eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Ingeniería de Computadores

Trabajo de Fin de Grado

Navegación autónoma de Micromouse mediante visión artificial

Autor

Sergio Hurtado Solórzano

2021

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Grado en Ingeniería Informática

Ingeniería de Computadores

Trabajo de Fin de Grado

Navegación autónoma de Micromouse mediante visión artificial

Autor

Sergio Hurtado Solórzano

Director

Jose A. Pascual

Resumen

El objetivo de este trabajo de fin de grado es el desarrollo de un prototipo de un robot diseñado acorde a la competición Micromouse, el cual trate de recorrer un laberinto desde la celda de inicio a la celda de destino en el menor tiempo posible de forma autónoma. En este proyecto, el guiado del Micromouse estará basado en visión artificial, es decir, realizará en todo momento la interpretación de las imágenes obtenidas a partir de una cámara a bordo como única forma de percibir el laberinto, prescindiendo de sensorización tradicional y de la adecuación o marcado del entorno. En particular, se emplearán las funciones que proporciona la librería de visión por computador OpenCV y se realizará un análisis de los diferentes métodos y funcionalidades que pueden emplearse para lograr un guiado efectivo del prototipo y de otros sistemas similares. El prototipo constará de un procesador ARM de bajo consumo montado sobre un chasis impreso en 3D, la cámara a bordo, la batería y un conjunto de componentes auxiliares además de los relativos a la motorización.

Índice general

Índice de figuras

Índice de tablas

1. CAPÍTULO

Introducción

La navegación autónoma es una capacidad propia de los robots móviles, la cual históricamente ha sido un importante campo de investigación, debido a las áreas de aplicación en las cuales su utilización puede suponer un enfoque disruptivo. La mejora en el hardware disponible y los tremendos avances en el software de control empleando inteligencia artificial, entre otros métodos, han permitido su uso en un abanico cada vez mayor de posibilidades y han propiciado su uso generalizado en la industria. Su actividad ha consistido principalmente en el transporte de cargas de un lugar a otro, pero también ha tenido un papel destacado en actividades desarrolladas en zonas en las cuales no es segura o viable la intervención de un operador humano.

En las próximas décadas, se espera una sustitución progresiva de la mano de obra humana por otra de naturaleza artificial y para este proceso es necesario realizar grandes avances que permitan una mayor autonomía de este tipo de sistemas. Para lograr estos objetivos, los sistemas deben ser capaces de interactuar con el entorno y aprender a desenvolverse satisfactoriamente ante factores y ambientes cambiantes, siendo capaces de ajustarse a los nuevos desafíos planteados.

Una parte crítica de los sistemas autónomos es la sensorización, ya que los dispositivos que la componen son su mecanismo de percepción del entorno, los cuales condicionan enormemente sus capacidades. Esta sensorización suele estar formada por un conjunto de dispositivos que se encargan de obtener una representación matemática de las características del entorno, pero suelen tener un propósito muy específico y es necesaria la combinación de un conjunto de ellos para la realización de tareas complejas. Por este mo-

tivo, uno de los campos de estudio más exitosos aborda el problema de la sensorización tratando de imitar la naturaleza.

La capacidad sensorial por excelencia en el mundo animal puede ser considerada la visión, gracias a la cual un animal es capaz de percibir la luz rebotada en su entorno e interpretarla para obtener información útil acerca del mismo. Este sistema visual ha sido perfeccionado durante millones de años en los seres vivos que conocemos y cada especie ha adoptado diferentes estrategias dependiendo de su entorno y su uso, proporcionándonos un gran repertorio de técnicas efectivas en las cuales basar nuestros sistemas.

La visión por computador trata de simular la visión animal mediante la obtención de información tridimensional de la estructura y propiedades del entorno a partir de imágenes bidimensionales empleando un razonamiento deductivo de forma automática. Esta disciplina científica tuvo su origen en las décadas de los años 60 y 70, en las cuales se formaron las bases de muchos de los algoritmos que se emplean hoy en día. Sin embargo, a partir de los años 2000 su desarrollo fue exponencial, al resultar de tremenda utilidad para aplicaciones industriales, médicas, policiales y militares. Esto propició que se destinaran numerosos fondos para su desarrollo y su proliferación. Estos sistemas, desde el inicio, se han visto limitados por una serie de problemas, sobre todo referentes a su coste y fiabilidad, pero han sido solventados con gran éxito. Tanto es así que han sido exportados para una gran cantidad aplicaciones, con excelentes resultados y su uso se ha extendido a campos muy variados como la medicina, agricultura, minería, etc.

No cabe duda de que los sistemas de visión por computador tendrán un papel fundamental en los robots del futuro y los dotará de unas capacidades nunca vistas. Por ello el objetivo de este TFG, aprovechando esta tendencia y la proliferación de este tipo de sistemas, es realizar una primera aproximación experimental realizando un prototipo capaz de ser guiado autónomamente y con fiabilidad por un laberinto Micromouse empleando una cámara. Esta prueba de concepto, servirá para analizar las técnicas de visión por computador y sus posibles usos en la navegación autónoma en entornos predecibles o poco cambiantes, permitiendo prescindir de la sensorización extra que emplean los robots móviles.

2. CAPÍTULO

Documento de Objetivos del Proyecto

En este capítulo se van a explicar los objetivos que se han marcado para este proyecto, comenzando por el propósito general de este TFG, que es el estudio de las técnicas de visión artificial y sus posibles usos en la navegación autónoma, permitiendo prescindir de la sensorización extra que emplean los robots móviles. Por lo tanto, este proyecto es una prueba de concepto en la cual se pretende estudiar una amplia variedad de técnicas de visión por computador en un entorno entorno predecible o poco cambiante de cara a explorar la posibilidad de extender este planteamiento a otros sistemas de la industria más complejos.

Para la realización de este proyecto, se empleará una cámara a bordo de un pequeño robot la cual se encargará de proveer las imágenes del entorno próximo mientras este trata de alcanzar el centro del laberinto en el menor tiempo posible, tratando de simular las competiciones de Micromouse.

Por lo tanto, toda la información que percibirá el robot provendrá de la cámara, tanto para el reconocimiento de las celdas como para la navegación y será tratada y procesada para la interpretación y utilización de la misma en la navegación. Debido a este planteamiento, la primera etapa consistirá en la familiarización con la competición Micromouse, sus características, reglas y particularidades, ya que delimitan las posibilidades y necesidades del prototipo.

Una vez identificado el entorno, la siguiente etapa consistirá en investigar y probar distintas técnicas de visión por computador hasta obtener un conjunto de procedimientos que proporcionen resultados útiles para la navegación, idealmente con un grado de fiabilidad

elevado. Para ello, se estudiarán las técnicas empleadas en aplicaciones similares para obtener una perspectiva de las posibilidades, de los puntos fuertes de cada una y de la posibilidad de importar dichas técnicas para este proyecto.

Posteriormente, se realizará un sistema de guiado y navegación para el robot. Este, consistirá en la búsqueda activa mediante la navegación por el laberinto del camino más corto hasta el centro del mismo, partiendo de la celda de salida, situada en uno de los extremos, de cara a realizar una segunda vuelta a la mayor en el menor tiempo posible, al igual que en la competición Micromouse. Para la realización de esta etapa, se optará por la simulación, con la intención de depurar los algoritmos sin la posibilidad de obtener problemas derivados del resto de pasos. En esta etapa también se implementará el mapeo del laberinto. Por último, se procederá a la creación del prototipo final, que consistirá en la agrupación de los productos obtenidos en las diferentes etapas.

En esta etapa el robot será ensamblado y se procederá a la implementación del software de control motriz y la introducción del resto de software de los sistemas del robot. Con la finalidad de lograr una fiabilidad elevada, el prototipo será evaluado y modificado de forma iterativa hasta el límite temporal del proyecto.

A continuación se detallan las tareas necesarias para la realización del proyecto:

1. Reconocer las etapas para el desarrollo del prototipo y realizar la pertinente planificación. Todo ello, junto a las tareas de seguimiento y control del proyecto que se distribuirán a lo largo del mismo.

2. Estudiar los requisitos y las características de la competición Micromouse y como adaptar el prototipo a dichos requisitos.

3. Analizar las técnicas más extendidas de visión por computador, profundizando en todos los pasos clave como la adquisición, procesado, análisis e interpretación de las imágenes y desarrollar un procedimiento o conjunto de ellos capaces de obtener en todo momento la información necesaria del laberinto para la navegación.

4. Estudiar las posibilidades que ofrece la librería OpenCV relativas a las técnicas de visión por computador elegidas.

5. Estudiar las técnicas de navegación y los algoritmos más populares empleados en la competición Micromouse o en aplicaciones similares e implementar un sistema para el prototipo que permita la navegación y mapeo del laberinto.

6. Integrar los diferentes sistemas en el prototipo tras su ensamblado final y evaluar su desempeño.

7. Analizar los resultados obtenidos durante el desarrollo del proyecto.

Por lo tanto, el trabajo relativo al prototipo puede agruparse de la siguiente forma:

- **Módulo de visión artificial** Se encargará del procesamiento e interpretación de las imágenes obtenidas mediante la cámara de a bordo generando información útil relativa al entorno inmediato del robot.
- **Módulo de control motriz** Se encargará de gestionar los movimientos del Micromouse.
- **Módulo de navegación y mapeo** Se encargará de la búsqueda del camino más corto empleando los datos obtenidos por el módulo de visión artificial como entrada y enviará ordenes al módulo motriz para que el robot se desplace en consecuencia. A su vez, se encargará del mapeo del laberinto y su representación.

3. CAPÍTULO

Micromouse

El proyecto está centrado en la competición de Micromouse, un evento que empezó en los años 70 del siglo pasado, el cual es especialmente popular en el continente asiático y que trata de pequeños robots-ratón completamente autónomos que intentan recorrer un laberinto de 16*16 desde una posición predeterminada hasta el centro en el menor tiempo posible. Para ello requieren el uso de una variedad de algoritmos de búsqueda en conjunción con estrategias efectivas para la búsqueda de las paredes y localización en el mapa del dispositivo, todo ello mientras avanza a gran velocidad por el mismo llegando los más rápidos a velocidades de más de 5 m/s.

3.1. Historia

En 1977 la revista de la IEEE introdujo el concepto de Micromouse y anunció la primera competición que fue llevada a cabo en Nueva York en 1979. Esta competición consistía en recorrer un laberinto de 10 x 10 buscando la salida y estuvo formada por 15 competidores de los aproximadamente 6000 candidatos que se presentaron. En aquella ocasión, el ganador fue un "sencillo" robot seguidor de paredes creado por Arthur Boland (ver Figura 3.1).

En 1980, el profesor John Billinsley de la universidad politécnica de Portsmouth modificó las reglas e introdujo la competición en Europa, la cual fue llevada a cabo en Londres. Las nuevas reglas, designaban como objetivo la búsqueda del centro del laberinto e introdujo medidas para evitar que los robots seguidores de paredes pudieran llegar al destino. En

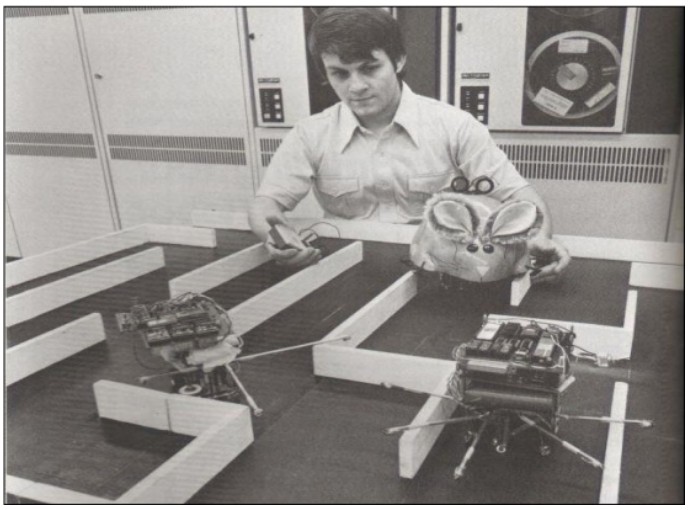


Figura 3.1: Los Micromouse Moonlight Special, Moonlight Express y Moonlight Flash (Boland, 1979)

esta ocasión hubo alrededor de 200 candidatos pero solo 100 lograron participar, de los cuales 9 fueron finalistas. De entre los finalistas, solo uno fue capaz de llegar al centro y se proclamó el primer y único vencedor, aunque la velocidad distaba mucho de la actual, con apenas unos 0,18 m/s o 0,65 km/h. En los años posteriores, el tiempo para alcanzar el destino fue disminuyendo rápidamente.

En 1985, se llevó a cabo la "Primera Competición Mundial de Micromouse" en Tsukuba, Japon. Para fomentar la participación, se enviaron laberintos a varios países y compitieron un gran número de ratones-robot provenientes de todo el mundo. Desde entonces, la competición no ha dejado de ganar adeptos, sobre todo en los países asiáticos donde goza de una mayor popularidad. Aunque se produjo una proliferación de estas competiciones en diferentes países, la mayoría de ellas se basaron en una serie de reglas generales las cuales se encuentran en el anexo A de este documento.

3.2. Laberinto

El laberinto de una competición de micromouse esta formado por un conjunto de celdas de igual tamaño que forman un cuadrado perfecto. Dichas celdas, están formadas por paredes de color blanco que se encajan sobre unos pilares todo ello sobre un suelo de color negro. La disposición de las paredes que forman las celdas y la retirada de algunas de ellas dan forma al trazado pudiendo generar una gran variedad de situaciones y de

caminos distintos, que ponen a prueba las capacidades de los dispositivos que participan en ella.

3.2.1. Tradicional

La versión tradicional emplea un laberinto de 16x16 celdas, de forma cuadrada, con celdas de 180 mm cada lado y 50 mm de alto. Los ratones-robot son completamente autónomos y deben encontrar sin ayuda externa el camino desde una posición de inicio determinada hasta el centro del laberinto. El robot debe ser capaz de situarse, descubrir las paredes circundantes mientras explora, generar el mapa del terreno y detectar cuando se encuentra en el destino. Una vez alcanzado dicho destino, el ratón generalmente realiza búsquedas adicionales por dentro del laberinto hasta encontrar la ruta óptima entre el origen y el destino. Al generar la ruta óptima, el robot tratará de realizar el trayecto en el menor tiempo posible. (ver Figura 3.2)

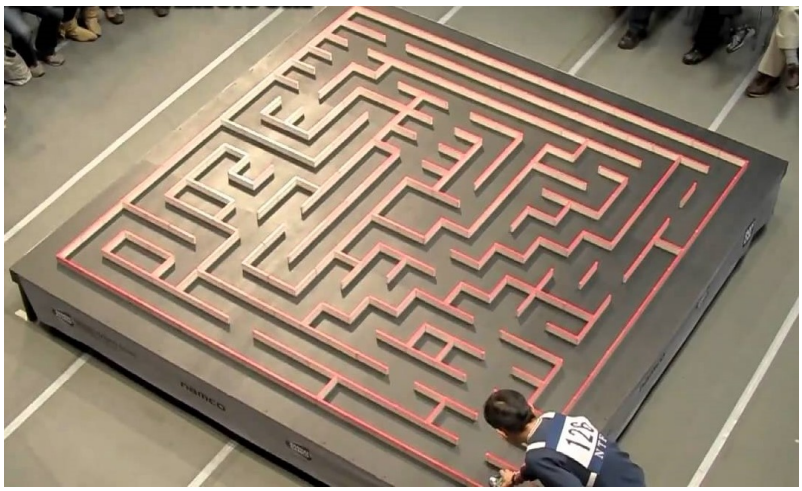


Figura 3.2: Laberinto tradicional de 16x16

3.2.2. Half Size

En 2009 fue introducida en Japón una nueva versión de Micromouse llamada “Half-Size Micromouse” haciendo referencia al hecho de que su tamaño es la mitad que la versión tradicional. En esta modalidad, se emplea un laberinto de 32x32 en vez de los 16x16 del laberinto tradicional. Para ello, las dimensiones de las celdas y las paredes han sido reducidas a la mitad, suponiendo un nuevo reto. El centro/destino, puede estar situado en

cualquier parte dentro del laberinto, pero dicha localización es revelada antes del evento por parte de los organizadores. En la última década, al igual que en Asia y Norteamerica, se han realizado varias competiciones Half-size en Europa. (ver Figura 3.3)

Figura 3.3: Laberinto half-size de 32x32

3.3. Robot

3.3.1. Rendimiento

Los Micromouse se encuentran entre los robots autónomos con mejor rendimiento, son robots extremadamente ágiles y veloces que pueden llegar a alcanzar velocidades de alrededor de 5 m/s, dependiendo del diseño del laberinto. El rendimiento de estos dispositivos ha ido incrementándose de forma considerable en la última década. Por ejemplo, en 2015 un robot ganador rondaba los 10 m/s^2 de aceleración hacia adelante y frenada, pudiendo tomar curvas con una aceleración de hasta 2g. Los robots más recientes, han sido equipados con un ventilador capaz de generar un vacío parcial debajo del microratón. Gracias a este ingenio, el rendimiento ha aumentado de forma notable logrando aceleraciones de más de 6g en curvas y aceleraciones en linea recta que fácilmente exceden las 2,5g.

3.3.2. Características técnicas

Un Micromouse esta formado por un conjunto de componentes relativamente simple, pero que debe estar perfectamente estudiado para obtener un sistema equilibrado y competente. Generalmente se componen de:

- Componentes estructurales y motrices, compuestos por el chasis, los reguladores de voltaje, controladores de motores, motores y ruedas.
- Sensorización, compuesta por sensores, controladores de posición y controladores auxiliares.
- Fuente de energía, consistente en baterías o pilas, ya que los motores de combustión interna no están permitidos.
- Procesador, tradicionalmente microcontroladores aunque los microprocesadores cada vez son más utilizados.
- Elementos para la retroalimentación, pantallas o luces que proporcionan información útil sobre el estado del robot.

Los robots, debido a las características propias de los laberintos de Micromouse, para desplazarse lo más rápidamente posible deben alcanzar grandes velocidades en el menor tiempo posible. Esto se logra con motores que proporcionen una gran aceleración y unas ruedas que permitan transmitir dicha potencia al suelo sin derrapar. Para esto, es imprescindible un buen diseño del chasis y una correcta distribución de los pesos que eviten que la parte delantera se eleve al acelerar o choque con el suelo al frenar.

3.4. Software

Tradicionalmente, los robot Micromouse debido a las limitaciones en capacidad de cómputo, fueron programados en lenguajes como C, de no muy alto nivel, en búsqueda de eficiencia. Esto ha ido cambiando con los años, ya que los nuevos microcontroladores y/o microprocesadores han cambiado dicha situación, facilitando enormemente el empleo de nuevos tipos de sensores, algoritmos y el uso de lenguajes de alto nivel.

El software de un Micromouse, básicamente se puede dividir en tres partes, el algoritmo de búsqueda y mapeo, el cual se encarga de calcular la mejor ruta y de generar el mapa,

la parte encargada de procesar los valores obtenidos por los sensores y de deducir en cada instante de tiempo su situación espacial y cinética y el software de control de los motores, el cual gestiona la movilidad del robot.

4. CAPÍTULO

Navegación autónoma

En este capítulo se trata una de las habilidades más relevantes en la robótica, la navegación autónoma. En particular, se analizarán los problemas que plantea y las soluciones más empleadas en cada caso. Todo ello junto con las tareas necesarias para el correcto funcionamiento de un sistema de navegación autónoma.

4.1. Introducción

Entendemos por navegación autónoma la capacidad de ir de un punto del espacio a otro, evitando obstáculos, sin ningún tipo de intervención humana. Esta, es una capacidad propia de los robots móviles, la cual históricamente ha sido un importante campo de investigación debido a su enorme potencial industrial, militar, científico, etc., ayudando en tareas de automatización y operando en áreas no aptas para un operador humano.

La mejora en el hardware disponible y los tremendos avances en el software de control empleando inteligencia artificial, entre otros métodos, han permitido su uso en un abanico cada vez mayor de posibilidades.

4.2. Nociones básicas

La navegación autónoma plantea una serie de problemas:

1. **Localización:** el vehículo debe conocer en todo momento dónde se encuentra.
2. **Mapeo:** el vehículo debe tener la capacidad de generar una representación de su entorno o mapa.
3. **Planificación de rutas:** el vehículo debe ser capaz de calcular el mejor camino para llegar a su destino.
4. **Seguimiento de rutas:** el vehículo debe ser capaz de seguir la ruta calculada evitando obstáculos.

4.2.1. Localización

La localización es un problema crítico estrechamente vinculado al resto, ya que es imprescindible conocer la ubicación del vehículo para poder realizar cualquiera de las tres tareas clave para la navegación autónoma (mapeo, planificación de rutas y seguimiento de las mismas). Este posicionamiento en el espacio se suele realizar empleando tres fuentes, la odometría, la observación de los sensores y un mapa del entorno los cuales trataré más adelante.

4.2.2. Mapeo

El mapeo es la tarea relativa a adquirir información del entorno y posicionar al robot en él. Dicha localización dentro del entorno puede ser absoluta, empleando sistemas de coordenadas, o relativa determinando la posición por uno o varios puntos conocidos del mapa.

Existen principalmente tres tipos de mapas, métricos, topológicos y mixtos:

Métricos

Son representaciones continuas del espacio real a escala en dos dimensiones mediante coordenadas. El espacio se suele dividir en cuadrantes de igual o diferente tamaño con resoluciones más finas (más precisión) o más gruesas. Este tipo de representación tiene como ventaja la construcción directa, junto con un aprendizaje y mantenimiento fáciles, pero a su vez genera un gran volumen de datos, lo cual supone un problema junto con la limitación a una resolución fija y la complejidad de planificación.

Topológicos

Representan el entorno como una serie de puntos que hacen referencia a zonas, regiones o localizaciones donde es posible encontrar al robot y almacenan las relaciones de vecindad entre los distintos puntos del mapa. Gracias a esas relaciones de vecindad es posible conocer como llegar de un punto a otro del mapa y su distancia, si se cuenta con los pesos de cada arista. En este tipo de representación simplificada, se desechan todos los detalles innecesarios, generando como resultado un mapa más compacto, sobre el cual es sencillo realizar tareas de planificación. En contraposición, resulta menos intuitivo, cuenta con menos información, la cual puede resultar útil en ciertos casos y su generación es más laboriosa.

Mixtos

Son resultado de la combinación de las características propias de los mapas topológicos y los mapas métricos con la intención de obtener las ventajas de cada uno de ellos y de reducir los problemas derivados de dichos tipos de mapa.

4.2.3. Planificación de rutas

Consiste en determinar y seleccionar el camino idóneo para el vehículo dependiendo de las características del entorno y del contexto en el que se encuentre. Con este fin se suelen emplear algoritmos de búsqueda de caminos o del camino más corto en conjunción con otros parámetros. (DFS,BFS, Dijkstra, Floodfill...)

