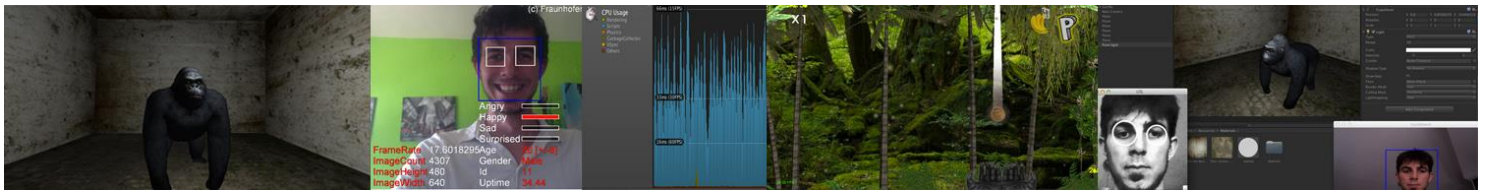




Tesina final de Máster 2013-2014

DetECCIÓN Y SEGUIMIENTO FACIAL EN TIEMPO REAL PARA UN VIDEOJUEGO EN C#



Autor: Steve Rossius
Director: Ramón Pascual Mollá Vayá

Universidad Politécnica de Valencia



Índice

Índice	2
1. Introducción	4
1.1. Motivación.....	4
1.2. Objetivos	4
1.3. Esquema de la obra	5
2. Estado del arte	6
2.1. Introducción histórica.....	6
2.2. Estado del arte del hardware	8
<i>Nintendo Wii</i>	9
<i>Acelerómetro y giroscopio</i>	10
<i>Microsoft Kinect</i>	10
<i>GOOGLE Tango</i>	11
2.3. Unity 3D y librerías de visión por computador	13
2.4. La detección facial: técnicas básicas	14
<i>Detección facial basada en el color</i>	14
<i>Detección de bordes</i>	15
<i>Detección facial basada en redes neuronales</i>	16
<i>Schneiderman y Kanade</i>	17
<i>Viola-Jones</i>	18
<i>Detectores combinados</i>	20
2.5. Técnicas básicas del seguimiento facial.....	21
<i>CAMShift</i>	21
<i>Compressive Tracking</i>	24
2.6. El filtro simple de Kalman.....	28
2.7. Corrección lumínica MSR y MSRCR.....	33
2.8. Holografía digital	35
<i>Efecto de Paralaje</i>	35
3. Diseño y especificación	38
4. Experimentación	41
4.1. Objetos del juego.....	41
4.2. Aspectos estéticos.....	42
4.3. Jugabilidad y evaluación	44
4.4. Conclusiones finales.....	46
5. Futuras líneas de investigación	48
6. Anexo I: Happy Gorilla GDD	50
6.1. Resumen del juego	50
6.2. Características principales	51
6.3. Jugabilidad	51
6.4. El mundo del juego	54
6.5. GUI del juego	55
7. Anexo II: Bibliografía	57

1. Introducción

1.1. Motivación

La continua evolución que se observa en el mundo de la informática, ha originado sistemas computacionales cada vez mas potentes y espectaculares. Siguiendo la ley de Moore [38] cuyo pronostico garantiza una duplicación de la cantidad de transistores contenidos en los procesadores, nos vemos rodeados de múltiples sistemas sofisticados y veloces. Aprovechando al máximo las capacidades que nos ofrecen a día de hoy estos dispositivos, se pretende desarrollar un sistema de diferentes filtros y módulos para el análisis y la extracción de características capaz de seguir los movimientos de un jugador real ubicado frente a la entrada de video del dispositivo.

Se trata de un sistema lo suficientemente simple para poder ejecutarse sin grandes requerimientos de hardware o software, pero lo suficientemente sofisticado para garantizar una fluidez y estabilidad en todo el proceso. Todo el sistema será puesto a prueba mediante el desarrollo de un videojuego exclusivo para comprobar las capacidades del algoritmo de detección y seguimiento desarrollado.

Mediante esta tesina se pretende verificar que mediante dispositivos convencionales como una cámara web, se puede alcanzar y realizar videojuegos basados en la detección y el seguimiento facial, sin la necesidad de basar el análisis en sistemas de hardware altamente sofisticados con múltiples sensores.

1.2. Objetivos

En esta tesina se presentará un nuevo sistema de control para un videojuego, basado en la detección facial y el seguimiento de la cabeza humana. El sistema se podrá utilizar en el software Unity 3D [28] para mover una escena tridimensional mediante el movimiento de la cabeza. Además, para probar a fondo las posibilidades que ofrece el sistema implementado para su uso en un videojuego real, se ha desarrollado un mini-juego de habilidad que refuerza las capacidades reactivas del jugador (ver anexo I).

Mediante ello se estudiará la viabilidad de ejecutar el código de detección y seguimiento conjuntamente con un videojuego con efectos y modelos relativamente vistosos y profesionales, optimizando el sistema para garantizar una experiencia de juego gratificante para el usuario. Mediante el efecto de paralaje se añadirá al proyecto una mayor profundidad visual, aprovechando los datos del seguimiento para modificar adecuadamente el comportamiento de la cámara de la escena.

1.3. Esquema de la obra

Para alcanzar esta meta, primero se hará una pequeña introducción histórica en el capítulo 2 de los diferentes acontecimientos que originaron la evolución actual. A continuación se hablará de los diferentes dispositivos con control multimodal que se han desarrollado en las últimas décadas. Después se presentará la plataforma de desarrollo utilizada para el proyecto para luego poder introducir las técnicas básicas de la detección facial. A continuación se realizará un estudio de dos de las técnicas de seguimiento desarrollados para esta tesina y se explicará un filtrado para la estimación de posiciones a partir de sus localizaciones anteriores. Después se presentará una técnica de corrección lumínica para algoritmos de bajo coste y se estudiará efectos tridimensionales para pantallas convencionales.

En el capítulo 3 se presentará el diseño y las especificaciones del algoritmo final compuesto por varias técnicas presentadas a lo largo de la tesina.

En el capítulo 4 se comprobará su correcto funcionamiento mediante el videojuego desarrollado exclusivamente para esta tesina. Se hará un pequeño repaso a las principales características del título desarrollado y cuales con los requerimientos para el sistema compuesto por detección y seguimiento facial. También se realizará una evaluación de diferentes combinaciones de sistemas con varias técnicas presentadas en el capítulo 2, concluyendo el capítulo con una conclusión y reflexión de la relación con la materia estudiada en el máster.

Para terminar con el capítulo 5, se estudiarán posibles formas de ampliar el código final para una posible aplicación en otras soluciones informáticas.

Los anexos están compuestos por el propio documento de diseño para el videojuego creado para la tesina y la bibliografía empleada para su realización.

2. Estado del arte

Desde los inicios de la computación digital hasta la tecnología avanzada actual, solo ha pasado poco más de medio siglo. La evolución y los avances tecnológicos alcanzados se describirán a continuación, con un enfoque a las técnicas necesitadas para la realización de esta tesina.

2.1. Introducción histórica

Desde el nacimiento de los videojuegos en aquel laboratorio de Higinbotham [37] en 1958, la interacción humano-máquina ha quedado limitada al uso de botones y joysticks. Estos botones han ido adoptando diferentes formas y tamaños, sus nombres se han reinventado numerosas veces, han aparecido en multitud de combinaciones de disposición y ergonomía, pero siempre se trataba de usar las manos para comunicarnos con el software. Debido a que este tipo de interacción ha sido la más eficiente y fiable durante décadas, las alternativas al control clásico no han pasado en la mayoría de los casos de la fase de prototipos o han sido calificados de simples juguetes tecnológicos de poca reputación.

Esta concepción clásica de los videojuegos y la manera de interactuar con ellos está cambiando radicalmente, especialmente desde la aparición de las primeras pantallas táctiles y la expansión de los llamados smartphones. Estos terminales móviles inteligentes están dotados de una potencia de computación equiparable a ordenadores domésticos, permitiendo la aparición de software cada vez más sofisticado y computacionalmente complejo. De esta manera se le permite al desarrollador crear aplicaciones con animaciones, entornos gráficos y características que años atrás eran exclusivas de ordenadores de alta gama, en dispositivos móviles accesible a una gran cantidad de usuarios.

También la aparición de sistemas de visión y captación de movimiento sofisticados para consolas domésticas, ha favorecido claramente el auge de técnicas alternativas de comunicación entre usuario y hardware. Tanto Nintendo, como posteriormente sus principales competidores Microsoft y Sony, han desarrollado todo un arsenal de dispositivos que permiten una comunicación muy distinta al clásico mando. El probablemente más sofisticado y aclamado es sin duda la cámara *Kinect* de Microsoft [22], compuesta por un conjunto de cámaras RGB y sensores infrarrojos que permiten la captación del movimiento y de los gestos del usuario, sin requerir ningún equipamiento adicional. El jugador se convierte de esta manera en el propio mando.

Nintendo en cambio, desarrolló un conjunto de mandos inalámbricos capaces de comunicarse con la consola mediante una barra de infrarrojos localizada junto a la televisión. Esta barra realiza una localización del jugador y sus mandos muy precisa, permitiendo el control del videojuego mediante el movimiento de los mandos además de los botones (figura 1).



Figura 1 Dos maneras diferentes de modificar la interacción humano-maquina: A la izquierda la cámara *Kinect* de *Microsoft* [22] y a la derecha el mando *Wii* de *Nintendo*

Lo verdaderamente sorprendente de esta transición hacia un control mucho más intuitivo y asequible, es que abre el mundo de los videojuegos a jugadores cada vez más “casual”. Hace apenas una década, aun se consideraba los videojuegos como un producto para unos clientes muy bien definidos. Hoy en día, los avances informáticos, las nuevas tecnologías de interacción con el hardware y una sociedad cada vez más digitalizada, han cambiado esta concepción de los videojuegos.

También la continua investigación en el campo de la detección facial, y sus correspondientes sub-campos del seguimiento y reconocimiento, han permitido alcanzar tasas de errores muy pequeñas. Estas tecnologías están siendo utilizadas por grandes compañías del sector informático, como por ejemplo *Facebook* [23], *Samsung* [26] o *Apple* [27], realizando tareas automáticas de etiquetado en fotos, control de dispositivos o reconocimiento automático. Los dispositivos que nos rodean captan cada vez con mayor detalle todas las características de nuestro alrededor, tanto las sociales como las físicas, permitiendo cada vez más una mayor integración en nuestra sociedad culminando en campos como la bioingeniería. Esta evolución pretende difuminar cada vez más los límites entre artificial y natural, creando de esta manera una sociedad que depende fuertemente de su tecnología.

Esta evolución de la tecnología no solo ha cambiado la manera de interactuar con los dispositivos que nos rodean, sino también nuestra manera de visualizar los datos que nos ofrecen. Desde televisores estereoscópicos hasta gafas de simulación 3D como *Oculus Rift* [24], están demostrando que el engaño de percibir lo virtual como real está cada vez más integrado en nuestra tecnología social.

2.2. Estado del arte del hardware

Durante toda la historia de los videojuegos y sus respectivas consolas, se puede observar la evolución de los dispositivos mediante los cuales el jugador interactuaba con su entorno. Desde la aparición de las salas arcade japonesas hasta la domesticación de las consolas con la Atari 2600: el concepto de botones y palancas no ha cambiado. Y durante todas las siguientes generaciones se ha mantenido constante, con más o menos botones y adoptando diferentes formas y tamaños. Hubo pocas excepciones que proyectaban un espíritu de expandir esta manera de control, como por ejemplo la Nintendo *Zapper* de 1985 [29] para la NES.

Este peculiar dispositivo con forma de pistola se usaba como mando en algunos juegos de disparos clásicos y permitía una estimación del disparo mediante la interacción con el tubo de rayos catódicos. Cuando se apretaba el gatillo, la pantalla se ponía negra durante un frame, y en la siguiente imagen se iluminaban en blanco los correspondientes targets del juego. El *Zapper* detectaba este cambio de negro a blanco y permitía estimar si se estaba apuntando o no. Todo el proceso era casi invisible para el ojo humano, y presenta uno de los primeros sistemas comercializados de control multimodal.



Figura 2 El *Zapper* original de Nintendo presentado en 1985 para su consola NES

Curiosamente, el *Zapper* prácticamente ha quedado sin uso a día de hoy, debido a que precisa de una televisión de rayos catódicos para su funcionamiento. Aunque fue un gran éxito en el momento de su aparición, no tuvo un impacto trascendental en el desarrollo de futuros mandos. Esto es debido a que la tecnología simplemente no estaba lista del todo para un uso doméstico.

Esto cambió con la revelación de la consola doméstica *Wii* de Nintendo [25] en el E3 de 2005 y desde ahí, el interés y la aceptación de métodos alternativos para el control de videojuegos ha ido fomentando la aparición de técnicas cada vez más futuristas y espectaculares de manejar las acciones en los mundos digitales. Con ello se intenta estimular sobre todo a los jugadores casuales, es decir, personas que no habían estado muy en contacto con videojuegos y por lo tanto tenían un nivel de capacidades y conocimiento menor que jugadores expertos. Se trataba principalmente de abrir el mercado a un grupo de consumidores mayor, disminuyendo de manera crucial la curva de aprendizaje inicial para atraer a más clientes.

Kulshreshth y LaViola Jr. [16] realizaron un estudio sobre los beneficios de la incorporación de un sistema de detección y seguimiento facial en la experiencia de los jugadores del videojuego. Para ello se eligieron 40 jugadores tanto casuales como expertos y se dividieron en dos grupos: el primero jugaba sin el seguimiento facial y el segundo grupo con seguimiento facial. También se eligieron cuatro videojuegos de diferentes tipos, entre ellos un FPS, un simulador de vuelo y un juego de conducción. Finalmente se llegó a la conclusión de que el seguimiento facial es de gran ventaja en juegos más pausados como los FPS y de menos beneficio en juegos más rápidos como los de conducción. Además se demostró que el seguimiento facial supuso una mejora únicamente para los jugadores expertos, que ya disponían de un amplio conocimiento de las mecánicas de los juegos. En el caso de los jugadores casuales no se detectó ningún tipo de mejora jugando con seguimiento facial. La inclusión de sistemas alternativos, tanto complementarios a maneras clásicas de control como incluso aquellas completamente sustituyentes, supone una mayor inmersión del jugador en los mundos virtuales, y en la mayoría de los casos enriquece la experiencia final del jugador.

Esta mayor inmersión en mundos digitales por parte del jugador es una de las actuales propuestas más interesantes de la industria. No solo se pretende desarrollar sistemas de control que permitan la emulación de movimientos y reacciones “más” humanas y naturales que sistemas tradicionales, sino también de métodos para transmitir mayor realidad al jugador mediante pantallas holográficas y visores 3D. Proyectos como *Google Tango* [21] o *Oculus Rift* [24] reaccionan frente a los propios movimientos del jugador, difuminando de esta manera el concepto clásico de control. Mediante gafas 3D el jugador percibe una inmersión visual total en los mundos digitales, sincronizándose los movimientos de su cabeza con la rotación de la cámara del juego.

A continuación se hará un breve resumen de las técnicas y dispositivos más representativas para el control multimodal en videojuegos.

Nintendo Wii

La Nintendo *Wii* [25] es prácticamente una extensión de la arquitectura de su consola predecesora: la *GameCube*; siendo únicamente un 1.5x hasta 2x más rápida que está. Debido a la filosofía de abstenerse a un conflicto directo contra Sony y Microsoft por el título de consola más potente gráficamente hablando, los ingenieros de Nintendo desarrollaron el llamado *Wiimote* (figura 1). Este mando inalámbrico, además de poseer los clásicos botones, permitía la captación de su movimiento por parte de la consola para transferirla directamente al control de la escena en el videojuego.

Mediante una simple barra de LED's infrarrojos posicionada cerca de la televisión, la consola era capaz de determinar la posición y dirección del mando con una gran precisión por debajo de una distancia máxima de cinco metros. Para ello el mando simplemente dispone de un sensor que capta la posición de la barra de LED's y la distancia entre sus dos esquinas y mediante triangulación calcula la posición del jugador en el campo de visión.

Además, para registrar movimientos bruscos horizontales el mando también dispone de un acelerómetro cuyo funcionamiento se explicará a continuación.

Acelerómetro y giroscopio

Un acelerómetro permite la medición de la aceleración que sufre cuando es movido. En su modelo más simple, no es nada más que una masa junto a un dinamómetro en dirección a la aceleración a medir. De esta manera, un dispositivo otorgado con esta tecnología es capaz de detectar la velocidad del movimiento, que, junto a otro complemento, permite un manejo sofisticado en un videojuego: el giroscopio.

Un giroscopio no es nada más que un cuerpo con simetría de rotación girando alrededor de un eje. Este eje mantiene su orientación independientemente a la fuerza externa que intente desviarlo gracias a la primera ley de Newton del movimiento. Gracias a esta propiedad, es muy sencillo averiguar la inclinación de un dispositivo una vez calibrado el punto de inicio.

