. El simple hecho de realizar las pruebas en una habitación con ventanas puede derivar en que la iluminación varíe dependiendo la hora del día y que la calibración realizada con anterioridad no valga a continuación. Por ello creímos necesario un proceso de calibración en el algoritmo de visión para poder garantizar que los valores que se segmentaran corresponden al color específico medido en la calibración con unos márgenes de tolerancia ajustables. De la misma manera trabajar en entornos poco iluminados implica obtener imágenes de baja calidad, lo que conllevará a resultados erróneos con mayor facilidad. Lo ideal es que el sistema de iluminación usado derive en pocos brillos y se consiga una iluminación homogénea para conseguir buenos resultados. Para poder apreciar los efectos de una correcta iluminación vamos a ver los resultados de distintas calibraciones de un mismo color variando las condiciones de iluminación.

Todas las pruebas son realizadas con el mismo color para poder apreciar la variación en las mediciones realizadas por el programa en función de la iluminación. En la figura 5.11 podemos ver los resultados de una calibración en un ambiente con baja iluminación, los resultados se pueden ver en el margen izquierdo ya que son impresos en el terminal. En la figura 5.12 se pueden ver los resultados obtenidos para el mismo color en el mismo escenario pero con iluminación natural. Finalmente en la figura 5.13 se añade una iluminación artificial proporcionada por la lámpara de tecnología LED de fabricación propia descrita en el anterior capítulo.