4.2.4. Seguimiento de rutas

El seguimiento de rutas o conducción se centra en seguir el camino elegido, evitando colisiones y adaptándose a otros eventos adversos que puedan ser encontrados en el mismo. Esta tarea se apoya en tres técnicas de navegación auxiliar:

Guiado

- Continuado: Consiste en el seguimiento continuo de un camino preestablecido debidamente marcado. Algunas de las técnicas de marcaje más habituales suelen ser

el marcaje mediante cable soterrado o empleando pintura. Debido a su escasa flexibilidad solo puede ser empleado en entornos sencillos, controlados y con pocos cambios.

- Baliza: Esta técnica requiere un sistema de señalización por balizas, instaladas previamente, las cuales actúan como puntos de referencia. Este tipo de guiado proporciona una mayor autonomía pero requiere de un sistema de corrección de errores odométricos mediante reposicionado.

- Marcas: Se basa en el seguimiento de unas marcas de referencia las cuales indican o delimitan el trazado, por ejemplo, nuestras carreteras emplean este tipo de señalización para indicar el trazado disponible.

Dead reckoning o navegación por estima

El dead reckoning o navegación por estima se trata de la inferencia de la ubicación actual de un vehículo por medios analíticos realizando cálculos basados en el rumbo y la velocidad de desplazamiento. Para ello se suelen emplear puntos de referencia, generalmente balizas, y se realizan operaciones trigonométricas que requieren eliminar constantemente los errores acumulados con continuos ajustes empleando varios sistemas de ayuda a la navegación. Este procedimiento suele implementarse mediante las siguientes técnicas:

- Odometría: Se trata de emplear los datos sobre la rotación de las ruedas para estimar cambios en la posición a lo largo del tiempo. Proporciona una buena precisión a corto plazo, su implementación no es costosa y permite tasas de muestreo muy altas. Los dispositivos más habituales empleados con este fin suelen ser los encoders o codificadores rotatorios. Las técnicas odométricas no tienen una validez absoluta ya que la rotación de las ruedas puede no corresponderse a un desplazamiento lineal, como pudiera darse el caso al patinar. En adición, existen otros problemas asociados a esta técnica que se pueden dividir en errores sistemáticos (mal alineamiento de las ruedas, los diámetros de las ruedas no son iguales...) y no sistemáticos (suelos resbaladizos, derrapes...).

- Sensores Doppler: Emplean el efecto de mismo nombre que básicamente consiste en interpretar la variación de la longitud de onda aparente entre la señal emitida y la recibida tras rebotar en algún objeto.

- Sensores inerciales: Son sensores que miden la aceleración y velocidad angular. Están compuestos por acelerómetros, giróscopos y magnetómetros, los cuales se encargan de medir la aceleración lineal con que se mueve el sensor, la velocidad angular y el norte magnético con el objetivo de estudiar el movimiento del sensor inercial completo en el plano o el espacio.

Evitación de obstáculos

Para un sistema de navegación autónoma son obstáculos todos los objetos que no aparecen originalmente en el mapa, por lo tanto, es imprescindible afrontar de forma adecuada la evitación de dicho imprevisto, pudiendo tener que hacer frente a una enorme casuística. Esto supone un aspecto fundamental a la hora de determinar la efectividad de dicho sistema dependiendo del uso para el que vaya a ser destinado.

La evitación de obstáculos puede ser afrontada empleando dos tipos de estrategia, de forma que el sistema pueda sortear de forma efectiva los obstáculos que se va a encontrar en el entorno en el que va a ser empleado. Por un lado, se puede afrontar una estrategia a la que podemos denominar sobreajuste en la cual se focaliza en la obtención de estrategias para esquivar un tipo concreto de obstáculo o un grupo reducido de ellos, sabiendo previamente que es altamente probable que estén presentes en dicho entorno. Sobre este obstáculo o grupo de ellos, se entrena multitud de veces al sistema acabando por generar una estrategia tremendamente efectiva, pero extremadamente focalizada en dicho conjunto reducido de objetivos. Esta estrategia está limitada a entornos bastante controlados y repetitivos.

Por otro lado, se puede afrontar otro planteamiento en el cual se trate de ser capaz de evitar una gran variedad de obstáculos, adoptando estrategias genéricas y flexibles. Este planteamiento tiene como ventaja ser capaz de gestionar una gran casuística, pero arroja peores resultados individuales en comparación, ya que no se amolda perfectamente a las particularidades de cada obstáculo.

5. CAPÍTULO

Visión por Computador

En este capítulo se va a tratar todo el proceso relativo a la visión por computador: desde la captación de imágenes hasta su interpretación pasando por todos los procesos de adecuación de la imagen necesarios. Estos conceptos son de gran importancia para la comprensión de las técnicas utilizadas en el prototipo y los motivos que han propiciado su uso. Para finalizar, se analizará también la librería OpenCV, la cual es el estándar de facto para la implementación de aplicaciones de Visión por Computador.

5.1. Introducción

La visión por computador consiste básicamente en la obtención de información tridimensional de la estructura y propiedades del entorno a partir de imágenes bidimensionales empleando un razonamiento deductivo de forma automática. Se trata de simular la visión animal empleando cámaras y un conjunto de operaciones matemáticas para la interpretación del entorno.

La arquitectura básica de cualquier sistema de Visión por Computador está formada por un sistema de captura de imágenes, que suele ser un conjunto de cámaras las cuales se encargan de la transformación de la luz que reciben en una matriz de píxeles, un procesado que se encarga de realizar las operaciones deseadas sobre las imágenes obtenidas por la cámara y un sistema de control el cual interpreta los resultados y toma decisiones de forma acorde. Estos pasos suelen ser realizados de forma secuencial pero pueden repetirse

múltiples veces o incluso retroceder a pasos anteriores para tareas complejas, contrastar datos o para corregir fallos.

Existen multitud de técnicas tanto para el tratamiento de las imágenes como para su interpretación por lo que me centraré en las que se suelen emplear para la navegación de robots autónomos, en concreto las que han sido empleadas o se ha planteado su uso para la realización de este proyecto.

5.2. Captura de imágenes

Una imagen es una representación bidimensional resultado de la adquisición de una señal proporcionada por un sensor, que convierte la información del espectro electromagnético en codificaciones numéricas.

Esta imagen puede tratarse de una imagen matricial o de un gráfico vectorial (objetos geométricos dependientes definidos por atributos matemáticos de forma, de posición, etc), dependiendo de si es estática o dinámica. Una imagen matricial esta formada por una matriz (o vector) de dimensiones N x M, que contiene en cada elemento de la matriz un valor discreto que cuantifica el nivel de información del correspondiente elemento, representado con un número finito de bits.

Los niveles de esa cuantificación suelen realizarse en potencias de 2 para facilitar su almacenamiento y procesamiento. Cuando una imagen es monocromática y contiene únicamente información sobre el brillo se suele decir que dicha imagen está en escala de grises, cuya escala va del 0 (correspondiente al color negro) al 255 (correspondiente al color blanco). Cuando una imagen contiene dos valores posibles por cada píxel estamos ante una imagen binaria, en la cual uno de los colores se emplea para el fondo y el otro para el contenido de la imagen.

Las imágenes en color emplean 3 canales que hacen referencia a los colores primarios en la síntesis aditiva del color (R)Rojo, (G)Verde y (B)Azul. Cada canal emplea 256 niveles de intensidad pudiendo formar una imagen de 16 millones de colores (256*256*256). También, se pueden emplear otros modelos de color como el HSV (Hue Saturation Value) o HSL (Hue Saturation Lightness).

En este proceso de captura de la imagen, la señal electromagnética en forma de luz, se transmite, refracta y refleja en la escena hasta que alcanza el área del sensor de la cámara.

Tras lo cual, el sensor produce la transducción a una señal eléctrica y un conjunto de elementos electrónicos digitalizan, formatean, transmiten y almacenan la imagen digital.

Debido a las características de este proceso, es imperiosa la adecuación de la imagen, ya que la información bruta obtenida cuenta con una gran cantidad de ruido, muestra algunos aspectos de forma alterada y cuenta con información extra que alteraría cualquier tipo de razonamiento deductivo que pudiera ser aplicado o conllevaría la necesidad de emplear gran cantidad de cómputo, lo cual en algunos casos es inasumible.

Esta captura está altamente influenciada por los siguientes factores:

5.2.1. Fuente de iluminación

La fuente de iluminación tiene un impacto decisivo en la captura de la escena ya que es la que proporciona la magnitud física que mide el sensor de la cámara. La iluminación de la escena puede ser natural o artificial. El empleo de sistemas de iluminación artificial, permite el control cromático al limitar la longitud de onda de la luz emitida a la deseada para un uso concreto, además de la posibilidad de ser orientada y focalizada en áreas concretas de la escena. Este control preciso puede ser utilizado para proyectar patrones sobre la escena, los cuales son empleados en técnicas de triangulación. Para la elección de una fuente de iluminación se deben tener en cuenta los factores resultantes de emplear dicha fuente como pueden ser sombras, reflejos o retornos de color no deseados al incidir sobre ciertos materiales.

5.2.2. Cámara

La cámara es el núcleo de cualquier sistema de visión por computador y es un factor determinante. Se encarga de transformar la luz reflejada por la escena en la imagen que se emplea como entrada del sistema. Los elementos más importantes de la cámara son la lente y el sensor. Los sensores más comunes son CCD y CMOS los cuales están basados en condensadores y transistores respectivamente, estos últimos han ganado mucha popularidad ya que su precio es menor y su calidad ha alcanzado a los sensores CCD. Los aspectos determinantes en un sensor son el tamaño, el número de píxeles con los que genera la imagen el sensor, el contraste, el brillo y el rango de longitudes de onda que es capaz de percibir. Todos estos aspectos deben ser tenidos en cuenta de cara a elegir el tipo de cámara adecuado para el uso que se le va a dar.

5.2.3. Objetivo

El objetivo es el dispositivo que se encarga de concentrar la luz reflejada por los objetos de la escena en el sensor de la cámara para generar la imagen. Esta formado por una serie de lentes convergentes y divergentes las cuales redireccionan los haces de luz. Las características principales son la luminosidad (interfiere en la cantidad de luz que le llega al objetivo), el número f (cociente entre su apertura máxima y su distancia focal) y la distancia focal (define el aumento o "zoom"). Las lentes empleadas generan una distorsión que pueden afectar de forma notable al desempeño de los algoritmos, ya que alteran la forma y disposición de los elementos en la imagen. Esto se hace muy evidente al observar la curvatura proporcionada a los elementos con lineas rectas apreciados en una imagen. En el siguiente ejemplo se aprecia perfectamente la excesiva curvatura en el horizonte de la imagen debido a la distorsión del objetivo: (ver Figura 5.1)

Figura 5.1: Distorsión producida por el objetivo de una cámara

5.2.4. Filtro

El filtro es el elemento encargado de permitir el paso de luz con ciertas propiedades, atenuando o eliminando la luz restante. Existe una gran variedad de filtros y pueden ser empleados con gran efectividad en los sistemas de visión por computador ya que permiten un gran control sobre la luz proveniente de la escena cuando no es posible controlar la fuente de iluminación o no se logra el resultado deseado.

5.3. Adecuación de la imagen

En esta etapa se eliminan partes no deseadas de las imágenes y se aplican transformaciones y filtros que facilitarán su posterior procesamiento.

- **Corrección de la distorsión:** Las distorsiones generadas por las lentes del objetivo pueden ser corregidas o al menos mitigadas realizando una calibración. Este proceso se lleva a cabo aplicando una transformación que emplea parámetros que han sido previamente obtenidos. Dichos parámetros se obtienen a partir de un proceso de análisis de las alteraciones que genera la cámara al fotografiar múltiples veces y en distintos ángulos un patrón conocido del que se saben sus valores reales. (ver Figura 5.2)

  En este proyecto, se ha empleado como patrón reconocible un tablero de ajedrez impreso en una hoja de papel, ya que la librería OpenCV dispone de varias funciones que facilitan este proceso. (ver Figura 5.3)

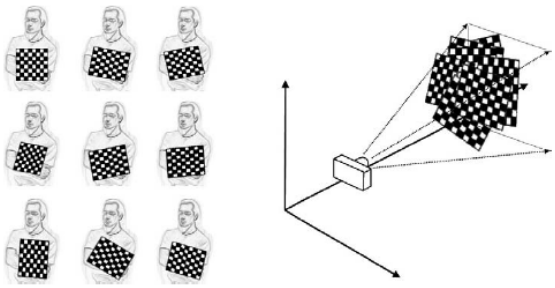


Figura 5.2: Proceso de calibrado de la cámara

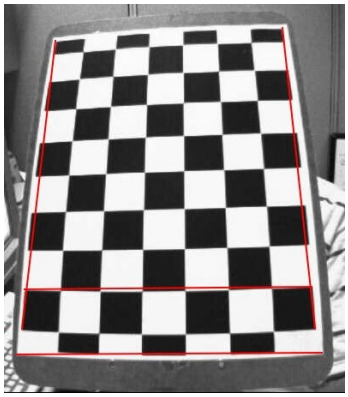


Figura 5.3: Distorsión producida por una cámara sobre un tablero de ajedrez

- **Preprocesamiento:** Se somete la imagen a procesos de eliminación de ruido, transformación del color y/o realce de la imagen. Existen multitud de procedimientos empleados en el procesamiento de imágenes:

  - **Operaciones sobre el histograma**
  - **Binarización por color**
  - **Operaciones aritméticas con imágenes**
  - **Filtrado**
  - **Operaciones geométricas**

5.3.1. Operaciones sobre el histograma:

El histograma de la imagen es el gráfico que se obtiene de la representación de la frecuencia de aparición de cada uno de los valores de una imagen en escala de grises del 0(negro) al 255(blanco), es decir, cada barra vertical indica el número de píxeles que contienen el mismo valor en la imagen. De esta forma se puede apreciar de forma visual que tonos son los predominantes observando las agrupaciones de píxeles en las zonas oscuras, medias o claras. En adición, el histograma normalizado surge a partir de dividir cada frecuencia entre el número total de píxeles de la imagen, representando cada valor obtenido la probabilidad de obtener un píxel con el valor de intensidad al que esta asociada la frecuencia. El histograma en sí, puede ser incluso empleado para el reconocimiento de imágenes, entre otros usos. (ver Figura 5.4)

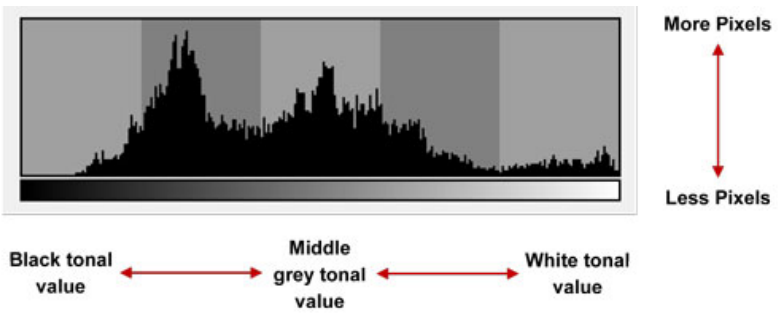


Figura 5.4: Histograma de una imagen

Umbralización del histograma

Se trata de obtener una imagen binarizada a partir de la original usando un valor umbral que divide en dos conjuntos los píxeles de la imagen, aunque existen métodos que solo alteran parte de los valores de la imagen. La elección del umbral es de gran dificultad y se han propuesto varios métodos automáticos con tal fin. Uno de los más famosos es el método de Otsu el cual propuso en 1979 emplear la varianza para calcular de forma automática el valor umbral. Este método busca minimizar la dispersión dentro de cada conjunto y para ello, calcula la media aritmética de los valores de gris de toda la imagen, y posteriormente de cada zona del histograma, acabando por calcular las varianzas de cada zona.

Como la intención es mantener las varianzas de cada zona lo menores posible, realiza el cociente de la varianza entre las zonas y la suma de las varianzas en cada zona, buscando que dicho cociente sea lo mayor posible.

Ecualización del histograma

Con el objetivo de mejorar el contraste de las imágenes se suele emplear la ecualización del histograma de una imagen con la intención obtener una distribución uniforme de los píxeles y que toda la escala de grises cuente con la misma probabilidad de aparecer en la imagen.

5.3.2. Binarización por color

Las imágenes en RGB están formadas por tres matrices diferentes por lo que su binarización resulta complicada. Para ello se suele emplear la conversión a otros modelos de color como HSL o HSV.

5.3.3. Operaciones aritméticas con imágenes

La aritmética de imágenes es la implementación de operaciones aritméticas simples pero aplicadas en imágenes, como la suma, resta, multiplicación y división. (ver Figura 5.5)

1. **Conjunción o AND entre dos imágenes:** Suele ser empleado para borrar partes no deseadas de las imágenes.

2. **Disyunción u OR entre dos imágenes:** Se suele emplear para añadir información una imagen.
3. **Negación o NOT:** Se emplea para obtener el negativo de la imagen.
4. **Suma, resta, multiplicación, división:** Suma, resta, multiplica o divide los valores de los píxeles de dos imágenes.

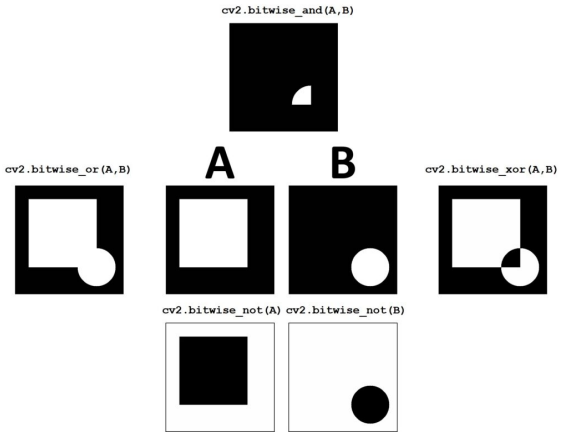


Figura 5.5: Ejemplo de operaciones aritméticas en imágenes en blanco y negro

5.3.4. Filtrado

El filtrado es un conjunto de técnicas cuya finalidad es obtener como resultado una imagen idónea para ser procesada con posterioridad o al menos más adecuada que la original. Existen dos tipos de filtro, los filtros en el dominio de la frecuencia y los filtros en dominio del espacio.

Los filtros en el dominio de la frecuencia, se encargan de calcular la transformada de Fourier de la imagen a mejorar, multiplicar el resultado por la función de transferencia de un filtro y, finalmente, tomar la transformada inversa de Fourier para llegar a la imagen mejorada.

Los filtros en el dominio del espacio, en cambio, trabajan directamente sobre los píxeles de la imagen. Los filtros más comunes son:

Filtros en el dominio de la frecuencia:

1. **Filtros pasa altos:** Atenúan las frecuencias bajas y mantienen sin variación las frecuencias altas.

2. **Filtros pasa bajos:** Atenúan las frecuencias altas y mantienen sin variación las frecuencias bajas.

3. **Filtros pasa banda:** Atenúan las frecuencias altas y bajas las cuales sobrepasen los umbrales superior e inferior.

Filtros en el dominio del espacio:

1. **Filtros de suavizado (Paso bajo):** Eliminan ruido y otros pequeños detalles que no son de interés afectando a las zonas que tienen muchos cambios de intensidad. Entre los filtros de suavizado destacan los de media, media ponderada, mediana, adaptativo y Gaussiano.

2. **Filtros de realce (Paso alto):** Acentúa los bordes y detalles y atenúa las zonas de tonalidad uniforme permitiendo una mejor identificación de los objetos de la imagen pero suele conllevar la aparición de ruido.

3. **Filtros de mejora de bordes:** Los bordes en una imagen se considera que son píxeles con un alto gradiente, es decir, con un rápido cambio de intensidad en alguna dirección dada por el ángulo. Existen varias técnicas empleadas en el realce de bordes:

    - **Desplazamiento y sustracción:** Consiste en realizar una copia de la imagen original, desplazarla horizontal o verticalmente y realizar la operación de resta a la imagen original con la copia desplazada.
    - **Gradiente y gradientes direccionales:** Realiza derivadas discretas en las direcciones x o y de la imagen o bien en otras direcciones.
    - **Laplaciano:** Están basados en la segunda derivada pero son muy sensibles al ruido.
    - **Prewitt, Sobel y Frei-Chen:** Se basan en el cálculo de la primera derivada midiendo las evoluciones y los cambios de una variable, básicamente se centran en detectar los cambios de intensidad.
    - **Canny:** Emplea el gradiente de la imagen pero utiliza dos umbrales para los bordes, permitiendo un control sobre el tipo de borde deseado.
    - **Roberts:** Utiliza las direcciones diagonales para calcular el vector gradiente empleando unas máscaras.

5.3.5. Operaciones geométricas

Las operaciones geométricas son las que modifican la posición o la forma de la imagen. Este tipo de operaciones son muy comunes a causa de las limitaciones relacionadas con la bidimensionalidad de la imagen.

- **Transformaciones geométricas:** Debido a las limitaciones relativas al ángulo de la cámara respecto a la escena suele ser necesario realizar una transformación de la perspectiva. Existen diferentes tipos de transformaciones pero su uso puede generar imágenes con ruido y otros problemas relacionados con la alteración de formas. [Opencv.org, 2021b]

  - **Translación:** Consiste en el traslado de un grupo de píxeles de una posición a otra de la imagen.
  - **Escalado:** Consiste en variar el tamaño de la imagen original a lo largo de cualquier eje de las coordenadas x e y. También puede ser escalada una zona de la imagen respecto al resto de la imagen, esto se conoce como zoom. Para ello se pueden aplicar distintas técnicas como repetir los valores de los píxeles previos o utilizar interpolación lineal para calcular el valor de un píxel intermedio entre dos con valores conocidos.
  - **Rotación:** Consiste en el giro rotativo de un conjunto de píxeles a un eje o a un punto dentro de la imagen

- **Operaciones morfológicas:** Las operaciones morfológicas proporcionan una gran variedad de usos como la eliminación del ruido, aislar o aglutinar elementos dispersos en un imagen, rellenar huecos, encoger objetos, etc.

  - **Erosión:** En el proceso de erosionado un kernel se desliza a través de la imagen. Un píxel de la imagen original (1 ó 0) sólo se considerará 1 si todos los píxeles que caen dentro de la ventana del kernel son 1, de lo contrario se erosiona (se hace a cero). Por tanto, todos los píxeles cerca de los bordes de los objetos en la imagen serán descartados dependiendo del tamaño del kernel. Como consecuencia, el grosor o el tamaño de los objetos en primer plano disminuye en la imagen.
  - **Dilatación:** Es el proceso inverso a la erosión.
  - **Apertura:** Se trata de la erosión seguida de la dilatación, se emplea para eliminar ruido.

- **Cierre:** Se trata de la erosión seguida de la dilatación, se emplea para eliminar ruidos como pequeños puntos negros.
- **Gradiente morfológico:** Es la diferencia entre la dilatación y la erosión de una imagen dando lugar a un resultado similar al contorno.
- **Top Hat:** Se emplea para resaltar objetos brillantes de interés sobre un fondo oscuro.
- **Black Hat:** Es la operación inversa a top hat.

5.4. Interpretación de la imagen

Una vez se han aplicado los cambios necesarios en la imagen se procede a interpretar los elementos que la componen.

