UNIVERSIDAD DE VALLADOLID

E.T.S.I. TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN

MENCIÓN EN TELEMÁTICA

Simulación de conducción con realidad virtual para el estudio de la seguridad y eficiencia del conductor

Autor:
D. Julián González Díaz

Tutor:
D. David González Ortega

Valladolid, 11 de Julio de 2017

TÍTULO:	**Simulación de conducción con realidad virtual para el estudio de la seguridad y eficiencia del conductor**
AUTOR:	**D. Julián González Díaz**
TUTOR:	**D. David González Ortega**
DEPARTAMENTO:	**Departamento de Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dña. Miriam Antón Rodríguez**
VOCAL:	**D. Mario Martínez Zarzuela**
SECRETARIO:	**D. David González Ortega**
SUPLENTE:	**D. Francisco Javier Díaz Pernas**
SUPLENTE:	**D. José Fernando Díez Higuera**

FECHA:	**11 de Julio de 2017**
CALIFICACIÓN:	

Resumen de TFG

El objetivo de este Trabajo Fin de Grado es el estudio del comportamiento de los conductores a través de realización de pruebas en un entorno simulado. Concretamente, el empleo de técnicas que permitan realizar una estimación de la dirección de la mirada a través de diferentes características obtenidas del movimiento de la cara. Los datos obtenidos de la estimación son almacenados, permitiéndonos conocer el comportamiento del conductor, además de evaluar cuál de las diferentes técnicas empleadas en esta tarea es más precisa. De esta forma, se pueden analizar los resultados obtenidos, corrigiendo fallos como despistes o educando a los usuarios en una conducción más preventiva y segura.

Este simulador se ha desarrollado con el motor de juegos Unity y el dispositivo para la detección del movimiento Kinect. Para la utilización del simulador, el usuario dispone del periférico Logitech G27, que incluye volante, pedales y cambio de marchas.

Palabras clave

Simulador, conducción, Unity3D, Kinect, seguridad vial y estimación de la mirada.

Abstract

The purpose of this Final Project is to study the behavior of the drivers thought making tests in a simulated environment. Specifically, the use of techniques which allow us to make a gaze estimation, using different characteristics obtained from the movement of the face. The data obtained from the estimation is stored, allowing us to know the behavior of the driver and to assess which of the different techniques employed in this task is more accurate. In this way, we could analyze the results, correcting fails such as dismissing or educating users in a more preventive and safer driving.

This simulator has been developed with the Unity game engine and using the Kinect motion detection device. To use the simulator, the user has the peripheral Logitech G27, which includes steering wheel, pedals, and gear lever.

Keywords

Simulator, driving, Unity3D, Kinect, road safety, and gaze estimation.

Agradecimientos

“Primero, a mi tutor David, por ofrecerme la oportunidad de realizar este TFG, y ayudarme en todo lo necesario para conseguirlo.

A mi familia, especialmente a mis padres y mi hermana, por acompañarme y apoyarme siempre en todo en estos años de carrera.

A mis amigos, tanto de toda la vida como a las nuevas amistades que he hecho durante esta etapa en Valladolid.

A todos los que han colaborado con el Proyecto realizando pruebas, gracias a los cuales este trabajo no hubiera sido posible.

A Javi y Rubén, por todos los ratos en el laboratorio y desayunos, que han hecho mucho más llevadero el trabajo.

A Ángel, que me ha aguantado toda la carrera, y cuya amistad ya estará para siempre.

A Rafa, cuya amistad me lleva acompañando desde siempre, por aguantarme todos estos años y saber sacarme una sonrisa.

Por último, A Aza, por todo el apoyo y todas esas conversaciones de risas que han hecho mucho más llevadero los últimos meses de intenso trabajo.”

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En esta primera sección, se realizará una introducción al documento. Primero, se tratarán las diferentes causas que motivan el desarrollo del trabajo, así como los objetivos que se buscan alcanzar con su realización. A continuación, se describirán tanto las diferentes fases como los métodos y medios empleados para la consecución del mismo. Por último, se comentará de forma breve la estructura del resto del documento, así como una breve descripción de los capítulos que lo componen.

1.1. Motivación

En la sociedad actual, la inmensa mayoría de personas posee carnet de conducir y utiliza el coche diariamente. La educación vial y la formación de los conductores son fundamentales para reducir tanto la frecuencia como la gravedad de los accidentes. Para ello, resulta una herramienta muy útil el empleo de simuladores de conducción. Estos nos permiten recrear situaciones conflictivas que se pueden dar en las carreteras reales, con las cuales poder entrenar desde los conductores sin experiencia hasta conductores más veteranos.

En los simuladores se puede mejorar el proceso de aprendizaje, gracias a que permiten a los usuarios novatos enfrentarse a situaciones particulares, frente a las cuales es fundamental la educación para que sepan afrontarlas en la realidad. Además, se pueden distinguir diferentes niveles de complejidad, enfocando dicho entrenamiento a los diferentes niveles de conocimientos que tenga el usuario. Gracias a esto, se puede preparar de forma controlada y sin ningún tipo de riesgo real a todo tipo de conductores, monitorizando todos los datos y pudiendo dar una retroalimentación mucho más completa que en un coche real.

Dentro de esta monitorización, resulta muy interesante conocer cuándo mira el usuario a los diferentes elementos durante dichas pruebas. Con esta información se puede conocer si el usuario es capaz de anticiparse a las diferentes situaciones, si revisa frecuentemente los espejos para conocer los elementos que le rodean, o si le afecta excesivamente el cansancio, entre otras. Esta información puede emplearse como retroalimentación para educar en una conducción más segura y preventiva.

Con ello se busca reducir las cifras de siniestralidad en la carretera. Pese a que estas han mejorado en los últimos años, todavía siguen siendo muy altas, y este trabajo con simuladores puede favorecer notablemente que se sigan reduciendo. Además, puede permitir conocer mejor a los conductores y aplicar dicha información a la mejora de los vehículos.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Grado es el desarrollo de diferentes métodos de estimación de la dirección de la mirada a través del movimiento de la cabeza mediante el uso de Kinect, con lo que desarrollar un método fiable para monitorizar a los usuarios en el simulador de conducción. De este modo, se pueden obtener más datos de los

mismos, y ofrecer una realimentación mucho más completa para mejorar su comportamiento.

Para ello partiremos de un escenario del simulador desarrollado previamente, el cual consistirá en una autovía con mucho tráfico, pensada para estudiar la anticipación y el conocimiento del resto de vehículos del entorno. Sobre él y empleando Kinect, se diseñarán y probarán diferentes métodos para estimar la dirección de la mirada a partir del movimiento de la cabeza. Toda la información de las pruebas será almacenada, para poder comparar su fiabilidad y evaluar los diferentes métodos. Para ello se empleará una población idéntica en las pruebas y se evaluarán los métodos de forma idéntica. Con ello se pretende decidir la mejor solución, de cara a su implementación en el simulador.

1.3. Fases y métodos

Las fases que se seguirán durante la realización del proyecto son las siguientes:

- Estudio sobre la seguridad vial, donde se revisará cierta información en cuanto a accidentes, conducción eficiente, etc. Con ello se pretende conocer el estado actual de la misma.
- Estudio acerca de diferentes simuladores de conducción ya existentes, de cara a estudiar cómo contribuyen a la seguridad vial y obtener cierta información útil a la hora de realizar el desarrollo.
- Análisis de las diferentes tecnologías existentes en el desarrollo de videojuegos, sobre los motores gráficos, de cara a comprender mejor la solución elegida y conocer sus virtudes y defectos.
- Estudio de las diferentes herramientas de detección del movimiento, de cara a elegir la más adecuada que se ajuste a nuestras necesidades, de forma motivada.
- Estudio de diferentes sistemas para reconocimiento de patrones, de cara a implementarlos de forma funcional en los métodos de estimación de la dirección de la mirada a partir de los datos de la cabeza como una posible solución.
- Desarrollo de los diferentes métodos de estimación de la dirección de la mirada, desde la explicación de su base teórica hasta el funcionamiento detallado y una guía de usuario.
- Realización de pruebas para los diferentes métodos, evaluando su fiabilidad en diferentes situaciones, y una comparativa que permita conocer de forma motivada el más interesante de implementar en el simulador.
- Desarrollo de las conclusiones y estudio de posibles líneas futuras, de cara a la mejora de los métodos y la ampliación del trabajo realizado.

1.4. Medios

Durante el desarrollo de este trabajo, se ha dispuesto de dos equipos, uno de trabajo y otro para la realización de pruebas. El de trabajo tiene las siguientes características:

- Procesador: Intel Core I7-4720HQ CPU @ 2.6GHz
- Memoria RAM: 8 GB DDR4
- Sistema Operativo: Windows 10 (64 bits)
- Tarjeta Gráfica: Nvidia GeForce GTX 860M

Las características del empleado en la realización de pruebas son:

- Procesador: Intel Core I7-4510U CPU @ 2.0GHz 2.6GHz
- Memoria RAM: 8 GB DDR3
- Sistema Operativo: Windows 10 (64 bits)
- Tarjeta Gráfica: Nvidia GeForce GTX 780

Además, se ha empleado el dispositivo Kinect, para realizar el seguimiento del movimiento de la cabeza. Para la conducción, se empleará el periférico Logitech G27, el cual está formado por volante, pedales y palanca de cambios.

1.5. Estructura de la memoria

La memoria se estructura en diferentes secciones, las cuales serán expuestas en los siguientes párrafos.

En el primer capítulo, se ve una introducción al trabajo, y las diferentes partes de las que consta.

En el segundo capítulo se tratará el tema de la seguridad vial, el cual es la motivación principal para el desarrollo del simulador de conducción. En ella se tratarán los diferentes conceptos básicos, como definiciones de seguridad vial o tráfico. A continuación, se estudiarán las estadísticas de accidentes en el año 2016, realizando una explicación de los diferentes factores que influyen en los mismos, y cómo se relacionan. Después se realizará una exposición de los diferentes tipos de accidentes, concluyendo con una serie de medidas útiles que ayudan a reducir su frecuencia y gravedad. Por último, se tratará el tema de la conducción eficiente, explicando en qué consiste y por qué es tan relevante en la actualidad, concluyendo con una serie de pautas para conseguir una conducción más respetuosa con el medio ambiente.

En el tercer capítulo, se tratarán diferentes ejemplos de simuladores de conducción que se emplean en la actualidad en diferentes ámbitos, como en prevención de riesgos laborales o educación. Se expondrán las diferentes características de cada uno, explicando brevemente su funcionamiento. Por último, y utilizando los mismos como ejemplo, se realizará un breve explicación sobre cómo estos contribuyen al aprendizaje de la seguridad vial.

En el cuarto capítulo, se estudiarán diferentes motores gráficos, de cara a evaluar el más apropiado para el desarrollo del simulador. Se realizará primeramente una pequeña definición de sus fundamentos generales. A continuación, se expondrán diferentes soluciones disponibles, detallando todas sus ventajas e inconvenientes. Por último, se realizará una comparativa que permitirá elegir la opción más adecuada de forma motivada.

En el quinto capítulo, se estudiarán algunos dispositivos de detección de movimiento disponibles en el mercado, para el seguimiento de los movimientos de la cabeza del usuario. Se explicarán de forma detallada todas sus características, tanto a nivel de hardware como de software disponible. Por último, se realizará una comparativa en función de las ventajas y desventajas de cada uno, que nos permitirá elegir de forma razonada la opción más adecuada.

En el sexto capítulo, se tratará el tema del reconocimiento de patrones, exponiendo de forma teórica el funcionamiento de los clasificadores software. Primero, se realizará una explicación teórica de qué es un patrón, y las relaciones entre sus características. Después, se realizará una breve explicación de los diferentes métodos de normalización de las mismas. En el siguiente paso, se expondrá de forma general la teoría relativa a este tipo de clasificadores. Por último, se expondrán los tipos de clasificadores, según sus características de funcionamiento, haciendo una explicación detallada de los dos clasificadores concretos usados en el proyecto.

En el séptimo capítulo, se mostrarán de forma detallada los diferentes métodos desarrollados para la estimación de la dirección de la mirada a través de patrones de características obtenidos del movimiento de la cabeza. Primero, se mostrará cómo se lleva a cabo la división de la pantalla en regiones. A continuación, se explicarán los diferentes métodos, comenzando por el preprocesamiento de las características utilizadas, prosiguiendo con la explicación de la calibración necesaria y la construcción del decisor, y finalizando con una muestra del funcionamiento general y un manual de usuario. Para los métodos que emplean clasificadores, se realizará una explicación motivada de los patrones empleados y del método de entrenamiento, seguida de una exposición de la configuración de los mismos, para concluir con el funcionamiento general y un manual de usuario.

Finalmente, en el octavo capítulo, se realizará un pequeño estudio económico sobre el coste necesario del proyecto.

En el noveno capítulo, se comentarán las conclusiones extraídas el trabajo, y se desarrollarán diferentes líneas futuras que permitan continuar con el desarrollo y mejora del simulador de conducción.

2. Seguridad vial y conducción eficiente

En este capítulo se va a tratar algunas cuestiones relacionadas con la seguridad vial que pueden ser entrenadas mediante simuladores de conducción. Además, se van a exponer normas para prevenir accidentes y actuar correctamente ante situaciones de riesgo. Por último, se va a tratar el tema de la conducción eficiente y cómo puede llevarse a cabo tanto para una reducción de la contaminación como para mejorar la seguridad en la conducción.

2.1. Situación actual de la seguridad vial

En la actualidad, podemos encontrar dos grandes retos a los que se enfrenta el conjunto de conductores tanto en España como muchos otros países de Europa: reducir el número de accidentes y la gravedad de los mismos, y reducir la contaminación que agrava tanto el problema del calentamiento global como el aumento del número de personas que padecen enfermedades respiratorias en las grandes ciudades.

Según la Dirección General de Tráfico, organismo encargado de regular la circulación de vehículos, el número de accidentes en los últimos años ha disminuido de forma progresiva, aunque en los años 2014 y 2015 las cifras de víctimas mortales se han estancado en torno a 1.680 por año. Además, la letalidad ha descendido notablemente hasta el orden de 1.2 puntos, indicando que de cada 100 heridos solamente 1 fallece [1].

Si bien estas cifras son positivas, todavía siguen siendo muy elevadas, y se han propuesto numerosos planes de actuación como campañas publicitarias para concienciar a la población, aumento de la cuantía de las multas, cursos de educación en la conducción u otros. Cabe destacar que, si bien hay diferentes factores que intervienen de forma interrelacionada en los accidentes de tráfico, en la mayoría de ocasiones la responsabilidad del conductor es el factor principal, siendo la principal causa de accidentes mortales. Por ello, la educación de conductores más responsables y conscientes es una prioridad para reducir las cifras de siniestralidad en las carreteras [2].

El segundo reto se refiere a la conducción eficiente, entendida como *"el modo de conducir el vehículo que tiene como objetivo lograr una bajo consumo de carburante a la vez que reducir la contaminación ambiental"* (D.G.T., 2016). Cada día esto cobra más importancia debido al alto porcentaje de contaminación que existe en las ciudades y al aumento del efecto invernadero. Las ventajas de esta forma de conducir son notables y afectan al día a día de los conductores. Por ejemplo, reducir el estrés provocado por la conducción, ahorro de carburante que se traduce en un ahorro económico a corto plazo, mejora de la conservación del vehículo que se traduce en un ahorro económico a largo plazo y reducción de la contaminación que mejora la calidad del aire respirado [3].

2.2. Accidentes y medidas de seguridad vial preventivas.

"Los accidentes de circulación representan actualmente uno de los principales problemas de salud pública con el que se enfrentan las sociedades modernas y su importancia en el futuro será creciente según la Organización Mundial de la Salud"

(D.G.T., 2016). Con esta afirmación se puede entender lo grave que es el problema de los accidentes de tráfico en la sociedad actual. Según se puede ver en el capítulo anterior, la tasa de mortalidad en accidentes de tráfico ha descendido progresivamente, gracias a que los conductores está mucho más concienciados, la calidad de los sistemas de seguridad es mayor y está mucho más extendidos, y a la mejoras de las vías de circulación, principalmente. Sin embargo, esta mejora no es suficiente y se debe continuar en la misma línea de cara a reducir estas cifras. En el año 2014 fallecieron 1.688, distribuidos como se muestra en la Tabla 2.1 [4].

USUARIOS DE LA VÍA	TOTAL	PORCENTAJE
CONDUCTOR	1.043	62%
PASAJERO	309	18%
PEATÓN	336	20%
TOTALES	1.668	

Tabla 2.1: Fallecidos en función de su condición en 2016

Para comprender de forma adecuada el motivo de las cifras actuales y poder llevar a cabo actuaciones que las reduzcan, debemos entender cuáles son las causas principales de los accidentes con víctimas mortales. *"Un accidente sobreviene, como resultado de una concurrencia desfavorable de múltiples factores, en un momento y lugar determinado"* (D.G.T., 2016). Estos factores se pueden agrupar en el llamado triángulo de la seguridad vial: la vía, el vehículo y el conductor; como se ve en [4].

Figura 2.1: Triangulo de seguridad vial

De estos tres factores, el que aparece de forma mayoritaria en los accidentes con víctimas mortales, es el factor humano, debido a una infracción de las normas de circulación o a un estado inadecuado para conducción de forma segura, como somnolencia o exceso de confianza. Esto pone en evidencia que, si bien no es el único motivo que provoca accidentes de tráfico, es el mayoritario, y por tanto la educación y concienciación de los conductores es una tarea fundamental para la mejora de la seguridad vial y la reducción de los accidentes de tráfico, con o sin víctimas mortales [4].

La función del conductor, como persona responsable del vehículo que maneja, es recabar la máxima información posible del entorno y procesarla para actuar en consecuencia. Este proceso lleva un tiempo, que puede ser variable en función de las condiciones climatológicas y de visibilidad, el estado de la vía, el estado del vehículo o el propio estado del conductor. Una vez recabada la información suficiente, la transforma en decisiones que al mismo tiempo se traducen en órdenes a sus pies y manos para dar una respuesta adecuada (acelerar, frenar, encender las luces, etc.). Cuando la información que recibe el conductor es excesiva o su capacidad de percepción está afectada por su estado psicofísico, la capacidad de respuesta se ve afectada negativamente y pueden surgir los errores que sobrevengan en un accidente.

En consecuencia, la capacidad del conductor se ve influida por los siguientes factores [4]:

- El estado físico, que afectará a su capacidad de percepción y de reacción ante los diferentes estímulos.
- El estado psíquico, que afecta a su concentración y a su toma de decisiones.
- El nivel de alerta, que provocará que en ciertas situaciones no recabe toda la información suficiente para llevar a cabo una toma de decisiones adecuada.
- El conocimiento de la normativa, que debe ser preciso para tener un comportamiento adecuado en cada situación.
- La competencia técnica, que dependerá de su habilidad como conductor, la experiencia, los buenos y malos hábitos adquiridos, o la asiduidad con la que conduce, entre otros factores.

Se debe mantener un equilibrio, en el cual tanto el conductor como el vehículo tengan en todo momento capacidad suficiente para obtener toda la información necesaria y procesarla de forma adecuada. Si este equilibrio se rompe, y las demandas del medio son excesivas para el conductor y el vehículo, estos no podrán obtener ni procesar la información de forma adecuada, lo que llevará a errores que en última instancia pueden provocar un accidente. Esto se refleja en la Figura 2.2, que muestra un diagrama llamado gráfica de Blumenthal y en el que se observa cómo la variación entre las exigencias del medio no debe superar nunca las capacidades del conductor y vehículo. En los casos que

esto ocurre, se producen errores que aumentan de forma exponencial las probabilidades de accidente [4].

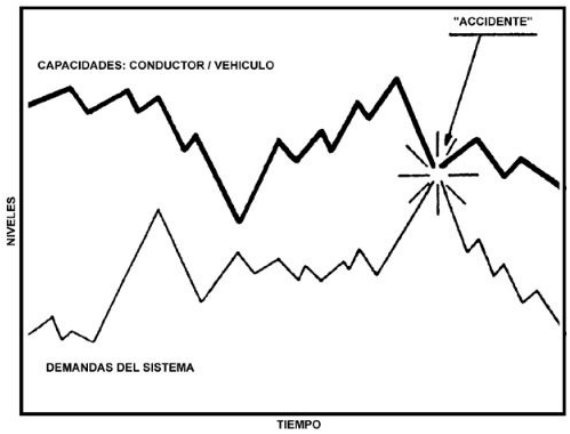

Figura 2.2: Diagrama de Blumenthal

Una vez comprendidas qué causas tienen los accidentes y cómo el factor humano puede afectar a los mismos, se debe analizar el accidente en sí. Se entiende como accidente *"cualquier evento como resultado del cual el vehículo queda de manera anormal, dentro o fuera de la carretera, o produzca lesiones en las personas o daños a terceros"* (D.G.T., 2016). Entonces se puede entender como accidente cualquier evento que provoca que el vehículo lleve a cabo maniobras de forma anormal de forma persistente, que acaben incurriendo en daños tanto personales como a terceros, en vehículos, personas u otros. Las maniobras anormales transitorias que no provoquen daños, se entenderán como infracciones de tráfico, por incumplimiento de las normas de circulación, y serán sancionadas en consecuencia [4].

Deben de cumplirse tres requisitos para que un accidente de tráfico sea considerado como tal:

- Que se produzca en una vía abierta a la circulación pública o tenga en ella su origen.
- Que se produzcan daños de cualquier tipo tanto personales como a terceros.
- Que al menos esté implicado en el mismo un vehículo en movimiento.

Hay diferentes clasificaciones para los accidentes, de las cuales cabe destacar la que refleja el modo en el que se producen [4]:

- Colisiones. Encuentros entre dos o más vehículos.
    - Frontales. Choque entre dos vehículos en su parte frontal
    - Embestidas. Un vehículo choca lateralmente con otro.
    - Reflejas. Varias colisiones sucesivas entre sí.
    - Por alcance. La parte frontal de un vehículo choca contra la parte posterior de otro.

    - Por raspado. Roce entre los laterales de ambos vehículos.
    - Múltiples. Cuando intervienen varios vehículos
- Choques. Cuando un vehículo impacta contra un elemento fijo de la vía.
- Salidas de la vía. El vehículo se sale de la calzada.
    - Despeñamientos. El vehículo cae por un desnivel
    - Vuelcos. El vehículo vuelca, tanto de campana como de tonel.
- Atropellos. Colisiones entre dos unidades de circulación entre las que existe una desproporción manifiesta.

Además, en un accidente existen tres fases bien diferenciadas, que condicionan el mismo y durante las cuales los diferentes agentes intervienen de formas diferentes [4]:

- Fase de precepción. En este momento el conductor o conductores debían haber percibido el peligro, como un hecho anormal que requería una respuesta adecuada, pero este se percibió de forma errónea o en un momento tardío para responder de forma adecuada al mismo.
- Fase de decisión. En ella los accidentados toma decisiones, de forma rápida, de cara a intentar minimizar el accidente. Esto se lleva a cabo a través de la realización de maniobras evasivas o empleo de posturas corporales que minimicen los daños del mismo.
- Fase de conflicto. En este parte se produce la culminación del accidente, en la cual se produce la colisión o la salida de la vía y en la cual el conductor ya no tiene ningún tipo de control para modificar la situación ni minimizar los daños.

Por último, y habiendo tratado tanto las causas de los accidentes como habiendo llevado a cabo un análisis de los mismos, se deben señalar una pautas adecuadas de comportamiento que mejoren la seguridad vial y minimicen tanto la probabilidad de accidentes como los daños una vez estos se ha producido. Estas pautas permitirán a los conductores un aumento de la seguridad en la conducción. Algunas medidas que deben tener en cuenta los conductores son [4]:

- Revisar de forma periódica y adecuada el estado del vehículo, para que responda de la forma más rápida y adecuada posible a las órdenes del conductor.
- El equipamiento de accesorios adecuados en función de las condiciones meteorológicas y la época del año.
- Fijarse un calendario de revisiones sistemáticas de diferentes partes del vehículo en función de la asiduidad con la que requieran ser revisadas.
- Estar en condiciones psicofísicas apropiadas para una correcta conducción del vehículo y una respuesta adecuada a los estímulos.

- Realizar descansos periódicos y no conducir durante largos periodos de tiempo, debido a que la fatiga reduce de forma drástica el tiempo de reacción.
- Tratar de ser conscientes de las limitaciones de cada persona, enfrentándose como conductor ante aquellos retos que sean viables y ante los que se esté preparado.
- Evitar la rutina que provoca malos hábitos, llevando una reflexión continua de mejora que permita al conductor mejorar día a día.
- Previsión del viaje a través de una adecuación específica del vehículo, consulta de la previsión meteorológica y otras que permitan al conductor conocer de antemano las posibles dificultades y estar preparado ante las mismas.
- Tratar de dar una respuesta positiva ante problemas como conflictos con otros conductores o situaciones de tráfico adversas, que ayuden a una conducción con bajos niveles de estrés.
- Cumplimiento de todas las normas de tráfico, aunque algunas en principio parezcan poco importantes o incluso erróneas, ya que tienen un motivo justificado y el no hacerlo reduce la seguridad.

2.3. Conducción eficiente y sostenible.

"La "conducción eficiente" es un nuevo modo de conducir el vehículo que tiene como objetivo lograr un bajo consumo de carburante a la vez que reducir la contaminación ambiental. A su vez se obtiene un mayor confort en la conducción y una disminución en los riesgos en la carretera." (D.G.T., 2016). Con esta definición se aproximan de forma bastante clara las pautas que deben regir una conducción para ser considerada eficiente, y los objetivos de la misma. Pero, ¿por qué es importante una conducción eficiente?

En la actualidad, el exceso de agentes contaminantes que expulsamos a la atmósfera en las distintas actividades que realiza el ser humano, como fábricas, automóviles o chimeneas de calefacción, es una de los mayores problemas a los que nos enfrentamos. En las grandes ciudades, las emisiones nocivas que provienen de los coches son las causantes del elevado nivel de contaminación que existe. Esta mala calidad del aire provoca una peor calidad de vida y un aumento de la aparición de enfermedades respiratorias, que agravan la salud de muchas personas. Además, se provoca un incremento del "efecto invernadero", que afecta al clima y agrava el problema del cambio climático. Por ejemplo, en Madrid se ha tenido que regular el número de conductores, limitándolo de diferentes formas en el centro de la ciudad, debido a los altos niveles de contaminación, como se puede observar en la Figura 2.3. Todos estos problemas provocan que cada vez se le dé más importancia a la conducción eficiente, la sensibilización de la población y la fabricación de vehículos más ecológicos [3] [5].

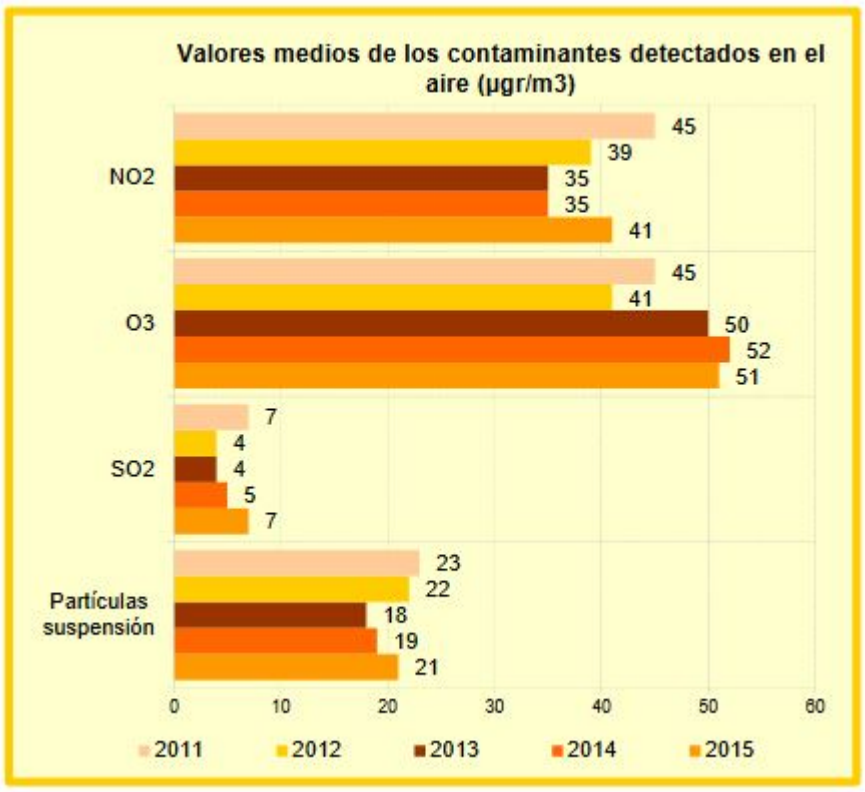

Figura 2.3: Valores medios de contaminación en Madrid.

En consonancia con este problema, los fabricantes de vehículos han desarrollado modelos mucho más eficientes, que reducen el consumo aumentando la eficiencia. Además, muchas marcas ya desarrollan, incluso de forma comercial, modelos híbridos, que reducen de forma drástica el consumo gracias a la incorporación de un motor eléctrico que apoya al de combustión convencional, principalmente a velocidades bajas, como las que se dan en una circulación por ciudad. Incluso empiezan a aparecer modelos de coches completamente eléctricos, que no producen ningún tipo de emisiones contaminantes. Sin embargo, con la tecnología actual, todavía no son viables debido a problemas como la falta de estaciones de carga suficientemente distribuidas y rápidas, para poder realizar viajes de larga duración y a la corta vida de las baterías que no permiten una autonomía suficiente, entre otros [3].

En los vehículos actuales podemos encontrar nuevos sistemas que favorecen una reducción del consumo y ayudan al conductor en este sentido [3]:

- Los cambios automáticos puede ayudar a reducir el consumo, ya que están preparados para optimizar el rendimiento. De la misma forma, muchos vehículos con cambio de marchas manual incorporan sistemas que indican al conductor cuándo debe cambiar de marcha para ahorrar combustible.
- El uso de turbocompresores aumenta la potencia y el rendimiento de los motores sin aumentar el consumo.
- El control electrónico de los motores actuales permite mejorar su rendimiento.

En contraposición, hay elementos de los vehículos y ciertas costumbres muy extendidas en los conductores, que provocan un aumento en el consumo y que en muchos casos son evitables. Algunos de estos elementos son [3]:

- El aire acondicionado. El llevar puesto el aire acondicionado, si bien mejora la comodidad durante el viaje, es uno de los factores con mayor incidencia en el consumo de combustible. Se recomienda mantener una temperatura en el vehículo de entre 21º-22º.
- Las ventanillas. Conducir con las ventanillas bajadas provoca una reducción de la eficiencia aerodinámica en el vehículo, lo que redunda en una mayor resistencia al aire provocando un aumento del consumo.
- Diagnosis del motor. Conviene revisar el estado del motor periódicamente para detectar averías ocultas que aumenten el consumo.
- Control de niveles y filtros. Los niveles y filtros deben estar en condiciones óptimas para no provocar un aumento del consumo.
- Control de la presión de los neumáticos. La falta de presión en los neumáticos provoca un aumento del rozamiento contra el pavimento, aumentando la potencia necesaria para mover el vehículo y reduciendo la eficiencia en el consumo.
- El uso de baca. Debido a la resistencia aerodinámica de la baca y del equipaje que contenga, se aumenta notablemente el consumo de combustible.

Debido a todos estos factores, además del estilo de conducción del conductor, el consumo del vehículo puede elevarse notablemente provocando un aumento de las emisiones y encareciendo los desplazamientos. Un conjunto de claves y pautas que se deben poner en práctica para llevar a cabo una conducción eficiente, entre otras, pueden ser [3]:

- En cuanto a la puesta en marcha.
  - Arrancar el motor sin pisar el acelerador.
  - En los motores diésel, esperar unos segundos antes de iniciar la marcha.
  - En los motores de gasolina, iniciar la marcha justo tras arrancar el motor.
  - Realizar el calentamiento del motor en movimiento.
- Usar la primera velocidad solo en el inicio de la marcha, cambiando lo antes posible a segunda.
- Circular el mayor tiempo posible con marchas más largas, reduciendo las revoluciones del vehículo.