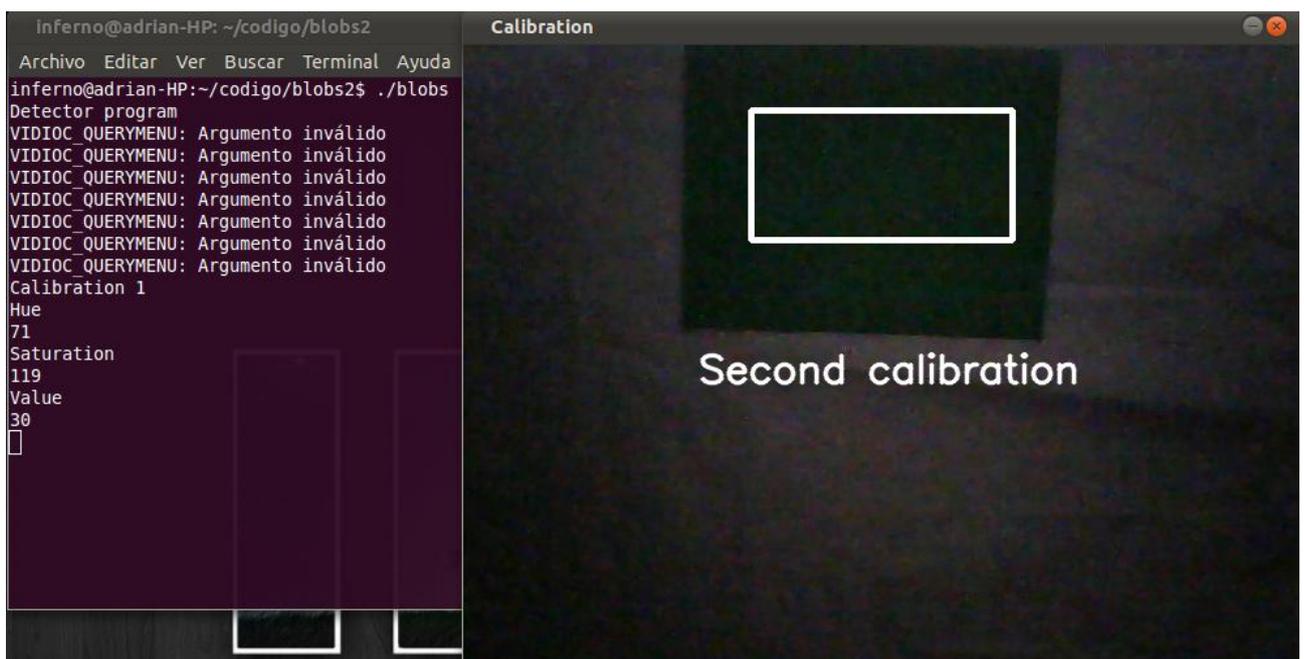


Figura 5.11: Calibración con baja iluminación

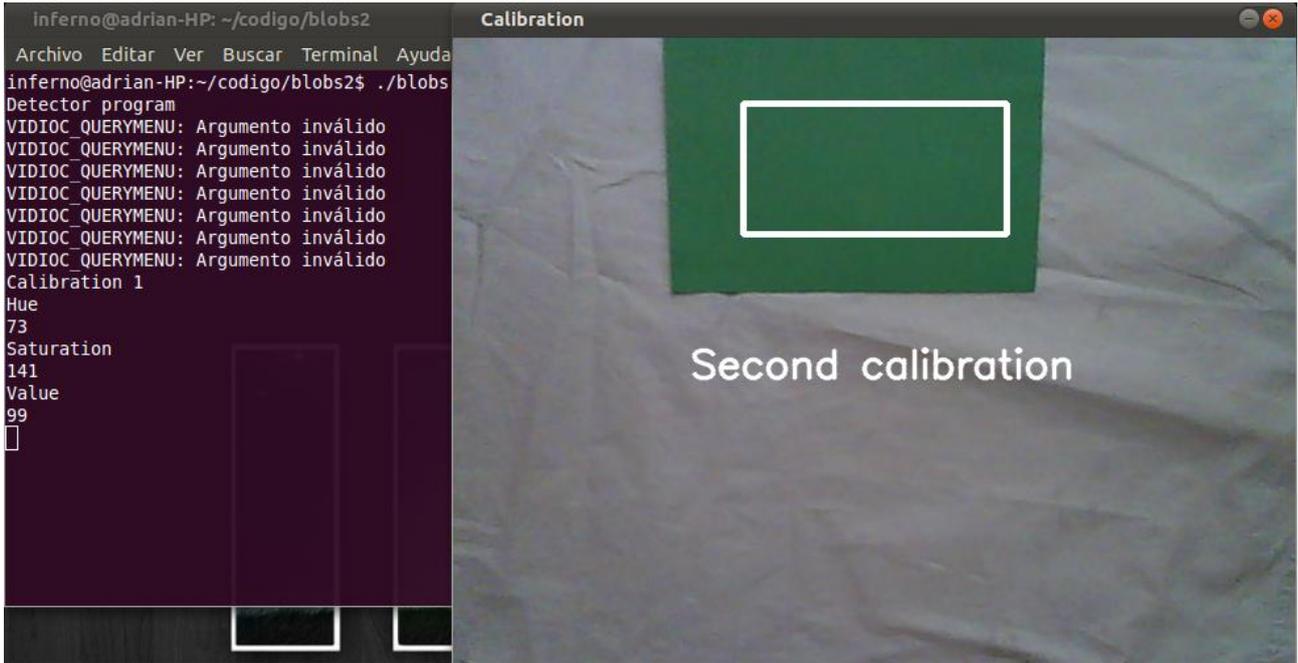


Figura 5.12: Calibración con iluminación natural (12 am)