- **Segmentación:** Detección de bordes, contornos, regiones y otras características de la imagen con la intención de separar los diferentes elementos de la escena.
- **Extracción de características:** Se obtiene una representación formal de los elementos segmentados en la etapa anterior.
- **Reconocimiento y localización:** Se trata de localizar al objeto en el espacio 3D empleando técnicas como la triangulación o la desviación respecto a puntos de referencia.
- **Interpretación:** A partir de la información obtenida en las etapas previas se interpreta el estado del entorno.

Estos conceptos serán desarrollados o contextualizados al explicar las técnicas empleadas para la realización de este proyecto.

5.5. OpenCV

OpenCV (Open Source Computer Vision Library) es una biblioteca de visión por computador e Inteligencia Artificial que fue desarrollada por Intel a finales de los años 90, la cual se ha convertido en una de las bibliotecas de visión artificial más popular durante los últimos 20 años gracias a que es código libre y a su hincapié en la facilidad de uso y rendimiento.

5.5.1. Antecedentes

Tradicionalmente, los sistemas de Visión Artificial se programaban en lenguajes de bajo nivel, debido al enorme gasto computacional requerido. Esto suponía un problema, ya que requería que estos sistemas fueran creados por programadores expertos en el lenguaje ensamblador de una máquina, por lo que estos programas fueron progresivamente siendo sustituidos por C. C suponía un gran avance, ya que permitía emplear programadores menos instruidos que podían aprovechar las abundantes bibliotecas que provee y el código generado podría ser empleado en multitud de equipos, probablemente con un desempeño aun mayor que su alternativa en ensamblador. Esta tendencia hacia un mayor grado de abstracción ha continuado y cada vez se han ido desarrollando sistemas de Visión Artificial en lenguajes de más alto nivel como Java, Matlab o Python gracias al aumento en capacidad de cómputo y al estar muchas de las funciones implementadas internamente en C o C++.

5.5.2. Historia de OpenCV

En enero de 1999 Intel se embarcó en la creación de una librería de visión artificial, la cual fue liberada al publico un año después. Tras ser liberado, las primeras versiones alfa ganaron rápidamente popularidad y en 2006 Intel lanzó la versión 1.0 a la cual le siguió la 2.0 en 2009. En 2012, el proyecto fue delegado a la fundación Opencv.org que es la que se encarga actualmente de su desarrollo empleando voluntarios. También cuenta con el apoyo de grandes compañías que apoyan el desarrollo como Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota... las cuales emplean la librería para sus productos.

5.5.3. Características

La finalidad de OpenCV es proveer de una infraestructura común para aplicaciones de visión artificial y acelerar su uso en productos comerciales, haciendo su código sencillo de emplear y modificar para las empresas. En la actualidad la librería cuenta con más de 2500 algoritmos optimizados los cuales pueden ser empleados para reconocer caras, identificar objetos, clasificar comportamientos humanos en vídeos, realizar seguimientos de cámara u objetos, extraer modelos en 3D de objetos, producir nubes de puntos 3D a partir de cámaras estereográficas, eliminar ojos rojos de una imagen, realizar seguimiento del

movimiento ocular, reconocer escenas, proveer funcionalidades para realidad aumentada y un largo etcétera según la propia organización.

OpenCV está implementado internamente en C++ para proporcionar un buen rendimiento, pero cuenta con interfaces en C++, Python, Java y MATLAB, para permitir un mayor grado de abstracción. Soporta los principales sistemas operativos y además permite conjuntos de instrucciones MMX y SSE y también cuenta con soporte en plataformas de computación en paralelo como CUDA y OpenCl. Todas estas características lo convierten en una buena elección de cara a realizar casi cualquier tipo de proyecto, ya que permite el desarrollo con un alto nivel de abstracción manteniendo unas estupendas capacidades de computo.

Otra característica muy interesante, es su estructura modular, la cual proporciona un gran control sobre el conjunto de funciones que se desea emplear y permite un gran ahorro de memoria. El conjunto principal esta compuesto por 15 módulos y cuenta con más de 57 módulos que aportan funcionalidades extra. [Opencv.org, 2021c]

6. CAPÍTULO

Implementación

En este capítulo se va a explicar el trabajo realizado para la implementación del prototipo y todos los estudios y conclusiones a las que se ha llegado durante dicho proceso. El código desarrollado se encuentra disponible en el siguiente repositorio [Hurtado, 2021].

6.1. Planteamiento

Este proyecto, surgió a partir de la idea de emplear un sistema de visión por computador para la realización de tareas relativa a la robótica. Inicialmente, se estudiaron varias opciones, cuyo nexo en común siempre era la sustitución de una parte de la sensorización que suelen emplear dichos sistemas robóticos. Esto dio lugar a una serie de ideas las cuales fueron analizadas en busca de su viabilidad. Posteriormente, mi tutor, tuvo una idea que resultó ser clave, ya que estaba muy relacionada con la temática que se estaba buscando y con un tipo de proyecto que permitía la experimentación y la realización de un proyecto interesante e ilustrativo.

Una vez definida la idea general, se procedió a realizar una búsqueda de la viabilidad de sustituir todos los sensores de un robot Micromouse por una cámara, a partir de la cual interpretar el entorno del robot y su situación y ser capaz de desempeñar las mismas funciones que un Micromouse convencional. Tras el análisis de numerosos aspectos y varios planteamientos posteriormente mencionados, se inició el proceso de desarrollo del mismo, hasta el día de hoy.

6.2. Contextualización

Para abordar la nada trivial sustitución de un sistema de sensores de demostrada eficacia, lo primero es plantear: ¿Qué se obtiene al sustituir dicho sistema por otro?.

Los primeros aspectos que se tienen en cuenta suelen ser coste, funcionalidades extra, flexibilidad, la posibilidad de emplearlo para diversas funciones dentro del mismo sistema, posibilidad de exportar este sistema a otras aplicaciones, visión de futuro... .

Si se analizan punto por punto estos aspectos es más que evidente, que los beneficios de emplear un sistema de visión por computador son muy altos y cuentan con una gran proyección a futuro, aunque es cierto que a día de hoy padecen un conjunto de problemas los cuales han limitado su proliferación.

Como primer punto, el coste de una cámara o de un conjunto de ellas, puede suponer un gran desembolso, pero dependiendo de la aplicación hay un sinfín de casos en los que un conjunto de sensores conllevan un desembolso mayor aún y más tras la generalización de las cámaras y su producción en masa con la consecuente bajada de precios.

En adición, en las últimas décadas hemos visto la generalización de los sistemas de visión por computador y la sustitución progresiva de diversos tipos de sensorización por estos últimos, comenzando por aplicaciones consideradas críticas o rentables como las militares y policiales y las relacionadas con industrias potentes. (ver Figuras 6.1 y 6.2)

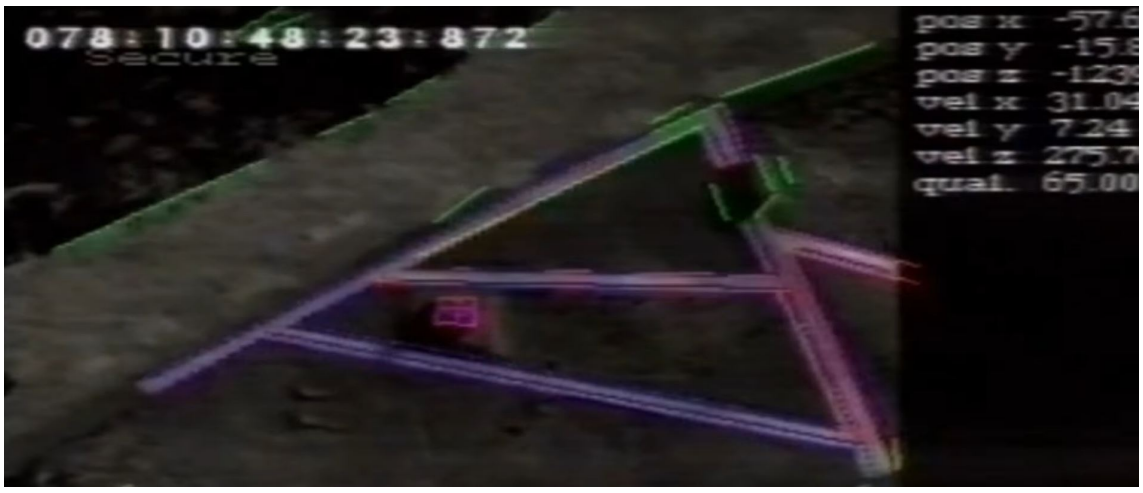


Figura 6.1: Fotograma de la aproximación del misil de crucero Taurus a su objetivo

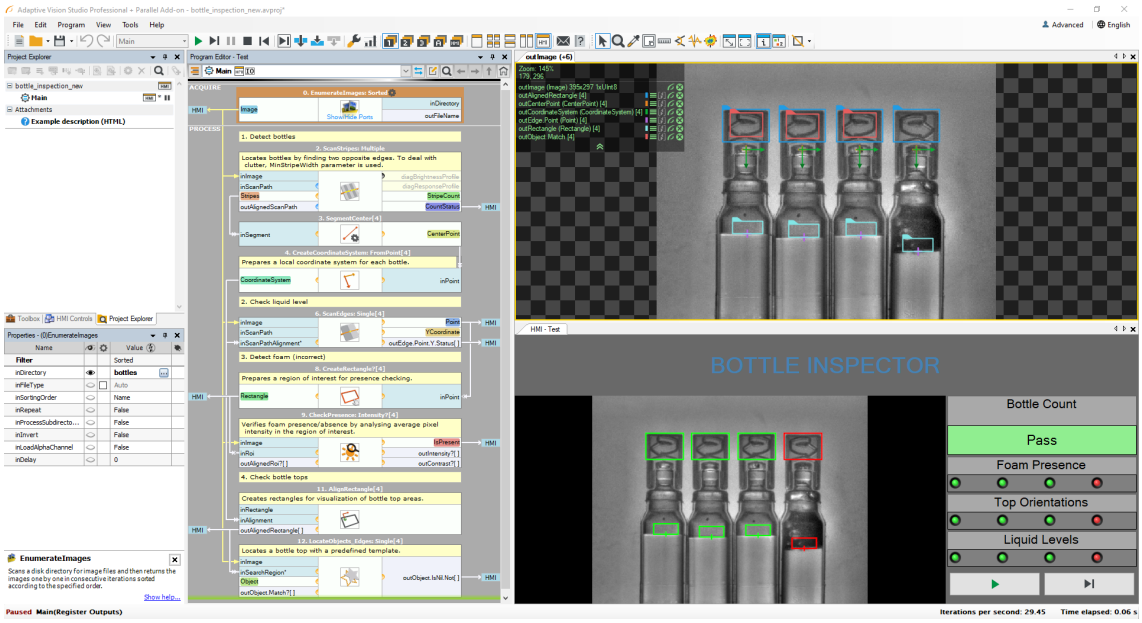


Figura 6.2: Software comercial de control de calidad mediante visión por computador

Estos sistemas, una vez han ido siendo pulidos, se han exportado rápida y satisfactoriamente a numerosos campos, igualando o incluso aumentando las capacidades previas.

Por otro lado, los problemas previos de estabilidad y la necesidad de implementarlos entornos controlados se han ido solventando, obteniendo productos muy maduros con soluciones innovadoras que pueden ser exportados a otros campos. Por lo tanto, teniendo en cuenta estos aspectos, parece más que razonable tratar de realizar este proyecto. Y no solo este, sino muchos otros proyectos similares o más complejos.

6.3. Primeras aproximaciones del módulo de visión artificial

Una vez planteado el problema, se buscaron características propias del entorno en el que se situaría el robot con la intención de identificar patrones repetitivos que pudieran ser empleados como referencia para el guiado.

Primero, se valoró emplear estrategias de cálculo de puntos característicos, que son capaces de reconocer ciertos patrones y dichas funciones se encuentran implementadas en la librería OpenCV pero debido a las características del laberinto, los puntos característicos se consideró que podrían ser alterados por el ruido y las deficiencias en la iluminación o los reflejos.

Más tarde, se obtuvo de internet un fragmento de una cámara a bordo de un Micromouse recorriendo un laberinto, lo cual proporcionó una perspectiva distinta y permitió observar como la intersección entre las paredes y el suelo podría ser aislada y empleada para guiar al robot.

Para comprobar la viabilidad de esta idea, se buscaron funciones implementadas en la librería OpenCV que pudieran ser empleadas para el reconocimiento de los bordes, pero el resultado sobre el vídeo, con abundante ruido, baja resolución e iluminación pobre, fue insuficiente.

Las pruebas en el vídeo, evidenciaron que los problemas de inestabilidad eran producidos por estos factores, por lo tanto se buscaron aplicaciones similares para analizar que métodos de preprocesado de la imagen eran efectivos o al menos usados con frecuencia.

La aplicación que a priori se asemeja más a este proyecto es la conducción autónoma, ya que el trazado está definido por las líneas que delimitan la carretera, al igual que las paredes del laberinto, por lo que fue el punto de partida.

6.3.1. Conducción autónoma

Existen multitud de técnicas, pero en este caso, se excluyen las que emplean inteligencia artificial y se opta por una aproximación tradicional. Analizando múltiples ejemplos, el procedimiento básico, se puede resumir en que las líneas son aisladas y procesadas de forma que se mantiene en todo momento al vehículo entre las líneas laterales. El proceso de aislamiento de las líneas generalmente se lleva a cabo empleando el color, el cual es el rasgo que mejor las distingue del resto de la imagen. (ver Figura 6.3)

Pero dichas lineas laterales no proporcionan toda la información necesaria para guiar al vehículo, o al menos desde esa perspectiva, ya que no es posible ver las distancias reales de la imagen. Para ello, debería disponer de una cámara paralela al suelo. Esto no siempre es posible, pero existe otra solución, realizar una transformación de la perspectiva a la imagen, tras la cual tomando la imagen frontal del robot se genere una imagen virtual paralela al suelo, lo que comúnmente conocemos como vista de pájaro. (ver Figura 6.4)

Esta transformación se encuentra implementada en la librería OpenCV y emplea unas coordenadas de origen y otras de destino las cuales son los puntos de referencia.

La librería no calcula automáticamente las coordenadas, por lo que es necesario realizar múltiples ajustes hasta obtener el resultado deseado. (ver Figura 6.5)

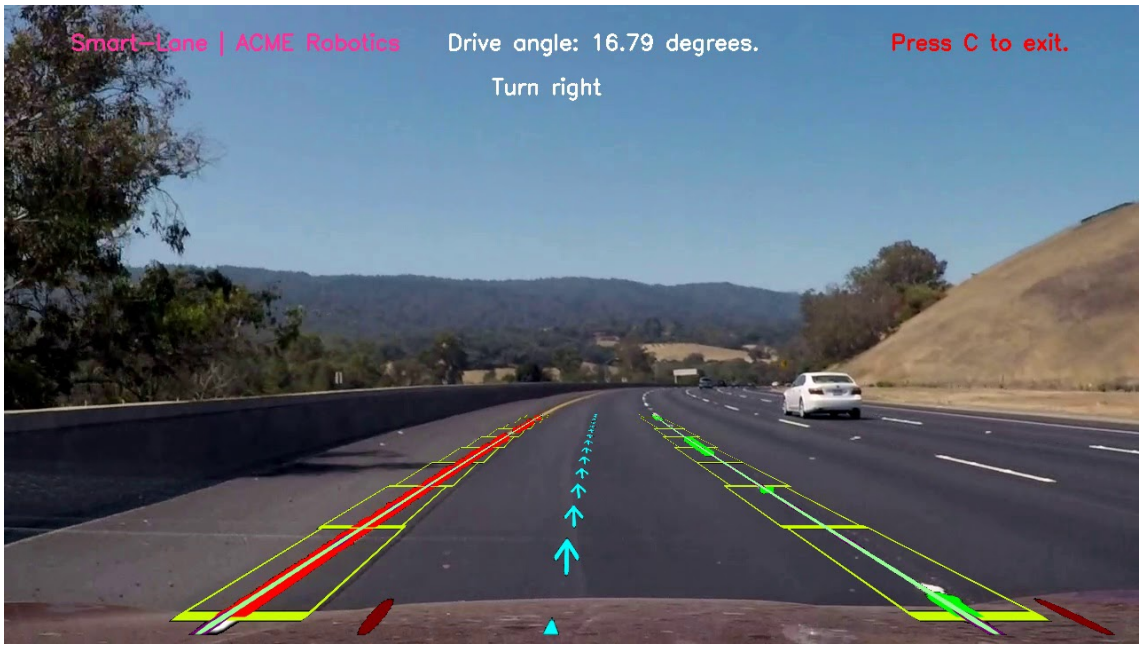


Figura 6.3: Vista obtenida de un programa encargado de interpretar las líneas de una carretera

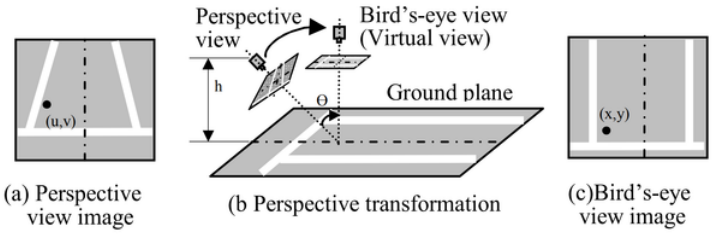


Fig. 1. Illustration of perspective transformation in a parking lot scene

Figura 6.4: Ilustración de la transformación de la perspectiva

Figura 6.5: Transformación real de la perspectiva

Esto tiene principalmente un problema, pese a que a priori puede parecer que la transformación se realiza de forma satisfactoria, se puede observar una distorsión cuando se sitúa delante de la cámara un objeto con una forma geométrica conocida, por lo que es necesario realizar más ajustes hasta que dicha distorsión sea mitigada.

Una vez se dispone de ambas perspectivas, se puede proceder a extraer las líneas. Como se ha mencionado anteriormente esto se suele realizar operando sobre el color, para extraer dicho rasgo distintivo de la imagen.

Existe una gran variedad de opciones, pero las más empleadas se centran en extraer el naranja y el blanco que son los colores con los que generalmente se delimitan los carriles. Para ello, se suele variar el modelo de color a HSV (Hue Saturation Value) o HSL (Hue Saturation Lightness) ya que son representaciones del color que permiten, de forma mucho más sencilla, aislar sus propiedades. El hue o matiz del color se representa con una combinación de rojo, verde y azul y la saturación indica lo colorido o tenue que es. De esta forma, se puede extraer el naranja a partir de un umbral de luminosidad y saturación al igual que las lineas blancas, extraídas a partir de los valores más altos de luminosidad, sin importar el matiz o la saturación. (ver Figura 6.6)

((a)) Imagen con modelo de color HSL **((b))** Imagen binarizada

Figura 6.6: Extracción de lineas mediante binarización

6.4. Aplicación de las técnicas en el prototipo

Una vez analizadas las técnicas empleadas en otras aplicaciones similares y sobretodo las relativas a los coches autónomos, se ha tratado de exportar dichos conceptos junto con otras técnicas variadas y aplicar los más interesantes, adaptándolos al entorno del robot. Las primeras pruebas, se realizaron sobre una celda de laberinto simulada empleando di-

ferentes materiales, pero no representaban un entorno realista, por lo que, posteriormente, todas las pruebas fueron realizadas sobre fragmentos de vídeo reales de Micromouse con una cámara a bordo, extraídos de internet. (ver Figura 6.7)

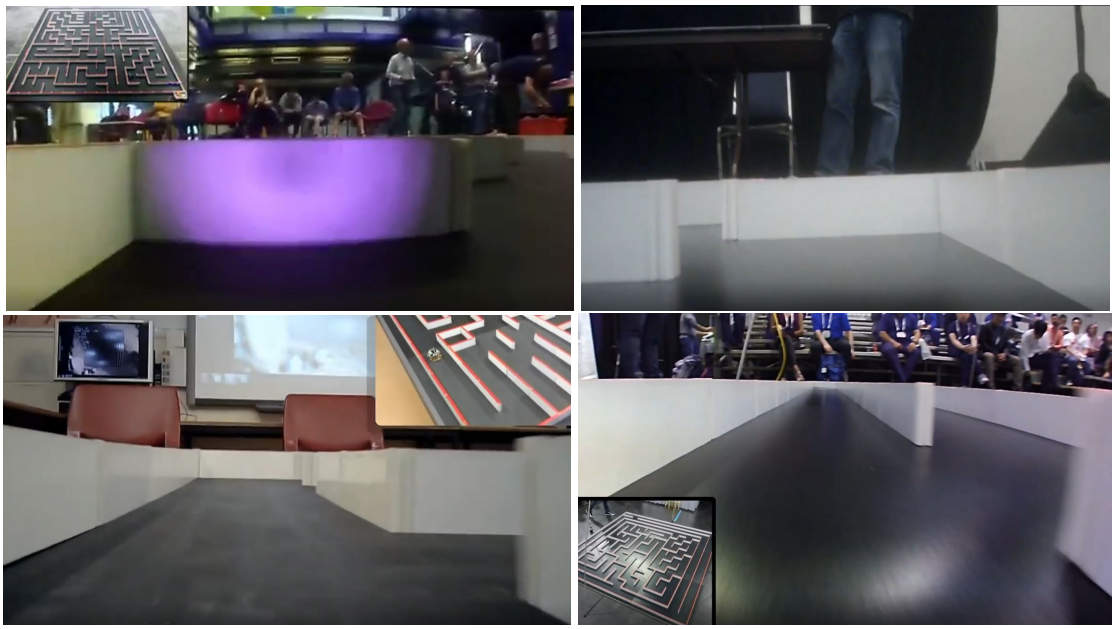


Figura 6.7: Fotogramas de los vídeos de prueba empleados

Como se puede observar, los vídeos empleados no representan un campo de pruebas sencillo (baja resolución, iluminación pobre y reflejos, ruido, distorsiones producidas por la lente...), sin embargo, son muy interesantes de cara al aprendizaje, ya que cada uno supone un reto distinto.

Otro problema añadido, son los recuadros en los que aparece el laberinto tomado desde otra cámara, los cuales eliminan en algunos casos una parte fundamental de la imagen.

6.4.1. Adecuación de la imagen

En este apartado se van a tratar las tareas que ha sido necesario realizar para la adecuación de la imagen. Las tareas tratadas en este punto, tratan de reflejar cronológicamente los avances realizados, y por lo tanto no contienen todos los métodos empleados, los cuales serán mencionados posteriormente, para otorgar una visión de conjunto.

Las primeras aproximaciones se han realizado sobre el vídeo1 (esquina superior izquierda de la Figura 6.7). Como se puede observar, la baja resolución del mismo genera un

pixelado en los bordes, lo cual dificulta la detección de la linea formada por las paredes y el suelo del laberinto. Para ello se ha procedido a aplicar diversos filtros con la intención de suavizarlo.

Como se puede observar, el vídeo dispone de una región que no aporta ninguna información, por lo que se ha retirado empleando una máscara. Además, debido a las características del laberinto cuyas paredes son siempre blancas y los suelos siempre negros, se ha considerado que el color no proporciona información de gran utilidad por lo que se ha decidido que es preferible realizar una conversión a escala de grises.