- Llevar a cabo una conducción preventiva, tratando de anticiparse en todo momento para evitar los frenazos y maniobras bruscas, buscando una circulación fluida y emplear el uso del freno motor cuando sea posible.
- Una distancia de seguridad adecuada, que nos permita tener más tiempo de reacción y poder prever de forma anticipada los imprevistos que puedan surgir.
- Bajar las pendientes con la marcha engranada y dejar que el coche se mueva con su propia inercia. Nunca se debe bajar una pendiente en punto muerto porque además de incrementar el consumo resulta extremadamente peligroso.
- Al circular en caravana, circular en la marcha más larga posible aumentando la distancia de seguridad, lo cual nos permitirá hacer cambios bruscos de velocidad y circular con mayor fluidez reduciendo el consumo.
- Parar el motor en aquellas paradas superiores a un minuto, ya que a partir de este tiempo el consumo que se produce al ralentí es mayor que el que supone volver a arrancar el motor.

Para finalizar, se van a destacar las ventajas que tiene, tanto a corto como a largo plazo, el empleo de técnicas de conducción eficiente. Estas ventajas afectan tanto al conductor en sí, ya que suponen un ahorro económico en combustible a corto plazo y una mejor conservación del vehículo a largo plazo; al resto de conductores, minimizando las conductas de riesgo; como al resto de personas y al planeta en sí mismo, reduciendo la contaminación y mejorando la calidad del aire. Este conjunto de ventajas se pueden ver como [3]:

- Mejora del confort de la conducción y disminución de la tensión. Ya que se reducen los movimientos bruscos y los ruidos provenientes del motor, esta conducción es mucho más cómoda tanto para el propio conductor como para los pasajeros. Este estilo de conducción es mucho más tranquilo, reduciendo los estados de estrés y redunda en una reducción de los conflictos con otros conductores y del riesgo y la gravedad de los accidentes.
- Ahorro económico de combustible y prolongación de la vida del vehículo. El estilo de conducción preventiva evita todos los cambios de velocidad bruscos y busca anticiparse lo antes posibles frente a obstáculos. Esto reduce el consumo suponiendo un ahorro de carburante notable incluso a corto plazo, además de provocar que los diferentes elementos del vehículo estén sometidos a menos esfuerzos. Esto aumenta la vida útil de los mismos y, por tanto, la reducción de los costes de mantenimiento del mismo y el aumento de la vida útil del coche.
- Reducción de la contaminación urbana que mejora la calidad del aire respirado. La emisión de gases contaminantes empeora la calidad del aire

en las ciudades, donde existe una gran concentración de vehículos que circulan la gran mayoría del tiempo. Esto reduce la calidad de vida de las personas y provoca un aumento en la aparición de enfermedades respiratorias. La reducción de los mismos es muy beneficiosa debido a que no solo se mejora la calidad de vida en los núcleos urbanos, sino también se reducen los problemas de calentamiento global que provocan el cambio climático mundial.

- Incremento en la seguridad vial.
    - Mantener una conducción preventiva aumentando la distancia de seguridad que permite disponer de un mayor tiempo de reacción para anticiparse a obstáculos.
    - Mejorar la velocidad media constante durante los viajes reduciendo los cambios de velocidad.
    - Mejorar la atención y concentración en la conducción, mejorando el tiempo de reacción y la prevención de accidentes.

3. Simuladores de conducción

Gracias al desarrollo de los sistemas informáticos y también, en gran medida, al desarrollo y expansión de los videojuegos en la actualidad, se han desarrollado numerosos programas para aprender con entornos simulados. Para la conducción, existen numerosos simuladores que permiten a los usuarios aprender y mejorar sus habilidades mediante simulaciones, lo cual resulta muy útil debido a que permite practicar sin la necesidad de un vehículo real, con todo el gasto que este supone. Además, permiten practicar situaciones de riesgo, las cuales no se pueden entrenar de otra forma, y permite a los conductores estar preparados y saber cómo reaccionar ante las mismas. Estos simuladores son un gran aporte para la seguridad vial, gracias a que permiten entrenar a los conductores sin ningún riesgo y de forma automatizada.

3.1. Soluciones propietarias

A continuación, se van a estudiar un conjunto de simuladores desarrollados por empresas privadas. Estos solo son algunos de los más conocidos entre la amplia gama de ofertas que existen en el mercado y están enfocados en diferentes ámbitos, como prevención de riesgos laborales o aprendizaje en autoescuelas.

3.1.1. Simuladores del grupo OTP

El grupo OTP es una empresa dedicada a la prevención de riesgos laborales en España, y se encarga de dar servicios para otras empresas en temas de consultoría, seguridad y otros temas relacionados. En este ámbito, han desarrollado un simulador de conducción que permita reducir la siniestralidad de los trabajadores en las carreteras [6].

Entre todos los accidentes de tráfico del año 2014, unos 482.013 fueron en horario laboral. Debido a estos datos, se detectó la necesidad de que la seguridad vial estuviera más presente en las empresas. Esto motivó al grupo OTP a desarrollar su simulador de conducción, que se ve en la Figura 3.1, como solución para empresas, que permitiera educar y entrenar a los trabajadores [7].

Figura 3.1: Simulación en el simulador del grupo OTP.

Este simulador permite adquirir competencias de conducción en todo tipo de situaciones, como la conducción en condiciones climatológicas adversas o en situaciones de tráfico denso. Además, permite mejorar de forma cómoda la precisión, velocidad y calidad de las destrezas empleadas en la conducción [7].

Dentro del simulador, podemos encontrar dos modalidades diferenciadas: la modalidad libre y ejercicios preestablecidos [7]:

- En la modalidad libre, se permite una selección previa de las condiciones, como pueden ser el tipo de escenario (rural, urbano, etc.), el tipo de vehículo y el clima (soleado, lluvia, niebla, nieve o aleatorio). El escenario se ajustará a dichas características y se permitirá sobre él la conducción libre, en la que se sucederán situaciones como las que se dan habitualmente cuando los empleados se desplazan con el vehículo.
- En los ejercicios preestablecidos, se dará la posibilidad de escoger entre 18 situaciones diferentes con un nivel de complejidad elevado, como realizar una frenada de emergencia, conducción sin sistema ABS o incorporaciones con tráfico denso, entre otras.

3.1.2. City Car Driving

Es un simulador de conducción con licencia de pago enfocado en la conducción realista en ciudad. Maneja un gran número de situaciones del tráfico y emplea físicas y gráficos de alta calidad para simular una conducción lo más cercana posible a la realidad. La gran ventaja que tiene es que permite a los usuarios añadir sus propios modelos de coches y modificar las físicas de los mismos, lo cual permite que sea completamente personalizable y expansible. Gracias a esto se puede emplear para entornos muy concretos, ya que se adapta a los mismos con mucha facilidad.

Entre sus principales características, se puede destacar [8]:

- Soporte multilenguaje para numerosos idiomas.
- Elección de las normas de conducción europeas y estadounidenses.
- Soporta conducción por la izquierda y por la derecha.
- Conducción diurna y nocturna.
- Numerosas condiciones climáticas.
- Genera situaciones de peligro y rutas de forma aleatoria.
- Permite un modo de grabación para poder analizar la simulación.
- Contiene 24 ejercicios con varios niveles de dificultad.
- Soporta conducción con cambio de marchas manual y automático.
- Maneja peatones, que generan diferentes situaciones de riesgo.
- Permite el empleo de Oculus Rift para una simulación superrealista.

Entre los mapas disponibles hay dos grandes ciudades completas, con diferentes niveles de dificultad, que además contienen autódromos con diferentes ejercicios de conducción defensiva ante situaciones de riesgo. Estas ciudades están diseñadas con

barrios diferentes con calles variadas, para que la conducción sea variada y el conductor se enfrente a diferentes tipos de vías y de situaciones, como se observa en la Figura 3.2.

Los vehículos son de diferentes tipos y tamaños, y además tienen versiones tanto para conducción por la izquierda como por la derecha. Todos están totalmente equipados con el sistema de controles, sonidos y luces para un manejo completamente realista. Además, los usuarios pueden añadir nuevos modelos, aunque se debe destacar que no se aporta ningún tipo de soporte oficial para los mismos [8].

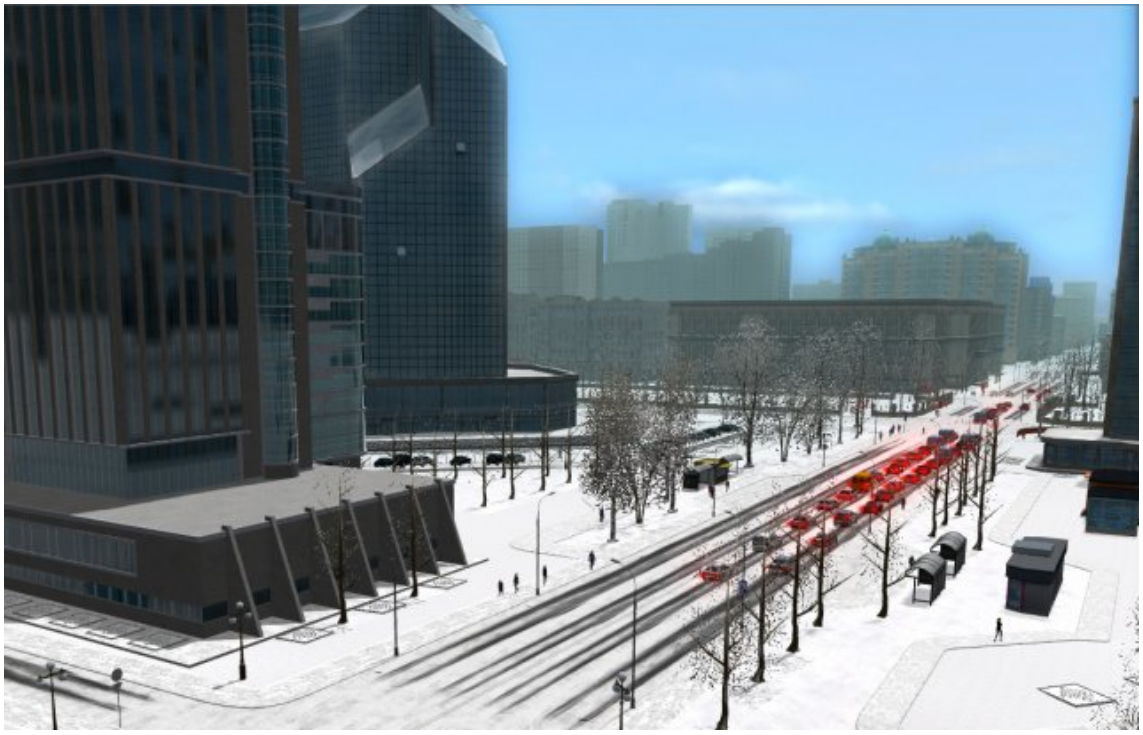

Figura 3.2: Simulador City Car Driving

Las misiones de conducción se pueden dividir en dos tipos: conducción libre y ejercicios de conducción defensiva [8].

La conducción libre permite al usuario moverse de forma libre por el escenario, enfrentándose a peligros de forma aleatoria. Esta modalidad tiene varios niveles de dificultad, que evalúan de forma progresiva la habilidad del conductor y su conocimiento de las normas de circulación. Los niveles de dificultad elevados se desbloquean tras superar de forma satisfactoria el nivel anterior, al igual que ciertos vehículos poco comunes. En el máximo nivel, se permitirá la elección de conducción plenamente libre, sin requerir el seguimiento de una ruta aleatoria preestablecida [8].

Los ejercicios de conducción defensiva se desbloquean al superar todos los niveles del modo libre. Estos consisten en una serie de pruebas de habilidad en la conducción ante situaciones de riesgo en los que se permite al conductor mejorar su habilidad con maniobras defensivas como frenado de emergencia, eslalon sobre superficie resbaladiza, etc. [8].

Como conclusión, este simulador nos permite entrenar todo tipo de diferentes situaciones y mejorar la calidad de la conducción. Además, la práctica de maniobras

evasivas como las que simula, permite a los usuarios entender cómo deben reaccionar ante diferentes situaciones de peligro, colaborando en reducir el número de accidentes y la gravedad de los mismos. Por último, cabe destacar que este simulador es extensible, permitiendo a usuarios avanzados establecer nuevas configuraciones para situaciones concretas de forma relativamente sencilla.

3.1.3. DriveSim

DriveSim es un simulador de conducción pensado para la educación, en ámbitos como autoescuelas, centros de formación, cursos de prevención de riesgos laborales u otros eventos similares. Está diseñado con orientación al aprendizaje, tanto para conductores novatos como para aquellos experimentados que deseen mejorar sus habilidades [9].

Este simulador realista nos permite practicar en un entorno libre de riesgos, realizando numerosos ejercicios de diferentes niveles. Estos nos permitirán conocer los errores que hemos cometido en cada uno y repetir su realización, mejorando como conductores. Además, están adaptados a la legislación de diferentes países.

Incorpora todo tipo de situaciones y contextos realistas que permiten a los usuarios mejorar como si estuvieran conduciendo un vehículo real, de forma controlada y corrigiendo los errores para buscar la mejora y el aprendizaje. Estas situaciones son variadas como puede ser la conducción con climatología adversa, diferentes comportamientos de otros conductores o la adaptación a diferentes tipos de pavimento, entre otras [9].

Algunas de las características más importantes son:

- Simulación tanto de tráfico real como de peatones.
- Diferentes configuraciones tanto de climatología como de zona horaria.
- Cuatro niveles diferentes como formación inicial, formación vial básica, formación vial avanzada y conducción eficiente.
- Diferentes vehículos con diferentes características.
- Disponible en diferentes idiomas como inglés, español, francés, chino, árabe y otros.
- Control de alumnos con impresión de informes de simulación por cada ejercicio y de forma comparativa.

Una de las grandes ventajas que se pueden observar en este simulador es su gran capacidad para personalizar los escenarios. Pudiendo modificar, de forma combinada, todas las diferentes características. Esto permite recrear infinidad de diferentes escenarios y situaciones para educar al conductor y mejorar sus habilidades.

El clima es configurable y admite situaciones de sol, lluvia, niebla y nieve. También permite elegir entre conducción diurna y nocturna.

Las vías pueden ser todo tipo, como urbanas, autovías, autopistas, carreteras nacionales, carreteras de montaña o caminos. En la Figura 3.3 se muestra una imagen el

simulador en un entorno de autovía. De forma complementaria, permite elegir el tipo de pavimento de las mismas en algunos casos como asfalto, tierra, adoquinado, gravilla y otros [9].

Figura 3.3: Simulador DriveSim en un entorno de autovía.

Los vehículos varían con diferentes características y modelos. Inicialmente, se puede elegir si el coche es diésel o gasolina. Como añadido, se puede seleccionar el tipo de tracción, eligiendo entre tracción delantera, tracción trasera y tracción 4x4. Además, se puede elegir la potencia variando entre 100, 120 y 150 caballos de vapor. Por último, cabe destacar que dispone de un modelo con cambio de marchas automático.

El comportamiento del resto de elementos de inteligencia artificiales creados por la simulación está diseñado de forma aleatoria, presentando diferentes comportamientos reales. Esto añade realismo y permite a los usuarios enfrentarse a situaciones de riesgo similares a las reales. Algunos de estos comportamientos pueden ser peatones que cruzan repentinamente la calle, corredores que cruzan de forma insegura, conductores con conductas agresivas, ejecución de maniobras poco seguras o erróneas por parte de otros vehículos y otras, que añaden realismo y permiten al conductor prepararse ante situaciones de riesgo que se pueden dar de forma real.

Otros parámetros configurables sobre el vehículo permiten el entrenamiento ante fallos o situaciones en las que diferentes sistemas de ayuda en la conducción estén desactivados, como puede ser conducción con o sin sistema ABS, asistencia frente a derrapes, dirección asistida, etc.

La dificultad se puede adaptar en función del usuario y los conocimientos que se quieran mejorar. Además, la modificación de todos los parámetros citados anteriormente, nos permite cambiar el estado dentro del mismo nivel, dando lugar a muchas variaciones que se adaptan a las necesidades del aprendizaje. Los niveles base que nos ofrece el simulador son formación inicial, conducción básica, conducción avanzada y conducción eficiente [9].

En resumen, el simulador DriveSim está enfocado a la educación vial y la enseñanza, desde conductores noveles a conductores experimentados, en ámbitos educativos y laborales. Las principales ventajas que nos permite frente a otros

competidores son la gran personalización de las condiciones y los escenarios, de cara a entrenar a los conductores ante todo tipo de situaciones, y la generación de informes de simulación y detección de fallos en la conducción, que permiten tanto al educador como al alumno conocer los puntos de mejora. Gracias a ello, es una opción recomendable para cursos en autoescuelas, prevención de riesgos laborales, cursos de amaxofobia y otro tipo de eventos de seguridad vial.

3.1.4. Simescar

Simescar es un simulador de conducción de automóvil, que funciona en conjunto con otra herramienta de software llamada Sócrates, y que está enfocado tanto a formación de nuevos conductores como a concienciación de conductores experimentados [10].

Este simulador trabaja sobre un hardware fijo, que recrea de forma fiel el interior del vehículo, para incorporar una recreación más realista de la realidad, mostrado en Figura 3.4. Sobre el mismo trabaja el simulador, que se gestiona mediante la herramienta antes mencionada llamada Sócrates, que además se encarga de gestionar los diferentes usuarios y mantener un historial sobre todos los ejercicios realizados [10].

Figura 3.4: Hardware de simulación para Simescar

El hardware dispone de características realistas que simulan fielmente el funcionamiento de los automóviles. Algunas de las mismas son [10]:

- Columna de dirección con *force feedback* (retroalimentación de fuerza) y volante de turismo real.
- Pedales de turismo real con motores de vibración integrados para recrear el efecto del ABS y el comportamiento del embrague.

- Panel de mandos realista con palancas de intermitente, bocina, luces y limpiaparabrisas.
- Cinturón de seguridad con sistema de bloqueo.
- Palanca de cambios manual de cinco velocidades.
- Tres monitores de 32'' HD *matrix display* para lograr una visión de 135º. Además, un pequeño monitor de 10'' que hace las funciones de panel de instrumentación.
- Posibilidad de integrar una plataforma de movimiento 2DOF, que recrea las sensaciones de fuerzas de aceleración, frenado y diferentes vibraciones sobre el conductor.
- Sonido envolvente 5.1 con la posibilidad del uso de auriculares.

El software está pensado para adaptarse a todos los niveles, enfocado, principalmente, a la educación vial y la enseñanza de conductores noveles y poco experimentados. Sin embargo, también dispone de diversos escenarios enfocados a la concienciación de usuarios experimentados, mostrando las consecuencias de sobrepasar los límites de velocidad o la conducción bajo los efectos de sustancias estupefacientes. Como añadido, posee herramientas que permiten al instructor interaccionar en tiempo real con la simulación mediante el puesto de instructor, y además almacenar los datos de las simulaciones y gestionar los diferentes escenarios [10].

La estructura de las simulaciones está enfocada de forma didáctica, siguiendo un plan de formación virtual orientado al aprendizaje. Los pasos que sigue este plan se pueden resumir de la siguiente forma [10]:

1. Evaluación inicial. Se realiza un cuestionario de aptitudes básicas y una breve prueba de circulación bajo situaciones imprevistas, que sirven para comprobar el nivel inicial del usuario.
2. Conceptos básicos del vehículo. Se estudia cómo realizar una comprobación básica de los reglajes del vehículo y permite al conductor familiarizarse con los controles y su puesta en marcha.
3. Maniobras básicas. Se trabajan maniobras de control del vehículo como manejo del volante o arranque en pendiente.
4. Maniobras de circulación. Se trabajan mecánicas previas a poder iniciar la circulación, como cambio de marchas, desplazamientos laterales o aparcamientos.
5. Aprendiendo a conducir. Se practica cómo maniobrar en rotondas, cruces y cómo se deben realizar correctamente los adelantamientos.
6. Aprendiendo a circular. Se realizan diferentes pruebas completas de conducción por ciudad, autopista y carreteras secundarias.

7. Situaciones de riesgo. Se llevan a cabo una serie de ejercicios controlados para aprender a reaccionar ante frenadas de emergencia, *aquaplanning*, o situaciones evasivas en conducción por ciudad o carretera.
8. Concienciación. Se realizan ejercicios de circulación nocturna simulando diferentes tasas de alcohol para concienciar de los efectos del mismo.
9. Examen. Para finalizar, se realizan unas pruebas de evaluación que permiten evaluar el aprendizaje y poner en práctica todo lo que se ha trabajado previamente.

Además, el simulador nos permite controlar las condiciones bajo las cuales se van a llevar a cabo los diferentes ejercicios para añadir complejidad en función de los conocimientos que se busque trabajar. Como se observa en la Figura 3.5, se podrán variar las condiciones climáticas, las condiciones de luz en función de la hora del día y la agresividad del resto de conductores y peatones. Esto permitirá enfocar el aprendizaje de múltiples maneras mejorando la experiencia transmitida a los usuarios [10].

Figura 3.5: Simulación en condiciones adversas con Simescar.

Como conclusión, Simescar es un simulador enfocado completamente a la educación y concienciación de nuevos conductores. Este entorno busca la experiencia más realista posible, con la intención de que los usuarios se adapten a las situaciones reales de la forma más fácil posible. Además, gracias a las herramientas de valor añadido que ofrece, permite a los instructores participar activamente en todo el proceso de aprendizaje, mejorando la experiencia de aprendizaje sin la exposición a riesgos reales. Gracias a esto, es una opción muy útil para centros de formación y cursos de educación vial.

3.2. Contribución de los simuladores al aprendizaje y seguridad vial

La educación vial es fundamental en nuestra vida diaria. Desde que somos niños pequeños, ya participamos activamente de entornos de circulación como peatones o como pasajeros. Sin embargo, esto conlleva unos riesgos elevados y el aprendizaje supone exponerse de forma necesaria a los mismos. De la misma forma, cuando comenzamos la vida como conductores, todo el proceso de aprendizaje desde el manejo del vehículo hasta la respuesta ante situaciones de riesgo, pasando por la práctica de una conducción eficiente y segura, requiere práctica. Esta práctica supone un riesgo elevado debido a que cualquier error que se cometa puede derivar en un accidente, y dado que la experiencia del conductor es mínima, la probabilidad de error es elevada. Como añadido, existen

personas que, por miedo a tener accidentes y por falta de confianza, deciden no atreverse a intentar aprender debido al riesgo que supone para ellas.

Frente a este problema surgió el uso de simuladores. Estos permiten no solo la práctica de conducción sin asumir ningún tipo de riesgo, sino también un mayor control sobre las condiciones y sobre los comportamientos del usuario. Gracias a esto, se puede educar y corregir mucho más exhaustivamente al alumno, además de permitirle poner en práctica ciertas situaciones que de otra forma no sería posible. Como añadido, los simuladores ayudan también de forma notable a esas personas que tienen miedo a conducir debido a los riesgos que entraña, ya que les permite aprender en un entorno completamente seguro.

Otra gran ventaja que nos permiten dichos simuladores es el estudio de los conductores. Gracias a disponer de un entorno controlado, se puede exponer a los conductores ante una multitud de diversas situaciones y estudiar sus formas de reaccionar ante ellas. Esto permite conocer detalladamente el modo de actuar y cómo el cuerpo se comporta durante las mismas, qué procesos se llevan a cabo en el cerebro, u otros datos de gran interés. Toda esta información es de muchísima utilidad en diversos campos, como en estudios para conocer cómo las personas se comportan ante situaciones de peligro, para el desarrollo de elementos de seguridad activa y pasiva que hagan más seguros los vehículos en el futuro, u otros.

En otros ámbitos, los simuladores de conducción nos permiten conocer mejor qué factores de riesgo afectan a los conductores. Gracias a que en un entorno simulado no existen riesgos reales, se pueden analizar con ellos comportamientos de usuarios sobre los que se aplique ciertos condicionantes como fatiga, alteración debido a diferentes sustancias como el alcohol, enfermedades que afecten a la visión, etc. La recopilación de datos nos permite conocer cómo estas personas se han comportado durante la misma, y analizar qué nuevos factores de riesgo aparecen y a qué afectan, o qué medidas se deben tomar en ciertos casos para aumentar la seguridad.

Este método también se ha comenzado a emplear en numerosas empresas para reducir los riesgos laborales, con resultados notablemente satisfactorios. Gracias al hecho de poder enfrentarse a situaciones de riesgo, prepararse ante ellas es una gran ventaja, además de que supone una formación transversal aplicable al resto de ámbitos de la vida diaria [7].

Como añadido, con los simuladores se puede concienciar a la gente, de forma mucho más directa, de los peligros de una conducción no segura y de ciertas acciones de riesgo. Debido a que el entorno libre de riesgos nos permite evaluar situaciones que en la realidad tendrían consecuencias desastrosas, se puede exponer a los conductores a escenarios en los que se realizan ciertas imprudencias a acciones inadecuadas como puede ser el exceso de velocidad o la falta de distancia de seguridad. Enfrentar al conductor ante estas situaciones y mostrar las consecuencias de forma directa, es un método útil para concienciar al conductor de los peligros y riesgos que se asumen, numerosas veces de forma inconsciente [7].

Por último, es importante destacar que el empleo de entornos controlados permite a los conductores mejorar su estilo de conducción para que sea más eficiente. Gracias a que el entorno simulado evalúa en tiempo real las acciones del usuario, puede ofrecer consejos para mejorar y adoptar un estilo de conducción más eficiente que reduzca el consumo y la contaminación. Esto es mucho más fácil de practicar con un simulador, ya que el sistema conoce de forma mucho más precisa todos los datos del vehículo y el entorno, además de que la inexistencia de riesgos reales permite al conductor centrarse en esto, lo que en un entorno real podría suponer una importante distracción.

Como conclusión, es destacable el gran número de ventajas que nos permite el uso de simuladores. La capacidad de enfrentar a los conductores a un sinfín de situaciones de forma controlada y monitorizada, nos permite no solo corregir y educar, sino también obtener datos, concienciar o establecer condicionantes o factores de riesgo de forma controlada; todo ello sin ningún tipo de riesgo. Gracias a ello, se aumenta de forma notable la seguridad de los conductores, se mejoran los vehículos y se estudian medidas en pos de la seguridad. Como añadido, se mejora el medio ambiente gracias a la práctica y entrenamiento de un estilo de conducción eficiente.

4. Herramientas de desarrollo: Motores gráficos.

En esta sección, se estudiarán las diferentes tecnologías disponibles en la actualidad para crear el simulador de conducción. Se centrará principalmente en el trabajo con motores gráficos, de cara a la creación del entorno y de la parte tanto de interfaz como de simulación de conducción realista. Además, se deben poder recoger datos sobre el vehículo y el sistema de detección de la dirección de la mirada, para que sean procesados y analizados posteriormente. Por último, se realizará una comparativa entre las diferentes herramientas y se decidirá, de forma motivada, la que mejor se adapte al tipo de aplicación que se quiere realizar.

4.1. Concepto de motor gráfico.

En la actualidad, debido a la gran popularidad de los videojuegos y al desarrollo de nuevas formas de interactuar con los mismos, como en la realidad aumentada y la realidad virtual, existen numerosas opciones a tener en cuenta a la hora de elegir un motor gráfico adecuado para desarrollar un simulador de conducción. Para comenzar, se va a analizar concretamente qué es y por qué elegir una opción no debe tomarse a la ligera.

Un motor gráfico es un componente software que permite al programador las funcionalidades básicas para la creación de entornos virtuales interactivos. Además, debe componerse de las siguientes partes [11]:

- Motor de *renderizado*: Este se encargará de manejar todas las figuras tanto en dos dimensiones como en 3, generando gráficamente todos los objetos visibles para el usuario.
- Motor de físicas: Este llevará a cabo todos los cálculos en cuanto al movimiento de todos los objetos, y su interacción con el conjunto de elementos del escenario como el suelo o las luces, además de todo el resto de objetos existentes sobre el mismo.
- Motor de red: Aunque no es estrictamente necesario, los diferentes motores gráficos habitúan a tener bibliotecas o interfaces que facilitan programación de escenarios que se puedan comunicar mediante la red, para dar servicios de comunicación entre usuarios en diferentes máquinas y juego en línea.

Además, en función del tipo de experiencia que se quiera ofrecer al usuario, podemos dividir el conjunto de videojuegos en dos grandes grupos [11]:

- Juegos en primera persona: Comprenden el conjunto de los mismos centrados en dar una experiencia hiperrealista, donde el usuario es el centro de la acción. En estos es importante lograr que el usuario se sienta dentro del mismo, de tal forma que sienta que está viviendo de forma real lo que sucede en pantalla.
- Juegos en tercera persona: Buscan una experiencia externa, similar a una película en la cual se observa a diferentes personajes vivir una historia, con

la que el usuario interactúa de diferentes formas. Mucho más centrado en la narrativa que en ofrecer una experiencia realista.

Para el caso del desarrollo de un simulador de conducción, el interés se centra en un desarrollo en primera persona, de la forma más realista posible, en la que cobran vital importancia el motor gráfico, pero principalmente el motor de físicas. Esto se debe al hecho de que se busca una experiencia hiperrealista, en la que el vehículo se comporte lo más parecido a la realidad, de tal forma que las acciones del conductor tengan los mismos efectos que si se realizaran sobre un vehículo real.

4.2. Principales motores gráficos.

Se pueden distinguir numerosos motores gráficos en la actualidad, algunos de los cuales están notablemente extendidos y se emplean en numerosas aplicaciones. Algunos de ellos son CryEngine, Unity, Unreal Engine, Sony PhyreEngine, Source o Frosbite. Con ellos se han hecho juegos muy exitosos como Far Cry, Ori and the Blind Forest, Batman: Arkham City, Dark Souls, Counter Strike: Global Offensive y Battlefield 3, respectivamente.

4.2.1. Cry Engine 3.

Este motor gráfico, perteneciente a la empresa *Crytek*, es la tercera versión del mismo, desarrollado inicialmente para el juego Far Cry. Este tiene el objetivo, con su tercera versión, de convertirse en el motor gráfico más potente de la industria, permitiendo a los creadores las herramientas para crear experiencias de clase mundial [12].

Comenzando por el apartado visual, soporta la tecnología DirectX12, con lo que busca permitir a los desarrolladores un mayor rendimiento. Con *renderizado* basado en físicas hiperrealistas, trata de permitir a los desarrolladores facilidad a la hora de crear experiencias inmersivas. Además, emplea diferentes tecnologías de teselado, para permitir optimizar el juego de forma gráfica en función de los requerimientos del desarrollador. Por último, cabe destacar que incorpora humo volumétrico, que permite simular entornos con partículas en suspensión como polvo o niebla, que interactúan con la luz ambiental, como se puede ver en la Figura 4.1.

Figura 4.1: Humo volumétrico en CryEngine 3.

A continuación, se estudia la interfaz de programación. Esta dispone de un editor de materiales, además de emplear el formato FBX para integrar el contenido digital en 3D. Como añadido, tiene un editor de figuras 3D, modeladas a partir de “cajas blancas” y un editor gráfico de flujo de juego, *FlowGraph*, que permite crear niveles complejos a partir de control de lógicas y eventos de forma estructurada y cómoda, como se observa en la Figura 4.2. Por último, incorpora una herramienta para la creación de cinemáticas con control temporal, lo que permite mayor facilidad a la hora de sincronizar tanto los eventos, movimientos y sonidos.

