UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA INFORMÁTICA

PROYECTO FIN DE CARRERA

DETECCIÓN AUTOMÁTICA DE PEATONES PARA LA ASISTENCIA A LA CONDUCCIÓN UTILIZANDO FUSIÓN SENSORIAL

Autor: Almudena Domínguez Fernández

Tutor: Raúl Arrabales Moreno

Abril, 2011

El secreto del éxito se encuentra en la sinceridad y la honestidad. Si eres capaz de simular eso, lo tienes hecho.

Groucho Marx

Resumen

El proyecto trata de abordar el problema de la detección automática de peatones en un contexto automovilístico utilizando diferentes sensores. El vehículo utilizado para la experimentación se encuentra en un entorno de simulación. Como sensores, cuenta con dos cámaras y un LRF (*laser rangefinder* o telémetro láser). El sensor láser que se utilizará es capaz de medir distancias en tiempo real. Combinando esta información de distancias con la información obtenida de las cámaras procesada usando técnicas de visión artificial, se espera poder establecer patrones de información sensorial correspondientes a los peatones y por tanto detectarlos.

Este proyecto está integrado en un proyecto mayor llamado POCIMA en el que colaboran el Departamento de Ingeniería de Sistemas y Automática y el Departamento de Informática de la Universidad Carlos III de Madrid. Este proyecto tiene como objetivo crear un Sistema de Ayuda a la Conducción que vigila la aparición de peatones en la parte frontal y el ángulo muerto del vehículo, pudiendo así alertar al conductor de su presencia.

El contexto en el que se evaluará el modelo será un entorno simulado, en el que se pueden encontrar peatones o viandantes en el recorrido de un vehículo motorizado, el cual incorporará los sensores mencionados anteriormente.

Abstract

The project focuses in the problem of automatic detection of pedestrians in the road using different sensors. The vehicle used for experimentation is situated in a simulation environment. The sensors used include two cameras and one LRF (laser rangefinder). The laser sensor can be used to measure distances in real time. Combining the range measurements from the laser with information from the cameras processed using computer vision techniques we aim to establish sensory patterns for pedestrians and therefore perform the detection task.

This project is integrated into a larger project called POCIMA coordinated by the Departaments of Systems and Automation Engineering Departament and Computer Science of Universidad Carlos III de Madrid. This project aims to create a driving assistance system that monitors the appearance of pedestrians in the front and the blind spot of the vehicle, and warn the driver of their presence.

The context in which the model will be assessed is a simulated environment, in which virtual pedestrians or passers can be found in the path of a motor vehicle endowed with the aforementioned sensors.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Marco de Trabajo

Los peatones son los participantes del tráfico más vulnerables, por ello es conveniente crear un sistema de medidas preventivas para minimizar o evitar el impacto de los posibles accidentes en los que están involucrados. España es el país de Europa con mayor tasa de atropellos mortales, según un estudio elaborado en 10 países europeos por el RACC Automóvil Club y presentado hace dos años [1]. El estudio recoge que 15,7 peatones mueren en España por cada millón de personas. La cifra sitúa la media nacional a la cabeza en muertes de viandantes, seguida por Italia (11,5) y Reino Unido (10,9).

Existen dos tipos de sistemas de seguridad para reducir las cifras de siniestralidad. Estos son los sistemas de seguridad activa que tratan de evitar el accidente y los sistemas de seguridad pasiva que intentan minimizar los daños producidos. Pero los sistemas de seguridad pasiva como el cinturón de seguridad o los airbags protegen al conductor del vehículo, no así al viandante.

Por este motivo el objetivo de este proyecto es poder prevenirlo con la detección de los peatones en la vía mediante un sistema inteligente que ponga en alerta al conductor. El sistema deberá estar integrado en un sistema más completo en el que se active o bien un sistema de frenado de emergencia o bien un sistema de alarma.

Las nuevas herramientas de la tecnología de la información y de la comu-

nicación (TIC) juegan un papel muy importante en la reducción de accidentes. Es el caso de las tecnologías encaminadas a incrementar la capacidad de control del coche, como el ABS o el ESP. Dentro de los ITS (*Intelligent Transportation Systems*), los Sistemas Avanzados de Ayuda a la Conducción (ADAS) suponen un paso más; predicen y evitan un accidente que el conductor por sí solo no puede controlar. Son sistemas orientados a obtener información tanto del estado del vehículo como de la carretera, y transmitirla al conductor, para que en base a ella pueda tomar las decisiones necesarias.

A pesar del potencial de estos sistemas, su integración en los vehículos disponibles en el mercado supone un elevado coste, barreras legales y el desconocimiento de los usuarios finales. Además, la situación extremadamente competitiva del sector automovilístico crea condiciones poco propicias para el desarrollo de estos sistemas.

1.2. Objetivos

El objetivo principal del proyecto es la detección de peatones, en el contexto de la conducción. Para ello se dispone de dos cámaras y un sensor láser, con los que se debe conseguir una fusión de información para concluir en una respuesta lo más fiable y rigurosa posible.

Por lo tanto los objetivos secundarios, que se derivan del principal, se enfocan esencialmente en el procesado de la información obtenida para su posterior evaluación. Por un lado se encuentra el procesado de las imágenes, que al tratarse de dos cámaras, permite la visión esteréo y por otro lado el procesado de la información recogida por el sensor láser. A continuación, es necesario extraer los elementos importantes, obstáculos en este caso, con todas las características que los describan. Luego para la verificación de la información, es necesario un algoritmo de clasificación que determine si el elemento localizado es un peatón u otro obstáculo. Con lo que se tendrán que cubrir todas las fases del reconocimiento de patrones: adquisición, segmentación, extracción de características y clasificación.

- Para la primera fase es necesario coordinar los sensores para obtener la información adquirida por estos, concretar el formato y establecer un orden de ejecución.
- En la segunda fase se tiene que escoger una técnica de segmentación, para

dividir la imagen en regiones de interés, es decir, los posibles obstáculos. Para el láser se utiliza agrupación de los puntos cercanos. Luego se debe pasar esa información a las imágenes.

- Para la tercera fase se deben procesar esa región de interés o ROI (*Region Of Interest*), con los datos adecuados, de forma que se pueda diferenciar si se trata de un peatón o no, y así completar la última fase de clasificación.

Todo ello de forma que se pueda combinar la información recogida por el láser y la información recogida por las cámaras, creando fusión sensorial. La meta de dicha fusión se centra en obtener un resultado más fiable y riguroso. Es muy importante determinar una tasa de fallos razonable y alcanzar un valor menor.

A partir de los resultados obtenidos se comparará con estudios previos para concluir si el método utilizado en este proyecto es factible para su utilización en un entorno real y no sólo en el entorno simulado. Se pretende desarrollar un software que posteriormente se podrá incluir en un vehículo físico y ser probado y adaptado en un entorno real.

1.3. Estructura del Documento

Este documento se ha estructurado de la siguiente forma:

- El capítulo 2 describe el estado del arte de las técnicas utilizadas en este proyecto como el reconocimiento de patrones, el procesado de imagen, la visión estereoscópica, la fusión sensorial y otros sistemas de detección automática de peatones estudiados.
- El capítulo 3 hace referencia a las herramientas utilizadas para el desarrollo e implementación del proyecto centrándose en los aspectos más importantes.
- El capítulo 4 se centra en todas las fases de realización del sistema final implementado, desde el análisis hasta la implementación.
- El capítulo 5 describe la experimentación realizada y la validación de los resultados obtenidos en las pruebas.

- El capítulo 6 expone las conclusiones, las líneas de trabajo futuro y las posibles mejoras del sistema implementado.

Capítulo 2

Estado del Arte

Dentro de la seguridad automovilista, la detección de peatones se está convirtiendo en un foco de creciente interés, al proteger a las personas más vulnerables de la vía. Para afrontar este problema se están siguiendo varias líneas diferentes de investigación.

El peso del funcionamiento de un detector de peatones típico, recae en el conjunto de rasgos extraídos. Para ello, se deben tener en cuenta los sensores que se utilizan para la percepción. Los sistemas basados en visión son muy utilizados. La visión artificial ofrece una serie de interesantes cualidades, que pueden ser aprovechadas con el fin de detectar a los viandantes, entre ellas el uso del color o la detección de bordes. Además es el método más próximo a la forma de percepción que utilizan las personas.

Sin embargo, existen varios problemas con los sistemas basados en visión. El escenario es complejo, los elementos aparecen de forma aleatoria y puede haber gran cantidad de ellos y las condiciones de iluminación pueden ser muy cambiantes y desconocidas. Además las personas pueden presentar una apariencia muy diversa, tanto de vestimenta, de postura y de proporciones y por otro lado se pueden encontrar oclusiones o agrupaciones de personas que impiden o dificultan el reconocimiento.

La visión artificial por sí misma no es suficiente en muchos casos por lo que se combina con otro de tipo de sensores como pueden ser los telémetros láser o sensores radar. Al combinar diferentes sensores, el sistema debe procesar toda la información adecuadamente para dar unos resultados coherentes, es lo que se

llama fusión sensorial.

Para llevar a cabo la clasificación, los modelos más usados son los referentes a la intensidad, el color de la piel o rasgos de la cara, la proporción y la forma del cuerpo humano y los modelos basados en el movimiento. Los humanos tienen un modo de caminar característico que puede ser utilizado como patrón para la clasificación. Los movimientos periódicos de los peatones siguen un patrón distintivo; el balanceo de las piernas caracteriza la oscilación única de los humanos.

2.1. Reconocimiento de patrones

El reconocimiento de patrones se define como una técnica mediante la cual a partir de información recogida del exterior, normalmente de los sensores, se calcula si está coincide con un patrón establecido con anterioridad. Para ello es necesario procesar dicha información adquirida por lo sensores para poder compararla con el patrón. Por patrón se entiende un posible objeto predefinido dentro del universo de trabajo [10].

Dentro del reconocimiento de patrones se diferencian cuatro fases básicas: adquisición de los datos, segmentación, extracción de características y clasificación como se muestra en la siguiente figura.

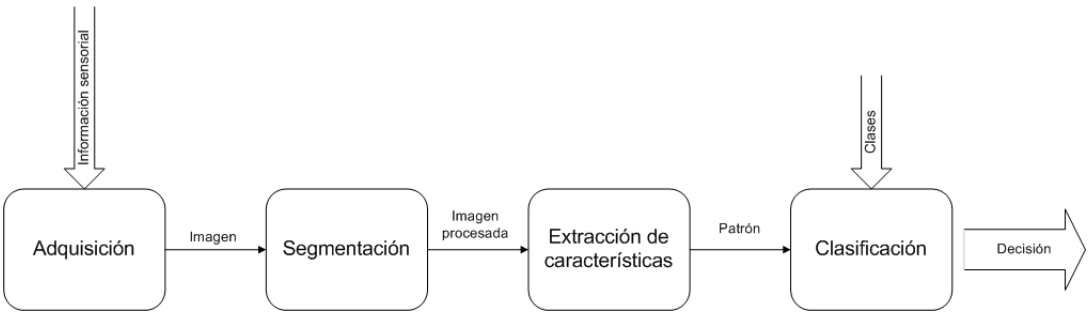


Figura 2.1: Esquema del proceso de reconocimiento de patrones.

2.1.1. Adquisición de datos

Esta tarea se centra en como los sensores recogen la información del exterior para su posterior procesado. Para este proyecto se utilizarán como sensores dos

cámaras y un sensor láser, por lo que se puede construir un modelo de la escena en tres dimensiones.

Normalmente en un sistema de visión artificial se obtienen imágenes como resultado de la adquisición, es decir, una matriz que representa el nivel de gris en cada punto de la escena. En el caso de visión estereoscópica, como es el caso en este trabajo, se obtienen dos imágenes de la misma escena pero desviadas una de la otra una distancia llamada disparidad binocular. Conociendo la situación y orientación de las cámaras es posible hallar mediante geometría epipolar la profundidad de un elemento de la escena, con lo que se tendrá para cada punto las tres coordenadas (x,y,z). El problema de la visión estereoscópica es la correspondencia, que surge del hecho de encontrar una región o elemento característico que se pueda localizar en las dos imágenes. Como es lógico no se puede hacer ese cálculo para elementos no visibles en las dos cámaras, con lo que la fusión con la información recogida por el telémetro láser podrá subsanar esta limitación.

La información sensorial recogida por el telémetro láser es adquirida mediante el cálculo del tiempo transcurrido entre la emisión y detección de un pulso láser que determina la distancia a los objetos que integran la escena de manera directa. Por tanto la información que se obtiene es un mapa de profundidad. El sensor láser cubre un rango de acción determinado, que viene dado por un campo de visión o ángulo que define la apertura máxima del rayo y un alcance máximo, es decir, la distancia máxima que es capaz de captar. Por lo tanto, se obtiene para cada punto dentro de ese rango de acción una distancia, que si no encuentra ningún obstáculo será igual al alcance máximo.

2.1.2. Segmentación

La segmentación es la fase más complicada del reconocimiento de patrones ya que trata de extraer los objetos individuales del medio en que se encuentran y distinguirlos entre sí. No se contempla el solapamiento o posible ocultación de los objetos, lo que supone una tarea extremadamente delicada, dada la enorme dificultad en conseguir mediante técnicas automáticas el aislamiento de los diferentes objetos que puedan hallarse en la imagen.

Como técnicas más habituales para definir los objetos están la detección de bordes, el análisis de texturas o la detección de movimiento, como se describe en [11]. Los objetos de la imagen tienen que cumplir similitud y conectividad entre los píxeles y discontinuidad en los bordes.

Las técnicas para obtener la segmentación se basan en la búsqueda de las zonas uniformes de la imagen o por el contrario, aquellas donde se produce un cambio. Según se detalla en [10], se puede utilizar un tipo de método, otro o ambos conjuntamente. Dentro de las técnicas basadas en la detección de bordes se encuentran el método del gradiente y la transformada de Hough con rectas o con circunferencias. Las técnicas que se basan en la uniformidad de regiones, también llamadas de umbralización contemplan métodos como histogramas laterales, etiquetado, crecimiento de regiones y división y unión de regiones (*Split and Merge*). También se pueden utilizar técnicas basadas en la forma y estructura de los objetos, como ocurre en este trabajo, las llamadas transformaciones morfológicas.

Otra forma de llevar a cabo la segmentación es mediante la visión estereoscópica y los mapas de disparidad. Además es posible obtener el tamaño real de los objetos que luego puede ser utilizado para la clasificación.

2.1.3. Extracción de características

Después de separar los objetos tiene lugar el proceso de representación o descripción individualizada de los mismos. Hay que hacer hincapié en la diferencia entre representación y descripción de los objetos. Mediante la representación es posible volver a reconstruir el objeto original, mientras que con la descripción no es necesario que sea reversible, pero sí que identifique al objeto de manera que la descripción sea independiente de cambios de tamaño, traslación o rotación. Para la labor de clasificación o interpretación de la imagen interesará obtener descripciones de los objetos, aunque una representación también puede servir como descripción.

Las características elementales están explícitamente presentes en los datos adquiridos y pueden ser pasadas directamente a la etapa de clasificación. Las características de alto orden, sin embargo, son derivadas de las elementales y son generadas por manipulaciones y/o transformaciones en los datos.

Las técnicas de representación de objetos más utilizadas son los códigos de cadenas, aproximaciones poligonales y signaturas para representar la frontera y quadtrees, segmentos frontera o esqueletos para definir el interior de la región.

Para las técnicas de descripción encontramos la misma separación, de frontera y de región. Los descriptores de frontera simples son la longitud, el diámetro o la curvatura, pero también encontramos otros derivados como los números de

forma, descriptores de Fourier y descriptores estadísticos. Los descriptores de región simples pueden ser la posición, el área, la compacidad y los derivados como descriptores topológicos o basados en texturas.

2.1.4. Clasificación

Tras los procesos de segmentación, extracción de características y descripción, cada objeto, queda representado por una colección o agrupación de descriptores denominada patrón. Existen tres formas de definir los patrones, con vectores, con cadenas o con árboles. Una clase representa a un conjunto de patrones que tienen alguna propiedad común. Cada patrón se supondrá perteneciente a una clase.

Una vez calculado el patrón asociado a un objeto individual, su reconocimiento automático se basa en determinar su grado de semejanza con los patrones prototipos de cada posible clase de objetos previamente definidos. El nivel de separación entre las clases depende de una selección de los descriptores adecuada. Las propiedades que debe cumplir un buen patrón son que tenga capacidad discriminante, fiabilidad, independencia, economía y rapidez de cálculo.

Primeramente es necesario definir las clases en el universo de trabajo. Normalmente la etapa en la que se define el universo de trabajo es trivial, el diseñador conoce las clases de objetos que se pueden identificar en la escena, es el llamado reconocimiento supervisado. Sin embargo, puede ocurrir que no se disponga de ese conocimiento por lo que es necesario recurrir a técnicas de agrupaciones o *clustering* formando las clases sobre la marcha, por tanto sin supervisión. En este proyecto se va a utitilizar reconocimiento supervisado, porque se conocen a priori las clases existentes. Una vez elegidas las clases el siguiente paso consiste en definir el vector de características a utilizar y la técnica de reconocimiento adecuada.

No existe un mecanismo concreto para la elección de las características contenidas en el vector, depende en gran medida de la aplicación, la intuición y la experiencia del diseñador. Se trata de experimentar con diferentes vectores y funciones discriminantes hasta encontrar la combinación adecuada.

Dentro de las técnicas de reconocimiento existen dos tipos de métodos, los basados en la teoría de decisión que tratan con descriptores cuantitativos y los métodos estructurales, que tratan con descriptores cualitativos. Se va a tratar más en profundidad los métodos basados en teoría de decisión que utilizan funciones

discriminantes para separar las clases, unas de otras. Se pueden dividir en tres bloques:

- **Adaptación (*Matching*)**: Los clasificadores basados en adaptación representan cada clase mediante un patrón prototipo. El patrón a asignar se cataloga según la clase más cercana. Existen dos métodos dentro de este tipo de clasificador, los de mínima distancia y adaptación por correlación.
- **Clasificadores estadísticamente óptimos**: Se basan en que el clasificador es óptimo cuando la clasificación proporciona la menor probabilidad de error. En este bloque se encuentran los clasificadores bayesianos.
- **Redes neuronales**: Este término engloba un conjunto de técnicas que proporciona soluciones flexibles, adaptables a cada problema, para cada una de las estrategias. Se definen las funciones discriminantes mediante patrones de entrenamiento.

2.2. Procesado de imagen y Visión estereoscópica

Dentro del procesado de imagen existen diversas técnicas que son más o menos útiles según el objetivo que se quiera alcanzar. Las técnicas más habituales son los filtrados, que realizan una transformación uniforme sobre la imagen, para, por ejemplo, quitar ruido o realzar bordes. Otras técnicas útiles sobre todo en la etapa de segmentación son las llamadas transformaciones morfológicas que tratan las imágenes como conjuntos de píxeles que se relacionan entre sí. Por tanto, mediante operaciones sobre conjuntos se puede llegar a segmentar la imagen y extraer características útiles para la fase de clasificación.

Referido al contexto del reconocimiento de patrones y más concretamente a la detección de peatones, las técnicas más utilizadas son las relacionadas con simetría vertical, el análisis del campo del flujo óptico que se basa en el movimiento o la visión estereoscópica.

La simetría vertical se basa en que las personas de pie tienen más bordes verticales que horizontales y además presentan cierta simetría en el eje vertical. Se recoge la información en un mapa de simetrías de los bordes horizontales. Si se recorre la imagen, por columnas, se puede identificar la región de interés como la región con mayor número de bordes horizontales.

El movimiento es una característica que se puede utilizar para localizar regiones de interés de una manera fiable, sin embargo sólo detectaría peatones en movimiento y además es necesario tener una secuencia de imágenes para obtener un resultado. En la fase de seguimiento también es necesaria una secuencia de imágenes y se utiliza para corroborar los resultados obtenidos y añadir información para la clasificación como la velocidad.

La visión estereoscópica permite mediante los mapas de disparidad, densos o dispersos, dividir la imagen en capas según la distancias a las que se encuentren los elementos de la imagen.

2.2.1. Transformaciones morfológicas

La transformación morfológica es una técnica de procesado no lineal de la imagen, interesada en la geométrica y forma de los objetos, no en intensidades. Los fundamentos del análisis y procesado morfológico se basan en el álgebra de conjuntos y en la topología.

Se parte de una imagen binarizada, normalmente, y un elemento estructurante (EE). Este EE es un patrón de ajuste que se usa como una sonda para examinar la estructura geométrica de una imagen. Puede ser considerado como una máscara de convolución a pesar de que en este caso se trabaja con operadores sobre conjuntos y la convolución se basa en operaciones aritméticas. Pueden tener cualquier forma (horizontal, vertical, cuadrada, etc.) su centro se sitúa en cada píxel de la imagen original, y se aplica la operación morfológica sobre los puntos situados bajo el EE.

Las transformaciones básicas, con las que luego se pueden formar algoritmos más complejos y específicos son la dilatación y la erosión. La dilatación añade todos los puntos del fondo que tocan el borde del objeto. Sirve para rellenar agujeros o bahías. La ventaja frente a un filtro de paso bajo es que para este caso el resultado es una imagen binaria, mientras que con el filtro hace falta aplicar un umbral previamente. La fórmula matemática que define este operador es como se muestra a continuación.

$$\delta_c(A) = A \oplus C = \left\{ x \mid (\widehat{C})_x \cap A \neq 0 \right\}$$

donde A representa un conjunto que define la imagen, C es el elemento estructurante y la dilatación se interpreta como la unión de estos dos conjuntos en base a

la reflexión de C con respecto a su origen y un desplazamiento x. La salida de la dilatación se peude explicar también como el conjunto de puntos barridos por el centro del EE mientras algún punto de C coincida con alguno de A.

La erosión, en cambio, elimina grupos de píxeles donde el EE no cabe. La salida de la erosión es el conjunto de puntos barridos por el centro del EE mientras se cumpla que todos los puntos de C están contenidos en A. Una de las aplicaciones más típicas de la erosión es la eliminación de detalles irrelevantes, desde el punto de vista del tamaño. Su fórmula en términos de conjuntos se muestra seguidamente.

$$\varepsilon_c(A) = A \ominus C = \{x \mid C_x \subset A\}$$

donde A, nuevamente, representa el conjunto de la imagen, C el elemento estructurante y la erosión es igual al conjunto de todos los puntos x, tales que C trasladado x, este contenido en A.

Con estas técnicas básicas, erosión y dilatación, se pueden crear algoritmos morfológicos más complejos como:

- Hit or Miss (acierta o falla): Determina la localización de cierta forma.
- Extracción de bordes: Se puede definir con la fórmula $A - (A \ominus EE)$
- Relleno de regiones: rellenar una región definida por una frontera.
- Extracción de componentes conectadas: encontrar, dada una semilla, que región de la imagen está conectada a ella.
- Envolvente convexa: es como si se rodease con una goma elástica la figura.
- Adelgazamiento: Reduce estructuras a espesor de un solo píxel.
- Engorde: determina la zona de influencia del esqueleto.
- Esqueletización: obtiene una estructura que es representativa en la descripción del objeto.
- Poda: eliminar componentes espurias resultantes de aplicar operaciones morfológicas de tipo adelgazamiento o esqueletización.

2.2.2. Visión Estereoscópica

La visión estereoscópica trata de la reconstrucción de información 3D a partir de múltiples vistas. Los fundamentos de la visión estereoscópica se encuentran en la geometría epipolar. Las imágenes son proyecciones 2D que suprimen información de profundidad. Está puede recuperarse mediante dos proyecciones o vistas y triangulación.

Dada la posición de los ojos en los humanos y la forma de moverlos las imágenes que se reciben en cada ojo son prácticamente iguales, con una diferencia en la posición relativa de los objetos. Estas diferencias relativas en la posición en cada imagen, la disparidad, tiene una relación directa con la distancia, profundidad a la que se encuentran los objetos entre sí, y al observador. El cerebro es capaz de interpretar esa diferencia y reconstruir la escena. Según *Marr* y *Poggio* [21] existen tres etapas en el proceso de recuperación de la estructura de una escena. Estas son, primero, seleccionar un punto característico de un objeto en una de las imágenes; segundo, encontrar el mismo punto característico en la otra imagen, y tercero, medir la diferencia relativa, disparidad, entre la posición de estos dos puntos. Se llama visión estéreo a esta capacidad de recuperar la estructura tridimensional de una escena a partir de, por lo menos, dos vistas o imágenes diferentes de la misma.

Considérese el sistema estéreo de la siguiente figura, donde se han colocado dos cámaras exactamente iguales, separadas una distancia $d = O_L - O_R$, con ópticas de distancias focales idénticas, situadas de forma que sus ejes ópticos sean colineales. El punto del mundo 3D P se proyecta en las dos imágenes en los píxeles p_L y p_R.

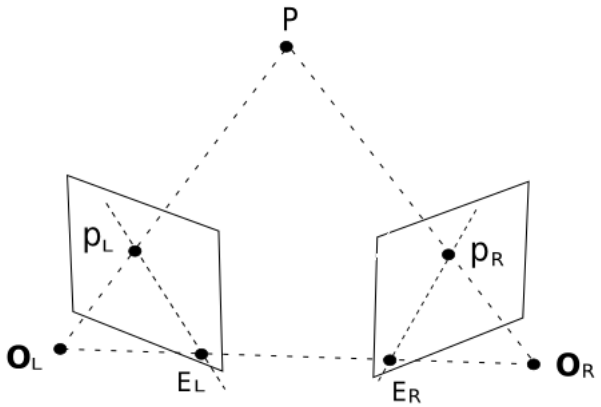


Figura 2.2: Ejemplo de geometría epipolar.

Si a partir de p_L se desea conocer la ubicación de p_R, es necesario proyectar en la imagen derecha los posibles puntos que puede ser p_R. La recta que forma todos esos puntos es la llamada línea epipolar. El epipolo, en la imagen representado como los puntos E_L y E_R, es la proyección del centro óptico de la imagen contraria. Y por último el plano epipolar se define por los centros ópticos O_L, O_R y el punto P. Tanto las dos proyecciones p_L y p_R del punto P como sus epipolos E_L y E_R, están contenidos en el plano epipolar.