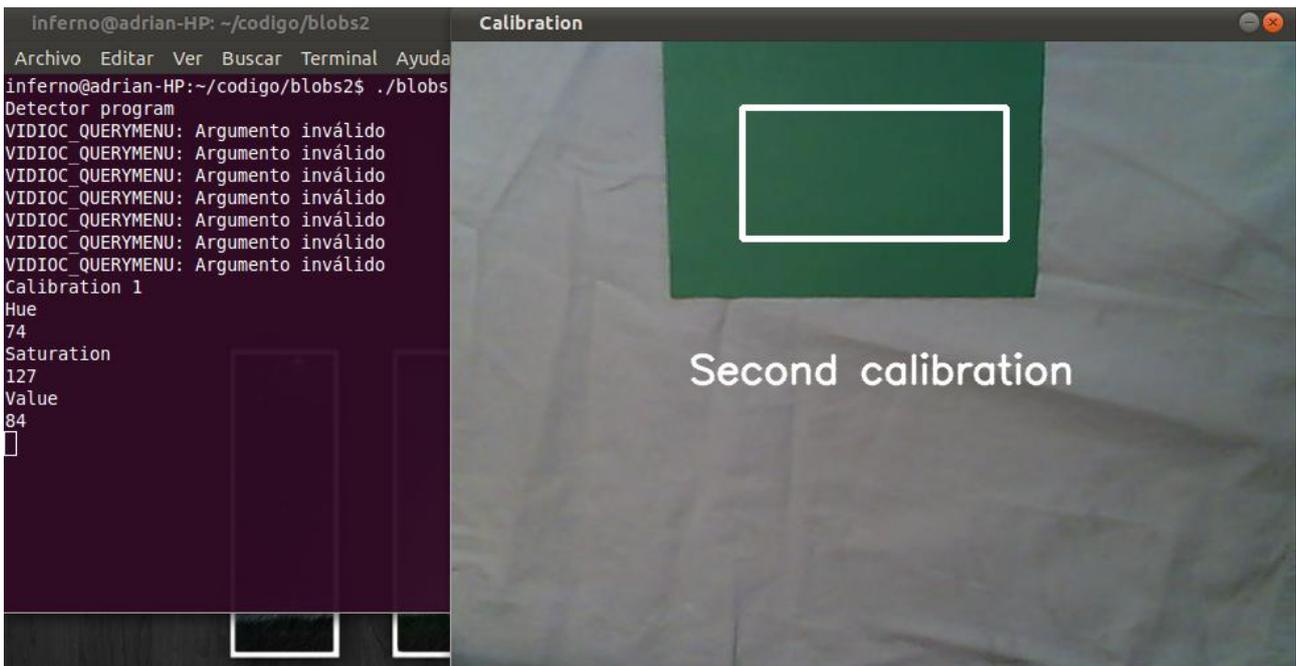


Figura 5.13: Calibración con iluminación natural (12 am) + iluminación artificial

Como podemos ver el efecto de la iluminación afecta notablemente a la intensidad (value) ya que esta es una medida de la cantidad de luz asociada al máximo de los tres valores RGB. Estos valores son los que se utilizarán posteriormente para segmentar los frames proporcionados por la webcam así que es de vital importancia realizar esta calibración inicial.

Una vez obtenidos los valores de calibración se segmenta cada frame de la imagen de acuerdo a los mismos, con unos márgenes de tolerancia que ayudan a minimizar el efecto de brillos y sombras permitiendo no perder información relevante. La segmentación de la imagen da lugar a imágenes binarizadas, una por cada color que se segmenta, y que posteriormente servirán para el etiquetado. Como hemos podido ver anteriormente la importancia de la iluminación era determinante en la primera parte del algoritmo. Esto continúa siendo así en el resto del mismo ya que si la iluminación es baja y no es homogénea, el proceso de segmentación y posteriormente el etiquetado no obtendrá buenos resultados. En la figura 5.14 podemos ver como el algoritmo no es capaz de identificar todas las etiquetas presentes en el entorno de trabajo. Las causas de esto las podemos ver en las dos figuras siguientes.



Figura 5.14: Entorno de trabajo con baja iluminación y resultados del etiquetado sobreimpresionados.

En las siguientes figuras podemos ver el resultado de la binarización para el color azul y verde respectivamente. La figura 5.15 corresponde a la segmentación del color azul. Como podemos ver aparte de las etiquetas hay presente ruido. Aunque este es mínimo y permite al algoritmo etiquetar correctamente, como podemos ver en la figura anterior, las condiciones de iluminación no son las idóneas. En la figura 5.16 vemos la segmentación correspondiente al color verde donde apenas se distinguen las etiquetas debido a la fuerte presencia de ruido en toda la imagen que dificulta el proceso de etiquetado. En el capítulo anterior, en las figuras 4.6, 4.7 y 4.8 pudimos ver los resultados en un ambiente con buena iluminación que representaría las condiciones ideales de trabajo donde la presencia de ruido es mínima en este paso.