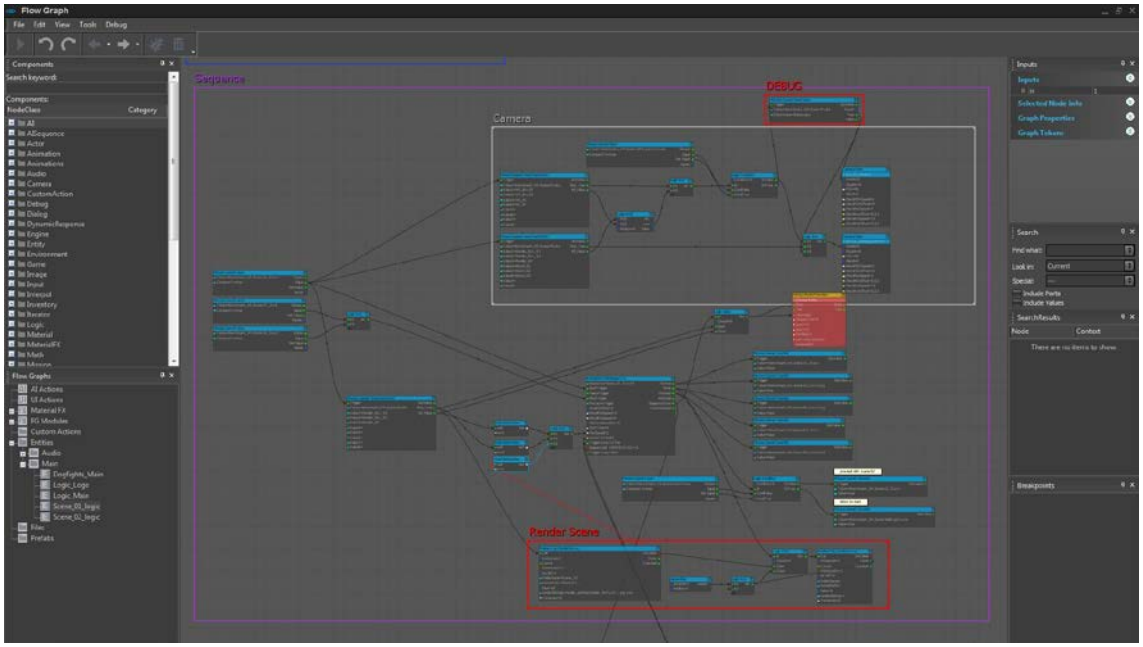

Figura 4.2: Interfaz de desarrollo de flujo en CryEngine 3.

Por último, es importante destacar que las plataformas sobre las que trabaja este motor son PC, Linux, Xbox ONE y Playstation 4. Además, los requisitos mínimos del sistema para programar sobre el mismo son:

- SO: Windows 7 o Windows 8.1
- Procesador: Intel Dual-Core 2GHz o AMD Dual-Core 2GHz.
- RAM: 4 GB.
- Procesador Gráfico: NVIDIA GeForce 400 Series.
- DirectX: Versión 11.
- Disco Duro: 8 GB.
- Tarjeta de Sonido: Compatible con DirectX.

4.2.2. Unreal Engine 4

La primera versión de este motor gráfico nació en 1998, cuando la empresa que lo desarrolló, *Epic Games*, lo empleó para desarrollar el videojuego *Unreal*. Desde entonces, ha evolucionado estando actualmente en su cuarta versión, y se ha empleado

principalmente en el desarrollo de videojuegos en primera persona. Esta última versión proporciona múltiples mejoras en cuanto a manejo de sombras y un mejor entorno gráfico de desarrollo.

Su arquitectura está abierta, y permite a los creadores adaptar el motor en función de las necesidades específicas y de rendimiento que requieran las distintas plataformas, optimizando su funcionamiento. Esto permite que pueda ser adaptado a todas las plataformas existentes en el mercado, aunque ya existen versiones optimizadas para la mayoría de ellas. Sin embargo, se trata de una plataforma de pago, aunque gran parte de su documentación es gratuita, y las cuotas de acceso al código no son excesivamente elevadas, pudiendo ser accesibles también a pequeños desarrolladores [11].

En cuanto a las características visuales de este motor, cuenta con *renderizado* fotorrealista en tiempo real, lo que permite una mayor eficiencia en cuanto al manejo de sombras y luces. Incluye efectos ambientales con calidad cinematográfica, como oclusión ambiental, profundidad de campo o destello de lente, entre otros. Además, permite la creación de grandes extensiones de terrenos exteriores, generados y mostrados al jugador con carga en memoria eficiente, lo que permite grandes extensiones de terreno de forma optimizada y sin un gran consumo de recursos. Esta última característica se puede observar en videojuegos de mundo abierto como *Batman Arkam City*, cuyo entorno se muestra en la Figura 4.3.

Figura 4.3: Entorno de *Batman Arkam City* creado con Unreal Engine 4.

Analizando el entorno de desarrollo, Unreal Engine 4 ofrece todas las herramientas necesarias para facilitar a los desarrolladores la creación de videojuegos. Para comenzar, una de las ventajas más importantes de este motor es que ofrece todo el código fuente del mismo de forma abierta, en lenguaje C++, lo que permite su adaptación en función de las necesidades de la plataforma y del videojuego que se quiera crear. Posee un editor de materiales notablemente flexible, que permite la personalización de los objetos con un gran nivel de detalle. Como añadido, dispone de un sistema de mercado

de *assets* compartido entre los desarrolladores, accesible desde la propia interfaz de programación, lo que permite compartir contenido que pueda ser de utilidad de forma cómoda y bien organizada. Por último, dispone de un editor visual de flujo de eventos e interacciones, que permite generar código de scripts de forma automatizada y estructurada con la herramienta *Blueprint*, lo que facilita notablemente las tareas complejas a los programadores. Su interfaz se puede ver en la Figura 4.4.

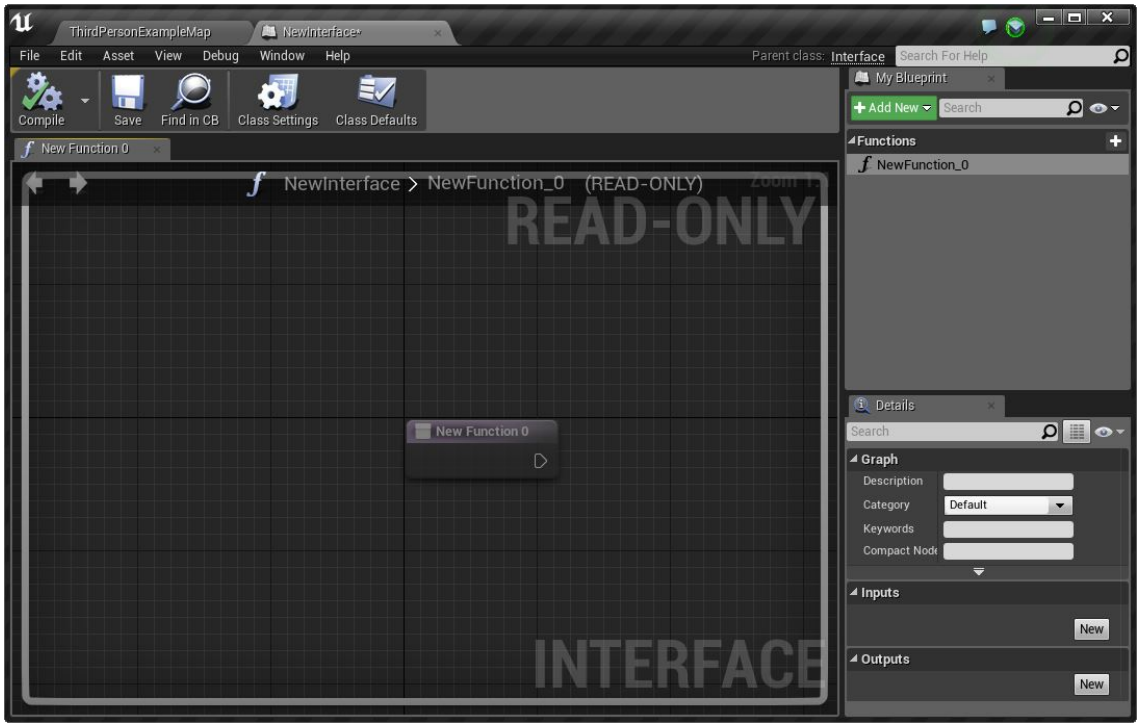

Figura 4.4: Interfaz *Blueprint* para el diseño de scripts en Unreal Engine 4.

Es importante destacar que Unreal Engine 4 es uno de los motores gráficos que más se han orientado a la programación de experiencias tanto de realidad virtual como de realidad aumentada. Para ello, posee un editor integrado dedicado exclusivamente a esta función, que permite trabajar o simular el funcionamiento del producto en tiempo real tal y como sería el producto acabado, lo que permite gran facilidad en la creación de experiencias. Como añadido, permite el desarrollo nativo para la mayoría de plataformas populares en el mercado, como *Oculus rift*, con alta calidad gráfica y un consumo de recursos eficiente.

Como características extra, este motor gráfico cuenta con un marco de trabajo para multijugador desarrollado y probado sobre numerosas plataformas y videojuegos diferentes, lo que permite una gran facilidad de desarrollo de arquitecturas cliente/servidor de forma escalable. Además, cuenta con una extensa documentación y tutoriales, que facilitan la iniciación y el uso del mismo, desde el nivel básico hasta el nivel avanzado [13].

Por último, es importante mencionar los requisitos mínimos requeridos para la realización de desarrollos con este motor, que son:

- SO: Windows 7/8 de 64 bits.

- Procesador: Quad-Core Intel o AMD, de 2,5 GHz o mejor.
- Memoria RAM: 8 GB.
- Tarjeta gráfica: Compatible con DirectX 11 o superior.

4.2.3. Unity

Es un motor gráfico multiplataforma, creado por *Unity Technologies*, que permite desarrollar de forma fácil y exportar tanto a sistemas móviles, consolas, entornos de escritorio e incluso Webplayers para los principales navegadores. El entorno unificado permite tanto el desarrollo y simulación en tiempo real en el ordenador, como un modo depuración sobre el hardware de destino, como por ejemplo un móvil. Además, ofrece herramientas para analizar el funcionamiento en las diferentes plataformas, además de permitir código en lenguaje JavaScript, C# o Boo [11].

Primero, vamos a comenzar estudiando las características visuales de este motor gráfico. Permite trabajar con iluminación global en tiempo real, con generación de sombras basadas en físicas y con sistemas de partículas controladas por curvas y gradientes, como pueden ser el humo o diferentes fluidos. Además permite trabajar con efectos como la profundidad de campo o el desenfoque de lente, aplicando un gran realismo a cómo la luz afecta a diferentes superficies.

Dispone de un sistema de físicas basado en la interacción de objetos con el escenario y con el resto de objetos, en función de la detección de colisiones dependiendo del movimiento. Además, incluye NVIDIA PhysX 3.3 para aportar mayor realismo a dichas interacciones. Por último, permite relacionar físicamente los objetos de forma gráfica gracias al uso de las regiones de colisión, que determinan cómo será ese objeto para el propio motor de físicas y cómo se relacionará con el resto de objetos en pantalla. Esto facilita notablemente al desarrollador la comprensión y creación de nuevos objetos que interactúan de forma controlada. Un ejemplo se puede ver en la Figura 4.5.

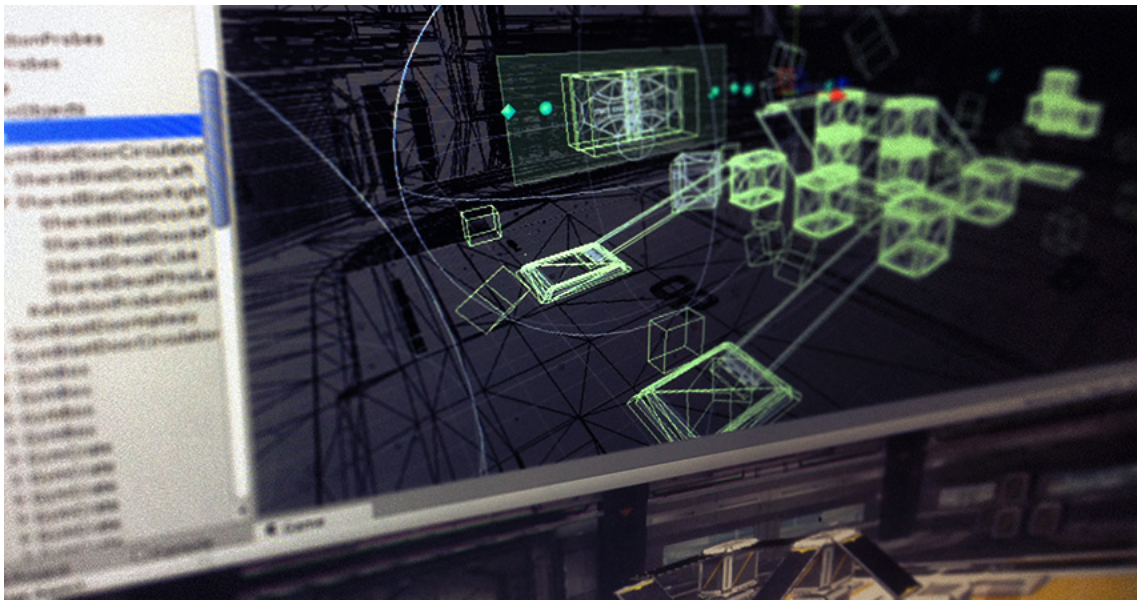

Figura 4.5: Sistema de regiones de colisión de un entorno en Unity.

El editor de entorno de Unity es sencillo e intuitivo, siendo fácil de manejar en cuanto a la manipulación de escenarios y estando integrado de forma nativa con Visual Studio para la creación y manejo de scripts. Permite un gran control del rendimiento con las herramientas de análisis, lo que permite un rendimiento fiable con una tasa de imágenes procesadas por segundo fluida, evitando cuellos de botella gráficos y controlando de forma fácil la carga de activos. Incorpora un mercado interactivo, dentro del propio editor, donde obtener *assets* tanto gratuitos como de pago realizados por otros equipos. Esto facilita la interacción entre los desarrollos, empleando recursos compartidos que facilitan la creación de videojuegos. Por último, cabe destacar la herramienta de la máquina virtual de simulación. Esta nos permite construir nuestro juego de forma virtual y simularlo de la misma forma que si estuviera construido para su versión final, lo que nos permite realizar pruebas de forma rápida y controlada con la misma visión que tendría el jugador si la aplicación estuviera concluida. La interfaz de desarrollo de Unity se puede ver en la Figura 4.6.

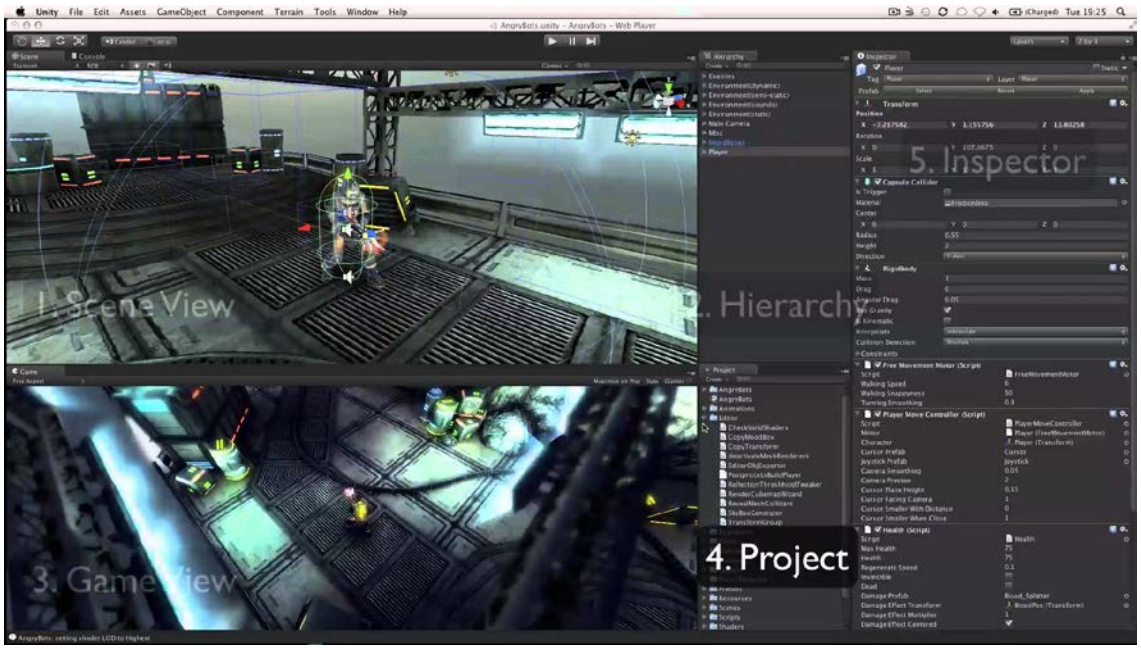

Figura 4.6: Interfaz de desarrollo de Unity.

Una de las mayores fortalezas de Unity es su gran versatilidad, y la facilidad de incorporar recursos a los proyectos en múltiples formatos. Este motor gráfico maneja tres lenguajes de programación: C#, JavaScript y Boo, lo que permite emplear cualquiera de ellos según las capacidades del desarrollador o las necesidades del producto que se esté llevando a cabo. Además, Unity soporta los siguientes formatos:

- Imagen: psd, jpg, png, gif, bmp, tga, tiff, iff, pict, y dds.
- Audio: mp3, ogg, aiff, wav, mod, it, y sm3.
- Video: mov, avi, asf, mpg, mpeg y mp4.
- Texto: txt, htm, html, xml y bytes.

Como añadido, se ofrecen una serie de servicios externos que facilitan a los desarrolladores el control sobre la monetización, control de estadísticas, colaboración entre equipos, u otras. Estos servicios externos son:

- Unity Ads: Permite generar ingresos, monetizando los videojuegos de forma automatizada gracias a la publicidad, de forma fácil con la ayuda de Unity.
- Unity Analitics: Permite conocer todas las estadísticas sobre los jugadores y el uso del producto, tiempo de juego, avance por el mismo y otros, que da información sobre la aceptación del producto.
- Unity Certification: Permite reconocer las capacidades del desarrollador de forma oficial como desarrollador certificado de Unity. Esto facilita el reconocimiento de cara a ser contratado en equipos de desarrollo mayores.
- Unity Cloud Build: Permite construir el videojuego en la nube, lo que es mucho más rápido y facilita las iteraciones entre los equipos.
- Unity Collaborate: Permite mantener un repositorio en la nube, para que todo el equipo de desarrollo pueda estar sincronizado de manera sencilla.
- Unity Everyplay: Permite la creación de una comunidad alrededor del videojuego, en la que los usuarios pueden compartir contenido y relacionarse, aumentado la vida y expansión del mismo.
- Unity IAP: Facilita la gestión de una tienda interactiva dentro del videojuego, facilitando las transacciones en tiempo real y portear las compras dentro de la aplicación.
- Unity Multiplayer: Con las interfaces de bajo nivel permiten optimizar el juego al entorno de red que se desee. Para alto nivel emplea un flujo de componentes similar al de la creación de escenarios, lo que facilita la programación.
- Unity performace reporting: Permite recopilar fallos en aplicaciones para diferentes dispositivos y plataformas en tiempo real, lo que permite aplicar soluciones rápidas minimizando el impacto en la experiencia de usuario.

Otra característica muy importante es que es un motor gráfico que cuenta con una extensa documentación en línea y numerosos tutoriales creados por la comunidad, lo que facilita a los nuevos programadores la iniciación en el desarrollo de videojuegos. Además, tiene una versión completamente gratuita con todas las prestaciones necesarias para la creación de videojuegos, como el entorno de simulación en tiempo real o el soporte multiplataforma. Sin embargo, esta versión gratuita es plenamente funcional en caso de que no se pretenda generar ingresos con el contenido creado, o siempre que ese ingresos sean menores a 100.000 dólares anuales.

En cuanto a las plataformas soportadas, como ya se expuso en la introducción, Unity permite, gracias a sus interfaces en alto nivel, adaptar el desarrollo sin modificar el

producto y compilarlo para una gran variedad de plataformas, lo cual es una de sus grandes virtudes y por lo que es tan reconocido. Debido al gran número de ellas, resulta adecuado mostrarlas en la Figura 4.7 [14].

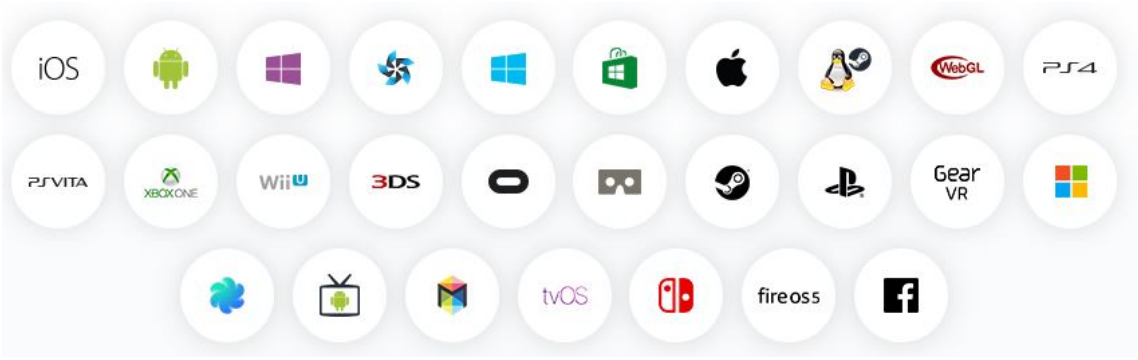

Figura 4.7: Conjunto de plataformas soportadas por Unity.

Por último, los requisitos del sistema para desarrollo con la última versión de Unity son:

- SO: Windows 7/8/10 o MAC OS X 10.8+.
- Procesador: Intel Dual-Core o AMD, de 2GHz.
- Memoria RAM: 4 GB.
- Procesador gráfico: Compatible con DirectX9 (Shader 3.0) o DirectX 11.

4.3. Comparativa de motores gráficos

Para realizar una comparativa que motive el uso del motor gráfico adecuado para muestro simulador de conducción, se van a mostrar las principales características en la Tabla 2.1.

	CryEngine 3	**Unreal Engine 4**	**Unity 3D**
Lenguaje de programación empleado	LUA	C++	C# JavaScript Boo
Herramientas de Scripting Visual	SÍ, FlowGraph	SÍ, Blueprint	NO
Código fuente disponible para modificaciones	NO	SÍ	NO
Documentación	SÍ	SÍ	SÍ
Tutoriales	Pocos	SÍ	SÍ
Comunidad activa	SÍ	SÍ	SÍ

Mercado de assets de la comunidad	NO	SÍ	SÍ
Adaptación a Realidad Virtual y Realidad Aumentada	NO	ALTA	MEDIA
Mantenimiento periódico	SÍ	SÍ	SÍ
Editor de materiales	SÍ	SÍ	SÍ
Plataformas de desarrollo	Windows Mac OS Linux	Windows Mac OS	Windows Mac OS
Plataformas para las que puede desarrollar	PC Linux Xbox ONE Playstation 4	Android IOS Google VR Oculus Rift Samsung Gear VR PC Mac OS Linux HTML5 Nintendo Switch PS4 Xbox ONE	Android IOS Google VR Oculus Rift Samsung Gear VR PC Mac OS Linux HTML5 Nintendo Switch Nintendo 3DS PS4 Xbox ONE
Coste de Uso	50€ al mes	Se debe aportar un porcentaje de las ganancias a partir de 3000€	Gratuito en la versión Personal
Facilidad de Introducción al mismo	Baja	Media	Alta

Tabla 4.1: Comparativa entre los diferentes motores gráficos.

Una vez tratados tanto los diferentes motores gráficos como la tabla resumen de sus principales características, y dada la información recogida, se decide descartar el motor gráfico CryEngine 3, debido a que no tiene licencia gratuita, es el más complejo y además no tiene muchos tutoriales.

Comparando Unity y Unreal Engine 4, son bastante similares. Sin embargo, algunas diferencias resultan importantes de cara a tomar la decisión de qué plataforma se adapta mejor a las necesidades y al tipo de desarrollo:

- Unity tiene una mayor comunidad de desarrolladores y de tutoriales, ya que está pensado para gente sin experiencia que quiera introducirse en el desarrollo de videojuegos. En comparación, Unreal Engine también posee extensa documentación y una comunidad sólida, pero su mayor complejidad provoca que sea más difícil de comenzar el desarrollo y se requieren ciertos conocimientos mínimos para comprender dichos tutoriales.
- Otra ventaja de Unity frente a Unreal Engine es que trabaja con una mayor variedad de formatos, pudiendo añadir multitud de recursos al proyecto de forma rápida y cómoda. Sin embargo, Unreal Engine es más limitado en ese sentido, y aunque posee un editor de materiales mucho más potente, también es notablemente más complejo para el uso que requiere nuestro simulador.
- En cuanto a las herramientas, Unity es más limitado ya que las que tiene son notablemente simples y es un motor gráfico muy básico. Esto provoca que para crear escenarios con un alto nivel de detalle y efectos complejos requiera mucho tiempo. Frente a esto, Unreal Engine tiene un gran número de herramientas que se actualizan periódicamente, que facilitan la tarea de crear entornos gráficos con alto nivel de detalle. Esto provoca que la creación de los mismos sea más rápida, aunque el tiempo que puede llevar dominar dichas herramientas es notablemente mayor.
- Unity no tiene características de modelado en tiempo real, lo que hace más complejo el desarrollo de escenarios complejos. En contraposición, Unreal Engine sí que las tiene, lo que facilita los desarrollos.
- Unity es mucho más simple gráficamente, y posee pocos efectos gráficos que permitan la creación de escenarios hiperrealistas con alto nivel de detalle. Por otro lado, Unreal Engine es un motor gráfico centrado en el realismo de los escenarios y en la creación de videojuegos donde la ambientación y la inmersión sean muy altas, donde el usuario pueda vivir una experiencia mucho más inmersiva.

Debido al entorno que queremos desarrollar, dado que es un simulador de conducción, este se centra principalmente en el realismo del entorno cercano, como puede ser la carretera y las señales. Además, la inteligencia artificial es mucho más importante que un extenso mapeado exterior hiperrealista, ya que el comportamiento del resto de vehículos debe adaptarse fielmente a la realidad. El comportamiento de las físicas de la

conducción, es otro factor importante de cara a dar una experiencia válida para la educación vial de una conducción segura y eficiente.

Por último, debido a que el desarrollo del simulador está enfocado a un entorno educacional, y que el conjunto de personas que colaboran en el desarrollo son estudiantes sin experiencia en el desarrollo de videojuegos, se decide emplear el motor gráfico Unity 3D. Esto se debe a su facilidad y accesibilidad para personas con poca experiencia, y de que las herramientas son más simples y fáciles de utilizar, aunque suficientemente potentes como para poder desarrollar de forma adecuada el simulador de conducción.

5. Detección de movimiento

En esta sección, se estudiarán algunos sistemas de detección del movimiento del usuario, integrables sobre nuestro motor gráfico Unity. A continuación, se realizará una comparativa de los mismos y se decidirá el uso de uno de ellos. Por último, se realizará una exposición de cómo se lleva a cabo la integración del método elegido, y de qué información se obtiene del mismo.

5.1. Sistemas de detección para videojuegos

Los sistemas de detección de movimiento, son elementos hardware que emplean una serie de sensores, como cámaras RGB, sensores de profundidad, sensores de infrarrojos, etc., para capturar el esqueleto humano y posicionarlo en un plano para extraer datos de su movimiento.

En la industria de los videojuegos, estos sistemas surgieron a principios del siglo XXI, con la aparición del dispositivo *EyeToy* para la plataforma *PlayStation 2*. Este consistía en una pequeña cámara que permitía capturar el movimiento del usuario, y convertirlo en una herramienta de control. Este sistema se hizo notablemente popular, lo que provocó que más adelante se desarrollaran algunos otros, tanto por la misma compañía, Sony Entertaiment, como sus competidoras Microsoft o Nintendo [15].

En nuestra aplicación, se busca el empleo de estas tecnologías de forma integrada en nuestro simulador de conducción, para monitorizar los movimientos del conductor. El objetivo será, a partir de la información de este tipo de dispositivos, estimar la posición de la mirada del conductor a partir de los datos del seguimiento de su cara. Esto nos permitirá obtener información sobre el modo de conducción, y si esta es preventiva, sin distracciones, si se mira de forma adecuada a los espejos, u otros datos de interés.

5.1.1. Kinect

Kinect es un dispositivo desarrollado por la empresa Microsoft, que lanzó al mercado su versión 2.0 en Octubre de 2014. Este fue pensado como un simple controlador de juego, pero debido a su estructura hardware y al conjunto de componente software desarrollado se puede emplear para detección de movimiento, aplicado a numerosos campos como estudio gestual, reconocimiento de patrones faciales, rehabilitación y educación postural o monitorización para educación mediante simuladores, entre otros. La forma del hardware de Kinect se puede observar en la Figura 5.1.

Figura 5.1: Hardware de Kinect.

Estudiando el hardware integrado, este dispositivo se compone de:

- Sensor de profundidad, que permite la detección de diferentes partes del cuerpo a partir de la variación de distancias del mismo al dispositivo, como si se tratara de un sistema de sonar.
- Cámara RGB, que permite captar las diferencias de color para construir siluetas y formas que distingan al usuario del entorno y capten sus movimientos.
- Micrófono de matriz múltiple, para la detección de sonido estéreo y reconocimiento de la posición del mismo en el espacio.
- Sensor de infrarrojos, que trabaja de forma complementaria al sensor de profundidad, capturando el esqueleto humano y posicionándolo en el plano, para realizar la detección del mismo.

Además, el uso de estos sensores se realiza de forma combinada, permitiendo agregar la información de los mismos obtenida por diferentes métodos, obteniendo numerosos datos útiles, como se puede ver en la Figura 5.2.

Figura 5.2: Detección de una escena con Kinect.

En cuanto a sus características, cabe destacar las siguientes [16]:

- Detección de cuerpo: Permite realizar el seguimiento de hasta 6 personas y 25 articulaciones por persona de forma simultánea, desde el centro de la espalda hasta las puntas de los dedos. Además, a partir de los tejidos conectivos de las articulaciones y la postura corporal, permite la obtención

de posturas anatómicamente correctas para realizar una interacción mucho más fluida.

- Herramienta de desarrollo: Kinect Studio es un software proporcionado por Microsoft, que permite realizar grabaciones y reproducciones de movimientos. Como añadido, posee un generador de gestos virtuales que permite compilar gestos personalizados. Estos podrán ser usados posteriormente por el sistema para la generación del código de forma automatizada, facilitando el desarrollo en aplicaciones.
- Gran resolución facial: el sistema permite el seguimiento y representación de la cara de una persona. Para ello, emplea más de 1000 puntos de referencia que consiguen una representación altamente fiable de la cara y de los gestos, así como información detallada de su movimiento.
- Compatibilidad de multiaplicación simultánea: está diseñado para que sea multiaplicación, es decir, que con un solo sensor puedan acceder varias aplicaciones de forma simultánea.
- Compatibilidad con Unity: Está pensado para integración con Unity, disponiendo de código oficial ya estructurado que se encarga de su inicialización y configuración para facilitar al usuario la obtención de parámetros funcionales útiles para el desarrollo de aplicaciones.
- Documentación oficial: Kinect cuenta con una comunidad activa de desarrolladores y con una documentación oficial notablemente detallada. Además está muy extendida, lo que facilita la búsqueda de información y el desarrollo de aplicaciones.

Por último, cabe destacar las características técnicas de esta tecnología, que son:

- Detección de profundidad: 512 x 424 30 Hz.
- Campo visual: 70 x 60
- Distancia máxima: 4,5 metros.
- Cámara a color: 1080p a 30Hz (15Hz con poca luz).
- Sensor FOV: 70 x 60
- Campo de infrarrojos: 512 x 424 30Hz.
- Dimensiones del sensor completo: 24,9 x 6,6 x 6,7 cm.
- Longitud del cable: 3m.
- Peso aproximado: 1,5 Kg.
- Interfaz: USB 3.0.