Así mediante la geometría epipolar es posible restringir la búsqueda de puntos semejantes en la otra imagen a una única línea, la línea epipolar. El problema de correspondencia se reduce a un problema de búsqueda unidimensional. Para ello es necesario antes rectificar las imágenes, para que la búsqueda se base en líneas horizontales, que es más fácil de implementar. Para ello se deben calibrar las cámaras, para que tengan un vector de desplazamiento paralelo a los planos de proyección.

La geometría epipolar dice dónde buscar las correspondencias pero no cómo. Existen varios métodos como los mapas de disparidad densos o dispersos y las técnicas basadas en correlación locales y globales. Pero los problemas de oclusión y ambigüedad son inevitables. Además son métodos lentos y que requieren gran capacidad de cómputo.

Cuando se puede recuperar la estructura de la escena, es posible utilizar ésta información para la segmentación. En el caso de poder obtener una secuencia estéreo es posible obtener la disparidad de los objetos en la escena, y utilizarlo como característica.

2.3. Fusión Sensorial

El objetivo de la fusión sensorial es la combinación coherente de medidas tomadas con sensores diferentes, incorporando la incertidumbre en las medidas a las propias estrategias de reconocimiento.

La percepción se define como la captación de información a través de los sentidos y su posterior procesamiento para darle significado. La percepción es la manera de interpretar y entender la información que recibimos a través de los sentidos; es selectiva, constructiva e interpretativa. Los cinco sentidos: vista, oído, olfato, tacto y gusto son exteroreceptores. Según el asociacionismo, las sensaciones se

perciben aisladas y la suma de todas ellas constituirá la percepción global. Según la escuela psicológica de la Gestalt, el todo es más que la suma de las partes, la impresión total es la de un todo coherente [9].

El procesamiento de datos sensoriales o estímulos y su interpretación son características básicas de los seres inteligentes para interactuar con el entorno. Los seres humanos combinamos los distintos estímulos percibidos de manera inconsciente, el cerebro registra momentáneamente todos los estímulos pero solo pasa al primer registro de memoria una cantidad limitada. En nuestro día a día fusionamos la vista, el olfato, el oído, el tacto y el gusto sin ningún esfuerzo aparente. Según la literatura, la integración multisensorial se divide en tres categorías: fusión de los datos sensoriales, planificación multisensor y arquitectura multisensor [2].

2.3.1. Fusión de datos sensoriales

Se combina la información de varios sensores o incluso información interna del propio sistema para conformar una única representación formal de toda la información.

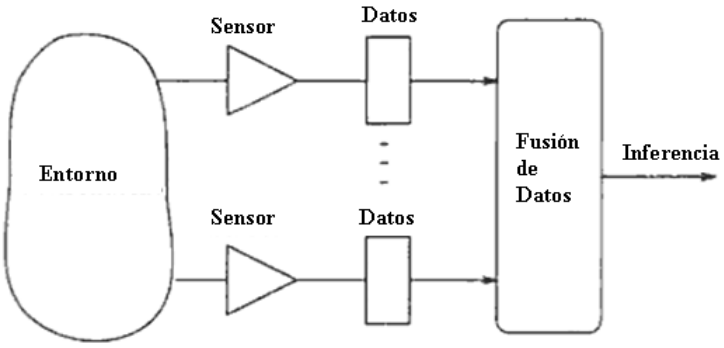


Figura 2.3: Fusión de Datos sensoriales [2]

Las posibilidades con el uso de un único sensor son muy limitadas. Esto se justifica por varias razones a considerar, como que exista redundancia para obtener una mejor estimación de los resultados; que si en un momento dado, alguno de los sensores no funciona correctamente o deja de funcionar el otro u otros, puedan realizar la tarea específica; la cobertura espacial y temporal se puede ampliar

aumentando el número de sensores; los errores provenientes por el ruido o un mal calibrado del sensor pueden ser corregidos; puede existir incertidumbre en los datos recogidos (p.e. por oclusión o ambigüedad) que puede ser solventada con el uso de varios sensores.

El objetivo de la fusión sensorial es combinar información proveniente de múltiples sensores. Las técnicas utilizadas para ello se pueden englobar en diversas áreas como la inteligencia artificial o la estimación estadística. Según *Luo y Kay* [3] las técnicas existentes pueden agruparse en cuatro categorías de nivel de fusión: de señal, píxel, característica y símbolo.

- **Nivel de señal**: Se juntan las señales de sensores similares para formar una única señal mas precisa y normalmente del mismo tipo que las percibidas. Es necesario que exista correspondencia espacial y temporal con las señales recibidas. Incluye métodos como el de media ponderada, filtros de Kalman, estimación bayesiana y métodos de teoría de decisión estadísticos.
- **Nivel de Píxel**: trata de combinar varias imágenes en una sola con mayor contenido de información. Se necesita correspondencia espacial y temporal al igual que las técnicas de nivel de señal. Los métodos que se incluyen son filtros lógicos, morfología matemática, algebra de imágenes y algoritmos Simulated Annealing.
- **Nivel de característica**: resulta de la combinación de las características derivadas de distintos sensores para completar una representación final. Los métodos que se utilizan son estimación por Gauss-Markov con restricciones, y filtros de Kalman extendidos.
- **Nivel de símbolo**: combina distintos símbolos con su medida de incertidumbre asociada, cada uno representa una decisión. También es referido como fusión de decisiones. Los métodos usados son estimación bayesiana, razonamiento evidencial Dempster-Shafer, reglas de producción con factores de certeza y lógica difusa.

Según *J.J. Clark* [4] existen dos tipos de algoritmos en fusión sensorial, los débiles y los fuertes. Los primeros se dividen en tres escenarios para llevar a cabo la fusión sensorial, el **competitivo** donde la información recogida por cada sensor hace referencia a lo mismo; el **complementario** donde cada sensor recoge información de aspectos diferentes y que conjuntamente se obtiene un resultado combinando estos aspectos como en una fórmula matemática; el **híbrido** que resulta de una mezcla de los dos anteriores. Los algoritmos de fusión de datos fuertes se

caracterizan por la dependencia de los sensores, es decir, cuando uno necesita la información procesada de otro.

2.3.2. Planificación Multisensor

La planificación trata de establecer en que momento se adquiere la información y como hacerlo. A veces la cooperación entre sensores es necesaria para lograr una tarea sensitiva. Necesita información sobre que sensores y actuadotes están disponibles, el estado del sistema interno, el estado del entorno externo y los requisitos de la tarea global.

El objetivo final es minimizar el tiempo, hacer un uso eficiente de los recursos sobre todo en entornos de tiempo real. En ocasiones los sensores y los algoritmos de procesado de datos tienen constantes de tiempo que varían considerablemente, y la estrategia seguida debe ser planeada correctamente.

Existe planificación estática o dinámica. La planificación estática tiene lugar antes de la inicialización del sistema, en la fase de diseño, y no varía durante la ejecución. La planificación dinámica, sin embargo, se modifica según los cambios producidos en el entorno.

2.3.3. Arquitectura Multisensor

La arquitectura multisensor trata de la organización de control y el flujo de datos. Se enfoca en aspectos del sistema como la modularización, la coordinación, la planificación, la robustez y la comunicación de datos entre los distintos módulos. La arquitectura multisensor puede ser dividida en la **organización de datos** que incluye la representación y flujo de los datos en el sistema y **la organización de control** que trata el flujo de control.

Las arquitecturas descentralizadas usan una red de sensores con procesamiento local y sólo necesitan comunicación nodo a nodo. Las redes de sensores distribuidas se usan para crear sistemas tolerantes a fallos.

2.3.4. Aplicaciones

La fusión sensorial es utilizada en multitud de áreas incluyendo la robótica, orientación y control de vehículos autónomos, reconocimiento de objetos automático, seguimiento de objetos y vigilancia aérea en el ámbito militar, monitorización de maquinaria compleja y fusión de señales en el ámbito médico. Este proyecto se puede incluir en el área de reconocimiento automático de objetos.

2.4. Detección Automática de peatones

Existen varios antecedentes de fusión sensorial para el reconocimiento de peatones, como el sistema Protector [5] que agrupa la información de radar, láser y video para conformar la fusión. El láser es capaz de captar la distancia al objeto y la velocidad del mismo, así como la clasificación. La información recogida por el radar se basa en las características de reflexión de los peatones y tiene ventajas como que no le afecta la visibilidad reducida, contempla los ángulos muertos, y no se ve afectado por inclemencias meteorológicas como la nieve, el hielo o el polvo. Con la visión artificial primero se define la búsqueda dentro de una región de interés (ROI), donde se detecta un objeto y seguidamente se clasifica para comprobar que es un peatón. Se utiliza triangulación para visión estereoscópica y poder calcular distancias.

En otros sistemas como el propuesto por Tatschke [6] en el que en lugar de avisar al conductor, se implementa un sistema de frenado automático, por lo que su procedimiento de detección debe ser más preciso y rápido que los otros, se basa en que es capaz de reaccionar antes que un humano. Para ello utiliza el láser, radar y cámara de infrarrojos para detectar el calor que desprenden los peatones. La fusión sensorial se establece cíclicamente en tres fases llamadas predicción temporal, asociación de datos y corrección de medidas. La primera establece que la información recogida por los sensores se establezca en el mismo formato, la segunda que los objetos detectados sean los mismos y en la tercera se establecen los posibles errores y se corrigen.

En [7] sin embargo se utiliza la distancia de Hausdorff[1] y redes neuronales para clasificar las regiones de interés, luego normaliza para que no le afecten las

[1]Método para comparar objetos y calcular distancias. Es más eficiente que los métodos de correlación clásicos porque utiliza max-min en lugar de multiplicación.

modificaciones de escala y se confirman las hipótesis con el paso de tiempo. El grado de confianza de la clasificación se basa sobre todo en el seguimiento. Para la fusión utiliza una arquitectura de competición con las salidas de los sensores ya procesadas mediante varios algoritmos.

En el trabajo de Zhao y Thorpe [23] se propone un sistema de detección de peatones basado en redes neuronales y visión estereoscópica. El algoritmo que utilizan se divide en tres pasos. Primero se realiza la segmentación en objetos candidatos mediante el mapa de disparidades descartando los objetos del fondo por umbralización; luego se divide en sub-imágenes de tamaño y forma del peatón; por último se utiliza una red neuronal entrenada que distinga las regiones de interés normalizadas a un tamaño de 30x65 según características basadas en la forma. Los resultados confirman que el sistema diseñado es capaz de detectar los peatones de diferente forma, tamaño, ropa, pose e incluso es invariante a las oclusiones; funciona correctamente en tiempo real y es robusto a iluminación y cambios del fondo.

En [22] proponen un sistema de fusión sensorial cooperativo con visión estéreo y un escáner láser para detectar obstáculos. Estima la posición, la altura, el ancho y la profundidad de los objetos para poder clasificarlos. Para ello primero se calcula el mapa de disparidad no denso y se crea un histograma de las disparidades, se extraen las superficies globales y se deducen los obstáculos. Para detectar el plano de la carretera utilizan la transformada de Hough, y después se calcula la intersección del obstáculo y la carretera para filtrar la información irrelevante. La idea es usar la descripción geométrica de la carretera obtenida por estereovisión para filtrar la información del láser. El sistema propuesto es robusto, preciso, soportado en tiempo real y es capaz de detectar obstáculos de noche, con una iluminación escasa.

Respecto a la detección de peatones mediante sensor láser existe un precedente propuesto por Fuerstenberg [8] que propone un algoritmo de seguimiento fiable que crea perfiles 2D del entorno. Las medidas recogidas se agrupan si parecen pertenecer al mismo objeto (segmentos) según un criterio de distancia entre los puntos medidos. Estos segmentos son representados por varios parámetros como los puntos izquierdo y derecho más cercanos al sensor y además se tiene el centro de gravedad geométrico de todos los puntos. Luego con el filtro de Kalman se reconocen los posibles obstáculos y se calcula la velocidad longitudinal y lateral del objeto. Si los objetos son tapados por otros cercanos, una reconstrucción del objeto tapado es posible. Para la clasificación se utiliza un algoritmo basado en la información dinámica, más concretamente en el movimiento de piernas característico de los seres humanos.

Capítulo 3

Herramientas

3.1. Microsoft Robotic Developer Studio (MRDS)

MRDS (*Microsoft Robotics Developer Studio*) [12] es una herramienta para la creación de aplicaciones robóticas, orientado al ámbito académico, desarrollo comercial y a los aficionados al mundo de la robótica, ya que es accesible y de fácil manejo. Tiene soporte de tiempo real orientado a servicios y está desarrollado sobre el entorno .NET y soporta varios lenguajes de programación como C#, VB.NET e Iron Phyton. Entre las ventajas existentes podemos destacar que es una plataforma extensible, escalable y además se puede utilizar tanto para un entorno simulado o un entorno real conectando el robot por un puerto serie del PC, Bluetooth, WiFi o modem RF. También es posible interactuar con el robot a través de interfaces basadas en Web, como JavaScript o HTML.

El entorno de desarrollo MRDS consta de tres componentes principales: el entorno de simulación VSE, el lenguaje VPL y el entorno de ejecución (CCR y DDS).

Respecto al primero está diseñado para utilizar en una variedad de escenarios que demandan una alta fidelidad, visualización y escala. La integración de las tecnologías AGEIA PhysX permite realizar una simulación física con una evolución madura y constante.

El lenguaje VPL, por otra parte, está enfocado principalmente a usuarios poco experimentados que necesitan adaptarse rápidamente a la plataforma, ya que es

muy sencillo y gráfico. Está basado en un modelo de programación de flujo de datos gráfico, es decir, no se debe expresar los comandos de forma secuencial, sino que se desarrolla como una cadena de montaje donde cada modulo tiene asignada una tarea y las entradas y las salidas de datos enlazan estos módulos.

Por último el entorno de ejecución se compone a su vez de dos elementos el CCR (*Concurrency and Coordination Runtime*) y el DSS (*Decentralized Software Services*) que se explican a continuación.

3.1.1. Concurrency and Coordination Runtime

El CCR se engloba en una librería de código manejado DDL (*Dynamically Lincked Library*), accesible para cualquier lenguaje que soporta .NET. Surge de la necesidad de controlar aplicaciones orientadas a servicios que manejan operaciones de entrada/salida asíncronas. Trata la concurrencia de procesos y explota el paralelismo dentro de lo posible. Dentro de la implementación CCR encontramos tres funcionalidades esenciales que son:

- Los puertos (*Port* y *PortSet*): son el punto de interacción entre dos componentes o módulos y consiste en una cola de tipo FIFO de elementos que van llegando. Para un elemento recién llegado si no hay ningún árbitro ligado al puerto se añade a la cola, en caso contrario se evalúa y se decide si se puede ejecutar. La clase del puerto es de tipo genérico, se puede definir la clase de los elementos que se reciben. En el caso de *PortSet* pueden coexistir elementos de distintos tipos en el mismo puerto.
- Los árbitros (*Arbiter*): como se ha dicho antes, evalúan los elementos entrantes para decidir que tarea o tareas deben ejecutarse. Existen dos escenarios, uno en el cual hay una respuesta independiente para cada petición de entrada y otro en el que se debe esperar a varias peticiones para generar una respuesta. Los tipos de árbitros que existen en CCR son los siguientes:
  - *Receive*: Tiene como parámetros en la llamada, un indicador que define si es persistente o no, un puerto donde se recibe la información y un manejador o método que la procesa.
  - *Choice*: Es similar al anterior pero se tienen dos flujos de ejecución según el contenido del elemento dentro del puerto. Si es de un tipo se ejecutará una acción y si es de otro, la otra acción. Por lo tanto se definen dos tareas en lugar de una. Se puede interpretar como un

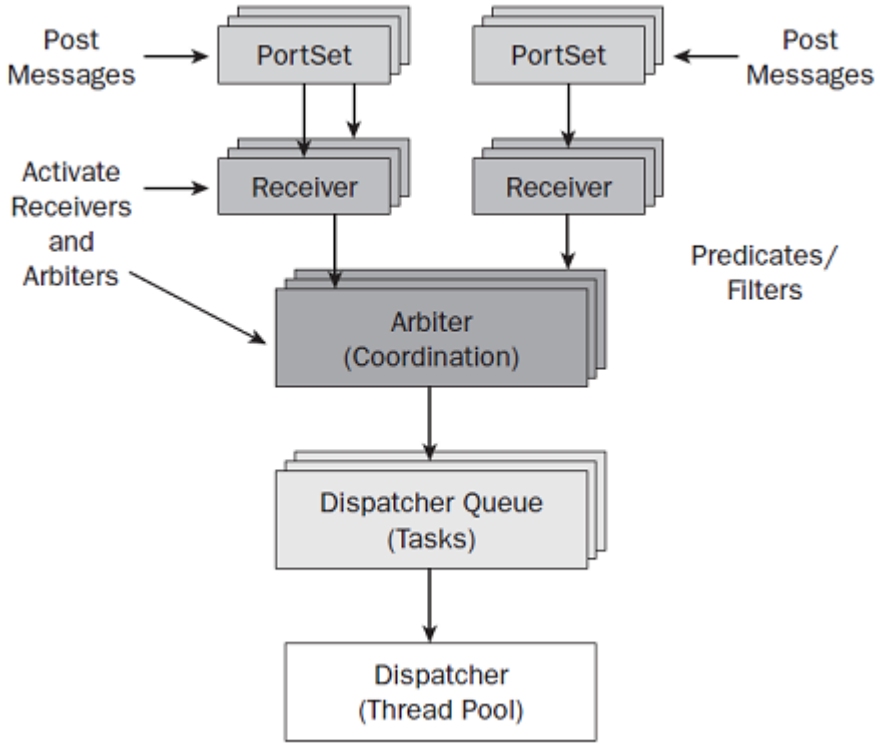


Figura 3.1: Esquema de los elementos que participan en la ejecución de un programa MRDS.

operador lógico OR. Se utiliza comúnmente para tratar fallos como con las excepciones.

- *Join*: En este caso se espera por varios puertos, es decir, cuando llegan los elementos esperados por dos puertos distintos entonces es cuando se ejecuta el manejador y nunca antes. Se puede traducir como el operador lógico AND. En términos de concurrencia, permite el rendezvous [1] , de tal forma que hasta que los dos procesos no acaban no se continúa con la ejecución.

También existen en el escenario en el que se reciba más de un mensaje los árbitros siguientes:

- *MultpleItemReceive*: Permite recibir un número especificado de mensajes en un sólo puerto.

[1]Rendezvous: método de sincronización en el cual dos procesos acuerdan sincronizarse en un punto de la ejecución.

- *MultiplePortReceive*: Se diferencia del anterior en que en lugar de un puerto normal utiliza un *PortSet*.

En el caso de tener muchas operaciones, como es lo normal en un servicio, para ello también existe un elemento llamado *Interleave* que define un modo de ejecución para un grupo de recibidores, que puede ser:

- *Tear Down*: el grupo se ejecuta una única vez y se cierra el *Interleave* automáticamente.
- Concurrente: todos los procesos se ejecutan paralelamente dentro de las posibilidades de las que se disponga.
- Exclusivo: Se ejecuta de forma atómica, es decir bloqueando al resto de procesos que puedan estar ejecutándose.

- Los despachadores (*Dispatcher*): Asignan cada tarea a un hilo del Sistema Operativo, es decir, evalúan los recursos HW disponibles y programan el orden de ejecución de las tareas y las políticas a seguir.

3.1.2. Decentralized Software Services

El DSS se sitúa en un nivel por encima del CCR y proporciona un modelo orientado a servicios. Los servicios son los bloques básicos que sirven para representar en HW sensores o actuadores, y en SW interfaces de usuario, directorios, etc. Los servicios se pueden comunicar entre sí mediante paso de mensajes. Está diseñado así, en bloques independientes, para poder facilitar la reutilización de código.

Un servicio esta compuesto por una serie de componentes como aparecen en la figura 3.2 .

- El identificador del servicio para la comunicación con otros servicios o para la interacción vía Web.
- El identificador de contrato que describe la funcionalidad del servicio y se incluye en la DLL y luego facilita la reutilización del mismo.
- El estado que es la representación del servicio en un punto del tiempo especifico.

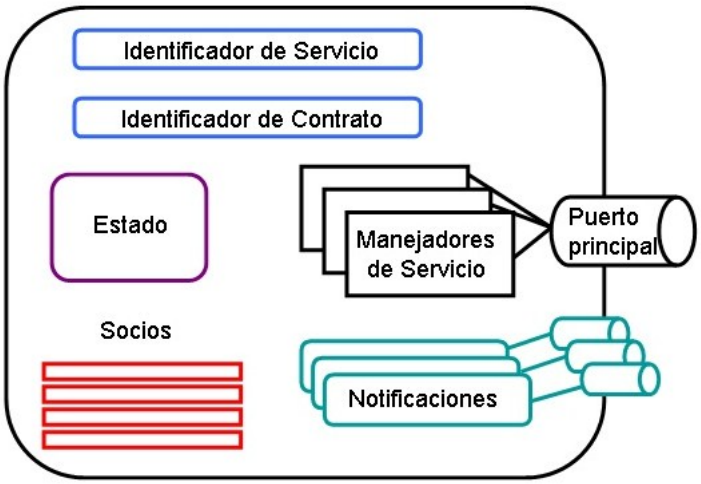


Figura 3.2: Representación de un servicio DSS.

- Los socios que son otros servicios de los que depende este o con los que interactúa.
- El puerto principal que se corresponde con el puerto de CCR, y recibe mensajes de otros servicios.
- Los manejadores de servicio que operan con los mensajes entrantes.
- Las notificaciones, que son mensajes generados por un evento, es decir, un cambio en el estado del servicio.

3.2. Microsoft Visual Studio

Microsoft Visual Studio es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET. Esta herramienta se ha utilizado en este proyecto para escribir la implementación en el lenguaje C#.

Microsoft Visual Studio admite Visual C# con un editor de código completo, plantillas de proyecto, diseñadores, asistentes para código, un depurador eficaz y fácil de usar, además de otras herramientas. La biblioteca de clases .NET Framework ofrece acceso a una amplia gama de servicios de sistema operativo y a

otras clases útiles y adecuadamente diseñadas que aceleran el ciclo de desarrollo de manera significativa.

C# es un lenguaje con seguridad en el tratamiento de tipos, que es a la vez sencillo y potente y permite a los programadores crear una gran variedad de aplicaciones. Combinado con .NET Framework, Visual C# permite la creación de aplicaciones para Windows, servicios Web, herramientas de base de datos, componentes, controles y mucho más.

Capítulo 4

Trabajo Realizado

4.1. Análisis

La fase inicial en el desarrollo de cualquier sistema software es determinar los requisitos del sistema y así poder conformar un producto que cumpla con las expectativas del cliente.

Existen dos niveles de requisitos, los requisitos de usuario y los requisitos de software. Para llevar a cabo la clasificación de los requisitos se han seguido los estándares propuestos por la ESA[1] [24] y [25] para este fin. Antes de definir los requisitos es necesario limitar el alcance y definir el entorno operacional.

4.1.1. Alcance del Software

El proyecto propuesto pretende realizar la detección de peatones o viandantes en la calzada utilizando fusión sensorial mediante visión artificial y detección con un telémetro láser. Estos dispositivos estarán integrados en un vehículo y se tendrá que poder realizar la detección en estático, sin contar con el movimiento del vehículo. Todo ello deberá ser un sistema de detección en tiempo real y automático por lo que se tendrá que efectuar de una manera sencilla y rápida.

[1]European Space Agency (Agencia Espacial Europea). Su división de software se encarga de regular y estandarizar las metodologías de diseño, ingenierías de requisitos y tecnologías desarrolladas en el espacio europeo.

4.1.2. Entorno operacional

El sistema será integrado dentro de otro sistema mayor llamado POCIMA que se trata de un sistema de transporte inteligente. El objetivo de los Sistemas de Transporte Inteligente es mejorar la seguridad, eficiencia y confort del transporte, mejorando la funcionalidad de los coches y las carreteras usando las tecnologías de la información.

4.1.3. Requisitos de usuario

Los requisitos de usuario definen los deseos y propuestas de los clientes y por tanto de los usuarios finales que harán uso de la aplicación. Dentro de los requisitos de usuario se encuentran otros dos niveles: los requisitos de capacidad y los requisitos de restricción.

Capacidades generales

Los requisitos de capacidad se pueden resumir como la siguiente lista, que incluye básicamente todo lo que se espera que haga el sistema:

- Detección de peatones o viandantes virtuales mediante sensores incluidos en el vehículo en una simulación por ordenador.
- Fusión de los datos obtenidos para fijar una respuesta.
- Clasificación del obstáculo. Respuesta de si es o no un peatón, y a qué distancia se encuentra del vehículo.
- Debe ser una aplicación extensible, por tanto flexible y bien estructurada.
- También debe cumplir que sea robusta ante iluminación o cambios de fondo.
- Y por último, tiene que ser capaz de reconocer peatones de diferentes tamaños, formas, o con ropa distinta.

Restricciones generales

- La detección se hará en un estado estático, es decir, no será posible utilizar el movimiento para realizar el reconocimiento.
- Sólo se tendrán en cuenta los obstáculos situados en el frontal del vehículo.
- La aplicación se tiene que integrar dentro de otro sistema. Por lo tanto, las entradas y salidas deben estar bien definidas y la interfaz de usuario carece de importancia.
- Se debe implementar en MRDS.

4.1.4. Modelo del sistema

Se ha seleccionado la utilización de un plano láser como apoyo a la visión estéreo pasiva. La visión estéreo presenta graves problemas con objetos que presenten colinealidad con la epipolar, en estos casos se presentan indeterminaciones y grandes errores en la etapa de correspondencia. El plano láser permite generar un patrón de intersección claramente diferenciado en ambas cámaras. Por tanto la fusión sensorial se realizará utilizando estas dos técnicas, la visión artificial y el plano láser.