Estas dos tecnologías incluidas en la inmensa mayoría de smartphones, permiten controlar la escena mediante inclinación, movimiento y agitación, compensando de esta manera la falta inicial de botones que presentan pantallas táctiles.

Microsoft Kinect

El sistema que más ha revolucionado el sector de videojuegos y casi más al desarrollo de sistemas de visión artificial es el dispositivo *Kinect* [22] (figura 3) de Microsoft presentado en 2010. Debido a su gran precisión y su muy bajo coste comparado con otras soluciones de visión comerciales, la cámara de Kinect ha sido objeto de estudio y desarrollo para muchos investigadores y programadores.

El sistema *Kinect* consta de una cámara RGB para texturizar la imagen digital y un sensor de profundidad compuesto por un laser de infrarrojos que proyecta una malla y un sensor CMOS monocromático capaz de extraer de esta malla la información tridimensional de la escena bajo luz ambiente. De esta manera se obtiene una imagen virtual lo suficientemente fiel a la realidad que permite al sistema interpretar los movimientos del jugador y transferirlos directamente a la acción del videojuego [6].

De esta manera, *Kinect* es capaz de interpretar gestos y movimientos de hasta seis jugadores según las especificaciones de Microsoft, aunque los desarrolladores limitaron este número únicamente al máximo número de personas que caben en el campo de visión del sistema.

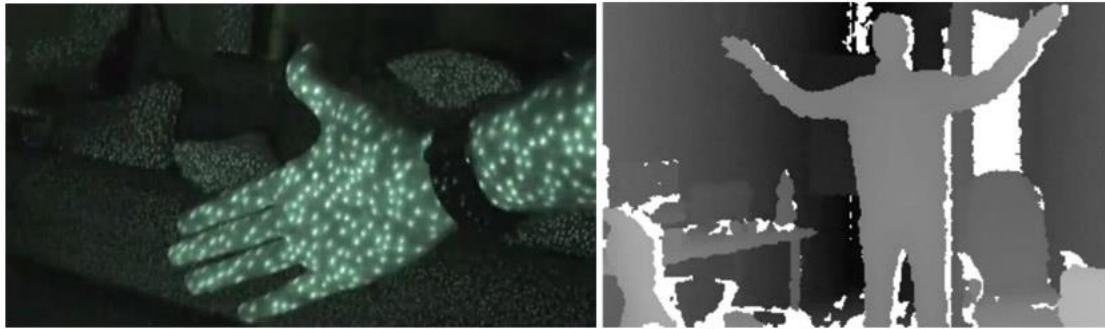


Figura 3 La malla proyectada al espacio en la imagen de la izquierda y a la derecha el mapa de profundidad de los objetos detectados, yendo de un gris claro (cerca) a un gris oscuro (lejos).

El sistema además incluye un array de micrófonos, permitiendo el reconocimiento de voz de múltiples jugadores. El sistema es por lo tanto capaz de detectar movimientos y gestos, además de reconocer caras y voces para el manejo, y todo ello sin la necesidad de un mando.

GOOGLE Tango

La idea principal detrás del proyecto *Tango* [21] es percibir el mundo real mediante los correspondientes sensores para modelar nuestro entorno en tiempo real de manera tridimensional. Surge del departamento llamada ATAP (*Advanced technology and projects*) originalmente de la propiedad de *Motorola*.

La recreación de la realidad ocurre mediante la toma de alrededor 250.000 medidas de la realidad por segundo. Estas medidas se toman a partir de varios sensores repartidos en las partes frontal y traseras de un dispositivo móvil:

- Una cámara trasera de 4MP con sensor infrarrojo trasera
- Una cámara de ojo de pez con un campo de visión de 180° trasera
- Una cámara de 120° frontal
- Una cámara de profundidad disparando a una resolución de 320x180 y 5Hz

De manera parecida al sistema *Kinect* de Microsoft [22], se proyecta una malla de puntos, cuyo tamaño permite estimar su distancia respecto al proyector. Mediante esta técnica se crea un mapa tridimensional del entorno del usuario, permitiendo la ejecución de todo tipo de contenidos y aplicaciones incorporando toda esta información. El proyecto se destina principalmente al mapeado exacto de interiores, donde se consiguen muy buenos resultados incluso bajo condiciones de poca visibilidad y oscuridad. La recreación de nuestro entorno puede ser utilizado para todo tipo de aplicaciones, entre otras la navegación en interiores y juegos gráficos de realidad aumentada sin marcadores.



Figura 4 Prototipo del dispositivo móvil con tecnología Tango a la izquierda. A la derecha: Oculus Rift y los 40 LED's infrarrojos mediante los que se realiza el seguimiento de los movimientos de la cabeza del usuario.

La mayoría de los sistemas presentados usan tecnología infrarroja para medir las distancias relativas entre el dispositivo y el usuario o la escena. También se ha desarrollado sistemas complejos con diferentes combinaciones de cámaras y otros sensores para captar ciertos movimientos y traslaciones. Los algoritmos y técnicas son propios para cada uno de estos dispositivos, por lo que su desarrollo, la emulación y el estudio requieren del hardware adecuado.

En esta tesina se pretende desarrollar un sistema estacionario capaz de detectar el movimiento del usuario con tecnología convencional, usando una simple cámara web como la que incluyen la mayoría de ordenadores.

2.3. Unity 3D y librerías de visión por computador

La detección facial de este trabajo servirá como base para el futuro desarrollo de un videojuego, integrando las funcionalidades de seguimiento con el control de un personaje o una escena. De esta manera resulta lógico plantear su desarrollo directamente sobre uno de los motores gráficos y entornos para la creación de videojuegos más aceptados y utilizados a día de hoy: Unity 3D [28].

Unity 3D es un paquete de software tanto para ordenadores Apple *Macintosh* y *PC Windows* capaz de crear contenidos para la inmensa mayoría de plataformas disponibles, entre ellas Sony *PS4*, Microsoft *XBOX One*, plataformas *Android* y Apple *IOS*. Unity 3D se caracteriza por un manejo simple e intuitivo y permite la creación rápida, sofisticada y multiplataforma de videojuegos.

El scripting en Unity está basado en el framework de Mono, la versión open-source del .Net framework de Microsoft, permitiendo la ejecución y el desarrollo en Javascript, Boo y C# [7].

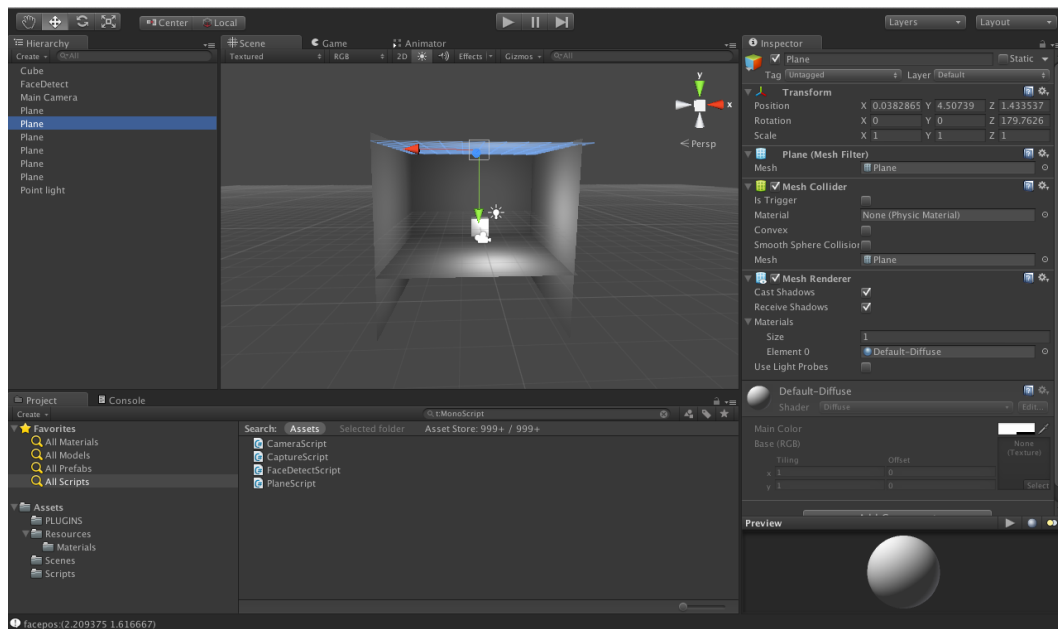


Figura 5 Ventana principal de Unity 3D con el programa desarrollado para este trabajo.

La tarea del propio reconocimiento se realizará mediante un script que se ejecutará dentro de Unity y utilizará la librería OpenCV especialmente diseñada para este tipo de tareas. OpenCV quizás sea la primera elección a la hora de elegir una biblioteca completa y sólida para el desarrollo de aplicaciones de visión artificial. Inicialmente desarrollada por Intel, alberga ya a más de 500 funciones escritas en C y C++ y optimizadas para el proceso de soluciones en tiempo real [8]. Debido a que el scripting de Unity utiliza C# como lenguaje, se necesita de un wrapper específico capaz de traducir las funciones de OpenCV. En este caso es necesario la instalación de la librería OpenCvSharp, que hará de puente entre OpenCV y Unity.

2.4. La detección facial: técnicas básicas

La detección facial y su correspondiente seguimiento y reconocimiento, han sido objeto de estudio y resultan ser un área de gran interés para compañías informáticas o incluso instituciones gubernamentales. Desde los años 90 se han ido proponiendo nuevos métodos y sus respectivas mejoras, convergiendo finalmente en los sistemas robustos y rápidos que se desarrollan actualmente.

La detección facial es la localización de las posibles caras que se pueden encontrar en una imagen, extrayendo para ello el centro de la cara, su orientación y su escala. Una vez detectada la cara se aplican algoritmos de seguimiento y/o reconocimiento, según el contexto y la finalidad de la aplicación. Este trabajo se centrará en la tarea del propio seguimiento que permitirá sincronizar el movimiento de la escena tridimensional con el recorrido de la cabeza del usuario dentro del campo de visión de la cámara.

Hoy en día se incluyen sistemas de detección y reconocimiento facial en numerosos aparatos que años atrás se consideraban utópicos. Cualquier videocámara de gama media suele ser capaz de detectar las caras y ajustar el enfoque de manera automática y rápida, generando imágenes de gran nitidez y calidad [36]. Smartphones son capaces de averiguar si el usuario está viendo la pantalla, y de esta manera, pausar la reproducción del contenido digital de forma automática cuando no lo esté haciendo. Y quizás uno de los campos más interesantes para la detección facial es la identificación personal, ya sea para desbloquear un portátil o para entrar en áreas restringidas que requieren un alto nivel de seguridad.

En todos los campos, la detección facial se caracteriza por requerir de una mínima cooperación por parte del usuario. El reconocimiento se produce de manera casi automática e instantánea, dependiendo del subsistema que realiza la extracción de información a partir de la imagen tomada del usuario. Una de las mayores dificultades que se percibe principalmente en la identificación personal, es el cambio constante que sufre una cara humana, dificultando de manera la tarea de reconocimiento enormemente. La biometría facial de una persona cambia radicalmente para el sistema si el usuario decide llevar gafas o dejarse barba. Además, el rendimiento del sistema se ve muy afectado por cambios de iluminación y posición de la cámara y del usuario. Para todos estos problemas se han propuesto soluciones más o menos eficientes que se expondrán a continuación. También se realizará un repaso de las técnicas más representativas, sus principales características y las diferentes metodologías empleadas.

DetECCIÓN FACIAL BASADA EN EL COLOR

El uso del color para detectar caras es una técnica comúnmente aplicada. Se trata de una manera sencilla y muy rápida de clasificar las diferentes regiones de una imagen basándose en su contenido cromático. La velocidad del procesamiento permite su uso en entornos de detección en tiempo real. Existen numerosos

espacios de color a partir de los cuales se realiza la extracción de información, cada uno con sus ventajas y desventajas.

Uno de los más extendidos y aceptados en la imagen digital es el espacio de color RGB (Red Green Blue), compuesto por tres mapas de bits que contienen toda la información de color según el tono. Se trata de un sistema aditivo, la suma de todos los colores genera el blanco, y su ausencia el negro.

El espacio HSV (Hue Saturation Value) es una transformación no lineal del espacio de color RGB y se aproxima más a la visión humana. La matiz (Hue) se representa en una región circular, cuyo centro es el blanco. La distancia del centro hacia el color en la región circular define la saturación. Cada color primario está separado del otro 120° y los colores mixtos están definidos entre los espacios resultantes. Una tercera dimensión es añadida para definir el negro, generando así un cono en el espacio, cuya altura viene dada por el brillo o valor.

YCbCr e Y'CbCr son espacios de color usados principalmente en la transmisión y difusión de imágenes digitales ya que eliminan gran parte de la redundancia que se presenta en el espacio RGB. "Y" es la componente lumínica y representa la imagen en escala de grises, mientras que Cb y Cr son las dos componentes de croma correspondientes al azul y rojo. Y' (con prima) en cambio, representa la misma información lumínica añadiendo una corrección gamma de la señal. Cabe destacar que no se trata de un sistema absoluto, ya que simplemente se recodifica la información contenida en el espacio RGB para mejorar la robustez frente a distorsiones y disminuir la redundancia. Igual que ocurre en el espacio HSV, la componente lumínica en el espacio YCbCr está separada de la propia información cromática, permitiendo una mayor robustez y independencia frente a cambios de iluminación.

Hsu et al. [13] usaron un detector basado en los espacios de color YCbCr y HSV para extraer las zonas básicas de la cara tales como la boca, los ojos y los contornos faciales aplicando mapas de características extraídos directamente del análisis de las componentes lumínicas y cromáticas de la imagen digital. Como limitación para una detección fiable, se precisa que al menos la zona de los ojos y la boca sean visibles para el sistema, consiguiendo así un porcentaje de éxito de casi 90%.

Detección de bordes

Mediante la detección de bordes se pretende diferenciar cambios importantes en la composición de la imagen digital. En el mejor de los casos la aplicación de un detector de bordes tiene como resultado un conjunto de líneas que describen los contornos del objeto. Además se consigue una disminución de los datos a procesar, facilitando la identificación del objetivo. En la practica, la detección de bordes no suele dar resultados óptimos de manera trivial y los contornos sufren de una fragmentación.

Yow y Cipolla [14] plantearon un detector de bordes aplicando un filtro gaussiano con la segunda derivada, donde los máximos locales indican la localización de las características faciales a detectar. Posteriormente se aplica un detector de bordes

de Canny en cada uno de estos máximos locales, uniendo los bordes encontrados según su posición, orientación y su grosor. Todas estas características se almacenan en un vector de características, al cual se recurre para calcular su matriz de medias y covarianzas respecto a un conjunto de entrenamiento. Una característica facial es dada por válida si la distancia de Mahalanobis del vector de características es menor a un umbral. Finalmente se evalúan y se agrupan todas las características usando una red Bayesiana. Mediante esta técnica, Yow y Cipolla [14] consiguieron un porcentaje de aciertos del 85% en un conjunto de 110 imágenes, permitiendo pequeñas variaciones en pose, escala y orientación. Su uso queda limitado debido a la complejidad del proceso de detección, haciendo su uso inviable en aplicaciones que requieren detección en tiempo real.

Detección facial basada en redes neuronales

La idea principal detrás de estos sistemas es el entrenamiento de una o varias redes neuronales para realizar una detección facial positiva o negativa, es decir, que en la imagen de test se ha encontrado una cara o no se ha encontrado. La dificultad en este método reside en la necesidad de un entrenamiento de la red con un conjunto de caras y además de un conjunto prototípico de no-caras. Todo lo que no sea una cara pertenece al conjunto de las no-caras, por lo que simplemente se aumenta el conjunto de training conforme avance la detección.

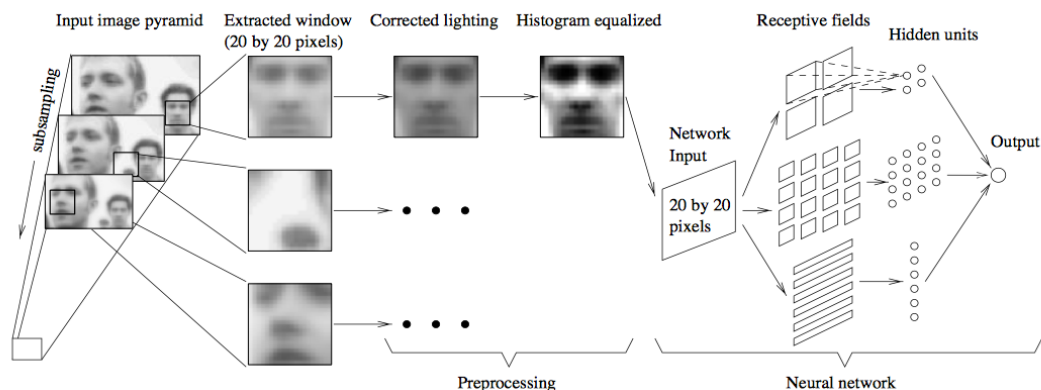


Figura 6 Mapa del algoritmo básico presentado por Rowley, Baluja y Kanade. [1]

El primer paso tras recibir la imagen con las caras, es un preprocesado. La imagen se escala a diferentes tamaños y de cada imagen se extraen regiones de 20x20 píxeles realizando un barrido por toda la imagen. A todas estas imágenes se les aplica una corrección de la luz y una ecualización del histograma para obtener mayor uniformidad y precisión mediante una máscara ovalada, solo modificando los píxeles dentro de esta región. Los píxeles que caen fuera se consideran como fondo y son irrelevantes para la detección por lo que no precisan de ningún procesado.