En conclusión, Kinect es una tecnología de una calidad media, accesible económicamente gracias a su uso extendido. Cuenta con una documentación adecuada y un soporte continuo. Además, dado que se va a trabajar en el reconocimiento del

movimiento de la cara, Kinect dispone de herramientas de alta precisión para esta tarea. Por último, y lo que es la gran ventaja para el desarrollo de nuestro simulador, es que está pensada para integrarse con Unity y existe código oficial que permite ahorrar numerosas horas de trabajo en el desarrollo desde cero del sistema de detección. Simplemente integrando esas funciones con Unity, se puede trabajar con los resultados de características del movimiento sin tener que diseñar los métodos para obtenerlas a partir de los sensores, los cuales ya está diseñados de forma muy fiable por los propios desarrolladores del dispositivo.

5.1.2. Xtion PRO

Xtion es un dispositivo desarrollado por la empresa Asus, y está diseñado para implementación en videojuegos y aplicaciones interactivas basadas en el movimiento del jugador. Se basa en la tecnología de sensores de profundidad, que se encargan de construir virtualmente la figura del usuario que se encuentra delante, y colocarla en un plano. A partir de esta, se obtienen los datos del movimiento.

Está pensada para multitud de diferentes aplicaciones, incluso enfocadas a la industria. Los diversos campos a los que se puede aplicar la tecnología son turismo, rehabilitación física, educación, conferencias y demostraciones de producto, videojuegos, medicina y muchos otros.

Para detectar la profundidad, el hardware se compone de dos cámaras, como se puede ver en la Figura 5.3 [17]:

- Un sensor de infrarrojos, que detecta la profundidad general en todo el plano.
- Un par de cámaras adaptativas, que permiten captar los movimientos en tiempo real con alto detalle, que generan de forma automatizada la información del movimiento de forma procesada.

Figura 5.3: Hardware Xtion Pro.

El hardware está pensado para dos utilidades de trabajo principales, aunque se puede emplear la información de estas en función de la aplicación:

- Detección de gestos: la solución hardware de Asus detecta el movimiento de las manos en tiempo real, con alta tasa de muestras. Esto está pensado

para poder utilizarlas como controlador, funcionando como interfaz de entrada para cualquier tipo de aplicación.

- Detección del cuerpo entero: esta segunda opción es mucho más genérica, y pretende adaptarse a cualquier tipo de aplicación y escenario. Aporta detallada información de todas las zonas del cuerpo, permitiendo al desarrollador centrarse en la información que sea de interés, empleando toda la información del cuerpo para la realización de las actividades dentro del entorno de programa. Un ejemplo de detección corporal se muestra en la Figura 5.4.
- Combinada: También permite el uso de ambas capacidades de forma combinada. Así se podrá obtener información detallada de los gestos en el movimientos de las manos mientras se analiza la posición del cuerpo, lo que permite realizar aplicaciones que, por ejemplo, requieran hacer movimientos con las manos en ciertas posturas corporales para el entrenamiento físico de realización de actividades como yoga.

Figura 5.4: Detección del cuerpo completo por Xtion.

Otras de las ventajas que ofrece es que es sencillo de programar, ya que es compatible con la solución OPENNI NITE, la cual es un *middleware* pensado para la detección del movimiento. Este es un SDK de código libre empleado en el desarrollo de librerías y aplicaciones para reconocimiento del movimiento en 3D. Dentro del mismo, está dividido en control gestual mediante las manos y control corporal completo. Además, tiene funciones como localización de las manos y seguimiento de gestos, analizador de escena que permite separar y distinguir al usuario del fondo, detección del esqueleto con precisión o reconocimiento de gestos, entre otros [18].

Para finalizar, es importante resaltar las características técnicas de esta tecnología:

- Tipo de sensor: Profundidad.

- Detector de profundidad: 640 x 480 30Hz (VGA) o 320 x 240 60Hz (QVGA).
- Campo de visión: 58 x 45.
- Distancia máxima: 3,5 metros.
- Interfaz: USB 2.0.
- Dimensiones: 18 x 3,5 x 5 cm.
- Lenguajes de programación: C++/C#.

En resumen, este dispositivo ofrece características de detección de movimiento notablemente completas, enfocadas tanto a reconocimiento gestual como a movimientos de todo el cuerpo. Sin embargo, la documentación está bastante obsoleta y además apenas tiene una comunidad de programadores activa. En general, es una tecnología que no se extendió lo esperado y actualmente se emplea en algunos campos muy concretos, y para aplicaciones en entornos cerrados no comerciales. Su facilidad de programación e integración gracias al middleware OPENNI la hace más accesible, pero la falta de información de código útil hace que esta ventaja pase a un segundo plano.

5.2. Comparativa entre los sistemas de detección

Para realizar una comparativa de forma útil, que permita elegir la solución más adecuada teniendo en cuenta el motor gráfico a emplear y las necesidades de nuestro sistema, se va a emplear la Tabla 5.1:

	Kinect 2.0	**Xtion Pro**
Documentación y soporte actualizado	SÍ	NO
Comunidad de desarrolladores activa	SÍ	NO
Lenguajes de programación	C++/C#	C++/C#
Integración con el motor gráfico Unity	SÍ	SÍ
Sensor de profundidad	SÍ	SÍ
Cámara de infrarrojos	SÍ	SÍ
Cámara RGB	SÍ	NO
Detección de cuerpo	SÍ	SÍ
Detección gestual	SÍ	SÍ
Detección facial detallada	SÍ	NO

Middleware de acceso al dispositivo	SÍ	SÍ
Detección de profundidad	512 x 424 30 Hz	640 x 480 30Hz (VGA) 320 x 240 60Hz (QVGA)
Campo de visión	70 x 60	58 x 45
Distancia máxima de detección	4,5 metros	3,5 metros
Cámara a color	NO	1080p a 30Hz (15Hz con poca luz)
Campo de infrarrojos	512 x 424 30 Hz	640 x 480 30Hz (VGA) 320 x 240 60Hz (QVGA)
Dimensiones del sensor	24,9 x 6,6 x 6,7 cm	18 x 3,5 x 5 cm
Interfaz	USB 2.0	USB 3.0

Tabla 5.1: Comparativa entre Kinect y Xtion.

Como se puede observar en la tabla, el dispositivo de Asus, Xtion, se ha quedado notablemente atrás, y pese a ser muy novedoso en su salida, la falta de popularidad ha provocado que en la actualidad no existan prácticamente desarrolladores que lo utilicen para la solución de problemas. Además no posee una comunidad activa, y sus características técnicas están obsoletas en la actualidad. Como añadido, el hecho de que no ofrezca de forma directa un sistema de detección de la cara, complica su adaptación para la resolución del problema planteado, ya que sería más costoso adaptar la detección del cuerpo que realiza esta tecnología para obtener la información útil necesaria.

Por otro lado, Kinect es un dispositivo notablemente extendido, con una gran comunidad de desarrolladores y con una documentación de calidad. Además, las características técnicas son mucho más modernas, sin por ello suponer un coste mucho más elevado debido a su comercialización en masa. Otras de las grandes ventajas de Kinect es que también está pensado para detección facial detallada, de forma que se adapta mucho mejor a nuestro problema. Por último, el hecho de que los propios desarrolladores de Kinect tenga código público disponible para Unity, ya adaptado para que los desarrolladores no tenga que programar a bajo nivel, obteniendo los valores necesarios de las características faciales de forma integrada, provoca que esta solución sea muy fácil de integrar sin coste.

En resumen, Kinect ofrece todo lo que necesita nuestro proyecto: detección facial fácil de integrar y sin tener que preocuparnos de realizar programación a bajo nivel. En contraposición, Xtion es una tecnología obsoleta, que apenas cuenta con documentación ni detección facial de forma específica. Debido a esto se decide escoger la tecnología Kinect para emplearla en el proyecto.

5.3. Integración de Kinect en Unity

En este apartado se va a explicar el conjunto de código ofrecido por los desarrolladores, y como este se incorpora a Kinect para, a partir de la comunicación con los controladores del sensor, calcular y obtener las diferentes características de la cara como son el marco donde está situada o los ángulos de giro de la misma, entre otros.

5.3.1. Método para la obtención de características

Para el análisis de código y la explicación de las diferentes clases, se va a emplear la documentación disponible en [19]. Además, como guía, se va a emplear el diagrama de flujo que se muestra en la Figura 5.5. Este representa todo el proceso desde que se activa e inicializa el controlador de Kinect desde la aplicación en Unity, hasta el punto en el que se devuelven los datos de características preparados para ser utilizados.

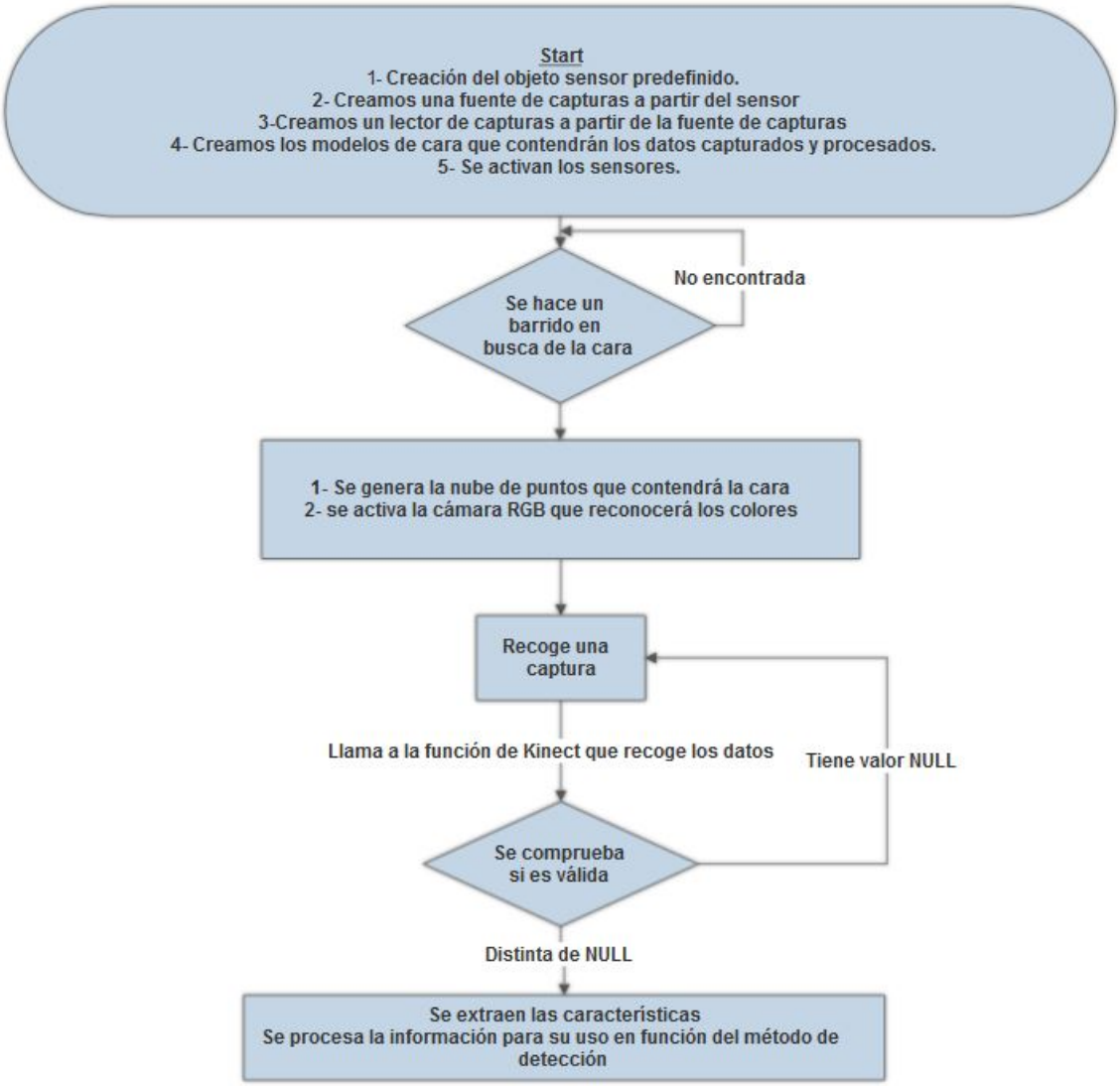

Figura 5.5: Diagrama de secuencia de captura de datos de Kinect.

El diagrama comienza con la llamada Start que se realiza desde el script de Kinect dentro de nuestro simulador de conducción. Esta se produce al arrancar el programa, ya que dentro de Unity durante el arranque se llama a todas las funciones Start de todos los scripts activos. Dentro de esta función:

- Se crea un objeto sensor genérico.
- Se crea una fuente de capturas, a partir del sensor.
- Se crea un lector de capturas, a partir de la fuente de capturas.
- Se crea un objeto que contendrá los datos de la cara relativos a la cámara RGB, obtenidos por Kinect a partir del procesamiento de la captura realizado por el lector de capturas.
- Se crea un objeto que contendrá los datos de Kinect relativos al sensor de profundidad, obtenidos a partir del procesamiento realizado por el lector de capturas.
- Se activa el sensor, con todos los objetos necesarios ya instanciados.

Una vez que se activa el sensor, se realiza un barrido en el que Kinect busca un modelo identificable como una cara, para poder comenzar a capturar. En caso de no encontrarlo, continuará haciendo barridos hasta que encuentre uno. En caso de encontrarlo, continuará con:

- Se generará una nube de puntos, en forma de malla alrededor de la cara, a partir de los datos obtenidos por el sensor de profundidad, como se ve en la Figura 5.6.

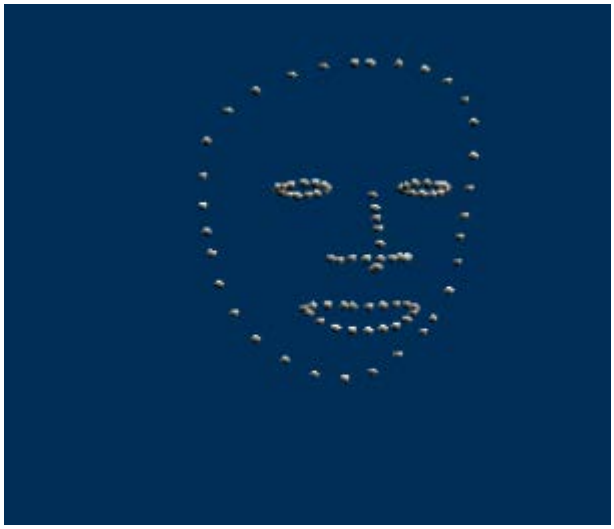

Figura 5.6: Nube de puntos generada por Kinect para la detección de la cara.

- Se activa la cámara RGB, que reconocerá los colores de las zonas donde se ha creado la malla, que coinciden con la cara.

En este punto, se recogerá una captura y se procesará, almacenando en los modelos previamente creados toda la información relativa a ambas cámaras de forma respectiva.

En el siguiente paso, el controlador llamará a la función de Kinect encargada de recoger los datos y ponerlos a disposición del desarrollador del simulador, para que sean empleados en función de sus necesidades. En esta función, el primer paso es comprobar si los valores se han capturado correctamente. En caso negativo, se saldrá del método, dando como resultado una captura fallida. En caso positivo, se extraerán de los modelos las características de interés, y se procesarán en función del método de estimación para obtener como resultado el punto o zona de la pantalla a la que el usuario está mirando.

5.3.2. Características obtenidas desde Kinect.

En cuanto a las distintas características que obtenemos de Kinect, debemos distinguir las que provienen de la cámara RGB que detecta colores, y los detectores de profundidad que detectan el movimiento y la posición espacial.

Las características de la cámara RGB son:

- HairColor: Contiene el color del pelo del usuario.
- Scale: Contiene el factor de escalado de la cara.
- SkinColor: Contiene el color de la cara del usuario.
- TriangleCount: El número de triángulos del modelo 3D que forman la cara del usuario.
- TriangleIndices: Contiene el conjunto de índices de todos los triángulos que forma el modelo.
- VertexCount: Contiene el número de vértices en el modelo de la cara.

Las características obtenidas a partir de la malla generada por el detector de profundidad son:

- FaceBoundingBox: Contiene un objeto rectángulo, que contiene la malla de la cara. Este objeto rectángulo en Unity contiene toda la información sobre los vértices, el tamaño, escalado, ancho y largo.
- FaceOrientation: Contiene un vector de tres coordenadas que indican la orientación de la cara, en ángulos relativos al centro del detector de Kinect.
- HeadPivotPoint: Obtiene el punto de la cabeza a partir de la cual gira.
- Quality: Devuelve un valor de la calidad de la medición de la cabeza.

Debido a nuestro interés en estimar la dirección de la mirada a partir del movimiento de la cabeza, los datos de interés que se emplearán en los diferentes métodos serán los ángulos de orientación de la cara, y el rectángulo que la contiene, para conocer su posición relativa respecto al centro del detector.

6. Reconocimiento de patrones

En esta sección se estudiará toda la teoría relativa a empleo de clasificadores software de patrones, de forma general. Primero se realizará un estudio de los patrones y sus características, con el análisis de características en función de su peso. A continuación se estudiarán algunos métodos de normalización de las características de los patrones, y como pueden mejorar el resultado en función de la distribución estadística de los datos. En el siguiente apartado se explicarán los diferentes tipos de clasificadores de forma general. Por último, se analizarán los dos clasificadores que se emplearán en los métodos de detección, desde sus fundamentos teóricos a su estructura de programación, de forma genérica.

6.1. Estudio de los patrones y sus características

Para comenzar, se debe explicar qué es un patrón. Según la definición empleada en [20] es “lo contrario del caos; es una entidad, vagamente definida, a la cual se puede dar un nombre”. Por ejemplo, un patrón puede ser una letra escrita en cursiva a mano, una cara humana o una imagen de una mano.

Dado un patrón, su clasificación puede consistir en una de estas dos tareas:

- Clasificación supervisada: En esta, se asigna a un patrón de entrada identificado una clase predefinida previamente.
- Clasificación no supervisada: En ella el patrón se asigna a una clase desconocida, que se va distinguiendo de las demás durante un aprendizaje. En él, las clases se van diferenciando en función de la similitud entre grupos de patrones.

El interés en el área de reconocimiento de patrones ha ido creciendo notablemente hasta la actualidad, debido a su extensiva aplicación en diferentes campos, como se puede ver en la tabla Tabla 6.1. Además, esto permite a los ordenadores la habilidad de reconocer racionalmente y tomar decisiones en función de ello de forma autónoma, de forma similar a como hace el ser humano. Por ejemplo, cuando un bebe empieza a aprender el abecedario, empieza a reconocer las letras por la diferencia de formas entre ellas, es decir, aprende a distinguir que unas son altas o bajas, unas tienen curvas y otras líneas rectas y demás diferencias. Este aprendizaje basado en la diferenciación permite no solo conocer las letras, sino también que cuando una persona adulta tiene que leer una tipología de letra que no conoce, no es necesario volver a aprenderla de cero, ya que siempre que esa tipología comparta las características con las que él conoce, como la forma y dimensiones de las letras, será capaz de reconocer las letras aunque sea la primera vez que las vea con esa morfología [20].

Dominio del problema	Aplicación	Patrón de entrada	Clases de patrones
Bioinformática	Análisis de secuencias	ADN/Secuencia de proteínas	Tipos de genes conocidos
Clasificación de documentos	Búsqueda en Internet	Texto de un documento	Categorías semánticas (Negocios, educación, deportes,…)
Análisis de imágenes	Máquina de lectura para ciegos	Imagen de un documento	Caracteres, palabras
Automatización industrial	Inspección de circuitos integrados	Imagen de rango	Defectos en el producto
Reconocimiento biométrico	Identificación personal	Cara, Iris, huella dactilar	Autorización de usuarios
Reconocimiento de voz	Teleoperador automático	Patrones de onda de las palabras	Palabras enunciadas

Tabla 6.1: Aplicaciones del reconocimiento de patrones.

En este punto, es importante hablar de las características que forman un patrón. Como se ha mencionado antes, en un sistema de clasificación de patrones se busca diferenciar unos patrones de otros, para formar clases diferentes en función de sus características. Para ello las características de las diferentes clases deben estar diferenciadas, de manera que, a partir de un patrón dado, por similitud, se pueda tomar la decisión de qué clase se ajusta mejor a ese patrón y, por tanto, conocer qué información representa, aunque sus características no sean idénticas a las que tiene esa clase.

Siguiendo con el ejemplo del reconocimiento de las letras, una característica posible para su reconocimiento es su altura, así tenemos letras que ocupan la altura de una línea, otras que sobresalen de esa altura hacia arriba y otras que sobresalen hacia abajo. Sin embargo, con esa única característica solo podemos dividir las letras en tres clases, pero no podemos diferenciarlas de forma inequívoca unas de otras. Para ello necesitamos que nuestro patrón posea otra serie de características, como pueden ser su ancho, con lo que podríamos, por ejemplo, empezar a generar más clases, aumentando las diferencias.

Como se puede ver en el ejemplo anterior, para realizar una clasificación adecuada, en la que se distinga el conjunto de información que se quiere diferenciar, es fundamental que las características contenidas en cada patrón aporten información relevante y suficiente. Para ello, existen diferentes métodos estadísticos que permiten conocer la información que aporta cada característica, si es diferenciada y útil; o si no

tiene peso (valor) suficiente para aportar información relevante, generando tiempo de procesamiento inútil y ruido.

6.2. Métodos de análisis y selección de características

Hay dos razones fundamentales por las cuales la dimensión de los patrones, es decir, el número de características de cada patrón, debe de ser lo más pequeña posible: debido al coste del procesamiento y debido a la precisión de clasificación. La reducción de la dimensión simplifica la representación y clasificación de dichos patrones. En consecuencia, el clasificador resultante será más rápido y empleará menos memoria. Además, un número menor de características provocará que el número de muestras de entrenamiento necesarias para que el clasificador aprenda será menor, lo cual es fundamental debido a que el número de muestras de entrenamiento siempre es limitado.

Por otro lado, reducir el número de muestras puede provocar una pérdida de información relevante, lo que causará una reducción en la potencia de discriminación entre patrones. En consecuencia, también puede provocar una reducción de la precisión en la clasificación.

Debido a este problema, es importante tener en cuenta algunos métodos de evaluación y selección de características, de forma que se realice un análisis de las mismas que permita discernir entre aquellas que son útiles, y las que son descartables. Estos métodos se muestran explicados en la Tabla 6.2 [20]:

Método	Funcionamiento	Comentarios
Búsqueda exhaustiva	Evalúa los resultados de error del clasificador en función de todas las posibles combinaciones de características en el patrón.	Garantiza encontrar la combinación óptima, pero no es aplicable a patrones con un número elevado de características.
Búsqueda "***Branch-and-Bound***"	Similar a la búsqueda exhaustiva, pero emplea solo un pequeño subconjunto de muestras para evaluar los resultados de error en el clasificador.	Garantiza encontrar la combinación óptima, pero para una elevada complejidad del patrón el algoritmo tiene una carga exponencial.
Búsqueda de la mejor característica individual	Se evalúa la dispersión de cada característica de forma individual, ordenándolas de mayor a menor peso.	Es notablemente simple, y permite eliminar ciertas características concretas.

Método	Funcionamiento	Comentarios
Búsqueda secuencial por adición	Elige la mejor característica de forma individual, y va añadiendo características, las cuales en combinación con la escogida minimizan el error cometido.	Es computacionalmente atractivo, para seleccionar un conjunto de características pequeñas, ya que analiza solo (escogidas-1) posibles resultados.
Búsqueda secuencial por descarte	Comienza con todas la características, y va probando los resultados de eliminar una en cada iteración	Debido a eliminar características aleatorias no puede alcanzar el subconjunto óptimo, y requiere más procesamiento que el anterior.
Selección "***Plus l-take away r***"	Subdivide el conjunto de características, eliminando algunas mediante búsqueda por adición (l) y otras por búsqueda por descarte (r).	Evita el problema del anidamiento en los anteriores, pero requiere escoger los valores de l y r (l > r).
Búsqueda dinámica secuencial por adición y búsqueda dinámica secuencial por descarte.	Una generalización del anterior, en el que los valores de l y r se calculan automáticamente y se van actualizando.	Es la opción óptima suponiendo un coste computacional viable.

Tabla 6.2: Métodos de selección de características.

Como conclusión, estos métodos de análisis de características son algoritmos cuyo objetivo es encontrar la mejor combinación de características conociendo los patrones de entrenamiento, tomando la decisión de compromiso entre calidad de la evaluación de características y coste computacional. Si bien los primeros son capaces de encontrar la solución óptima para el problema concreto, son inviables debido al alto procesamiento requerido para patrones con un número de características mínimamente elevado. En contraposición, los más simples y rápidos pueden ser poco realistas a la hora de solventar el problema, ya que pueden existir escenarios en los cuales aporten una información distorsionada que empeore el sistema.

6.3. Preprocesado de patrones, normalización.

El preprocesamiento de los datos es una conjunto de técnicas que consiste en la modificación de las características, de forma que formen parte de un rango controlado, al mismo tiempo que conserven sus características estadísticas para que puedan ser

procesadas por los clasificadores. Entre ellas, la técnica principal es la normalización, la cual se trata de realizar un reescalado de los valores de forma que pertenezcan a un rango específico, típicamente de 0 a 1. Este proceso mantiene las diferencias estadísticas de las características de los patrones, pero modificando los valores absolutos para facilitar su procesamiento.

Estas técnicas son fáciles de aplicar, empleando algoritmos simples como transformaciones de redimensionamiento como normalización media-cero o decimal. Además, son muy útiles para los casos en los que los datos son muy irregulares, con valores muy altos o muy bajos, o en caso de que no sigan distribuciones Gaussianas. Esta normalización ayuda a mejorar los datos redistribuyéndolos de forma que se aproximen más a distribuciones ideales mejor soportadas por los clasificadores [21].

Dentro de los métodos de normalización, cabe destacar los siguientes:

- **Normalización Máx-Min:** Este método realiza un reescalado de las características de los patrones, del rango de valores que los contiene, al rango [0,1]. Para ello se aplica la siguiente fórmula:

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

  donde max_A es el valor máximo de una característica en el conjunto de todos los patrones, min_A es el valor mínimo que toma una característica en el conjunto de todos los patrones, new_max_A es el nuevo máximo de esa característica (1 en nuestro caso) y new_min_Aes el nuevo mínimo de nuestro rango (0 en nuestro caso). Así cada característica se reescala de forma individual para todos los patrones, quedando todas ellas conservando la misma distribución estadística pero contenida entre [0,1].

- **Normalización Z-score:** También llamada normalización media-cero, esta técnica usa la media y varianza de cada característica para todo el conjunto de patrones de entrenamiento para llevar a cabo la normalización. Para ello emplea la siguiente fórmula:

$$v' = \frac{v - \mu_A}{\sigma_A}$$

  donde μ_A es la media de esa característica en el conjunto de todos los patrones y σ_A es la varianza de esa característica. Esto normaliza cada característica de forma que la distribución resultante tiene media cero y varianza 1. Esta media y varianza calculadas deben ser empleadas en el sistema final, de forma que la información eliminada de las muestras sea conocida por el clasificador. En otro caso, la respuesta será errónea ya que el entrenamiento se realizará con datos diferentes a los de evaluación. La gran ventaja de esta normalización estadística es que reduce el efecto de

los outliers en los datos (los datos que aleatoriamente pueden tomarse de valores no esperados).

- **Normalización de escala decimal:** Consiste en modificar el conjunto de características por un factor de 10, o lo que es lo mismo, desplazar la coma que marca los decimales a izquierda o derecha. Para ello se emplea la fórmula:

$$v' = \frac{v}{10^m}$$

donde m es el entero menor tal que $|v'| < 1$.

6.4. Teoría general de clasificadores

Como se ha visto en los apartados anteriores, un clasificador estadístico consiste en un software que, dado un patrón dado con una serie de características, y el cual ha sido previamente entrenado con un conjunto de patrones similares, es capaz de asignar a ese patrón una clase en función de dichas características. Estos clasificadores siguen una estructura como se ve en la Figura 6.1 [20]. Se debe tener en cuenta que se habla de los clasificadores con clasificación supervisada, los cuales van a ser los únicos estudiados en este trabajo.

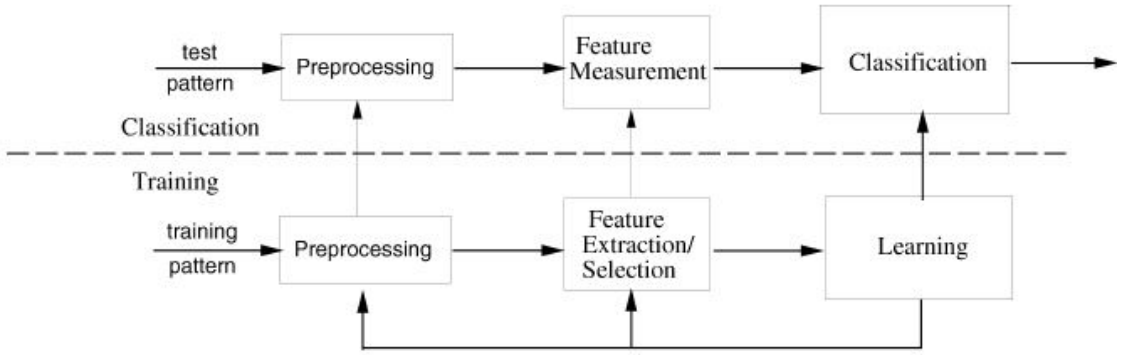

Figura 6.1: Estructura de un clasificador estadístico.

Las diferentes fases que sigue el clasificador, desde que se inicia hasta que devuelve el primer patrón clasificado en una clase son las siguientes:

- Entrenamiento: Se selecciona un conjunto de patrones numeroso y representativo, del cual se conoce la clase a la que debe pertenecer previamente. Este se hace pasar por un preprocesamiento, en el que sus características se normalizan, elimina el ruido de todas las operaciones necesarias para obtener un conjunto compacto y representativo. Después se extraen las características que se van a emplear en el sistema, preparando los patrones adecuados para el entrenamiento. Por último, se realiza dicho entrenamiento, en el cual, a partir de los patrones y la clase

esperada de cada uno de ellos, se reparte el espacio muestral en las clases en función de dichos patrones.

- Clasificación: Una vez que el sistema ya está entrenado, se van realizando capturas de nuevos patrones, los cuales se desconoce a priori la clase a la que pertenecen. Primero sufren el mismo preprocesamiento que los patrones de entrenamiento, para ser similares en cuando a características y que el clasificador pueda asignarles la clase de forma adecuada. A continuación, se extraen las características y se envían al clasificador. Por último, este las evalúa, indicando como salida la clase a la que pertenecen las mismas.

Cabe destacar que en función de las características de los patrones y el número de ellas por patrón, requerirá al clasificador más patrones de entrenamiento, para que pueda asignar espacios muestrales lo más correctos posibles a cada clase. Pero, debido a que el número de patrones de entrenamiento siempre es limitado, puede haber casos donde no se puedan obtener suficientes valores de entrenamiento para el número de características, lo que es conocido como la maldición de la dimensionalidad.

6.4.1. La maldición de la dimensionalidad

El rendimiento de un clasificador depende de la relación entre el tamaño de los patrones, el número de patrones de entrenamiento y la complejidad del sistema. Es ampliamente conocido que la probabilidad de error en la clasificación aumenta mucho más rápido que el aumento de características en cada patrón. Con un aumento de características por patrón lineal, el aumento de las muestras de entrenamiento necesarias para mantener la misma probabilidad de error aumenta de forma exponencial. Esto es lo que se conoce como “la maldición de la dimensionalidad”.

Esto provoca que cobre gran importancia el análisis de las características, la información aportada por las mismas y el dimensionamiento del tamaño de los patrones. Esto se menciona previamente en la página 59.