A continuación se presenta el diagrama 4.1 en el que se puede observar el funcionamiento básico del sistema y las entradas y salidas del mismo. Este, es un escenario de fusión competitivo, la información recogida se refiere a lo mismo, pero proveniente de distintos sensores y la fusión se realiza una vez adquirida toda la información.

Se ha optado por un entorno virtual para realizar la ejecución y las pruebas de la aplicación. Esto es debido a que se puede probar en situaciones que comprometerían la funcionalidad de un prototipo; el aprendizaje es rápido y permite modificaciones sobre la marcha; se puede trabajar sobre modelos que todavía no están diseñados completamente y es más económico, porque el coste del hardware final es costoso. Es recomendable realizar primero un prototipo que funcione en un entorno virtual y a continuación adaptarlo para que lo haga también con dispositivos reales y en escenarios reales, ya que en un entorno virtual no existe el ruido y hay que tenerlo en cuenta para su funcionamiento futuro.

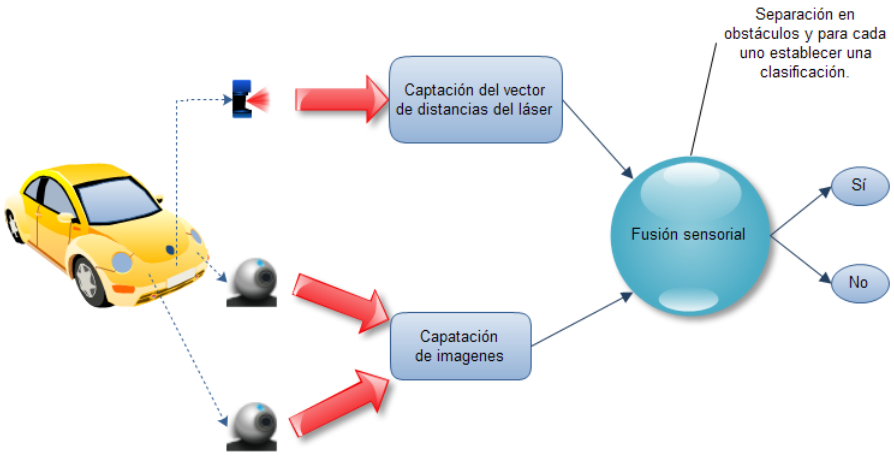


Figura 4.1: Esquema de funcionamiento general del sistema implementado.

4.1.5. Requisitos de Software

Los requisitos de software son los que tienen como fuente el mismo desarrollador del sistema y por lo tanto son mucho más detallados y concretos pero siempre son derivados de los requisitos de usuario o han sido negociados previamente con el cliente.

Esta clase de requisitos se puede dividir en dos tipos, requisitos funcionales y no funcionales, y dentro de este último grupo se incluyen también diferentes tipos.

REQUISITOS FUNCIONALES:

IDENTIFICADOR	RS-001
TÍTULO	Visión estereoscópica
DESCRIPCIÓN	Capacidad de capturar las imágenes de dos cámaras para proveer información en tres dimensiones.
NECESIDAD	Esencial
PRIORIDAD	Alta

IDENTIFICADOR	RS-002
TÍTULO	Sensor láser
DESCRIPCIÓN	Utilización de un telémetro láser para añadir información y ejecutar la fusión sensorial. El sistema debe ser capaz de capturar el haz del sensor láser en un vector de distancias.
NECESIDAD	Esencial
PRIORIDAD	Alta

IDENTIFICADOR	RS-003
TÍTULO	Sincronización de sensores
DESCRIPCIÓN	Se debe esperar a tener todos los datos para iniciar el proceso.
NECESIDAD	Conveniente
PRIORIDAD	Alta

IDENTIFICADOR	RS-004
TÍTULO	Fusión sensorial
DESCRIPCIÓN	Equiparación de obstáculo en el vector de distancias del láser con el mismo obstáculo en la imagen.
NECESIDAD	Esencial
PRIORIDAD	Alta

IDENTIFICADOR	RS-005
TÍTULO	Segmentación y Etiquetado
DESCRIPCIÓN	La información debe ser procesada para separar los posibles obstáculos que se encuentren tanto en las imágenes como en el vector de distancias y diferenciarlos entre ellos.
NECESIDAD	Conveniente
PRIORIDAD	Alta

IDENTIFICADOR	RS-006
TÍTULO	Extracción de características y clasificación
DESCRIPCIÓN	De cada obstáculo se debe extraer la información necesaria para poder clasificarlo como un peatón o cualquier otra cosa.
NECESIDAD	Esencial
PRIORIDAD	Alta

REQUISITOS NO FUNCIONALES:

De consumo de recursos

IDENTIFICADOR	RS-007
TÍTULO	Carga computacional y tiempo limitados
DESCRIPCIÓN	Al tratarse de un sistema en tiempo real es necesario tener en cuenta que la carga computacional debe ser rebajada al máximo para que el tiempo de respuesta sea lo más rápido posible.
NECESIDAD	Conveniente
PRIORIDAD	Media

Fiabilidad y Disponibilidad

IDENTIFICADOR	RS-008
TÍTULO	Baja tasa de fallos
DESCRIPCIÓN	Es necesario que los falsos negativos sean casi nulos ya que se tiene que procesar una alerta lo más fiable posible.
NECESIDAD	Esencial
PRIORIDAD	Alta

IDENTIFICADOR	RS-009
TÍTULO	Funcionamiento continuado
DESCRIPCIÓN	El sistema estará en continuo funcionamiento, analizando la información que llega cada 60 milisegundos.
NECESIDAD	Conveniente
PRIORIDAD	Media

Manejo de errores

IDENTIFICADOR	RS-010
TÍTULO	Captación errónea de imágenes
DESCRIPCIÓN	Si las imágenes no son captadas correctamente saltará una excepción.
NECESIDAD	Conveniente
PRIORIDAD	Media

IDENTIFICADOR	RS-011
TÍTULO	Captación errónea del plano láser
DESCRIPCIÓN	Si el sensor láser no es capaz de transferir la información saltará una excepción.
NECESIDAD	Conveniente
PRIORIDAD	Media

Interfaz

IDENTIFICADOR	RS-012
TÍTULO	Respuesta del clasificador
DESCRIPCIÓN	Al tener el sistema que integrarse en otro, la respuesta o salida final será un simple booleano que indique si se trata o no de un peatón y la distancia a la que se encuentra.
NECESIDAD	Opcional
PRIORIDAD	Baja

IDENTIFICADOR	RS-013
TÍTULO	Imágenes en color
DESCRIPCIÓN	Las imágenes captadas serán en color con la codificación BGR (azul, verde, rojo), dando lugar a un vector en el que cada píxel se define como tres posiciones contiguas, una para cada color.
NECESIDAD	Opcional
PRIORIDAD	Baja

IDENTIFICADOR	RS-014
TÍTULO	Tamaño de la imagen
DESCRIPCIÓN	Las dimensiones de las imágenes, medido en píxeles, serán de 320x220 y el vector que las define tendrá un tamaño de 211200 (320*220*3).
NECESIDAD	Conveniente
PRIORIDAD	Media

IDENTIFICADOR	RS-015
TÍTULO	Amplitud del láser
DESCRIPCIÓN	La cobertura dada por el láser será de una apertura de 180 grados y una distancia máxima de alcance de 8 metros. El vector que contendrá la información será de 360 posiciones, por lo que cada posición corresponde a 0'5º.
NECESIDAD	Conveniente
PRIORIDAD	Media

Restricción

IDENTIFICADOR	RS-016
TÍTULO	Escenario virtual
DESCRIPCIÓN	La simulación y las pruebas se realizarán en un escenario virtual debido a que el costo es menor.
NECESIDAD	Opcional
PRIORIDAD	Media

IDENTIFICADOR	RS-017
TÍTULO	Programación orientada a servicios
DESCRIPCIÓN	Del requisito de usar el programa MRDS, se deriva que se tenga que utilizar una programación orientada a servicios como define dicha aplicación.
NECESIDAD	Esencial
PRIORIDAD	Media

IDENTIFICADOR	RS-018
TÍTULO	Concurrencia asíncrona
DESCRIPCIÓN	Por el mismo motivo que el anterior es inevitable tener que trabajar con concurrencia asíncrona, ya que la aplicación de MRDS lo proporciona por defecto.
NECESIDAD	Esencial
PRIORIDAD	Media

IDENTIFICADOR	RS-019
TÍTULO	Puertos y contratos
DESCRIPCIÓN	Es necesario hacer uso de puertos y contratos en el caso de querer comunicar varios servicios entre ellos, como ocurre en este caso.
NECESIDAD	Esencial
PRIORIDAD	Media

IDENTIFICADOR	RS-020
TÍTULO	Utilización de C#
DESCRIPCIÓN	De los lenguajes de programación permitidos para trabajar con MRDS es el más versátil y apropiado.
NECESIDAD	Conveniente
PRIORIDAD	Media

4.2. Diseño

Después de la especificación de requisitos es necesario realizar un diseño, previo a la implementación propiamente dicha y establecer cómo debe construirse el sistema antes de construirlo.

4.2.1. Arquitectura

La aplicación diseñada se basa en una arquitectura orientada a servicios. Cada componente básico de la aplicación es un servicio DSS de MRDS que se comunican entre sí por medio de mensajes o notificaciones. Un servicio es la unidad de orquestación (entre sensores y actuadores). Por este motivo podría definirse como una arquitectura de sistema de eventos donde un conjunto de componentes, en este caso servicios, esperan a que ocurra un evento que les afecte.

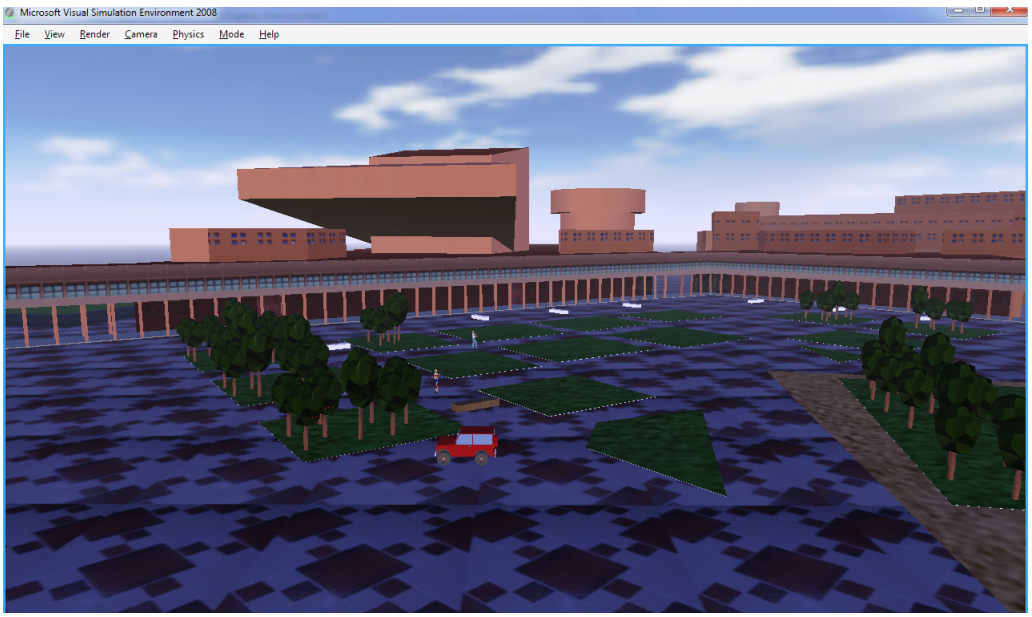


Figura 4.2: Entorno de simulación del campus de Leganés de la uc3m.

La arquitectura definida por MRDS se basa en el paso asíncrono de mensajes. Se focaliza en la coordinación de estos mensajes y oculta las primitivas de concurrencia tradicionales como son los hilos, semáforos, etc. Por lo tanto para modelar el sistema se puede hacer según un modelo secuencial, sin tener que estar

pendiente de la concurrencia. Cada servicio define unos socios o textitpartners que son con los que puede comunicarse y enviar mensajes, por lo que son dependientes de los datos, más concretamente de las entradas. Estos mensajes están tipificados y se insertan en los puertos creados acorde con el tipo específico. Los puertos son colas tipo FIFO que comunican los servicios entre sí. Para cada entrada al servicio se define un árbitro o recibidor que maneja el mensaje de la forma que se desee.

En la aplicación que se diseña, es necesario la creación de un nuevo servicio, que se llamará `fusion` y que se tendrá que comunicar con otro ya creado llamado campus que contiene el entorno de simulación, que es una recreación del campus de Leganés de La Universidad Carlos III de Madrid como se veía en la imagen 4.2.

A continuación se muestra un diagrama de los servicios que interactúan en la escena y como se relacionan unos con otros. El servicio principal es el llamado fusión. Los presentados en color azul simbolizan los servicios creados para este proyecto, los que están dibujados en amarillo son los servicios genéricos para comunicarse con los dispositivos físicos externos y los que aparecen como rectángulos grises representan entidades que se integran en el entorno virtual.

El anterior diagrama presenta un esquema de los servicios involucrados en la aplicación y como se relacionan entre sí. El servicio principal es `fusion`, que soporta la funcionalidad relacionada con la detección y es el que se ha desarrollado para este proyecto. Este está relacionado directamente con el servicio `campus` donde se encuentran todas las entidades del entorno de simulación en el que se interactúa. La relación entre estos dos servicios se realiza a través del manifiesto que se define para cada uno de ellos. El servicio `campus` a su vez también incluye otros socios en el manifiesto, estos son `Dashboard` y `SimulatedEngine`. Estos otros servicios ya estaban implementados anteriormente y han sido utilizados para poder realizar las pruebas en un entorno simulado.

El servicio `fusion` también se relaciona con los servicios genéricos `webcam` y `sickLRF` que definen las funciones y las operaciones básicas de los sensores de webcam y láser respectivamente. Por tanto deben relacionarse o bien con una entidad física, o bien con una entidad virtual, como es en este caso. Esas entidades se denominan robocam1, robocam2 y simulatedlrf. La relación con estas entidades se realiza a través de la creación de puertos para hacer posible la comunicación. Sin embargo la relación con los servicios genéricos se realiza a través de la incorporación de las librerías especificadas en el diagrama: `Microsoft.Robotics.Services.WebCam.Proxy` y `Microsoft.Robotics.Services.Sensors.SickLRF.Proxy`.

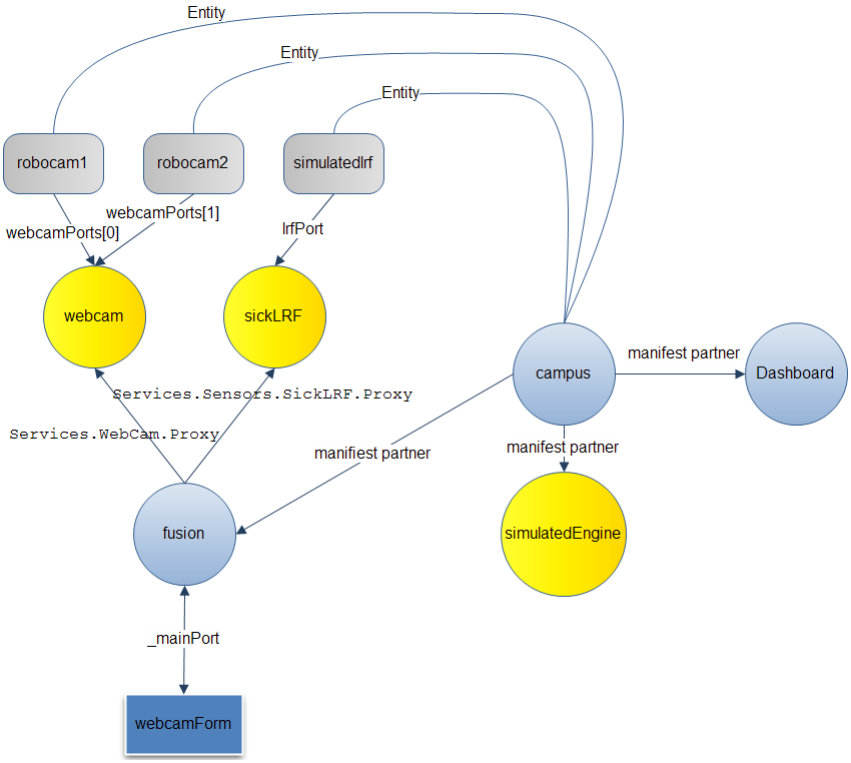


Figura 4.3: Coordinación de los servicios.

Por último, el servicio `fusion` también se relaciona con `webcamForm` que es un panel o formulario que sirve como medio de transmisión de las respuestas del sistema, para mostrar las lecturas de los sensores. Por tanto se comunica con este por medio del puerto principal del servicio, el `_mainPort`.

4.2.2. Diagrama de flujo

La figura 4.4 muestra los procesos que tienen lugar en la ejecución del programa y la transición con la que transcurren. Como se puede ver no existe ningún nodo final. Esto es debido a que se trata de una ejecución continuada, mientras siga existiendo información de entrada se tendrá que seguir generando una salida.

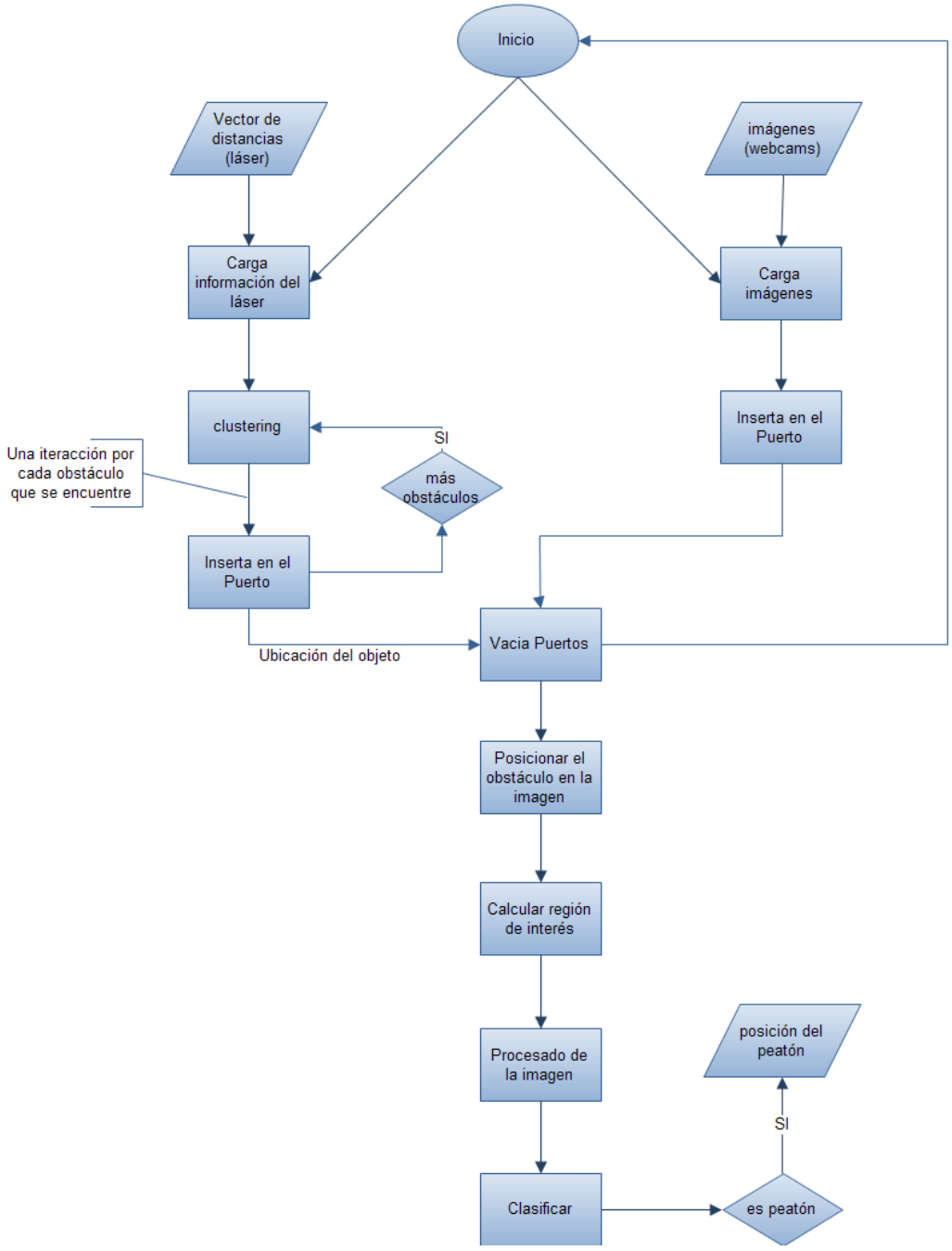


Figura 4.4: Diagrama de flujo.

Al iniciarse el sistema se crean dos hilos uno para cada tipo de sensor que terminarán cuando se reciba en el puerto toda la información recogida por los sensores. A continuación se inician paralelamente dos líneas distintas de ejecución, una que vuelve al inicio para seguir capturando información y la otra para procesar la que se tenía en el puerto. El procesamiento consiste en detectar en la imagen la posición dada por el sensor láser y definir una región de interés. Seguidamente se realizan distintas operaciones en la imagen para obtener las características relevantes del objeto y se clasifica según estas como un peatón u otro objeto. Si el resultado de la clasificación es afirmativo se debe dar una respuesta de cuál es la posición del objeto y la distancia a la que se encuentra.

4.2.3. Entradas y Salidas del sistema

Es muy importante definir en este caso las entradas y salidas, ya que se está hablando de un sistema de control automático donde las entradas corresponden a la percepción sensorial y las salidas son acciones.

En el caso de las entradas se tendrán las dos cámaras Web para la parte de visión artificial y un telémetro láser, que estarán conectados al servicio principal por medio de los servicios genéricos definidos para este tipo de dispositivos. Con este sistema es más factible poder incorporar los dispositivos físicos o cambiar el modelo de los mismos sin que afecte al funcionamiento del sistema.

El servicio SICK Laser Range Finder (telémetro láser) se interconecta con la serie Sick LMS200 de telémetro láser. Éste servicio demuestra cómo interconectar el SICK laser range finder sobre un puerto serie. Se configura el LRF para ejecutar en un modo de monitorización continuo, ejecución de barridos de 180 grados en intervalos de 0.5 grados en el máximo de tarifa de datos disponible. Cada vez que LRF informa de un barrido completo del servicio, el servicio aporta una notificación de reemplazo que contiene los datos LRF.

El servicio WebCam sirve para definir un puerto que comunique con una cámara Web. Si un servicio de cámaras web está disponible en los servicios en ejecución, el servicio WebCam visualizará las imágenes de la cámara de forma automática. Las imágenes obtenidas serán en color codificadas con el sistema BGR (azul, verde y rojo) y de unas dimensiones de 320x220 píxeles. Este servicio implementa funciones como `QueryFrame()` que sirve para obtener la última actualización de la imagen.

Para las salidas, en este caso es una única salida, no se define concretamente ningún comportamiento, ya que no entra dentro del alcance de este proyecto pero según la respuesta ofrecida por el sistema se podría implementar un sistema de frenado automático o algún tipo de alarma que avise al conductor de un posible riesgo. La información que se transmite como salida implica una clasificación de dos clases, peatones o no peatones, además también informa, en el caso positivo, de la distancia a la que se encuentra y la posición dada por el ángulo de barrido del sensor láser.

4.2.4. Justificación de las técnicas escogidas

Lo primero que hay que definir es como realizar la fusión de la información y procesar dicha información en bruto para que se convierta en información relevante para el sistema.

Se ha optado por utilizar una fusión por asociación de información. Si la fusión se hace en un alto nivel puede que en la fase de captación se descarte información necesaria que no es interesante desde un único punto de vista de un sensor (*Early fusion concept*). Sin embargo, la tarea de detectar objetos en las imágenes sin ninguna información a parte de la propia imagen, es un problema difícil de resolver. Por eso, lo que se hace, es recoger la información del sensor láser, interpretarla y buscar la correspondencia con la imagen. Teniendo la posición del posible obstáculo localizado en la imagen es posible descartar parte de la misma quedándose así, únicamente con la región de interés.

Se probó a utilizar una técnica de detección de bordes, con un filtro de Sobel diagonal para poder definir los contornos y las siluetas de los elementos de la imagen. Pero a la hora de segmentar resultaba complicado definir los límites de la figura que representaba al obstáculo porque suponía examinar cada píxel estableciendo que si está dentro del contorno pertenece a la figura. Pero el contorno no tiene porque quedar cerrado y resulta un procedimiento lento y complicado.

En un primer momento se pensó utilizar un mapa de disparidad en el que se separa la imagen en planos según la profundidad a la que se encuentren, como se puede ver en la siguiente figura. Pero supone una carga computacional muy elevada, no recomendable para un sistema como este, que tiene que funcionar en tiempo real. Todo esto surge del problema de la correspondencia de regiones, es decir, encontrar algo característico que se pueda localizar en las dos imágenes. Esto provoca una cantidad considerable de ambigüedades o ruido que es muy difícil

de filtrar, sin contar con el tiempo que eso llevaría. Por todo ello está técnica se descartó y se sustituyó por una mucho más sencilla que aportaba la información necesaria en un periodo de tiempo corto y controlado. Además al estar las cámaras bastante separadas la una de la otra, al realizar el mapa de disparidad, los elementos cercanos al plano de las cámaras se veían duplicados como en el ejemplo que se muestra a continuación.

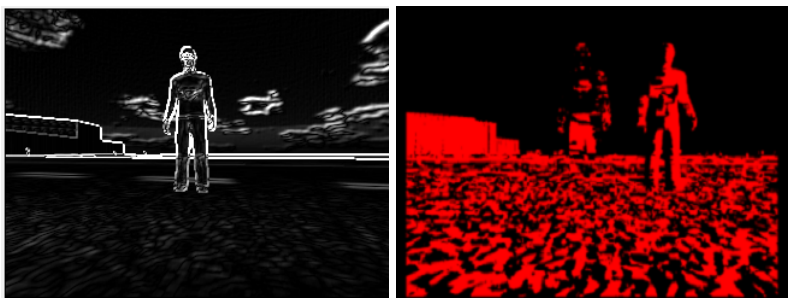


Figura 4.5: Detección de bordes y mapa de disparidad para el caso 1.