Figura 5.15: Segmentación del color azul

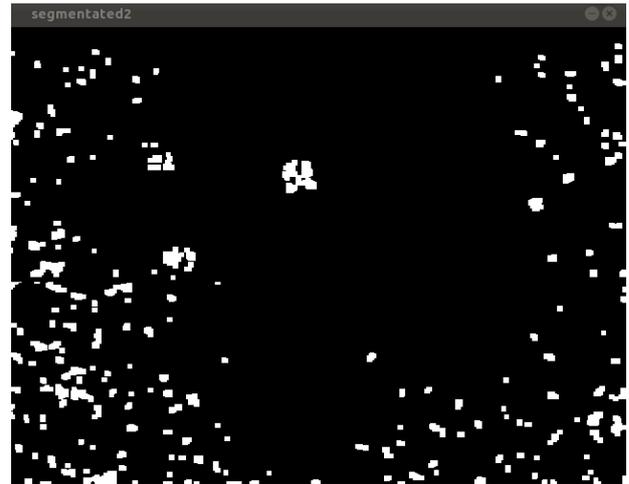


Figura 5.16: Segmentación del color verde

Una vez que el etiquetado es realizado se muestra sobre la imagen original el resultado así como dos mapas adicionales que representan el entorno de trabajo con la posición, dirección y espacio ocupado por los robots.

El análisis y tratamiento de cada frame individualmente y como secuencia continua dará lugar a que el proceso parezca inmediato para el usuario final pudiendo ver en tiempo real el resultado. En las siguientes figuras se puede apreciar 3 frames correspondientes a un movimiento de los dos robots en el entorno de trabajo y como el algoritmo es capaz de analizarlos y mostrar los resultados a medida que se van moviendo.

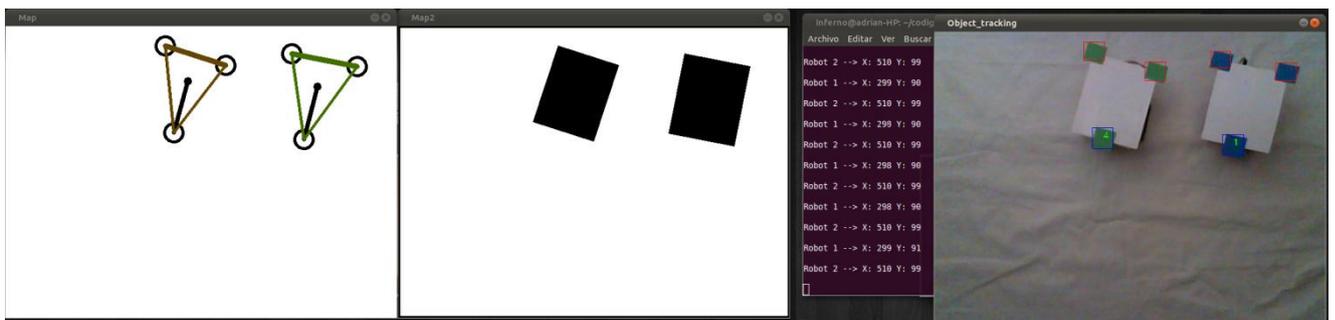


Figura 5.17: Frame 1 de secuencia de movimiento y resultados



Figura 5.18: Frame 2 de secuencia de movimiento y resultados

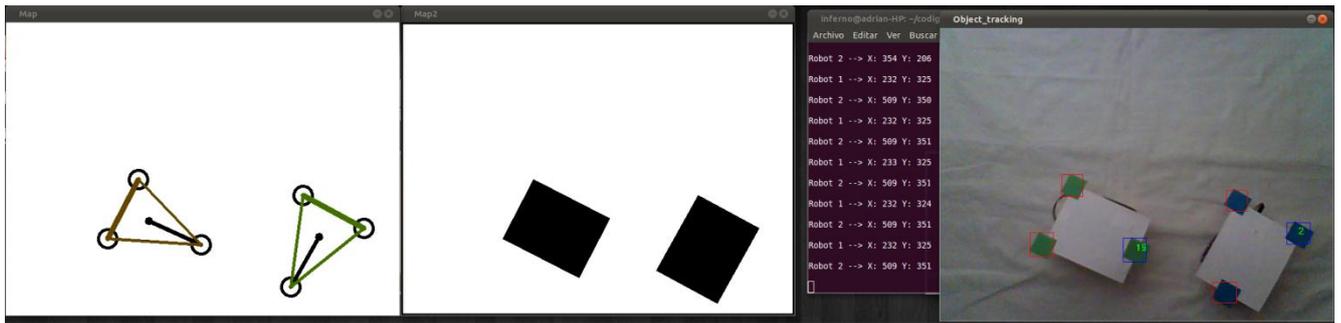


Figura 5.19: Frame 3 de secuencia de movimiento y resultados

Se mide el tiempo de procesamiento estableciendo un contador para medir la duración de cada iteración dentro del programa, esto es, para cada frame recibida de la cámara cuanto tiempo tarda el programa en procesar y analizar la misma para obtener los resultados.