Si bien puede ser complicado establecer una regla fija que funcione de forma óptima para todos los sistemas de clasificación, está generalmente aceptado que se deben usar al menos 10 veces más de muestras de entrenamiento que número de características hay por patrón. Además, cuanto más complejo sea el sistema, más muestras de entrenamiento deben usarse. Así, por ejemplo, si tenemos un sistema con 7 características, debemos emplear al menos 70 patrones de entrenamiento por cada clase que el sistema debe diferenciar.

6.4.2. Tipos de clasificadores estadísticos

Una vez estudiado de forma genérica cómo funcionan los clasificadores estadísticos, se van a explicar los diferentes tipos [20]:

- **Comparación de plantillas**: Son los más sencillos, siendo los primeros que surgieron, y se basan en comparar los patrones con ciertas plantillas

de valores obtenidas del entrenamiento. Una vez comparado con todas, se toma la decisión de que el patrón pertenece a la clase más cercana. El patrón reconocido se decide evaluando cambios de escalado, traslaciones y rotaciones. Esto es muy caro computacionalmente, debido a que se tienen que realizar numerosas comparaciones, pero es viable debido a la mejora actual de los procesadores. Algunos ejemplos de este tipo de clasificadores son: *Template matching*, *Nearest Mean Classifier* o *Subspace Method*.

- **Aproximación estadística**: En este tipo de clasificadores, los patrones se representan como puntos de d dimensiones, siendo d el número de características por patrón. El objetivo es dividir el espacio d-dimensional en regiones, dentro de las cuales se van a situar los patrones en función de sus características. Dado el patrón de entrenamiento, el objetivo es que el espacio se divida en clases, en función de la concentración de puntos de una clase en una zona u otra del espacio. Estas zonas se dividen siguiendo la distribución estadística de los valores de cada zona. Algunos clasificadores que siguen esta filosofía son: N*earest Neighbor Rule*, *Bayes plug-in*, *Logistic Classifier*, *Parzen Classifier*, *Fisher Linear Discriminant* o *Support Vector Machine*.

- **Enfoque sintáctico**: Se emplean en reconocimiento de patrones complejos, y se caracterizan por adoptar una perspectiva jerárquica del problema. En esta metodología, se ve cada patrón complejo como un conjunto de subpatrones, que se pueden clasificar individualmente. El conjunto de subpatrones básicos son llamados primitivas, y los patrones complejos se representan en términos de interrelaciones de esas primitivas. Como analogía, se suele comparar este tipo de clasificadores como un lenguaje, en el que los patrones complejos serían las frases; y las primitivas que se obtienen de la división en subpatrones serían las palabras. Este tipo de clasificadores es muy usado en situaciones en las que los patrones tienen una estructura conocida y definida, como formas de onda o imágenes. Un ejemplo de este tipo de clasificadores es Binary Decision Tree.

- **Redes neuronales**: Las redes neuronales se ven como extensos sistemas de computación en paralelo, en los que hay un grandísimo número de procesadores con numerosas interconexiones. Estos modelos emplean gran número de principios de organización (como aprendizaje, generalización, adaptabilidad, tolerancia a fallos y representación distribuida), y han demostrado ser muy útiles en aplicaciones prácticas. Una de sus grandes capacidades es que son capaces de adaptarse a los datos incluso cuando las relaciones no son lineales, debido a su estructura en forma de neuronas, que están interconectadas unas con otros a través del peso entre la salida de la anterior y la entrada de la siguiente. Las redes

neuronales usadas más comúnmente son las basadas en retroalimentación. Estas se organizan en capas, y tienen conexiones unidireccionales entre las neuronas de una capa y la siguiente. El entrenamiento consiste en la actualización de las conexiones y de los pesos de dichas conexiones, lo que las permite adaptarse de forma eficiente a los datos. Algunos ejemplos prácticos son: *Multi-layer Perceptron* y *Radial Basis Network*.

6.5. Clasificadores utilizados

Por último, se van a estudiar en este apartado los dos clasificadores empleados en la estimación de la dirección de la mirada en función de patrones. Se van a estudiar sus fundamentos de forma genérica, su estructura desde su inicialización hasta el reconocimiento de patrones de evaluación, y algunas opciones de configuración disponibles.

6.5.1. MLP: Perceptrón Muticapa

Como se ha estudiado en la sección anterior, MLP (Multi-Layer Perceptron, Perceptrón Multicapa) es un clasificador perteneciente al tipo de redes neuronales. Esta es una red neuronal multicapa, perteneciente al grupo de redes neuronales retroalimentadas. Las características de la misma en cuanto a rendimiento son las siguientes [20]:

- Es muy sensible a las diferentes características de los patrones.
- Es sensible al sobreentrenamiento, debido a su estructura cambiante.
- El entrenamiento es lento.
- Capacidad de clasificación no lineal.
- Confiable, produce valores de clases con porcentaje de acierto elevado.
- Necesita normalización y regularización de parámetros.

El MLP empleado es la solución contenida en la librería OpenCV [22], la cual está soportada y con actualizaciones periódicas, además de contar con documentación detallada y ejemplos funcionales que facilitan la implementación.

Las características de esta implementación son las siguientes:

- Soporta las funciones de activación Identidad $(f(x) = x)$ y la función Sigmoide Simétrica $\left(f(x) = \beta * \frac{(1-e^{-\alpha x})}{(1+e^{-\alpha x})} \right)$.
- Soporta entrenamiento tanto con propagación inversa como propagación directa.
- Permite elegir el número de neuronas en cada capa, y el número de capas ocultas.

- Realiza normalización automática, con el método Max-Min, tanto de los valores de entrada, como de los valores de salida.

Una vez estudiados los parámetros que soporta, vamos a ver cómo se realiza su construcción, desde que se arranca el método hasta que nos devuelve los valores percibidos, como se ve en la Figura 6.2.

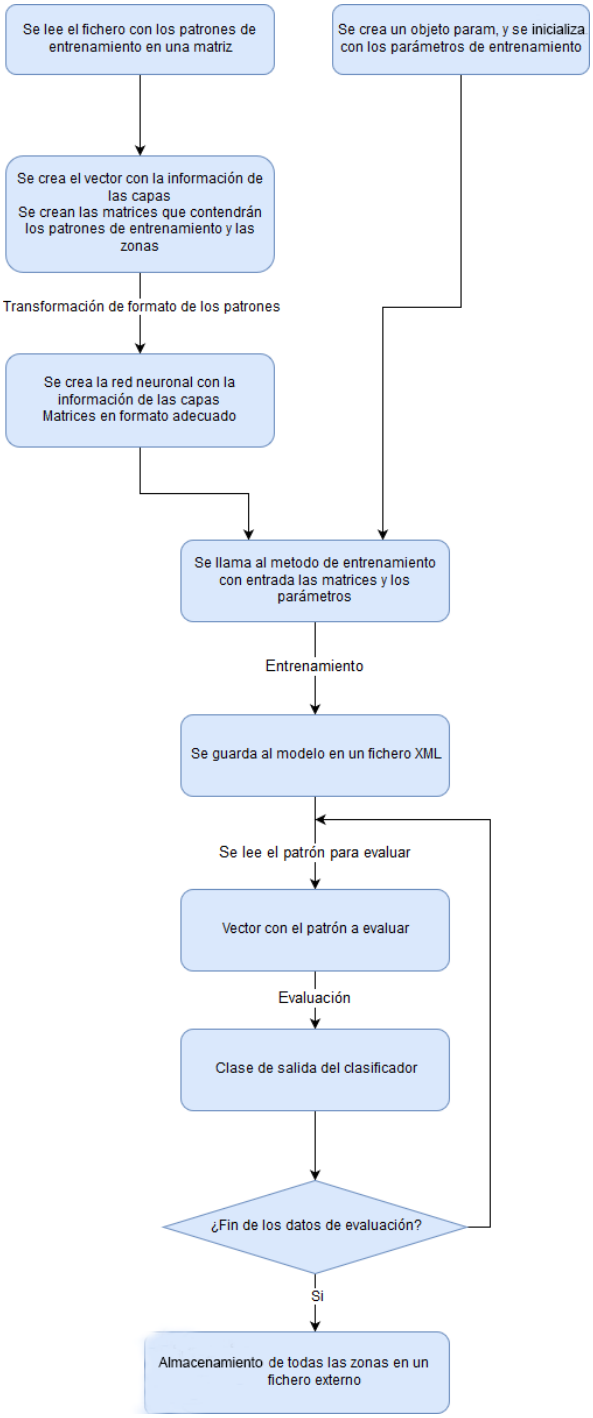

Figura 6.2: Diagrama de flujo del clasificador MLP

Como se puede observar en el diagrama, al inicio se crea el objeto que contendrá los parámetros de entrenamiento. Además, se leen del fichero todos los datos de entrenamiento y se almacenan en una matriz. El siguiente paso consiste en la copia de

estas matrices, tanto de patrones como de zonas detectadas, en los objetos adecuados para que puedan ser utilizadas por la red neuronal. Al mismo tiempo, se inicializan los parámetros de las capas de la red, y se crea el objeto que representa el clasificador.

En el siguiente punto, ya están todos los datos preparados, entonces se pasa a la red neuronal previamente creada las matrices de patrones y de clases, y el conjunto de parámetros de entrenamiento previamente definidos. Una vez finaliza el entrenamiento, se guarda la configuración de la red neuronal, información de las capas, conexiones y pesos de cada conexión, en un fichero xml comprensible para su uso en ejecución.

Una vez ya hemos entrenado el clasificador, vamos capturando patrones de entrada y lo transformamos al formato adecuado para que sea leído por nuestra red. Entonces llamamos a nuestra red neuronal, y ejecutamos el método de evaluación sobre el patrón capturado, devolviéndonos la clase estimada a la que pertenece al patrón.

Por último, almacenamos el valor capturado y repetimos este ciclo, hasta que se acaban los datos para evaluar. Entonces guardamos todas las salidas detectadas en un fichero y finalizamos.

6.5.2. SVM: Máquina de soporte vectorial

Como se ha estudiado en el apartado anterior, este clasificador pertenece al grupo de aproximadores estadísticos, los cuales emplean un espacio vectorial de dimensiones iguales al número de características por patrón, para dividir el espacio en regiones y relacionar cada patrón con la clase más cercana dentro del espacio. Las características de este tipo de clasificadores recogidas en [20] son:

- Depende del escalado de las características.
- El aprendizaje es iterativo y relativamente lento.
- Trabaja con valores no lineales.
- No es sensible al sobreentrenamiento.
- Buenos resultados en aplicaciones generales.

Para la implementación, se va a usar el SVM (Support Vector Machine, Máquina de Soporte Vectorial) integrado en la librería OpenCV [23], la cual está soportada y con actualizaciones periódicas, además de contar con documentación detallada y ejemplos funcionales que facilitan la implementación.

Las características de esta implementación son las siguientes:

- Los tipos de formulación que soporta son:
    - C_SVC: *Support Vector Classification* con separación c para outliers.
    - NU_SVC: *Support Vector Classification* con separación en función del parámetro de decisión calculado durante el entrenamiento.

    - ONE_CLASS: Estimación de la distribución, por la cual todos los patrones pertenecen a una misma clase, separada del resto del espacio.
    - EPS_SRV: *Support Vector Regression*. La distancia entre los vectores de entrenamiento y de evaluación debe ser menor que un parámetro p para ser válidos.
    - UN_SRV: *Support Vector Regression*. La distancia entre los vectores de entrenamiento y de evaluación debe ser menor que un parámetro v calculado en el entrenamiento para ser válidos.
- Los tipos de núcleo soportados son:
    - LINEAR: Se realiza clasificación lineal sin mapeado, es la opción más rápida.
    - POLY: Se emplea un núcleo polinómico.
    - RBF: Emplea un núcleo basado en la función *Base Radial*, con un funcionamiento adecuado en múltiples escenarios.
    - SIGMOID: Emplea un núcleo sigmoidal.
- Permite un modo de entrenamiento automático, en el que los parámetros del modelo se optimizan automáticamente, a través del entrenamiento con subpatrones, hasta que el error de ellos es 0.
- Realiza normalización automática con el método Max-Min, de la entrada.

Una vez estudiadas las múltiples opciones que nos permite el uso de esta librería, vamos a estudiar el flujo del programa desde que se inicia hasta que se obtiene la zona detectada. Para ello se emplea la Figura 6.3.

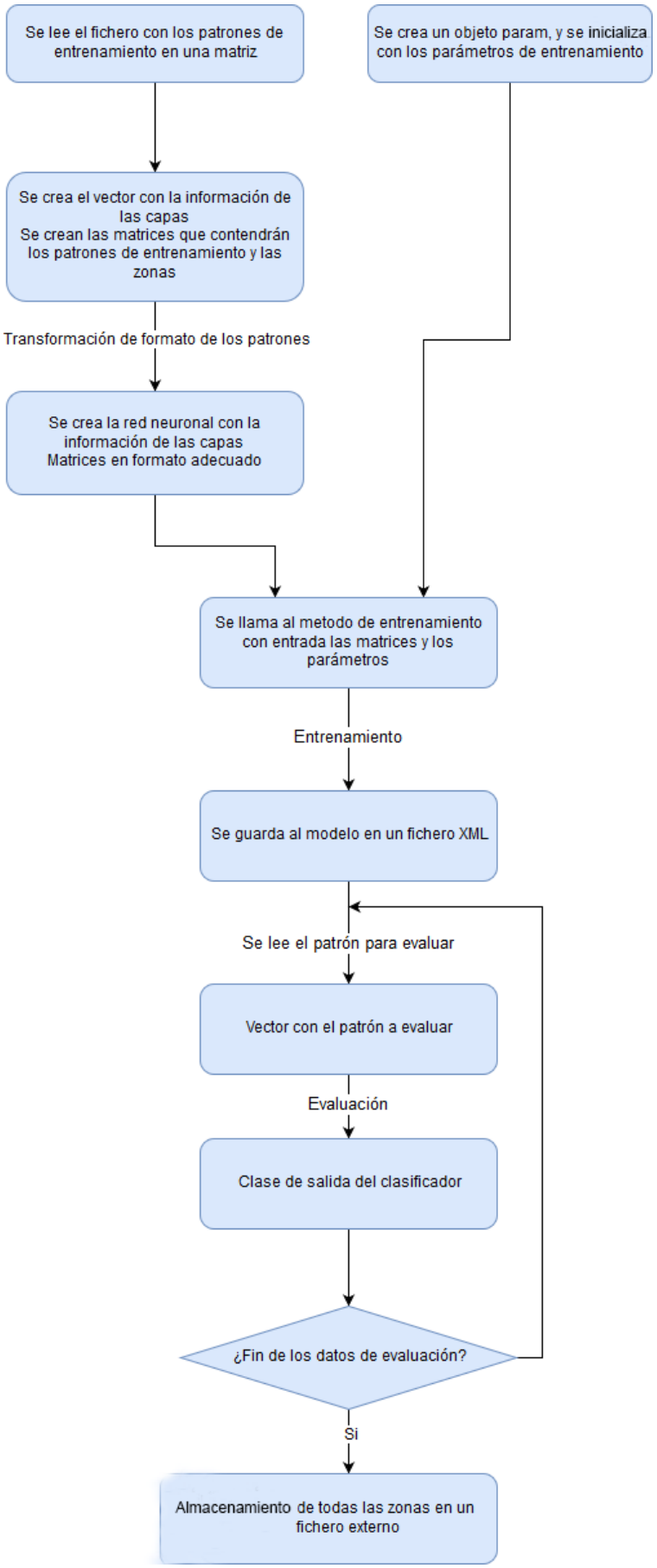

Figura 6.3: Diagrama de flujo del clasificador SVM.

Como se puede observar en el diagrama, al inicio se crea el objeto que contendrá los parámetros de entrenamiento. Además, se leen del fichero todos los datos de entrenamiento y se almacenan en una matriz. El siguiente paso consiste en la copia de estas matrices, tanto de patrones como de zonas detectadas, a los objetos adecuados para que puedan ser utilizadas por la red neuronal. Al mismo tiempo, se inicializan los parámetros de las capas de la red, y se crea el objeto que representa el clasificador.

En el siguiente punto, ya están todos los datos preparados, entonces se pasan, a la red neuronal previamente creada, las matrices de patrones y de clases, y el conjunto de parámetros de entrenamiento previamente definidos. Una vez finaliza el entrenamiento, se guarda la configuración de la red neuronal, información de las capas, conexiones y pesos de cada conexión en un fichero xml comprensible para su uso en ejecución.

Una vez ya hemos entrenado el clasificador, vamos capturando patrones de entrada y los transformamos al formato adecuado para que sean leídos por nuestra red. Entonces llamamos a nuestra red neuronal, y ejecutamos el método de evaluación sobre el patrón capturado, devolviéndonos la clase estimada a la que pertenece el patrón.

Por último, almacenamos el valor capturado y repetimos este ciclo, hasta que se acaban los datos para evaluar. Entonces guardamos todas las salidas detectadas en un fichero y finalizamos.

7. Estimación de la dirección de la mirada

En esta sección, se explicará por qué y cómo se realiza la división de la pantalla en diferentes regiones. Además, se explicarán los diferentes métodos diseñados para realizar la estimación de la dirección de la mirada, desde las características principales y su procesamiento, la forma de construir el decisor de regiones, la calibración necesaria para adaptar el sistema al usuario, hasta un guía de funcionamiento y manual de usuario.

7.1. División de la pantalla en regiones

Para comenzar, vamos a estudiar cómo es la interfaz de usuario y la forma de dividir la misma en unidades de información. Para ello, se representan las mismas en la Figura 7.1.

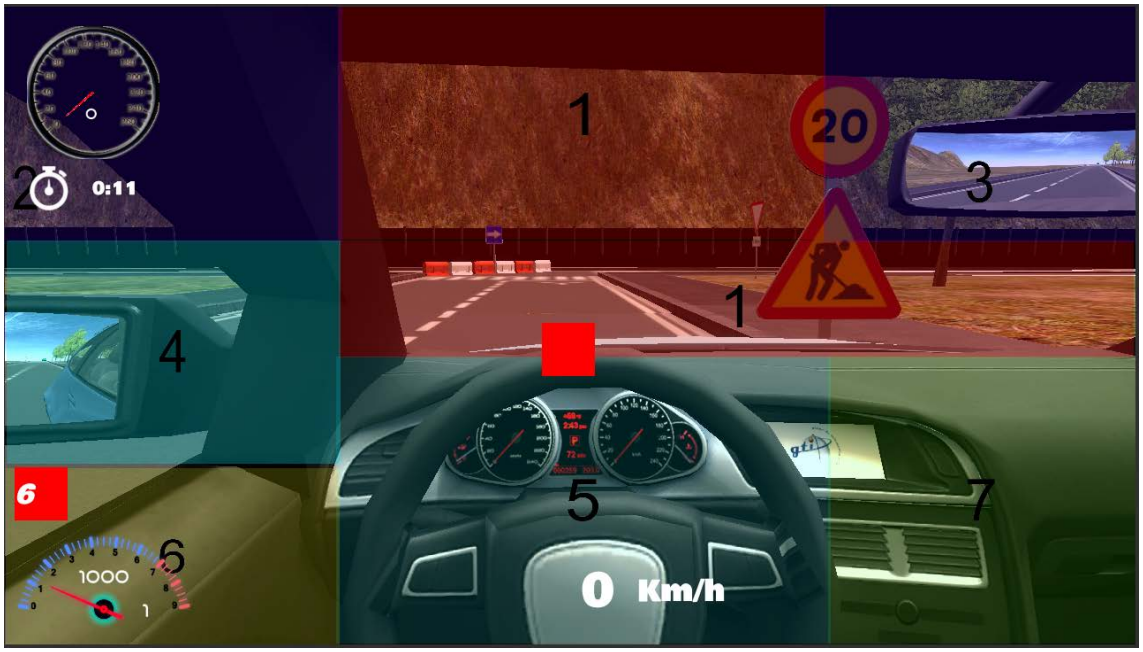

Figura 7.1: Pantalla del simulador dividida en regiones de detección.

Como se puede ver en la imagen, hay siete regiones importantes con información de interés, que son diferenciables:

- Región 1: Es la vista central del conductor, y abarca la luna delantera visible. Es la posición que más se mira, y a partir de la cual se producen los desplazamientos de forma mayoritaria al resto de zonas.
- Región 2: Es la zona superior izquierda. Es pequeña, y en ella se sitúan el cuentakilómetros y el tiempo de simulación.
- Región 3: Es la zona superior derecha, y en ella se coloca el espejo interior.
- Región 4: Es la zona central izquierda. Es pequeña, y en ella está contenida el retrovisor izquierdo.
- Región 5: Zona inferior central de la pantalla, donde está situado el volante y la velocidad.

- Región 6: Es la parte inferior izquierda, y contiene el cuentarrevoluciones del coche.
- Región 7: Es la zona inferior derecha, y no contiene ninguna información relevante para el conductor. Simplemente es el paragolpes.

7.2. Método 1: Estimación bidimensional con decisor estático

Este es el primer método desarrollado, y también el más sencillo. En él se emplean los ángulos básicos, con los cuales a través del calibración y algunas transformaciones se evalúa la coordenada x e y en la que se sitúa la mirada. Entonces, a partir de un decisor de zonas precalculado de forma fija para cada uno de los usuarios, se decide la zona a la que está mirando.

7.2.1. Preprocesamiento y selección de parámetros

Primero, vamos a explicar los parámetros elegidos, los cuales son el *yaw* y el *pitch* de la cabeza recogidos por Kinect. El *yaw* es el ángulo equivalente al movimiento de izquierda a derecha, similar al que realizamos cuando decimos que no a algo. Por el contrario, el *pitch* es el ángulo que mide el movimiento de arriba hacia abajo, como al decir que sí. Se decide emplear estos dos parámetros, descartando la inclinación lateral, debido a que no es un movimiento que se realice de forma natural dentro de este escenario, como se recoge en [24]. Por otro lado, suponemos que en el sistema el usuario acompañará en cierta medida el movimiento de la mirada con la cabeza, evaluando esta característica dentro de la calibración.

Una vez tenemos los parámetros a emplear, vamos a realizar una serie de transformaciones que nos permitan simplificar el sistema.

Primero, vamos a modificar los grados devueltos por Kinect, ya que formatea los mismos en el rango [0,360]. Para ello vamos a modificar este rango para que vaya de [-180,180] en ambos ejes. Con esta transformación se modificarán los ejes para que sean similares a la disposición habitual. Esta transformación se representa en la Figura 7.2.

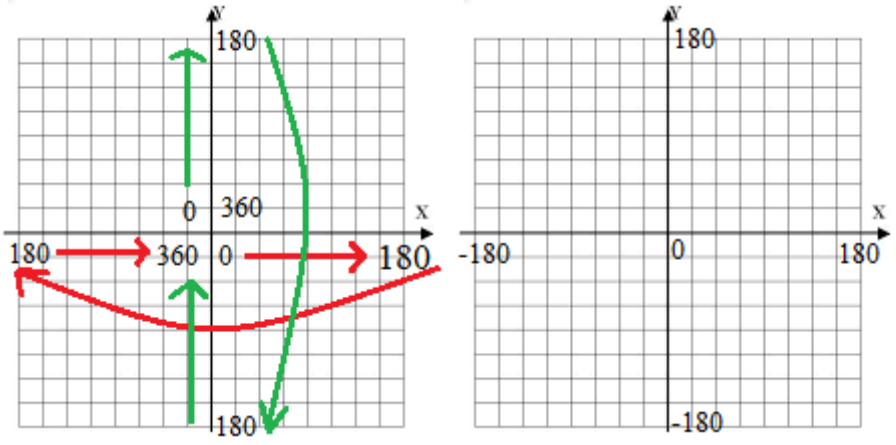

Figura 7.2: Modificación de los ejes de *yaw* y *pitch*.

Una vez transformados los ejes para situar el punto central el (0,0), vamos a emplear la distancia conocida entre el usuario y la pantalla, para transformar los ángulos en centímetros, que correspondan a distancia recorrida sobre la superficie de la pantalla. Para ello necesitamos conocer previamente la distancia entre el usuario y la pantalla, la cual se introducirá manualmente en la fase de calibración. A partir de ella, aplicamos las siguientes transformaciones:

$$x\_move = tg(new\_yaw) * screen\_distance$$

$$y\_move = tg(new\_pitch) * screen\_distance$$

Siendo x_move el desplazamiento en centímetros del giro de la cabeza sobre el eje x, y_move el desplazamiento en centímetros del giro de la cabeza sobre el eje y, y screen_distance la distancia del usuario a la pantalla. Esto se representa además sobre la Figura 7.3.

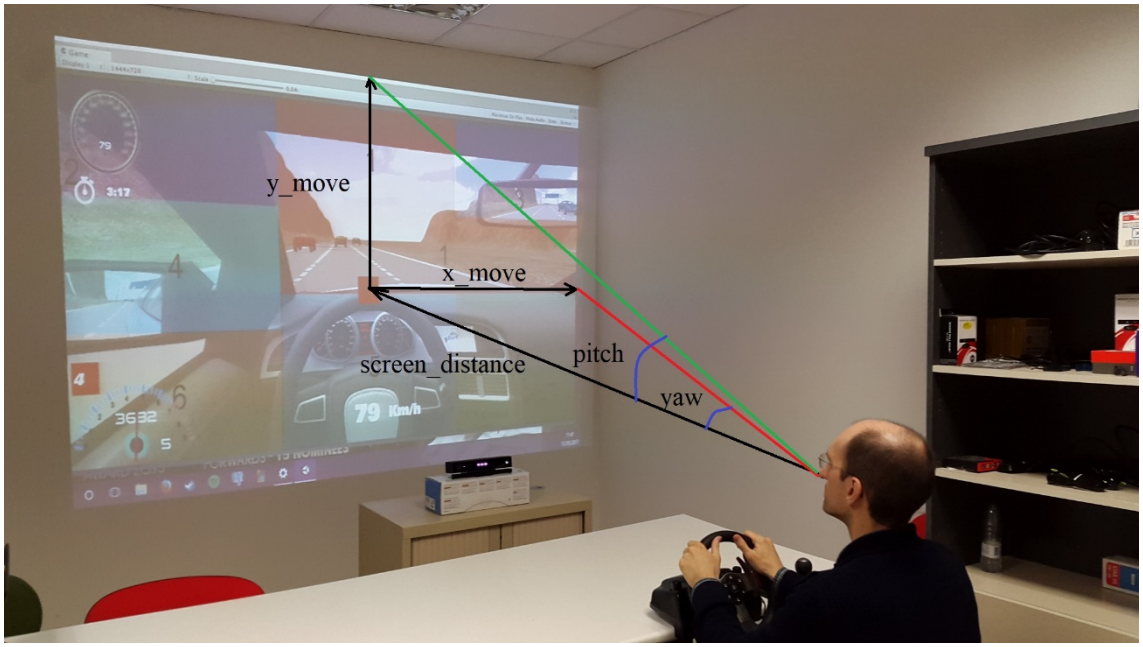

Figura 7.3: Transformación de grados a centímetros sobre la pantalla.

Con estas transformaciones, ya tendríamos preparados para procesar los dos parámetros de entrada de este método, que son la distancia del desplazamiento de la cabeza sobre la pantalla sobre ambos ejes. Es importante destacar que para la estimación se emplearán una serie de valores que se encargarán de compensar la diferencia de movimiento entre la mirada y la cabeza, que serán explicados durante el capítulo siguiente de calibración.

7.2.2. Calibración del sistema

En este capítulo se van a estudiar los diferentes parámetros que, añadidos a los valores procesados en el capítulo anterior, van a permitir realizar una estimación de la dirección de la mirada a partir de los ángulos capturados.

Primero, se pedirá al usuario que, en posición de conducción, mire al centro de la pantalla, para obtener nuestros valores de referencia. A partir de ellos, se emplearán como centro de coordenadas y todos los valores serán modificados de tal forma que empleen como punto de referencia el punto central. Para ello se emplearán las siguientes transformaciones:

$$x\_move = x\_move - x\_init$$

$$y\_move = y\_move - y\_init$$

Siendo x_init e y_init las coordenadas del punto capturado cuando el usuario mira al punto central.

En el siguiente paso, se debe abstraer el sistema para suponer una situación ideal. Suponiendo que la cabeza acompaña con su movimiento a la mirada, podemos entender el desplazamiento de la mirada sobre la pantalla como la suma de la contribución de dos movimientos: el movimiento de la cabeza más el movimiento de los ojos. Sabiendo además, que el movimiento de la cabeza lo tenemos perfectamente caracterizado por Kinect, y que se produce una relación directa, se toma la decisión de emplear como estimador un valor escalar, obtenido durante la fase de calibración, tal que:

$$si\ x > 0 \rightarrow x = x\_move * escalado\_x\_pos$$

$$si\ x < 0 \rightarrow x = x\_move * escalado\_x\_neg$$

$$si\ y > 0 \rightarrow y = y\_move * escalado\_y\_pos$$

$$si\ y < 0 \rightarrow y = y\_move * escalado\_y\_neg$$

Siendo x e y las distancias finales empleadas para estimar el movimiento de la mirada sobre la pantalla, y escalado los cuatro valores obtenidos de los cuatro ejes.

Así, durante la fase de calibración se pedirán al usuario cinco situaciones diferentes:

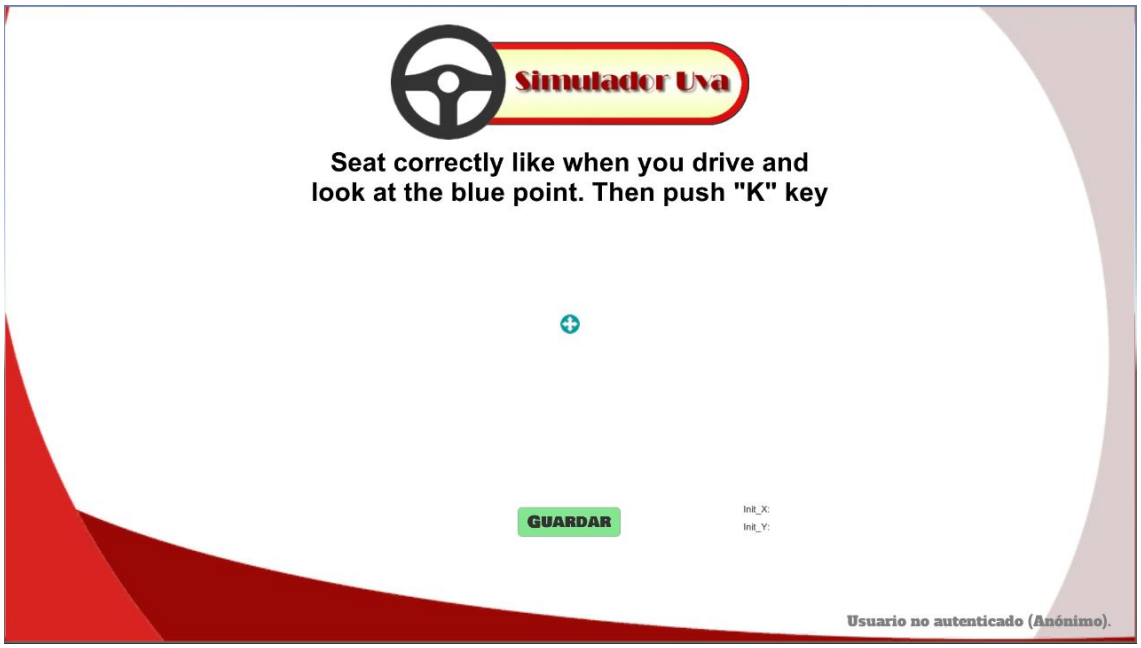

Figura 7.4: Pantalla del punto central de calibración

- Primero mirar al punto central de la pantalla, del cual se obtendrán las coordenadas del punto central, para usar dicho punto como referencia. Este se muestra como el punto azul en la Figura 7.4.