La técnica utilizada se basa en los mismos principios de visión estereoscópica que la técnica anterior, pero en lugar de encontrar una correspondencia para cada punto o píxel, lo que se realiza es una sustracción de las imágenes para así descartar todo lo que esté a una distancia suficientemente grande, es decir lo considerado como fondo. Al ser la disparidad en esos puntos nula, se dibujará en la nueva imagen de color negro. Así es posible distinguir fácilmente la figura de lo considerado como fondo.

Aún así, utilizando la técnica antes descrita, sigue existiendo el ruido en la imagen. Para ello se ha tenido que realizar un procesado de la imagen consistente en la utilización de operadores morfológicos. Se ha escogido este tipo de método porque se basa en las formas como conjuntos y no trata la imagen homogéneamente, si no que depende de las formas. En este caso, el ruido se concentra en lo más alejado al vehículo, es decir, el fondo, y lo más cercano, es decir, el obstáculo, está más nítido. Con las trasformaciones morfológicas se puede suavizar los contornos de la figura sin afectar a toda la imagen, como pasaría con un filtro lineal.

Respecto a los atributos elegidos para realizar la clasificación, otra elección importante en este proyecto, se ha optado por algo simple y rápido y así se elimina una gran cantidad de información poco útil para el clasificador. Las proporciones de las personas, aunque se puedan asemejar a otros elementos de la carretera, definen adecuadamente la clasificación. Se discriminan los obstáculos según los

atributos de altura y anchura de la figura. Estas medidas son calculadas en una escala lo más cercana a la realidad para poder diferenciar bien los obstáculos. Si sólo se tuvieran en cuenta las medidas adquiridas de la imagen, el mismo peatón sería distinto a diferentes profundidades lo que llevaría a una clasificación errónea. Se utiliza información adicional, procedente de la base de conocimiento para poder establecer las restricciones de tamaño que discriminan si se trata o no de un peatón.

4.3. Diseño detallado

Para explicar mejor el diseño se describirán paso a paso cada una de las fases que se van produciendo a lo largo de la ejecución del programa. Por orden de realización se describirá como se produce la detección de obstáculos; como se calcula la correspondencia de los datos de los dos sensores, es decir la fusión sensorial; como se define la región de interés; como se procesa la imagen para conseguir los descriptores de cada obstáculo y finalmente, como se clasifica según esa información.

4.3.1. Detección de obstáculos

La información de la que se dispone es procedente de dos tipos de sensores, el telémetro láser y las dos cámaras, dispuestas una a cada lado del primer sensor. Es decir, el sensor láser está situado en el frontal del vehículo, justo en el centro y cada cámara está ubicada a 500 milímetros del mismo. Por tanto hablaremos de cámara derecha e izquierda para diferenciarlas una de la otra.

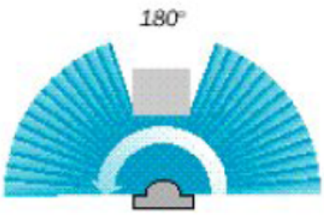


Figura 4.6: Radio de acción del LRF.

La información proveniente del sensor láser se presenta como un vector de

360 posiciones, que corresponde con 0.5º cada posición, por lo tanto el máximo de amplitud medible es de 180 grados. Para cada posición del vector se tendrá la distancia medida hasta el obstáculo, siendo en el caso de no encontrarse ninguno, una distancia de 8000 milímetros, tomadas las medidas de derecha a izquierda, como se observa en la siguiente figura.

Figura 4.7: Dirección de los ejes en la imagen.

Por tanto para adecuar la información del láser con las imágenes es necesario invertir el vector para que las medidas aparezcan de izquierda a derecha, ya que la coordenada x en las imágenes aparece de la misma forma.

La información del telémetro láser, almacenada en el vector de distancias se puede consultar a través de la URL *http://localhost:50000/simulatedlrf*. MRDS aporta un servicio para poder observar la ejecución y el funcionamiento de las aplicaciones a través del navegador Web. Así se pueden gestionar más fácilmente todos los dispositivos activos y la información que transmiten. Posteriormente se muestra un ejemplo de lo que sería el vector de distancias obtenido por el sensor.

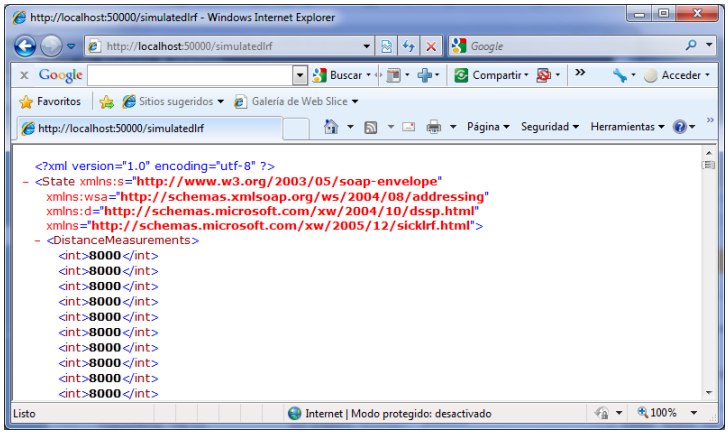


Figura 4.8: Visión del barrido láser mediante la URL.

A continuación es necesario agrupar las medidas obtenidas de tal manera que se separen los obstáculos detectados unos de otros. Por ello se crea una estructura para cada uno de ellos en la que se guarda la siguiente información:

- Posición inicial: primera posición del vector con menos de 8000 mm. A partir del último obstáculo almacenado.
- Longitud: número de posiciones del vector que ocupa el obstáculo incluyendo los espacios entre segmentos.
- Profundidad: distancia mínima encontrada desde el obstáculo hasta el sensor.
- Ancho: cálculo en centímetros de la amplitud del obstáculo a partir del Teorema del coseno.

Para diferenciar un objeto de otro se establece que la diferencia de profundidades sea mayor o igual a 400 mm. y la diferencia en el eje horizontal sea suficiente para denotar objetos diferentes dependiendo de la profundidad del objeto detectado. La separación máxima se ha fijado a partir de una serie de muestras de ejemplo de diferentes escenarios elegidos expresamente para este cometido.

4.3.2. Establecer correspondencia entre sensor láser e imagen

De lo que se trata es de establecer el punto de la imagen donde se sitúa el objeto encontrado como obstáculo en el láser. Para ello, es necesario, conociendo el ángulo donde se sitúa el obstáculo, establecer una correspondencia con el ángulo perteneciente a la imagen. Las cámaras cubren un rango de apertura de 72 grados, por ello, existirán objetos detectados por el sensor láser que no encontrarán la correspondencia en las imágenes y por tanto serán descartados para la consiguiente evaluación. Esto es debido a que si no se pueden ubicar en el rango de las cámaras, se asume que esos obstáculos no están dispuestos en el frontal del vehículo y por tanto no suponen un peligro inminente.

Para establecer la correspondencia entre ángulos se hace uso de los teoremas trigonométricos de seno y del coseno. El ángulo a hallar es el indicado en la figura como β.

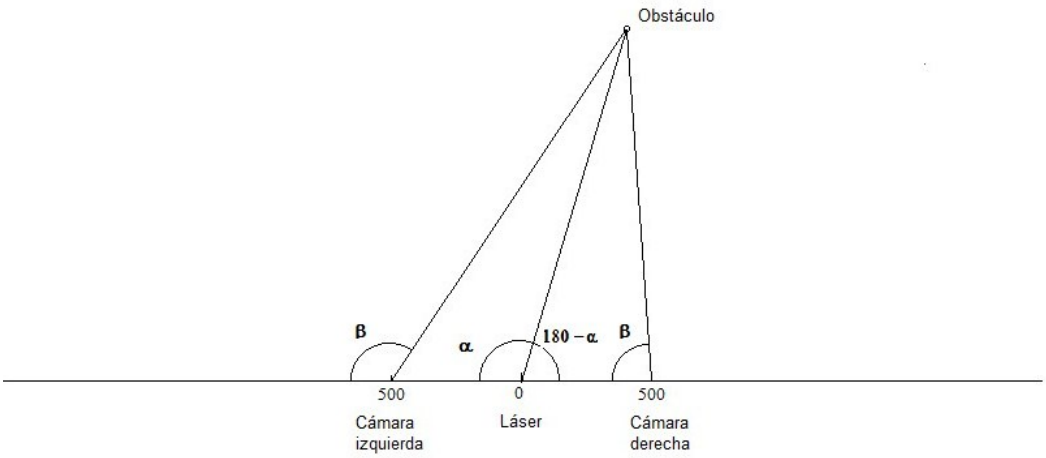


Figura 4.9: Definición de un punto según los 3 sensores.

Pero al tratarse de un rango de amplitud de 72º en el caso de las cámaras el ángulo β se transformará en $\beta - 54$ ya que es el ángulo restante. Si se parte del centro y se dibujan dos ángulos de $36º(72/2)$ a cada lado, el ángulo restante resulta ser $90 - 36 = 54$.

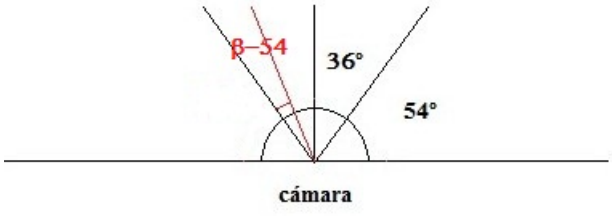


Figura 4.10: Rango de amplitud de las cámaras.

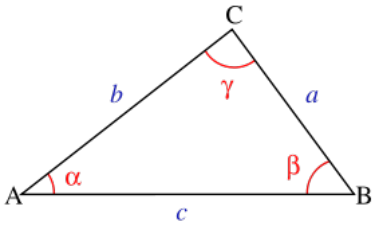
Así se tiene el ángulo dentro del rango de 0 a 72º que corresponde a la proyección de la imagen.

Primeramente se debe calcular el ángulo β, para lo cual se utilizará el triángulo formado por los puntos de posición del láser, posición de la cámara y posición del obstáculo. Conocidas las longitudes de dos de los lados, una es 500 mm. que llamaremos x_{cam} y la otra, que se obtiene del vector de distancias del sensor láser, que llamaremos $dist$; también se conoce el ángulo α que es la posición en el vector de distancias del inicio del obstáculo Por lo tanto se puede calcular el lado y los ángulos restantes.

En primer lugar se halla la longitud del lado que falta, aplicando el Teorema del coseno, que relaciona un lado del triángulo con los otros dos y con el coseno del ángulo formado por estos dos lados [15].

Teorema 1 (Teorema del coseno) *Dado un triángulo ABC, siendo , β, γ, los ángulos, y a, b, c, los lados respectivamente opuestos a estos ángulos entonces:*

$$c^2 = a^2 + b^2 - 2ab\cos(\gamma)$$

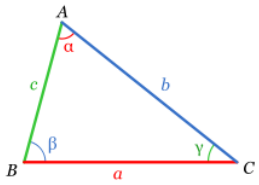


En el caso concreto que se tiene que tratar, la formula quedaría como:

$$d^2 = dist^2 + x_{cam}^2 - 2 * dist * x_{cam} * \cos(\alpha)$$

Teorema 2 (Teorema del seno) *Si en un triángulo ABC, las medidas de los lados opuestos a los ángulos A, B y C son respectivamente a, b, c, entonces:*

$$\frac{a}{\operatorname{sen} A} = \frac{b}{\operatorname{sen} B} = \frac{c}{\operatorname{sen} C}$$



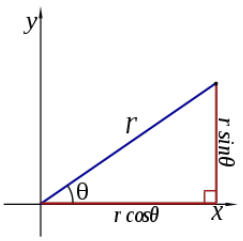
Una vez hallado d se pueden calcular los otros dos ángulos haciendo uso del Teorema del seno que es una relación de proporcionalidad entre las longitudes de los lados de un triángulo y los senos de los ángulos respectivamente opuestos [16].

Con todo ello se consigue el valor de β y de d, que se tendrán que transformar en una coordenada x de la imagen. Se puede realizar una correspondencia directa de coordenadas polares a coordenadas cartesianas.

Teorema 3 (Conversión de coordenadas polares a restangulares) *Definido un punto en coordenadas polares por su ángulo θ sobre el eje x, y su distancia r al centro de coordenadas, se tiene:*

$$x = r * \cos\theta$$

$$y = r * \operatorname{sen}\theta$$



En el caso que nos concierne el eje y corresponde con el eje x y se puede definir dicha coordenada como:

$$x = d * \text{sen}\,\beta$$

Pero el valor de x sería una distancia real y se tiene que hallar el píxel concreto de la imagen por lo tanto se tendrá que aplicar una escala para calcularlo. Como no se tiene ninguna referencia sobre la relación distancia real/distancia en la imagen se calcula mediante triángulos, la equivalencia que existe entre la mitad de la imagen (se conoce que son 160 píxeles) y la distancia real en la profundidad a la que se halla el obstáculo. Así se obtiene la escala válida únicamente para el caso concreto de ese obstáculo. Por tanto, primero se calcula el correspondiente de 160 píxeles con una distancia real en un plano vertical a una distancia dada (la profundidad). Después se establece la escala para ese plano vertical y se calcula el correspondiente para la distancia x.

Como la información trasmitida por el sensor láser es bidimensional, no se dispone de la altura del objeto, por lo que dicho dato deberá ser calculado, utilizando únicamente como recurso las imágenes.

4.3.3. Definir región de interés

Para definir la región de interés (ROI) además de una coordenada x que defina la posición del objeto, es necesario definir una coordenada y, por este motivo, se ha estipulado que sea la mitad de la imagen, ya que se corresponde con el horizonte y el objeto tendrá que estar definido en la intersección con ésta línea. Para asegurarnos de que el obstáculo, si es un peatón, este incluido enteramente dentro de la ROI, se limita la altura como 1250 mm. hacia arriba y 1250 mm. hacia abajo partiendo del horizonte. Este dato se puede considerar información conocida a priori, exterior al sistema, dada por un experto, que conozca las proporciones normales de una persona.

En el caso del ancho no existe ese problema ya que en este caso se conoce la posición de inicio y la longitud del obstáculo medido en grados. Así se puede establecer la anchura en milímetros con el Teorema del coseno de la siguiente manera:

$$2 * ancho = dist1^2 + dist2^2 - 2 * dist1 * dist2 * \cos$$

Siendo $dist1$, la distancia del sensor láser al punto más a la izquierda del objeto, $dist2$ la distancia hasta el punto más a la derecha del objeto y el ángulo medido

entre el punto inicial y final. Con estos datos es suficiente para poder aplicar el Teorema del coseno y hallar el valor real del ancho del objeto. Con esto y con la escala obteniada anteriormente se puede hallar el ancho del objeto medido en píxeles y se puede ajustar la región apropiadamente.

4.3.4. Procesado de la imagen

Una vez que ya se tiene la región de interés que contiene al objeto que se quiere clasificar, lo siguiente que se debe realizar es la extracción de características. Hay que definir correctamente las características o atributos que se quieren utilizar para la clasificación, pues de eso depende que sea más o menos precisa. Según la literatura sobre este tema, para este cometido se pueden utilizar muy diversas opciones, desde medidas básicas como el área a técnicas más complejas como números de forma, las basadas en texturas o los descriptores de *Fourier*.

Al ser este sistema en tiempo real, lo que prima es que los atributos seleccionados para la clasificación sean lo más simples posibles y por lo tanto, los más rápidos de calcular para la aplicación. Por este motivo se ha escogido trabajar con la altura y la anchura, ya que caracteriza suficientemente a las personas para diferenciarlas de, por ejemplo, árboles o señales que son dos de los objetos con forma similar más habituales en la calzada.

En cualquier análisis de imágenes se han de seguir los siguientes pasos: adquisición, preprocesado, segmentación, extracción de características y reconocimiento o clasificación, como se muestra en la imagen. Siempre contando con la información proveniente de la base de conocimiento que aporta una información exterior al sistema.

Como se observa en el diagrama 4.11 cada fase está asociada a un nivel de dificultad que está relacionado con la cantidad de información que le llega de la base de conocimiento; cuanto más alto es el nivel, más importancia tiene el conocimiento exterior.

La fase de adquisición se ha realizado obteniendo la ROI, también parte de la extracción de características ya que para obtener el ancho del objeto se ha tenido que hacer a través de la información del láser en lugar de realizarse en el análisis de la imagen. Ahora se pretende realizar el preprocesado y la segmentación.

Para la segmentación, como se ha explicado anteriormente se han utilizado las

Figura 4.11: Esquema general del análisis de imágenes [10].

técnicas basadas en transformaciones morfológicas. Antes de eso, es preciso, en primer lugar, umbralizar por el color de la carretera aplicando un filtro a la imagen que ponga de color blanco todos los píxeles que tengan un tono grisáceo como el de la calzada; y en segundo lugar, calcular la sustracción de las dos imágenes, la izquierda y la derecha, para así descartar lo considerado como fondo de la escena. Con ello se binariza la imagen coloreando el fondo de negro y el resto de la imagen en color blanco.

Realizada la fase de preprocesado, es momento para comenzar con la fase de segmentación. Para ello se utilizan los operadores morfológicos básicos de dilatación y erosión. Estos operadores se utilizan en distinto orden y con distintos elementos estructurantes según sea la imagen a procesar. Se va a diferenciar unas imágenes de otras según el nivel de blanco que éstas contengan, es decir, según se tenga más fondo o menos. Este valor se define como una constante que puede ser modificada, pero que en este caso se ha decidido que tenga un valor del 60 %. La elección de este valor se justifica empíricamente. Sabiendo los resultados que debe dar el sistema para cada una de las pruebas, se ha visto que para el caso 2 el nivel de blanco obtenido es del 66 % y tiene que ser superado el umbral, así que al ser 66 mayor que 60 es correcto. Sin embargo en el caso 4, para uno de los obstáculos, se tiene un valor del 54 % y no es suficiente para realizar la limpieza más agresiva. Por tanto se establece que el umbral debe estar entre el 0.54 y 0.66,

siendo al final 0.6.

Como se ha explicado anteriormente, esta distinción se ha realizado, teniendo en cuenta los resultados obtenidos al hacer las pruebas. Con las transformaciones morfológicas no existe un patrón de ejecución, los operadores se aplican según el caso de una manera u otra por lo que es importante la experiencia e intuición del desarrollador. En este caso al realizar diferentes pruebas con varios EE se advirtió que se podían distinguir dos casos muy distintos, uno en el que se puede diferenciar casi de inmediato la silueta de la persona y otro en el que además de la persona existen otros objetos en la escena que dificultan su reconocimiento al mezclarse todos ellos en una gran mancha blanca.

Figura 4.12: Binarización de la imagen en dos casos distintos.

En este ejemplo se ve como la imagen de la izquierda correspondería con el primer caso y la de la derecha al segundo. Al tener el edificio de detrás del obstáculo cerca y no poder considerarlo como fondo, en la resta se ve afectada la imagen para el posterior reconocimiento.

Entonces, en el primer caso, los operadores aplicados son menos agresivos y simplemente limpian un poco la imagen para poder distinguir correctamente el peatón. Se utiliza un EE (elemento estructurante) vertical para eliminar las líneas horizontales y uno circular para unir todas las partes en un único conjunto. En el segundo caso se utiliza mayoritariamente la erosión con EE de formas cuadradas para poder eliminar las formas pertenecientes a los edificios u objetos similares.

En el primer caso se aplican unos operadores, según este orden:

1. Erosión con EE vertical de tamaño 5.
2. Dilatación con EE circular de tamaño 5.

3. Y erosión con EE circular de tamaño 5.

Sin embargo para el segundo se han realizado las siguientes transformaciones:

1. Erosión con EE circular de tamaño 5.
2. Dilatación con EE circular de tamaño 5.
3. Erosión con EE cuadrado de tamaño 5.
4. Erosión con EE cuadrado de tamaño 5.
5. Y dilatación con EE circular de tamaño 5.

Figura 4.13: Transformación morfológica en dos casos distintos.

Los resultados parecen confirmar que es posible diferenciar el peatón del resto de la imagen.

4.3.5. Etiquetado

El etiquetado es el proceso de segmentación de imágenes que consiste en dar a cada píxel una etiqueta que representa la pertenencia a un conjunto o agrupación de píxeles. Para este cometido se ha utilizado un algoritmo llamado *Merge Labelling* [17] que consiste en pasar de una imagen binarizada, es decir, en blanco y negro, a otra en la que cada objeto quede separado de los otros dando a cada uno un color diferente. Los objetos se entiende que son las agrupaciones de píxeles en blanco, unidos unos con otros por proximidad de un lado o por las esquinas, es decir, la vecindad se define en 8 direcciones, incluyendo las diagonales.

Este algoritmo tiene dos claras ventajas, es rápido y de complejidad lineal, lo que supone una característica fundamental para un objetivo imprescindible en este proyecto, que se pueda implementar en tiempo real. Pero la desventaja es que objetos que estén unidos por algún píxel los tomará como uno sólo. Como en este caso se saben que los objetos están separados, o por lo menos que se separan en la medida de lo posible, es factible tomar las medidas adecuadas. Por su simplicidad, este algoritmo es perfecto para esta aplicación.

El algoritmo recibe la imagen con los píxeles a etiquetar activos, o sea, en blanco, y se pasa a una matriz donde cada objeto tiene una etiqueta distinta, como se muestra en la siguiente ilustración.

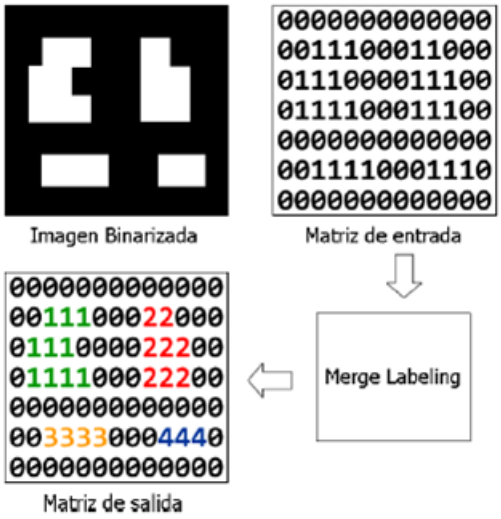


Figura 4.14: Proceso de etiquetación, entradas y salidas del algoritmo [17].

Los pasos a seguir en el algoritmo por cada pixel, por cada fila y columna, son los siguientes:

1. Si el píxel está etiquetado ya:
   a) Si todos los vecinos que están etiquetados es con la misma etiqueta, se marcan los no etiquetados con esa etiqueta.
   b) En caso contrario se saca una lista con todos los vecinos etiquetados y se modifican los que estén erróneos para que tengan todos la misma (merge).
2. Si el píxel no tiene etiqueta:

a) Si todos sus vecinos no están etiquetados o no tiene vecinos, se marca el píxel con el número de la etiqueta correspondiente y se pone la misma a todos los vecinos sin etiquetar. Por último se incrementa el número de etiqueta.

b) Si todos los vecinos etiquetados son diferentes se procede a hacer merge otra vez.

c) Si los vecinos etiquetados son todos iguales se etiqueta al píxel igual y a los no etiquetados también.

La fusión de objetos o merge consiste en que cuando un mismo objeto tiene mezcladas varias etiquetas, ya que la matriz se recorre de manera horizontal, se debe unificar a una sola. Para ello se debe extraer una lista con todos los píxeles incluidos en ese mismo objeto y cambiarlos a la etiqueta menor. Para obtener la lista se almacenan los vecinos de los vecinos de forma no recursiva, sino lineal, hasta que no existan más elementos.

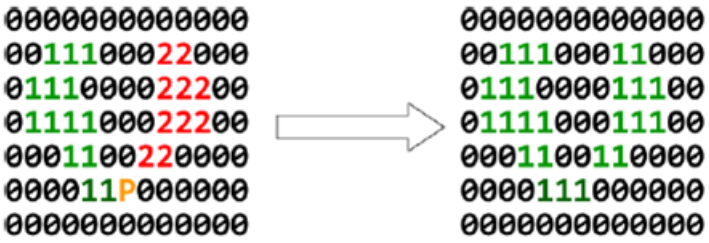


Figura 4.15: Proceso de fusión de etiquetas 1 y 2.

El algoritmo no se ha modificado apenas para integrarlo en el sistema. Simplemente se ha traducido al lenguaje C# y se han introducido las estructuras de datos adecuadas, como las listas de pixeles para almacenar a los vecinos. Además en lugar de tener la imagen únicamente en niveles de gris se tiene en color por lo que por cada píxel se tienen 3 posiciones. Con esto es posible implementar una solución con colores, pero supone mayor complicación en el momento de evaluar un píxel. El resultado obtenido se puede comprobar en la figura 4.16.

Ahora para identificar el peatón lo que se hará es definir para cada uno de los objetos el número de pixeles que ocupa y la altura. Para calcular la altura se almacenan el primer píxel perteneciente al objeto empezando por arriba y el último y se restan. Con esto, el objeto identificado como peatón será el de mayor tamaño y se tendrá también su altura para poder clasificarlo.

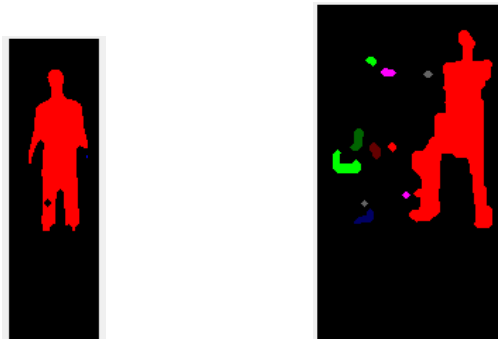


Figura 4.16: Etiquetado en dos casos distintos.