A continuación la red neuronal clasificará cada una de estas regiones de 20x20 píxeles como cara o no-cara. La red neuronal dispone de diferentes tipos de unidades escondidas, que en su conjunto permiten la extracción de características locales potencialmente representativas para la detección facial.

Debido a que la salida de una única red neuronal puede presentar multitud de falsos positivos, se han desarrollado dos técnicas diferentes para disminuir este número. La primera de ellas trata de unir las caras encontradas para cada localización y escala. Si el número de caras en ese espacio es mayor que un umbral, entonces se puede clasificar como cara. La segunda opción es el entrenamiento de diferentes redes neuronales, cada una inicializada con pesos aleatorios y entrenadas con conjuntos de no-caras diferentes y procesar de esta manera si la imagen contiene una cara o no.

Schneiderman y Kanade

En la vida real, la inmensa mayoría de las fotos con personas, no suelen ofrecer una vista frontal y optima para la detección facial. Las caras están orientadas y rotadas de mil maneras, dificultando extremadamente su detección para sistemas de visión artificial. Basándose en esta conocida problemática, Schneiderman y Kanade [2] propusieron el uso de dos clasificadores, cada uno para una diferente orientación, es decir: uno para la pose frontal y otro para el perfil derecho. Aprovechándose de la simetría facial, el clasificador del perfil izquierdo es simplemente un reflejo del derecho, disminuyendo de esta manera la carga computacional.

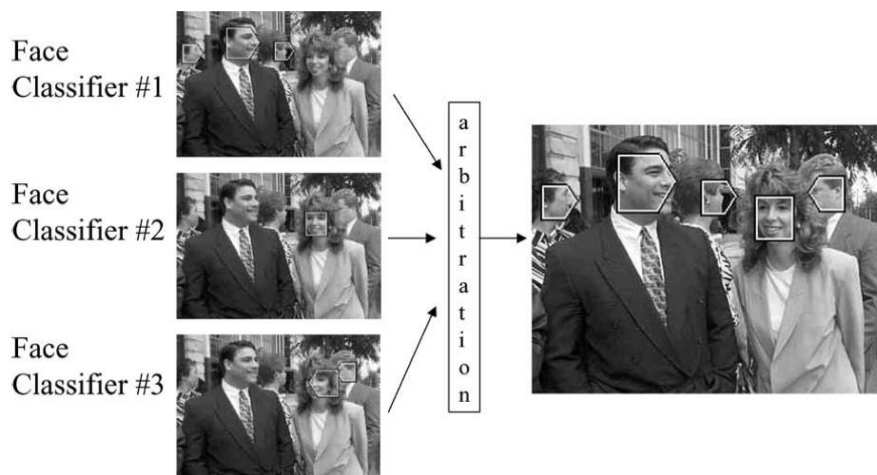


Figura 7 Combinación de los diferentes clasificadores propuestos por Schneiderman y Kanade [2]

La detección se realiza recorriendo la imagen original y sus versiones escaladas mediante los diferentes clasificadores, permitiendo de esta manera la detección de caras de diferentes tamaños y orientación. En el caso de encontrar la misma cara múltiples veces, se combina el resultado, como en el caso de la mujer en la figura 7.

El clasificador basa su funcionamiento en la distinción entre partes. Una cara es un conjunto de partes, típicamente hablando se trata de la nariz, los ojos etc. Con estas partes se forman conjuntos compuestos por variables estadísticamente dependientes.

$$\frac{P(image|object)}{P(image|non-object)} > \lambda = \frac{P(non-object)}{P(object)} \quad (1)$$

Esta adaptación de la regla de Bayes permite calcular la proporción de probabilidad (termino de la izquierda), y si está es superior al umbral λ , el clasificador decide que el objeto está contenido en la imagen, es decir que se ha encontrado una cara.

Viola-Jones

Se trata de un framework extremadamente robusto y rápido en la detección de caras, pero muy lento en la fase de aprendizaje. Principalmente se compone de tres contribuciones: el uso de imágenes integrales, el algoritmo de aprendizaje basado en *AdaBoost* y el uso de varios clasificadores en cascada. Para minimizar los efectos de cambios en la iluminación se precisa de un preprocesado de las imágenes, aplicándose una normalización de media y varianza.

El primer elemento novedoso del framework es el uso de la llamada imagen integral. Se trata de calcular el valor de cada pixel $p(x, y)$ como suma de todos sus vecinos de arriba y a su izquierda (Ec. 2). Esta técnica puede ser aplicada a cualquier posición y escala de la imagen, formando rectángulos que contienen la diferencia de la suma de sus pixeles de cada área. De esta manera se puede expresar la suma de cualquier área rectangular dentro de la imagen original como:

$$I(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (2)$$

$$sum = I(C) + I(A) - I(D) - I(B) \quad (3)$$

donde A, B, C y D pertenecen a la imagen integral I (figura 8). En el desarrollo de esta tesina, se ha optado por escribir un script propio para calcular la imagen integral de cualquier imagen monocromática. Primeramente se genera la imagen integral y se rellena con 0 toda la matriz de floats.

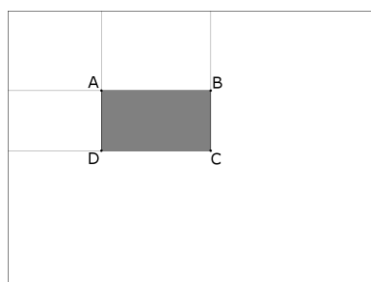


Figura 8 Área definida por cuatro esquinas A, B, C y D dentro de la imagen integral

Después se procede a calcular cada uno de los valores de la primera fila como la suma entre el valor real del pixel y el sumatorio de todos los pixeles anteriores.

$$I(0, y) = i(0, y) + I(0, y - 1) \quad (4)$$

De igual manera se procesa la primera columna.

$$I(x, 0) = i(x, 0) + I(x - 1, 0) \quad (5)$$

Para el resto de la imagen, el valor de cada pixel de la imagen integral viene dado por la suma de todos los pixeles a su izquierda y superiores a el al valor actual del pixel en la imagen original restándole el sumatorio de la diagonal según:

$$I(x, y) = i(x, y) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1) \quad (6)$$

Dichas características rectangulares pueden ser procesadas muy rápidamente usando esta representación intermedia y se denomina características *Haar*. Estas características se definen por la combinación de varios rectángulos blancos y negros adyacentes de igual tamaño que mantienen su posición relativa.

El segundo elemento es una variación del algoritmo de aprendizaje *AdaBoost* capaz de seleccionar las características *Haar* que mejor separan las caras positivas de las negativas (Fig. 9), y posteriormente realizar el entrenamiento del clasificador con estas características. *AdaBoost* combina varios de estos clasificadores débiles para formar un clasificador fuerte. Los clasificadores se denominan débiles por la simple razón de que su porcentaje de acierto no es mucho mayor que el que se consigue adivinando aleatoriamente. Un clasificador débil $h_j(x)$ está compuesto de esta manera por la característica f_j , un umbral θ_j para dicha característica y un indicador de paridad p_j que muestra la dirección de la desigualdad. *AdaBoost* elije un conjunto de estos clasificadores débiles asignándoles pesos formando de esta manera el clasificador fuerte.

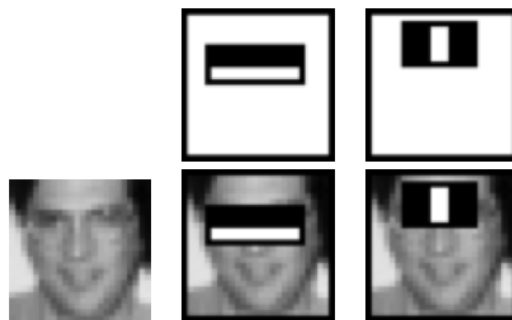


Figura 9 Los dos mejores clasificadores *Haar* para la tarea de detección facial, según el algoritmo *AdaBoost*.

Por último se crea una cascada de clasificadores, disminuyendo de esta manera el tiempo de computo y consiguiendo una mejora notable en el rendimiento del clasificador final. Esta cadena de filtros rechaza regiones estadísticamente no interesantes en una fase muy temprana y se retienen aquellas regiones con mayor probabilidad de contener una cara. El orden de los filtros dentro de la cascada es definido justamente por los pesos asignados por *AdaBoost*, dando prioridad a aquellos filtros con pesos elevados para eliminar regiones de poco interés en una fase temprana del proceso.

Con esta combinación final de clasificadores débiles formando un clasificador fuerte, se alcanza una mejora en tiempo de un 15x frente a las redes neuronales y un 600x frente al método de *Schneiderman-Kanade*, siendo los resultados equiparables a los otros métodos. Justamente por esta ventaja de velocidad, el método presentado por Viola y Jones es el más utilizado a día de hoy.

Detectores combinados

Actualmente existe una gran tendencia en combinar varios de los algoritmos presentados en cascada, obteniendo de esta manera mejores resultados que usando los algoritmos primitivos.

Anila y Devarajan [15] propusieron un detector facial compuesto por operadores Sobel, un detector de bordes propio para analizar sub-ventanas y el uso de redes neuronales para realizar la propia clasificación de las características faciales. En un primer proceso se eliminan ruidos mediante filtros de media y se aplica una ecualización del histograma para optimizar el contraste de la imagen digital. Después se aplica el operador Sobel para realizar una detección simple de bordes. Al resultado se le aplica otro detector capaz de diferenciar entre regiones pertenecientes al fondo y a la propia cara mediante el uso de sub-ventanas definidas por los bordes de Sobel y clasificándolos según su media. Esta salida es procesada en una red neuronal de propagación hacia atrás compuesta por 4 neuronas de entrada, al menos una capa de neuronas ocultas de tamaño 4 y una neurona salida. La clasificación consigue un porcentaje de acierto de un 95,33% en el conjunto de imágenes BioID compuesta por 1520 cara monocromáticas de 23 individuos diferentes bajo diferentes condiciones lumínicas y diferentes tamaños de las caras respecto al fondo.

2.5. Técnicas básicas del seguimiento facial

Una vez detectada la cara, se inicializa un proceso de seguimiento. El seguimiento tiene como objetivo devolver la posición y el tamaño actual de la cara a seguir en una sucesión de imágenes de la manera más exacta y fiable posible, excluyendo y anulando el mayor número de efectos distorsionadores posibles. El seguimiento supone una concordancia temporal de las imágenes que contienen la cara a seguir, y por lo tanto un desplazamiento fluido del objetivo.

El seguimiento debería de ser lo más restrictivo posible, permitiendo pequeñas variaciones de forma, tamaño y color de la cara, pero excluyendo a otras caras completamente diferentes a la cara objetivo y otras áreas que se pueden interpretar como semejantes en color, tono o forma. Diferenciar albaricoques de manzanas rojas es una tarea casi trivial para un humano, pero terriblemente difícil para un sistema de visión artificial. Esta tarea del seguimiento pertenece a otro campo de la visión artificial: el reconocimiento facial.

Además, el seguimiento tiene que ser computacionalmente menos costoso que una nueva detección facial para justificar su aplicación. De otra manera se podría lanzar con cada nueva imagen una nueva detección, garantizando de esta manera la localización exacta de la cara del usuario según los criterios del detector. A continuación se expondrán dos algoritmos de seguimiento estudiados e implementados para esta tesina.

CAMShift

El algoritmo de seguimiento CAMShift (*Continuously Adaptive MeanShift*) [32] se debe entender como una ampliación del MeanShift y se compone principalmente por cuatro pasos: creación del histograma, cálculo de la probabilidad facial de cada pixel, desplazar el rectángulo que contiene la cara a su nueva posición y calcular su tamaño y ángulo de giro.

El histograma de color solo se calcula para el rectángulo contenedor de una cara ya detectada y cuyo desplazamiento se quiera seguir. Este rectángulo inicial suele ser la salida directa de una detección facial previa a CAMShift. Para caracterizar los valores exactos que describen la piel humana conviene usar el espacio de color HSV, compuesta por los tres canales de tono, saturación y matiz. El mayor número de píxeles en este histograma lo formaran los píxeles pertenecientes a la tonalidad de la piel.

Para generar el histograma es necesario definir su resolución, formato de salida y el rango que permite clasificar cada una de las barras del histograma según su valor original de la imagen de entrada.

```
int[] histSize = new int[] { 30 };
float[] hRanges = { 0.0f, 20f };
float[] sRanges = { 50f, 255f };
```

```
float[][] ranges = { hRanges, sRanges };
hist = new CvHistogram (histSize, HistogramFormat.Array, ranges, true);
```

Previamente al propio cálculo del histograma es necesario recortar únicamente aquella zona de la imagen que sea de interés, es decir, la cara del usuario. Sabiendo del detector facial donde se encuentra, simplemente se necesita trazar una región de interés y convertir la cara contenida al espacio de color HSV.

```
smallImg.ROI = trackingWindow;
IplImage HSV = new IplImage (smallImg.ROI.Size, BitDepth.U8, 3);
Cv.CvtColor (smallImg, HSV, ColorConversion.BgrToHsv);
smallImg.ResetROI();
```

El siguiente paso es aislar la componente H correspondiente a la tonalidad y generar a partir de un umbral el valor que definirá la máscara. Este filtrado se realizará para cada nueva imagen, generando cada vez una máscara individual para el frame actual.

```
Cv.InRangeS (HSV, new CvScalar (maxH[0]-5, maxS[0]-5, maxV[0]5, 0), new CvScalar (maxH[0]+10,
maxS[0]+70, maxV[0]+50, 0), mask);
```

Una vez obtenida la máscara se procede a calcular el histograma no acumulativo con el canal H de la imagen original. El histograma se normaliza según el pico máximo de píxeles pertenecientes a una tonalidad. El histograma así obtenido es el mismo para todo el resto del proceso, y se recurrirá a él para clasificar si los nuevos píxeles mediante la probabilidad facial.

```
hist.Calc (imgHUE, false, mask);
hist.GetMinMaxValue (out minValue, out maxValue);
hist.Normalize (smallImg.Width * smallImg.Height * 255 / maxValue);
```

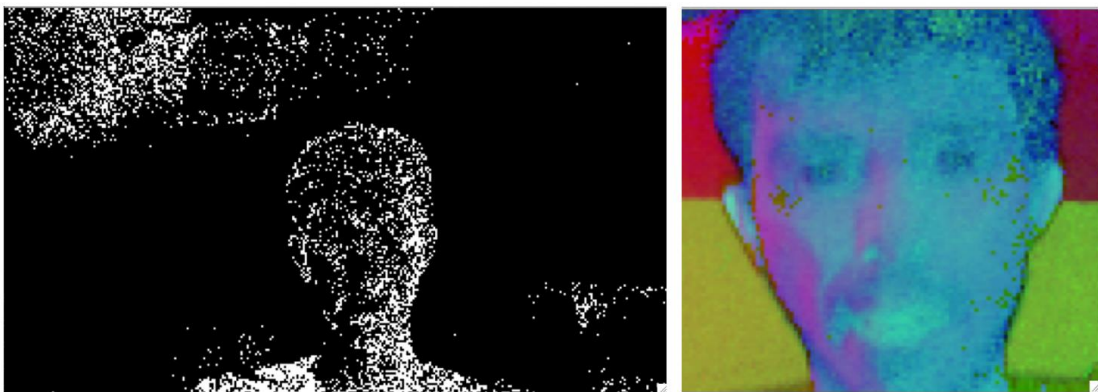


Figura 10 Retroproyección del histograma, filtrando únicamente tonos de piel. A la derecha se observa la cara convertida a espacio de color HSV. Toda la cara tiene el mismo tono, por lo que filtrar esta componente es más fácil.

La probabilidad facial de cada píxel no es otra cosa que determinar que probabilidad tiene el tono de cada píxel nuevo de pertenecer a cierta tonalidad del

histograma inicial. Este proceso se conoce como retroproyección del histograma y permite encontrar las tonalidades definidas en el histograma inicial en cualquier imagen posterior rápidamente. Asignando esta probabilidad, los píxeles más cercanos al tono de la piel reciben un peso mayor, mientras aquellas zonas muy distintas reciben pesos pequeños. Esto se traduce en una retroproyección que representa con blanco la tonalidad buscada, con grises aquellas cercanas y con negro las tonalidades completamente distintas.