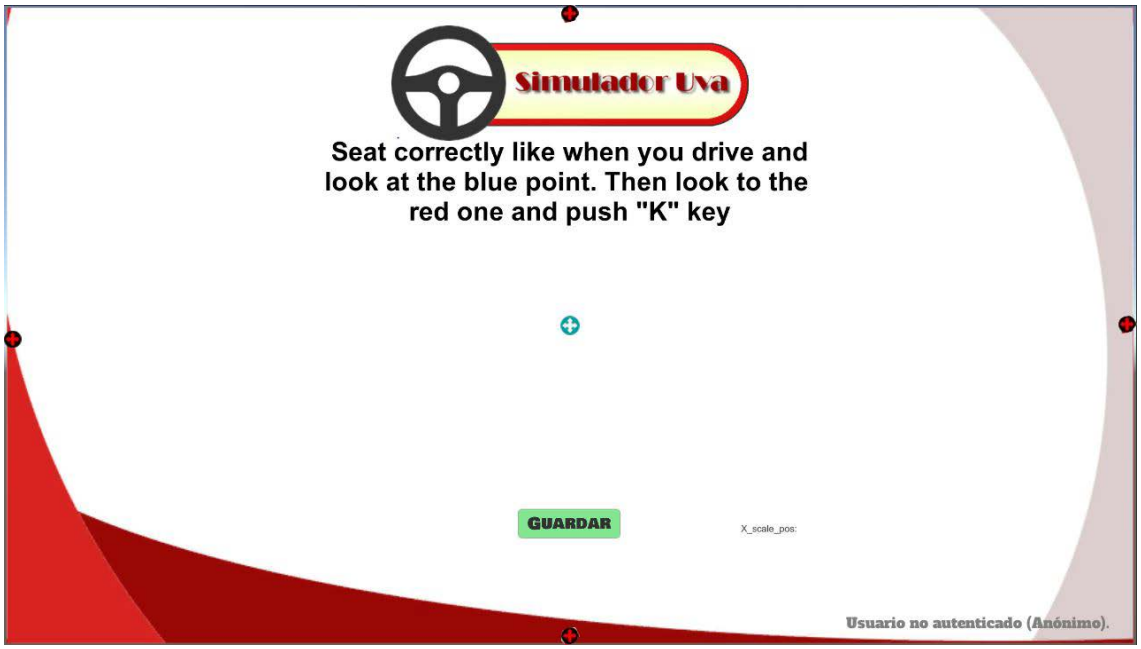

Figura 7.5: Puntos de calibración del escalado de la cabeza para estimar la mirada.

- Mirar sucesivamente desde el centro de la pantalla al borde de la misma en los cuatro ejes, en los cuatro puntos que se muestran en rojo en la Figura 7.5, y de los cuales se calculará el escalado con las fórmulas:

$$|escalado\_x\_pos| = \frac{x\_move}{screen\_wide}$$

$$|escalado\_x\_neg| = \frac{x\_move}{screen\_wide}$$

$$|escalado\_y\_pos| = \frac{y\_move}{screen\_high}$$

$$|escalado\_y\_neg| = \frac{y\_move}{screen\_high}$$

Por último, como último valor necesario para conocer dicho escalado, faltaría introducir el ancho de la pantalla, el cual también es fijo, y la altura, que se obtiene a partir del ancho, conocida la relación de aspecto (16:9).

7.2.3. Constructor del decisor

Una vez tenemos todos los parámetros de entrada configurados y en disposición de ser utilizados en el sistema, vamos a construir el decisor de zonas. Este se encargará de evaluar, a partir del desplazamiento estimado de la mirada, la zona a la que se está mirando. Para ello recibirá como entradas x e y, previamente calculados como el desplazamiento en centímetros multiplicado por el escalado en cada eje, como se ve en el capítulo anterior.

La división de la pantalla en regiones se realiza como está explicada en la sección anterior. Además, sabemos cuánto mide de alto y ancho la pantalla, porque son valores necesarios para realizar el calibración. Las regiones tienen una disposición fija, que no cambia durante la ejecución. Por tanto, podemos saber el tamaño relativo de las mismas con relación a estos valores.

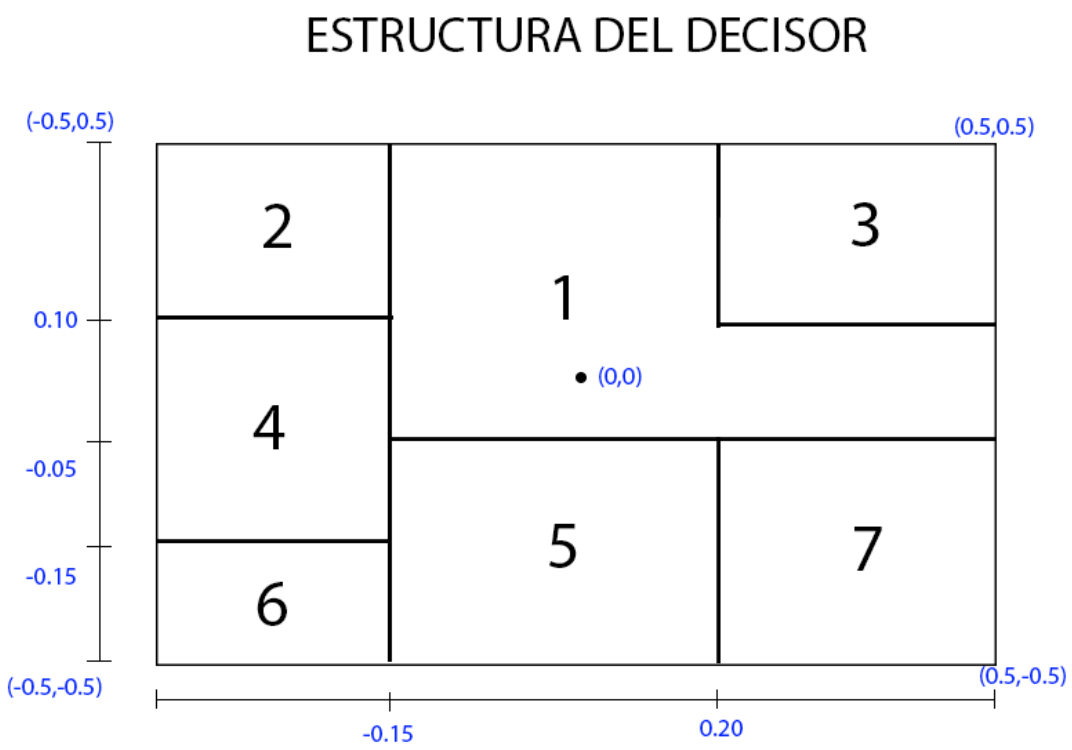

Figura 7.6: Estructura del decisor en función de las diferentes regiones.

Como se muestra en la Figura 7.6, los tamaños relativos de las secciones nos permiten que, multiplicando dichos valores por el ancho y largo de la pantalla respectivamente, conocer la medida absoluta de las mismas. Dado que el sistema es capaz de calcular el desplazamiento de la mirada en centímetros, podemos, en función de las medidas, averiguar a qué zona pertenece en cada instante, y calcular la posición a la que está mirando el usuario.

Por ejemplo, para detectar la zona 3 se tendría que cumplir:

$$si\ x > (0{,}2 * screen\_wide)\ \&\&\ y > (0{,}1 * screen\_high)$$

7.2.4. Funcionamiento general

En este punto, ya estarían preparados todos los componentes necesarios para construir nuestro estimador de la mirada del conductor. En el software en funcionamiento, los pasos serían los siguientes:

- Se arranca el simulador, esperando a que Kinect detecte al conductor y empiece a calcular los valores del seguimiento de la cabeza.
- Una vez que Kinect está en funcionamiento, primero tenemos que introducir, en las opciones, los valores de la distancia del usuario a la pantalla y del ancho de la misma.
- A continuación, se realiza la fase de calibración, en la que el usuario mirará primero al punto central, para estimar los valores iniciales del punto medio de la pantalla. Después irá recorriendo diferentes pantallas en las que se colocará mirando al centro y realizará un desplazamiento natural de la

mirada hacia cada uno de los laterales de la pantalla, para calcular los escalados de los cuatro semiejes.

- En este punto, el sistema ya estaría listo para comenzar las simulaciones, para lo cual se debe volver al menú inicial guardando los resultados, dando a comenzar y seleccionando el escenario.
- Por último, en las opciones previas del escenario, se deberá activar Kinect, que habrá almacenado todos los valores previamente calibrados. Empleando los métodos explicados anteriormente, calculará la estimación de la dirección de la mirada, introduciéndola en el decisor, y devolviendo por pantalla la zona estimada.

7.2.5. Manual de usuario

En este último capítulo de este método, se va a realizar una detallada exposición visual, de todos los pasos que se deben realizar, con las pautas adecuadas, para poner el sistema en funcionamiento de forma correcta.

1. Arrancar el sistema, habiendo medido con un metro el ancho de la pantalla, y la distancia desde la punta de la nariz del usuario en posición de conducción al centro de la pantalla.
2. Una vez en el menú principal, se **debe esperar** a que la frase cambie de "Estado Kinect: OFF" a "Estado Kinect: ON". Una vez esto haya sucedido, se seleccionará en el menú mostrado en la Figura 7.7 la opción "Opciones".

Figura 7.7: Menú principal de simulador de conducción.

3. Aparecerá una pantalla como la que se puede ver en la Figura 7.8. Entonces, deberá introducir la distancia a la pantalla en centímetros en el recuadro debajo de "Distancia a la pared", y el ancho de la pantalla en el

recuadro debajo de “Ancho de la pantalla”. Entonces se pulsará sobre la opción “Calibrar Kinect”.

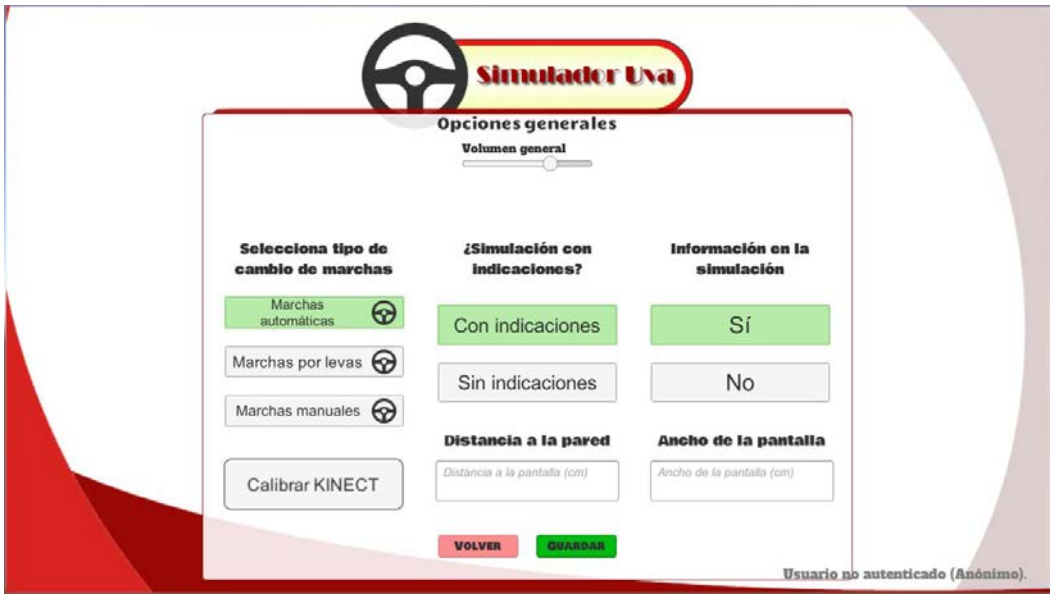

Figura 7.8: Menú de opciones del simulador de conducción.

4. Se abrirá una ventana como la de la Figura 7.9. **Entonces el usuario debe colocarse en la misma posición que tendrá durante la conducción, y mirar de forma natural y fija al punto azul situado en el centro de la pantalla**. En ese momento, el asistente debe pulsar la tecla “K”, en el teclado del ordenador que está lanzando la simulación. Una vez hayan aparecido los valores correspondientes a la izquierda de los textos “Init_X:” e “Init_Y”, se pulsará el botón “guardar” para pasar a la siguiente etapa.

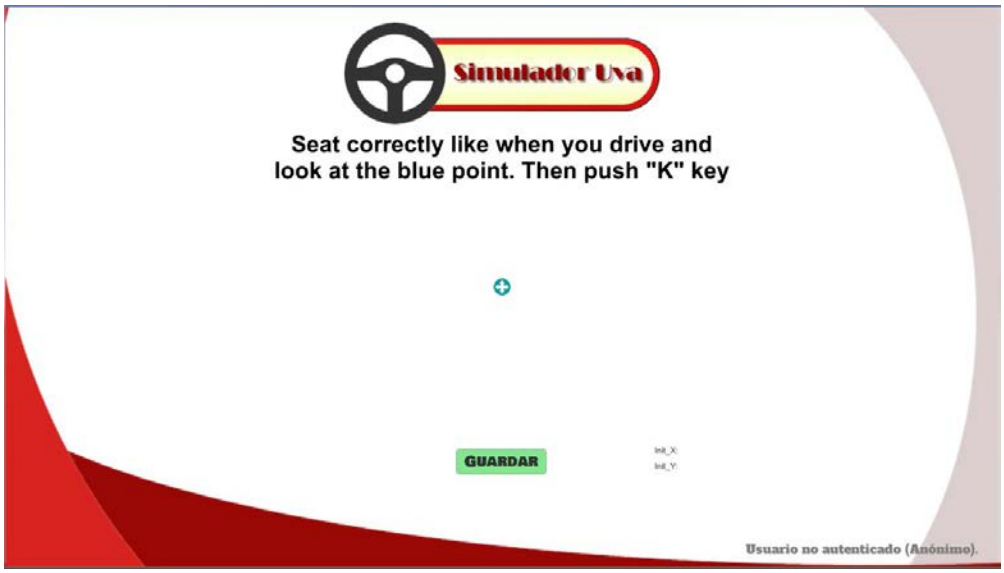

Figura 7.9: Pantalla del calibración para evaluar el punto central.

5. En la siguiente ventana, mostrada en la Figura 7.10, se realizará la calibración del eje X negativo. Para ello, se pedirá al usuario que, en

posición de conducción, se coloque mirando al punto central de color azul. En ese momento, **se solicitará al usuario que mire al punto izquierdo, de forma natural, y se mantenga fijo en esa posición**. En ese momento, el supervisor deberá a volver a pulsar la letra “K”. Entonces en la parte inferior se mostrará el valor del escalado ya calculado, el cual debe pertenecer al rango (1,10) aproximadamente. Debe pertenecer a este rango porque se ha comprobado tras realizar pruebas que el sistema no funciona adecuadamente para los usuarios cuya relación entre la dirección de la mirada y el movimiento de la cabeza es mayor de 10. En caso de no capturarse, o tener valores extraños, se puede repetir el proceso en la misma pantalla todas las veces que sea necesario. Para ello solo es necesario volver a pulsar la letra “K”, y el sistema volverá a calcularlo y mostrarlo. Una vez se muestre un valor satisfactorio, se pulsará el botón guardar para pasar a la siguiente pantalla.

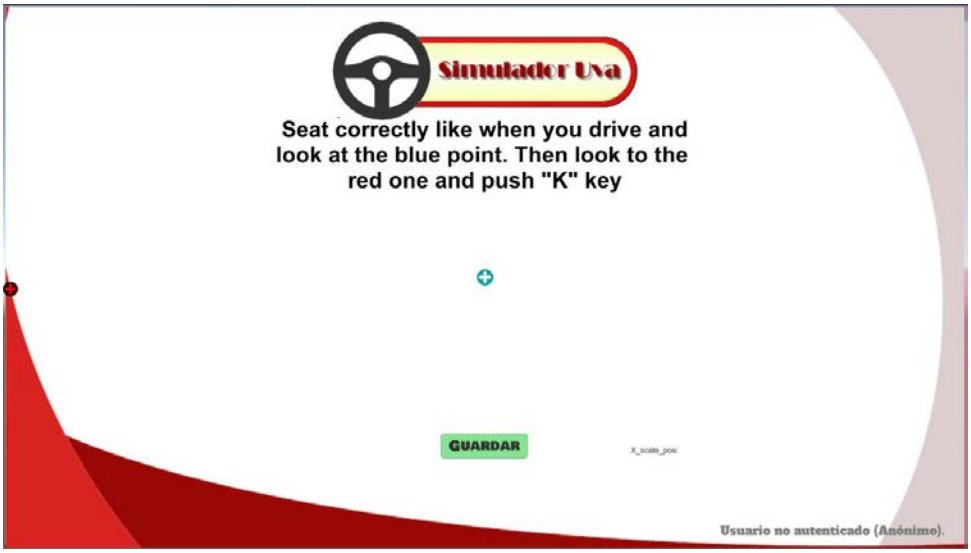

Figura 7.10: Pantalla de calibración del escalado del eje X negativo.

6. De forma a similar en la nueva fase, se realizará el mismo proceso pero esta vez mirando al nuevo punto ubicado en la parte derecha de la pantalla, como se muestra en la Figura 7.11.

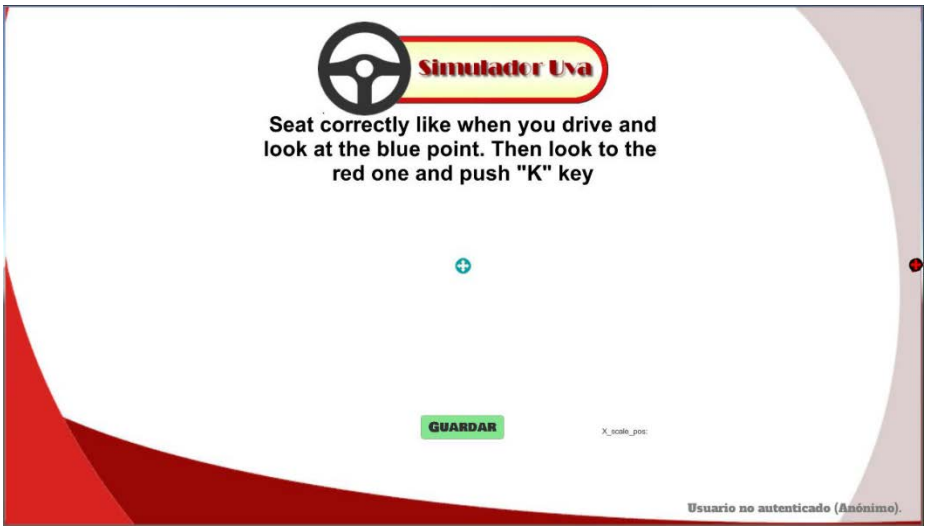

Figura 7.11: Pantalla de calibración del escalado del eje X positivo.

7. De la misma manera en la pantalla siguiente, mostrada en la Figura 7.12, se realizará el mismo procedimiento para calibrar el eje Y positivo.

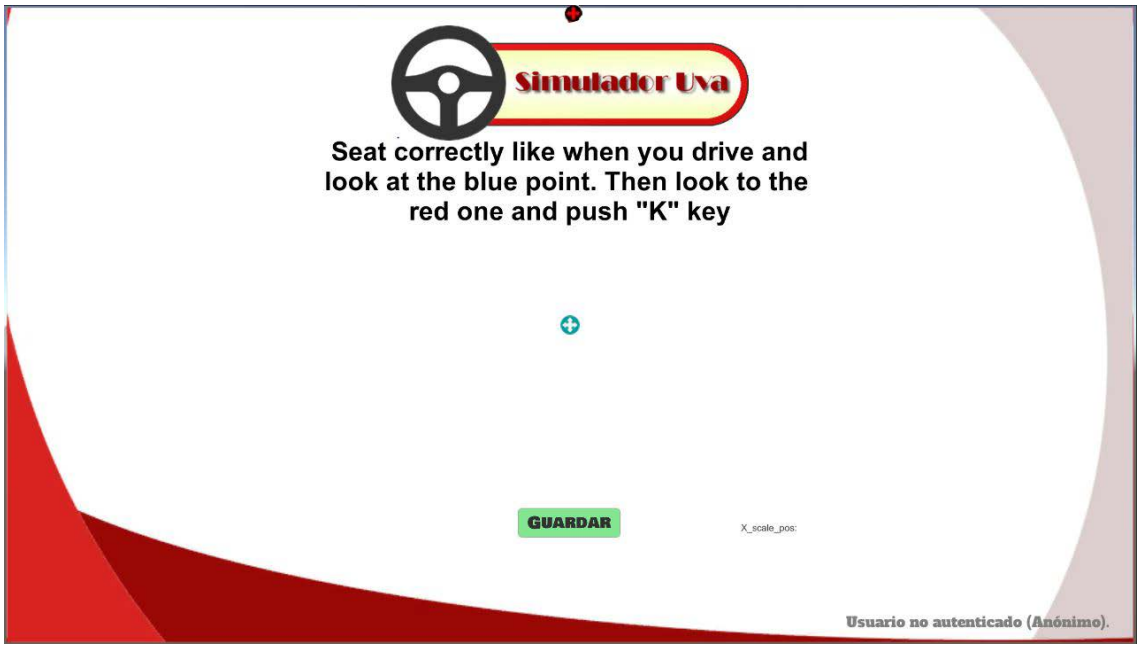

Figura 7.12: Pantalla del calibración del escalado del eje Y positivo.

8. Por último, se realizará el mismo procedimiento para el eje Y negativo, sobre la siguiente ventana, mostrada en la Figura 7.13.

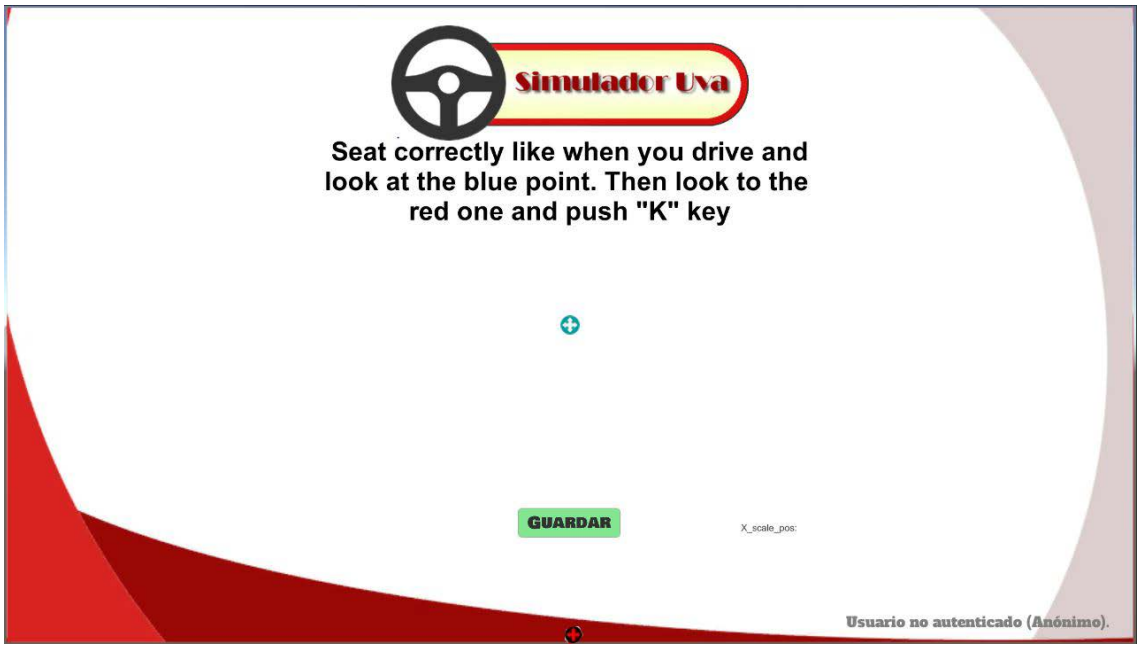

Figura 7.13: Pantalla del calibración del escalado del eje Y negativo.

9. En este punto se volverá al menú de opciones representado en la Figura 7.8. Entonces se pulsará el botón “Guardar”, y se volverá al menú principal, con Kinect ya preparado para comenzar las simulaciones.

10. Desde el menú principal, se podrán comenzar las simulaciones. Una vez se seleccione una, se mostrará inicialmente otra ventana de opciones como la de la Figura 7.14. En ella, **se debe pulsar el botón con el texto “Face tracking”**. Una vez se pulse el mismo, Kinect comenzará a realizar estimaciones de la zona a la que está mirando el conductor. Una vez el resto de opciones estén configuradas en función de las necesidades del usuario, se pulsará el botón “Comenzar”, para iniciar la simulación.

Figura 7.14: Menú de opciones en el escenario antes de comenzar la simulación.

7.3. Método 2: Estimación bidimensional con decisor adaptado

Este método se basa en el método anterior para modificar la forma de construir el decisor, de forma que cada usuario pueda calibrar las diferentes regiones en función de cómo mira a las mismas. También se puede ver este como una evolución, en la que se realiza una estimación en función no solo de la diferencia entre el movimiento de la mirada y de la cabeza, sino también la zona concreta de la pantalla a la que el usuario esté mirando.

Para ello se parte de los mismos parámetros de entrada que el método anterior, y se realiza la misma calibración básica, con la misma forma de procesar los parámetros. Sin embargo, se modifica la forma de construir el decisor, que en vez de tener unos valores fijos medidos sobre la pantalla, pasa a utilizar valores capturados del propio usuario. Esto provoca que las diferentes fronteras entre las regiones dependan de cómo el usuario mira a las mismas, y no de un valor fijo medido previamente.

Esta mejora se debe al problema planteado en [25], que expone que la forma de mirar sobre una pantalla a diferentes regiones no se relaciona de forma directa con el movimiento de la cabeza, sino de forma relativa a los grados de giro necesarios para que una persona pueda ver un objeto dentro de su campo de visión.

7.3.1. Calibración del decisor

Para poder construir nuestro decisor en función del usuario de sistema, se van a calibrar un conjunto de puntos, en función de las regiones expuestas anteriormente. De esta forma, se le pedirá al usuario que mire de forma natural a todo el conjunto de puntos, almacenando los valores de desplazamiento de la mirada obtenidos por Kinect. Una vez estén estos valores almacenados, se emplearán para construir el decisor, de tal forma que se decida que el usuario están mirando a una zona u otra si los valores calculados en tiempo real están dentro del intervalo calculado al mirar a los cuatro puntos frontera de la región.

Para esta calibración, se empleará una nube de puntos como la que se muestra en Figura 7.15.

Figura 7.15: Nube de puntos para calibrar el decidor adaptado.

En ella se pueden observar puntos comprendidos entre el 1 al 23, de los cuales del 1 al 12 se utilizan para evaluar las fronteras en el eje X, por lo que cuando el usuario mire a los mismos solo se capturará el valor X devuelto por el sistema. Del punto 13 al 23, se emplearán para evaluar las fronteras de las regiones sobre el eje Y. Una vez calibrados todos los puntos, se emplearán para construir el decisor que permita evaluar la región donde mira el usuario.

Las regiones de decisión, representadas con los puntos fronteras, se muestran en la Figura 7.16. En ella se ve cómo se relaciona la nube de puntos con las distintas regiones. Cabe destacar que, para que la calibración se realice de forma correcta, se debe mirar a cada punto después de mirar al centro, que en este caso lo representa el punto 18.

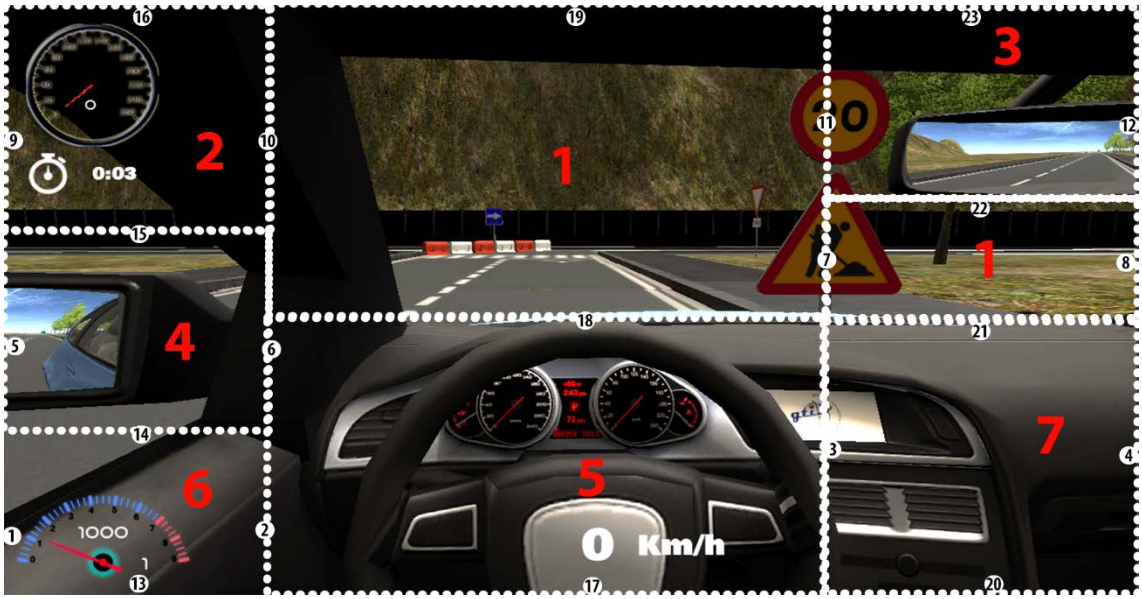

Figura 7.16: Regiones de decisión adaptadas.

7.3.2. Construcción del decisor adaptado.

Una vez calibrado todos los puntos de la nube, la construcción del decisor es relativamente simple. Consistirá en modificar esos valores calculados de forma estática a partir del ancho de la pantalla, por los valores obtenidos de los propios puntos frontera. Es importante destacar que, debido a que las regiones deben ser rectangulares, hay dos regiones 1, que representan a la vista frontal del conductor. Los puntos frontera de cada región se muestran en la Figura 7.16.

Siguiendo con el ejemplo del método anterior, para evaluar la región 3, se emplearía la lógica:

$$si\ x > (x\_p11)\ \&\&\ y > (y\_p22)$$

7.3.3. Funcionamiento general

En este punto, ya tenemos nuestro nuevo método preparado para su uso. Es importante destacar que, debido a que es una evolución del primero, gran parte de su funcionamiento es idéntico, añadiendo una nueva fase de calibración en la que el usuario tendrá que mirar de forma natural a todas las zonas, siempre volviendo al punto central que se emplea como referencia.

Entonces, los pasos para que el sistema funcione son los siguientes:

- Se arranca el simulador, esperando a que Kinect detecte al conductor y empiece a calcular los valores del desplazamiento de la cabeza.
- Una vez que Kinect está en funcionamiento, primero tenemos que introducir en las opciones los valores de la distancia del usuario a la pantalla y del ancho de la misma.
- A continuación, se realiza la fase de calibración, en la que el usuario mirará primero al punto central, para estimar los valores iniciales del punto medio de la pantalla. Después irá recorriendo diferentes pantallas en las que se colocará mirando al centro y realizará un desplazamiento natural de la mirada hacia cada uno de los laterales de la pantalla, para calcular los escalados de los cuatro ejes.
- Se realiza la calibración del decisor, mirando a cada uno de los puntos de forma sucesiva. Después de mirar a cada punto, es necesario volver al punto central (18) para emplear la misma referencia en todos. Una vez se indique que ya se han calibración todos los puntos, se guardan los resultados.
- En este punto, el sistema ya estaría listo para comenzar las simulaciones, para lo cual se debe volver al menú inicial guardando los resultados, dando a comenzar y seleccionando el escenario.
- Por último, en las opciones previas del escenario, se deberá activar Kinect, que habrá almacenado todos los valores previamente calibrado.

Empleando los métodos explicados anteriormente, calculará la estimación de la mirada, introduciéndola en el decisor, y devolviendo por pantalla la zona estimada.

7.3.4. Manual de usuario

En este último apartado de este método, se va a realizar una detallada exposición visual, de todos los pasos que se deben realizar, con las pautas adecuadas, para poner el sistema en funcionamiento de forma correcta.