En primer lugar vamos a ver el tiempo de procesamiento en el caso en el que hay dos etiquetas estáticas en el entorno de trabajo. Los tiempos obtenidos para el procesamiento de cada frame se pueden ver en el histograma de la figura 5.20.

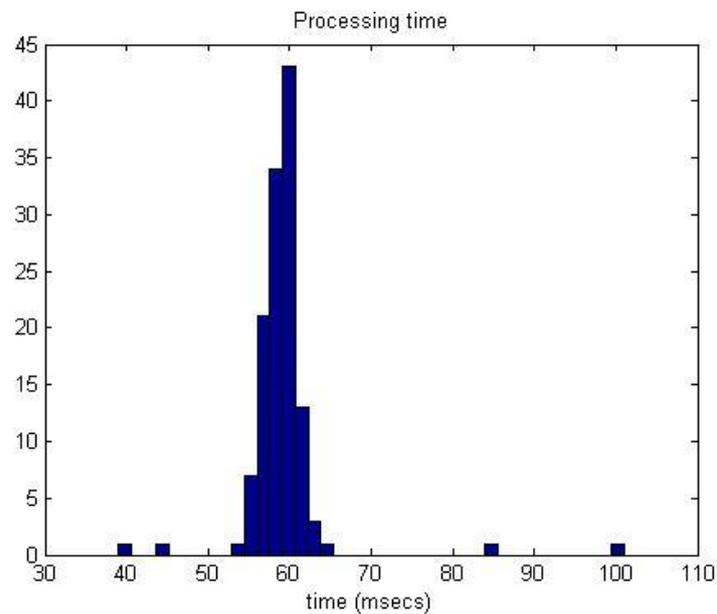


Figura 5.20: Tiempo de procesado para un entorno experimental con dos etiquetas estáticas

A continuación vamos a ver los tiempos obtenidos para el caso en el que una etiqueta se mantiene estática y la otra en movimiento, esto es, un robot parado y el otro moviéndose en el entorno de trabajo.

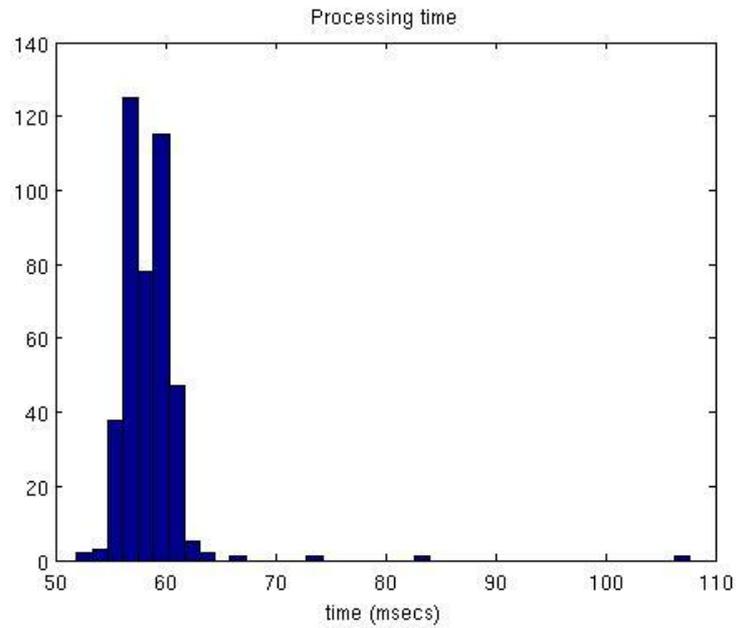


Figura 5.21: Tiempo de procesado para un entorno experimental con una etiqueta dinámica y otra estática

Finalmente vamos a ver la correspondiente a un entorno experimental con los dos robots moviéndose.