Filtrado del ruido

El proceso de filtrado de una imagen es una parte crítica dentro del preprocesado de la imagen, ya que hasta los cambios más sutiles generan diferencias notables en los algoritmos que son ejecutados posteriormente sobre la imagen. Para las pruebas se ha comenzado empleando las funciones recomendadas en el apartado de filtrado de la página web de OpenCV [Opencv.org, 2017g] [Opencv.org, 2017k].

El primer filtro de suavizado aplicado ha sido el filtro de covolución 2D, el cual aplica un kernel encima de un píxel, añade los por ejemplo 25 píxeles (en este caso kernel de 5x5) debajo del kernel, obtiene el promedio y reemplaza el píxel central con el nuevo valor promedio. El resultado de dicho filtro no ha sido satisfactorio, por lo que se ha procedido, tal y como indica la propia documentación de OpenCV a emplear técnicas que difuminan en menor medida los bordes.

De este conjunto de funciones, se ha comenzado con el conocido filtro gaussiano. Como se puede ver en las fotos de ejemplo 6.8 y 6.9, trabaja con un kernel cuadrado con un tamaño en píxeles impar, que genera un resultado más o menos borroso y sin detalles dependiendo del tamaño del mismo, yendo de menor a mayor.

También se ha probado otro filtro muy parecido, el denominado filtro de la media, además del filtro de la mediana, todos ellos con resultados ligeramente perceptibles a simple vista pero prácticamente idénticos para las funciones que se ejecutan a continuación. Estas conclusiones se obtienen con un kernel de 5x5, el cual se ha considerado que es el más idóneo por los resultados obtenidos. (ver Figura 6.10)

Tras estos intentos, se ha probado con el filtro bilateral, el cual es efectivo eliminando el ruido pero mantiene definidos los bordes, todo ello a costa de una ligera pérdida de rendimiento.

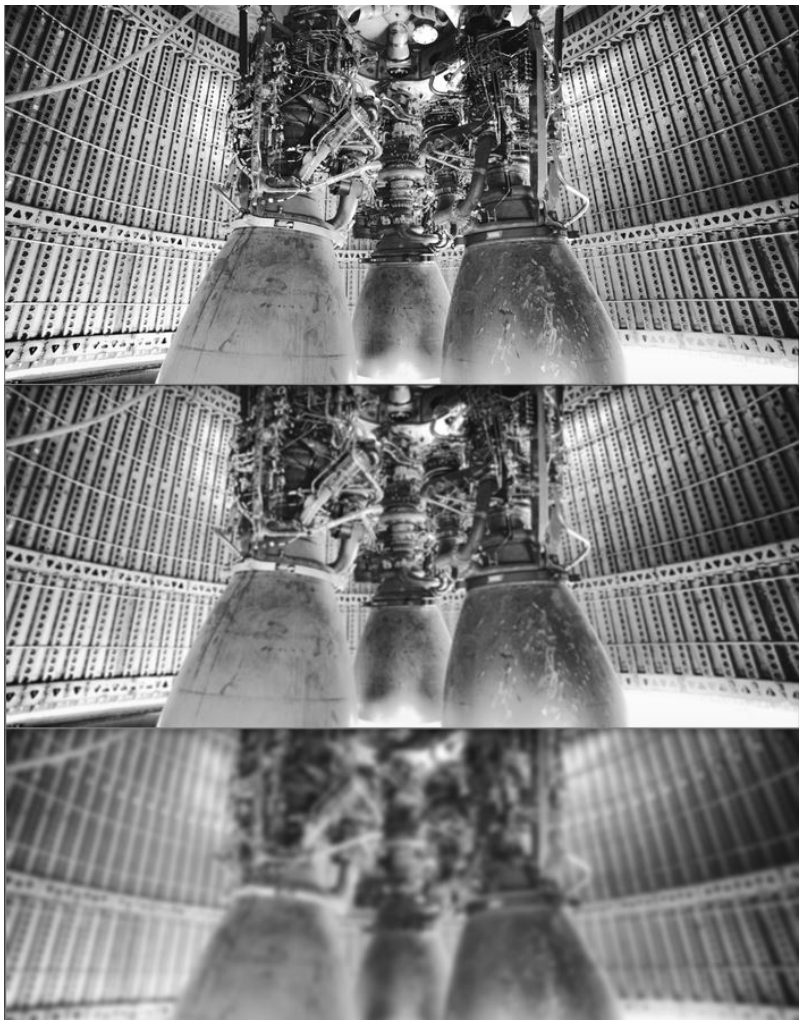


Figura 6.8: Aplicación del filtro gaussiano con kernels de 3x3 y 9x9 sobre una imagen con muchos detalles

En este tipo de filtro, al igual que en el filtro gaussiano, el valor de intensidad en cada pixel de la imagen es reemplazado por una media ponderada de los valores de intensidad de los píxeles cercanos. Pero en este caso, solo se tienen en cuenta los píxeles con intensidades similares al píxel central, preservando los bordes ya que estos están definidos por una gran variación de intensidad.

El resultado obtenido con este filtro, visualmente, es similar a los anteriores, aunque se pueden apreciar diferencias en cuanto al resultado generado tras los algoritmos de detección de bordes, apreciando una mejora interesante.

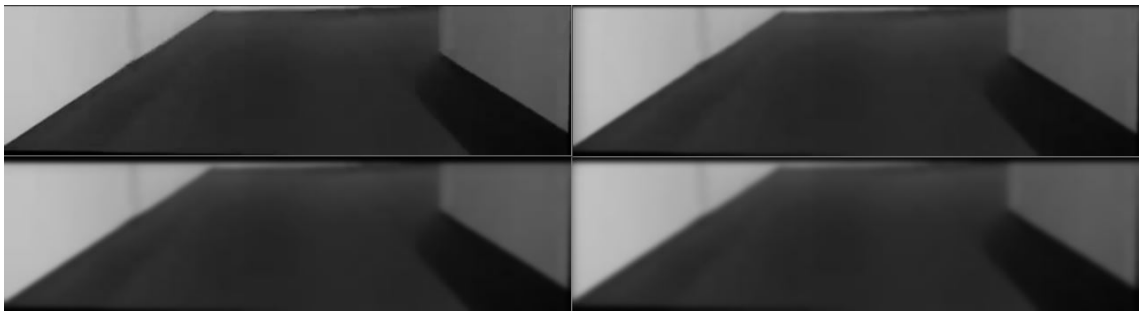


Figura 6.9: Aplicación del filtro gaussiano con diferentes kernels de 3x3, 9x9, 15x15 y 19x19 sobre un fotograma del vídeo1

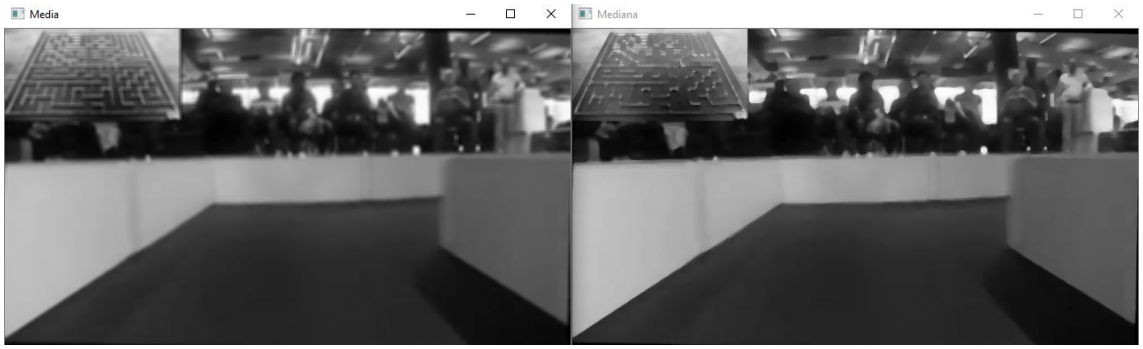


Figura 6.10: Filtros de media y mediana con kernel de 5x5 aplicados sobre un fotograma del vídeo1

6.4.2. Interpretación de la imagen

Una vez realizada una adecuación de la imagen se ha llevado a cabo el proceso de extracción de los elementos que la componen [Opencv.org, 2017d], comenzando con la detección de bordes, con la intención de definir claramente la línea divisoria entre las paredes y el suelo, la cual podrá ser interpretada posteriormente.

Detección de bordes

Las técnicas de extracción de bordes, se emplean para extraer la información estructural, reduciendo enormemente la cantidad de datos que deben ser procesados y por ello son ampliamente empleados en los sistemas de visión por computador.

Para ello, se han realizado pruebas con multitud de funciones disponibles en la librería, pero la dificultad a la hora de elegir las más efectivas ha aumentado exponencialmente

debido al gran número de combinaciones que se pueden realizar entre las funciones de filtrado y de detección de bordes, las cuales están muy relacionadas.

Por este motivo, se comenzó a realizar las pruebas de detección de bordes empleando la imagen tratada con el filtro gaussiano al no apreciar diferencias notables y otorgar prioridad al rendimiento. Como se menciona anteriormente, existe una amplia gama de funciones centradas en la detección de bordes, de las cuales destacan Sobel y Canny.

El operador **Sobel** [Opencv.org, 2017l] se basa en el cálculo de la primera derivada midiendo las evoluciones y los cambios de una variable.

Para la detección de los bordes basada en Sobel de una imagen, se realiza una derivada en el eje x y otra en el eje y, con la intención de analizar los cambios de intensidad en dichas direcciones. Posteriormente, en cada punto de la imagen se obtiene una aproximación del gradiente a partir de los resultados anteriores y se compara la magnitud de dicho gradiente respecto a un umbral, para decidir que píxeles son bordes y representarlos más brillantes en un fondo negro.

Para el empleo de este operador, es necesario realizar una conversión a escala de grises, ya que de lo contrario se procesa cada canal por separado. Como este paso se realiza en el anterior paso, no es necesario tratar la imagen de nuevo.

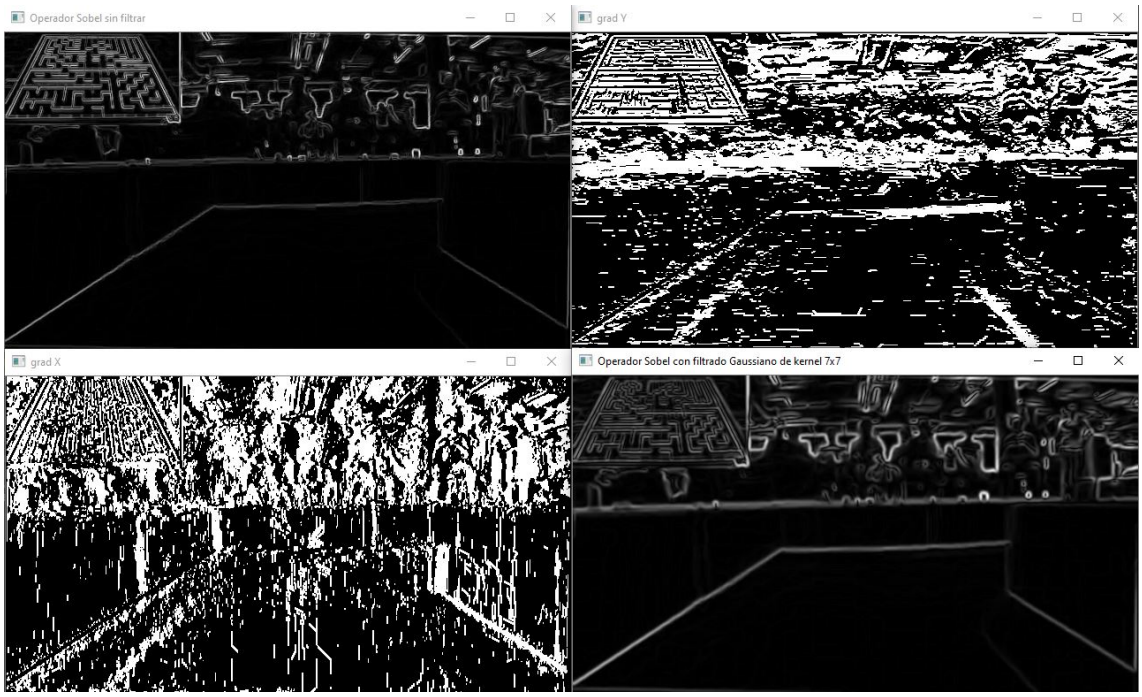


Figura 6.11: Obtención de bordes mediante el operador Sobel, con y sin filtrado previo

Tal y como se ha mencionado anteriormente, se puede apreciar los cambios de intensidad presentes en la imagen,en sentido horizontal y vertical en las ventanas “grad X” y “grad Y”. (ver Figura 6.11)

En dichas imágenes, destacan la gran cantidad de puntos brillantes que indican que hay una diferencia de intensidad, sin embargo, se evidencia la dificultad de obtener las dos líneas formadas entre las paredes y el suelo, debido al ángulo de 45º respecto a la horizontal en el que se encuentran en las imágenes obtenidas por la cámara.

Esta composición a partir de las variaciones en ambos ejes, genera líneas imperfectas o con algo de ruido, como se puede observar en la ventana superior izquierda, pero queda ligeramente mitigado al emplear el filtrado previo, en este caso un filtro Gaussiano con kernel 7x7 cuyo efecto puede ser observado en la ventana inferior derecha.

A su vez, se llevaron a cabo múltiples pruebas con diferentes filtrados previos o con kernels de mayor o menor tamaño, los cuales mostraron resultados dispares. Estas pruebas, expusieron dos tendencias generales, a mayor tamaño del kernel de filtrado mayor grosor del borde pero peor definido y a mayor kernel en el operador de Sobel mayor cantidad de ruido y líneas no deseadas. Dichas tendencias, tienen un problema en común, la necesidad de realizar operaciones a posteriori sobre las imágenes obtenidas, de cara a mejorar la interpretación de las líneas, ya que es necesario adecuar las imágenes de entrada a las características propias de los algoritmos encargados de interpretar dichas líneas.

Por otro lado, **Canny** probablemente sea el método más utilizado [Opencv.org, 2017a]. Es un método multietapa, que al igual que el método de Sobel emplea el gradiente de la imagen, pero en este caso emplea dos umbrales los cuales otorgan un mayor control para la elección del borde deseado.

Al igual que el anterior operador, Canny es sensible al ruido, por lo que es necesario realizar un filtrado previo al tratamiento de la imagen, preferentemente Gaussiano, debido a que pese a que la función óptima se describe como la suma de cuatro términos exponenciales, puede ser aproximada empleando la primera derivada de una función Gaussiana.

Como el borde de una imagen puede estar orientado en diferentes direcciones, este algoritmo utiliza cuatro filtros: horizontal, vertical y dos diagonales.

El algoritmo de detección de bordes de Canny está compuesto por 5 pasos:

1. Aplicación de un filtrado para eliminar el ruido.
2. Encontrar los gradientes de intensidad de la imagen.

3. Aplicar el umbral de magnitud de gradiente o la supresión de los casos por debajo del límite inferior para deshacerse de los bordes ficticios.
4. Aplicar un umbral doble para delimitar la detección de bordes.
5. Realizar un seguimiento de la arista por histéresis con la intención de suprimir todas las aristas débiles y no conectadas a aristas fuertes.

Debido a la creciente necesidad de algoritmos robustos y precisos, el algoritmo de Canny se ha visto superado y ha necesitado una serie de mejoras:

- El filtrado Gaussiano, supone una traba, ya que se requiere un filtro que suavice de forma efectiva el ruido, pero generando una distorsión menor de los bordes. Esto se ha logrado con filtros adaptativos que evalúan la discontinuidad entre los valores de gris de cada píxel. Otros métodos similares han mostrado igualmente resultados prometedores.
- Mejoras en las técnicas de cálculo de la magnitud y la dirección del gradiente. La magnitud y la dirección pueden ser calculadas empleando una gran variedad de operadores de detección de bordes. El más común es emplear un Sobel de 3x3 pero existen opciones mejores como un Sobel con kernel de 5x5 o el operador de Prewitt.
- Método robusto para la identificación de los umbrales. La identificación de los dos umbrales de forma empírica es una tarea compleja ya que depende de multitud de variables, sobre todo en el caso del tratamiento de un vídeo ya que las condiciones van cambiando constantemente. Para solventarlo, se suele emplear métodos que automatizan parcialmente el proceso como puede ser el método de Otsu.
- El adelgazamiento del borde. El método tradicional no siempre proporciona una respuesta única por borde, por lo que se han desarrollado técnicas de morfología matemática para adelgazar el borde detectado
- Uso de curvelets (extensión de la transformada de Ondícula). Estos se han utilizado en lugar del filtro gaussiano y la estimación del gradiente para calcular un campo vectorial cuyas direcciones y magnitudes se aproximan a la dirección y la intensidad de los bordes de la imagen.

La viabilidad de estas mejoras, ha sido estudiada, pero gracias a las características estables del entorno del Micromouse, se ha considerado que el coste beneficio de aplicarlas no

justificaba su empleo, ya que se obtenían resultados apropiados con el método tradicional acompañado de pequeñas modificaciones. Por otro lado, las posibilidades de mejora se han visto limitadas a las proporcionadas por la librería, ya que no era objeto de este TFG profundizar en este tipo de algoritmos y su implementación.

Debido a las características de este método de detección de bordes, ha sido necesario realizar varias aproximaciones para obtener un resultado adecuado.

Como se ha mencionado anteriormente, el filtrado, es uno de los puntos a mejorar de la técnica tradicional del algoritmo Canny. Este aspecto, ya ha sido tratado en el punto previo "preprocesado", pero es importante mencionar que se han revisado en este punto una gran cantidad de combinaciones diferentes dentro del abanico de posibilidades que ofrece OpenCV, al tener más importancia los resultados obtenidos tras realizar la operación de obtención de bordes que los resultados basados en la percepción a simple vista.

El otro aspecto fundamental ha resultado la elección empírica de los dos umbrales que delimitan la detección de los bordes. Este ha sido el mayor problema al que me he enfrentado en este paso. Es una tarea mucho más compleja de lo que parece a simple vista.

El valor de estos umbrales depende enormemente de las características de la imagen, de como haya sido tratada previamente y sobretodo si se trata de un vídeo, en el que cada fotograma puede presentar unos cambios notables respecto a los anteriores.

Para este menester, se ha creado un pequeño panel de control, al que se le han ido añadiendo funciones. Gracias a este panel, se ha logrado cambiar durante las pruebas, en tiempo de ejecución el valor de ambos umbrales junto con el de otros parámetros relacionados como pueden ser el tamaño del kernel del filtro empleado en ese momento, con la intención de aumentar la variabilidad de las pruebas y obtener de forma ágil conclusiones. (ver Figura 6.12)

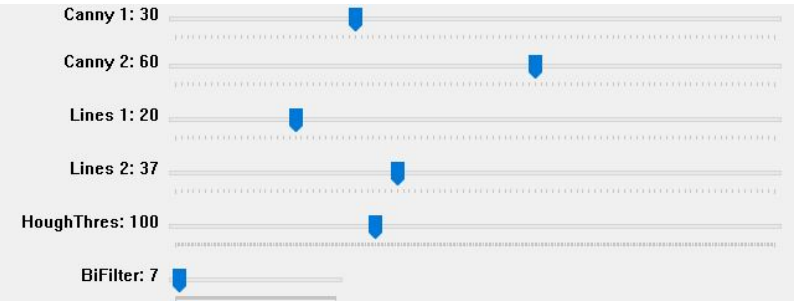


Figura 6.12: Fragmento del panel de control empleado en las pruebas

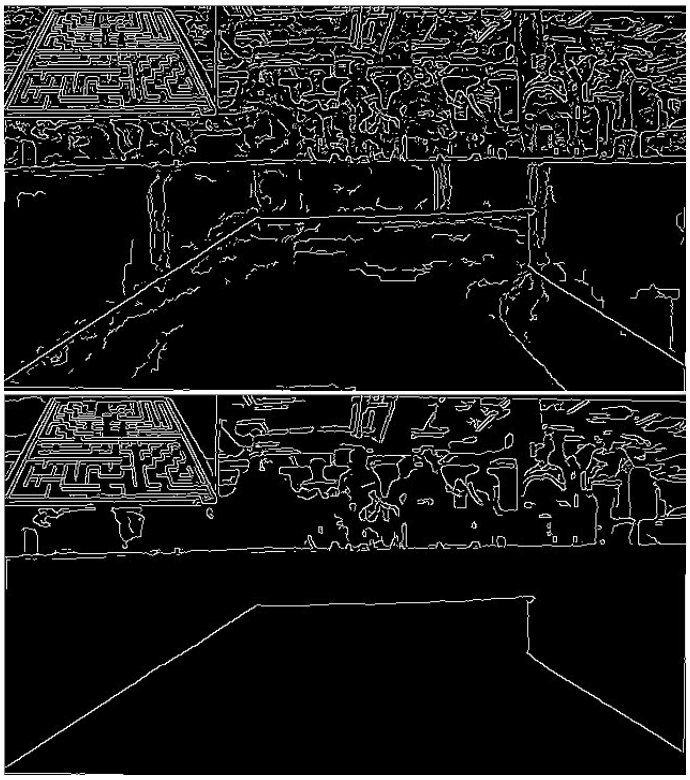


Figura 6.13: Extracción de bordes mediante Canny con el segundo umbral bajo o alto

Los siguientes ejemplos son una pequeña representación de las pruebas realizadas y la gran variedad de resultados que proporcionan: Como se puede observar en las imágenes 6.13, una pequeña diferencia en los umbrales, en este caso el segundo umbral, tiene una gran repercusión sobre el resultado, de ahí que resulte crítico encontrar unos valores adecuados. Esto, una vez más depende enormemente del filtrado previo, por lo tanto todos los ajustes deben ir acorde.

En conclusión, como se puede observar, el resultado del método de extracción de bordes de Canny es mejor que el del método de Sobel. Se podría incluso afirmar que es una versión mejorada del mismo o una versión más completa. Por lo tanto, para este proyecto se ha decidido emplear este método.

Mejora de bordes

Otro aspecto a tener en cuenta para este proyecto, son las líneas que forman los bordes. Como se puede apreciar, una persona podría identificar perfectamente el trazado con el resultado obtenido, pero si se trata de procesar dichas líneas de forma matemática surgen varios problemas, ya que no son líneas perfectamente rectas, ni continuas, ni de un grosor regular.

Una forma de solventar parcialmente este problema son las operaciones morfológicas, como por ejemplo la erosión, la dilatación y las operaciones que hacen uso de ambas. Estas operaciones generalmente se agrupan en el conjunto de opciones para la adecuación de la imagen, pero como se menciona en el capítulo [5] "Visión", no existe un orden determinado para la aplicación de herramientas de tratamiento de una imagen, pudiendo iterar varias veces sobre el mismo paso o volver reiteradamente a pasos anteriores.