4.3.6. Clasificación

La clasificación, para el caso específico de análisis de imágenes, es la fase en la que se reconoce o identifica cada objeto de la escena. Tras los procesos de segmentación, extracción de características y etiquetado, cada objeto queda representado por una colección o agrupación de descriptores, denominada patrón. Una clase o categoría es una familia de patrones que comparte alguna propiedad en común. Para el reconocimiento automático es importante que los patrones pertenecientes a la misma clase tengan características similares y los que sean de clases diferentes presenten características diferenciadas. Cada patrón se supondrá perteneciente a una clase y sólo a una.

Para este proyecto se han diferenciado dos clases que son peatón u otro obstáculo que se definirán como (p, o) y la característica discriminante en este caso es el tamaño del objeto. Para ello se calculan la anchura y la altura del obstáculo para diferenciar unos patrones de otros. Los patrones son bidimensionales $x = (a, h)$, donde a define la anchura y h define la altura.

Existe una problemática con la elección de características o atributos y es que definan correctamente la pertenencia única a una de las clases. Al ser exclusivamente dos clases no son necesarios patrones demasiado detallados. También hay que tener en cuenta que este proyecto se ejecuta en tiempo real por lo que se debe elegir atributos fáciles de calcular que no consuman demasiado tiempo ni tengan demasiada carga computacional. Tienen que ser suficientes para definir bien el nivel de separación entre clases.

Existen dos tipos de clasificación, las realizadas con métodos supervisados y las realizadas con métodos no supervisados. En este caso concreto se trata de una

clasificación con supervisión ya que se conoce el número de clases y las características asociadas a cada una. Si no fuese así sería necesario establecer, según las muestras obtenidas, el número de clases y las funciones discriminantes.

Al ser clasificación supervisada, supone la utilización de la información de la base de conocimiento en gran medida. Por ello se ha establecido que la altura normal de una persona está entre 800 y 2400 milímetros, basándose en estadísticas, es decir, información exterior al sistema.

La medida de la altura calculada por la aplicación no es demasiado precisa y hay que tener en cuenta que al hacer el cambio de la escala de píxeles a milímetros se toman como referencia las medidas tomadas en la horizontal, por lo que, al ser la imagen más ancha que alta y cambiar la escala con esta referencia, la medida que sale es algo menor que el dato real. La media de estatura en España en el 2001 era de 1665 mm. para una población de edad mayor de 15 años [18]. Pero al tener el posible error de cálculo implícito en la cifra obtenida por el sistema, además de tener que incluir en la clasificación a los niños supone que el rango que establece el siguiente gráfico no sea suficiente y se amplíe, quedando en 800mm. más y menos que la media.

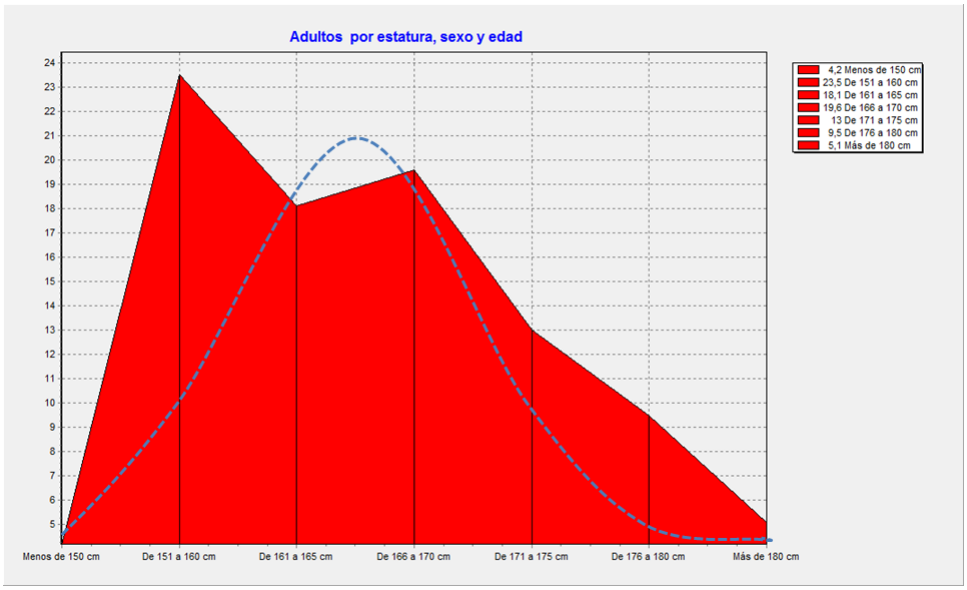


Figura 4.17: Gráfica de estatura de adultos mayores de 15 años en la UE.

El gráfico muestra el número de personas (eje vertical) que hay con cierta estatura (eje horizontal) .Los datos han sido obtenidos del instituto nacional de estadística [19]. Se puede observar que entre 150 y 160 centímetros está la gran

mayoría. Pero la altura es un dato que cumple perfectamente con la distribución normal, con lo que se podría definir de una forma genérica con una campana de gauss como se observa en el gráfico. Si se utilizara para la clasificación un sistema basado en lógica difusa sería muy útil contar con el conjunto representado por la campana de gauss para definir la pertenencia mayor o menor a la clase peatón según la altura.

La anchura biacrominal de una persona, es decir la distancia de hombro a hombro, oscila entre 320 y 360 mm. según la tabla de medidas antropométricas de [20]. Pero al igual que con el caso de la estatura, puede haber errores de cálculo además de que el peatón puede encontrarse de perfil o caminando y la medida tomada puede variar considerablemente. Por este motivo se ha establecido que el rango de valores válidos para un peatón oscile entre 200 y 600 mm. Las restricciones de tamaño finalmente se definen de la siguiente forma, tomando h como la altura y w como la anchura:

$$h = [800, 2400]mm.$$

$$w = [200, 600]mm.$$

En la gráfica 4.18 se puede observar la distribución de los patrones estudiados en la fase de experimentación. Como se puede apreciar, las clases están distribuidas de tal forma que los obstáculos clasificados como peatones se ubican en la parte superior derecha, mientras que la otra clase se posiciona en la parte inferior.

Los datos utilizados para la realización de la gráfica son los que se muestran en la siguiente tabla.

obstáculo	alto (mm)	ancho (mm)
caso1	1328	367
caso3	1141	520
caso4 (1)	970	276
caso4 (2)	1293	258
caso2 (1)	359	270
caso2 (2)	0	142
caso2 (3)	541	209
caso5	1608	113

Tabla 4.1: Patrones de los obstáculos encontrados

Con esta distribución de los patrones se puede establecer una función discriminante que clasifique a cada uno dentro de una de las dos clases. La función que

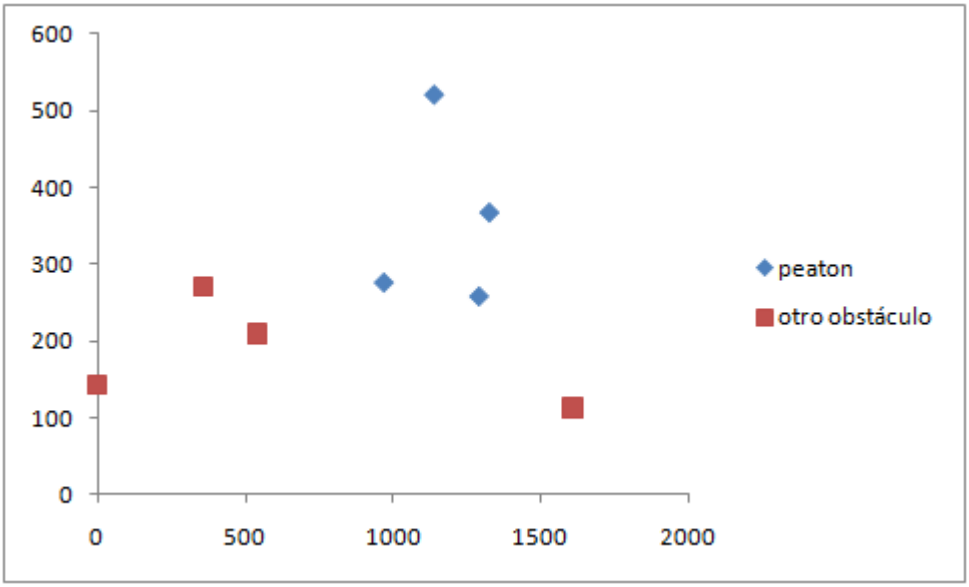


Figura 4.18: Gráfica de la distribución de los obstáculos encontrados.

se ha elegido es que si el patrón está dentro de los rangos de estatura y anchura explicados anteriormente será clasificado como peatón y en caso contrario como otro tipo de obstáculo.

4.4. Implementación

El servicio creado para la aplicación contiene una serie de elementos o estructuras clave para su utilización. MRDS define que un servicio puede contener puertos, manejadores y árbitros para controlar la línea de ejecución. A continuación se explican los elementos usados por el servicio `fusion` que es el que contiene la mayor funcionalidad del sistema.

4.4.1. Puertos

Un elemento esencial de todo servicio es su puerto principal. Este puerto es un mecanismo CCR que implementa una cola FIFO (*First-in, First-out*) de mensajes. A este puerto llegan los mensajes que contienen las peticiones, encapsuladas dentro de determinadas clases, que provienen desde otros servicios asociados, y

las respuestas generadas por el servicio en contestación a dichas peticiones. Este mecanismo se encuentra implementado en la clase genérica Port. Una instancia de esta clase solo puede recibir mensajes de un determinado tipo, que es especificado durante su instanciación.

Los puertos incluidos son los siguientes:

- `_mainPort`: El puerto principal. Se utiliza para comunicarse con el panel que muestra las imágenes y la información del láser.
- `webcamPorts[]`: Este es un vector que contiene los dos puertos para comunicarse con las cámaras Web. Por tanto, webcamPorts[0] corresponde con la cámara derecha y webcamPorts[1] con la cámara izquierda y se conectan directamente con los dispositivos virtuales definidos en campus, robocam1 y robocam2.
- `lrfPort`: Este es el puerto que conecta con el dispositivo virtual que define el sensor láser y que se denomina simularedlrf en el servicio campus.
- `_myPortimg`: Este puerto se ha creado para agrupar la información adquirida por las cámaras Web y mandarlo en el mismo servicio para que se ejecute el manejador correspondiente. Por tanto el tipo de datos que transfiere es una estructura que contiene las dos imágenes y el tamaño de las mismas.
- `_myPortlrf`: Este puerto se ha creado con el objetivo de enviar la información relevante de los obstáculos identificados por el telémetro láser. El tipo de datos que transfiere es una estructura que contiene un identificador, la primera posición del vector de distancias donde comienza el obstáculo, la longitud con respecto al vector, la profundidad a la que se encuentra y el ancho medido en milímetros.
- `resultPort`: Con este puerto se consigue coordinar la recepción de la información procedente de los sensores de tal forma que cuando se reciba toda la información completa se pueda ejecutar de nuevo la actualización de la misma.
- `_dateTimePort`: Este puerto gestiona el tiempo, para controlar que un manejador se ejecute cada cierto tiempo, evitando que se ejecute continuamente pudiendo producir un bloqueo en el sistema.

4.4.2. Manejadores

El servicio implementado contiene distintos manejadores que permiten procesar la información contenida en los mensajes que se reciben por los diferentes puertos, ya sea el puerto principal o los puertos de comunicación con los servicios asociados.

Los manejadores, o *handlers*, incluidos son los siguientes:

- `UpdateSensorData`: Actualiza la información proveniente de los sensores. Activa el árbitro que ejecuta la función de reconocimiento `findROI` y se llama recursivamente a sí mismo para que el bucle de ejecución sea infinito.
- `UpdateLRF`: Actualiza la información del vector de distancias. Pinta en el panel la información y llama al método `splitInObjects` que divide o identifica el vector en objetos.
- `UpdateImage`: Actualiza las imágenes y las envía por el Puerto `_myPortimg` para proceder a la identificación de los posibles obstáculos.
- `findROI`: Es el manejador principal. Una vez se tenga toda la información se debe realizar la fusión y determinar si el obstáculo encontrado puede ser o no clasificado como un peatón.

4.4.3. Árbitros

En una aplicación MRDS, el envío y recepción de mensajes entre los distintos servicios se realiza de forma asíncrona, por lo que CCR permite declarar árbitros, denominados *Arbiters*, que permiten continuar con la ejecución de otras partes de código mientras se espera a la llegada de mensajes a través de algún puerto.

Estos árbitros coordinan una serie de receptores encargados, a su vez, de observar un determinado puerto a la espera de la llegada de un determinado tipo de mensaje. Tras la llegada de un mensaje, el receptor adecuado para ese tipo de mensaje lo captura y el árbitro indica que acciones se deben llevar a cabo. Un árbitro puede invocar distintos manejadores dependiendo del tipo del mensaje que llegue al puerto sobre el que actúa.

Los árbitros utilizados para crear la aplicación son los siguientes:

```
Activate(Arbiter.ReceiveWithIterator(false, _dateTimePort,
UpdateSensorData));
```

Este árbitro utiliza un iterador para ejecutar el procedimiento manejador. El receptor asociado escucha el puerto `_dateTimePort`, a través del cual llegaran las notificaciones de tiempo. Cuando estas notificaciones sean recibidas, el manejador UpdateSensorData ejecuta las acciones correspondientes. El primer parámetro, false, indica la persistencia del receptor. Sólo se mantendrá disponible para recibir un único mensaje y después, se deshabilitaría automáticamente.

```
Activate(Arbiter.JoinedReceive<images,obstacle>(false,
_myPortimg, _myPortlrf, findROI));
```

Con este árbitro se aprovecha la concurrencia dada por el CCR. Se han añadido dos puertos, `_myPortlrf` que recibe uno a uno los obstáculos encontrados con el sensor láser y el otro `_myPortimg` que transfiere las imágenes. Con esto se crea un árbitro JoinReceiver que lo que hace es esperar a que haya un elemento en cada puerto y ejecuta el manejador utilizando la información recibida. El manejador es `findROI`, un método para encontrar las regiones de interés. El primer término esta puesto como false, porque sólo se ejecutará una vez por cada envío recibido.

```
Activate(Arbiter.MultipleItemReceive(false, resultPort,
2, allComplete =>
     {Activate(Arbiter.ReceiveWithIterator(false,
       _dateTimePort, UpdateSensorData));
     TaskQueue.EnqueueTimer( TimeSpan.FromMilliseconds(60),
       _dateTimePort); }));
```

Este árbitro consiste en que se debe esperar a que en el puerto `resultPort` existan dos elementos y estén completos. Entonces se ejecuta el bloque que aparece entre corchetes. Este bloque contiene la sentencia que ejecuta el primer árbitro explicado y después una sentencia para que cambie el estado del puerto `_dateTimePort`. Cada 60 milisegundos se envía una señal.

```
yield return Arbiter.Choice(webcamPorts[0].QueryFrame(),
success =>
         {rgbData1 = success.Frame;
          size = success.Size;},
failure =>
         {LogError(failure.ToException());});
```

Este árbitro se repite en el código, ya que existe uno para la imagen derecha, con `webcamPorts[0]` y `rgbData1` y otro para la imagen izquierda con `webcamPorts[1]` y `rgbData2`. Consiste en que, una vez se ejecuta la sentencia `QueryFrame()` que obtiene la imagen correspondiente a la cámara web, la respuesta obtenida puede ser satisfactoria o fallida. Según devuelva la sentencia se ejecuta un código u otro.

4.4.4. Flujo de ejecución

1. Inicio del servicio y creación del panel
2. Se presiona el botón de Inicio y se ejecuta el manejador `Start`
3. Se ejecuta el manejador `UpdateSensorData`
4. Se ejecuta el manejador `UpdateLRF` [Actualiza la información del vector de distancias, la muestra en el panel y divide el vector en posibles obstáculos, almacenando los datos necesarios y enviando cada uno a través del puerto `_myPorlrf`]
5. Se ejecuta el manejador `UpdateImage` [Renueva las imágenes y las agrupa en una estructura, junto con el tamaño, para mandarlo a través del puerto `_myPortimg`]
6. Si se reciben las dos imágenes y el obstáculo identificado con el láser en los puertos correspondientes se ejecuta el manejador `findROI`
   a) Si el objeto se puede localizar en la imagen derecha, se calcula el ancho y el alto del mismo y se clasifica. Si es un peatón se muestra la información relevante por pantalla.
   b) Si el objeto no se puede identificar en la imagen derecha se prueba con la izquierda y se procede igual que con la derecha.

c) Si no se puede localizar en ninguna de las dos imágenes se muestra un mensaje por pantalla notificando que el obstáculo está fuera del alcance del sistema.

7. Si se tienen todos los datos actualizados se ejecuta de nuevo `UpdateSensorData`

Capítulo 5

Experimentación

En este apartado se describen detalladamente los experimentos o pruebas realizados con la finalidad de comprobar si los algoritmos propuestos en la aplicación desarrollada son capaces de cumplir con el objetivo propuesto.

Para ello se explica primero en que consiste cada paso de la ejecución y posteriormente se detalla con cada caso, los resultados obtenidos en cada uno de los pasos.

El flujo de operaciones, paso a paso, es como sigue:

1. Captura de imágenes izquierda y derecha y la información del láser.
2. Evaluar si es posible posicionar el obstáculo en alguna de las dos imágenes. Primero se intenta con la imagen derecha, si no se puede, se intenta con la imagen izquierda y si tampoco es posible, se descarta el objeto, ya que no se halla exactamente en el frontal del vehículo.
3. Se busca la posición correspondiente del láser con la imagen.
4. Se representa en la imagen un rectángulo de bordes rojos para delimitar la ROI, que tiene como punto central la posición del objeto calculada anteriormente y un ancho igual al doble del ancho calculado con los datos del laser y un alto de 2500mm.
5. Se obtiene una nueva imagen con las dimensiones del recuadro para la imagen derecha y para la imagen izquierda.

6. A continuación se extrae el color de la carretera para evitar ruido, en las dos imágenes izquierda y derecha. Al realizar esta acción se consigue que en el siguiente paso, donde se restan las imágenes para quitar el fondo, se pueda definir la calzada como parte del fondo, no siendo así. Al unificar el color de toda la superficie la sustracción de las dos imágenes resulta nula, y por tanto se considera fondo.

7. Seguidamente se procede a la resta de las dos imágenes para poder determinar los objetos que sean cercanos. Esto es debido a que si está muy lejos los píxeles son iguales y al realizar la resta aparece en negro. El resto de la imagen aparece en blanco y así se consigue binarizar la imagen para su posterior procesado.

8. Teniendo la imagen en blanco y negro se pueden utilizar los operadores morfológicos de erosión y dilatación que según resulte la imagen se hará de una forma o de otra. Si existe una mayoría de píxeles blancos, se efectuará más operadores de erosión para poder limpiar la imagen y sacar la silueta del obstáculo.

9. Por último se procede a realizar la segmentación o etiquetado, dibujando cada objeto identificado como diferente de un color distinto.

10. Conociendo el tamaño de cada objeto se puede obtener la altura y descartar en caso de que no cumpla las restricciones impuestas para reconocerlo como peatón. Para determinar el objeto principal, se realiza a través del tamaño: el de mayores dimensiones se considerará el objeto principal y por tanto el que se va a evaluar.

A continuación se muestran una serie de imágenes regogidas con el programa para ejemplificar los resultados obtenidos con el método descrito. Para cada caso se explicará cada uno de los pasos antes descritos apoyado todo ello, gráficamente, con capturas de pantalla.

5.1. Caso 1

Este primer caso de prueba presenta un peatón situado en el frontal del vehículo, lo que sería el caso más común y más simple y como se muestra más adelante mediante el algoritmo implementado se consigue encuadrar perfectamente al peatón dentro de la ROI y lo identifica sin inconvenientes.

1. Captura de información. Se presenta la muestra obtenida del telémetro láser y las dos imágenes correspondientes a las cámaras web, la de más a la izquierda corresponde con la cámara izquierda y la derecha con la derecha.

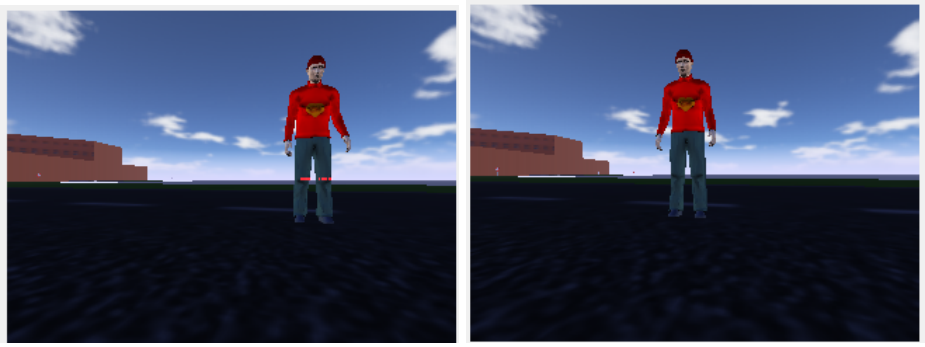


Figura 5.1: Vistas de las dos cámaras web del caso 1.

La gráfica siguiente expone en el eje vertical la distancia o profundidad medida en milímetros y en el eje horizontal los grados del recorrido del haz. Se puede ver claramente como sobresalen dos segmentos que corresponden con las piernas del peatón visto en la imagen. Es como ver al peatón desde arriba y el barrido del láser choca directamente con las piernas de éste.

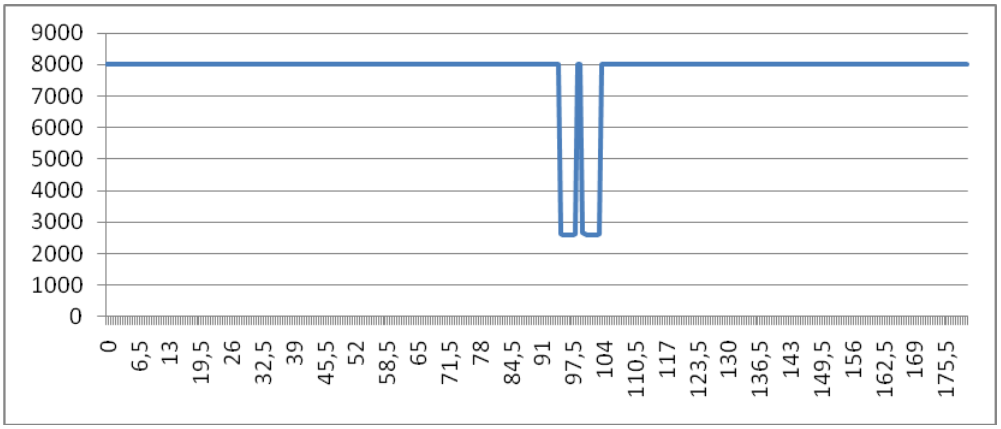


Figura 5.2: Gráfico del vector de distancias del caso 1.

2. Es posible posicionar el obstáculo en la imagen derecha. Como se puede ver en la gráfica, la posición del obstáculo dada por el telémetro láser está entre 95º y 103'5º, que se puede ubicar dentro del rango comprendido por la cámara derecha.

3. En este caso la posición del láser es 95º con una profundidad de 2575mm. y una amplitud de 8'5º que da una posición $x = 151$ e $y = 120$.

4. El rectángulo para definir la ROI tiene como punto central el mencionado en el punto anterior, un ancho igual a 64 píxeles y un alto de 216 píxeles.

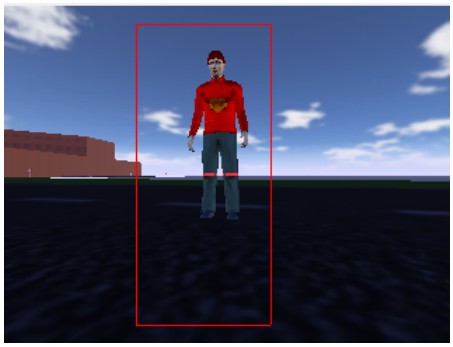


Figura 5.3: Vista del rectángulo que define la ROI en caso 1.

5. Se obtiene una nueva imagen con las dimensiones del recuadro para la imagen derecha y para la imagen izquierda. La imagen izquierda no incluye al peatón, lo que hará más fácil la distinción de lo que es fondo de lo que no.

Figura 5.4: ROI derecha e izquierda para el caso 1.

6. Al extraer los píxeles del color de la calzada, la imagen queda como se puede ver a continuación. El resultado es bastante bueno, porque a pesar de que hay algunos píxeles que no corresponden con la calzada que los colorea de blanco, no afecta al siguiente paso y la calzada queda totalmente identificada y reconocida como fondo.

Figura 5.5: Extracción de carretera en caso 1.

7. Al realizar la sustracción de las dos imágenes el resultado es bastante satisfactorio. En este escenario el fondo queda claramente separado del obstáculo con lo que se consigue fácilmente identificar la silueta del mismo. Lo único que queda hacer, para definir totalmente la silueta es limpiar la imagen eliminando las líneas correspondientes con el horizonte que se realiza con los operadores morfológicos.

Figura 5.6: Sustracción de imágenes en caso 1.

8. Para limpiar la imagen se procede a utilizar operadores morfológicos. Se comienza por una erosión con elemento estructurante vertical de tamaño 5, luego una dilatación con elemento estructurante circular de tamaño 5 también y erosión con el mismo EE. Es decir de los dos flujos de ejecución, se procesa el que establece una eliminación de ruido menos agresiva, ya que el nivel de blanco en la imagen es escaso. Se consigue unificar la figura pero se pierden detalles como las manos y los pies.

Figura 5.7: Transformación morfológica en el caso 1.

9. Al final se consigue que en la imagen se puedan identificar dos elementos únicamente. El objeto pintado de color rojo que representa a la persona, al ser claramente el de mayor tamaño, se elige como el representante del obstáculo y se calcula su altura.

Figura 5.8: Etiquetación en el caso 1.