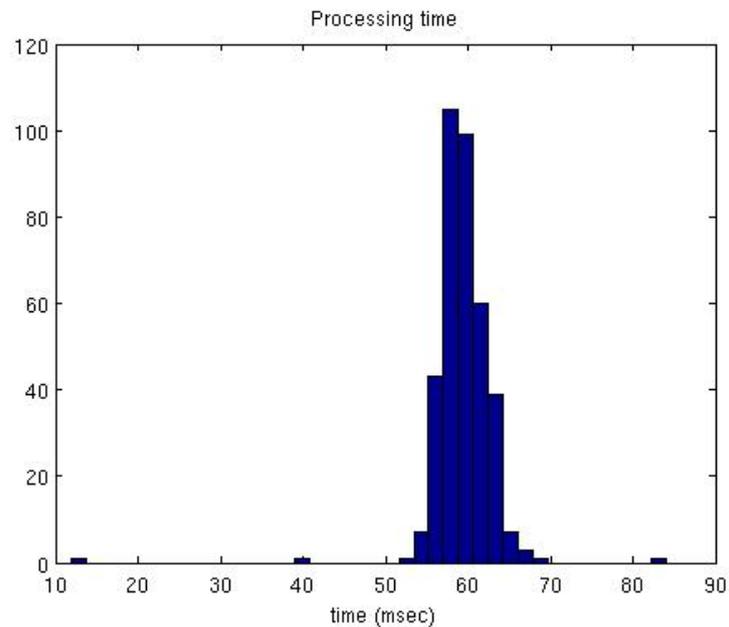


Figura 5.22: Tiempo de procesado para un entorno experimental con dos etiquetas dinámicas

Como podemos ver el tiempo de procesamiento es prácticamente constante y se sitúa en torno a los 60msec / frame independientemente de si las etiquetas están en movimiento o no. Al no usar ningún algoritmo de tracking tiene sentido que el tiempo de procesamiento sea constante ya que segmentamos y etiquetamos cada frame

individualmente.

Antes de finalizar el programa se detalla por consola el porcentaje del tiempo que han estado localizados los robots en el entorno de trabajo. Con un simple cálculo se hace la proporción entre el número de frames en el que están localizados los dos robots y el número de frames totales analizados en el transcurso del programa. Para saber si están localizados los robots se buscan tres subetiquetas del color correspondiente a un determinado robot en cada iteración del algoritmo. Si el número de etiquetas se corresponde con este valor es que el robot está presente y localizado.

Estando los robots todo el tiempo en el área de trabajo y con condiciones ideales de iluminación se consigue que el algoritmo los detecte y sea capaz de seguirlos el 94% del tiempo.

6. Presupuesto y planificación del proyecto

Creemos necesario detallar el presupuesto que ha conllevado la realización de este proyecto. Únicamente evaluaremos los costes materiales ya que es difícil evaluar los costes humanos al trabajar con herramientas Open Source. Las comunidades en estas plataformas dedican mucho tiempo tanto al desarrollo de hardware como de software para que el resto de desarrolladores se beneficie y cree nuevos proyectos. Por tanto es complicado estimar un coste para el trabajo desarrollado por las comunidades Open Source ya que crean y publican sin ánimo de lucro, únicamente con el fin de hacer accesible el conocimiento a todo el mundo.

Concepto	Precio	Unidades	Total
Webcam Logitech C210	20.70 €	1	20.70 €
Arduino UNO	21.72 €	2	43.44 €
XBEE modules	23.18 €	3	69.54 €
XBEE shields	14.22 €	2	28.44 €
Plástico impresora 3D	1.65 €	2	3.3 €
Pila 9V	1.83 €	2	3.66 €
Portapilas Clip	0.21 €	2	0.42 €
Jack alimentación	0.85 €	2	1.7 €
Servo Futaba	10.99 €	4	43.96 €
Juntas tóricas, tornillos y cable	2€	2	4 €
		Coste Total	219.16 €

La planificación del proyecto se realizó de forma que paralelamente se fueran desarrollando la parte de visión y la parte de robótica. Mientras se programaba el algoritmo de visión y se hacían pruebas con etiquetas se iban construyendo los robots y programando y probando las comunicaciones inalámbricas

7. Conclusiones y aplicaciones futuras

7.1 Introducción

En este capítulo se detallarán los objetivos conseguidos, las conclusiones extraídas de la realización del trabajo y las aplicaciones futuras que derivarán del mismo.