A partir del histograma original y el canal H de la imagen actual se procede a calcular la retroproyección y se combina con la máscara binaria que se ha generado para este frame.

```
hist.CalcBackProject (imgHUE, backProject);
backProject.And (mask, backProject);
```

El siguiente paso es desplazar la estimación del rectángulo contenedor de la cara hacia su nueva posición respecto a la probabilidad calculada previamente dentro del rectángulo contenedor actual. Para ello se averigua el centro del área con mayor densidad de píxeles con la mayor probabilidad de tono buscado y se desplaza el rectángulo en dicha dirección. Este proceso es el que se conoce como MeanShift y se repite varias veces para centrar bien la nueva posición del rectángulo contenedor.

El método implementado en OpenCv tiene como entrada la propia retroproyección calculada a partir del histograma, la ventana inicial para la búsqueda y los criterios para finalizar la búsqueda. Estos criterios están definidos por la precisión deseada y el número de iteraciones para la propia búsqueda. La salida es el nuevo rectángulo que contiene la nueva posición de la cara.

```
Cv.CamShift (backProject, trackingWindow, new CvTermCriteria (10, 1), out trackcomp);
trackingWindow = trackcomp.Rect;
```

La gran diferencia con respecto al MeanShift reside en que se calcula el tamaño y el ángulo de giro de cada nueva posición encontrada, generando así la adaptación continua.

CAMShift es así, un algoritmo rápido que requiere poca capacidad computacional y parece ideal para una implementación en tiempo real donde las imágenes provienen directamente de una WebCam. La librería OpenCV incluye ya un método completo de CAMShift, que permite la creación de la retroproyección del histograma y el propio seguimiento permitiendo el ajuste de diversos criterios.

A pesar de todas las ventajas, CAMShift también tiene diversas debilidades. Extraer el tono de piel exacto a la hora de realizar el histograma de la cara a seguir, es un proceso delicado, ya que conlleva un ajuste individual de los canales de la imagen HSV para su correcto filtrado. En el caso de ajustar los parámetros para un tono de piel caucásico, limitaría el uso de la aplicación a personas pertenecientes a otras etnias cuyo color de piel diverge del ajustado para el filtrado. Esta limitación sería inaceptable en el contexto multinacional en el que se convive hoy en día.

Otra gran desventaja es que no se contempla la problemática que surge a la hora de detectar más de una cara en el campo de visión de la cámara. En este caso CAMShift no sería capaz de localizar el centro del rectángulo contenedor, ya que ahora la distribución de píxeles con valores aceptados estaría dispersado en diversos bloques, haciendo un seguimiento en concreto casi imposible.

Lo mismo ocurre cuando se inicializa CAMShift con una cara, que a lo largo del proceso de seguimiento desaparece del campo de visión y es sustituido por una mano por ejemplo. CAMShift localizaría el centro del rectángulo en la mano, ya que dispone de las mismas tonalidades de piel que la cabeza. De esta manera se podría llegar a realizar el seguimiento de cualquier parte del cuerpo desnudo, lo que resultaría inadecuado para una aplicación de seguimiento facial. Para corregir esta desventaja de CAMShift, se puede aparejar el algoritmo con un detector de formas entrenado para reconocer características faciales. Esto supondría añadir una carga computacional más elevada dependiendo del número de características, mermando de esta manera la gran ventaja inicial del CAMShift que era su gran velocidad.

La misma problemática ocurre en entornos con fondos muy parecidos a la de la piel humana, dificultando extremadamente la localización correcta del centro del rectángulo de la cara a seguir. Tampoco se contempla la variación lumínica, incluyendo entornos con poca luz, tonalidades muy saturadas o a contraluz.

Compressive Tracking

Como visto anteriormente, el resultado de CAMShift [32] es muy sensible a cambios de luz y por lo tanto de la tonalidad a seguir. Esto se debe a que los valores son fijados al principio del algoritmo, manteniéndose constantes durante todo el proceso. Obviamente cualquier cambio lumínico en el entorno una vez inicializado el proceso, tendrá como resultado la pérdida del seguimiento, ya que los valores que CAMShift busca, ya no se encuentran en la imagen. De esta problemática surgen algoritmos de aprendizaje, capaces localizar el objetivo aunque hayan cambios lumínicos, de pose, distorsiones causadas por movimiento y oclusión.

En general se diferencian entre dos grandes grupos: aquellos que entrenan su modelo a partir de una base de datos previamente al lanzamiento del seguimiento, y aquellos otros que actualizan en tiempo real el modelo dependiendo de los cambios que se evalúan en la escena. Estos dos grupos se conocen como algoritmos *offline* y *online* respectivamente.

El resultado de los algoritmos *offline* es directamente proporcional al número de modelos con los que se ha entrenado el algoritmo, y se requiere de una base de datos con modelos muy definidos y lo suficientemente amplia para garantizar un seguimiento bajo condiciones variables. Para la aplicación a desarrollar en esta tesina, este requerimiento es inviable, ya que no hay manera de conocer y entrenar el algoritmo con modelos del usuario previamente a que éste lo use. De esta manera se precisa de un algoritmo de aprendizaje *online*, capaz de actualizar

(dependiendo de un factor de aprendizaje) el modelo del usuario en tiempo real y conforme se produzcan los cambios en la escena.

Estos algoritmos suelen presentar un drift, que se traduce como un error a la hora de actualizar el modelo de apariencia y que conlleva a la creación de un modelo distorsionado que puede llegar a ser muy diferente al original que se estaba siguiendo. La mayoría de algoritmos *online* corrigen este error mediante una supervisión parcial del proceso de actualización del modelo, seleccionando las muestras buenas y evitando las malas.

El algoritmo utilizado para esta tesina en cambio, actualiza el modelo de características extraídas del dominio comprimido, no precisando ningún tipo de supervisión a la hora de ajustar el modelo. Estas características son extraídas a partir de una imagen reducida en dimensionalidad, permitiendo una separación entre fondo y cara aplicando un simple clasificador Bayesiano. Además, estas características son de carácter aleatorio y se extraen de igual manera entre fondo y cara asignándoles pesos positivos y negativos respectivamente. El funcionamiento del algoritmo se describe a continuación.

Igual que pasaba con CAMShift, se precisa de una cara ya detectada para inicializar el seguimiento. Esta cara es de nuevo la salida directa de un detector de caras lanzado previamente. Para actualizar el clasificador se extraen muestras aleatorias tanto positivas pertenecientes a la cara, como otras negativas de puntos lejanos a la cara.

Cada una de estas características $\mathbf{z} \in \mathbb{R}^{w \times h}$ es convolucionada con un conjunto de filtros rectangulares h de dimensiones i y j tal que:

$$h_{i,j}(x,y) = \begin{cases} 1, & 1 \leq x \leq i, 1 \leq y \leq j \\ 0, & \text{en caso contrario} \end{cases} \quad (7)$$

Cada imagen filtrada es representada mediante un vector columna que se concatenan en otro vector multi-escala $\mathbf{x} \in \mathbb{R}^m$, donde $m = (wh)^2$, comprendiendo valores entre 10^6 a 10^8 . A continuación se proyectan los valores de \mathbf{x} sobre otro vector $\mathbf{v} \in \mathbb{R}^n$, resultado de aplicar una matriz de medidas aleatoria, dispersa e independiente R . Esta matriz está definida por:

$$r_{ij} = \sqrt{s} * \begin{cases} 1 & \text{con probabilidad } 1/2s \\ 0 & \text{con probabilidad } 1 - 1/s \\ -1 & \text{con probabilidad } 1/2s \end{cases} \quad \text{con } s = m/4 \quad (8)$$

Esta matriz es computada únicamente al principio del proceso y se mantiene constante. De esta manera solo se necesita almacenar las posiciones no nulas de cada fila de R y las posiciones de aquellos filtros rectangulares de la imagen de entrada que corresponden con estas posiciones. A continuación se puede procesar el vector \mathbf{v} utilizando la matriz aleatoria y dispersa R para extraer las características rectangulares que se computan usando la técnica de la imagen integral descrita por Viola-Jones [3].

De esta manera se obtiene un vector v de baja dimensionalidad resultado de convolucionar un filtro rectangular con la intensidad de un punto específico de la imagen de entrada:

$$v_i = \sum_j r_{ij} x_j \quad (9)$$

Cada muestra $z \in \mathbb{R}^m$ tiene por lo tanto su representación de baja dimensionalidad a través del vector $v = (v_1, \dots, v_n)^T \in \mathbb{R}^n$ con $m \gg n$ a partir del cual se construye el clasificador Bayesiano. Para ello se asigna a cada muestra una etiqueta que indica si se trata de una muestra positiva (perteneciente a la cara) o negativa (no perteneciente a la cara).

$$H(v) = \log \left(\frac{\prod_{i=1}^n p(v_i|y=1)p(y=1)}{\prod_{i=1}^n p(v_i|y=0)p(y=0)} \right) \quad (10)$$

Las distribuciones $p(v_i|y=1)$ y $p(v_i|y=0)$ son de tipo gaussiano definido por los cuatro parámetros $(\mu_i^1, \sigma_i^1, \mu_i^0, \sigma_i^0)$ que son actualizados mediante el factor de aprendizaje λ . Estos parámetros están definidos según:

$$\sigma^1 = \sqrt{\frac{1}{n} \sum_{k=0}^{n-1} (v_i(k) - \mu^1)^2} \quad (11)$$

$$\mu^1 = \frac{1}{n} \sum_{k=0}^{n-1} v_i(k) \quad (12)$$

Esta técnica de proyección aleatoria y dimensionalidad baja presenta una de sus principales ventajas frente a métodos tradicionales como PCA [33] (Principal component analysis) en que no precisa de una continua actualización de su modelo de apariencia. Además, estos métodos son especialmente sensibles a oclusiones parciales ya que su matriz de medidas no es independiente de los datos. La robustez frente a ruido en el procesamiento se consigue mediante la extracción de características aleatorias a diferentes escalas.

Los problemas de ambigüedad del sistema a la hora de clasificar correctamente las características extraídas de la imagen, se minimizan gracias a que siempre se asegura tener una característica positiva más correcta que las demás basándose en su probabilidad.

Para la configuración del sistema se ha optimizado todos los parámetros para permitir la ejecución simultánea de otra aplicación gráfica. De esta manera se ha rebajado el número de características de 50 a 35, debido a que la propia funcionalidad de la aplicación garantiza de que el jugador va a estar presente la mayor parte del tiempo de ejecución. Después de recibir la primera información desde la cámara web y una exitosa detección facial, se procede a iniciar el algoritmo mediante su *init*. Para realizar esta inicialización se ejecutan los siguiente métodos en el siguiente orden:

1. *HaarFeature*
2. *sampleRect* (tanto positivo como negativo)
3. *integrallImage*
4. *getFeatureValue* (tanto positivo como negativo)
5. *classifierUpdate* (tanto positivo como negativo)

El código del programa inicializa primero esta cantidad definida de características Haar con las dimensiones de una cara detectada previamente mediante el método *HaarFeature*. Estas características se crean de manera aleatoria dentro del espacio definido por la dimensión de la cara y se componen de un número aleatorio y par de cuadrados entre 2 y 4. Por cada característica se almacena esos cuadrados formando un vector de vectores.

Después estas coordenadas de las características Haar son computadas según si pertenecen a la cara o al fondo, generando dos vectores de rectángulos pertenecientes a las muestras positivas y negativas usando el método *sampleRect*.

Tras calcular la imagen integral del frame actual, se procede a calcular el valor de dichas muestras en una matriz de 35 características por cada vector de muestras. Esta operación se realiza tanto para las muestras negativas como las positivas mediante *getFeatureValue*. Finalmente se reajustan los parámetros de las distribuciones según las ecuaciones 11 y 12 usando la desviación típica aplicada en el método *classifierUpdate*. Una vez inicializado el sistema en este orden, se empieza a procesar las imágenes nuevas aplicando las características positivas y negativas extraídas de la cara usando una estimación para la nueva posición del usuario con este orden:

1. *sampleRect* (para la estimación)
2. *integrallImage*
3. *getFeatureValue*
4. *radioClassifier*

Una vez se ha estimado la nueva posición correctamente, se procede a actualizar de nuevo los parámetros de las distribuciones σ y μ tal y como se realizó en el proceso de inicialización, asignando la salida de la estimación como posición actual.

5. *sampleRect* (tanto positivo como negativo)
6. *getFeatureValue* (tanto positivo como negativo)
7. *classifierUpdate* (tanto positivo como negativo)

Este orden de ejecución representa el bucle principal del programa y cada nueva imagen es procesada según este esquema.

Mediante una nueva llamada al método *sampleRect* en la siguiente pasada del sistema, es decir con la siguiente imagen posterior a la inicialización del algoritmo, se procede a estimar la posición de la cara a partir de una nueva clasificación de las características Haar. Esta clasificación ocurre respecto la nueva imagen, la posición anterior y mediante una ventana de búsqueda y difiere de esta manera del método

sampleRect que se aplica para realizar la inicialización y actualización. Se vuelve a calcular la imagen integral del frame actual para poder realizar una nueva asignación de pesos usando de nuevo el método *getFeatureValue* para actualizar los parámetros de la distribución gaussiana con *classifierUpdate*.

El algoritmo resultante es el descrito por *Zhang et al.* [20] y es robusto frente a oclusiones parciales, rotaciones del objetivo, cambios de escala y condiciones lumínicas desfavorables. Además el sistema presentado es restrictivo permitiendo un seguimiento de únicamente la persona con la que se inicializó el algoritmo, siendo de esta manera también un algoritmo de reconocimiento.

2.6. El filtro simple de Kalman

El filtro de Kalman [31] es un algoritmo recursivo de procesamiento de datos capaz de predecir a partir de todas las medidas disponibles las variables en cuestión. Esta predicción pretende ser más precisa que la medida real con ruido, precisamente por basarse en las medidas discretas anteriores a dicha predicción. La recursividad se basa en la covarianza utilizada para realizar la siguiente predicción, asignando mayores pesos a aquellos valores que más se aproximan a los reales.

El filtro permite predicciones muy precisas en sistemas dinámicos lineales, aun en casos en los que el objetivo este solo parcialmente visible e incluso completamente oculto.

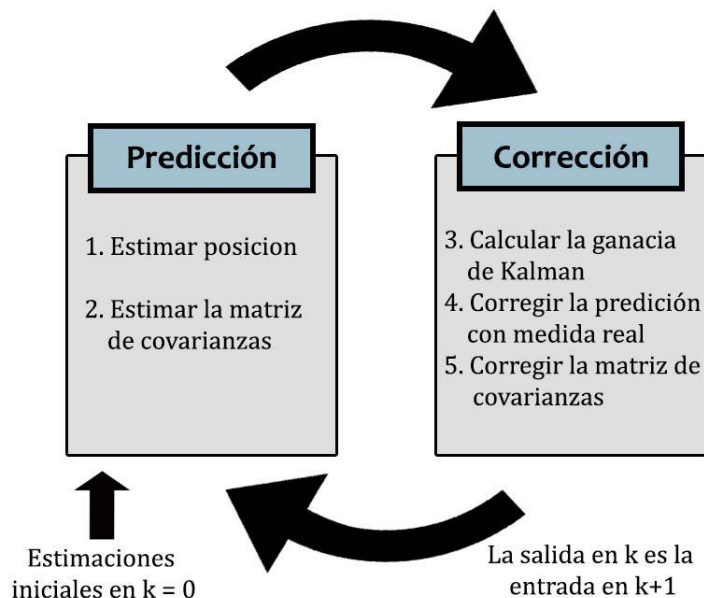


Figura 11 Diagrama de proceso del filtro simple de Kalman

Principalmente se distingue entre dos pasos dentro del propio filtro (figura 11): la predicción a partir de mediadas anteriores y la corrección de dicha predicción usando medidas actuales si existen.

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1|k-1} \quad (13)$$

$$P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_k \quad (14)$$

La predicción a priori $\hat{x}_{k|k-1}$ (13) del instante k respecto al instante anterior, es de esta manera función de la matriz de transiciones y la predicción corregida en la iteración anterior $\hat{x}_{k-1|k-1}$. La matriz de transiciones F_{k-1} define el modelo físico del sistema y establece como cambia el sistema a través del tiempo. Se trata de una matriz de dimensiones $n \times n$, compuesta obligatoriamente por ecuaciones lineales.

La matriz de covarianza para la predicción denominada $P_{k|k-1}$ (14), también es una matriz de dimensiones $n \times n$, en cuya diagonal se encuentran las varianzas de los elementos de la predicción $\hat{x}_{k|k-1}$. Cuanto más grande sea el valor de la varianza, más inexacta es la predicción. La matriz de covarianza se actualiza en cada iteración, convergiendo en el mejor de los casos a los valores óptimos para la predicción.

En el caso del seguimiento facial, una convergencia demasiado rápida tiene como resultado una gran dispersión entre el punto medido real y la correcta predicción de dicho punto por parte del filtro. Por ello es necesario introducir un ruido Q_k añadido a la matriz de covarianzas en cada iteración, denominado ruido de proceso o error de covarianza del proceso.