10. La altura calculada, en número de píxeles es de 115, que pasándolo a un valor real, con la escala calculada anteriormente se transforma en 1328 mm. Teniendo en cuenta que el ancho calculado es de 367 mm. se cumplen las restricciones de tamaño, por lo que se puede afirmar que el obstáculo identificado es un peatón que se encuentra en una posición central tendiendo a la derecha, a 95º, desde el frontal y a una distancia de 2575 mm.

5.1.1. Análisis de resultados

Como se ha visto, este caso se ha resuelto satisfactoriamente. Se puede destacar que la altura calculada es menor de lo que debería ser. Esto es debido a la escala con la que se ajusta el valor en píxeles a un valor real. Esta escala se calcula según el ancho de la imagen, que es mayor que el alto. Por tanto al utilizar la

misma escala para ajustar los valores horizontales y verticales se obtienen valores diferentes.

5.2. caso 2

Este escenario muestra el caso en el que el sensor láser detecta cinco objetos pero únicamente tres pueden ubicarse en el rango de apertura de las cámaras. Los objetos identificados son árboles que detecta como objetos separados y la clasificación se hace correctamente. También hay que tener en cuenta que el coche no se encuentra sobre la carretera, para demostrar que la clasficación también se puede llevar a cabo sin filtrar por el color de la calzada.

La dificultad de este caso radica en la separación que existe entre árboles, ya que se podría identificar como un único obstáculo dos árboles muy juntos.

1. A continuación se muestran las imágenes recogidas por las cámaras web y el vector de distancias.

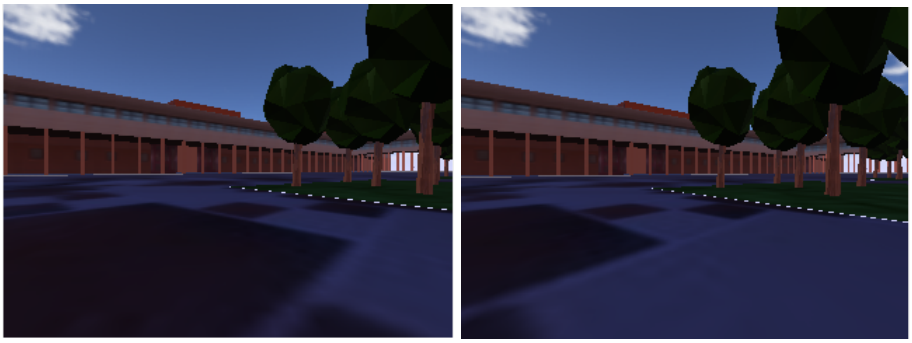


Figura 5.9: Vistas de las dos cámaras web del caso 2.

   En la gráfica 5.10 se pueden identificar claramente los cinco obstáculos. Tres de ellos, los tres primeros, empezando a contar por la izquierda, pueden posicionarse dentro de la imagen, no siendo así para los otros dos que están en un ángulo de 144,5º y 157'5º.

2. Es posible posicionar los tres primeros obstáculos en la imagen derecha. El primero se sitúa en 102º, el segundo en 119º y el tercero en 125'5º.

3. Para cada árbol, se calculan la posición y la distancia desde el vehículo. El primero en 102º está a una distancia de 7377 mm. con una longitud de 2º

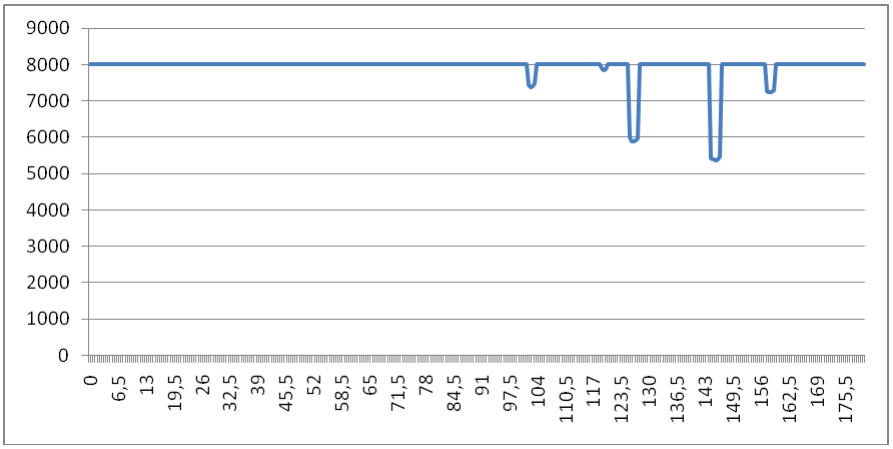


Figura 5.10: Gráfico del vector de distancias del caso 2.

que se traduce a una posición en el eje horizontal de 195 píxeles y el ancho calculado es de 270 mm. El segundo árbol en 119º está a una distancia de 1850 mm. con una amplitud de 1'5º, con lo que la posición x es 266 píxeles y el ancho obtenido es de 142 mm. Y el tercer árbol en 125'5º está a una distancia de 5896 mm., un largo de 2'5º, lo que se transformará en una posición x de 296 píxeles y un ancho igual a 209 mm. Todos los obstáculos tienen la misma posición y de 120 píxeles ya que coincide con la mitad de la imagen.

4. Para cada árbol de los tres posicionados en la imagen se define un rectángulo. Para el primero las dimensiones del rectángulo son 20x115 píxeles, para el segundo de 8x80 píxeles y para el tercero de 16x76 píxeles.

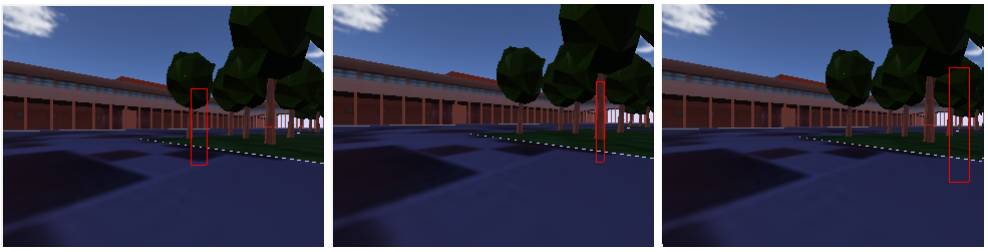


Figura 5.11: Vistas del rectángulo que define la ROI en caso 2.

5. Teniendo las dimensiones del recuadro se recorta en la imagen derecha y en la izquierda para los tres obstáculos.

Figura 5.12: ROI derecha e izquierda para el caso 2.

6. No se pueden extraer los píxeles del color de la calzada y la imagen queda como se puede ver a continuación. En este escenario, el vehículo no está sobre la calzada. Por este motivo la extracción de la misma (de los píxeles de ese color) no se hace patente, se cambian ciertos píxeles pero resulta despreciable. En este ejemplo no hubiese sido necesario realizar este paso, pero tampoco afecta a los siguientes.

Figura 5.13: Extracción de carretera en caso 2.

7. La sustracción de las dos imágenes en este caso no resulta beneficioso. Al estar los objetos bastante alejados y definirse una ROI de pequeñas dimensiones, es muy difícil diferenciar el fondo y poder extraerlo de la imagen. Así resulta un conjunto de tres imágenes bastante confusas.

Figura 5.14: Sustracción de imágenes en caso 2.

8. En estos tres casos, el nivel de píxeles blancos es bastante superior a la media, por lo que se recurre a la ejecución de los operadores morfológicos

correspondientes, quedando las figuras siguientes. Se produce una erosión más agresiva y el resultado es, en el segundo obstáculo, que se eliminen todos los píxeles blancos, quedando la figura completamente negra y sin ningún objeto que identificar. En los otros dos se identifican varias figuras sin forma concreta.

Figura 5.15: Transformación morfológica en el caso 2.

9. En el momento de etiquetar, en la primera figura se identifican dos formas independientes, en la segunda nada, como se vio en el paso anterior y en la tercera se pueden reconocer 5 formas distintas.

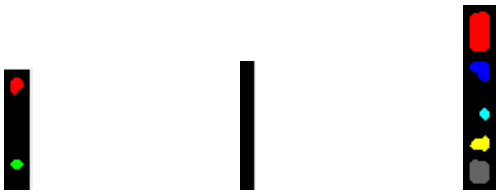


Figura 5.16: Etiquetación en el caso 2.

10. Para cada obstáculo se calcula la altura de la figura mayor, en el primero con un resultado de 11 píxeles que pasados a la escala real se obtiene un valor de 359 mm. En el segundo la imagen es totalmente negra, no hay ninguna figura mayor, por tanto el valor de la altura es 0. Para el tercer caso, se obtiene la altura de la forma roja, con un resultado de 25 píxeles, que haciendo la escala se obtiene una altura de 541 mm. Teniendo la altura y el ancho calculado anteriormente se puede clasificar cada obstáculo. El resultado es, en los tres casos, que no se trata de peatones.

5.2.1. Análisis de resultados

Al no tratarse de personas, el ancho del obstáculo no es suficientemente grande y ocurre que se identifique una ROI pequeña y con ello la imposibilidad de

descartar bien el fondo de la escena. Aún presentándose estas dificultades se puede obtener una clasificación satisfactoria dado que las medidas obtenidas para los tres obstáculos no cumplen con las restricciones de tamaño que discriminan en una clase u otra.

5.3. Caso 3

El siguiente, es un caso en el que el peaton se muestra caminando y por tanto las piernas están bastantes separadas, lo que podría llevar a que se detectara la persona como dos obstáculos diferentes, en lugar de uno sólo. Además se sitúa en la esquina y al tener que definir la región de interés, se saldría de los límites de la imagen. Esto se ha corregido haciendo que si es más grande que la imagén se ciña a los límites de esta. Otra característica a destacar en este escenario es un edificio detrás del obstáculo, suficientemente cercano al vehículo de tal manera que no se puede descartar como fondo.

1. A continuación se muestran las imágenes 5.17 recogidas por las cámaras web y 5.10 el vector de distancias.

Figura 5.17: Vistas de las dos cámaras web del caso 3.

2. En la gráfica 5.18 se ven los dos segmentos que representan las piernas del peatón. Como se puede observar están bastante cercanos al vehículo y situados en 72º y 82º. La imagen que se selecciona es la derecha.

3. Se consigue catalogar como un único objeto los dos segmentos del vector de distancias y se establece que la distancia a la que está es de 1817mm. y la amplitud es de 14º. La posición horizontal se define como 55 píxeles.

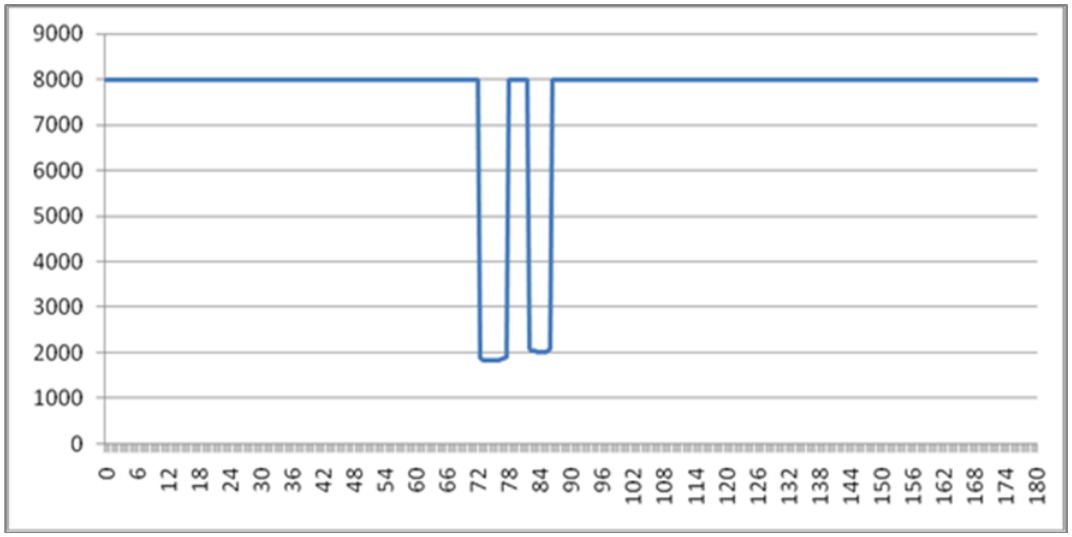


Figura 5.18: Gráfico del vector de distancias del caso 3.

4. El rectángulo definido para encuadrar al obstáculo si se toman las medidas del ancho y la altura establecidas se saldría de los bordes de la imagen. En este caso lo que se ha hecho es definir el recuadro de tal forma que no se salga de los límites de la imagen ya que no tiene sentido. Al final las dimensiones son de 128x240 píxeles.

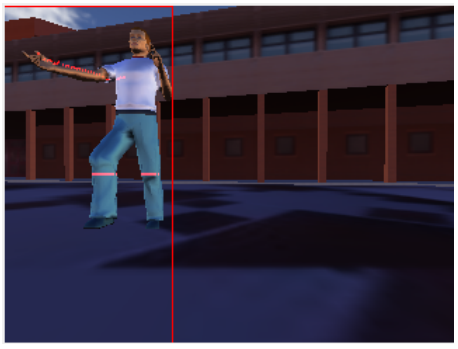


Figura 5.19: Vista del rectángulo que define la ROI en caso 3.

5. Teniendo las dimensiones del recuadro se acopla para las dos imágenes, derecha e izquierda. Como se puede ver, la imagen izquierda no contempla al peatón, lo que resultará muy beneficioso para poder descartar el fondo apropiadamente.

Figura 5.20: ROI derecha e izquierda para el caso 3.

6. Al filtrar por el color de la carretera, la imagen se transforma como se ve a continuación. Se extraen los píxeles de ciertas partes del suelo únicamente, ya que esta escena no se sitúa en la calzada, si no en el campus. En este caso este paso podría ser irrelevante para el objetivo, pero al no afectar al resto del proceso se ejecuta igualmente.

Figura 5.21: Extracción de carretera en caso 3.

7. El fondo en esta escena, consiste en un edificio que se encuentra en un plano cercano al del peatón por detrás de éste. Por este motivo al realizar la resta de las dos imágenes resulta que no es posible descartar el fondo. Pero si se puede divisar en cierto modo la silueta del peatón. Así en el siguiente paso se tratará de limpiar la imagen para que se pueda identificar más nítidamente.

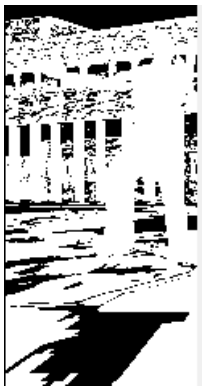


Figura 5.22: Sustracción de imágenes en caso 3.

8. Al tener más nivel de blanco en la figura obtenida se procede a desarrollar el flujo de ejecución correspondiente, que es más agresivo, y se consigue limpiar bastante la imagen. Se pueden identificar varias figuras pequeñas y otra de mayor tamaño correspondiente al peatón. La figura más grande siempre será la más cercana al vehículo, porque al realizar la sustracción de las imágenes, las figuras del fondo o desaparecen o se quedan difuminadas.

Figura 5.23: Transformación morfológica en el caso 3.

9. Este caso es el más complicado y el que más tiempo lleva en la fase de etiquetado. Esto es debido a que hay demasiadas formas a identificar. Aún así se hace en un tiempo razonable y se relaciona acertadamente al obstáculo con la forma más grande, la de color rojo. Las manchas que antes aparecían abajo en la figura anterior no aparecen en este paso porque al realizar el etiquetado se descartan. Esto sucede porque al tener un máximo de 20 colores, y se desechan algunos en el proceso, no se pueden identificar más de un número determinado de elementos. No crea graves consecuencias ya que el objeto que se quiere clasificar siempre será el más grande y se podrá identificar. En cualquier caso es posible solucionarlo añadiendo más colores al algoritmo.

Figura 5.24: Etiquetación en el caso 3.

10. Se puede identificar que la forma más grande de la imagen es la de color rojo y su altura en número de píxeles es igual a 141 que traducido a milímetros es de 1141. Teniendo en cuenta que el ancho resultó ser de 520 mm el objeto se puede clasificar como un peatón que se encuentra a 1817 mm del vehículo en un ángulo de 72º.

5.3.1. Análisis de resultados

Al final el resultado es correcto. La clasificación se realiza apropiadamente y es posible minimizar, en la fase de etiquetado, el fondo que no se había podido descartar en la sustracción de imágenes. Resulta difícil diferenciar en la resta, que píxeles pertenecen al peatón y cuales al fondo. Sin embargo al estar el obstáculo en un primer plano, se consigue mediante transformaciones morfológicas, llegar a un estado en el que es posible diferenciar la silueta del obstáculo.

5.4. Caso 4

El siguiente ejemplo muestra, en el frontal del vehículo, a dos viandantes, una mujer y un niño. Así se podrá comprobar la funcionalidad cuando hay más de un peatón y cuando uno de ellos es de poca altura. La visión de la calzada es escasa, porque los peatones se encuentran sobre un paso de cebra, con lo que también pueden existir dificultades cuando se filtra por el color de la carretera.

1. A continuación se muestran las imágenes recogidas por las cámaras web izquierda y derecha respectivamente.

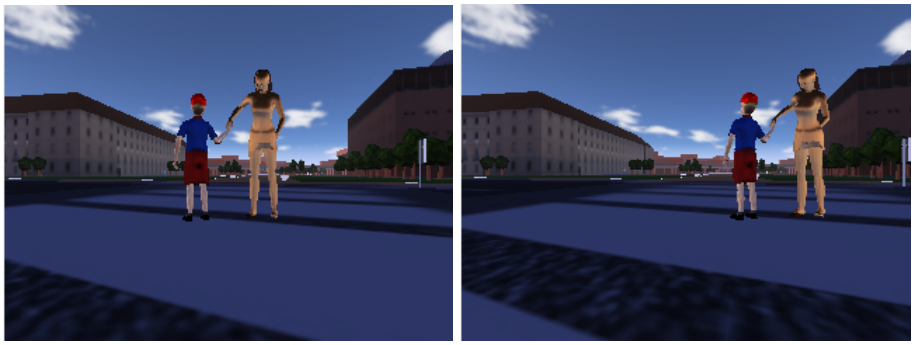


Figura 5.25: Vistas de las dos cámaras web del caso 4.

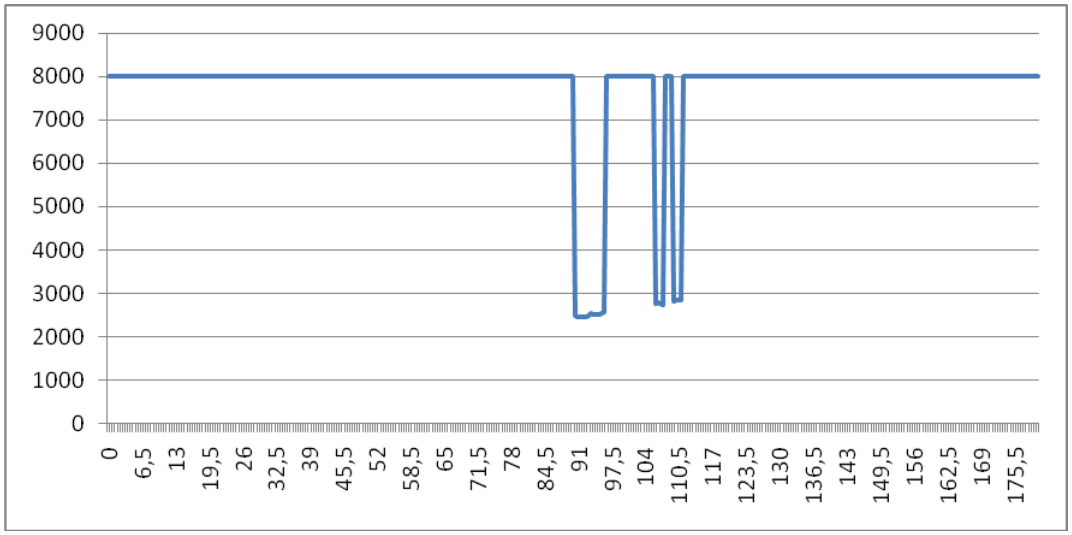


Figura 5.26: Gráfico del vector de distancias del caso 4.

2. El vector de distancias refleja como el niño, al tener poca altura lo reconoce como un único segmento, mientras que la mujer consta de dos segmentos. Los dos obstáculos se encuentran bastante cercanos al vehículo y ligeramente a la derecha. Concretamente se sitúan en 90'5º y en 106º, por lo que la imagen con la que se trabajará será la derecha.

3. Las distancias a las que se sitúan son el primero a 2459mm. y el segundo a 2747mm. La amplitud se define como 6º en el primer caso y 5'5º en el segundo. Luego se calcula la coordenada x resultando 126 para el primero y 189 para el segundo.

4. Para definir la ROI, se ha tenido que calcular el ancho y el alto en función de la profundidad que resulta como 44x210 píxeles para el niño y 48x224 para la mujer.

Figura 5.27: Vistas del rectángulo que define la ROI en caso 4.

5. El siguiente paso consiste en recortar las dos imágenes, derecha e izquierda, según las dimensiones obtenidas anteriormente. Se puede observar como para el segundo caso se complica un poco la situación ya que la imagen izquierda contempla al niño con lo que al realizar la sustracción se tendrán problemas con dicha situación.

Figura 5.28: ROI derecha e izquierda para el caso 4.

6. Como se ha comentado antes, el filtrado por el color no hace mucho debido a que los peatones se encuentran sobre un paso de peatones. Aún así se consigue descartar ciertas partes de la imagen que no son relevantes para el posterior análisis.

Figura 5.29: Extracción de carretera en caso 4.

7. Para el primer caso la sustracción de imágenes resulta ser satisfactoria. Sin embargo para el segundo caso al tener al otro peatón en la imagen izquierda supone que la sustracción distorsione la escena y se vean las dos figuras mezcladas. Pero como se verá no afecta tanto al análisis porque se puede calcular la altura aproximadamente y se puede clasificar correctamente.

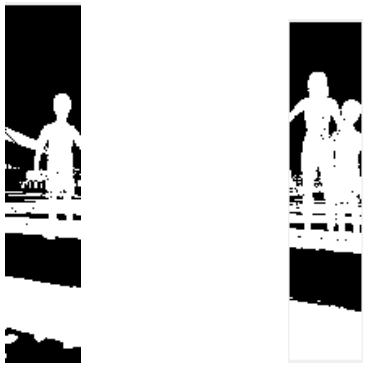


Figura 5.30: Sustracción de imágenes en caso 4.

8. Con las transformaciones morfológicas se consigue, aunque no con mucha exactitud definir mejor las siluetas de los obstáculos. Para el objetivo de conseguir la altura, el resultado es conveniente.

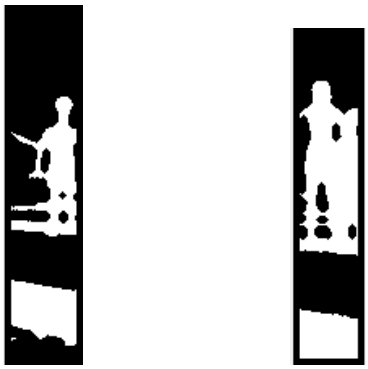


Figura 5.31: Transformación morfológica en el caso 4.

9. Para el etiquetado, como se ve, se detectan dos formas en los dos casos. La de abajo en las dos imágenes, que corresponde con el paso de cebra, se descartará ya que es de menor tamaño.

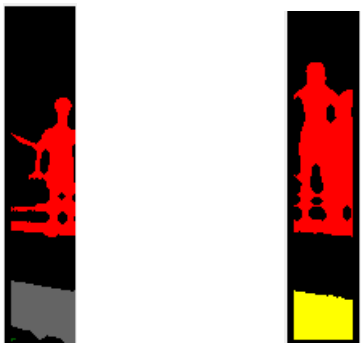


Figura 5.32: Etiquetación en el caso 4.

10. Las formas rojas de los imágenes anteriores corresponden con la silueta obtenida del obstáculo y la altura calculada para cada una son de 87 y 109 píxeles respectivamente. Después se calcula mediante la escala el valor en milímetros que es de 970 y 1293. El ancho calculado anteriormente tiene un valor de 276mm. y 258mm. Por tanto, los dos obstáculos pueden ser identificados como peatones.

5.4.1. Análisis de resultados

Este caso resulta de los más complicados. Aún así los resultados son acertados. El hecho de que los dos peatones estén juntos y los distinga como dos obstáculos diferentes es un gran paso. El problema surge cuando al hacer la sustracción de imágenes se tienen las dos siluetas superpuestas como una única figura. Al estar la figura de mayor tamaño más a la derecha, tiene como efecto que la medida de la altura no se vea afectada por este suceso. Sin embargo si hubiese sido al contrario se tendría que el valor de la altura calculada sería incorrecto.

5.5. Caso 5

Este último caso muestra una señal frente al vehículo. La señal, con respecto a la altura puede ser similar a una persona, pero no así el ancho, al menos en este caso. Es importante no clasificar este tipo de obstáculos como posibles peatones, por eso se incluye este ejemplo en las pruebas realizadas.

1. A continuación se muestran las imágenes recogidas por las cámaras web, izquierda y derecha respectivamente.

Figura 5.33: Vistas de las dos cámaras web del caso 5.