La operación de dilatación, la cual es explicada en el capítulo de visión, cuando actúa sobre los bordes obtenidos en el anterior paso genera un efecto expansivo sobre el borde, rellenando huecos y aumentando el grosor del mismo. Al igual que sobre el borde, esta operación también se aplica al resto de píxeles de la imagen, pudiendo llegar a expandir alguna línea indeseada o generar imperfecciones o alteraciones en la forma de los objetos. (ver Figuras 6.14 y 6.15)

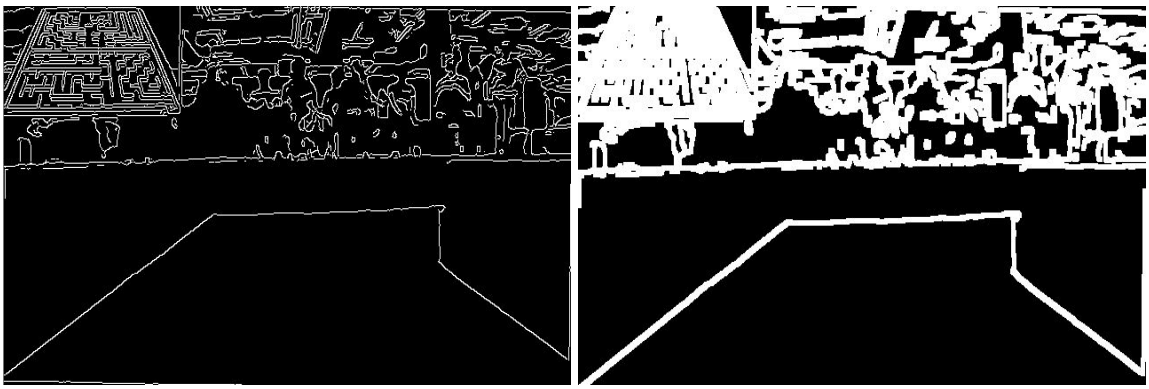


Figura 6.14: Operación morfológica de dilatación (2 iteraciones, kernel 3x3) sobre un fotograma (Aplicado método Canny)

La operación de erosión, realiza el efecto opuesto, simula una erosión, tras la cual el grosor de la línea disminuye. (ver Figura 6.16)

Esta operación no tiene sentido aplicarla en algunas imágenes con patrones finos, ya que

Figura 6.15: Operación morfológica de dilatación (2 iteraciones, kernel 3x3) sobre un fotograma con un umbral Canny excesivamente bajo (Aplicado método Canny)

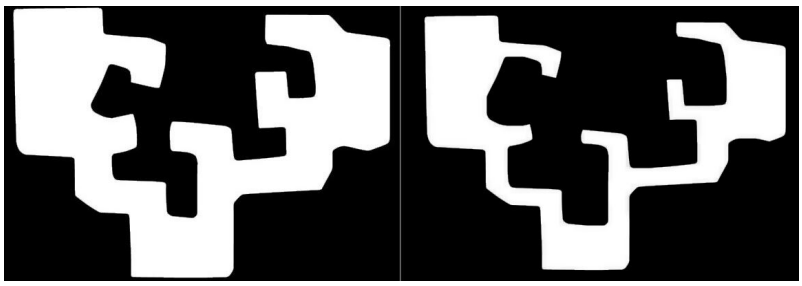


Figura 6.16: Operación de erosión aplicada al logo de la UPV/EHU

se obtiene una imagen completamente en negro, como sería el caso del fotograma del vídeo.

Al igual que la operación de dilatación, la operación de erosión puede ser realizada empleando diversos tamaños de kernel, e incluso kernels con distintas morfologías. Estas operaciones, son iterativas, es decir, se pueden aplicar reiteradas veces sobre el resultado de la anterior. En la librería OpenCV esta operación es muy sencilla gracias a un parámetro que permite indicar el número de veces que se va a realizar [Opencv.org, 2017c] [Opencv.org, 2017h].

Como se puede apreciar en estas imágenes estas operaciones por separado pueden llegar a generar problemas, llevando a analizar en cada caso el coste-beneficio de aplicarlas. Un borde con mucho grosor (ver Figura 6.17) o un borde muy fino e inconexo generan resultados indeseados al ser interpretados posteriormente. Sin embargo, estas operaciones son de gran utilidad si se combinan de forma que eliminen el ruido presente, o mejoren la calidad de las líneas.

Como se puede observar en la Figura 6.18 el resultado cuenta prácticamente con las características deseadas, líneas mejor definidas, conexas y con bordes más regulares, o al

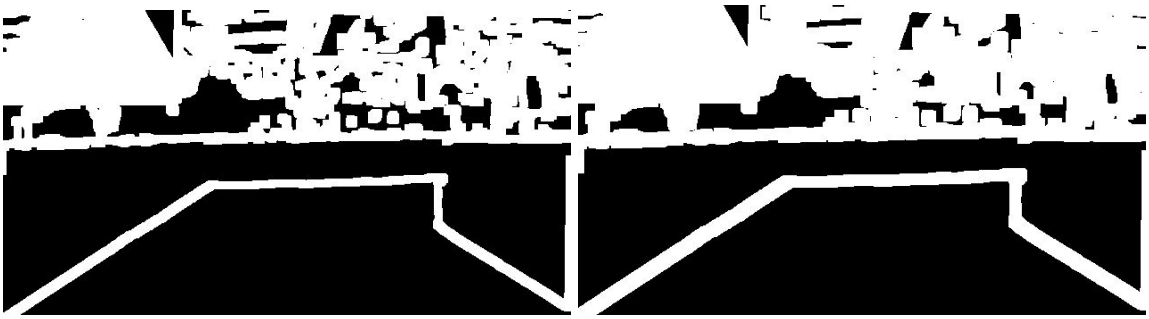


Figura 6.17: Operación morfológica de dilatación con 4 y 6 iteraciones respectivamente (kernel 3x3) sobre un fotograma (Aplicado método Canny)

Figura 6.18: Operación morfológica de dilatación (3 iteraciones, kernel 3x3) seguida de una erosión (2 iteraciones, kernel 3x3)

menos, propensos a ser detectados de forma adecuada por los algoritmos encargados de interpretarlos posteriormente.

Una vez aisladas las características deseadas, se pasa al proceso de extracción de las mismas.

Extracción de las características

El objetivo fundamental de este punto, es lograr interpretar las líneas divisorias entre las paredes y el suelo, las cuales han sido acentuadas y aisladas de la mayoría de información presente en la imagen gracias al trabajo previo. La interpretación, hace referencia a la conversión de las líneas de la imagen a una representación matemática, empleando puntos. Estos puntos están posicionados dentro del mapa de píxeles en unas coordenadas concretas que pueden ser interpretadas fácilmente por otros algoritmos.

Con este fin se han empleado tres tipos de algoritmos que se podrían agrupar de la siguiente forma:

1. Basados en la obtención de figuras
2. Basados en la obtención de puntos
3. Basados en la interpretación de contornos

Al tratarse de líneas, el razonamiento inicial más lógico, es tratar de emplear un algoritmo de obtención de figuras, más concretamente de líneas.

Transformada de Hough

Es una técnica que permite encontrar figuras que puedan ser expresadas matemáticamente como rectas, circunferencias o elipses [Wikipedia.org, 2020].

La implementación en OpenCV [Opencv.org, 2017f] de esta técnica emplea dos parámetros gracias a los cuales se puede elegir la longitud mínima de la línea y la distancia máxima entre los segmentos, todo ello junto a un umbral de intersecciones necesarias para ser considerada línea.

Esta operación tiene como salida una lista de líneas, las cuales están formadas por pares de “rho” y “theta” ya que emplea un sistema de coordenadas polares. ρ representa la distancia entre el origen de coordenadas y el punto (x, y), mientras que θ es el ángulo del vector director de la recta perpendicular a la recta original y que pasa por el origen de coordenadas.

A partir de esos dos elementos, se pueden representar las líneas sobre la imagen empleando la función “cv2.line” [Opencv.org, 2017b] teniendo la posibilidad de emplear un código de colores para distinguir las diferentes líneas dependiendo del ángulo. (ver Figura 6.19)

Los fotogramas empleados para el ejemplo son la selección de una región de interés del empleado en el resto de ejemplos, ya que las personas y el laberinto situados en la parte superior alteraban el resultado generando un número excesivo de líneas. También es importante mencionar, que la transformada de hough se realiza sobre las imágenes con bordes realzados de la izquierda y se representan las líneas obtenidas sobre la imagen original.

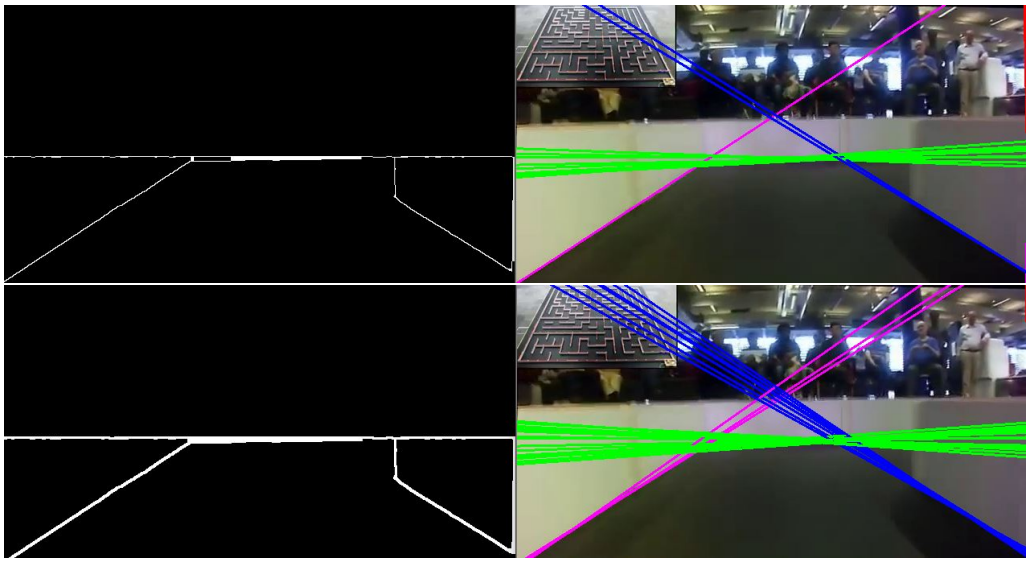


Figura 6.19: Obtención de las líneas mediante la transformada standard de Hough, aplicada sobre dos fotogramas con distinto grosor de borde

Como se puede apreciar en las imágenes, con esta técnica los resultados dependen enormemente del grosor de los bordes de la imagen, ya que si cuentan grosor considerable esta función puede llegar a detectar varias líneas.

Por otro lado, con bordes finos puede suceder lo contrario, y no encontrar ninguna línea, ya que si dicha línea no es lo suficientemente recta o continua el algoritmo no es capaz de tenerla en cuenta. Por lo tanto, surgen nuevos tipos de problemas, relacionados con la geometría de las formas presentes en la imagen. Para solventar estos problemas, existe un gran abanico de posibilidades, multiplicando aún más el número de alternativas disponibles y la dificultad de elección de un conjunto de ellas.

Otro aspecto importante, es la cantidad de cómputo que requiere esta función, haciéndola poco atractiva de cara a ser empleada en un pequeño robot, cuya capacidad de procesamiento es limitada. Por ello, se suele emplear la transformada de Hough probabilística la cual es más eficiente y se encuentra igualmente implementada en la librería OpenCV. (ver Figura 6.20)

Como se puede apreciar la diferencia entre ambos resultados es considerable, pasando de unos pocos segmentos en la imagen inferior a más de mil en la superior. Esto se debe a la modificación de los parámetros disponibles en la implementación de la transformada en la librería OpenCV [Opencv.org, 2017e].

En la imagen superior se ha empleado un tamaño máximo de segmento reducido, junto

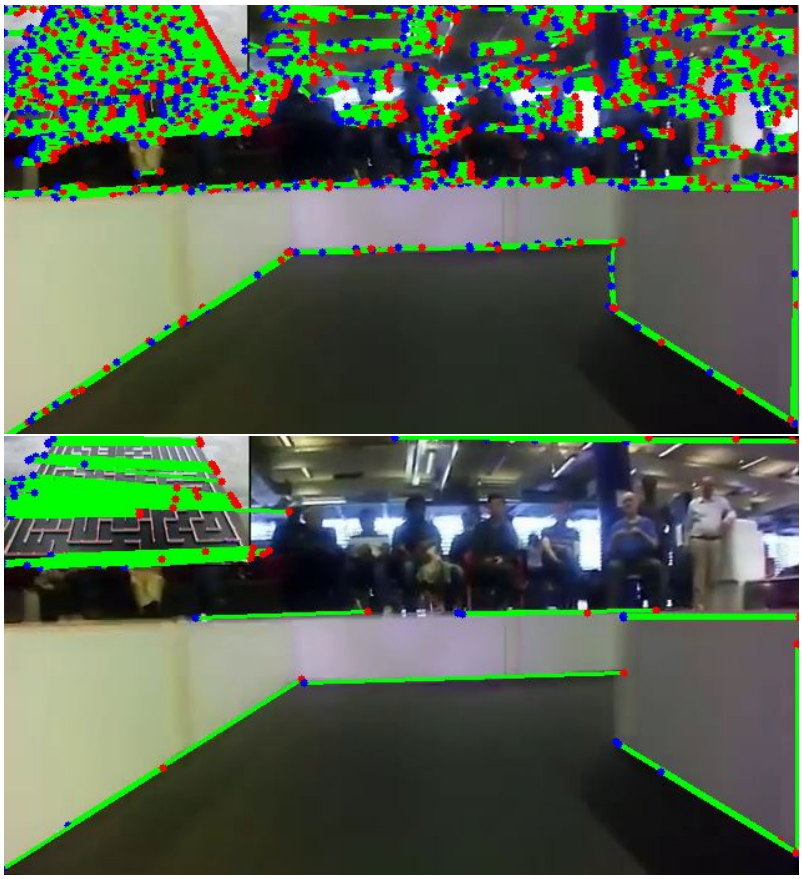


Figura 6.20: Obtención de las líneas mediante la transformada probabilística de Hough, aplicada sobre el mismo fotograma pero modificando el umbral, la longitud máxima de los segmentos y la longitud máxima entre segmentos

con una distancia máxima entre segmentos también reducida y un umbral elevado.

Esto lleva a la interpretación de los bordes obtenidos anteriormente como una enorme cantidad de segmentos que definen perfectamente los bordes del trazado al que se va a enfrentar al robot. Sin embargo, tal cantidad de puntos son difícilmente tratables, ya que aumentan la necesidad de cómputo y la semántica requerida para la obtención de conclusiones, además de resultar innecesarios.

Por otro lado, si se realiza un buen ajuste de estos parámetros, se puede obtener una imagen como la inferior de la Figura 6.20, en la que cada borde es interpretado empleando como máximo uno o dos segmentos.

Para ello, se cuenta con otro parámetro extra, un umbral, el cual hace referencia al número mínimo de intersecciones necesarias para detectar una línea. Es decir aumenta el grado de certeza de que donde se ha interpretado dicho segmento, hay una línea.

Otra opción muy interesante son los algoritmos basados en la obtención de puntos, o mejor dicho, puntos clave.

Detectores de esquinas (Shi-Tomasi o GoodFeaturesToTrack()

Se basan en la obtención de las esquinas más prominentes de una imagen o de una región de dicha imagen. El algoritmo de Shi-Tomasi, es una evolución del algoritmo de detección de esquinas de Harris y la implementación en OpenCV [Opencv.org, 2017j] permite seleccionar o bien el método de Shi-Tomasi o el de Harris. La implementación de este método, tiene como entrada una imagen en escala de grises y permite indicarle el número de esquinas que se desea encontrar, un valor umbral entre 0 y 1 para la selección de la calidad de dichas esquinas y una distancia euclídea mínima entre las esquinas detectadas.

A partir de dichos parámetros, la función busca todas las esquinas presentes en la imagen y desecha las que se encuentren por debajo del umbral. Posteriormente, ordena las restantes en orden descendente y elige la esquina más prominente, desecha todas las que se encuentren en la distancia mínima indicada por parámetro y devuelve las N esquinas que destaquen más. (ver Figura 6.21)

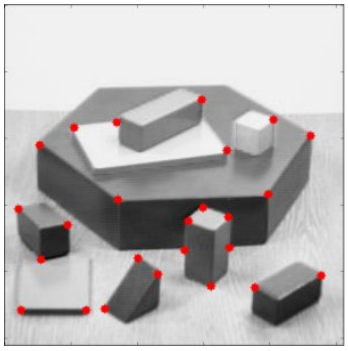


Figura 6.21: Ejemplo de la aplicación de la función goodFeaturesToTrack() (método de Shi-Tomasi) obtenida de la documentación de OpenCV

Como se puede apreciar en la Figura 6.22, la identificación de los bordes por parte de esta función es muy efectiva y permite definir perfectamente el borde mediante los puntos. Gracias a los parámetros de configuración, se puede tener un alto control del tipo de representación deseada, yendo desde una gran cantidad de puntos por sección, muy próximos, a una representación muy simplificada formada por un número muy reducido de ellos, contando prácticamente con uno por esquina.

Nota: Los puntos situados en la parte superior de la imagen, son falsos positivos derivados

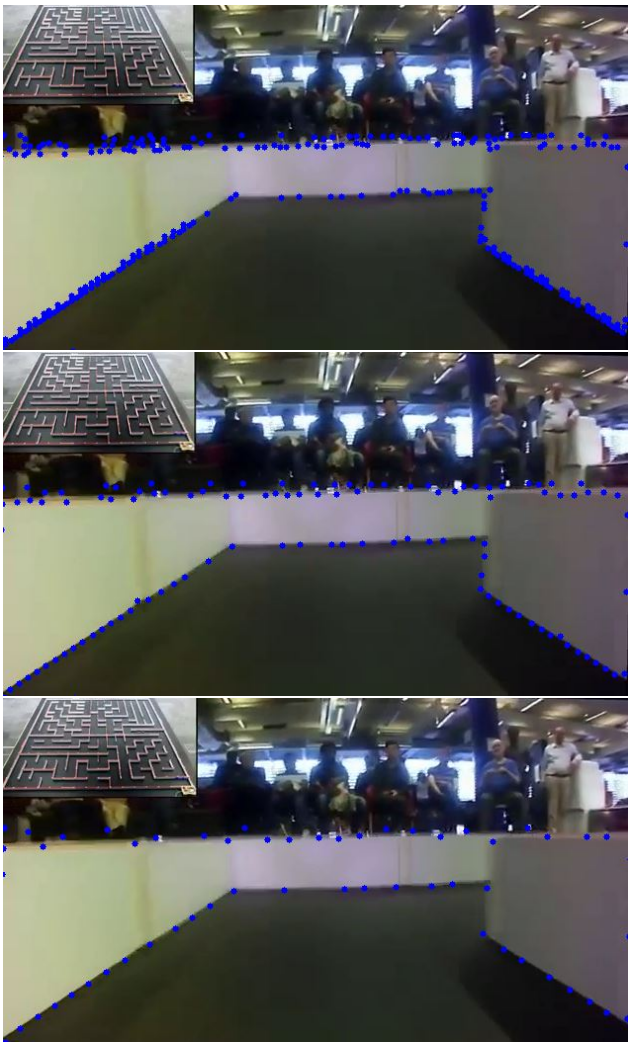


Figura 6.22: Ejemplo de la aplicación de la función goodFeaturesToTrack() con diferentes parámetros sobre el mismo fotograma

de la extracción de una región de la imagen para eliminar la parte no correspondiente al laberinto. Este problema es debido a las características del vídeo (imagen pobre y ángulo no adecuado), pero puede ser solucionado en las pruebas reales.

Esta representación del entorno mediante puntos, permite afrontar la navegación empleando una amplia variedad de estrategias, dependiendo de la interconexión que se realice entre esa nube de puntos.

Por lo tanto, se podría afirmar que una gran cantidad de puntos, permitiría definir con mayor exactitud un borde. Sin embargo, una mayor cantidad de puntos no siempre es interesante, como ocurre con la imagen superior de la Figura 6.22, ya que puede suceder que se generen agrupaciones alrededor de ciertas áreas. Esto se puede solucionar, aumentan-

do la distancia mínima entre esquinas y variando el umbral de detección, como se puede observar en la imagen central de dicha figura.

No obstante, debido a las líneas rectas formadas por las paredes del laberinto, en este caso es más adecuada una aproximación con menos puntos, como se puede observar en la imagen inferior de la Figura 6.22.

Por último, es interesante mencionar la detección de contornos, ya que es una funcionalidad muy versátil que puede ser empleada de múltiples de formas para la extracción de características reseñables o interesantes para la navegación.

Detector de contornos findContours()

Otra forma de interpretar los bordes de la imagen es la detección de contornos. Partiendo de la consideración de que un contorno está formado por los límites de una forma cuyos puntos tienen la misma intensidad, este método almacena las coordenadas cartesianas de dicho contorno.

Como dicha información no siempre es útil, la implementación en OpenCV [Opencv.org, 2017i], permite la elección del método de aproximación al contorno. Este puede ir desde el almacenamiento de todos los puntos del contorno, hasta una aproximación muy simplificada, en la cual emplea únicamente dos puntos, eliminando así toda la información innecesaria, por ejemplo en caso de aplicarse sobre una línea recta.

Para el correcto funcionamiento de esta función, es altamente recomendado realizar un procesado previo de la imagen además de una mejora de bordes. Por lo tanto, las mejoras que han sido aplicadas en la imagen hasta este punto, son el mejor punto de partida para esta función.

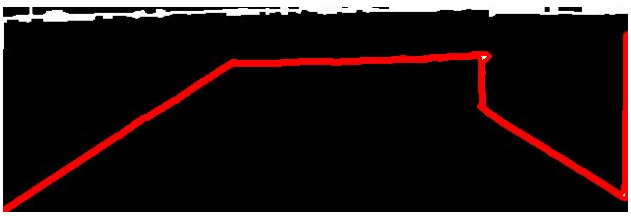


Figura 6.23: Ejemplo del uso de la función findContours()

Como se puede apreciar en la Figura 6.23, esta función es capaz de detectar todo el borde inferior de la imagen, en este caso en un solo contorno, debido al efecto visual y a las operaciones realizadas anteriormente. Esta función cuenta con una gran cantidad de fun-

ciones auxiliares en la librería OpenCV, las cuales permiten su uso en una gran variedad de aplicaciones. La más destacada, es la encargada de dibujar los contornos obtenidos en la imagen, llamada “drawContours()” la cual tiene una opción muy interesante que permite seleccionar cual de los contornos obtenidos representar, basándose en una jerarquía. Se entiende como jerarquía a la relación entre los contornos, ya que, por ejemplo, pueden estar contenidos unos dentro de otros. Esta jerarquía se elige en el proceso de búsqueda de bordes, ya que es uno de los parámetros de la función “findContours()”.

En conclusión, se puede apreciar que es una herramienta con mucho potencial, pero en principio, su aplicabilidad parece ser menor que la del resto de métodos de extracción de características. Sin embargo, más adelante se demuestra que puede ser de gran utilidad empleada con un enfoque distinto.