1. Arrancar el sistema, habiendo medido con un metro el ancho de la pantalla, y la distancia desde la punta de la nariz del usuario en posición de conducción al centro de la pantalla.
2. Una vez en el menú principal, se **debe esperar** a que la frase cambie de "Estado Kinect: OFF" a "Estado Kinect: ON". Una vez esto haya sucedido, se seleccionará en el menú mostrado en la Figura 7.17 la opción "Opciones".

Figura 7.17: Menú principal de simulador de conducción.

3. Aparecerá una pantalla como la que se puede ver en la Figura 7.18. Entonces, deberá introducir la distancia a la pantalla en centímetros en el recuadro debajo de "Distancia a la pared", y el ancho de la pantalla en el recuadro debajo de "Ancho de la pantalla". Entonces se pulsará sobre la opción "Calibrar Kinect".

Figura 7.18: Menú de opciones del simulador de conducción.

4. Se abrirá una ventana como la de la Figura 7.19. **Entonces el usuario debe colocarse en la misma posición que tendrá durante la conducción, y mirar de forma natural y fija al punto azul situado en el centro de la pantalla**. En ese momento, el asistente debe pulsar la tecla “K”, en el teclado del ordenador que está lanzando la simulación. Una vez hayan aparecido los valores correspondientes a la izquierda de los textos “Init_X:” e “Init_Y”, se pulsará el botón “Guardar” para pasar a la siguiente etapa.

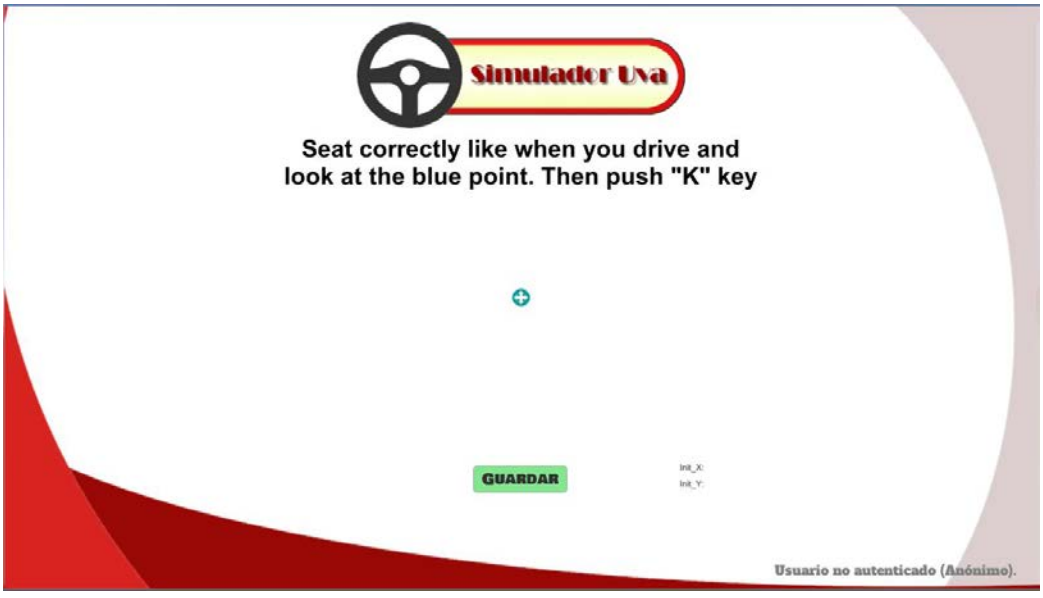

Figura 7.19: Pantalla del calibración para evaluar el punto central.

5. En la siguiente ventana, mostrada en la Figura 7.20, se realizará la calibración del eje X negativo. Para ello, se pedirá al usuario que en posición de conducción, se coloque mirando al punto central de color azul. En ese momento, **se solicitará al usuario que mire al punto izquierdo,**

de forma natural, y se mantenga fijo en esa posición. En ese momento, el supervisor deberá volver a pulsar la letra “K”. Entonces en la parte inferior se mostrará el valor del escalado ya calculado, el cual debe pertenecer al rango (1,10) aproximadamente. En caso de no capturarse, o tener valores extraños, se puede repetir el proceso en la misma pantalla todas las veces que sea necesario. Para ello solo es necesario volver a pulsar la letra “K”, y el sistema volverá a calcularlo y mostrarlo. Una vez se muestre un valor satisfactorio, se pulsará el botón guardar para pasar a la siguiente pantalla.

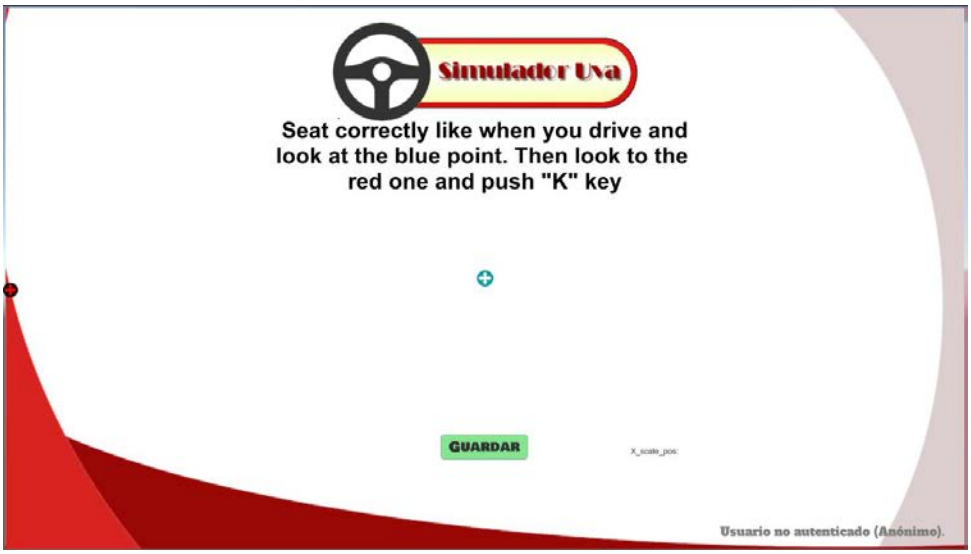

Figura 7.20: Pantalla de calibración del escalado del eje X negativo.

6. De forma similar en la nueva fase, se realizará el mismo proceso pero esta vez mirando al nuevo punto ubicado en la parte derecha de la pantalla, como se muestra en la Figura 7.21.

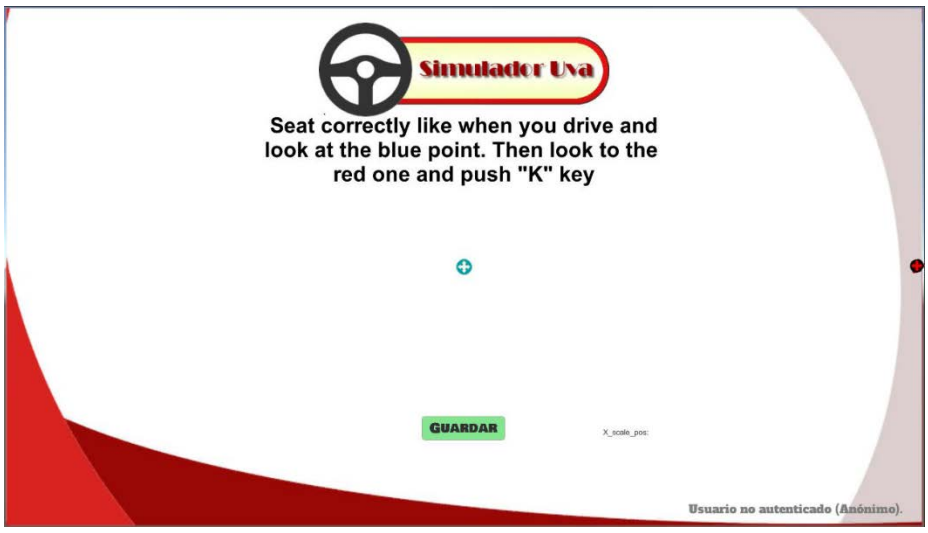

Figura 7.21: Pantalla de calibración del escalado del eje X positivo.

7. De la misma manera en la pantalla siguiente, mostrada en la Figura 7.22, se realizará el mismo procedimiento para calibrar el eje Y positivo.

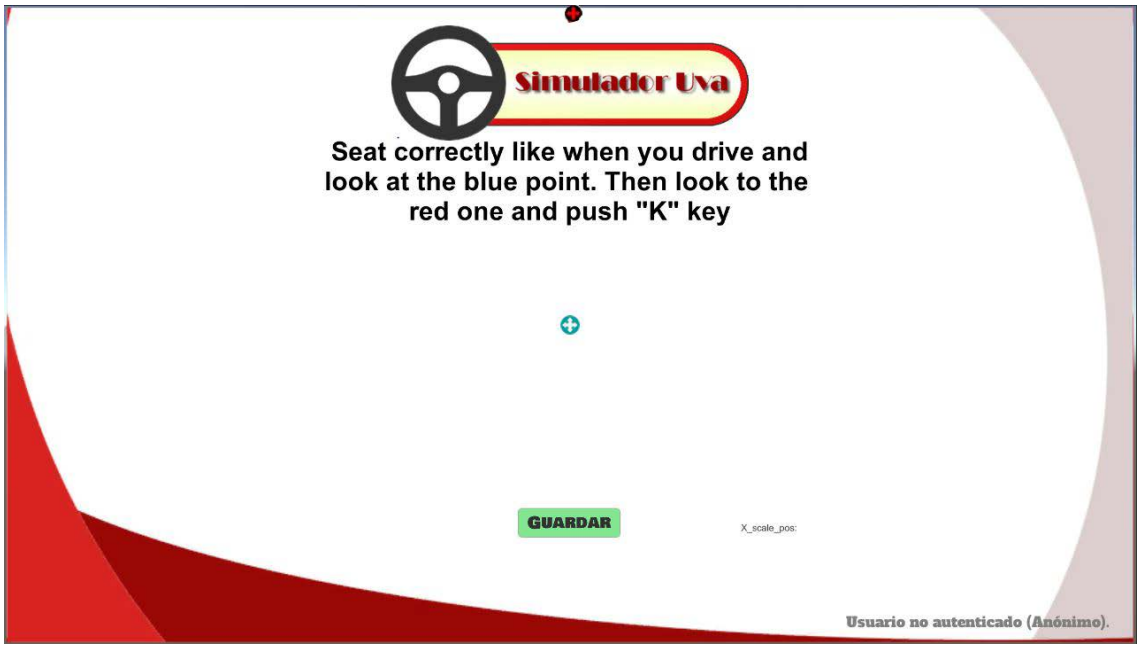

Figura 7.22: Pantalla del calibración del escalado del eje Y positivo.

8. A continuación, se realizará el mismo procedimiento para el eje Y negativo, sobre la siguiente ventana, mostrada en la Figura 7.23.

Figura 7.23: Pantalla del calibración del escalado del eje Y negativo.

9. Por último, se calibrará la nube de puntos, para recoger los valores que empleará el decisor. Para ello se mostrará por pantalla una ventana similar a la Figura 7.24. En ella, se pedirá que el usuario mire a todos los puntos de la siguiente forma. **Primero el usuario se colocará en el centro mirando al punto 18. Entonces, mirará de forma natural, acompañado con la cabeza, al punto 1. Cuando el usuario esté mirando deberá mantener esa postura, y el supervisor pulsar la letra "K", entonces se solicitará mirar al punto siguiente**, para lo cual se

deberá repetir todo el proceso. Una vez que se haya mirado a todos los puntos se informará por pantalla, y se saldrá dando al botón “Guardar”. En caso de error en algún punto, se podrá retroceder para volver a capturar dicho valor pulsando la tecla “L”. Los puntos se evaluarán después de cada captura, sabiendo que los de la zona izquierda son negativos y los de la derecha positivos para los números en el rango [1,12]; y negativos los de la zona inferior y positivos los de la zona superior en el rango [13,23]. El punto 18 debe tener valor cercano a 0, por considerarse el origen de coordenadas. Una vez que se haya capturado cada valor, se mostrará el mismo en la parte inferior de la pantalla, junto al nombre del punto para el cual se ha almacenado dicho valor. Para avanzar al finalizar debe pulsar el botón “Guardar”.

Figura 7.24: Pantalla para calibrar la nube de puntos.

10. En este punto se volverá al menú de opciones representado en la Figura 7.8. Entonces se pulsará el botón “Guardar”, y se volverá al menú principal, con Kinect ya preparado para comenzar las simulaciones.

11. Desde el menú principal, se podrán comenzar las simulaciones. Una vez se seleccione una, se mostrará inicialmente otra ventana de opciones como la de la Figura 7.25. En ella, **se debe pulsar el botón con el texto “Face tracking”**. Una vez se pulse el mismo, Kinect comenzará a realizar estimaciones de la zona a la que está mirando el conductor. Una vez el resto de opciones estén configuradas en función de las necesidades del usuario, se pulsará el botón comenzar, para iniciar la simulación.

Figura 7.25: Menú de opciones en el escenario antes de comenzar la simulación.

7.4. Método 3: Empleo de clasificadores

En este tercer método, vamos a emplear para la estimación de la mirada los clasificadores. Estos, como se explica en la sección 6, se emplean en muchos ámbitos con buenos resultados. Para ello, vamos a emplear el mismo método que está desarrollado en [24]. Aunque este se desarrolla pensando en un vehículo real, vamos a emplearlo adaptado al simulador, para poder evaluar sus resultados y compararlos con los métodos propios, sobre todo el número dos por ser una evolución del primero.

Además, se van a emplear para ello dos clasificadores diferentes, con la misma forma de entrenamiento y funcionamiento. Esto nos permitirá estudiar las diferencias entre ambos, y ver cuál se adapta mejor a nuestro sistema de forma práctica.

7.4.1. Preprocesamiento de selección de parámetros

Como parámetros de entrada al sistema, se deciden emplear los recogidos en [24]. Estos son:

1. Yaw: Rotación de izquierda a derecha de la cabeza.
2. Pitch: Rotación de arriba hacia abajo.
3. Punto central sobre el eje X: A partir del rectángulo de la cara recogido por Kinect, y tras realizar una media de todos los valores de este punto cuando se pide al usuario que mire al centro durante la fase de calibración, se calcula el punto central de la mirada en el sistema sobre el eje X.
4. Punto central sobre el eje Y: De forma similar al punto anterior, también obtenemos el punto central de la mirada sobre el eje Y.

5. Desplazamiento del rectángulo X: A partir del punto central, calculamos para cada patrón el desplazamiento del rectángulo de la cara sobre el punto central representado en la característica 3, sobre el eje X.
6. Desplazamiento del rectángulo Y: De la misma manera que en la característica anterior, se calcula el desplazamiento del rectángulo sobre el eje Y tomando como referencia el valor de la característica 4.
7. Tamaño del rectángulo: Se calcula el área del rectángulo de la cara y se introduce como última característica.

Una vez analizadas las 7 características que se plantean inicialmente, vamos a comprobar gracias a los métodos de análisis de características, la calidad de las mismas en función del peso. Para ello, vamos a utilizar un script en Matlab con un conjunto de 1100 patrones obtenido de una calibración de entrenamiento obtenido como referencia.

Para ello se va a emplear el método de Búsqueda por característica individual explicado en 6.2. Se emplea este método debido a que es el más sencillo, y no requiere mucho tiempo de procesamiento, dando unos resultados aproximados, pero suficiente para realizar un estudio inicial de cómo las características se adaptan o no de un sistema real a nuestro simulador. Una vez procesamiento en Matlab, se obtienen los resultados mostrados en la Figura 7.26.

```
>> [a b]=j2(CARAC,DIAG)

a =

     1     2     5     6     7     4     3


b =

   36.4484   68.3513   71.3605   72.8544   74.2352         0         0
```

Figura 7.26: Resultados de la evaluación de las 7 características.

Se puede observar de forma ordenada, que las características analizadas para nuestro sistema tienen diferentes pesos. Primero la más útil es el *yaw*, seguido del *pitch*. A continuación, con pesos muy similares, están el desplazamiento del rectángulo de la cabeza en el eje Y, y el desplazamiento del rectángulo de la cabeza en el eje X, seguido por el tamaño del rectángulo de la cabeza. Por último y con peso 0, el promedio del punto central del rectángulo de la cabeza mirando al centro de la pantalla tanto sobre el eje X como el eje Y.

Debido a que hay dos características con peso 0, lo cual implica que las mismas no aportan ninguna información al decisor, se deciden eliminar en nuestros patrones, aunque sigue siendo necesario su cálculo para servir de referencia al desplazamiento del rectángulo de la cabeza. Una vez descartadas, se vuelve a analizar el patrón, para estudiar cómo se han modificado ahora los pesos y comprobar que es válido para nuestro sistema. Los resultados es muestran en la Figura 7.27.

```
>> [a b]=j2(CARAC,DIAG)

a =

     1     2     3     4     5


b =

   36.4484   68.3513   71.3605   72.8544   74.2352
```

Figura 7.27: Resultado de la evaluación de 5 características.

Como se representa en la Figura 7.27, ahora ya todas las características dan información útil en nuestro simulador. Además, los valores de los pesos son los mismos debido a que el análisis de las características se realiza de forma individual, es decir, sin tener en cuenta el resto de características en el proceso en el método de evaluación de características escogido.

7.4.2. Generación del fichero de datos de entrenamiento

Una vez que conocemos como son nuestros patrones de entrenamiento, vamos a realizar el diseño del fichero de entrenamiento. Este debe realizarse previamente para cada usuario, en una fase de calibración controlada en la que se obtengan numerosos valores de cada zona de forma controlada, para que el clasificador pueda realizar el aprendizaje y generar un modelo para dicha simulación. Como se explica en la sección 6.4, se establece de forma general que el mínimo de patrones de entrenamiento necesarios para un sistema clasificador es 10 veces el número de sus características para cada zona. Debido a que son 5 características, deberían usarse al menos 50 patrones por zona. En el simulador se toma la decisión de emplear para ello tres veces más, capturando 150 patrones para cada zona.

Otra característica necesaria de obtener es el promedio del centro del rectángulo de la cabeza cuando se mira al centro de la pantalla, para que sirva como punto de referencia para representar los desplazamientos del mismo. Para ello, después de mirar a cada zona, se pedirá al usuario que vuelva a mirar al centro durante un tiempo, en el cual se capturaran los valores representativos del centro de rectángulo de la cabeza. Al finalizar el calibración se realiza el promedio obteniendo el punto central, recalculando los valores de desplazamiento del rectángulo de la cabeza almacenados durante la fase de captura de cada zona. Esta forma de obtener el punto central es necesaria, debido a que después de mirar a cada zona el volver a mirar al punto central de referencia puede ser diferente, y se quiere incluir cómo afecta esa información al sistema.

Una vez que se tienen en cuenta todos los parámetros necesarios para la obtención adecuada de las características del conjunto de patrones de entrenamiento, se diseña una pantalla de calibración como la que se muestra en la Figura 7.28.

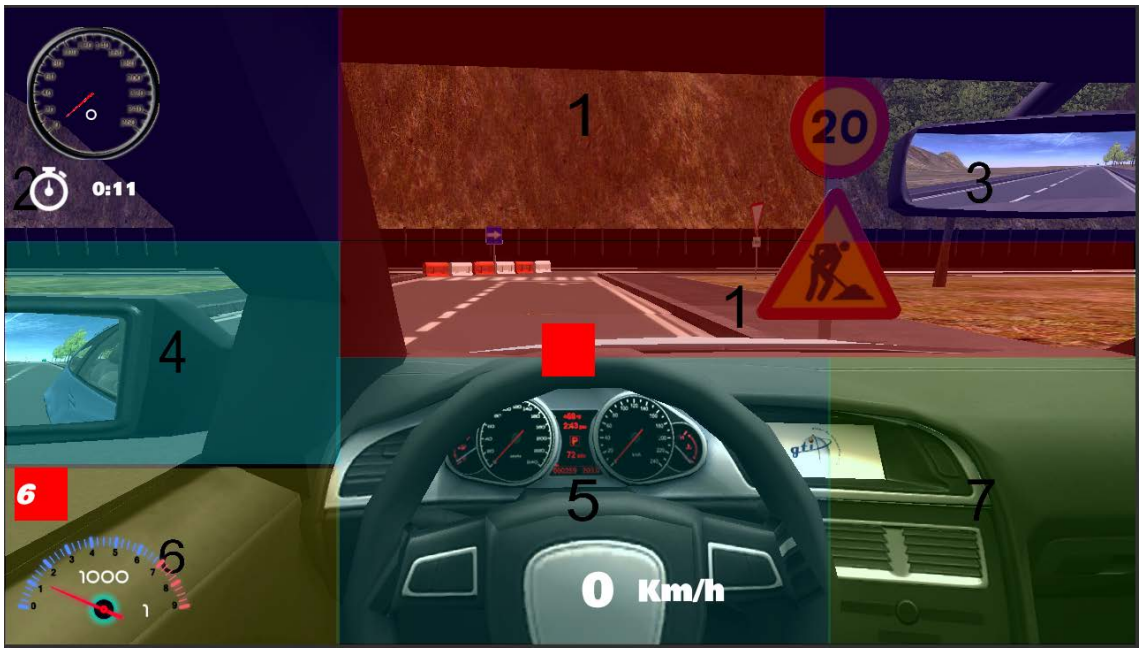

Figura 7.28: Pantalla para la generación de ficheros de entrenamiento.

La pantalla para realizar la calibración es igual a la vista que tendrá el usuario durante el simulador. En ella se muestra el punto central como el cuadrado rojo, que será el que informará, durante la evaluación explicada en la siguiente sección, sobre cuándo el usuario debe indicar el valor de la región a la que está mirando. Además se muestran las diferentes regiones con colores, para que el usuario pueda determinar fácilmente dónde está mirando en cada momento. La fase de generación del fichero de entrenamiento consta de los siguientes pasos:

1. Se mira a la región central durante 10 segundos de forma fija, en la que el sistema capturará 150 valores de las coordenadas del centro del rectángulo de la cabeza.
2. Se mira a la región 1 durante 10 segundos, recorriendo la zona central con la mirada de forma natural, pero sin acercarse a los bordes, para recoger 150 patrones que caractericen dicha región. En ella se recogerán los valores absolutos del centro del rectángulo de la cabeza, que será procesados después una vez se haya calculado el centro, para obtener los desplazamientos del mismo.
3. Después se vuelve a mirar al centro de forma estable, al igual que el punto 1, recogiendo otros 150 valores.
4. Después se mirará a la siguiente región, en este caso la dos, y se recogerán otros 150 valores de forma análoga al punto 2.
5. Este proceso se repetirá hasta que se finalice la captura de valores del método 7, después de la cual se finalizará la generación del fichero de entrenamiento.
6. Después se obtendrá el punto medio del centro del rectángulo de la cabeza, haciendo la media de todos los valores capturados en las coordenadas X e Y cuando se estaba mirando al centro. Una vez conocido dicho punto, se

transformarán los valores de los centros del rectángulo de la cabeza de cada patrón capturado cuando se miraba a cada región, a través de las fórmulas:

$$despl\_x\_bb = centro\_x\_bb - media\_centro\_x$$
$$despl\_y\_bb = centro\_y\_bb - media\_centro\_y$$

Con ello se obtendrán los desplazamientos del centro del rectángulo de la cabeza para cada patrón de cada zona, referenciados al punto central para ese usuario.

7. Una vez todos los datos listos, se escribirá el fichero con todos los datos de entrenamiento, preparado para entrenar el clasificador. Este fichero tiene la forma mostrada en la Figura 7.29.

```
1 1:1.25 2:16.2299995422363 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:5.56411483253589 7:4800
1 1:1.5 2:16.4400005340576 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:5.56411483253589 7:4950
1 1:1.35000002384186 2:15.6700000762939 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:6.06411483253589 7:4884
1 1:1.35000002384186 2:16.7399997711182 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:5.56411483253589 7:4950
1 1:1.89999997615814 2:16.8600006103516 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:5.56411483253589 7:4800
1 1:2.02999997138977 2:17.0699996948242 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:5.56411483253589 7:4800
1 1:1.6599999666214 2:16.5100002288818 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:5.56411483253589 7:4875
1 1:0.0700000002980232 2:16.4099998474121 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:5.56411483253589 7:4875
1 1:1.05999994277954 2:16.1900005340576 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:5.56411483253589 7:4875
1 1:0.790000021457672 2:15.710000038147 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:5.56411483253589 7:4800
1 1:1.26999998092651 2:15.2799997329712 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:5.56411483253589 7:4875
1 1:1.25999999046326 2:15.7799997329712 3:1012.71961722488 4:720.935885167464 5:-2.21961722488038 6:5.06411483253589 7:4940
1 1:0.28999999165535 2:15.0799999237061 3:1012.71961722488 4:720.935885167464 5:-2.21961722488038 6:4.56411483253589 7:4875
1 1:0.579999983310699 2:14.8000001907349 3:1012.71961722488 4:720.935885167464 5:-2.21961722488038 6:4.56411483253589 7:4875
1 1:-1.4099999666214 2:14.3199996948242 3:1012.71961722488 4:720.935885167464 5:-2.71961722488038 6:4.06411483253589 7:4864
1 1:-1.12000000476837 2:13.8800001144409 3:1012.71961722488 4:720.935885167464 5:-3.21961722488038 6:4.06411483253589 7:4810
1 1:-1.00999999046326 2:14.0100002288818 3:1012.71961722488 4:720.935885167464 5:-2.21961722488038 6:4.06411483253589 7:4940
1 1:-0.870000004768372 2:14.3599996566772 3:1012.71961722488 4:720.935885167464 5:-2.21961722488038 6:4.06411483253589 7:4940
1 1:-0.00999999977648258 2:13.1700000762939 3:1012.71961722488 4:720.935885167464 5:-2.21961722488038 6:4.06411483253589 7:4940
1 1:0.0299999993294477 2:14.7799997329712 3:1012.71961722488 4:720.935885167464 5:-1.71961722488038 6:4.06411483253589 7:4864
1 1:1.01999998092651 2:13.9799995422363 3:1012.71961722488 4:720.935885167464 5:-1.71961722488038 6:4.06411483253589 7:4864
1 1:1 2:13.9200000762939 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:4.06411483253589 7:4940
1 1:1.20000004768372 2:14.3599996566772 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:4.56411483253589 7:4950
1 1:1.61000001430511 2:15.710000038147 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:4.56411483253589 7:4950
1 1:3.41000008583069 2:15.5600004196167 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:5.06411483253589 7:4940
1 1:1.91999995708466 2:15.5200004577637 3:1012.71961722488 4:720.935885167464 5:-0.21961722488038 6:5.56411483253589 7:4875
1 1:1.35000002384186 2:15.8900003433228 3:1012.71961722488 4:720.935885167464 5:-0.21961722488038 6:6.06411483253589 7:4940
1 1:3.09999990463257 2:14.4799995422363 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:6.06411483253589 7:5016
1 1:1.92999994754791 2:14.9099998474121 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:6.06411483253589 7:4864
1 1:1.37000000476837 2:14.6599998474121 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:6.06411483253589 7:4940
1 1:1.46000003814697 2:14.5200004577637 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:6.06411483253589 7:4940
1 1:0.560000002384186 2:17.7299995422363 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:7.06411483253589 7:5092
1 1:-0.0799999982118607 2:18.7600002288818 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:7.56411483253589 7:5025
1 1:-0.409999996423721 2:17.9799995422363 3:1012.71961722488 4:720.935885167464 5:-1.71961722488038 6:7.06411483253589 7:5168
1 1:-0.5 2:16.9099998474121 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:7.06411483253589 7:4958
1 1:0.5 2:17.4099998474121 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:6.56411483253589 7:5025
1 1:0.550000011920929 2:17.1399993896484 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:7.06411483253589 7:4810
1 1:-0.0799999982118607 2:16.5799999237061 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:7.06411483253589 7:4810
1 1:0.259999990463257 2:16.4699993133545 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:7.06411483253589 7:4810
1 1:0.409999996423721 2:16.6599998474121 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:7.06411483253589 7:4810
1 1:0.600000023841858 2:16.9400005340576 3:1012.71961722488 4:720.935885167464 5:-1.71961722488038 6:7.56411483253589 7:4818
1 1:-0.680000007152557 2:16.0100002288818 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:7.06411483253589 7:4810
1 1:-0.259999990463257 2:18.5499992370605 3:1012.71961722488 4:720.935885167464 5:-1.71961722488038 6:7.56411483253589 7:4818
1 1:1.30999994277954 2:19.5699996948242 3:1012.71961722488 4:720.935885167464 5:-1.21961722488038 6:8.56411483253589 7:4891
1 1:2.70000004768372 2:19.1399993896484 3:1012.71961722488 4:720.935885167464 5:-0.71961722488038 6:8.56411483253589 7:4818
1 1:1.97000002861023 2:17.2800006866455 3:1012.71961722488 4:720.935885167464 5:-0.21961722488038 6:9.06411483253589 7:4810
1 1:2.77999997138977 2:19.1499996185303 3:1012.71961722488 4:720.935885167464 5:0.28038277511962 6:9.06411483253589 7:4884
1 1:3.86999988555908 2:19.4200000762939 3:1012.71961722488 4:720.935885167464 5:0.28038277511962 6:8.56411483253589 7:4818
1 1:3.67000007629395 2:18.7000007629395 3:1012.71961722488 4:720.935885167464 5:-0.21961722488038 6:8.06411483253589 7:4810
1 1:2.84999990463257 2:17.8199996948242 3:1012.71961722488 4:720.935885167464 5:-0.21961722488038 6:8.06411483253589 7:4810
1 1:3.47000002861023 2:17.2199993133545 3:1012.71961722488 4:720.935885167464 5:-0.21961722488038 6:7.06411483253589 7:4810
1 1:1.54999995231628 2:15.6400003433228 3:1012.71961722488 4:720.935885167464 5:0.28038277511962 6:6.56411483253589 7:4950
1 1:2.57999992370605 2:17.0499992370605 3:1012.71961722488 4:720.935885167464 5:0.78038277511962 6:7.06411483253589 7:4810
1 1:3.69000005722046 2:16.3299999237061 3:1012.71961722488 4:720.935885167464 5:0.78038277511962 6:6.56411483253589 7:4875
1 1:2.75999999046326 2:15.8199996948242 3:1012.71961722488 4:720.935885167464 5:1.28038277511962 6:6.06411483253589 7:5016
1 1:1.86000001430511 2:15.460000038147 3:1012.71961722488 4:720.935885167464 5:1.28038277511962 6:5.56411483253589 7:4950
1 1:0.889999985694885 2:15.7700004577637 3:1012.71961722488 4:720.935885167464 5:0.78038277511962 6:5.06411483253589 7:4940
```

Figura 7.29: Modelo del fichero de entrenamiento del clasificador.

Es importante destacar que este valor de referencia del rectángulo de la cara se almacenará, para que después al ir generando durante la simulación los patrones se utilicen para calcular los desplazamientos del mismo e introducirlo en los patrones correspondientes. En la práctica los patrones se generarán de forma análoga a los de entrenamiento, de forma que se pasen al clasificador para que este nos determine las zonas a las que mira el usuario.

7.4.3. Diseño y características del clasificador MLP

Una vez que tenemos generado el fichero de entrenamiento, y cómo se generarán los patrones durante la simulación, vamos a configurar y preparar nuestro clasificador MLP, explicando concretamente cómo está configurado y qué proceso sigue desde su arranque hasta que nos devuelve valores estimados de las zonas a las que mira el conductor.

La implementación realiza de MLP en nuestro sistema se lleva a cabo combinando 7 MLPs binarios. Cada uno se encargará de detectar una zona como válida (valor 1) y descartar el resto (valor 0). Una vez que cada patrón sea procesado por las 7 redes, se evaluará su salida de forma conjunta, para decidir la zona a la que se está mirando. Para ello se debe conocer que la salida es un valor normalizado entre 0 y 1, siendo más próxima a 1 cuanto más próximo es ese patrón al conjunto entrenado como válido y 0 cuanto más alejado. Así, se combinarán los 7 clasificadores binarios, determinando que la zona detectada coincide con el clasificador que dé mayor valor de salida. Por ejemplo, si el clasificador 4 da un valor cercano a 1 mientras que el resto dan valores en torno al 0, se comprobará que el MLP con la salida máxima es el 4 y, por tanto, la zona a la que se está mirando es la número 4.