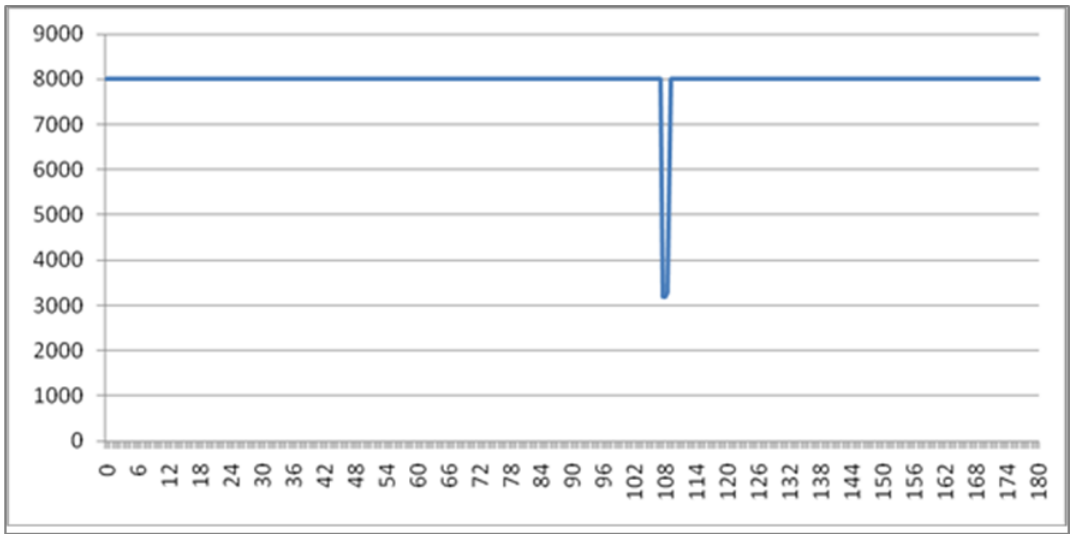


Figura 5.34: Gráfico del vector de distancias del caso 5.

2. En el vector de distancias 5.34 se puede reconocer un único segmento muy estrecho, ligeramente situado a la derecha de la escena, en la posición correspondiente a 107'5º.

3. El obstáculo se encuentra a una profundidad de 3194 mm. y la amplitud es de 1'5º. Luego se calcula la coordenada x y se obtiene que son 191 píxeles.

4. La región de interés que se muestra a continuación como un recuadro rojo se define por las proporciones calculadas antes del ancho y la altura. El tamaño resulta ser 16x180 píxeles.

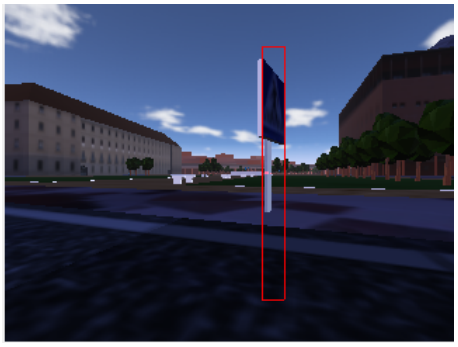


Figura 5.35: Vista del rectángulo que define la ROI en caso 5.

5. Al recortar las imágenes, en la izquierda no se encuentra más que el fondo, con lo que se podrá definir mejor la silueta del objeto. Pero el suelo no es sólo calzada, ya que se incluye también la acera, lo que complicará un poco la situación.

Figura 5.36: ROI derecha e izquierda para el caso 5.

6. Al filtrar por el color de la carretera se pueden descartar ciertos fragmentos del suelo aunque no todo. Esto provoca que la altura calculada se vea afectada por este paso y resulte finalmente más de lo que sería realmente.

Figura 5.37: Extracción de carretera en caso 5.

7. Al restar las imágenes se puede eliminar parte del fondo. Pero al ser la ROI bastante reducida, el nivel de blanco es mayor de lo esperado.

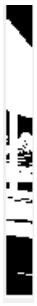


Figura 5.38: Sustracción de imágenes en caso 5.

8. En este paso se transforma la imagen con los operadores morfológicos correspondientes y el resultado es bueno. Se consigue quitar esa parte de suelo que podía hacer aumentar considerablemente la altura del obstáculo y el valor queda más cercano al dato real.

Figura 5.39: Transformación morfológica en el caso 5.

9. Las figuras identificadas en la imagen son únicamente dos, la primera, de color rojo es la que corresponde con el objeto y de la cual se extrae el dato de la altura.

Figura 5.40: Etiquetación en el caso 5.

10. La figura antes obtenida resulta tener una altura de 116 píxeles que al pasar a la escala real se obtiene un valor de 1608 mm. Al ser el ancho, un valor muy pequeño, de 113 mm., la clasificación concluye que no se trata de un peatón.

5.5.1. Análisis de resultados

Este caso es importante, ya que tanto los árboles como las señales son obstáculos muy comunes en las carreteras y pueden tener características similares a los viandantes, morfológicamente hablando. Es necesario poder identificar claramente la ubicación del objeto y las dimensiones del mismo para que los resultados sean acertados. En este caso en concreto se puede deducir que no se trata de un peatón por el ancho del obstáculo. Esta medida se calcula a partir de los datos proporcionados por el láser que detecta el poste de la señal. Por lo tanto el ancho calculado será menor que las proporciones establecidas para un peatón y su clasificación resulta acertada.

Capítulo 6

Conclusiones y Líneas Futuras

6.1. Conclusiones del proyecto

Este proyecto surge de la necesidad de crear un sistema avanzado de asistencia a la conducción que sea capaz de detectar peatones en entornos urbanos. La seguridad de los viandantes es insuficiente, siendo los mayores perjudicados en los accidentes de tráfico. Los sistemas de seguridad existentes en el mercado están más orientados a la seguridad pasiva, es decir, procurar que las consecuencias del accidente supongan un mínimo riesgo para los usuarios de la vía. Sin embargo con un sistema de seguridad activa como el que se propone se puede evitar que se produzca el accidente.

El sistema propuesto se compone de tres sensores: dos cámaras y un telémetro láser integrados en el frontal del vehículo. Según estudios anteriores, los sistemas basados únicamente en visión no son suficientemente potentes para afrontar el problema de la detección de peatones de forma fiable y precisa. Por ello se añade la información del sensor láser que otorga al sistema una mayor fiabilidad y robustez. El sensor láser es capaz de detectar obstáculos sin que se vea afectado por la luminosidad o las condiciones meteorológicas adversas. Con las dos cámaras se puede obtener una representación de la escena en tres dimensiones para poder calcular las dimensiones reales de los obstáculos encontrados.

Para la fusión sensorial, es decir, la combinación de la información procedente de los sensores, se ha utilizado un modelo asociativo. Primero se procesa la información del telémetro láser por su rapidez y precisión para posteriormente es-

tablecer una correspondencia del objeto detectado en las imágenes y así extraer las características pertinentes para llevar a cabo la clasificación.

Existen dos tipos básicos de reconocimiento, las técnicas basadas en movimiento (*motion-based*) y las técnicas basadas en los rasgos (*shape-based*). Las basadas en el movimiento se centran en las características rítmicas o patrones de movimiento únicos en los humanos. Pero tienen ciertas limitaciones: puede que las piernas no sean visibles en la imagen, se requiere una secuencia de imágenes que retarda el reconocimiento y no se puede detectar peatones quietos o con otro tipo de movimiento como puede ser saltando. Sin embargo las técnicas basadas en la forma de los peatones pueden detectarlos tanto estática como dinámicamente. El problema es el gran rango de variaciones en apariencia: pose, luz o ropa y posibles problemas de oclusión. Existen sistemas que se centran en la cara pero tienen como inconveniente que si el peatón no está mirando a la cámara no es posible detectarlo.

En este trabajo para discriminar si se trata de un peatón u otro objeto se utilizan las dimensiones y la distancia a la que se encuentra. Con estos rasgos es posible clasificar de forma rápida y fiable si se trata de un peatón, pero existen ciertos problemas de reconocimiento en ciertas situaciones como si se encuentran obstáculos con proporciones humanas, si el color de la ropa del peatón es igual que el color de la calzada o si existen dos peatones muy juntos, que no es posible detectarlos por separado.

Es muy importante que el proyecto sea válido para implementarlo en tiempo real y con una memoria y capacidad de cómputo limitados. Si este sistema se incorpora en los procesadores embebidos en un coche, se debe tener en cuenta que los métodos utilizados sean suficientemente sencillos para que se puedan ejecutar sin problemas. Por este motivo el método de clasificación podría definirse como un sistema experto muy sencillo, consistente en una única regla que restringe el tamaño del objeto detectado.

Existen estudios anteriores sobre sistemas muy similares al propuesto. La principal diferencia es que en este trabajo se han utilizado transformaciones morfológicas para limpiar la imagen de ruido. Esto hace que el enfoque cambie en el sentido de que en fases anteriores se utilizan métodos muy sencillos que agilizan todo el proceso en lugar de utilizar métodos más complejos que pueden conseguir mejores resultados, pero en cambio, ralentizan muchísimo la ejecución. Si por ejemplo en lugar de utilizar la sustracción de imágenes se utilizase un mapa de disparidad los resultados serían más detallados, pero se necesitaría un procesador suficientemente potente para poder implementarse en tiempo real.

Al tener en cuenta dos tipos de escenario, se puede generalizar a casi cualquier entorno en el que se pueda encontrar el vehículo. Uno de ellos se caracteriza por tener un fondo limpio, es decir, no se encuentran más obstáculos que los que se quieren detectar a partir de una distancia considerable. El segundo escenario, más urbano, cuenta con que detrás del obstáculo puedan aparecer más elementos no identificados como fondo cuando deberían serlo, como pueden ser edificios u otros coches. Se podría decir que la forma de distinguir entre estos dos tipos de escenario no es suficientemente precisa, pero en el entorno virtual funciona correctamente.

En cuanto a la eficiencia del sistema, al haber sido implementado en un entorno simulado y de forma estática, no se puede asegurar su viabilidad en la integración en un sistema real. Pero al haber sido desarrollado en MRDS es posible el paso de un entorno simulado a uno real sin tener que realizar demasiados cambios en el código. Con respecto a que el estudio se ha realizado teniendo en cuenta que el escenario es estático, no quiere decir que no se pueda ejecutar en un entorno dinámico, pero en este caso será necesario acotar las pruebas, ya que el entorno simulado posee ciertas limitaciones.

Se puede decir que la aplicación propuesta es bastante completa, cubriendo todas las fases del reconocimiento de patrones, adquisición, segmentación, extracción de características y clasificación. Todo ello integrado dentro de la fusión sensorial, que es clave que se realice correctamente para que los resultados conseguidos sean fiables.

6.2. Líneas futuras propuestas

En este trabajo se ha propuesto un sistema que sea capaz de detectar peatones mediante fusión sensorial de visión estéreo y la información procedente del sensor láser en un contexto automovilístico. Se ha acotado de tal forma que la aplicación implementada realiza todas las fases del análisis, incluida la clasificación, pero no se ha diseñado el sistema de respuesta, ni la integración en un entorno real. Existen numerosas técnicas validas y posibles para afrontar el problema estudiado, que deja muchas líneas de investigación abiertas. A continuación se exponen las principales.

6.2.1. Integración en un sistema real

Como línea principal que se puede seguir a partir de este proyecto está la integración de esta aplicación en un sistema completo que conecte con algún tipo de actuador y transmita una respuesta al sistema. Es necesario definir los actuadores y la orquestación con la información dada por los sensores. El trabajo realizado define como se procesa la información de los sensores para obtener una respuesta fiable, lo siguiente que se debe hacer es definir una acción según la respuesta lograda. Algunos sistemas implementan un sistema de frenado de emergencia, otros, la gran mayoría, una señal de alarma o aviso, para que sea el conductor el que decida como actuar según el caso. Para el caso del sistema de frenado automático se requiere que la implementación sea muy fiable y robusta, con una tasa de fallos mínima y un tiempo de respuesta menor que los reflejos humanos.

6.2.2. Realimentación del sistema

También se puede ampliar la implementación incluyendo un seguimiento de las detecciones que realimente el sistema y aporte mayor información. Se puede realizar incluyendo un identificador a cada objeto detectado, para confirmar que existe en la siguiente imagen y se puede calcular la dirección de movimiento y la velocidad del objeto detectado.

El seguimiento de las respuestas obtenidas reduciría los falsos positivos y negativos. Pero no es una tarea fácil. Por un lado, los movimientos del coche hacen que la inclinación de la cámara varíe, afectando a la posición de los obstáculos detectados. Por otro lado, el algoritmo debe poder hacer frente a los falsos negativos, ya que si en una imagen se pierde una detección, debería estimarse su movimiento. El algoritmo de seguimiento deberá ser robusto para solucionar estos problemas así como resolver la asignación de cada región de interés al peatón correspondiente.

6.2.3. Mejora del clasificador

El clasificador utilizado puede definirse como un sistema experto que mediante reglas restringe el tamaño de los obstáculos para incluirlos en una clase u otra. Las reglas obtenidas se basan en estudios estadísticos pero se podría realizar con

un banco de pruebas más amplio que haga el sistema de clasificación más ajustado y preciso.

También se podría elegir una red neuronal que se entrene con las ROI y así se establezca de forma automática la función discriminante. Otra opción es tener patrones de las dos clases y mediante emparejado de coincidencias o *matching* establecer a que clase pertenece. Para ello sería necesario que las ROI estén normalizadas y que sean invariantes ante transformaciones de escala, rotación y traslación.

6.2.4. Combinar con otros sensores

Se puede añadir cualquier tipo de sensor que produzca información adicional o para añadir redundancia de información y hacer más fiable el sistema. Puede que el tamaño como atributo para establecer la clasificación resulte insuficiente en algunos casos. Por ello se propone como otra posible línea de investigación la fusión con más sensores. Existen estudios que utilizan un sensor radar como complemento al láser y las cámaras para poder controlar las zonas no visibles o ciegas, como los ángulos muertos.

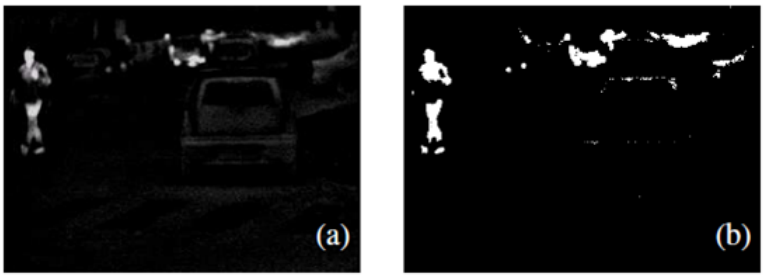


Figura 6.1: Segmentación basada en intensidad [26].

Cuando se realiza la segmentación se ha utilizado la sustracción de imágenes, en lugar de los mapas de disparidad. Para crear los mapas de disparidad con visión estereoscópica se necesita una capacidad de cómputo elevada y los resultados acaban siendo poco precisos y con mucho ruido. Sin embargo con el espectro infrarrojo, al contrario que ocurre en el espectro visible, los cambios de iluminación no le afectan tanto. Los niveles de intensidad en una imagen de infrarrojos son representativos de la temperatura de las superficies de los objetos. Los peatones, generalmente, emiten más calor que los objetos estáticos, como los árboles o la carretera. Por tanto, esas regiones que contienen peatones aparecerán más brillan-

tes que el resto y se puede realizar la segmentación en base a esta información como se expone en [26].

6.2.5. Ampliar los objetivos

Como última propuesta se sugiere incorporar al sistema de detección a otros elementos que se ven afectados como potenciales víctimas en caso de accidente de tráfico como son los ciclistas y motoristas. Para ello sería necesario cambiar el clasificador e incluir estas dos clases o una sola que agrupe a los dos. Además no sería posible hacer la clasificación únicamente teniendo en cuenta las dimensiones por lo que se debería añadir información como la silueta o la velocidad o realizar la comparación con patrones y algoritmos como el del vecino más cercano. También se podrían incluir dinámicas de movimiento de peatones en el simulador y detectar patrones de movimiento típicos.

Definiciones y Acrónimos

Binarizada : Referente a una imagen dicromática, normalmente en blanco y negro.

Convolución : Proceso matemático que transforma dos funciones f y g en una tercera función que en cierto sentido representa la magnitud en la que se superponen f y una versión trasladada e invertida de g. En procesado de imagen la mayoría de los filtros usan una matriz de convolución.

Disparidad : En visión estéreo la distancia que separa a un punto de su correspondencia en la otra imagen. Es inversamente proporcional a la profundidad.

Fondo : El resto de la imagen que no es un obstáculo.

Objeto u Obstáculo : Se refiere al plano más cercano de la región de interés en la imagen, que corresponde al elemento detectado por el sensor láser.

Píxel : Superficie homogénea más pequeña de las que componen una imagen, que se define por su brillo y color. En el vector que define la imagen las tres posiciones que definen el color en una posición concreta.

Socio o Partner : Se trata de dos servicios que tienen la posibilidad de comunicarse entre ellos a través de sus puertos. Se tiene que establecer de antemano que son socios para poder establecer la comunicación.

Umbralización : Técnica de segmentación mediante la cual se divide la imagen en grupos de píxeles según el nivel de gris, o color de cada uno.

Visión estereoscópica : La capacidad de a partir de dos imágenes recrear la escena en tres dimensiones.

CCR Concurrency and Coordination Runtime.

DLL Dynamically Lincked Library.

DSS Descentralizaed Software Services.

HW HardWare.

FIFO First in First Out.

MRDS Microsoft Rbotic Developer Studio.

ROI Region of Interest.

SW SoftWare.

VSE Visual Simulation Environment.

VPL Visual Programming Languaje.

Apéndices

Apéndice A

Gestión del proyecto

A.1. Planificación

El proyecto expuesto se puede dividir en cinco grandes fases. Estas son: la documentación, el análisis, el diseño, la implementación y la evaluación. Cada una de esas fases, a su vez, ha sido dividida en diversas tareas, para así, poder llegar al objetivo final, consiguiendo objetivos intermedios más sencillos de realizar.

Se puede representar la estructura del proyecto mediante el desglose de tareas de la figura A.1.En el cual no se han incluido las tareas relativas a la creación del presente documento, ni a los preparativos de la presentación del mismo. Dichas tareas, con sus objetivos, dependencias, duración y esfuerzo son detalladas a continuación:

1. Documentación
    - Descripción: Estudio de las técnicas utilizadas para la detección de peatones, así como los conceptos en los que se basan. También se debe analizar estudios anteriores que resuelven el mismo problema y el software utilizado.
    - Dependencia: No existe ninguna dependencia con las demás tareas.
    - Duración: 12 semanas.
    - División: Esta fase se puede dividir en las siguientes tareas:

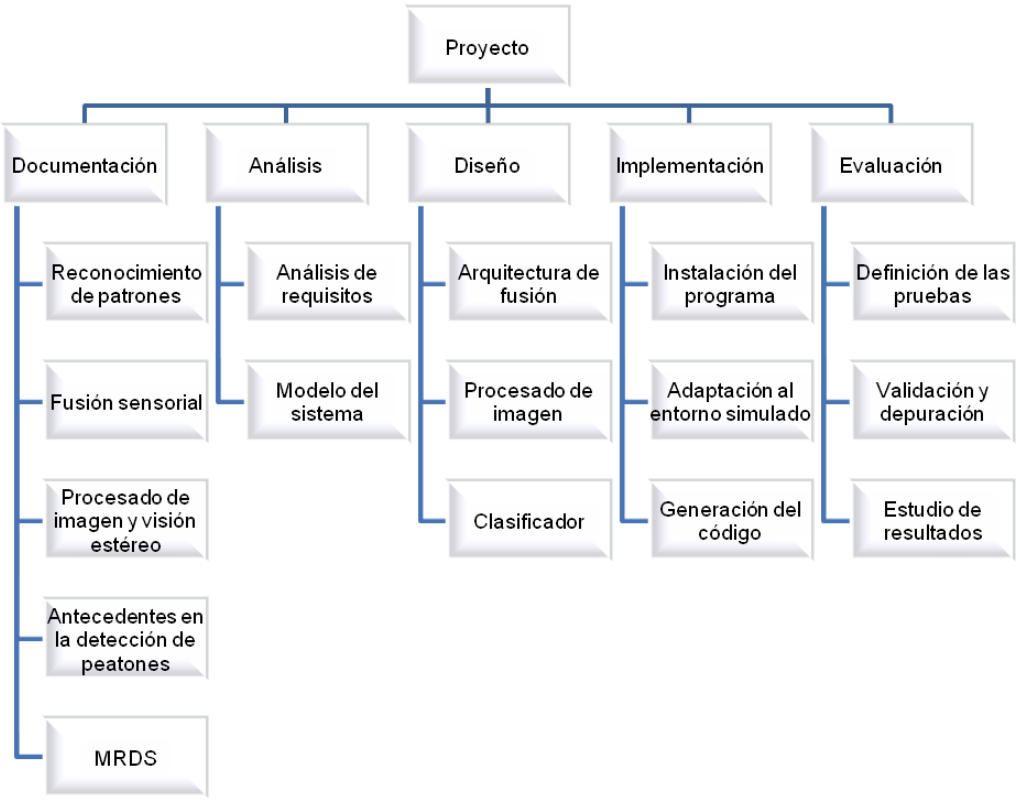


Figura A.1: División de tareas.

a) Reconocimiento de patrones

- Descripción: Estudio del esquema básico utilizado para el reconocimiento de patrones, con todas sus fases. Adquisición, segmentación, extracción de características y clasificación.
- Dependencia: No existen dependencias.
- Duración: 2 semanas.
- Recursos: Ingeniero 0.5 hombre/mes.

b) Fusión sensorial

- Descripción: Estudio de las diferentes formas de implementar la fusión sensorial. Los modelos y arquitecturas existentes.
- Dependencia: No existen dependencias.
- Duración: 2 semanas.
- Recursos: Ingeniero 0.5 hombre/mes.

c) Procesado de imagen y visión estéreo

- Descripción: Estudio de las múltiples técnicas que existen para el procesado de imagen y los objetivos que se alcanzan con cada una. También los conceptos base sobre visión estereoscópica.
- Dependencia: No existen dependencias.
- Duración: 2 semanas.
- Recursos: Ingeniero 0.5 hombre/mes.

d) Antecedentes en la detección de peatones

- Descripción: Analizar los diferentes modelos propuestos que abordan el mismo problema planteado en este proyecto.
- Dependencia: No debe comenzar hasta que se finalicen las tareas 1.a, 1.b y 1.c.
- Duración: 3semanas.
- Recursos: Ingeniero 0.5 hombre/mes.

e) MRDS

- Descripción: Estudio del software MRDS y del lenguaje C para poder llevar a cabo la implementación.
- Dependencia: No existen dependencias.
- Duración: 3 semanas.
- Recursos: Ingeniero 0.75 hombre/mes

f) Análisis

- Descripción: Se debe realizar un análisis de la documentación anterior para poder establecer cómo resolver el problema. Para ello es necesario definir el alcance y los requisitos de la aplicación.
- Dependencia: Esta actividad depende de la anterior, la actividad número 1, de documentación.
- Duración: 4 semanas.
- División: Esta fase se puede dividir en las siguientes tareas:

enumerate

g) Análisis de requisitos

- Descripción: Descripción y posterior análisis de los requerimientos de la aplicación que definen el alcance y limitan el proyecto.
- Dependencia: Depende de todas las tareas anteriores, por tanto de la tarea 1.e.
- Duración: 2 semanas.
- Recursos: Ingeniero 0.25 hombre/mes.

h) Modelo del sistema

- Descripción: Describir el dominio del proyecto y el modelo del sistema. Que elementos intervienen, como se comunican y las entradas y salidas que se requieren.
- Dependencia: esta tarea depende de todas las anteriores, por lo tanto dependerá directamente de la tarea 2.a.
- Duración: 2 semanas.
- Recursos: Ingeniero 0.5 hombre/mes.

2. Diseño

- Descripción: En esta fase se pretende realizar todo el modelo de diseño para poder llevar a cabo la implementación. Se escogen las técnicas más adecuadas y se define el flujo de ejecución.
- Dependencia: Depende de la fase de análisis por tanto no podrá comenzar hasta que no haya concluido la tarea 2.
- Duración: 7 semanas.
- División: Esta fase se puede dividir en las siguientes tareas:

a) Arquitectura de fusión

- Descripción: Se define como se realizará la fusión de los dos tipos de sensor, las cámaras y el láser.
- Dependencia: Depende de la tarea 2.b.
- Duración: 3 semanas.
- Recursos: Ingeniero 1 hombre/mes.

b) Procesado de imagen

- Descripción: Teniendo en la imagen la información de la fusión se debe procesar para obtener una información más refinada y especifica. Se debe elegir que técnicas son las más apropiadas.
- Dependencia: Depende de la tarea 3.a
- Duración: 2 semanas.
- Recursos: Ingeniero 1 hombre/mes.

c) Clasificador

- Descripción: Hay que elegir los atributos que definen la clasificación. Cuando se tienen todos los datos pertinentes se debe establecer una función discriminante que decida si un objeto detectado es o no un peatón.

- Dependencia: Depende de la tarea 3.b.
- Duración: 2 semanas.
- Recursos: Ingeniero 0.75 hombre/mes.

3. Implementación

   - Descripción: Una vez se haya decidido como debe ser la aplicación se realiza la implementación del mismo en un entorno de simulación dado.
   - Dependencia: Depende de la tarea de Diseño, no empezará hasta que concluya la actividad 3.
   - Duración: 11 semanas
   - División: Esta fase se puede dividir en las siguientes tareas:

   a) Instalación del programa
      - Descripción: Es necesario instalar el software MRDS para poder comenzar con la implementación.
      - Dependencia: Depende de la tarea 1.e.
      - Duración: 2 semanas.
      - Recursos: Ingeniero 0.5 hombre/mes.

   b) Adaptación al entorno simulado
      - Descripción: El entorno de simulación está ya construido y se debe implementar un nuevo servicio que conecte con él y realice las funciones necesarias. Se debe comunicar este servicio con los sensores incluidos en el vehículo de forma que se pueda recibir la información directamente.
      - Dependencia: Depende de la tarea 4.b.
      - Duración: 4 semanas.
      - Recursos: Ingeniero 1 hombre/mes.

   c) Generación del código
      - Descripción: Cuando ya está todo el entorno montado se puede comenzar a generar el código que implemente el diseño planteado.
      - Dependencia: Depende de las tareas 4.b y 3.c.
      - Duración: 5 semanas.
      - Recursos: Ingeniero 1 hombre/mes.