Semántica o código de control

Una vez llegado a este punto, es necesaria la implementación de un código que se encargue de la conversión de las líneas o puntos obtenidos referentes a las paredes del laberinto a un conjunto de datos que puedan ser empleados para la navegación. Para este trabajo, existen dos conjuntos de datos fundamentales: los relativos a las distancias entre el robot y las paredes (frontal, y laterales) y los relativos a la existencia o no de un camino viable, es decir, a si existen paredes interponiéndose entre las celdas. Ambos conjuntos de datos están profundamente ligados e incluso se pueden llegar a deducir unos a partir de los otros. Sin embargo, la perspectiva actual de la cámara del vídeo no permite obtener de forma fiable todos los datos. Para solventar este problema, existen dos soluciones: mover físicamente la cámara o aumentar su número, o bien realizar una transformación virtual de la perspectiva . En este caso, al tratarse de un vídeo de pruebas, no se dispone de la primera opción, por lo que se procede a realizar la transformación virtual.

Transformación de la perspectiva

La transformación de la perspectiva (ver Sección 5.3.5) es un tipo de transformación geométrica muy empleada en visión por computador ya que permite la obtención de información muy útil sin la necesidad de alterar físicamente la cámara. Este es uno de los métodos más populares empleados en la conducción autónoma ya que entre otras cosas permite mantener al vehículo dentro de los carriles (ver Figura 6.5). En este caso, su aplicabilidad es más que evidente ya que las líneas que forman las paredes con el suelo del

laberinto pueden hacer las veces de las líneas que delimitan un carril en la carretera. Además, tras los pasos realizados, contamos con unas líneas con el grosor adecuado y aisladas del resto de la información del entorno. Una vez aplicada la transformación se obtiene la siguiente imagen: Figura 6.24

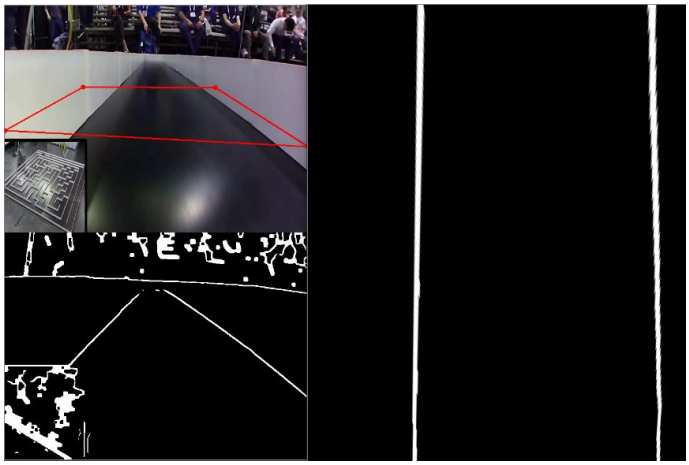


Figura 6.24: Transformación de la perspectiva aplicada sobre un fotograma del vídeo 4

Para una mejor comprensión de la operación, se ha representado sobre la imagen original (esquina superior izquierda de la Figura 6.24), en color rojo, el perímetro del área que va a ser transformada. Esta representación, es muy útil ya que la elección del área a transformar es de vital importancia y cualquier leve variación puede llegar a generar grandes distorsiones o la generación de una imagen inservible. Por todo ello, se ha creado un pequeño panel de control encargado de controlar la ubicación de los cuatro puntos origen, para realizar pruebas hasta la obtención de la perspectiva adecuada.

La operación de transformación se realiza en dos pasos, primero la función "getPerspectiveTransform()" calcula la matriz de transformación a partir de los cuatro puntos de origen y los cuatro puntos de destino. En este caso los cuatro puntos de origen son los cuatro puntos marcados en rojo mencionados anteriormente y los puntos de destino son las esquinas de la nueva imagen. Posteriormente, la función "warpPerspective()" aplica la transformación de la imagen, empleando la matriz obtenida en el paso anterior.

Como se puede observar, el resultado es satisfactorio pese a estar limitado por las características del vídeo, ya que hay un recuadro en la zona inferior el cual tapa la zona más cercana al robot, es decir, la parte más importante para el guiado del mismo y a esto se suma la tendencia del robot del vídeo a escorarse hacia la pared izquierda. Aún así, los vídeos han supuesto un campo de pruebas muy interesante y por ello han sido empleados

reiteradamente hasta disponer del prototipo completamente ensamblado.

En este punto, tras la obtención de la imagen deseada, se vuelve a generar el dilema de cual es la técnica adecuada para la interpretación de las líneas. Por este motivo, se ha procedido a la repetición de las pruebas, empleando la transformada de Hough (estandar y probabilistica), la detección de esquinas y la detección de contornos.

Tras numerosas pruebas, se han obtenido resultados muy prometedores con todos los métodos. Todos ellos, son capaces de reconocer claramente las líneas, permitiendo emplearlas como puntos de referencia para la navegación del robot, lo cual es un gran avance. Sin embargo, estas líneas no son reconocidas como una unidad, sino que los algoritmos las interpretan como una sucesión de segmentos, lo cual requiere una interpretación posterior de los datos obtenidos para poder emplearlos.

Corrección de la distorsión:

Tras las pruebas realizadas empleando métodos de extracción de las características, se hizo evidente que por algún motivo las funciones utilizadas no reconocían las líneas como una unidad, pese a parecer más o menos rectas a simple vista. Y esto derivaba en la necesidad de implementar más código, restando rendimiento y añadiendo complejidad, para la gestión de la multitud de líneas y puntos generados.

Al principio esto fue achacado a la mala calidad del vídeo y las condiciones en las que había sido tomado, pero posteriormente al ir mejorando la calidad de los bordes, se percibía de forma más evidente la ligera curvatura que tienen las formas de la imagen, producto de la distorsión del objetivo de la cámara (ver Figura 5.2.3).

Como no se tenía un control sobre el vídeo, se dejó de lado, dando por hecho que al tener control sobre la cámara del prototipo esto iba a poder ser solventado sin mayores contratiempos, pero al realizar en uno de los pasos la transformación de perspectiva, se descubrió la existencia de un método que permitía la corrección de la distorsión producida por la cámara (ver Sección 5.3) consistente en la realización de un calibrado de la misma con la intención de obtener los parámetros relativos a la distorsión del objetivo y la aplicación de una transformación correctiva de la imagen.

Para este fin, Opencv dispone de un conjunto de funciones [Opencv.org, 2015] que facilitan enormemente la tarea. Por un lado, es necesario el reconocimiento de patrones, que empleados como puntos de referencia ayudan a obtener los valores relativos a la dis-

torsión. Con este fin, se suele emplear la función “findChessboardCorners()” que detecta tableros de ajedrez y la disposición de sus recuadros, este proceso se realiza en múltiples imágenes tomadas con diferentes ángulos y en distintas posiciones.

Posteriormente, se realiza la calibración con la función “calibrateCamera()”, a la cual se le pasan las listas de puntos obtenidos en la función anterior y esta devuelve una serie de parámetros como la matriz de la cámara, los coeficientes de distorsión, la rotación y los vectores de translación los cuales pueden ser empleados con distintos propósitos. Como en este caso la intención es reducir la distorsión, la función “undistort()” es la elegida ya que empleando la matriz de la cámara y los coeficientes de distorsión obtenidos en el paso anterior devuelve la imagen perfectamente rectificada.

Como se puede apreciar, estas funciones facilitan enormemente la calibración, sin embargo, muchas veces los resultados no son los esperados. Esto se debe principalmente a la mala detección del tablero de ajedrez situado en las imágenes de referencia. Para evitar este problema, existen funciones que representan gráficamente la detección del patrón, como “drawChessboardCorners()“, permitiendo seleccionar únicamente las imágenes en las que este ha sido reconocido sin errores. (ver Figura 6.25)

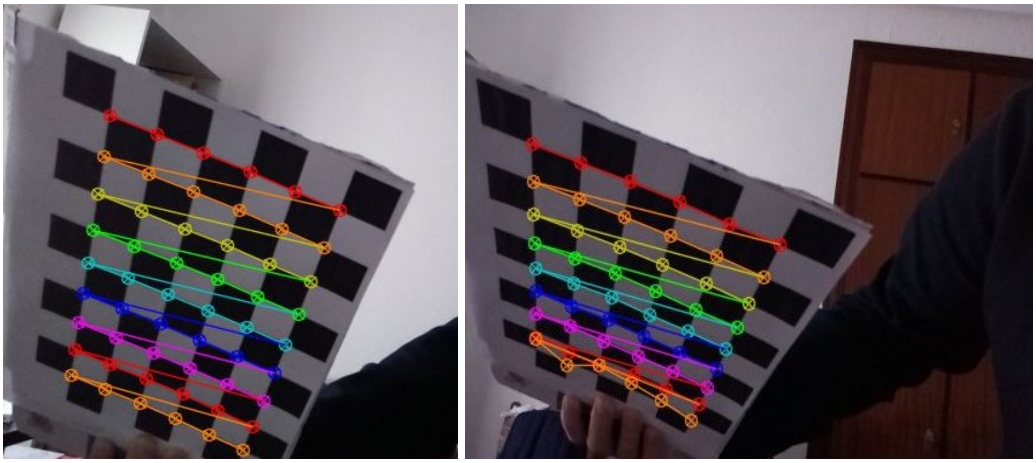


Figura 6.25: Ejemplo de detección satisfactoria (izq) y fallida (dcha) del tablero de ajedrez en las pruebas realizadas para el prototipo

Una vez corregida la distorsión, se puede proceder a la extracción de los datos relativos a las líneas, ya que ahora son claramente más rectas y por lo tanto fáciles de interpretar. Por lo tanto, puede resultar interesante un replanteamiento del procesado previo de cara a reducir el cómputo necesario o bien con la intención de obtener mejores resultados.

Sin embargo, este procedimiento no puede ser aplicado a los vídeos de pruebas ya que

como se ha mencionado anteriormente no se dispone de la cámara para poder realizar el proceso de calibración, ni se dispone a estas alturas de un prototipo en el que poder efectuar las pruebas, por lo que este replanteamiento se aplaza hasta que esto sea posible.

Navegación mediante vista de pájaro

Al no poder aplicar una corrección de la distorsión, fue necesario implementar código que se encargara de estimar donde se encontraban las líneas a partir de los puntos obtenidos, con la intención de mantener permanentemente al robot centrado entre ambas paredes. Con este fin se realizaron varias aproximaciones, pero se veían condicionadas por la cantidad de puntos, ya que si se optaba por el empleo de un número elevado de ellos la carga computacional era excesiva y si se empleaba un número reducido, podían darse situaciones en las que las paredes no fueran interpretadas de forma estable. Tras varias iteraciones de ensayo y error se lograron varios métodos efectivos, pero surgió el problema de las celdas en las cuales no existía una pared lateral, es decir, intersecciones entre caminos.

Debido a la perspectiva, la gestión de estas intersecciones era complicada y se llegó a la conclusión de que lo ideal era el empleo de la imagen original y de la vista de pájaro simultáneamente si la capacidad de cómputo del prototipo lo permitía, ya que las pruebas se estaban realizando en un ordenador con mejores prestaciones.

Sin embargo, se vio la posibilidad de dividir cada fotograma en segmentos, de forma que al tratarlos individualmente fuera posible una mejor interpretación del área frontal del robot, desde la zona más lejana hasta la más próxima, permitiendo un gran control sobre el entorno empleando únicamente la vista de pájaro.

Los segmentos, fueron creados con la intención de dividir horizontalmente las líneas en fragmentos de menor tamaño, los cuales representaran áreas concretas del frontal del robot que pudieran ser interpretadas individualmente. Las primeras pruebas, se realizaron con la división de la imagen en 6 fragmentos horizontales, los cuales contenían un segmento de ambas líneas laterales o incluso la frontal. Por lo tanto, de esta forma se lograba conocer en todo momento en que segmentos había paredes laterales o en cual se encontraba la frontal, si se daba el caso, permitiendo frenar o virar a la distancia adecuada.

Para el tratamiento de cada segmento, se volvió a analizar que método de extracción de características era mejor. Pero en este caso, hubo un claro vencedor al descubrir nuevas funcionalidades de la función de detección de contornos “findContours()”. Estas nuevas

funcionalidades permitían una detección casi perfecta segmento a segmento de la línea y no generaban un conjunto de puntos los cuales debían ser interpretados posteriormente, sino que gracias a una funcionalidad, permitía obtener el centro de los contornos.

Al tratarse de dos contornos por segmento, correspondientes a ambas líneas laterales, se optó por una subdivisión vertical en dos partes, dividiendo ambas líneas. Al tratarse de un único fragmento de línea por segmento, el punto central del contorno es la posición del centro de la línea en dicho segmento. Esto permite llevar un control de la línea presente en cada segmento y medir la distancia entre el prototipo y la línea en cada uno de ellos, pudiendo emplear la información de forma individual para el guiado, paliando los efectos de las detecciones erróneas o las discrepancias producidas por la perspectiva. Esto puede verse en la Figura 6.26.

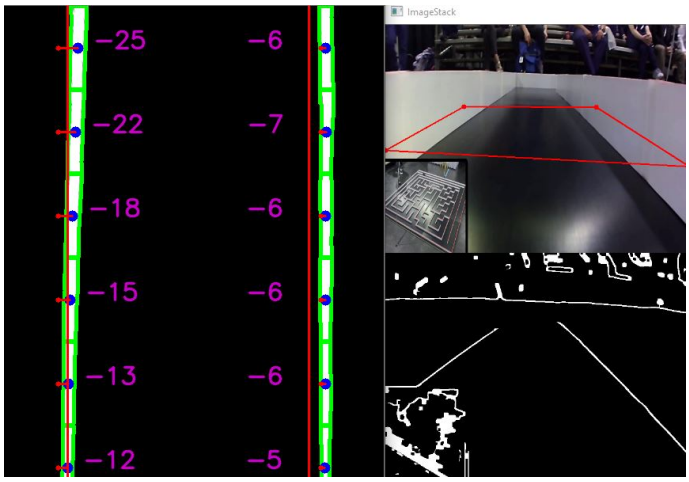


Figura 6.26: Ejemplo del reconocimiento aplicado a un fotograma durante la navegación por un pasillo del laberinto

En dicha figura, se puede apreciar una ligera desviación del robot hacia la pared derecha, este comportamiento se puede apreciar a lo largo de todo el vídeo ya que por algún motivo el robot del ejemplo tenía cierta tendencia a escorarse. Para visualizar la deriva, se ha representado en los 12 segmentos la desviación del robot respecto al centro de los pasillos, indicando en cada segmento el centro de la línea con un punto de color azul y la posición en la que debería encontrarse con un punto de color rojo.

Además, para mayor claridad y facilidad a la hora de realizar las pruebas, se ha añadido un conjunto de etiquetas las cuales indican la desviación existente en cada segmento. Estas, han sido marcadas en color morado y van desde valores positivos en caso de encontrarse escorados a la izquierda a valores negativos en caso de estarlo hacia la derecha.

Como se puede observar, gracias a la información disponible en este punto, es posible guiar perfectamente al robot centrado a lo largo de un pasillo, sin embargo, como se puede ver en la Figura 6.27 las intersecciones suponen un nuevo reto. Y es que debido a la transformación de la perspectiva, se muestran como un desvío lateral, no siempre con el mismo ángulo.

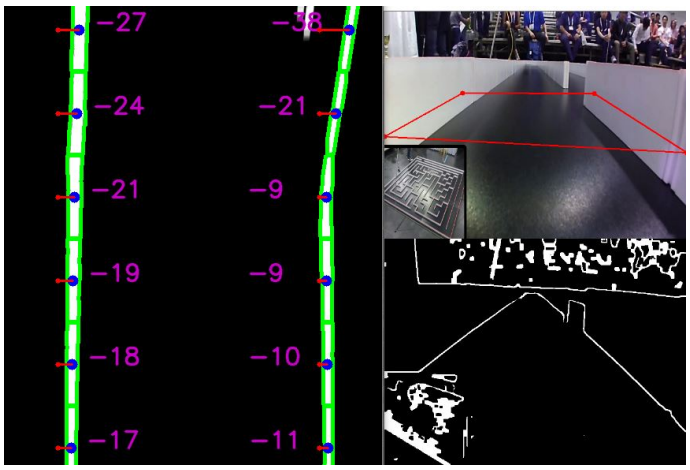


Figura 6.27: Ejemplo de la interpretación del fotograma que contiene el comienzo de una intersección

Gracias a la división en segmentos, es posible detectar su presencia, conociendo su posición relativa al robot, como se puede apreciar en la Figura 6.28. Además en caso de ser necesaria una mayor certeza, puede ser aplicada una segmentación aún mayor de la imagen.

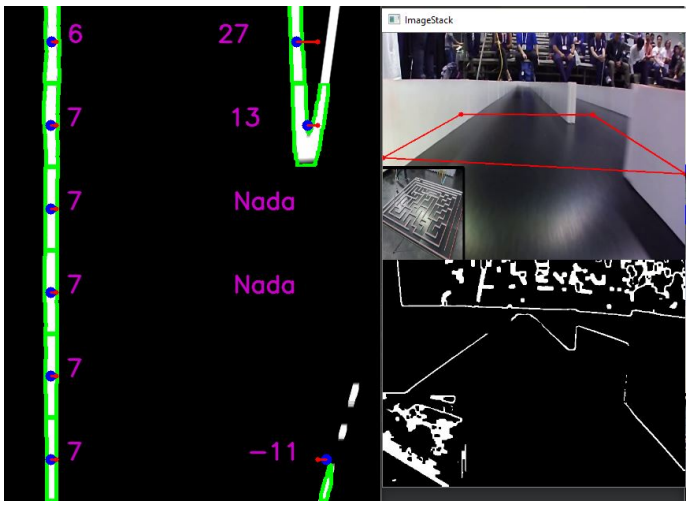


Figura 6.28: Ejemplo de la interpretación del fotograma que contiene una intersección

Pero esto no es todo, como he mencionado anteriormente, también es necesaria la detección de las paredes frontales. Como puede observarse en la Figura 6.29, la pared frontal aparece marcada en rojo en dos de los segmentos. Esta detección, es posible realizarla gracias a una funcionalidad relacionada con la función “findContours()”, la cual permite reconocer la orientación de los contornos mediante el uso de kernels rectangulares de tamaño variable.

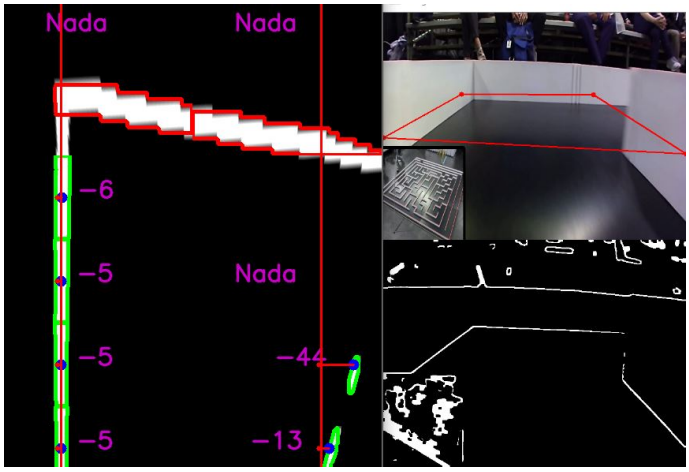


Figura 6.29: Ejemplo de la interpretación de un fotograma que contiene una pared frontal y una intersección

En la siguiente imagen (Figura 6.30), se puede apreciar el momento en el cual dicha pared frontal se encuentra muy cerca del robot y no se detectan más líneas. Por todo ello, se puede llegar a afirmar que es posible orientar al robot por el laberinto con un grado de fiabilidad elevado, siempre y cuando se logren unos mínimos en el procesado previo a este punto.

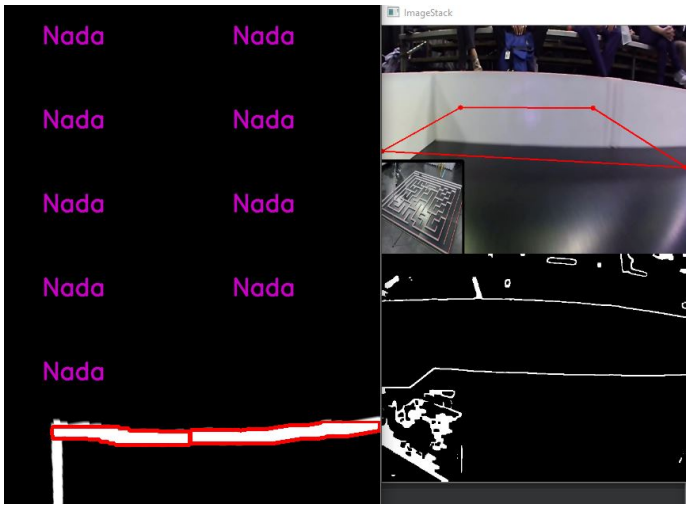


Figura 6.30: Ejemplo de la interpretación de un fotograma que contiene una pared frontal y una intersección

Incremento de la fiabilidad

Cuando se trata de obtener un grado de fiabilidad elevado, suelen ser empleados sistemas redundantes. En este caso, esta estrategia puede aplicarse en todos los pasos o incluso en conjuntos de ellos, afrontando diferentes estrategias que confluyan en un resultado satisfactorio para el uso que se le va a dar.

Para no alargar excesivamente este apartado, se van a mostrar un par de ejemplos para aclarar este concepto. El primero de ellos es la adecuación de las líneas, un proceso que abarca diferentes pasos, desde la adecuación de la imagen hasta la extracción de las características. Tal y como se ha ido mencionando anteriormente, existe un gran abanico de posibilidades en cada uno de los pasos, cada uno con sus pros y sus contras y estos influyen enormemente en los resultados de los pasos posteriores.

Por lo tanto, a la hora de elegir cuales son las estrategias más efectivas, surgen una cantidad cada vez mayor de combinaciones, las cuales son imposibles de abarcar experimentalmente sin extenderse en el tiempo. Por ello, es común generar estrategias que no tienen buenas prestaciones en todos los casos.

Una estrategia capaz de aumentar las probabilidades de una correcta detección de todas las líneas es el empleo de operaciones aritméticas (ver Sección 5.3.3) sobre dos o más imágenes finales resultado de distintos procesos de tratamiento, de forma que se obtengan las características positivas de las diferentes tácticas empleadas y se disminuyan los efectos secundarios asociadas a algunas de ellas.

Por puntualizar, para este prototipo, se han empleado dos tipos de filtrado distintos, seguidos de dos estrategias de detección y mejora de bordes, una cuyo resultado generaba líneas precisas pero pecaba de un déficit de detección de las líneas poco claras y una que detectaba todas las líneas, pero también las líneas fantasma provenientes de reflejos o ruidos. Al operar sobre las imágenes finales, se ha obtenido un mayor éxito en la detección de líneas y se han eliminado o reducido en gran medida las falsas detecciones.

El segundo ejemplo aplicado en el proyecto, es la redundancia en la navegación, empleando las dos perspectivas disponibles, es decir, la original y la vista de pájaro, cuya interpretación conjunta ayuda enormemente en los fotogramas complejos, sobre todo en las intersecciones y los cambios de dirección. De esta forma, es posible contrastar los resultados o al menos seguir teniendo un método válido en caso de fallar uno de ellos.

6.5. Módulo de navegación y mapeo

Para la realización de este módulo, es necesario realizar un planteamiento de cuales son los problemas básicos de cualquier sistema de navegación (ver Sección 4) pero aplicados en el entorno del laberinto.