El segundo paso es la corrección de la predicción obtenida, a partir las medidas reales z_k correspondientes al instante actual. Para ello se calcula la innovación o residual de la medida \hat{y}_k (15), que representa la diferencia entre la medida real obtenida y su predicción. La matriz de medida H_k con dimensiones $m \times n$ simplemente indica que valores de todos los datos son las medidas de interés.

De una manera similar se calcula la innovación de la covarianza S_k (16), que establece un relación entre el error de covarianza R_k real de los sensores que realizan la medida, y la covarianza predicha $P_{k|k-1}$. La matriz de covarianza de las medidas R_k es una matriz de dimensiones $m \times m$, y es constante en todo el proceso debido a que el filtro no puede cambiar la precisión y el error de los sensores utilizados. Estimar bien los valores de los errores de covarianza del proceso Q_k y de la medida R_k , ha sido objetivo de número de estudios, ya que su correcta definición tiene repercusión directa sobre el funcionamiento del filtro.

$$\hat{y}_k = z_k - H_k\hat{x}_{k|k-1} \quad (15)$$

$$S_k = H_kP_{k|k-1}H_k^T + R_k \quad (16)$$

$$K_k = P_{k|k-1}H_k^T S_k^{-1} \quad (17)$$

A partir del residual de la covarianza S_k se calcula la ganancia de Kalman K_k (17). Esta ganancia define cuanto de la diferencia entre la medida real y la predicha se añade a la predicción en la fase de corrección.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \hat{y}_k \quad (18)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (19)$$

Finalmente se obtiene el valor corregido $\hat{x}_{k|k}$ (18), sumando a la predicción a priori $\hat{x}_{k|k-1}$ obtenida en (13), la cantidad de innovación \hat{y}_k ajustada por la ganancia K_k . La matriz de covarianza corregida $P_{k|k}$ (19) es función de la misma ganancia y la covarianza predicha en (14).

Aunque OpenCV incluya ya un método para generar un filtro Kalman, se ha optado por realizar un filtro propio debido a que el Wrapper utilizado no incluye un script estable del filtro. Ni se podía realizar ninguna predicción ni tampoco una corrección debido a que no se asignaba bien las variables.

La matriz de transiciones se ha definido para un sistema de 6 parámetros dinámicos: tres puntos para la localización tridimensional y tres más para sus correspondientes velocidades. Mediante el filtro de Kalman se pretende estimar tanto la posición x e y, así como la profundidad en el eje z originando la siguiente matriz 6x6:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz de medidas de tamaño 3x6 describe las medidas relevantes para el filtrado y únicamente incluye al punto tridimensional de la cara.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

La matriz de covarianza de las medidas R que se añade a la innovación de la covarianza se mantiene constante durante todo el filtrado y se define como:

$$\begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$$

La matriz de covarianzas es de 6x6 y es una matriz diagonal cuyos valores describen el grado de certeza que se tiene acerca del valor.

```

500  0  0  0  0  0
0    500 0  0  0  0
0    0   500 0  0  0
0    0   0   500 0  0
0    0   0   0   500 0
0    0   0   0   0   500
    
```

Para la fase de corrección se precisa de una cara encontrada previamente, por lo que primeramente se asignara la nueva posición real de la cara r a la variable *trackingwindow* que se usará para realizar la nueva predicción y corrección de la covarianza.

```

trackingWindow = r;
measurement.mSet (0, 0, trackingWindow.X);
measurement.mSet (1, 1, trackingWindow.Y);
measurement.mSet (2, 2, trackingWindow.Width);

xest = F * x; //kalman predict
FT = F.Transpose ();
Pest = F * P * FT + Q; //estimated covariance
y = measurement - (H * xest); //measurement residual
HT = H.Transpose ();
S = (H * Pest * HT) + R; //innovation covariance
S.Invert (SI);
K = Pest * HT * SI; //kalman gain
x = xest + (K * y); //kalman correct
P = (eye - (K * H)) * Pest; //corrected covariance
    
```

En este modelo del filtro de Kalman, $xest$ es la predicción obtenida a partir de la ecuación 13. De igual manera se obtiene la covarianza estimada $Pest$, el residual de la medida y la innovación de la covarianza a partir de las ecuaciones 14, 15 y 16 respectivamente. Cabe destacar la importancia de añadir un ruido artificial R a dicha innovación. Si se omite, el sistema converge demasiado rápido, encontrando la matriz de covarianzas perfecta para la predicción actual. Esta matriz se mantiene constante durante el resto del procesamiento, impidiendo de esta manera un reajuste necesario ante un nuevo movimiento del objetivo a seguir.

Finalmente se obtiene la ganancia de Kalman como producto entre la covarianza estimada, la traspuesta de la matriz de medidas y la matriz invertida de la innovación de la covarianza según la ecuación 17.

La corrección de los valores x y P depende de la ganancia previamente calculada y corresponde a las ecuaciones 18 y 19 respectivamente.

```

trackingWindow.X = (int)x.mGet (0, 0);
trackingWindow.Y = (int)x.mGet (1, 1);
trackingWindow.Width = (int)x.mGet (2, 2);
    
```



```
scale2 = 1f / ((float)trackingWindow.Width * 2.0f / 100f);
```

```
facepos = new Vector3((trackingWindow.X + trackingWindow.Width / 2.0f) / CAPTURE_WIDTH, (trackingW  
indow.Y + trackingWindow.Height / 2.0f) / CAPTURE_HEIGHT,scale2);
```

La corrección de la predicción únicamente se lleva a cabo si se dispone de una medida real, pasando directamente el valor y la matriz de covarianzas de la predicción como $\hat{x}_{k|k}$ y $P_{k|k}$ para la siguiente iteración.

Se asignan los nuevos valores corregidos al *trackingwindow* actual y se establece la posición definitiva del seguimiento *facepos* como centro del rectángulo que se ha trazado para contener a la cara.

En el caso en el cual no se pueda recurrir a una medida real para realizar la corrección de la predicción, únicamente se calculará la predicción de la posición y la matriz de covarianzas.

```
//estimated position
```

```
x = F * x;
```

```
FT = F.Transpose ();
```

```
//estimated covariance
```

```
P = F * P * FT + Q;
```

```
trackingWindow.X = (int)x.mGet (0, 0);
```

```
trackingWindow.Y = (int)x.mGet (1, 1);
```

```
trackingWindow.Width = (int)x.mGet (2, 2);
```

```
scale2 = 1f / ((float)trackingWindow.Width * 2.0f / 100f);
```

```
facepos = new Vector3((trackingWindow.X + trackingWindow.Width / 2.0f) / CAPTURE_WIDTH, (trackingW  
indow.Y + trackingWindow.Width / 2.0f) / CAPTURE_HEIGHT,scale2);
```

De esta manera se consigue que el filtro simple de Kalman vaya alternando sus estimaciones con aquellas que ha corregido mediante una medida real. Aquí reside la gran ventaja del filtrado: aunque no se disponga de una medida real, el sistema es capaz de predecir por la dirección y la velocidad que tenía el objetivo en la imagen anterior, una posición aproximada actual. Claro esta, que conforme más predicciones sin corrección se ejecuten seguidamente, más divergencia existirá entre la medida real y la predicha por el filtro.

2.7. Corrección lumínica MSR y MSRCR

Uno de los mayores problemas en la detección y el seguimiento facial es el continuo cambio de las condiciones lumínicas que se pueden presentar en la escena. El seguimiento frecuentemente queda interrumpido si la escena presenta de repente un exceso de brillo y la tonalidad de la imagen varía. Esto puede afectar de manera especialmente negativa a algoritmos de seguimiento basados justamente en la extracción y el filtrado de esta característica como lo es CAMShift [32].

Por ello resulta lógico desarrollar un algoritmo capaz de reducir e incluso anular estos cambios bruscos de brillo, color o saturación. La teoría Retinex [18] intenta justamente corregir estas diferencias que se pueden producir en la imagen digital respecto a la realidad, y adaptar el proceso a la corrección automática que tiene lugar en la visión humana. El algoritmo presentado por *Rahman et al.* [19] se fundamenta en este principio, incrementando el contraste local especialmente en imágenes con grandes rangos dinámicos que contienen tanto zonas muy oscuras como fuentes de gran brillo. También se pretende aumentar la constancia del color y reducir los efectos de la relación brillo/color.

$$R_i(x_1, x_2) = \sum_{k=1}^K W_k \{ \log I_i(x_1, x_2) - \log [F_k(x_1, x_2) * I_i(x_1, x_2)] \} \quad (20)$$

Mediante el uso de varias envolventes Gaussianas de diferentes anchos se asegura una compresión del rango dinámico y un rendimiento en el ajuste tonal adecuado. La forma final del algoritmo multi-escala Retinex (MSR) está descrito en la ecuación 20, siendo i el canal actual, $*$ el símbolo de la convolución, (x_1, x_2) la coordenada del pixel según un sistema cartesiano, I_i la imagen actual, R_i la salida del proceso, W_k el peso asociado con la función de la envolvente F_k y K el número de funciones envolventes. La función de la envolvente viene definida por:

$$F_k(x_1, x_2) = \kappa \exp[-(x_1^2 + x_2^2)/\sigma_k^2] \quad (21)$$

σ_k es la desviación típica de las envolventes Gaussianas y su magnitud controla la extensión de la envolvente. Toda la función es normalizada según κ , la inversa del sumatorio de la función de la envolvente gaussiana en un pixel.

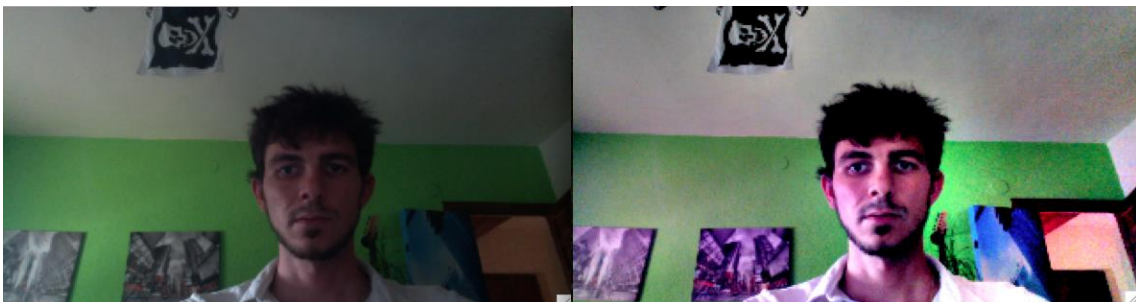


Figura 12 Resultado tras aplicar el filtrado Retinex multi-escala en una escena con iluminación baja.

Mediante el uso de tan solo tres diferentes escalas ya se obtienen resultados que corrigen ampliamente el rango dinámico y de tono. Únicamente en zonas de la imagen que se componen por áreas grandes monocromáticas se observa una

desaturación, originando un efecto que refuerza tonos grises en vez de su color original. Para corregir el impacto de este efecto no deseado se ha añadido un algoritmo de restauración de color.

Este algoritmo añadido permite una restauración de color de las zonas afectadas por la falta de saturación. Además se pretende preservar la constancia del tono de color en las zonas uniformes, aunque debido a que la visión humana tampoco percibe con total constancia los colores, una pequeña variación no afecta al resultado percibido. El factor de restauración α describe la corrección mediante la siguiente transformada:

$$\alpha_i(x_1, x_2) = f \left[\frac{I_i(x_1, x_2)}{\sum_{n=1}^N I_n(x_1, x_2)} \right] \quad (22)$$

donde $\alpha_k(x_1, x_2)$ es el coeficiente de restauración en la banda espectral i de todas la N bandas existentes. El filtrado completo de Retinex multi-escala con corrección de color MSRCR queda de la siguiente manera:

$$R_i(x_1, x_2) = \alpha_i(x_1, x_2) \sum_{k=1}^K W_k \{ \log I_i(x_1, x_2) - \log [F_k(x_1, x_2) * I_i(x_1, x_2)] \} \quad (23)$$

El coste computacional del filtrado Retinex es costoso debido a la redundancia usada a la hora de aplicar el filtro a las diferentes escalas. Su aplicación queda limitada de esta manera únicamente a sistemas de detección facial de gran velocidad que además requieran de este tipo de corrección lumínica posterior a la obtención de una imagen para mejorar su rendimiento. Por ello se ha evaluado el funcionamiento conjuntamente con el algoritmo CAMShift, debido a que la técnica de Compressive tracking presenta ya de por si una gran robustez frente a estos cambios de brillo, saturación y contraste.

2.8. Holografía digital

Otra manera de aumentar la inmersión del jugador en los mundos digitales, es el uso de dispositivos capaces de reproducir los contenidos en más de un plano de proyección. De esta manera se pretende simular la tridimensionalidad real de los objetos en un dispositivo que en un principio solo es plano.

A lo largo de la historia se han desarrollado numerosas técnicas para dotar de más profundidad a las imágenes planas, empezando en la holografía clásica hasta llegar a pantallas 3D [34] de nueva generación.

La idea principal detrás de la ilusión de la tridimensionalidad empleada en dispositivos modernos, es alimentar cada ojo con una imagen distinta. Debido a que el ojo humano percibe la luz a través de un único punto, el cerebro ha aprendido a estimar el tamaño de los objetos a partir del ángulo que forman sus extremidades respecto a la pupila. Por esta razón cada ojo únicamente ha de recibir la imagen correspondiente a su punto de vista, que, debido a la distancia que hay entre ellos, será ligeramente distinta para cada ojo.

Algunos de estos dispositivos precisan de gafas especiales que controlan que imagen recibe cada ojo, bien por oclusión activa o un filtrado mediante polarización. Existen alternativas como la auto-estereoscopia [30] que no obligan al usuario a llevar gafas para percibir el efecto de tridimensionalidad, pero fuerzan a que el individuo se encuentre en uno de los marcados para poder ver el efecto. Si el usuario se desplaza ligeramente, únicamente se verá una imagen difuminada, resultado de mezclar ambas imágenes individuales.

Otra alternativa de generar más planos es simular la profundidad hacia dentro de la pantalla de manera completamente virtual. Esta técnica se conoce como paralaje.

Efecto de Paralaje

Debido a la predicción que el ojo humano realiza, los objetos cercanos al ojo tienden a deformarse en su tamaño, aparentando ser más grandes de lo que realmente son. El efecto opuesto se observa en objetos lejanos al observador que aparentan ser mucho más pequeños.

Esto es el principio básico en el que se fundamenta el efecto conocido como paralaje. El Paralaje describe el aparente movimiento que un objeto adquiere cuando el observador se mueve alrededor suya. Muchos juegos 2D utilizan este efecto para dotar de más profundidad a sus niveles, utilizando varias capas de profundidad cuya velocidad de movimiento se acelera conforme más cerca al espectador estén (figura 10).

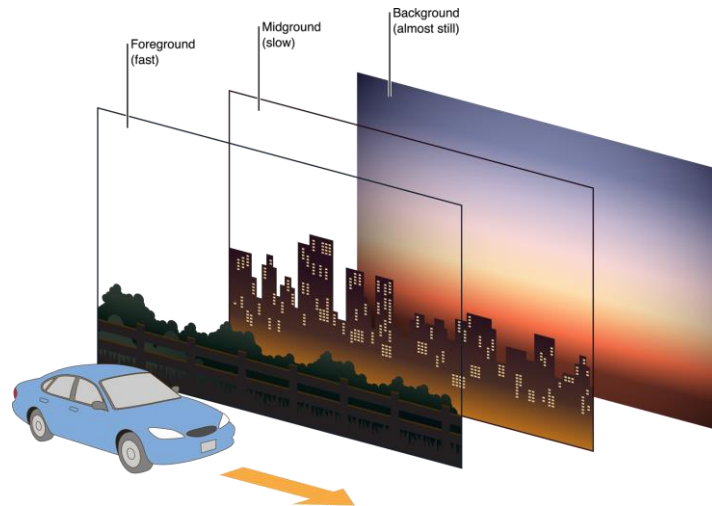


Figura 13 Conforme más se aleje el plano del coche, menor será su movimiento en dirección contraria al coche, consiguiendo de esta manera el conocido efecto de paralaje.

De igual manera se puede emplear el mismo principio para emular más profundidad respecto al punto de vista de un espectador. El movimiento ya no se percibirá respecto al coche como en la figura 13, sino a partir de la posición relativa del usuario respecto a la pantalla. Para ello se usan tanto acelerómetros y giroscopios, como algoritmos de detección facial para estimar la posición del espectador respecto a la pantalla en la que se reproduce el efecto.

Mediante la detección facial es fácil determinar el ángulo necesario para dotar de una profundidad virtual a la pantalla, que pasa de ser un plano a un espacio 3D en el que se desarrolla la acción del juego. Para conseguir el efecto de paralaje, es necesario que la cámara apunte al centro de la imagen y únicamente se gire formando un semicírculo a lo largo del eje horizontal y vertical según la posición del jugador manteniendo siempre la misma dirección de enfoque.