4. Evaluación

   - Descripción: Para comprobar que el programa funciona correctamente y que los resultados son los esperados se diseñan las pruebas que a la vez sirven para depurar los posibles errores que existan.
   - Dependencia: Depende de la anterior tarea de implementación, la actividad 4.
   - Duración: 9 semanas.
   - División: Esta fase se puede dividir en las siguientes tareas:

   a) Definición de las pruebas
      - Descripción: Se establecen 5 escenarios, todos distintos unos de otros, que examinan aspectos concretos de la implementación que pueden fallar.
      - Dependencia: Depende de la tarea 4.c.
      - Duración: 2 semanas.
      - Recursos: Ingeniero 0.25 hombre/mes.

   b) Validación y depuración
      - Descripción: Una vez estén definidos todos los casos de prueba, se ejecuta la aplicación con cada uno y se examina si falla. En caso positivo se depura el código y se vuelve a probar.
      - Dependencia: Depende de la tarea 5.a.
      - Duración: 5 semanas.
      - Recursos: Ingeniero 0.75 hombre/mes.

   c) Estudio de resultados
      - Descripción: Depurado totalmente el programa se examinan exhaustivamente los resultados obtenidos.
      - Dependencia: Depende de la tarea 5.b.
      - Duración: 2 semanas.
      - Recursos: Ingeniero 0.5 hombre/mes.

Como medida del tiempo de la duración de todas las tareas, ha sido considerada una semana. Entendiendo por semana únicamente 5 días laborables, es decir de lunes a viernes. En cuanto al cálculo de los recursos humanos utilizados, se ha considerado que una persona trabaja únicamente 8 horas al día. Por lo tanto, si los recursos están medidos en hombres/mes, si el valor es 1, indicará que esa persona ha trabajado 8 horas al día en esa tarea.

A continuación se muestran, en primer lugar, el diagrama de Gantt resumido A.2 con las fases principales y en segundo lugar, el diagrama de Gantt expandido A.3 con todas las tareas y todas las dependencias que existen entre ellas.

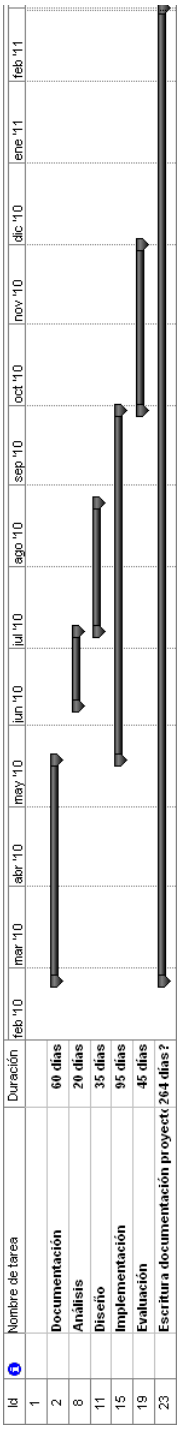


Figura A.2: Diagrama de Gantt resumido.

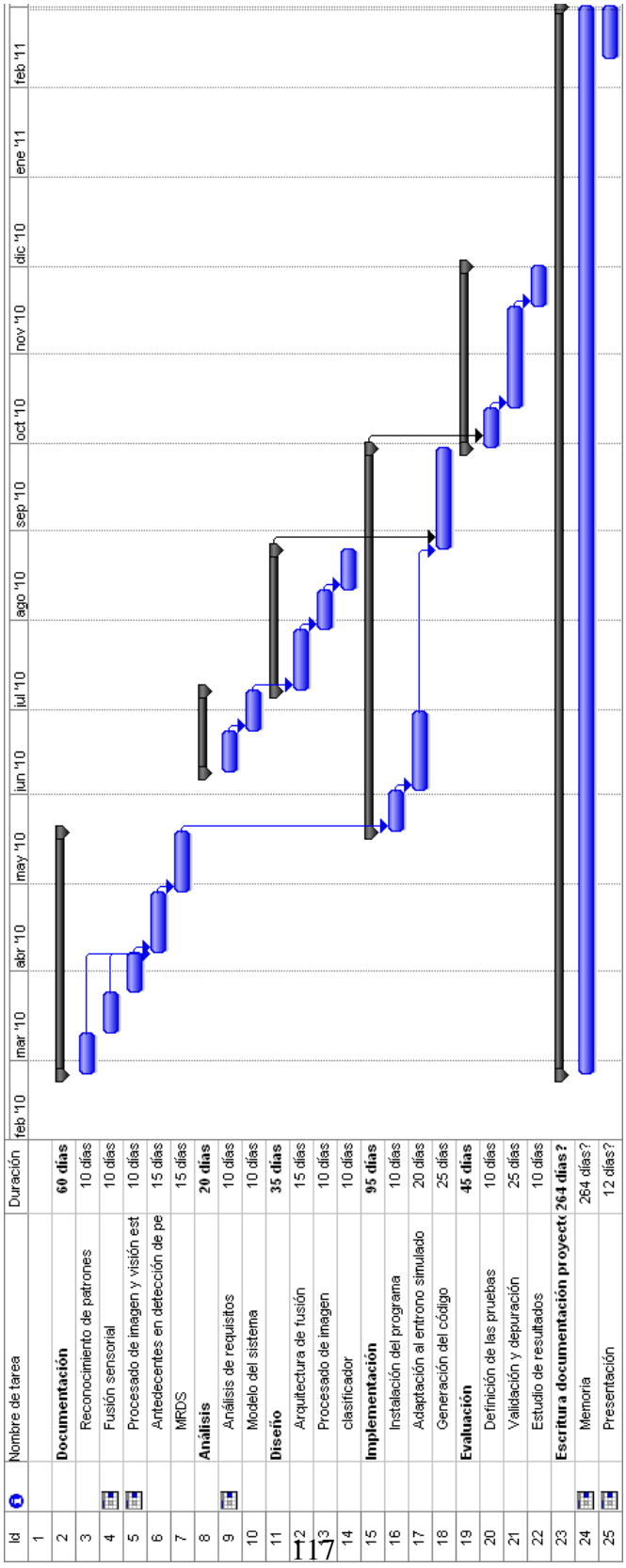


Figura A.3: Diagrama de Gantt.

Como se puede ver la duración total del proyecto ha sido de un año, teniendo en cuenta que la dedicación en ciertas épocas ha sido parcial. La realización del documento consta como se ha realizado durante todo el periodo de realización del proyecto porque se ha ido haciendo simultáneamente con el resto de tareas pero con una dedicación menor.

En la tabla A.1 se muestra un resumen de toda la información sobre la división de actividades y tareas. En ella, además, se especifica tanto el tiempo empleado como los recursos consumidos por cada una de las tareas.

En relación con la planificación del proyecto cabe destacar las siguientes consideraciones y observaciones:

La ejecución del proyecto se desarrolló en líneas generales, según lo previsto en la planificación. En primer lugar destacar que algunos procesos se realizaron incluso en menor tiempo que el establecido como es el caso de las tareas relacionadas con el desarrollo de los métodos que implementan el procesado de la imagen.

En otros casos se puede argumentar que ciertas etapas como la adaptación al entorno simulado han durado más de lo previsto debido a que surgieron problemas para desarrollar la interconexión de los servicios. Es decir, enlazar el servicio implementado con los ya creados para que se pudieran comunicar y se pudiera establecer el flujo de información. Por otro lado la etapa de creación de la arquitectura de fusión también sufrió algunos retrasos porque resultó más complicado de lo que se esperaba, establecer la correspondencia entre los sensores.

Tarea	**Duración (semanas)**	**Recursos (hombres/mes)**	**Total horas**
1. Documentación			
1.a Reconocimiento de patrones	2	0.5	40
1.b Fusión sensorial	2	0.5	40
1.c Procesado de imágenes y visión estéreo.	2	0.5	40
1.d Antecedentes en detección de peatones	3	0.5	60
1.e MRDS	3	0.5	60
Total	12		240
2. Análisis			
2.a Análisis de requisitos	2	0.25	40
2.b Modelo del sistema	2	0.5	20
Total	4		60
3. Diseño			
3.a Arquitectura fusión	3	1	120
3.b Procesado de imagen	2	1	80
3.c Clasificador	2	0.75	60
Total	7		260
4. Implementación			
4.a Instalación del programa	2	0.5	40
4.b Adaptación al entorno simulado	4	1	160
4.c Generación de código	5	1	200
Total	11		400
5. Evaluación			
5.a Definición de las pruebas	2	0.25	20
5.b Validación y depuración	5	0.75	150
5.c Estudio resultados	2	0.5	40
Total	9		210
6. Escritura de la documentación del proyecto			
6.a Escritura del presente documento	73	0.05	146
6.b Preparación de la presentación	2	0.25	20
Total	75		166
Total de horas			1336

Tabla A.1: División de tareas

A.2. Presupuesto

A continuación se presentan justificados los costes globales de la realización de este Proyecto Fin de Carrera. Suponiendo para los recursos humanos unos costes/hora hipotéticos, que tendrían que adaptarse dependiendo del mercado. Tales costes, imputables a gastos de personal y de material, se pueden deducir de las siguientes tablas.

Herramienta SW	Coste licencia (€)
Windows 7 Proffesional	309,00 €
MRDS	Libre en MSDNAA
Microsoft Visual Studio	Libre en MSDNAA
Microsoft Office 2007	407,94 €
LaTeX	Libre
TOTAL (€)	716,94 €

Tabla A.2: Costes del SW

En la anterior tabla se muestra el coste de los recursos software utilizados para el desarrollo del proyecto. Las licencias de MRDS y Microsoft Visual Studio han sido obtenidas de a través del portal Microsoft Developer Network Academia Alliance (MSDNAA) para estudiantes y personal docente de la universidad Carlos III.

Categoría	Coste (€/hora)	Dedicación	Total horas
Ingeniero en Informática Senior	25 €	0.3	400,8 horas
Ingeniero en Informática Junior	12 €	1	1336 horas
TOTAL (€)			26052 €

Tabla A.3: Costes recursos humanos

Esta tabla muestra el coste de los recursos humanos, siendo el ingeniero en informática Senior el tutor de este proyecto, Raúl Arrabales Moreno y el ingeniero en informática junior la autora del documento, Almudena Domínguez Fernández.

La última tabla A.4 presenta el coste de los recursos hardware, que para este caso sólo se tiene en cuenta un ordenador portátil, con la amortización correspondiente.

Recurso HW	Coste (€)	% Dedicado	Dedicación (meses)	Periodo depreciación	Coste imputable
Ordenador Portátil	1000,00 €	100 %	12	60	200
TOTAL					200 €

Tabla A.4: Costes HW

Herramientas SW	716,94 €
Recursos humanos	26052,00 €
Amortización	200,00 €
Costes indirectos	5393,79 €
TOTAL (€)	32362,73 €

Tabla A.5: Resumen costes

Los costes indirectos que el proyecto haya podido acarrear, como puede ser el consumo de energía eléctrica, entre otros, han sido estimados en un 20 % del total del proyecto. Se obtiene por tanto, deducido de todas las tablas anteriores, un total de 32362,73 €como presupuesto final del presente proyecto.

Apéndice B

Especificaciones técnicas y parametrización

B.1. Requisitos del sistema

Este proyecto se ha realizado utilizando la tecnología MRDS. Por eso, es necesario adquirir las herramientas necesarias para probar la aplicación. Las herramientas necesarias son:

Microsoft Robotics Developer Studio 2008 Esta herramienta se puede adquirir a través del portal Microsoft Developer Network Academia Alliance (MSDNAA), si somos usuarios registrados. Para estudiantes de la Universidad Carlos III Madrid el enlace web es:

`http://msdn30.eacademy.com/elms/Storefront/Home.aspx?campus=ucarlos_info`.

Microsoft Visual Studio 2008 Igual que ocurre con la herramienta anterior, es posible descargar una versión académica de la misma a través de la página de MSDNAA.

B.2. Instalación

Para incorporar el entorno de simulación y el servicio implementado se ha de descomprimir el archivo pocima.zip, que contiene el código fuente, en la carpeta packages de MRDS y luego ejecutar DssProjectMigration.exe, indicando el path donde se ha descomprimido.

Además hay que añadir el archivo que contiene las texturas y mallas para completar la instalación del entorno de la universidad. Para ello hay que crear un directorio llamado pocima en "store/media" y descomprimir en esa carpeta el archivo pocima_3d_30_04_2010.zip.

El directorio donde se encuentra el código fuente debe contener las siguientes carpetas:

- **Campus**: que contiene el entorno de la universidad.
- **Carrito**: que contiene las especificaciones del vehículo con sus sensores incluidos.
- **Fusión**: que contiene la implementación de la fusión sensorial y muestra los resultados obtenidos.
- **PocimaDashBoard**: que contiene la implementación de un panel que muestra el control de mandos del coche y visualiza el barrido del láser.

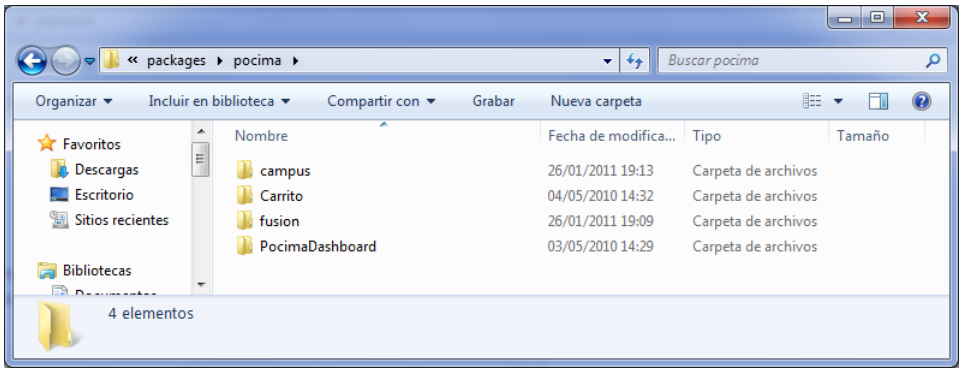


Figura B.1: Captura de pantalla del directorio packages/pocima.

B.3. Configuración y Ejecución

Para ejecutar el servicio de fusión se puede hacer abriendo el archivo fusion.cs en Visual Studio se pulsa F5 y se genera todo el entorno y el panel con la visualización de los procesos. Se debe pulsar el botón Inicio y se generará una solución para el escenario elegido. En la siguiente sección se explica cómo elegir el escenario o generar uno nuevo.

Otra opción es ejecutar el archivo desde la línea de comandos. En primer lugar se debe compilar con el siguiente comando desde el directorio correspondiente al proyecto:

```
msbuild fusion.csproj
```

Y el resultado es como se ve en la siguiente imagen.

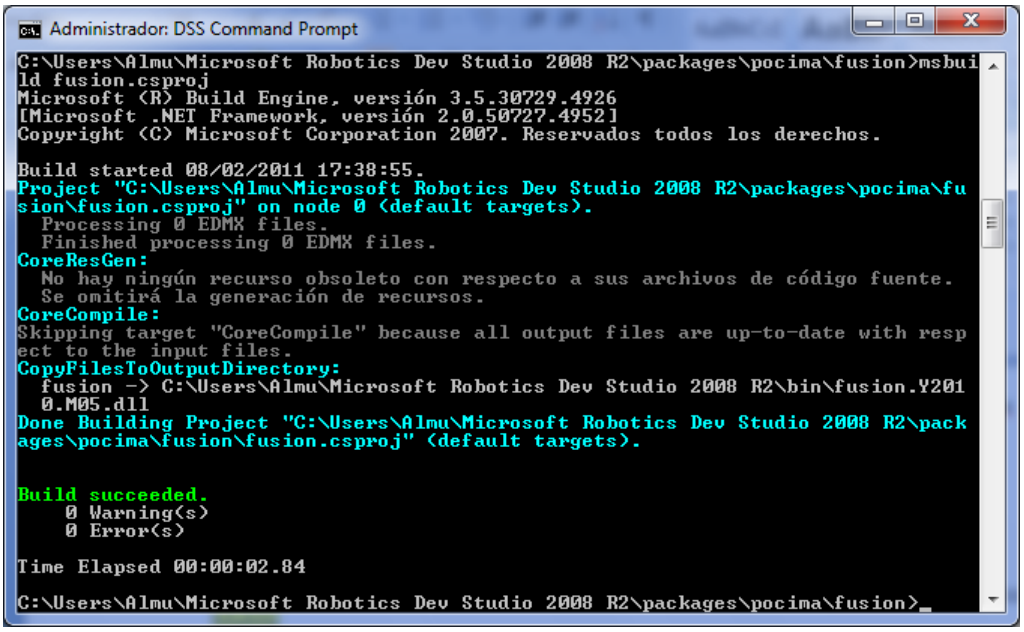


Figura B.2: Compilación del servicio fusión por línea de comandos.

A continuación se debe ejecutar el servicio utilizando el manifiesto fusion.manifest.xml con el siguiente comando:

```
dsshost /p:50000 /t:50001 /m:fusion.manifest.xml
```

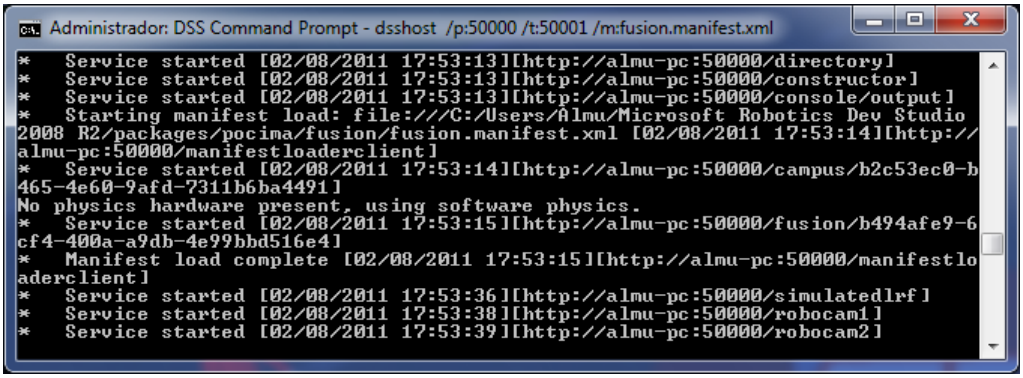


Figura B.3: Ejecución del servicio fusion desde línea de comandos.

B.3.1. Selección de Escenario

Para elegir el escenario se debe abrir el directorio "pocima/campus", abrir el archivo campus.sln en Visual Studio e ir, en el código, al método `CrearMundo()` en el cual se encuentra la sentencia `PonerCarrito()` que sitúa el coche en un punto del entorno dando por las coordenadas (x, y, z). En el método están presentes los cinco casos utilizados para las pruebas, pudiendo añadirse otros.

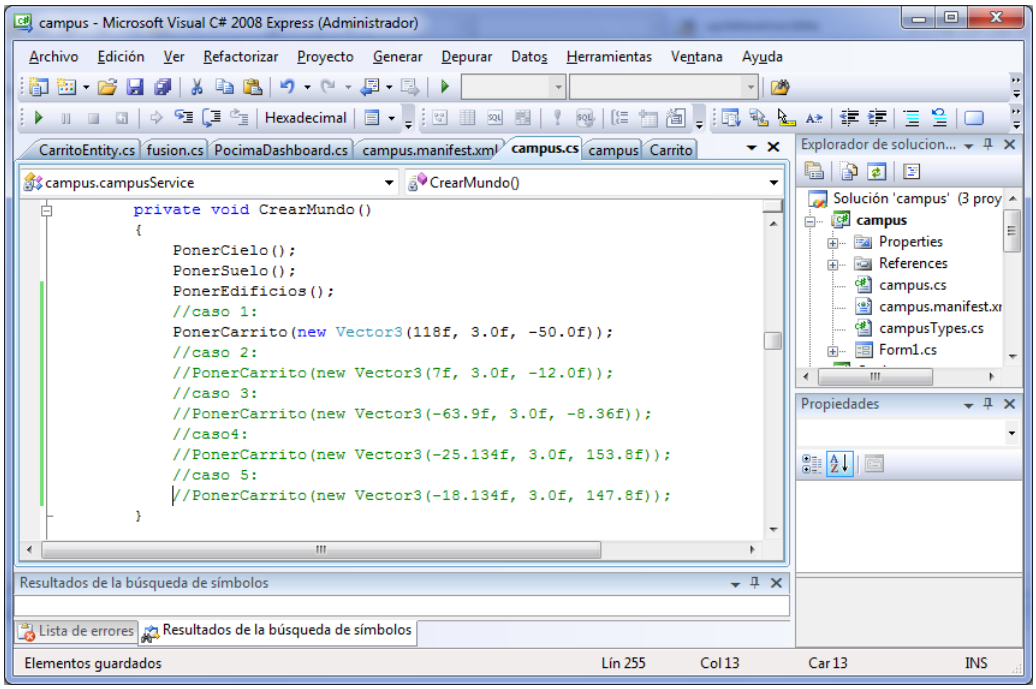


Figura B.4: Código de campus.sln.

B.3.2. Parámetros

Se ha definido una constante llamada `white_threshold`, que define el umbral de nivel de blanco en la imagen. Justo antes de realizar las transformaciones morfológicas que eliminan el ruido de la imagen, es necesario establecer que camino se va a seguir. Existen dos opciones, una en la que la limpieza es más agresiva y otra mucho más suave. Para ello se define esta constante que decide cuál de estas dos opciones elegir. Tiene un valor actual de 0.6, es decir, el 60 % de la imagen. Este valor se ha definido así según las pruebas realizadas.

Para cambiar los términos utilizados en la clasificación se debe ir al principio del archivo fusion.sln y modificar las cifras de las constantes definidas para establecer los valores máximos y mínimos:

- `min_height`: la altura mínima.
- `max_height`: la altura máxima.
- `min_width`: la anchura mínima.
- `max_width`: la anchura máxima.

Actualmente el rango es de 800mm. a 2400mm. para la altura de un peatón y entre 200mm. y 600mm. para la anchura.

Bibliografía

[1] http://www.accidentestrafico.es/espana-pais-con-mas-casos-de-peatones-fallecidos-por-habitante/ (último acceso 24/04/2010)

[2] *Multisensor Fusion: a minimal representation Framework*; Rajive Joshi and Arthur C. Sanderson; Intelligent Control and Intelligent Automation, Vol. 11; 1999.

[3] *Data Fusion and sensor integration: State of the art*; R.C. Luo y M. G. Kay; Data Fusion in Robotics and Machine Intelligence; 1992.

[4] *Data Fusion For Sensory Information Processing Systems*; J.J. Clark, A.L. Yuille; Kluwer Academia; 1990.

[5] *A multi-sensor approach for the protection of vulnerable traffic participants, the PROTECTOR project*; D.M. Garvrila, M. Kunert, Ulrich Lages; IEEE Instrumentation and Measurement Technology Conference; Mayo 2001.

[6] *Detection of road users in fused sensor data streams for collision mitigation*; L. Walchshäusl, R. Lindl, K. Vogel, T. Tatschke; Página 1, 10th International Forum on Advanced Microsystems for Automotive Applications, Berlin; Abril 2006.

[7] *An image processing system for driver assistance*; U. Handmann, T. Kalike, C. Tzomakas, M. Werner, W. V. Seelen; Image and Vision Computing Vol. 18 páginas 367-376; Abril 2000.

[8] *Pedestrian recognition in urban traffic using laserscanners*; K. Ch. Fuerstenberg, V. Willhoeft; 8th World Congress on Intelligent Transport Systems; 2001.

[9] *Diccionario de Psicología*; Editorial Oceano.

[10] *Reconocimiento de formas y visión artificial*; D. Maravall Gómez-Allende; Ed. ra-ma; 1993.

[11] *Visión por computador: Fundamentos y Métodos*; A. De la Escalera; Ed. Prentice Hall; 2001.

[12] *Professional, Microsoft Robotics Developer Studio*; Kyle Johns, Trevor Taylor; Ed. Wiley; 2008.

[13] Foro de MRDS: http://social.msdn.microsoft.com/Forums/es-ES/roboticsdss/threads

[14] videos de Microsoft Robotics Studio Developer Center: http://msdn.microsoft.com/es-es/robotics/bb383569

[15] http://es.wikipedia.org/wiki/Teorema_del_coseno (último acceso 02/11/2010).

[16] http://es.wikipedia.org/wiki/Teorema_del_seno (último acceso 02/11/2010).

[17] *Segmentación en imágenes basado en etiquetación de píxeles*; Gonzalo Luzardo; Blog de Multimedia y Visión Artificial; http://blog.espol.edu.ec/gluzardo/2009/04/10/segmentacion-de-imagenes-basada-en-etiquetacion-de-pixeles; Abril 2009.

[18] http://www.ine.es/jaxi/tabla.do (último acceso 29/12/2010)

[19] http://www.ine.es/jaxi/tabla.do?type=pcaxis&path=/t25/p442/histo/a1999/l0/&file=02082.px (último acceso 29/12/2010)

[20] http://landaben.dointeractiva.com/archivos/287-4-Pdf/MedAntrop.pdf (último acceso 29/12/2010)

[21] *A computational theory of human stereo vision*; D. Marr y T. Poggio; Mayo 1979.

[22] *Cooperative Fusion for Multi-Ostacles Detection With Use of Stereovision and Laser Scanner*; Raphaël Labayrade, Cyril Royere, Dominique Gruyer and Didier Aubert; Journal Autonomous Robots; Septiembre 2005.

[23] *Stereo and Neural Network-Based Pedestrian Detection*; Liang Zhao y Chuck Thorpe; IEEE Transactions on Intelligent Transportation Systems vol. 1 nº 3; Septiembre 2000.

[24] *Guide to the user requirements definition phase ESA Board for Software Standardisation and Control (BSSC)*; ESA PSS-05-02 Issue 1 Revision 1; Marzo 1995.

[25] *ESA software engineering standards, ESA(2) Board for Software Standardisation and Control (BSSC)*; ESA PSS-05-0 Issue 2; Febrero 1991.

[26] Tesis doctoral: *Detección de peatones en el espectro visible e infrarrojo para un sistema avanzado de asistencia a la conducción*; Cristina Hilario Gómez; Universidad Carlos III de Madrid; Departamento de Ingeniería de Sistemas y Automática; Octubre 2008.

Es mejor permanecer callado y parecer tonto que hablar y despejar las dudas definitivamente.

Groucho Marx