7.2 Objetivos conseguidos

Se han conseguido los siguientes objetivos con la realización de este trabajo:

- Crear una plataforma de experimentación con un sistema de visión capaz de situar y orientar robots en el entorno de trabajo.
- Construcción física de prototipos.
- Integrar los siguientes elementos a nuestro robot: placas controladoras Arduino, servomotores, módulos de radiofrecuencia.
- Conexión de la placa Arduino UNO con comunicación serie al ordenador.
- Ser capaces de teleoperar desde el ordenador dos robots móviles.

7.3 Conclusiones

A pesar de que la solución de un control centralizado en flotas de robots no es nueva y ha sido ya desarrollada para aplicaciones industriales, militares o de exploración y toma de datos se demuestra que se puede construir un sistema fiable y con buenos resultados basándonos en herramientas Open Source. Esto hace que este tipo de aplicaciones estén disponibles para todo el mundo debido a la gran comunidad existente detrás de estas plataformas.

Se desarrolla un sistema de visión únicamente con un ordenador y una webcam. El resultado es un sistema de bajo coste, ampliable y que proporciona tasas de localización en el entorno experimental superiores al 90%. No hay falsos positivos ya que nunca se detecta un robot que no está presente gracias a las condiciones controladas del entorno. La limitación del sistema se encuentra en los bordes de la cámara ya que no es posible detectar un robot si no se detecta alguna de las etiquetas por estar fuera de rango en las

imágenes. El sistema desarrollado en el trabajo puede ser la base para construir sistemas de mayor complejidad en el futuro.

En la parte de electrónica la plataforma Arduino ofrece prestaciones óptimas para la construcción de robots móviles. La facilidad para la conexión de sensores y actuadores así como la integración con otras plataformas como pueden ser módulos de radiofrecuencia o módulos GPS hace que las limitaciones sean casi inexistentes a la hora de construir robots.

El conjunto del trabajo, al estar desarrollado sobre la base de las plataformas Open Source potencia el desarrollo y elimina barreras al conocimiento. La publicación del mismo bajo este tipo de licencias puede hacer que otros usuarios de las comunidades Open Source profundicen, identifiquen nuevas necesidades, desarrollen y publiquen a través de estos trabajos. Además el valor que implica desarrollar un sistema de bajo coste es enorme ya que impulsa la creación al reducir limitaciones económicas existentes tanto en el campo de la inteligencia artificial (IA) como en el campo de la robótica. El presupuesto demuestra que el trabajo es altamente reproducible ya que para tratarse de un sistema de visión y de unos robots móviles teleoperables el coste total es permisible.

7.4 Aplicaciones futuras

Las aplicaciones futuras de este trabajo estarán orientadas principalmente a la planificación de trayectorias para sistemas multirobot. Partimos del mapa derivado del trabajo en el que mediante una imagen binarizada se muestran los obstáculos así como los límites perimétricos del entorno. Los obstáculos son los propios robots presentes en el campo de trabajo que ocupan un determinado espacio y que evidentemente no estará libre para que pase por allí otro robot. Teniendo en cuenta esta descripción dinámica del entorno un robot podrá hacer planificación de trayectorias teniendo los datos en tiempo real de la situación del área de trabajo.

Además de hacer el sistema ampliable a un número N de robots se buscará la inclusión en algoritmos de tracking que sean capaces de mantener localizado al robot incluso cuando existen oclusiones parciales de la imagen o elevado ruido.

Otras aplicaciones futuras pueden consistir en localización y mapeados del entorno (SLAM). Una localización precisa conlleva tener una referencia adecuada del entorno, un mapa que contenga los obstáculos que el robot puede encontrar en su camino y esto es posible aplicarlo a partir del trabajo realizado. A partir de la localización y basándose en múltiples sensores como pueden ser ópticos se pueden obtener mapeados del entorno precisos para la navegación.

Referencias

[1] N. Bouraqadi, L. Fabresse y A. Doniec. On Fleet Size optimization for Multi-Robot Frontier-Based Exploration, 7th National Conference on Control Architectures of Robots (CAR2012), 2012.

[2] Norman Weiss, Lars Hildebrand. An exemplary robot soccer Vision System, CLAWAR/EURON Workshop on Robots in Entertainment, Leisure and Hobby, 2004.