En cuanto a configuración, las 7 redes tienen los mismos parámetros de configuración y entrenamiento, además de tener la una estructura idéntica. La única modificación es el conjunto de patrones que debe tener salida 1 para cada uno. Esto debe realizarse de esta manera para que las salidas de todas puedan ser comparables. Conocido esto, vamos a explicar el diseño de una red, sabiendo que el resto son exactamente iguales.

Para comenzar, vamos a establecer los parámetros de entrenamiento que tendrá la red. Estos, relacionándolos con la teoría expuesta en el punto 6.5, son los siguientes:

- param.term_crit: Indica el límite de iteraciones a partir de las cuales el sistema se considera que está entrenado. Para ello se emplean el número de iteraciones y el valor de convergencia de la red, considerando finalizado el entrenamiento cuando cualquiera de los dos sea válido. En nuestro caso, el número de iteraciones máximas es 1000 y el parámetro épsilon para la convergencia es 0.001. Con menos iteraciones, los resultados de prueba con ficheros de entrenamiento similares a los del sistema real en funcionamiento son peores, siendo, para valores de iteraciones más altos, iguales. Además, para dicho valor de épsilon el acierto del sistema es el máximo obtenido entre los valores evaluados. Es destacable que, como se estudia en el capítulo anterior, este tipo de clasificadores es sensible al sobreentrenamiento, y poner más iteraciones excedería los límites recomendados de forma general.
- param.train_method: Indica el modo, entre los soportados, de entrenamiento. En este caso, se elige el más utilizado que es “back propagation” (propagación desde las neuronas de salida hacia las de entrada). Además, este método requiere dos parámetros adicionales:

- param.bp_dw_scale: Representa la fuerza de cada gradiente en el cálculo del peso. Se emplea el valor recomendado 0,1.
- param.bp_moment_scale: Representa el valor de la fuerza de la diferencia entre los pesos de dos iteraciones sucesivas. Se emplea el valor recomendado 0,1.

Una vez configurados los parámetros de entrenamiento, vamos a diseñar las redes que formarán nuestro clasificador. Será una red [5,14,1], con 5 neuronas de entrada para las 5 características de nuestro simulador, 14 neuronas en la capa intermedia, y una neurona de salida ya que es un decisor binario.

Las 14 neuronas de la capa intermedia se han decidido debido a que al existir 7 zonas que detectar, la diversidad de los datos de entrenamiento se agrupará alrededor de 7 conjuntos de patrones. Entonces se duplica ese valor para que nuestra red pueda distinguir pesos diferencias suficientes para interpretar todas las zonas, y asignar un valor correcto en la salida. Además, se realizaron pruebas con otros valores y fueron las que mejores resultados obtuvieron. Así las 7 redes que forman el clasificador quedarían con la forma mostrada en la Figura 7.30.

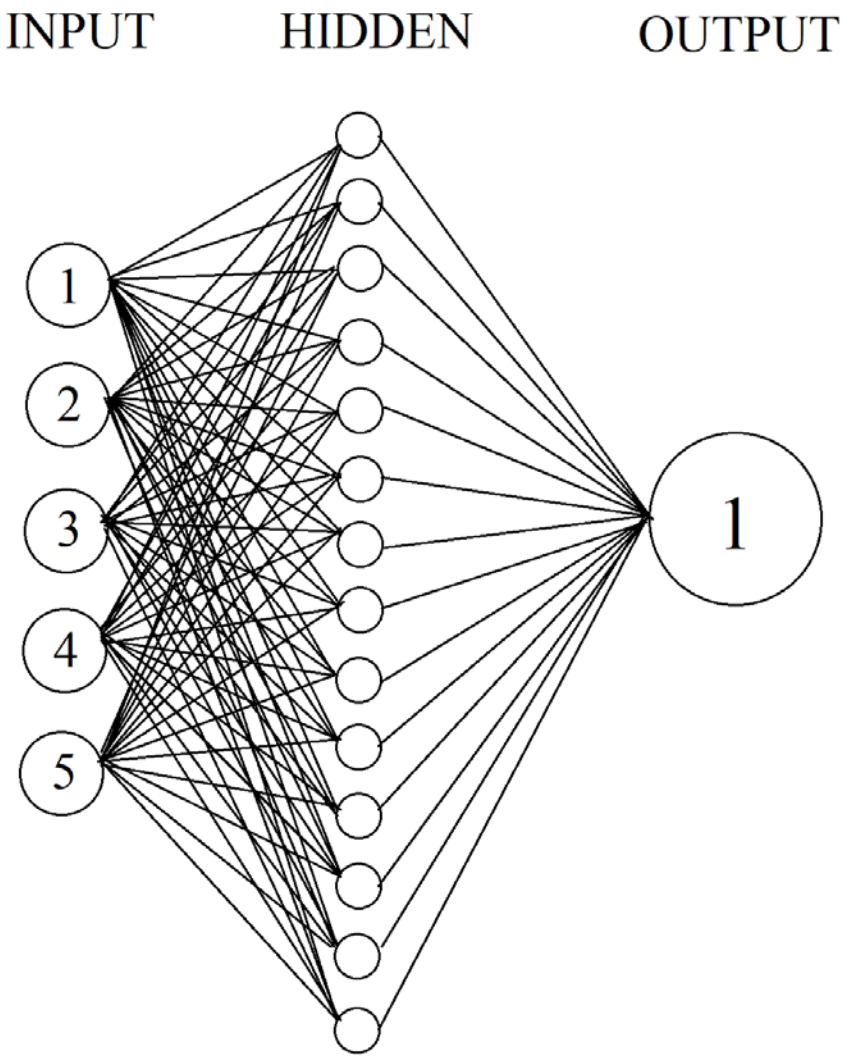

Figura 7.30: Estructura de las redes neuronales que forman el clasificador.

Por último, la función de activación que se empleará es la Sigmoide Simétrica, explicada en el capítulo anterior 6.5, y que es la recomendada.

Una vez configuradas todas las características de las redes, simplemente se lee el fichero que contiene todos los patrones de entrenamiento y se rellenan dos matrices: una que contiene un patrón por fila y todas las características en columnas, y otra que contiene el resultado esperado para cada patrón en el mismo índice que el índice de fila de la matriz.

A continuación, se crean las redes con la configuración expuesta anteriormente, y se realiza el entrenamiento empleando los parámetros configurados y las matrices que se acaban de crear. Una vez finaliza el entrenamiento, se guarda el modelo para que pueda ser utilizado para recrear dicha red en el momento de la simulación. Es importante destacar que estas redes implementadas normalizan de forma automática la entrada y la salida, por lo que no es necesario que se realice de forma manual previamente.

Por último, se arrancarían todas las redes con la configuración establecida previamente durante el aprendizaje, y se enviarían los patrones según se generen durante la simulación, recogiendo los valores de salida y combinándolos, evaluando el mayor de ellos para detectar la zona exacta. Además se guardaría en un vector, para escribirlos en un fichero de resultados el finalizar dicha simulación.

7.4.4. Diseño y características del clasificador SVM

De forma análoga al clasificador anterior, los pasos para su creación y configuración son similares. Sin embargo, en este caso, el sistema va a ser un solo clasificador vectorial, que devuelve directamente como valor entero la zona a la que está mirando al usuario. De esta forma, la configuración es mucho más sencilla y el entrenamiento es más rápido.

En primer lugar, vamos a establecer los parámetros de entrenamiento, relacionados con la teoría expuesta en el capítulo 6.5:

- param.term_crit.type: Similar a la red neuronal, vamos a limitar el entrenamiento tanto a iteraciones como al valor épsilon, empleado para evaluar la convergencia del clasificador. Debido que este clasificador no es sensible al sobreentrenamiento, como se muestra en [20], se decide poner el número de iteraciones a 100.000. Además, tras la realización de pruebas con diferentes valores, de forma general es un valor que funciona muy bien, y nos permite alcanzar de forma segura la convergencia. Por otro lado, el valor de convergencia épsilon será el mismo que en el caso anterior, 0,001.

A continuación, vamos a configurar los parámetros del clasificador, los cuales son:

- param.svm_type: Este parámetro controla el tipo de formulación que empleará el clasificador vectorial, de las diferentes opciones comentadas

en el capítulo 6.5. De ellas vamos a escoger C_SVC, que es el clasificador vectorial genérico más estándar.

- param.C: El parámetro que controla la separación de los outliers para este clasificador. Para él se emplea el valor por defecto 1.
- param.kernel_type: El tipo de núcleo que se utilizará en el clasificador. Entre las opciones disponibles, se escoge LINEAR, que es el núcleo lineal. Esto se debe a que es el más sencillo y rápido, característica necesaria para nuestro simulador ya que nos interesa un bajo tiempo de respuesta.

Una vez todo el sistema contiene la configuración adecuada, se crea el clasificador, empleando los parámetros expuestos. El siguiente paso es análogo al clasificador anterior, y consistirá en leer el fichero de entrenamiento y crear las matrices tanto de patrones de entrenamiento, como el vector de resultados esperados para los patrones de entrenamiento. En el siguiente paso, se indicará al clasificador que entrene con la información contenida en las matrices, y empleando los parámetros explicados en la hoja anterior. Además, se guardará el modelo generado, para que pueda cargar los resultados del entrenamiento directamente antes de arrancar el escenario de simulación donde se generarán los patrones de evaluación.

Una vez con el escenario cargado, el clasificador ya entrenado comenzará a recibir patrones de entrada, los cuales se pasarán a una matriz para que sean evaluados por el clasificador, dando como salida un entero con la zona estimada. Además, esta zona se recogerá en un vector, que se escribirá en un fichero de resultados al finalizar la simulación.

7.4.5. Funcionamiento general

En este punto, ya tendríamos todo nuestro sistema listo para funcionar empleando uno de los dos clasificadores. Debido a que los ficheros de entrenamiento son idénticos para ambos, al igual que la forma de generar los patrones de evaluación, el sistema se comporta de la misma forma empleemos uno u otro. Por ello, tanto este capítulo como el siguiente es el mismo para los dos, comportándose los clasificadores como si se trataran de componentes software independientes.

Primero al arrancar el sistema, debemos ir a las opciones y generar el fichero de entrenamiento para ese usuario y esa simulación. Esto se debe a que la posición del usuario puede ser diferente incluso siendo el mismo que en otras simulaciones, y que diferentes usuarios miran diferente.

Durante la calibración, se pedirá al usuario que mire 10 segundos a la zona central de forma fija y después 10 segundos a la zona central de la región 1, realizando pequeños movimientos. Esto se repetirá sucesivamente para todas las zonas, finalizando la calibración cuando el usuario termine de mirar a la zona 7. Una vez realizado la calibración, se pulsará dos veces más al botón de calibrar para que se realice el almacenamiento del fichero de entrenamiento.

Una vez guardado, se saldrá al menú principal y se iniciará la simulación, donde será necesario pulsar la opción para activar el Kinect y que se empiecen a capturar patrones.

Por último, al pulsar el botón que finaliza la simulación o al acabar esta de forma automática, se guardarán en un fichero los datos de las regiones recogidas, para conocer las zonas donde ha mirado el usuario.

7.4.6. Manual de usuario

En este último capítulo, se va a realizar una detallada exposición visual de todos los pasos que se deben realizar, con las pautas adecuadas, para poner el sistema en funcionamiento de forma correcta.

1. Arrancar el sistema, y esperar a que se muestre el menú principal. Una vez en el menú principal, se **debe esperar** a que la frase cambie de "Estado Kinect: OFF" a "Estado Kinect: ON". Una vez esto haya sucedido, se seleccionará en el menú mostrado en la Figura 7.31 la opción "Opciones".

Figura 7.31: Menú principal del simulador de conducción.

2. Aparecerá una pantalla como la que se puede ver en la Figura 7.32. Entonces, deberá pulsar la opción "Calibrar Kinect".

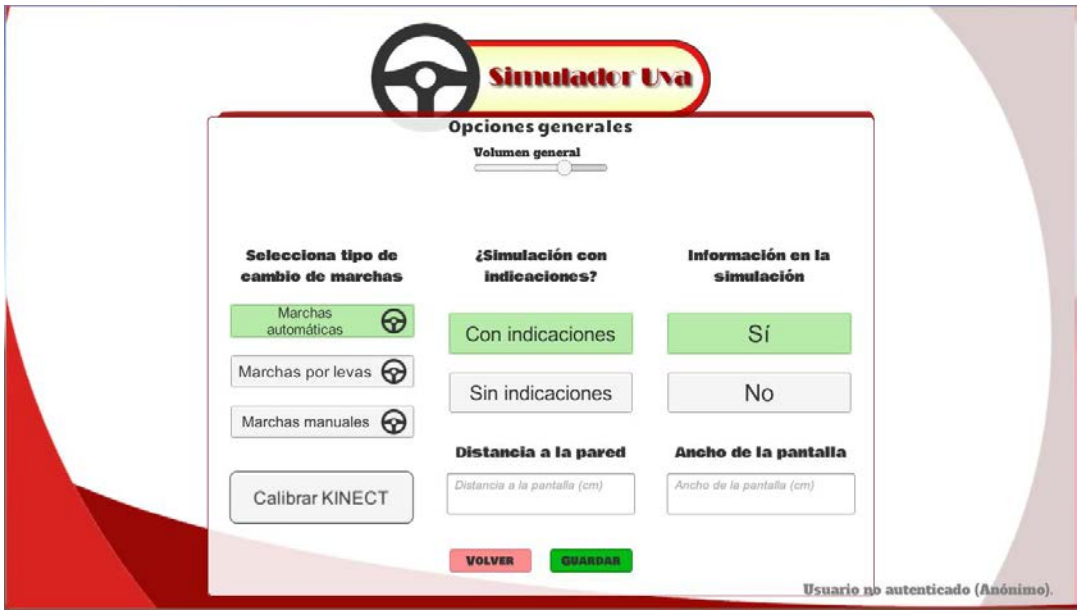

Figura 7.32: Menú de opciones del simulador de conducción.

3. Se abrirá una ventana como la de la Figura 7.33. **Entonces el usuario se situará en posición de conducción, mirando al cuadrado rojo central de forma fija. Entonces el supervisor pulsará la tecla "K"**, con lo que el sistema comenzará a capturar valores. Debe **mantenerse fijo** en esa posición durante 10 segundos, hasta que el texto superior cambie indicando que se ha finalizado de calibrar el centro e indicando la siguiente zona a la que debe mirar.

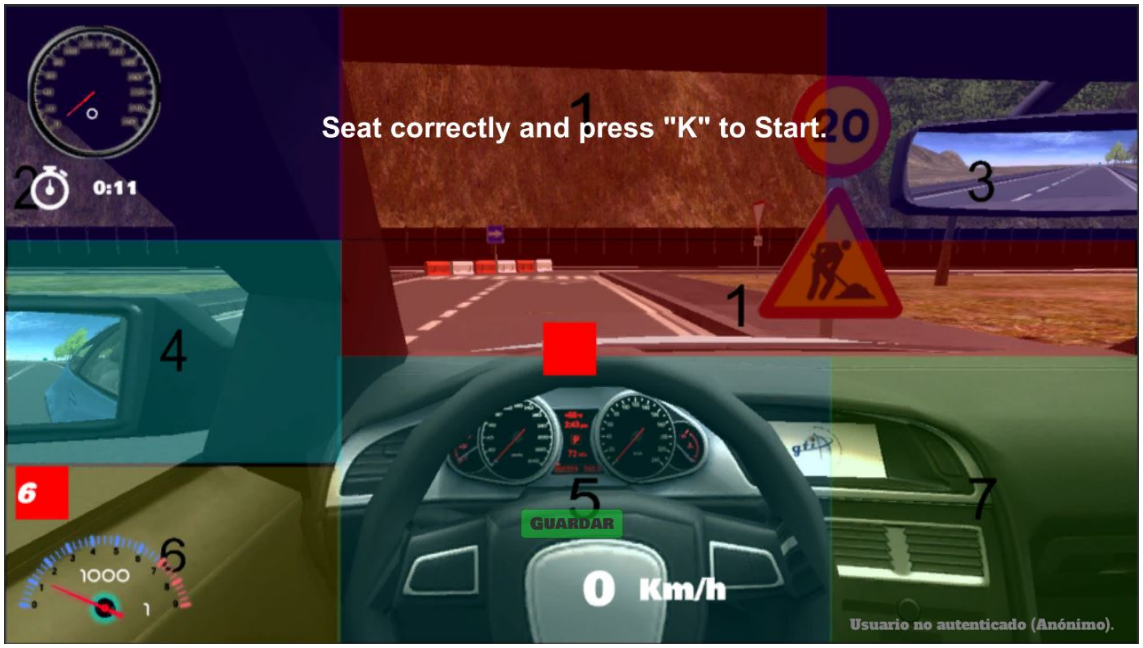

Figura 7.33: Pantalla de calibración del método 3.

4. El usuario mirará al centro de la zona indicada, **por la zona central de dicha región con pequeños movimientos**. En ese momento el supervisor pulsará la tecla "K", lo que iniciará la captura de patrones. El usuario **debe continuar de esa forma durante 10 segundos**, hasta que el texto superior

cambie indicando que la captura de patrones ha finalizado. Entonces, se mostrará un texto indicando que vuelva a mirar al punto central, como se indica en el capítulo anterior.

5. Este ciclo se repetirá tantas veces como zonas haya en la pantalla, hasta que el usuario haya mirado al centro de la última zona, en este caso la número 7. Entonces, el supervisor **deberá pulsar la tecla "K", con lo que se indicará que la captura y procesamiento de los datos ha finalizado. Entonces deberá pulsar otra vez la tecla "K", para que se realice el almacenamiento de los datos de entrenamiento en el fichero correspondiente**. Una vez todo se haya realizado correctamente, se mostrará un texto indicando que el entrenamiento ha finalizado. Entonces, se pulsará el botón "Guardar" para volver al menú de opciones (Figura 7.32), y después al botón "Volver" para salir al menú principal (Figura 7.32).

6. Desde el menú principal, se podrán comenzar las simulaciones. Una vez se seleccione una, se mostrará inicialmente otra ventana de opciones como la de la Figura 7.34. En ella, se **debe pulsar el botón con el texto "Face tracking"**. Una vez se pulse el mismo, Kinect comenzará a realizar capturas de los patrones generados a partir del conductor. Una vez el resto de opciones estén configuradas en función de las necesidades del usuario, se pulsará el botón "Comenzar", para iniciar la simulación.

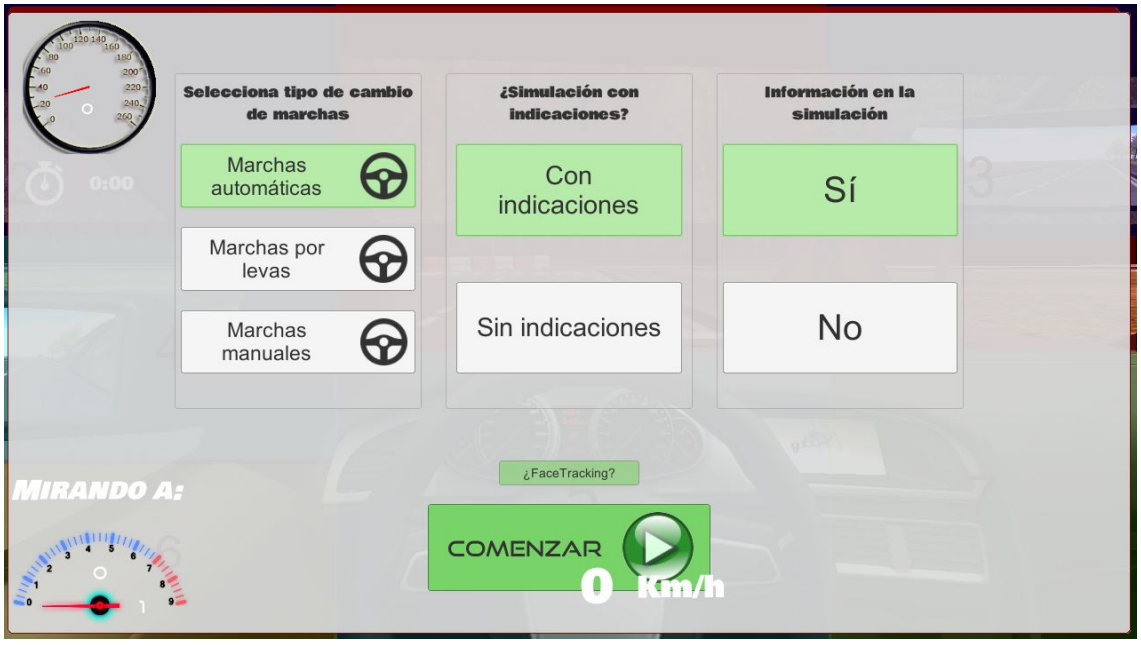

Figura 7.34 Menú de opciones en el escenario antes de comenzar la simulación.

8. Presupuesto económico

Para la realización de este Trabajo de Fin de Grado, se ha empleado un dispositivo Kinect y el periférico Logitech G27 Racing Wheel, además de los dos ordenadores utilizados. Para el desarrollo del proyecto ha sido también necesario el software del motor gráfico Unity, en su versión Personal, el cual no tiene coste debido a que la solución software no se emplea de forma comercial. Teniendo en cuenta los precios, el gasto en recursos materiales es de:

• Logitech G27 Racing Wheel:	359,00€
• Kinect:	99,98€
• Ordenador portátil MSI GE70 2PE:	1.100,99€
• Ordenador de sobremesa montado por componentes:	800,00€
• Proyector:	100,00€
Total:	2459,97€

En cuanto a las horas dedicadas en el desarrollo del trabajo, se calculan unas 450 horas. Teniendo en cuenta que el sueldo de un desarrollador de videojuegos puede ser sobre 25€/hora, se calcularían en torno a 11.250€.

El gasto total ascendería a 2459,97 + 11.250 = 13.709,97 €.

9. Conclusiones y líneas futuras

En este capítulo, se expondrán las conclusiones obtenidas con la realización de este Trabajo de Fin de Grado, además de algunas líneas futuras de interés con las que se podría continuar trabajando en el proyecto.

9.1. Conclusiones

Tras finalizar este proyecto, se han logrado desarrollar diferentes métodos funcionales para una estimación relativamente precisa de la dirección de la mirada a través del movimiento de la cabeza. Además, se ha logrado una integración plena de los mismos con el simulador de conducción. Esta información es posible almacenarla, para que sea tratada como retroalimentación para los usuarios, corrigiendo ciertos fallos, que no sería posible detectar en un coche real.

Estas pruebas se pueden realizar todas las veces que sean necesarias por los mismos escenarios, permitiendo a los usuarios comprobar su progreso. Gracias esto, se les puede demostrar la importancia de una conducción adecuada y segura, como no sería posible en una situación real. Este proceso controlado de aprendizaje es muy útil tanto para los usuarios como para los educadores, ya que permite conocer técnicas de enseñanza de seguridad vial más fáciles y eficientes, además de permitir a cada usuario un aprendizaje a medida sin que esto requiera ningún tipo de exposición a los riesgos existentes en una vía de circulación real.

Por otro lado, se han conseguido evaluar diferentes métodos para realizar la estimación de la dirección de la mirada en el simulador, estudiando cuáles se adaptan mejor al mismo. Esto nos permite implementar el que mejor se ajuste a las necesidades del mismo, mejorando la información obtenida de los usuarios y, por tanto, la realimentación que se les aporta de cara a mejorar su conducción.

En el ámbito del uso de clasificadores capaces de manejar patrones de forma general, se ha comprobado que se adaptan al simulador de forma correcta, y que son una solución válida a tener en cuenta. Sin embargo, también se ha podido comprobar que, si bien estos se comportan de forma notable en gran variedad de aplicaciones y ámbitos diferentes, el diseño de una solución concreta pensada únicamente para este escenario, es capaz de dar mejores resultados de forma más eficiente.

Mediante las simulaciones con diferentes sujetos de prueba, se han podido comprobar las diferentes soluciones no solo en función de su fiabilidad, sino también de su facilidad de uso. Esto es muy relevante debido a que estos sistemas de estimación requieren una fase de aprendizaje para adaptarse a cada usuario, que si resulta demasiado larga o compleja puede no llevarse a cabo de forma correcta. Esto provocaría que el sistema no se ajustara de forma correcta a dicho usuario, provocando un funcionamiento erróneo del mismo y devolviendo una realimentación inútil.

9.2. Líneas futuras

Como líneas futuras del trabajo, una de las mejoras del simulador podía ser gráfica, desarrollando una creación de los elementos del escenario más realista. Esta mejora podría afectar tanto al entorno exterior, como al resto de vehículos y a la interfaz de usuario. Por ejemplo, situando la información de la velocidad en el velocímetro del vehículo.

En cuanto al resto de vehículos, se podría mejorar la inteligencia artificial de los mismos, haciendo que se comportaran de forma mucho más realista y autónoma, no con un circuito predeterminado que provoca que ciertas fases de la simulación resulten repetitivas cuando se realizan sucesivas veces.

En cuanto al hardware empleado para las simulaciones, sería interesante desarrollar un entorno fijo, más similar al interior de un coche, que tuviera los elementos con una disposición similar, lo que aumentaría el realismo y la calidad del aprendizaje.

Otra opción interesante, conociendo la forma de mirar del usuario, sería desarrollar métodos para detectar estados inseguros para la conducción, como puede ser somnolencia o embriaguez.

Para el método 2, se podría diseñar una calibración más natural, de forma que el usuario no fuera tan consciente de que se está evaluando su forma de mirar. Esto permitiría que el usuario se comportara de forma más realista durante esta fase, mejorando la detección durante la simulación.

Establecer algún método fiable que permita detectar, sin la necesidad de ver por pantalla las zonas detectadas por Kinect, que el sistema no ha sido calibración de forma correcta y que no está tomando los datos de forma adecuada. Por ejemplo, detectando comportamientos a priori anómalos, como mirar constantemente en una zona que esté fuera de la pantalla.

Referencias

1] D.G.T., «Anuario estadístico de accidentes 2015,» 2016. [En línea]. Available: http://www.dgt.es/Galerias/seguridad-vial/estadisticas-e-indicadores/publicaciones/anuario-estadistico-de-accidentes/anuario-accidentes-2015.pdf. [Último acceso: Enero 2017].

2] D.G.T., «Políticas viales,» 2017. [En línea]. Available: http://www.dgt.es/es/seguridad-vial/politicas-viales/. [Último acceso: Febrero 2017].

3] D.G.T., «Conducción eficiente,» 2016. [En línea]. Available: http://www.dgt.es/PEVI/documentos/catalogo_recursos/didacticos/did_adultas/Conduccion_eficiente.pdf. [Último acceso: Febrero 2017].

4] D.G.T., «Cuestiones de seguridad vial, conducción eficiente, medio ambiente y contaminación,» 2016. [En línea]. Available: http://www.dgt.es/Galerias/seguridad-vial/formacion-vial/cursos-para-profesores-y-directores-de-autoescuelas/XIX-curso-de-profesores/Seguridad-Vial.pdf. [Último acceso: Febrero 2017].

5] Comunidad de Madrid, «Anuario estadístico 2016,» 2017. [En línea]. Available: http://www.madrid.es/UnidadesDescentralizadas/UDCEstadistica/Nuevaweb/Publicaciones/anuesta/nuevos/Anuario%20Estad%C3%ADstico%20Municipal/Anuario%20estad%C3%ADstico%202016/Cap%C3%ADtulo%201%20-%20Territorio%20y%20Medio%20Ambiente.pdf. [Último acceso: Febrero 2017].

6] Grupo OTP, «Simulador de conducción,» 2017. [En línea]. Available: http://www.grupotp.org/simulador-de-conduccion/. [Último acceso: Marzo 2017].

7] COTPRLCV, «Seguridad vial, simuladores para aprender jugando,» 2017. [En línea]. Available: www.cotprlcv.es/seguridad-vial-simuladores-para-aprender-jugando/. [Último acceso: Marzo 2017].

8] Forward Development, «City Car Driving,» 2017. [En línea]. Available: http://citycardriving.com/products/citycardriving. [Último acceso: Marzo 2017].

9] Drivesim Simulation S.L., «Drivesim,» 2017. [En línea]. Available: http://drivesimsimulator.com/. [Último acceso: Marzo 2017].

10] Simumak, «Simescar,» 2017. [En línea]. Available: http://simumak.com/es/simescar. [Último acceso: Marzo 2017].

11] C. González Muñoz, L. Sanagustín Grasa, D. Romero San Martín y M. Gracia Bandrés, «Análisis: Motores gráficos y su aplicación en la industria,» 29 Diciembre 2014. [En línea]. Available:

http://www.aragon.es/estaticos/GobiernoAragon/Departamentos/InvestigacionInnovacionUniversidad/Areas/Sociedad_Informacion/Documentos/Estado del arte GameEngines y su impacto en la industria.pdf. [Último acceso: 21 Junio 2017].

12] Crytek, «CryEngine: Features,» Crytek, 2017. [En línea]. Available: https://www.cryengine.com/features. [Último acceso: 21 Junio 2017].

13] Epic Games, «Unreal Engine Features,» 2017. [En línea]. Available: https://www.unrealengine.com/features. [Último acceso: 22 Junio 2017].

14] Unity Technologies, «Un editor flexible y con múltiples prestaciones,» 2017. [En línea]. Available: https://unity3d.com/es/unity/editor. [Último acceso: 22 Junio 2017].

15] R. Marks, «EyeToy, Innovation and Beyond,» 3 Noviembre 2010. [En línea]. Available: https://blog.us.playstation.com/2010/11/03/eyetoy-innovation-and-beyond. [Último acceso: 24 Junio 2017].

16] Microsoft, «Kinect for Windows SDK,» 2017. [En línea]. Available: https://msdn.microsoft.com/library/dn799271.aspx. [Último acceso: 24 Junio 2017].

17] ASUSTek Computer Inc., «Xtion PRO,» 2017. [En línea]. Available: https://www.asus.com/es/3D-Sensor/Xtion_PRO/overview/. [Último acceso: 24 Junio 2017].

18] PrimeSense, «NITE 2.2.0.11,» 24 Octubre 2013. [En línea]. Available: http://openni.ru/files/nite/index.html. [Último acceso: 24 Junio 2017].

19] Microsoft, «Face tracking Library,» 2017. [En línea]. Available: https://msdn.microsoft.com/es-es/library/dn782034.aspx. [Último acceso: 24 Junio 2017].

20] A. K. Jain, R. P. W. Duin y J. Mao, «Statistical Pattern Recognition: A Review,» *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, nº 1, pp. 4-37, 2000.

21] W. H. Atomi, The effect of data preprocessing on the performance of artificial neural networks techniques for classification problems, Faculty of Computer Science and Information Technology: University Tun Hussein Onn Malaysia, Diciembre, 2012.

22] OpenCV Dev Team, «Neural Networks,» 24 Enero 2017. [En línea]. Available: http://docs.opencv.org/2.4/modules/ml/doc/neural_networks.html. [Último acceso: 25 Junio 2017].

23] OpenCV Dev Team, «Support Vector Machine,» 24 Enero 2017. [En línea]. Available:

http://docs.opencv.org/2.4/modules/ml/doc/support_vector_machines.html. [Último acceso: 25 Junio 2017].

24] L. Sung Joo, J. Jaeik, J. Ho Gi, P. Kang Ryoung y K. Jaihie, «Real-Time gaze estimator based on driver´s head orientation for fordward collision warning system,» *IEEE transactions on intelligent transportation systems,* vol. 12, nº 1, pp. 254-266, 2011.

25] S. Li, L. Zicheng y S. Ming-Ting, «Real time gaze estimation with a consumer depth camera,» *Information Sciences,* vol. 1, nº 320, pp. 346-360, 2015.