Esta posición es la salida del detector facial descrito previamente. El script es añadido directamente a la cámara de la escena, recibiendo desde el detector las posiciones del espectador. Debido a que el movimiento es circular, se multiplica la componente x e y de la posición por π , obteniendo los valores de α y β .

```
float alpha = facepos.y * Mathf.PI;
float beta = facepos.x * Mathf.PI;
float scale = facepos.z;
```

Se define el vector $pos(10 \cos \beta + 25, 10 \cos \alpha + 11, -10 \sin \beta - 40)$ para forzar la cámara a apuntar siempre al centro de la pantalla, ya que únicamente variará su posición en semicírculos. Es necesario multiplicar por 10 cada elemento del vector pos para limitar el movimiento semicircular tanto a lo largo del eje horizontal como del vertical. Este límite está definido por el punto en el cual el espectador abandona por completo el campo de visión de la cámara hacia todos los lados.

```
Vector3 pos;
pos.x = 10.0f * Mathf.Cos(beta) * (scale) + 25;
pos.y = 10.0f * Mathf.Cos(alpha) * (scale) + 11;
```

```
pos.z = -10.0f * Mathf.Sin(beta) * (scale)-40;
```

Finalmente se aplica el vector *pos* al movimiento de la cámara, orientando el objetivo a la dirección del escenario mediante el uso de `Quaternion.LookRotation`. Para suavizar el movimiento se ha aplicado además una interpolación esférica desde la posición previa al movimiento *aux* y la nueva posición calculada en el paso anterior *pos*. Por último se guarda la nueva posición en el vector auxiliar para el siguiente frame.

```
transform.position = pos;  
Vector3 relativePos = new Vector3(-pos.x+25,-pos.y+11,55);  
Quaternion rotation = Quaternion.LookRotation(relativePos);  
transform.rotation = Quaternion.Slerp(aux,rotation,Time.deltaTime * 600f);  
aux = rotation;
```

3. Diseño y especificación

Para la realización de este trabajo se ha optado por utilizar uno de los diversos clasificadores *Haar* en cascada incluidas con la librería OpenCV [8]. Se trata de la técnica descrita por *Viola y Jones* [3], ya que se obtiene los resultados más rápidos, con menos carga computacional y una precisión aceptable. Se ha descartado el uso de API's y SDK's ya que la mayoría son de pago y no permiten un control total sobre los parámetros ajustables necesarios para la detección. Debido a que la construcción del clasificador es individual, la nueva problemática reside en la dificultad para poder utilizarlo desde Unity [28].

Los archivos xml que se incluyen con OpenCV son clasificadores ya entrenados con diferentes conjuntos de caras y no caras. Para averiguar cual de ellos proporciona el mejor resultado, se ha realizado un test con un conjunto de 200 imágenes cogidas de *Instagram* y *Twitpic*, de las cuales 150 son caras y las restantes no caras [5].

- `haarcascade_frontalface_alt.xml` → 45% (67/150) caras detectadas
- `haarcascade_frontalface_alt_tree.xml` → 29% (44/150) caras detectadas
- `haarcascade_frontalface_alt2.xml` → 45% (68/150) caras detectadas
- `haarcascade_frontalface_default.xml` → 55% (82/150) caras detectadas
- `haarcascade_profileface.xml` → 15% (23/150) caras detectadas

Entre los cinco diferentes clasificadores de OpenCV, `alt.xml`, `alt2.xml` y `default.xml` han detectado además un total de cuatro falsos positivos cada uno, entre las 50 imágenes con no-caras. Visto el resultado se ha optado por utilizar el clasificador `haarcascade_frontalface_default.xml` con una tasa de acierto de 55%. Se consideró la unión del clasificador de cara frontal y el que detecta caras de perfil, pero debido a que se trata del desarrollo de un sistema alternativo de control en un videojuego, carece de sentido detectar la cara si el jugador no está mirando la pantalla, es decir se encuentra de perfil a ella.

Una vez hecha la detección en el primer frame, el rectángulo contenedor de la cara es pasado al siguiente modulo: el seguimiento mediante *Compressive tracking* [20]. Este algoritmo es capaz de realizar el seguimiento incluso en situaciones con cambios bruscos en la iluminación, contextos con baja iluminación, oclusiones parciales de la cabeza a seguir y variaciones en la pose. Esta robustez se obtiene mediante una selección aleatoria de características *Haar* que se distribuyen también de manera aleatoria entre la cara encontrada previamente y cualquier punto de la imagen no perteneciente a la cara, es decir el fondo. De esta manera se define un modelo capaz de separar el fondo de la cara en cuestión. Este modelo se actualiza dependiendo de un factor de aprendizaje, permitiendo su adaptación a entornos variables.

El propio proceso de captación de imagen utilizando la librería OpenCV para acceder a la WebCam, es costoso y requiere aproximadamente 0.2 segundos, lo que supone un rendimiento de unos 15 -20 frames por segundo en la escena de prueba. Esta velocidad es inviable para su incorporación en un videojuego que además supondrá una carga grafica y computacional adicional al propio seguimiento. Por

este motivo se ha optado por crear un hilo de ejecución paralelo al programa principal, que se encarga de generar imágenes mediante la WebCam.

La comparación del tiempo de ejecución por detección/seguimiento en cada frame revela que efectivamente realizar una detección al principio y aplicar algoritmos de seguimiento para los frames posteriores es más rápido:

- Viola-Jones en cada frame: $\pm 0.4\text{ms}$
- Viola-Jones + CamShift: $\pm 0.2\text{ms}$
- Viola-Jones + Compressive Tracking: $\pm 0.5\text{ms}$
- Viola-Jones + Compressive Tracking + Kalman + Multihilo: $\pm 0.2\text{ms}$

CAMShift [32] es rápido pero sensible a demasiados factores que aparecen con frecuencia en un seguimiento convencional. La combinación final de Compressive tracking y Kalman es mucho más robusta y junto a la ejecución multi-hilo, consigue tasas de frames altas y estables.

Con esta combinación final de Viola-Jones, Compressive tracking, filtrado Kalman y ejecución multi-hilo, el flujo del programa es el siguiente: Cada vez que se genera una nueva imagen, está es enviada al programa principal para su análisis, extrayéndose de ella la cara a seguir mediante Viola-Jones. Mientras se está a la espera de una nueva imagen, el bucle principal ejecuta una predicción de la posición facial mediante un filtrado de Kalman [31] hasta que se reciba un nuevo frame. En este instante se vuelve a procesar la imagen mediante Compressive Tracking y se actualiza el filtrado de Kalman con la nueva posición (Fig. 14).

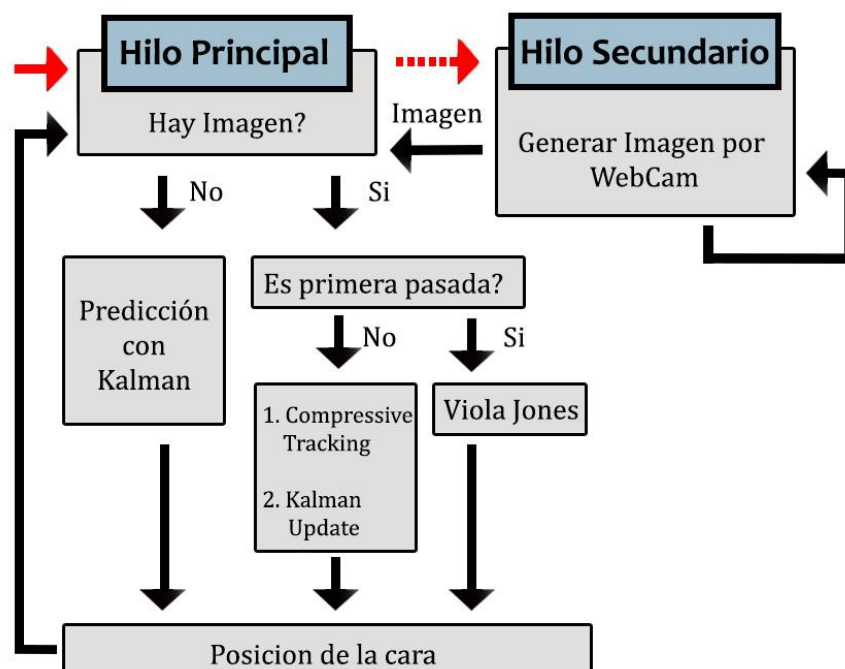


Figura 14 Hilo de ejecución del programa principal

La relación entre filtrado de Kalman y obtención de imágenes reales de la cámara es de 4 a 1, es decir cada 4 estimaciones de la posición se realiza una corrección

mediante Compressive tracking a partir de la posición real. La técnica de Compressive tracking sigue siendo muy costosa, por lo que esta relación de ejecución permite procesar en los intervalos de estimación mediante Kalman otros datos como renderizados y lógicas de juego.

Para la fase de desarrollo para la detección y el posterior seguimiento, se ha creado una pequeña escena en Unity3D compuesta por un modelo de un gorila ubicado en el interior de un cubo (figura 15). Mediante el seguimiento de la cara del usuario se permite el manejo de la cámara principal que girara alrededor del modelo del gorila. La interacción ocurre tanto a lo largo del eje horizontal como en el vertical.



Figura 15 Representación del juego final en el que se mueve la cámara mediante la detección facial.

4. Experimentación

Para demostrar la posible utilización del algoritmo desarrollado en un videojuego real, se ha optado por programar un pequeño juego propio. Este juego usará tanto la detección y el seguimiento facial como el efecto de paralaje aplicado a la cámara del sistema.

En esta demostración es de especial interés el número de frames por segundo que alcanza el juego junto al algoritmo, ya que éste ha de ser superior a 30 frames como mínimo durante todo el funcionamiento. En el caso de ser inferior, el jugador percibirá movimientos ralentizados y bruscos que podrían resultar ser molestos para la experiencia total del juego. El mini juego a realizar será un juego orientado a entrenar la capacidad reactiva del jugador y presentará elementos móviles que cruzaran la pantalla a determinadas velocidades y tienen que ser recogidos por el jugador para conseguir puntos.

El jugador controla las manos de un gorila mediante el uso de su cabeza, desplazando las manos únicamente a lo largo del eje horizontal en la parte inferior de la pantalla.



4.1. Objetos del juego

El jugador puede recoger diferentes objetos en el tiempo del juego. La partida tiene una duración de 1 minuto y el objetivo del juego es recoger objetos con efectos positivos para el jugador para aumentar la puntuación. Cada uno de los objetos es creado por un generador de objetos. Los generadores se sitúan en la parte superior de la pantalla donde producen con probabilidades individuales cada uno de los objetos. Además aparecen en posiciones aleatorias a lo largo del margen superior, creando así una experiencia de juego diferente para cada partida. Cada objeto es generado con una rotación aleatoria. En la parte inferior de la pantalla existe una barra que destruye todos los objetos que el jugador no recoge para disminuir la carga computacional y mantener así la tasa de frames lo más alto posible.



Plátano: si se recoge, el jugador recibe 10 puntos. Existen 4 diferentes generadores de plátanos que se colocan en puntos aleatorios fuera del campo de visión del jugador en el borde superior de la pantalla. La probabilidad de que se genere un plátano es de 90%.



Plátano triple: al tratarse de tres plátanos, el jugador recibe 30 puntos. Igual que el plátano simple, existen 4 generadores que producen estos plátanos triples con una probabilidad de 90%.



Piña: La piña aumenta el tamaño de las manos del gorila un 50% durante 10 segundos, permitiendo así recoger más objetos. Cuando se acaben los 10 segundos las manos vuelven a su tamaño original. La piña es generada con una probabilidad de 22.5%.



Plátano podrido: El plátano podrido resta al jugador la cantidad de 10 puntos y aparece con una probabilidad de 22.5%.



Coco: El coco se comporta como una pelota de un breakout [35] clásico, rebotando en las paredes laterales y en la superior. Cada vez que rebota en la mano se aumenta el multiplicador de puntos y su velocidad. En el caso de tocarse tres veces, el coco se rompe en dos mitades. Si estos trozos de coco se recogen, el jugador es recompensado con 200 puntos.



Reloj: el reloj amplía el tiempo de juego en 2 segundos, permitiendo al jugador recoger más objetos. Igual que el coco, su probabilidad es de 22.5%.



Letras: En total existen 5 diferentes letras que combinados crean la palabra "happy". Si el jugador consigue coleccionar las 5 diferentes letras entra en el modo especial del agujero negro. En este modo de juego, las manos del gorila desaparecen y el jugador pasa a controlar un agujero negro capaz de moverse a través de toda la pantalla.

Únicamente en este modo de juego aparecen serpientes que el jugador deberá esquivar. En el caso de que se toque una serpiente, el juego vuelve a su modo normal. En el modo agujero negro solamente caen plátanos y plátanos triples para aumentar rápidamente la puntuación del jugador.

4.2. Aspectos estéticos

Para dar al juego un acabado más profesional se han incluido algunos efectos visuales, sonidos y animaciones.

La percepción del anteriormente descrito efecto de paralaje se ha reforzado usando partículas que simulan el polvo en la jungla. Estas partículas caen de un generador próximo a la cámara.

Al principio de la partida se ha creado una cuenta atrás desde tres hasta 0 para dar tiempo al jugador de familiarizarse unos instantes con el control de las manos mediante los movimientos de su cabeza. Hasta que esta cuenta atrás no se haya acabado, no se inicializa la generación de objetos.

Para cada objeto se ha incluido efectos sonoros además de la música de fondo que emula el ambiente de la jungla. Además se ha incluido un contador en la esquina inferior derecha que muestra el tiempo actual de la partida. Cada vez que el jugador recoge un objeto se muestran en el lugar en el que se ha producido el contacto entre la mano y el objeto, la puntuación que aporta dicho objeto. En el caso

de que se recoja un reloj, se muestra el extra de dos segundos que se suma al tiempo restante total. Si el tiempo del juego es inferior a 5 segundos, el indicador empieza a parpadear en color rojo aumentando su tamaño. Además se reproduce un pitido por cada segundo alertando de esta manera al jugador de que el tiempo está a punto de agotarse.

En el panel de letras se percibe a todas las letras disponibles en el orden adecuado desactivados. Una vez que el jugador recoja una letra que aun no tenia, su campo correspondiente se iluminará. Al abandonar el modo agujero negro, todas las letras se vuelven a desactivar. El agujero negro es representado mediante una esfera negra con rotación propia a cuyo alrededor un shader genera un efecto de distorsión. Todos los objetos se ven atraídos por el propio agujero, modificando su aceleración conforme más cerca esté el objeto del centro del agujero.

El multiplicador cambia de color conforme se aumente su valor, pasando de blanco a amarillo a partir de 4, y rojo a partir de 6. Si se alcanza un multiplicador mayor o igual a seis se enciende una animación de fuego detrás del multiplicador. El multiplicador se reinicia a uno si un coco abandona el campo de la partida por la parte inferior.

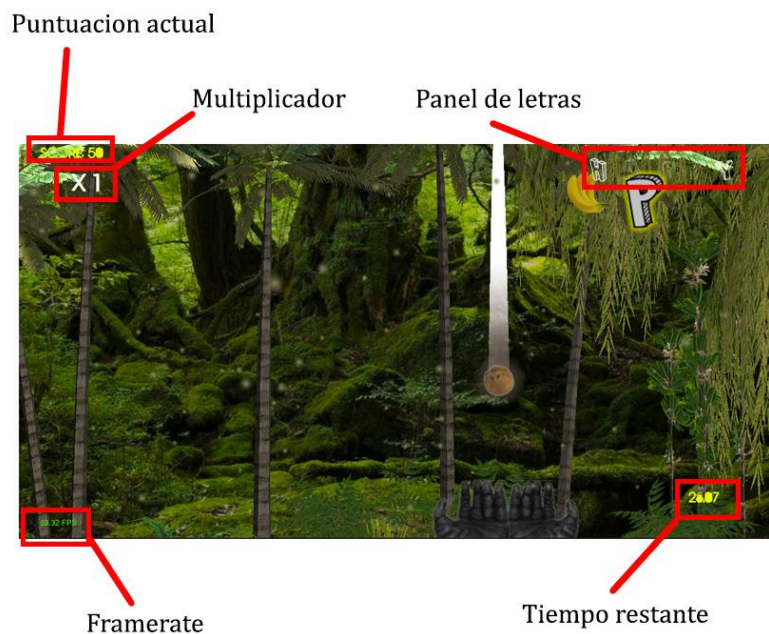


Figura 16 Pantalla del juego final en funcionamiento

También se ha creado un menú principal en el que el jugador puede elegir entre el modo demostración y el llamado arcade que contiene al mini juego presentado. La demostración es el modo presentado previamente y contiene la escena con el modelo del gorila en la cual se puede mover la cámara mediante el uso de la cabeza (figura 16). Además se ha creado un menú final de juego que aparece cuando haya pasado el tiempo del mini juego, ofreciendo al jugador su puntuación actual y su mejor puntuación alcanzada hasta el momento. También se le da la posibilidad de reiniciar la partida o volver al menú principal.