[3] Vamsi Mohan Peri. Fuzzy logic controller for an Autonomous Robot, master thesis, Department of Electrical and Computer Engineering, Cleveland State University, 2002.

[4] R. Alami, S. Fleury, M. Herrb, F. Ingrand, F. Robert. Multi Robot Cooperation in the Martha Project, IEEE Robotics & Automation Magazine, Mar 1998, vol 5, pages 36 - 47.

[5] Creative Commons. Online: <http://creativecommons.org/> (Última visita: Agosto 2012)

[6] Creative Commons. Tipos de licencias Creative Commons. Online: http://es.wikipedia.org/wiki/Licencias_Creative_Commons (Última visita: Agosto 2012)

[7] SGPS Project. Online: www.sgpsproject.org/ (Última visita: Septiembre 2012)

[8] GNU. Sistema operativo GNU. Online: <http://www.gnu.org/> (Última visita: Agosto 2012)

[9] Thingiverse. Diseño digital del robot. Online: <http://www.thingiverse.com/thing:18264> (Última visita: Septiembre 2012)

[10] Universidad Carlos III de Madrid. Wiki del grupo de impresoras 3D Open-Source. Online: http://asrob.uc3m.es/index.php/Impresora-3D_Open_Source (Última visita: Agosto 2012)

[11] Universidad Carlos III de Madrid. Proyecto Clone Wars. Online: http://asrob.uc3m.es/index.php/Proyecto:_Clone_wars (Última visita: Agosto 2012)

[12] Arduino. Página principal. Online: <http://arduino.cc/es/> (Última visita: Agosto 2012)

[13] Iearobotics. Tutorial de trucaje de los servos. Online: <http://www.iearobotics.com/proyectos/cuadernos/ct2/ct2.html> (Última visita: Agosto 2012)

Anexo I (Código)

En este anexo se presenta el código desarrollado para la realización de este trabajo.

- Programa C++ con el algoritmo de visión.
- Programa usado para la teleoperación a través del puerto serie.
- Programa Arduino para cada robot.

Online:

http://www.sgpsproject.org/JavierVGomez/index.php?title=Adri%C3%A1n_Jim%C3%A9nez

Anexo II (Fuentes Adicionales)

A continuación se detallan las páginas web que han proporcionado fuentes adicionales de imágenes para la memoria del trabajo. Las figuras no detalladas son de elaboración propia o extraídas directamente del trabajo realizado.

Fuente figura 2.1: Club el Aleph. Online: <http://clubelaleph.multiply.com>

Fuente figura 2.2: Carnegie Mellon University. Online: <http://www.cmu.edu>

Fuente figura 2.3: Laboratorio de robótica. Universidad Carlos III de Madrid. Online: <http://roboticslab.uc3m.es>

Fuente figura 3.1: Webvision. Online: <http://webvision.med.utah.edu/>

Fuente figura 3.2: Wikipedia. Online: <http://commons.wikimedia.org>

Fuente figura 3.3: Dept. de Informática y Sistemas. Universidad de Murcia. Online: <http://dis.um.es/>

Fuentes figuras 3.4 y 3.5: Wikipedia. Online: <http://commons.wikimedia.org>

Fuente figura 3.6: La guerra de los mundos. Online: <http://www.laguerradelosmundos.net>

Fuente figura 3.7: Flash & Math. Online: <http://www.flashandmath.com>

Fuente figura 3.9: Wikipedia. Online: <http://commons.wikimedia.org>

Fuente figura 3.10: Futaba. Online: <http://futaba-rc.com/>

Fuente figura 3.11: Aquaticus. Online: <http://aquaticus.info/>

Fuente figura 4.1: Logitech. Online: <http://www.logitech.com>

Fuente figuras 4.12 y 4.13: Thingiverse. Online: <http://www.thingiverse.com>

Fuente figuras 4.14 y 4.15: Asociación de robótica Universidad Carlos III de Madrid.
Online: <http://asrob.uc3m.es>

Fuente figuras 4.16 y 4.17: Arduino <http://www.arduino.cc/es/>

Fuente figura 4.18: RAMSU Robot. Online: <http://ramsurobot.blogspot.com.es/>

Fuente figuras 4.19 y 4.20: XBEE. Online: <http://www.xbee.cl/>