El Micromouse, debe ser capaz de situarse en todo momento en una celda concreta y dentro de dicha celda, debe conocer el estado de las paredes que le rodean como por ejemplo que caminos viables hay desde dicha celda y cuales han sido ya recorridos. A su vez, partiendo de dicha información, debe ser capaz de generar rutas viables hacia el destino y una vez este sea alcanzado, debe interpretar cuál de las rutas es la más rápida, para un segundo trayecto a gran velocidad. Este paso, según las reglas de la competición puede realizarse en el camino de retorno a la casilla de inicio, proporcionando un reconocimiento extra del laberinto.

Para la realización del software de navegación se ha optado por la simulación, para la cual se ha implementado un código de generación de laberintos sobre el cual realizar las pruebas pertinentes.

También, se han plasmado algunos laberintos reales de competición para proporcionar un campo de pruebas más realista.

6.5.1. Laberinto

La representación del laberinto, se ha realizado empleando únicamente la librería OpenCV y se ha desestimado la idea de emplear software adicional.

Para ello se ha hecho uso de la función “cv2.line”, la cual traza una línea del color y grosor deseados entre dos puntos. Estas líneas han sido representadas sobre una matriz de píxeles de color negro, la cual ha sido tratada como una imagen. (ver Figura 6.31)

Esta representación, sin embargo, ha requerido la implementación de código de control sobre las celdas, ya que es necesario tener en cuenta varios aspectos como la pertenencia de una pared a dos celdas o el color empleado para la representación de las mismas, gestionando cada operación de eliminación o trazado de línea.

Otro aspecto, es la señalización de las etiquetas, las cuales se mencionan en el siguiente apartado y hacen referencia a los pasos que es necesario realizar desde cada celda hasta el centro del laberinto.

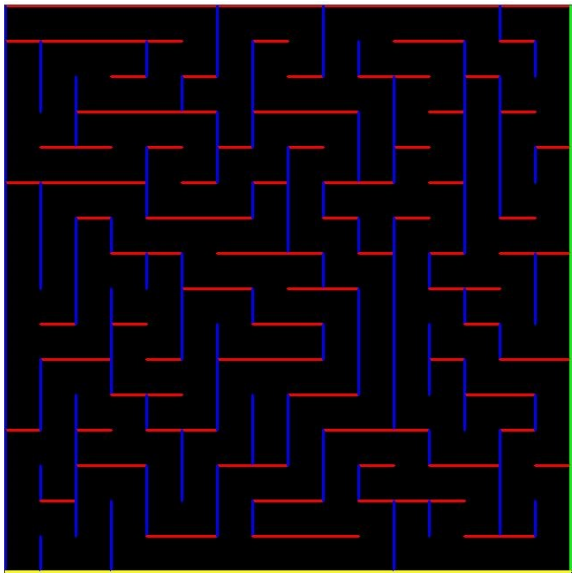


Figura 6.31: Representación de un laberinto empleando la librería OpenCV

Esto se ha realizado calculando la posición relativa del centro de cada celda y empleando una función de la librería que permite escribir texto. Esta funcionalidad al igual que la anterior ha sido necesario implementarla generando funciones que gestionen como se borra o añade información en cada celda.

También, ha sido necesario realizar el mismo proceso con la simulación de navegación, ya que se trataba de mostrar de forma visual como se van descubriendo las celdas según va avanzando el robot por el laberinto. Esto se ha logrado mostrando las paredes de color tenue o vivo para diferenciarlas, junto con unas cruces de colores que indican el recorrido realizado por el robot a través del laberinto.

6.5.2. Algoritmo de búsqueda

La elección de un algoritmo de búsqueda para un sistema de navegación autónomo es una tarea crítica que hay que analizar con detenimiento.

Algunos de los algoritmos más comunes para un Micromouse suelen ser A*, DFS, BFS, Dijkstra y Floodfill. Este último, es muy interesante ya que la forma en la que se desarrolla concuerda enormemente con la navegación del robot-ratón.

El algoritmo Floodfill [Wikipedia.org, 2021a], de inundación, o algoritmo de relleno por difusión, es muy común en los programas de dibujo y se emplea, por ejemplo, para rellenar áreas de un determinado color, ya que su característica principal es extenderse de un punto a otro continuamente simulando una especie de inundación.

La competición tiene como objetivo fundamental, llegar a una de las cuatro casillas de destino situadas en el centro del laberinto desde una casilla de inicio situada en uno de los extremos. Por ello, una versión de este algoritmo, es la que se suele emplear en los laberintos para hallar la mejor ruta hasta el centro, simulando una inundación desde el centro del mismo, la cual se va extendiendo hasta la casilla de inicio.

Una vez alcanza dicha casilla, cada celda contiene el número de pasos necesarios que se deben recorrer desde dicha celda para alcanzar el centro del laberinto, por lo tanto, si se realiza un recorrido avanzando en cada celda hacia la contigua con menor valor se llega al centro por el camino más corto.

En la siguiente Figura 6.32, se puede observar el laberinto de la competición de Micromouse organizada por la APEC del año 2017 representado sin etiquetas (izq) y con las respectivas etiquetas tras aplicar el algoritmo floodfill desde el centro del laberinto ignorando las paredes (dcha).

En la Figura 6.33, por el contrario, se ha aplicado el algoritmo Floodfill desde el centro del laberinto teniendo en cuenta las paredes, por lo tanto, en cada celda la etiqueta indica el número de casillas que es necesario recorrer para alcanzar el centro a través de los caminos disponibles. Como se puede apreciar en las etiquetas representadas, las rutas han sido extraídas correctamente mediante el algoritmo Floodfill y bastaría con navegar según los valores de las mismas.

Esto solo se daría en caso de conocer previamente el laberinto, pero no es el caso, ya que el robot debe ir descubriéndolo durante la fase de búsqueda navegando por su interior. Por lo tanto, el proceso es el siguiente.

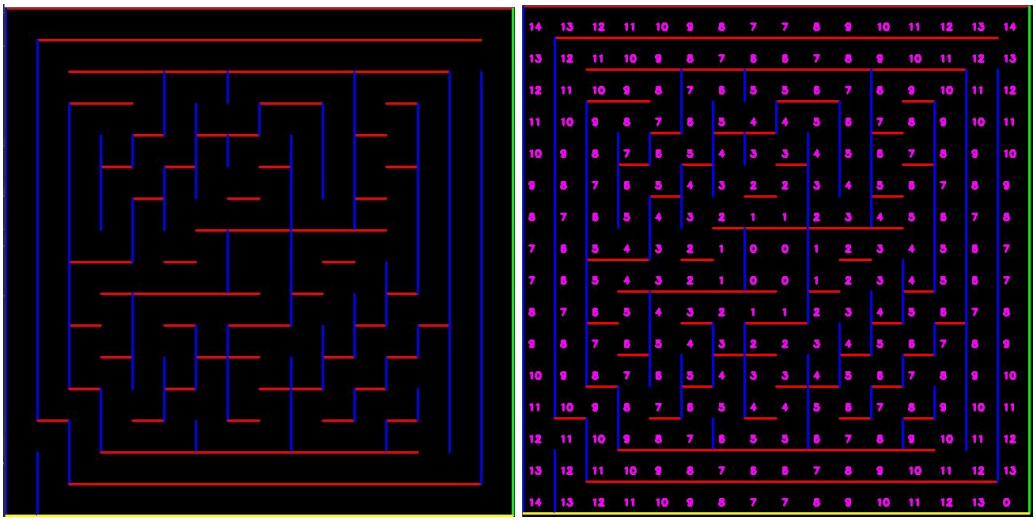
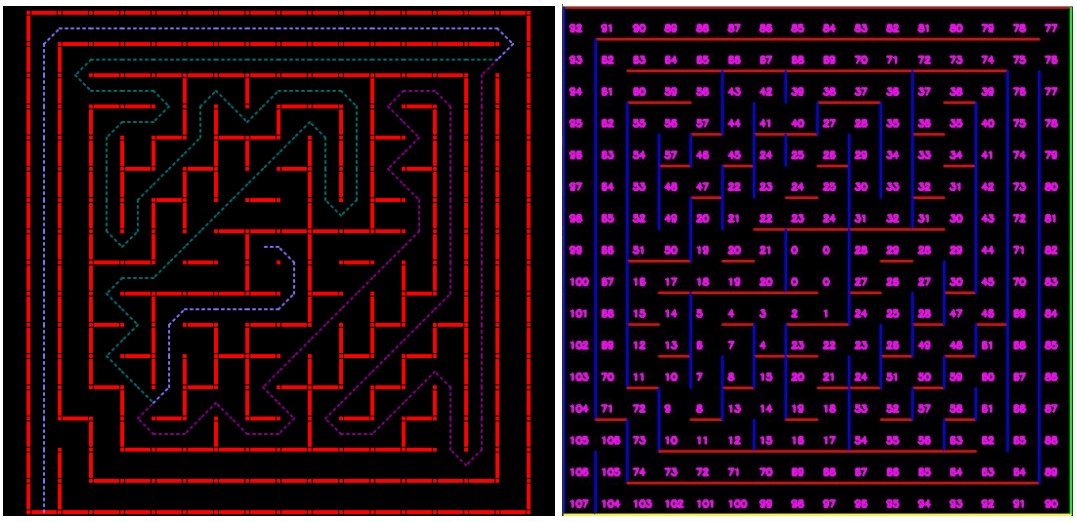


Figura 6.32: Aplicación del algoritmo Floodfill ignorando paredes



((a)) Rutas del laberinto

((b)) Aplicación del algoritmo Floodfill condicionado por paredes

Figura 6.33: Detección de las rutas mediante el algoritmo Floodfill (paredes conocidas)

Se parte de un laberinto sin paredes conocidas, excepto las de la celda en la que se encuentra, por lo que el algoritmo debe ser ejecutado cada vez que se avance a una nueva celda para recalcular la ruta y la distancia por celda respecto al centro. (ver Figura 6.34)

En cada iteración, el algoritmo trata de orientar al robot hacia el centro con la intención de llevarlo por el camino más corto, pero al desconocer las paredes que se interpondrán en dicho camino deberá recalcular la ruta según perciba la presencia de nuevas paredes.

Tras la elección del camino a seguir y al alcanzar cada celda, el robot detecta las paredes circundantes e interpreta su posición en el laberinto, siendo necesaria una replanificación de la ruta.

Cuando el robot llega a un callejón sin salida, o ha de deshacer un camino ya recorrido, marca dichas casillas como celdas no viables y prosigue hasta alcanzar el destino. Esto se puede observar en la Figura 6.35, en la que hay celdas que han sido desestimadas ya que formaban parte de un camino no viable y ha sido necesario retroceder.

Una vez alcanzado el destino, según la competición, se pueden adoptar dos estrategias: detener el robot y extraerlo manualmente o continuar con el camino inverso tratando de obtener una ruta más rápida.

Para este proyecto, se ha optado por la segunda opción, ya que puede proporcionar una información valiosa. Por lo tanto, una vez alcanzado el centro, se realiza el proceso inverso, es decir, se “inunda” desde la casilla de inicio y de esta forma se recorren partes no conocidas del mapa en busca de otra ruta más rápida.

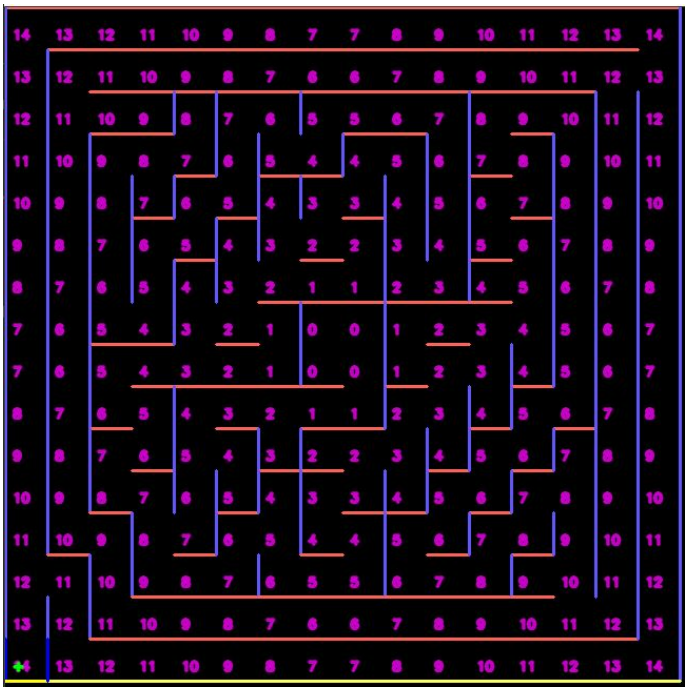


Figura 6.34: Inicio de la navegación empleando el algoritmo Floodfill (Las paredes en color tenue son las no descubiertas por el robot hasta el momento)

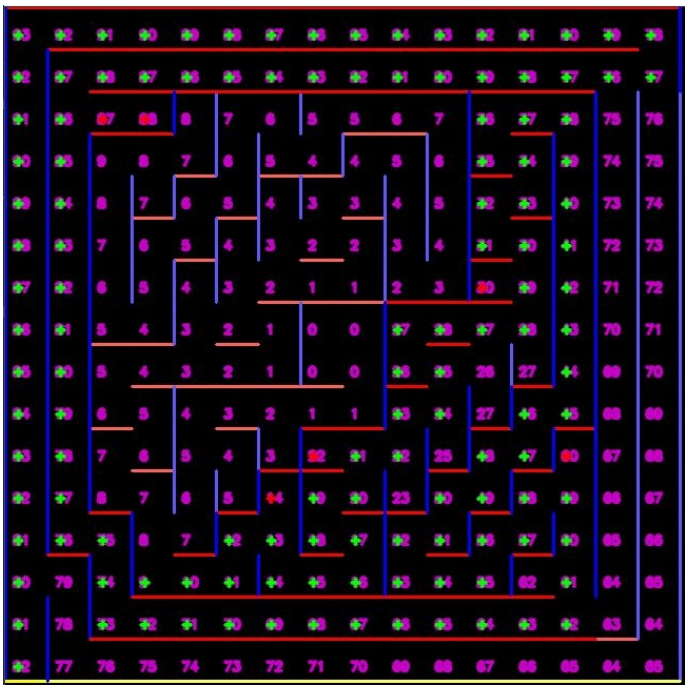


Figura 6.35: Progreso de la navegación (Las celdas marcadas en verde indican el recorrido y las marcadas en rojo han sido desestimadas)

7. CAPÍTULO

Gestión y planificación del Trabajo de Fin de Grado

Para la realización de cualquier proyecto, son necesarias una gestión y planificación bien planteadas de cara a controlar el progreso del mismo y prevenir posibles riesgos. En un proyecto de estas características, la necesidad de un buen planteamiento de estas cuestiones se hace aún más evidente, influyendo enormemente en el desarrollo del mismo en el plazo disponible.

En consecuencia, es necesario dividir el trabajo en distintas fases independientes. Para este proyecto se han identificado las siguientes: análisis de la viabilidad, planificación, desarrollo, seguimiento y control y documentación. Dentro de cada fase, se han definido paquetes de trabajo como unidades para la planificación del trabajo los cuales se describen posteriormente.

7.1. Diagrama EDT

La Estructura de Descomposición del Trabajo (E.D.T) es importante ya que permite mostrar de forma clara la subdivisión del proyecto en paquetes de trabajo de menor tamaño y por lo tanto, más manejables. (ver Figura 7.1)

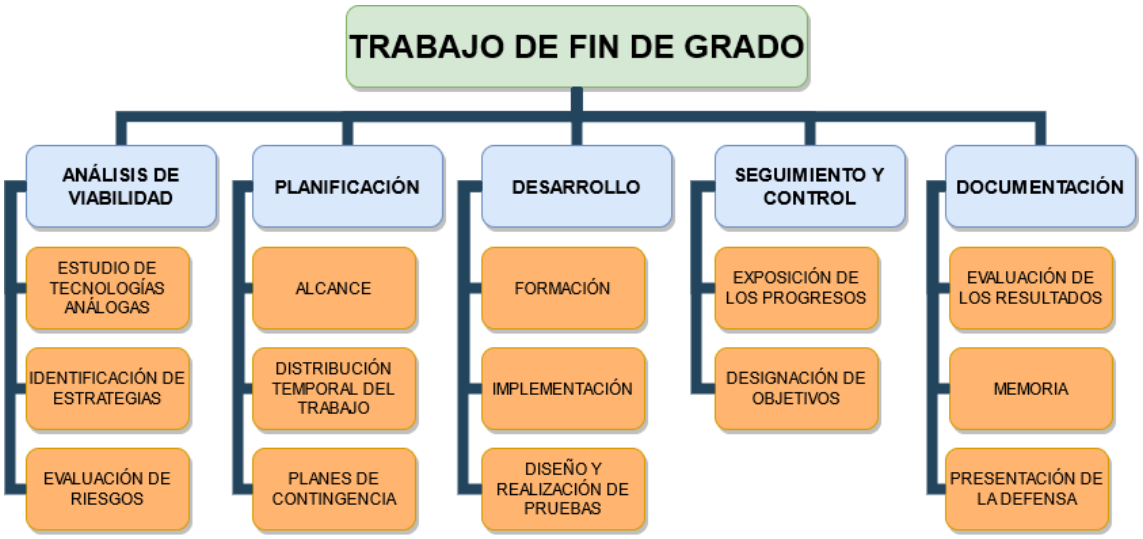


Figura 7.1: Diagrama EDT

7.2. Descripción de las fases que componen el EDT

A continuación, se procede a describir las tareas contenidas en los paquetes de trabajo que componen el proyecto.

7.2.1. Análisis de la viabilidad

Esta fase está formada por los paquetes de trabajo relativos al estudio de la viabilidad de la realización de este proyecto. Este proceso es crítico, debido al enfoque novedoso que presenta, carente de ejemplos previos para esta aplicación. Esta falta de antecedentes tecnológicos puede deberse a diferentes motivos, pero entre ellos destacan la percepción por parte de terceros de una posible dificultad excesiva o la interpretación de que esta tecnología no es la adecuada para esta aplicación en concreto. Por este motivo, es de vital importancia identificar en las etapas iniciales del trabajo las posibilidades y los riesgos de cara a afrontar el trabajo en el tiempo disponible y de esta forma responder a las cuestiones planteadas. Para la realización de esta fase se han identificado las siguientes tareas:

Estudio de tecnologías análogas

Para estudiar la viabilidad de un proyecto, una de las aproximaciones más exitosas es la importación y adaptación de tecnologías empleadas en proyectos similares, las cuales

hayan proporcionado buenos resultados. Por lo tanto, con esta tarea se trata de obtener una base sobre la cual trabajar, basada en la experiencia previa.

Identificación de estrategias

Consiste en el análisis de las características que definen al proyecto y en el diseño de tácticas que permitan lograr los objetivos que se están planteando, empleando los recursos de los que se dispone de cara a obtener conclusiones sobre la viabilidad del mismo. Para este proceso es importante tener conocimientos acerca de las posibilidades tecnológicas existentes de cara a la generación de estrategias efectivas, por lo que está íntimamente relacionado con la tarea anterior.

Evaluación de riesgos

Un aspecto fundamental previo a la inmersión en la realización de cualquier proyecto es el análisis de riesgos. Este debe estudiar la posibilidad de que no se puedan cumplir en tiempo o forma las tareas identificadas y debe ser producto de una evaluación objetiva y realista.

7.2.2. Planificación

Esta fase se concentra en el primer mes del desarrollo del proyecto y trata de programar la distribución de la carga de trabajo de forma uniforme durante todo el proyecto. Para ello, se debe identificar el alcance del proyecto, y posteriormente realizar una división en tareas del mismo, todo ello junto a una serie de planes de contingencia que permitan afrontar de forma satisfactoria los imprevistos que vayan surgiendo.

Alcance

Consiste en la definición de los objetivos que persigue el proyecto y su coste temporal, para la realización de una planificación acorde a dichos objetivos.

Distribución temporal del trabajo

Esta tarea trata de programar equitativamente la distribución de la carga de trabajo a lo largo del tiempo del proyecto.

Planes de contingencia

Al tratarse de un proyecto de larga duración, es muy probable la aparición de contratiempos. Para evitar los efectos que estos puedan producir sobre el desarrollo del proyecto, es necesaria la creación de estrategias que permitan afrontarlos minimizando su impacto. Para ello, es necesario realizar un estudio de cuales son los imprevistos con mayor probabilidad de ocurrir y generar tácticas que permitan reconducirlos.

7.2.3. Desarrollo

Esta fase comprende todo el proceso de desarrollo del proyecto, compuesto por una parte formativa relativa a las tecnologías y herramientas que serán utilizadas y una parte centrada en la implementación de las diferentes partes que componen el prototipo, el cual es el núcleo del proyecto.

Formación

Esta tarea tiene como objetivo el aprendizaje de las tecnologías y herramientas que se utilizarán y el entorno de trabajo donde se pondrán en marcha. Esta tarea puede subdividirse en el estudio de las tecnologías y el estudio de las herramientas y es la continuación, pero con mayor profundidad de las tareas previas referentes al estudio de tecnologías análogas e identificación de estrategias.

El estudio de las tecnologías se puede dividir en:

- Estudio de las técnicas de visión por computador: Analizar las diferentes técnicas empleadas en visión por computador y observar la implementación empleada en aplicaciones similares como los coches sin conductor.
- Análisis del entorno del proyecto (laberinto) para la identificación de patrones: Observar las características propias de los laberintos empleados en las competiciones

de Micromouse para obtener patrones o referencias repetitivas que caractericen el trazado.

- Estudio de la viabilidad del guiado sin sensorización: Observar la sensorización empleada para el guiado de este tipo de sistemas y analizar si es posible sustituirla por un sistema basado en la interpretación de las imágenes obtenidas por la cámara integrada en el dispositivo o si es necesaria la realización de un sistema híbrido.
- Estudio de las técnicas y herramientas empleadas en proyectos similares: Analizar las diferentes estrategias empleadas en proyectos exitosos similares (ej. técnicas empleadas en coches autónomos y UAVs) y estudiar la posibilidad de adaptar dichos procedimientos al proyecto.

El estudio de las herramientas, está formado por la búsqueda de información acerca del software y del hardware del que se va a hacer uso

- Estudio de la biblioteca OpenCV: Estudiar las diferentes funciones disponibles, sus posibilidades y sus usos habituales.
- Estudio de la placa con el chip ARM: Estudio de las características que ofrece el hardware sobre el que se va a trabajar y sus limitaciones.

Implementación

Esta tarea agrupa los pasos necesarios para la implementación del software del Micromouse. Está estructurada en tres paquetes de trabajo referentes a las funcionalidades principales del robot.

- Desarrollo de la visión artificial: Paquete de trabajo en el que se incluye el filtrado, preprocesado, análisis e interpretación de la información del entorno percibida a través de la cámara situada en el dispositivo.
- Desarrollo del módulo de gestión motriz: Trata del desarrollo del módulo encargado de transmitir las ordenes de movimiento a las ruedas para el desplazamiento del robot por el laberinto.
- Desarrollo del módulo de control: Paquete de trabajo relativo al diseño del módulo encargado del procesamiento de la información obtenida por el módulo de visión,

su interpretación y generación de un mapa además del guiado efectivo del robot a través del laberinto transmitiendo las ordenes de desplazamiento al módulo de control motriz.