Figura 17 Menú principal del inicio del juego a la derecha y a la izquierda el menú de final de partida, que permite un reinicio o un abandono de la partida actual.

La aplicación final se ha compilado para sistemas Macintosh y se ha incluido un script que instala automáticamente el wrapper OpenCVSharp necesario para la ejecución en las librerías del sistema al inicializar la aplicación. El uso de la tecnología en dispositivos móviles se ha descartado, debido a que el movimiento de la cabeza puede ser emulado girando y inclinando el propio dispositivo, por lo cual el propio seguimiento facial carecería de sentido. En este caso sería muchísimo más práctico acceder directamente a los datos del giroscopio y acelerómetro que la mayoría de dispositivos móviles incluyen ya de serie.

Para una información más detallada sobre los contenidos creados para el juego véase también el anexo I.

4.3. Jugabilidad y evaluación

El juego creado para esta tesina demuestra con éxito de que el algoritmo puede funcionar conjuntamente con la carga grafica y computacional que supone un videojuego de tipo casual. En ningún momento se sobrepasa el limite de 30 frames necesarios para garantizar una experiencia de juego suave y fluida. La tasa de frames normal se sitúa entre 52 y 58 con un procesador Intel i7 a 2.8 GHz con grafica integrada Intel HD 3000. Todas las pruebas y evaluaciones se han realizado usando únicamente esta máquina con dichas características. Se han realizado cuatro pruebas: la primera usando la configuración final compuesta por Compressive tracking [20], Kalman [31] y multi-hilo, después usando únicamente CAMShift [32], una otra configuración usando CAMShift, Kalman y multi-hilo y finalmente CAMShift, Kalman, multi-hilo y filtrado Retinex multi-escala.

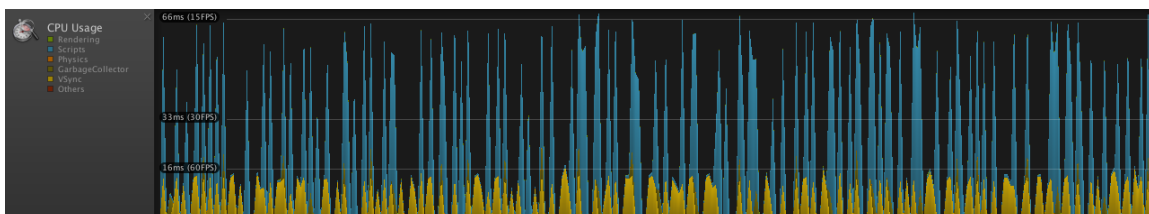


Figura 18 Profiler del juego en tiempo real usando la técnica de compressive tracking más filtrado Kalman: en azul se distingue claramente la carga computacional del script de detección facial. Cada vez que la cámara dispone de una nueva imagen se puede observar un pico en la carga. Entre los picos se observan las estimaciones por Kalman de mucha menor carga. En amarillo se distingue la función VSync que limita la tasa de refresco de la pantalla para suavizar el juego.

El procesamiento de la detección y el seguimiento facial fiable y con suficiente resolución para garantizar un seguimiento fluido y robusto frente a ruido como oclusiones parciales y otras caras, es computacionalmente costoso. Gracias a la ejecución multi-hilo que separa la cámara del hilo principal se consigue un algoritmo fiable para su uso en otras aplicaciones. La estimación de la posición mediante el filtrado de Kalman que se ejecuta en los instantes en los que no se dispone de una imagen actual procedente de la cámara, posibilita tasas de frames mayores (figura 18).

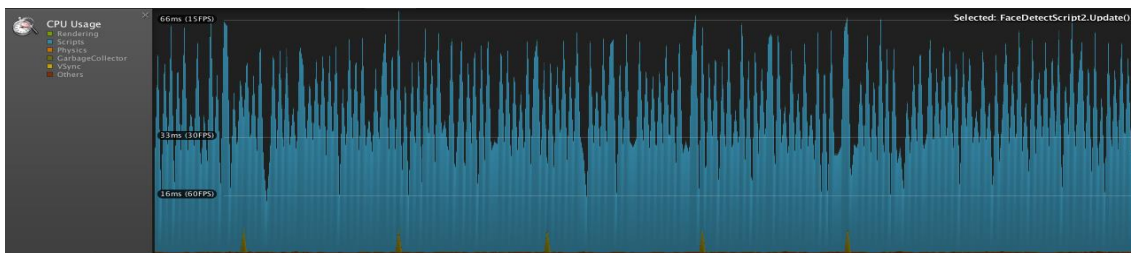


Figura 19 Profiler del juego utilizando la técnica de CAMShift: En contraste al profiler del compressive tracking, la carga computacional del algoritmo de seguimiento es mucho más constante.

Con el algoritmo de seguimiento CAMShift, la carga computacional es mucho más constante entre 0 y 33ms, además de que la experiencia del juego no es fluida, debido a los constantes reajustes de la ventana que contiene la cara para el seguimiento (figura 19). Únicamente usando CAMShift para en el procesamiento, la tasa de frames es muy baja, rondando los 20fps. Esto es debido a que todo el sistema se queda a la espera de la siguiente imagen procedente de la WebCam para seguir el funcionamiento. De ahí la gran importancia de la solución multi-hilo.

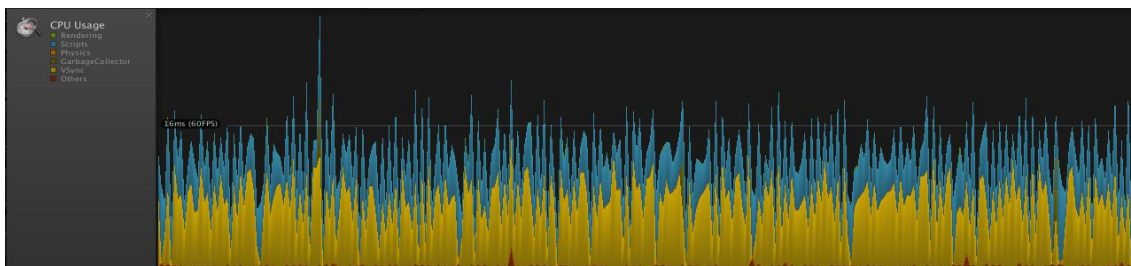


Figura 20 Profiler del juego utilizando CAMShift, Kalman y multi-hilo: La carga por el hilo de ejecución extra y la necesaria sincronización en amarillo es mayor, debido a que CAMShift es un procesamiento más rápido y se ejecuta casi alternando con la estimación mediante filtrado Kalman.

Añadiendo a CAMShift una estimación mediante el filtrado Kalman para la ejecución multi-hilo, se observa, además de la esperada subida de la tasa de frames entorno a los 50fps, la carga adicional del modulo de VSync encargado de sincronizar los hilos (figura 20). Esto es debido a que el filtrado de Kalman y CAMShift se ejecutan con una relación de 1 a 1, es decir por cada medida real únicamente se realiza una predicción, lo que está en claro contraste con la relación 4 a 1 que se obtenía mediante Compressive tracking. La ventaja de predecir en los instantes que no se tenga imagen es de esta manera insignificante. La experiencia de juego con esta configuración es fluida pero el seguimiento es inexacto y muy sensible a demasiados factores de ruido.

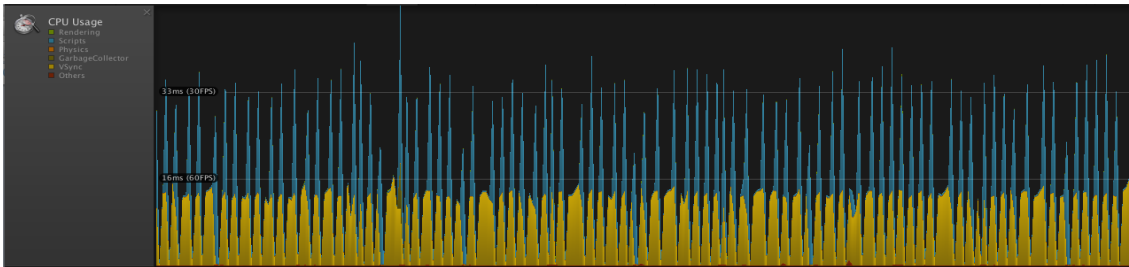


Figura 21 Profiler de la configuración CAMShift, Kalman, multi-hilo y filtrado Retinex multi-escala. Se observa que la carga se reparte de manera parecida al Compressive tracking, debido a que el filtrado Retinex requiere mayor computación.

Para disminuir el impacto de los factores lumínicos, se ha añadido un filtrado Retinex aplicado a cada nueva imagen. De esta manera la relación entre estimación mediante Kalman y procesamiento de nuevas imágenes es de aproximadamente 3 a 1, creando un perfil parecido al Compressive tracking. Los picos de la figura 21 reflejan los máximos del coste computacional que supone la cadena formada por filtrado Retinex, CAMShift y corrección de Kalman.

Debido a que el algoritmo de Compressive tracking es suficientemente robusto frente a cambios lumínicos que el filtrado Retinex intenta disminuir, no hace falta incluir un algoritmo adicional a la cadena final. La precisión alcanzada por el Compressive tracking recompensa el mayor coste computacional, y los mayores intervalos entre las imágenes reales es aprovechado eficientemente por la estimación mediante filtrado Kalman.

4.4. Conclusiones finales

De esta manera, usar Compressive tracking, un filtrado de Kalman y el traslado de la obtención de las imágenes a otro hilo, es la mejor configuración para la experiencia de juego y la propia carga computacional. Esta metodología combina la fiabilidad del seguimiento con una tasa de frames alta y estable.

A lo largo de esta tesina se han aplicado numerosas metodologías empleadas a lo largo de diversas asignaturas del máster. Elementos de las asignaturas de reconocimiento y análisis de formas han ayudado a desarrollar las habilidades necesarias para un entendimiento de algoritmos más complejos para el propósito de esta tesina. En la detección y el seguimiento también han servido las bases de la visión 3D otorgadas en la asignatura del mismo nombre.

También las asignaturas de gráficos por computador y computación grafica han ayudado a entender los principios básicos necesarios para la creación de contenidos tridimensionales, albergando un temario extenso desde iluminación hasta la creación de shaders.

La asignatura de biometría ha contribuido con una extensa base de las técnicas más empleadas y comunes en el seguimiento y la detección de objetos y formas, pero también de elementos biológicos como la cara del mismo jugador. La propia asignatura de videojuegos ha contribuido con diversas metodologías y estrategias

para el desarrollo de videojuegos profesionales, empezando por el mismo documento de diseño, hasta llegar a la optimización y las fases de test del videojuego. Para el desarrollo del propio videojuego exclusivo empleado para demostrar el algoritmo, se ha aprovechado los conocimientos adquiridos en asignaturas como producción de imagen digital y avances en computación gráfica. En esta misma asignatura, se trabajaba principios básicos para la holografía, los que, conjuntamente con los conocimientos adquiridos en realidad aumentada, han permitido desarrollar algunos efectos usados en el juego.

Sin la realización de los estudios en campos y temarios tan amplios como los citados anteriormente, no hubiera sido posible realizar esta tesina. Como se ha podido observar, realizar un juego con técnicas modernas de detección y sobre todo de seguimiento, es posible. A parte se ha conseguido el objetivo usando un hardware más que convencional como lo es una cámara web y un ordenador, prescindiendo de esta manera de sensores sofisticados y caros como lo presentan algunos sistemas de última generación.

5. Futuras líneas de investigación

El sistema de detección y seguimiento facial presentado en esta tesina es lo suficientemente eficiente para funcionar en la mayoría de sistemas actuales de manera fluida. Además se ha conseguido una alta fiabilidad y resolución del seguimiento que mantiene su funcionalidad incluso en condiciones inicialmente desfavorables como oclusiones parciales, malas condiciones lumínicas y la aparición de múltiples objetivos.

Aunque su funcionamiento haya sido descrito mediante el desarrollo de una aplicación de entretenimiento, su campo de aplicación es mucho más amplio. El sistema podría utilizarse en aparatos de nueva generación como *Google Glass*, donde se podría emplear para identificar y analizar ciertas características de las personas pertenecientes al entorno social del usuario. El *Fraunhofer Institut* [17] presento recientemente un sistema capaz de determinar según las expresiones faciales un estado de animo de entre enfadado, alegre, triste y sorprendido, usando un algoritmo sofisticado de detección y análisis facial llamado *SHORE*. Además el algoritmo es capaz de determinar el sexo y la edad aproximada de los usuarios a partir de los datos extraídos de las caras detectadas (figura 22).

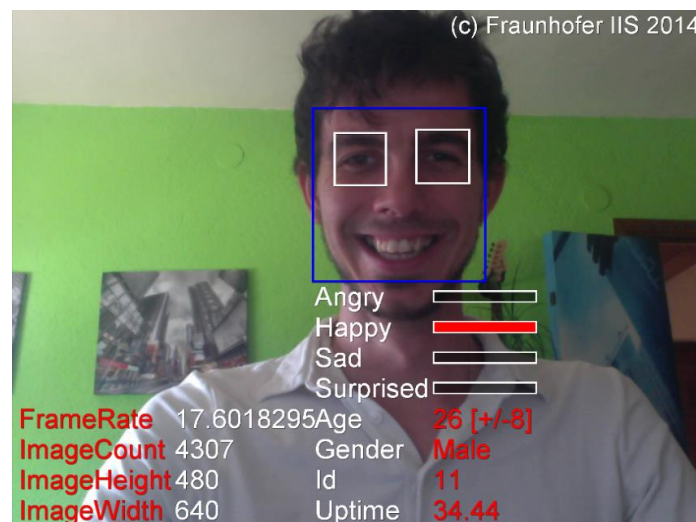


Figura 22 Resultado de una detección y análisis mediante la versión libre del algoritmo SHORE. Tanto la edad aproximada como el sexo del usuario es reconocido con mucha exactitud. La detección más el análisis es muy costoso y se llega a unos 15-20fps. En el dispositivo de *Google Glass* la tasa de frames es incluso menor, de unos 4-6fps. Además el algoritmo es sensible a oclusiones parciales y lumínicas, dejando de reconocer la cara y asignando un nuevo número de identificación en el caso de reencontrarla.

Refinando el detector se podría incluso llegar a realizar un seguimiento de las pupilas del usuario para optimizar el renderizado en contenidos multimedia. En este caso solamente se realizaría un renderizado de la zona de enfoque del usuario, dejando el resto de la pantalla con una resolución mucho menor. Esta técnica tendría un posible uso en dispositivos como *Oculus Rift*, donde además solo se necesitaría seguir un único ojo debido a que la posición del segundo puede ser estimada fácilmente ya que se trata de un dispositivo directamente fijado en la cabeza. De esta manera se podría disminuir considerablemente los mareos de los usuarios por jugar ligeramente desenfocados. Se podría incluso compensar la

carga adicional que supone incluir pantallas de más resolución en el sistema, renderizando únicamente la zona de enfoque del usuario como explicado anteriormente.

La detección y el seguimiento facial también podría emplearse para la autoestereoscopia, dotando al dispositivo con una cámara permitiendo que la pantalla proyecte las imágenes pertenecientes a cada ojo exactamente al punto en el que se encuentre el espectador. La dificultad en esta solución es la disposición variable de los píxeles, que se tendrían que localizar en un tipo de malla móvil para ajustar su focal según la posición del espectador. El multiplicador cambia de color conforme se aumenta su valor, pasando de blanco a amarillo a partir de 4, y rojo a partir de 6. Si se alcanza un multiplicador mayor o igual a seis se enciende una animación de fuego detrás del multiplicador.

6. Anexo I: Happy Gorilla GDD



6.1. Resumen del juego

El juego pretende introducir el control multimodal mediante los movimientos de la cabeza del jugador, únicamente requiriéndose para ello, de una cámara web. Este punto es de especial interés, ya que la mayoría de dispositivos con su respectivo software existentes en el mercado, obligan a adquirir tecnologías propias y sofisticadas con un coste adicional para el usuario final muy elevado. Se trata pues, de ofrecer una experiencia nuevo de control que requiera un mínimo de tecnología disponible para la mayor parte de los usuarios.

El juego se desarrollará en un principio únicamente para dispositivos *Macintosh*, aunque una adaptación a plataformas Windows no se excluye para un futuro. No será compatible con ningún tipo de dispositivo móvil, ni se adaptará para plataformas *iOS* o *Android*, debido a que la tecnología empleada no permite una experiencia de juego agradable en dichos dispositivos.

El juego consiste en el uso adecuado de tecnología de detección y seguimiento facial para fines lúdicos. Se trata de dos modos de juego: una modo de demostración de la tecnología y un modo arcade. En el modo demostración, el jugador únicamente es capaz de trasladar la cámara en un entorno 3D alrededor de un modelo tridimensional de un gorila. En el modo arcade, el jugador controla aparte de la cámara, las manos de un gorila. El juego se crea a partir de la recolecta de diversos objetos que caen desde el borde superior de la pantalla, y que serán recogidos o evitados por el jugador. El juego finaliza automáticamente tras acabarse el tiempo del juego fijado inicialmente en un minuto. De esta manera, el juego consiste en hacer feliz al gorila mediante la recogida de la mayor cantidad de plátanos.