Diseño y realización de pruebas

En esta tarea se diseñan y realizan las pruebas para evaluar el correcto desempeño del Micromouse en su misión de recorrer el laberinto, analizando a su vez el correcto funcionamiento de todos los módulos que lo forman. Debido al límite temporal del proyecto, algunas de estas tareas dependientes de la disponibilidad del laberinto serán opcionales. Esta tarea puede dividirse en dos partes:

1. Diseño de las pruebas:
   - Diseño de las pruebas de interpretación visual:
     - Pruebas empleando vídeo: Diseñar las pruebas a realizar empleando grabaciones del recorrido en primera persona de otro robot por un laberinto de competición.
     - Pruebas en entorno real (Opcional): Diseñar las condiciones necesarias para poner a prueba las capacidades del robot para desenvolverse en un entorno real (Iluminación, características físicas del laberinto como color y rugosidad, juntas de unión de las paredes...).
   - Diseño de las pruebas de movimiento: Diseñar las pruebas a realizar para comprobar las funciones motrices del robot.
   - Diseño de las pruebas de navegación (Opcional): Diseñar las pruebas a realizar para comprobar la navegación del robot por segmentos de prueba del laberinto.
   - Diseño del laberinto completo (Opcional): Diseñar la estructura del laberinto para la realización de las pruebas. (Tamaño, forma, complejidad...)
   - Diseño de las pruebas en el laberinto (Opcional): Diseñar las pruebas a realizar para analizar el desempeño general del robot en el laberinto creado.
2. Realización de las pruebas:
   - Documentación de los resultados: Se valorará el desempeño del robot en las diferentes pruebas valorando positiva o negativamente el resultado en cada caso. En el supuesto de que el resultado sea ambiguo o negativo se valorarán los

momentos temporales concretos y las condiciones en los que se haya obtenido el resultado deseado y se anotarán, para una posterior revisión y análisis de los mismos.

- Análisis de los resultados: Se analizarán los casos en los que se ha obtenido el desempeño esperado y se compararán con los casos en condiciones similares en los que este no ha sido el deseado, para la búsqueda de patrones entre ellos. Cuando el desempeño sea el esperado en repetidas ocasiones, se dará por finalizada satisfactoriamente la prueba. Cuando las pruebas asociadas a una tarea sean satisfactorias, se dará la tarea por concluida. Cuando un resultado sea negativo, se analizará junto con el resto de resultados similares para tratar de localizar el origen del problema y trazar un plan para solventarlo.

7.2.4. Seguimiento y Control

El seguimiento y control se desarrolla durante todo el ciclo de vida del proyecto y controla el correcto desarrollo del mismo. Dentro de este paquete se han definido dos tareas, la exposición de los progresos y la designación de objetivos.

Exposición de los progresos

Trata de la exposición al tutor de forma semanal de los logros obtenidos, de los problemas encontrados y del estado general del proyecto, así como el planteamiento y resolución de las dudas.

Designación de objetivos

La designación de objetivos, hace referencia a la replanificación a corto plazo de los objetivos, tras la exposición de los progresos. Esta tarea se realiza junto al tutor y permite un desarrollo ágil y ajustado temporalmente a los hitos obtenidos.

7.2.5. Documentación

Esta fase agrupa las tareas necesarias para la realización de la documentación del trabajo y los resultados y la presentación de la defensa.

Evaluación de los resultados

En esta tarea se valoran los resultados obtenidos tras la realización del proyecto. Dicha valoración es fruto de las diferentes pruebas realizadas al prototipo y a los módulos que lo forman. Esta valoración es de gran importancia ya que permite señalar que partes del trabajo han destacado y que partes del mismo han obtenidos resultados pobres aportando una visión global de cara a posibles mejoras fuera de este trabajo.

Memoria

Para la realización de la memoria del proyecto se empleará el sistema de composición de textos *LaTex*, en concreto, la herramienta *Overleaf*. La información correspondiente a cada apartado, será redactada en un fichero diferente. Dichos ficheros serán agrupados de forma ordenada al final del proyecto para completar la memoria. El documento final será producto de una redacción formal de dicha memoria.

Presentación de la defensa

Esta tarea comprende los trabajos necesarios para la presentación de la defensa, es decir, la realización de las diapositivas y la preparación de la exposición.

7.3. Estimación y desviación temporal de las tareas

En la Tabla 7.1 se muestra agrupado por paquetes el coste temporal estimado de cada una de las tareas y el tiempo real invertido en la realización de cada tarea. También se indica el cómputo total de horas invertidas que fue estimado al comienzo del proyecto y el tiempo total empleado para la finalización del proyecto.

Como se puede apreciar ha habido una desviación temporal entre el tiempo estimado y el tiempo real empleado, esto se ha debido a la naturaleza del prototipo, ya que los robot Micromouse son proyectos a largo plazo caracterizados por mejoras constantes orientadas a la obtención de un robot cada vez más capaz y competitivo. Con esta idea, se ha continuado con el desarrollo del mismo una vez superados los objetivos marcados, los cuales definían unos mínimos. Por otro lado, el tiempo dedicado a la redacción de la memoria si que ha excedido los plazos plateados, debido a la dificultad que ha supuesto intentar plasmar brevemente y de forma comprensible la amalgama de pruebas realizada.

	Estimación (h)	Real (h)
Análisis de viabilidad	20	23
Estudio de tecnologías análogas	12	14
Identificación de estrategias	5	6
Evaluación de riesgos	3	3
Planificación	25	21
Alcance	15	14
Distribución temporal del trabajo	5	3
Planes de contingencia	5	4
Desarrollo	145	196
Formación	50	80
Implementación	75	91
Diseño y realización de pruebas	20	25
Seguimiento y control	10	9
Exposición de los progresos	7	7
Designación de objetivos	3	2
Documentación	95	120
Evaluación de los resultados	10	5
Memoria	55	90
Presentación de la defensa	30	25
TFG	295	369

Tabla 7.1: Tabla de desviaciones.

7.4. Metodología de trabajo

Para el desarrollo de este proyecto se ha adoptado por un ciclo de vida incremental, produciendo distintas versiones finales de los módulos que tras cada iteración, van sucesivamente recibiendo funcionalidades. En dichos ciclos,se va progresivamente construyendo el resultado del proyecto, según se va descubriendo más sobre el mismo.

Se ha seleccionado esta metodología dadas las características del proyecto, ya que se trata de un proyecto con un fondo muy complejo y que depende de muchos factores. En concreto, el módulo de visión artificial, idealmente debe de ser tratado de forma iterativa, probando de esta forma, diferentes estrategias de cara a obtener un sistema fiable de detección del trazado y esto puede ser logrado mediante varias aproximaciones que deben ser probadas de forma experimental.

Por lo tanto, con esta metodología se podrán probar y valorar las distintas versiones generadas para tratar de encontrar un método óptimo para recorrer el laberinto o al menos bastante fiable.

8. CAPÍTULO

Conclusiones y trabajo futuro

Para finalizar este documento se expondrán las conclusiones más relevantes que han sido extraídas de la realización del proyecto y se tratarán las posibles líneas de trabajo futuras.

En esta memoria, se ha tratado plasmar todo el trabajo realizado en estos meses de desarrollo del proyecto, describiendo las partes más relevantes de todo el proceso de creación del prototipo junto con los problemas encontrados y las soluciones que se han empleado.

La naturaleza de este trabajo ha sido una prueba de concepto, una demostración de que es posible la creación de sistemas robóticos que prescindan de sensorización y que empleen en su lugar técnicas simples de visión artificial. Este ha sido un aspecto clave en el planteamiento del trabajo, la intención era alejarse de técnicas complejas o computacionalmente costosas basadas en inteligencia artificial mostrando que es viable el empleo de funciones sencillas empleadas con creatividad. Por ello, esta ha sido la parte más extensa y en la que más se ha profundizado.

Este planteamiento, cada vez es más viable gracias a la enorme variedad de opciones que proporcionan las librerías de visión artificial, como es el caso de OpenCV la cual ha servido de base. Por lo tanto, lo que se quiere transmitir, es que es necesario un cambio de mentalidad, una pérdida del temor que generan este tipo de tecnologías y un acercamiento a los desarrolladores, los cuales deben plantearse su uso, incluso en aplicaciones simples ya que gradualmente la dificultad que las caracteriza irá situándose a la par de las dificultades inherentes al uso de sensores.

Un aspecto que he observado, ha sido la tendencia cada vez mayor de crear y emplear funciones con un mayor grado de abstracción o por así decirlo de un nivel más alto, que estén

formadas por un conjunto de funciones. Es decir, implementaciones que tienen como entrada una imagen en crudo y como salida proporcionan una serie de puntos, coordenadas o líneas perfectamente definidos.

Esto facilitará a futuros desarrolladores la realización de proyectos complejos sin la necesidad de profundizar en ciertos conceptos, permitiendo la popularización de estas tecnologías al poder ser empleadas por gente menos instruida en este ámbito.

Respecto al trabajo realizado, el mayor reto ha sido la comprensión del abanico de posibilidades existentes en el ámbito de la visión artificial y la elección meditada de un conjunto de estrategias. Para ello se han realizado un gran número de pruebas empleando una gran cantidad de combinaciones y se ha buscado gran cantidad de información, ya que es necesario tener un conjunto de nociones básicas para comprender la dimensión del problema.

Con este fin, se han estudiado muchos proyectos, tanto relacionados con la visión artificial como con la robótica y la navegación autónoma o con la competición de Micromouse tratando de partir de la experiencia previa de otros proyectos. Esta base ha permitido las primeras iteraciones del desarrollo y pruebas, pero estaban caracterizadas por el desconocimiento acerca de la motivación del empleo de ciertos pasos ya que no se tenía una visión de conjunto.

Sin embargo, hubo un punto de inflexión tras el cual se comenzó a entender el porqué de las cosas y permitió profundizar y mejorar en su uso. Esto fue debido a que la mayoría de las funciones y de las técnicas están íntimamente relacionadas y algunos ejemplos han ayudado al entendimiento de conceptos similares.

Otro aspecto que resultó determinante en el desarrollo del proyecto, fue la no disposición del prototipo y del circuito hasta el final del tiempo de desarrollo, lo que derivó en la simulación de este último. Por un lado, se simuló el laberinto, de cara a la creación del software relativo a la navegación, aunque es necesario comentar que debido a la naturaleza del proyecto se hubiera procedido igualmente a la simulación para la detección precoz de errores. Por otro lado, fue necesario el empleo de un campo de pruebas para el módulo de visión artificial, un pseudolaberinto sobre el que realizar pruebas, y para ello se emplearon los vídeos, los cuales supusieron una dificultad extra que sin embargo fue positiva de cara al aprendizaje.

Este proyecto, debido a su duración, me ha servido para obtener experiencia de cara a afrontar proyectos a largo plazo, ya que se aleja del tipo de proyectos que se han ido realizando durante la carrera. También me ha servido para valorar más aún una buena

planificación y más en un tipo de proyecto con tantas opciones disponibles y con tan poca experiencia previa en las tecnologías empleadas. Además, me ha servido de vía de iniciación en el mundo de la visión artificial, el cual es un campo que ha despertado mi interés y en el que profundizaré en el futuro.

Por último, la creación de un robot Micromouse es una tarea a largo plazo que requiere sucesivas mejoras a lo largo del tiempo para obtener un prototipo competitivo. Esto, se ha visto limitado al tiempo asignado para la realización del trabajo, pero ha sido suficiente para sentar las bases. Sin embargo, me hubiera gustado disponer de más tiempo para las pruebas con el prototipo en el laberinto.

De cara a un futuro próximo, sería muy interesante progresar con el proyecto y obtener un prototipo capaz de competir, ya que es un excelente método de aprendizaje y permite la experimentación de tecnologías que pueden aplicarse en otros ámbitos de forma sencilla, debido a la regularidad del laberinto. Además, existen dos líneas de trabajo para la mejora del prototipo. Al igual que en un Micromouse convencional, se pueden mejorar tanto las prestaciones físicas del robot como la programación relativa a la navegación, pero en este caso, también es posible obtener grandes avances en el módulo de visión artificial.

En cuanto a las líneas de trabajo futuras, sería muy interesante añadir y experimentar con otros recursos relacionados con técnicas avanzadas de visión y la inteligencia artificial los cuales incrementarían las prestaciones enormemente, e implementarlos en el robot.

Bibliografía

[AA.VV., 2016] AA.VV. (2016). *Conceptos y métodos en visión por computador*. Grupo de Visión del Comité Español de Automática.

[Azkune, 2011] Azkune, G. (2011). Navegación autónoma. https://cuentos-cuanticos.com/2011/11/12/navegacion-autonoma/.

[Geeksforgeeks.org, 2019] Geeksforgeeks.org (2019). Find and draw contours using opencv. https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python.

[Hurtado, 2021] Hurtado, S. (2021). Repositorio. https://github.com/ht4/CV-Micromouse.

[Micromouseonline.com, 2017a] Micromouseonline.com (2017a). Micromouse history. http://www.micromouseonline.com/micromouse-book/history/.

[Micromouseonline.com, 2017b] Micromouseonline.com (2017b). Micromouse rules. http://www.micromouseonline.com/micromouse-book/rules/.

[Opencv.org, 2015] Opencv.org (2015). Opencv camera calibration. https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html.

[Opencv.org, 2017a] Opencv.org (2017a). Canny edge detector. https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.html.

[Opencv.org, 2017b] Opencv.org (2017b). Drawing functions. https://docs.opencv.org/master/d6/d6e/group__imgproc__draw.html#ga7078a9fae8c7e7d13d24dac2520ae4a2.

[Opencv.org, 2017c] Opencv.org (2017c). Eroding and dilating. https://docs.opencv.org/3.4/db/df6/tutorial_erosion_dilatation.html.

[Opencv.org, 2017d] Opencv.org (2017d). Feature detection. https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html.

[Opencv.org, 2017e] Opencv.org (2017e). Feature detection (houghlinesp). https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga8618180a5948286384e3b7ca02f6feeb.

[Opencv.org, 2017f] Opencv.org (2017f). Hough line transform. https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.

[Opencv.org, 2017g] Opencv.org (2017g). Image filtering. https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html.

[Opencv.org, 2017h] Opencv.org (2017h). Morphological transformations. https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html.

[Opencv.org, 2017i] Opencv.org (2017i). Opencv contours. https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html.

[Opencv.org, 2017j] Opencv.org (2017j). Shi-tomasi corner detector & good features to track. https://docs.opencv.org/3.4/d4/d8c/tutorial_py_shi_tomasi.html.

[Opencv.org, 2017k] Opencv.org (2017k). Smoothing images. https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html.

[Opencv.org, 2017l] Opencv.org (2017l). Sobel derivatives. https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html.

[Opencv.org, 2021a] Opencv.org (2021a). Opencv contours hierarchy. https://docs.opencv.org/master/d9/d8b/tutorial_py_contours_hierarchy.html.

[Opencv.org, 2021b] Opencv.org (2021b). Opencv geometric transformation. https://docs.opencv.org/master/da/d54/group__imgproc__transform.html.

[Opencv.org, 2021c] Opencv.org (2021c). Opencv modules. https://docs.opencv.org/master/.

[Opencv.org, 2021d] Opencv.org (2021d). Structural analysis and shape descriptors. https://docs.opencv.org/master/d3/dc0/group__imgproc__shape.html#gadf1ad6a0b82947fa1fe3c3d497f260e0.

[Wikipedia.org, 2020] Wikipedia.org (2020). Transformada de hough. https://es.wikipedia.org/wiki/Transformada_de_Hough.

[Wikipedia.org, 2021a] Wikipedia.org (2021a). Flood fill. https://en.wikipedia.org/wiki/Flood_fill.

[Wikipedia.org, 2021b] Wikipedia.org (2021b). Micromouse. https://en.wikipedia.org/wiki/Micromouse.

[Wikipedia.org, 2021c] Wikipedia.org (2021c). Sistema sensorial. https://es.wikipedia.org/wiki/Sistema_sensorial.

A. ANEXO

Reglas de la competición Micromouse

Aunque existen ligeras variaciones entre las diferentes competiciones existen una serie de reglas las cuales definen los diferentes aspectos de la competición.
[Micromouseonline.com, 2017b].

A.1. Robot

- El robot debe ser completamente autónomo y no recibir ayuda externa.
- El robot debe tener unas dimensiones de 25 cm de ancho por 25 cm de largo como máximo, pero no tiene restricción de altura.
- Todos los robots deben estar equipados con un gancho o un enganche adecuado para retirarlo desde el centro del laberinto si esto fuera necesario.
- El método de detección de las paredes es a elección del creador, pero el robot no debe exceder una fuerza tal que dañe el circuito.
- El método de propulsión queda a elección del creador, pero la fuente de alimentación no debe generar polución, por lo que un motor de combustión interna podría ser descalificado.
- si los jueces consideran que un robot es susceptible a dañar el laberinto o ensuciarlo no permitirán que participe. Nada puede ser depositado en el laberinto. El robot

debe recorrer el laberinto y no debe saltar, trepar, rayar, dañar o destruir las paredes del laberinto.

A.2. Competición

- El tiempo requerido para viajar desde el cuadrado de inicio hasta el cuadrado central es denominado “run time” o tiempo del trayecto. El tiempo empleado para el trayecto del destino al inicio, (retorno del robot a la casilla inicial) no se considera parte de dicho “run time”. El tiempo total transcurrido desde la primera activación hasta el inicio de cada “run” también es medido. Esto se denomina tiempo de búsqueda. Si el micromouse requiere cualquier asistencia manual en cualquier tiempo del concurso, se considera "tocado".

- Cada robot dispone de un máximo de 10 minutos de funcionamiento. Esto puede ser reducido a 6 minutos si hay muchos robots que pueden ser considerados competentes.

- Los jueces tienen la potestad para solicitar que un Micromouse sea retirado pronto si no progresa o se vuelve aburrido, o si debido a su comportamiento errático pone en peligro la integridad del laberinto.

- La puntuación de un Micromouse se obtendrá calculando un tiempo de desventaja por cada “run” siguiendo la siguiente fórmula: Puntuación del tiempo de desventaja = Tiempo del trayecto o “run time”+ La penalización por búsqueda + La penalización por tocar el robot, donde: La penalización por búsqueda = 1/30 del tiempo de búsqueda, en segundos, asociado a dicho “run”. La penalización por tocar el robot = 3 segundos más 1/10 del tiempo del trayecto, en segundos, si el robot ha sido tocado en algún momento antes del trayecto.

- Cuando el Micromouse llega al recuadro de destino, debe detenerse y permanecer en el centro o continuar explorando otras partes del laberinto, o bien retroceder hasta el inicio. Si el robot elige detenerse en la casilla central, debe ser levantado de forma manual y reiniciado por el que lo manipule. El levantamiento manual debe ser considerado tocar el robot y tiene como consecuencia una penalización por tocar el robot la cual será añadida en todas las ejecuciones posteriores. Si el Micromouse no decide detenerse en la casilla de destino, puede no ser parado manualmente y reiniciado.

- El tiempo de cada trayecto se medirá desde el momento en el que el robot abandona la casilla de salida hasta que entre en la casilla de destino. El tiempo total en el laberinto (tiempo de búsqueda) será medido desde que el robot es iniciado por primera vez.

- El tiempo necesario para sortear el laberinto se medirá manualmente por los encargados del concurso, o por sensores infrarrojos situados en el inicio y en el destino. Si son empleados los sensores infrarrojos, el sensor de inicio deberá estar situado en la unión entre la casilla de inicio y el siguiente recuadro. El haz infrarrojo de cada sensor deberá estar posicionado de forma horizontal y a aproximadamente 1 cm sobre el suelo.

- El procedimiento de inicio del Micromouse deberá ser simple y no permitir la adopción de estrategias por parte del manipulador. Por ejemplo, la decisión de adoptar un enfoque veloz al trayecto al centro debe ser tomada por el Micromouse de forma autónoma. El procedimiento de inicio deberá ser comunicado a los jueces cuando el robot sea registrado el día del concurso.

- El manipulador del robot, dispondrá de un minuto desde el momento en el que el robot es extraído de su caja para realizar los ajustes que considere necesarios sobre los sensores del Micromouse. Sin embargo, no se puede realizar una selección de estrategias ni introducir algún tipo de información en memoria sobre la configuración del laberinto.

- El tiempo de búsqueda comenzará tras trascurrir el minuto aunque el manipulador continúe realizando ajustes en los sensores.

- Si un ratón tiene problemas, el manipulador puede solicitar a los jueces permiso para retirar al robot y empezar desde el principio. El Micromouse no puede ser reiniciado simplemente porque el robot ha tomado el camino equivocado, esta interpretación quedará en manos de los jueces. Los jueces pueden añadir un tiempo de penalización por el reinicio.

- Si alguna parte del robot debe ser reemplazada durante su desempeño, como pueden ser las baterías o las EPROMs o si es necesario realizar cualquier ajuste significativo, entonces la memoria del robot deberá ser borrada antes de volver a empezar. Los ligeros cambios en los sensores probablemente sean permitidos, pero las operaciones sobre la velocidad o la estrategia están expresamente prohibidas sin un borrado

de memoria. Se da por hecho, que el robot tendrá algún tipo de software almacenado en las EPROMs. Sin embargo, en algunos casos bajo criterio de los jueces, pero no en circunstancias normales, el Micromouse puede emplear una RAM respaldada por batería para descargar software de control si la memoria es borrada accidentalmente durante un trayecto. Los manipuladores, en este caso, deberán convencer a los jueces de que se ha vuelto a cargar el software original.

- Si no se ha realizado ningún trayecto exitoso, el juez puede realizar una valoración cualitativa del desempeño de los robots, basado en la distancia recorrida, el comportamiento intencional en vez del aleatorio y la calidad del control.

- Si un Micromouse decide retirarse debido a problemas técnicos, los jueces pueden, bajo su criterio, permitir que vuelva a intentarlo posteriormente en el concurso y se considerará que el robot ha tenido un tiempo extra de búsqueda de 3 minutos (es decir, si un ratón se retira después de cuatro minutos, al reiniciarlo se cuenta como si hubiera tardado siete minutos y solo tendrá tres minutos más para el trayecto). Es probable que este permiso se retire si el programa está lleno o va con retraso.

- Los jueces usarán su valoración para otorgar los premios, que además de los premios principales pueden incluir premios para otros aspectos específicos como por ejemplo el más barato, el más ingenioso, el mejor acabado y el más entretenido.

- Antes de que el laberinto sea revelado, los encargados del concurso deberán aceptar y enjaular los ratones. Los manipuladores colocarán el Micromouse al inicio bajo las instrucciones de los encargados.

- En circunstancias normales, no se pueden transferir partes de un Micromouse a otro. Sin embargo, los jueces pueden permitir cambios de baterías o controladores de posición en casos excepcionales si estos se deben a daños accidentales. En ese caso la memoria debe ser borrada con el cambio de controlador.