6.2. Características principales

El juego no desarrolla ninguna historia. La motivación principal del jugador consiste en superar su puntuación que le es asignada tras recoger los diferentes objetos que aparecen en la pantalla y finalizarse el tiempo del juego. El juego es desarrollado para un único jugador, es decir no se contempla la opción multi-jugador. Existe la posibilidad de que varios jugadores se turnen para cada partida pero debido a las propias características del algoritmo de seguimiento, una vez inicializado el juego, no se podrá cambiar de jugador con la partida ya en marcha.

Se precisa de una cámara convencional de video externa o integrada en la computadora. Las posiciones de la cámara y de diversos objetos del juego, se disponen según la posición real del usuario respecto a la cámara.

Existen dos tipos de objeto: los positivos y los negativos. Los objetos positivos ayudan al jugador a subir su puntuación. Hay diferentes maneras de ayudar positivamente al jugador: mediante la suma directa de puntos a la puntuación final o mediante el uso de algún tipo de objeto especial que tendrá efectos especiales sobre el comportamiento del jugador. Estos efectos pueden ser cambios de estado de las manos del gorila, suma de tiempo adicional o mediante el desbloqueo de métodos especiales para recoger objetos. Los objetos negativos restan puntuación o finalizan abruptamente algunos modos de juego, quitando al jugador todas las bonificaciones obtenidas hasta entonces.

La aplicación instala automáticamente las librerías necesarias para su ejecución en las carpetas de librerías del sistema operativo. Esto ocurre mediante scripting al entrar por primera vez en el menú principal.


Todos los modelos y sonidos son de origen gratuito y libre de derechos de autor. Los modelos de las frutas y demás objetos han sido creadas a partir de imágenes reales y han sido procesadas con la herramienta *Adobe Photoshop*.


6.3. Jugabilidad

El usuario es capaz de mover e interactuar con los diferentes objetos del juego mediante el uso exclusivo de su cabeza. En ambos modos de juego (demostración y arcade), el usuario tiene el control de la cámara. Mediante la traslación lateral de la cara, se aplica la rotación correspondiente a la cámara en el juego, transmitiendo de esta manera una mayor sensación de profundidad según el efecto conocido como paralaje. Adicionalmente al control sobre la cámara, el jugador controla en el modo arcade además a unas manos de gorila mediante su cabeza. Los movimientos de estas manos se limita a traslaciones únicamente a lo largo del eje horizontal. Cumpliendo ciertos criterios, el jugador es capaz de desbloquear un modo de control adicional que permite una traslación bidimensional en la escena, requiriendo los correspondientes movimientos del jugador. Este modo especial de control se denomina modo agujero negro, y representa la máxima bonificación que el jugador puede conseguir. En este modo el jugador puede conseguir




puntuaciones muy elevadas, siempre y cuando esquive objetos negativos que le obligaran a terminar el modo agujero negro de manera abrupta.


Todos los objetos se generan en posiciones aleatorias en la parte superior de la pantalla. Cada objeto dispone de una probabilidad de aparición para cada generador. Todos los objetos se generan con un ángulo de giro aleatorio entre 0° y 360°. Existen cinco tipos de generadores: el generador de plátanos, el de plátanos triples, un generador de objetos, uno de serpientes y otro de letras. A continuación se describen los diferentes generadores, especificando los efectos y probabilidades de cada objeto.

Generador de plátanos	Efecto	Probabilidad
	+10 en la puntuación final	90% de que genere un plátano y 10% de que genere nada





Generador triple	Efecto	Probabilidad
	+30 en la puntuación final	90% de que genere tres plátanos y 10% de que genere nada

Existen hasta 4 generadores tanto de plátanos simples como de triples. En el modo de juego normal, únicamente uno de cada está activo. Los otros tres se activan en el modo agujero negro, ayudando de esta manera a conseguir más puntuación.

Generador de objetos	Efecto	Probabilidad
	+5 segundos sobre el tiempo total de juego	22.5% de probabilidad
	-10 directamente sobre la puntuación total	22.5% de probabilidad
	2x el tamaño de las manos durante 10s	22.5% de probabilidad

	<p>Comportamiento como en un breakout. Después de tres toques explota para puntos adicionales. Recoge objetos.</p>	<p>22.5% de probabilidad</p>
---	--	------------------------------


Hay un único generador de objetos, que tiene una probabilidad de un 90% de generar alguno de los objetos descritos en la tabla de arriba, y un 10% de probabilidad de no generar nada. El comportamiento del coco merece especial atención ya que no se trata de un objeto coleccionable en el propio sentido de la palabra. Cuando las manos del gorila entren en contacto con el coco, su comportamiento cambia y adquiere una física diferente. Ya no depende de la gravedad como todos los otros objetos, sino que rebota en todos los laterales con los que colisiona excepto con el inferior, pareciéndose de esta manera a la bola de un juego clásico de breakout. Cuando el jugador no es capaz de para la bola en su trayectoria hacia el bore inferior, se destruye automáticamente, reiniciando también el multiplicador. Además se le otorga al coco una nueva dirección aleatoria cuando colisiona con las manos del gorila para evitar comportamientos demasiado predecibles en el movimiento del coco. El ángulo de rebote varía según $\pm 40^\circ$ respecto al eje vertical. También se le añade una velocidad adicional aleatoria entre 1 y 4 veces la inicial. El coco tiene además un spin constante sobre su eje z adicional al ángulo de giro aleatorio con el que se ha generado.

Generador de letras	Efecto	Probabilidad
	<p>Añade la letra "H"</p>	<p>18% de probabilidad</p>
	<p>Añade la letra "A"</p>	<p>18% de probabilidad</p>
	<p>Añade la letra "P"</p>	<p>36% de probabilidad debido a que el jugador ha de recoger 2.</p>
	<p>Añade la letra "Y"</p>	<p>18% de probabilidad</p>

En la parte superior derecha se localiza el indicador de letras. Este panel indica las letras ya recogidas, alumbrado las correspondientes casillas. El panel forma la palabra "HAPPY", en referencia al título del propio juego. El jugador necesita

recoger por lo tanto un modelo de cada letra, excepto de la “p” que precisa de dos unidades. Una vez recogidas todas las letras, el juego inicia el modo agujero negro citado anteriormente. En este modo también se activan los 4 generadores de serpiente conjuntamente con los 8 de plátanos triples y simples. Además, cada plátano cambia su física cuando se acerca a una distancia al agujero negro, creando de esta manera un efecto de campo gravitatorio cuyo centro es el agujero. Todos los plátanos próximos al agujero son atraídos por él.

El control del agujero negro pasa de ser unidimensional a ser bidimensional, es decir también se desplaza por el eje vertical siguiendo los movimientos del jugador.

Generador de serpientes	Efecto	Probabilidad
	Finaliza el modo agujero negro, volviendo el juego a su funcionalidad original	90% de probabilidad de que se genere.

El modo agujero negro finaliza cuando el jugador se choca con una serpiente, volviendo al modo normal donde el jugador controla las manos del gorila.

El multiplicador se ubica en la parte superior izquierda y tiene un valor inicial de uno, es decir, cada plátano es calculado con su valor original a la puntuación total. Con cada toque exitoso del coco con la mano, el multiplicador se aumenta en una unidad. El multiplicador tiene un efecto directo sobre la puntuación que los plátanos añaden al contador total. Si el multiplicador está en dos, un plátano simple adquiere el valor de 20 puntos en vez de 10 y así sucesivamente. Si el coco se pierde por la parte inferior de la pantalla, el multiplicador se reinicia de nuevo a uno.

El indicador de tiempo se ubica en la zona inferior derecha y muestra el tiempo restante de la partida actual mediante una cuanta atrás con intervalos de 0.1 segundo. El contador empieza en un minuto, es decir 60.0 segundos. Cuando el juego llegue a 5 segundos restantes empieza a parpadear y a variar de tamaño con cada medio segundo para alertar al jugador visualmente del final de la partida. El tiempo es manipulable mediante la recogida de relojes que aumentan en 5 segundos el tiempo total de la partida.

6.4. El mundo del juego

El juego transcurre en un entorno amazónico, con la respectiva ambientación plasmada en todos los decorados y elementos de juego. Los escenarios están llenos de modelos de palmeras y demás vegetación que se puede encontrar en las junglas tropicales y selvas amazónicas. El sonido también recrea los sonidos que emite la fauna tropical y ayuda a la ambientación en todas las escenas.

En el juego arcade, el avatar que controla el jugador es un gorila en su hábitat natural. En el modo demostración se puede observar a un gorila desde una perspectiva humana.

Todos los sonidos usados recrean las características del objeto y escena que representan. Al recoger un plátano suena un sonido que recuerda a un “tick”, mientras que si se recoge un plátano podrido suena un eructo. El coco emite sonidos de rebote al chocar contra cualquier lateral o las manos del gorila. El menú principal y el modo arcade presentan unos sonidos de animales tropicales con diferentes rugidos y efectos. Cuando el jugador entra en el modo agujero negro, la música se vuelve de percusión y acompaña a la bonificación conseguida. Las serpientes también emiten sus sonidos característicos al caer del firmamento.

6.5. GUI del juego

El juego se compone de tres escenas: el menú principal, el modo demostración y el modo arcade. El menú principal permite elegir entre ambos modos de juego, iluminándose el modo seleccionado en rojo cuando el ratón pasa por encima. El flujo del juego se describe a continuación.

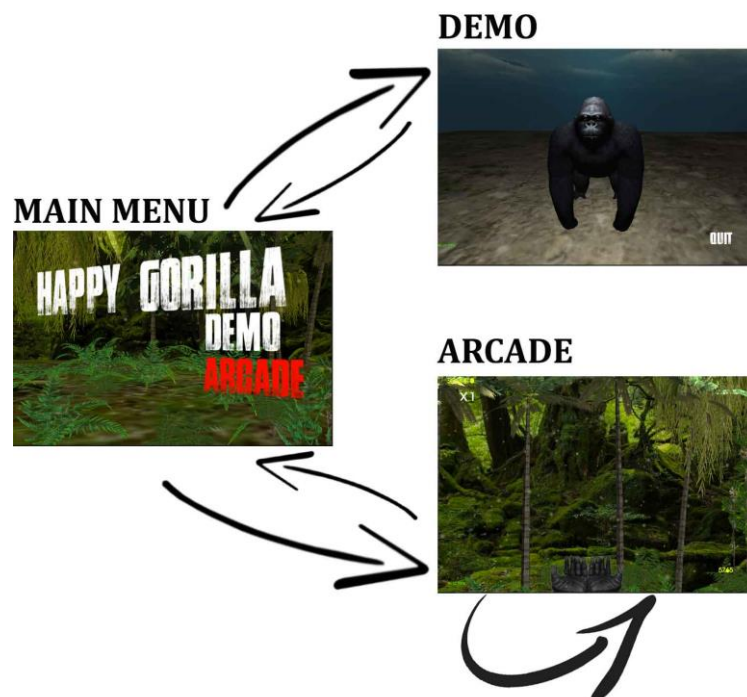


Ilustración 1 Mapa del flujo de juego

Desde el modo demostración solo se puede abandonar hacia el menú principal del juego. Debido a que no hay límite temporal ni una finalidad del juego en este modo que no sea la visualización del algoritmo con una escena tridimensional, este modo solo se termina abandonándolo. El modo arcade en cambio, da la opción de reiniciar con otra partida el juego tras finalizarse el tiempo de 60 segundos. También se le permite al jugador abandonar la partida actual y volver al menú principal.

Las opciones de los menús se trasladan desde el exterior no visible de la pantalla, hasta su posición final de manera continua en un intervalo de tiempo de aproximadamente 2 segundos. De esta manera se crea un efecto visual muy profesional, agradable y dinámico.

El menú de final de partida en el modo arcade incluye además un indicador de la puntuación máxima alcanzada en toda la trayectoria del juego. Esta puntuación se guarda también entre diferentes reinicios del juego.

7. Anexo II: Bibliografía

- [1] Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. In *IEEE Patt. Anal. Mach. Intell.*, volume 20, pages 22–38, 1998.
- [2] H.Schneiderman and T.Kanade. A statistical method for 3D object detection applied to faces and cars. In *International Conference on Computer Vision*, 2000.
- [3] Paul Viola and Michael Jones. Rapid Object Detection using a boosted cascade of simple features. In *International Conference on Computer Vision and Pattern Recognition*, 2001.
- [4] Stan Z.Li and Anil K.Jain. Handbook of Face Recognition 2nd edition. ISBN 978-0-85729-931-4 Springer-Verlag London Limited 2011
- [5] Baltazhar Rouberol blog. Comparison of face detection tools. 2012
- [6] Jarrett Webb and James Ashley. Beginning Kinect Programming with the Microsoft Kinect SDK. ISBN 978-1-4302-4104-1 Apress 2012
- [7] Will Goldstone. Unity Game Development Essentials. Packt Publishing 2009
- [8] Gary Bradski and Adrian Kaehler. Learning OpenCV. O'Reilly Media, Inc. 2008
- [9] Umoove. <http://www.umoove.me>
- [10] faceAPI. <http://www.seeingmachines.com/product/faceapi/>
- [11] faceSDK. <http://www.luxand.com/facesdk/>
- [12] Live Driver. <http://www.image-metrics.com/>
- [13] Hsu R.L., Abdel-Mottaleb M. & Jain A. Face Detection In Color Images. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002
- [14] Yow K.C. & Cipolla R. Feature-Based Human Face Detection. Image and Vision Computing 1997
- [15] S. Anila, N. Devarajan. Simple and Fast Face Detection System Based on Edges. International Journal of Universal Computer Science. 2010
- [16] A. Kulshreshth, J. J. LaViola Jr. Evaluating Performance Benefits of Head Tracking in Modern Video Games
- [17] Fraunhofer Institute for integrated circuits IIS, Shore – Detections of faces and facial expressions
- [18] Edwin H. Land, The Retinex theory of color vision, 1977

- [19] Zia-ur Rahman, Daniel J. Jobson & Glenn A. Woodell, Retinex processing for automatic image enhancement, NASA Langley Research Center Hampton, Virginia
- [20] Kaihua Zhang, Lei Zhang & Ming-Hsuan Yang. Real-Time Compressive Tracking. Depart. of Computing, Hong Kong Polytechnic University 2012
- [21] Google Tango ATAP <https://www.google.com/atap/projecttango/> - project
- [22] Microsoft Kinect SDK y página principal para desarrolladores para la plataforma Windows, <http://www.microsoft.com/en-us/kinectforwindows>
- [23] Facebook face recognition for automatic tagging in user uploaded potos <https://www.facebook.com/help/122175507864081>
- [24] Oculus Rift Official Homepage <http://www.oculusvr.com>
- [25] Nintendo Wii Official Homepage <http://wii.com>
- [26] Samsung Facial Recognition on the Samsung Galaxy S4 <http://www.androidcentral.com/how-use-facial-recognition-samsung-galaxy-s4>
- [27] Apple face detection for iPhoto http://support.apple.com/kb/HT3442?viewlocale=en_US&locale=en_US
- [28] Unity 3D Official Homepage <http://unity3d.com>
- [29] Nintendo NES Zapper wikia Homepage http://nintendo.wikia.com/wiki/NES_Zapper
- [30] Auto-estereoscopia entrada Wikipedia <http://en.wikipedia.org/wiki/Autostereoscopy>
- [31] Maria Isabel Ribeiro, Kalman and Extended Kalman Filters: Concept, Derivation and Properties, Institute for Systems and Robotics, Lisboa 2004
- [32] David Exner, Erich Bruns, Daniel Kurz & Anselm Grundhöfer. Fast and Robust CAMShift Tracking. Johannes Kepler University Linz, Austria
- [33] Dong Hyun Jeong, Caroline Ziemkiewicz, William Ribarsky & Remco Chang. Understanding Principal Component Analysis Using a Visual Analytics Tool. Charlotte Visualization Center, UNC Charlotte
- [34] Grass Valley. White Paper 3D Televisions 2010
- [35] Atari Inc. Breakout game [http://en.wikipedia.org/wiki/Breakout_\(video_game\)](http://en.wikipedia.org/wiki/Breakout_(video_game))
- [36] M. Rahman, M. Gamadia & N. Kehtanravaz. Real-time Face-based Auto-Focus for Digital Still and Cell-Phone Cameras. Dept. of Electr. Eng., Univ. of Texas at Dallas 2008

[37] William Higinbotham http://en.wikipedia.org/wiki/William_Higinbotham & <http://history-computer.com/ModernComputer/thinkers/Higinbotham.html>

[38] Ley de Moore. http://es.wikipedia.org/wiki/Ley_de_Moore