1. Resumen

Memoria de proyecto realizado en Vicomtech sobre tecnologías 3D e interacción.

Se han analizado diferentes aspectos de la representación tridimensional estereoscópica y de la interacción humano-máquina. Para demostrar la aplicabilidad del resultado de este análisis, se ha desarrollado una aplicación orientada a análisis meteorológico que implementa algoritmos basados en estos resultados.

Índice

2. Introducción

Esta es la memoria del Proyecto Fin de Carrera de los estudios de Ingeniería en Informática, realizados por el alumno Jon Goenetxea Imaz en la universidad MGEP-MU localizada en Arrasate-Mondragón.

Este proyecto ha sido realizado en Vicomtech, empresa situada en el parque tecnológico Miramon, en la ciudad de Donostia-San Sebastián, y ha sido dirigido por Luis Unzueta y Aitor Moreno, teniendo a Osane Lizarralde como tutora.

Para esta memoria, se entregará un documento que contendrá toda la información sobre el desarrollo del proyecto. Así mismo, se entregará a titulo de anexo, un vídeo demostrativo de la aplicación desarrollada.

Este documento se divide en varios apartados:

- Se hará una introducción de las tecnologias que toman parte, comentado implementaciones actuales y usos más comunes. También hará una introducción a la problemática del proyecto, haciendo posible determinar los objetivos marcados para la consecución de este. Para terminar, se listarán las diferentes fases que se han llevado a cabo.
- Se profundizará en los puntos más significatibos del desarrollo, explicando un poco más en profundidad cada una de las fases, y que se intenta conseguir con ellas.
- En el apartado de conclusiones, se presentarán los resultados obtenidos, así como los problemas más significativos a la hora de desarrollar el proyecto. También se incluiran las conclusiones a las que se han llegado, segido de nuevos caminos o propuestas para proximos desarrollos basados en este proyecto.
- Para la última sección, se han guardado una serie de explicaciones más en profundidad de los algoritmos utilizados para resover la problemática propuesta. Esta documentación clasificada en modo de anexo, estará referenciada a lo largo del texto.

2.1. Estado del Arte

Para el desarrollo de este proyecto ha sido necesario mezclar diferentes disciplinas del ámbito informático. Entre ellas, destacan los gráficos 3D, la representación estereoscópica y el reconocimiento de gestos. En este capítulo se hará una breve introducción de como se ve en la actualidad cada una de estas, dejando para posteriores apartados el profundizar en estas tecnologías.

2.1.1. Gráficos 3D en la actualidad

Desde antaño se ha visto la necesidad de representar la realidad en imágenes. Estas representaciones se utilizan tanto para definición de estructuras, arte, diagramas para unificar conceptos, etc...

Desde el antiguo egipto hasta no hace mucho, esto se realizaba sobre el papel o sobre algo similar. No obstante, desde que Iván Sutherland presentó su trabajo en el MIT en 1962, haciendo que un ordenador dibujase una línea en un monitor CRT, muchos esfuerzos se han orientado a pasar esta necesidad de representar la realidad a ámbitos informáticos.

Hoy en día tenemos una gran variedad de librerías gráficas que se pueden utilizar para estos fines, así como un gran catálogo de aplicaciones de modelado con las que se pueden representar edificios, piezas de mecanizado, estructuras, objetos, personas y cualquier cosa que se pueda imaginar.

Estas representaciones, además de la altura y la anchura, también representan la profundidad de los objetos, con lo que se pueden hacer representaciones en tres dimensiones. Cada vez se desarrollan técnicas que consiguen que estas representaciones sean cada vez más realistas (ver figura 1), hasta el punto de confundirlos con imágenes reales.

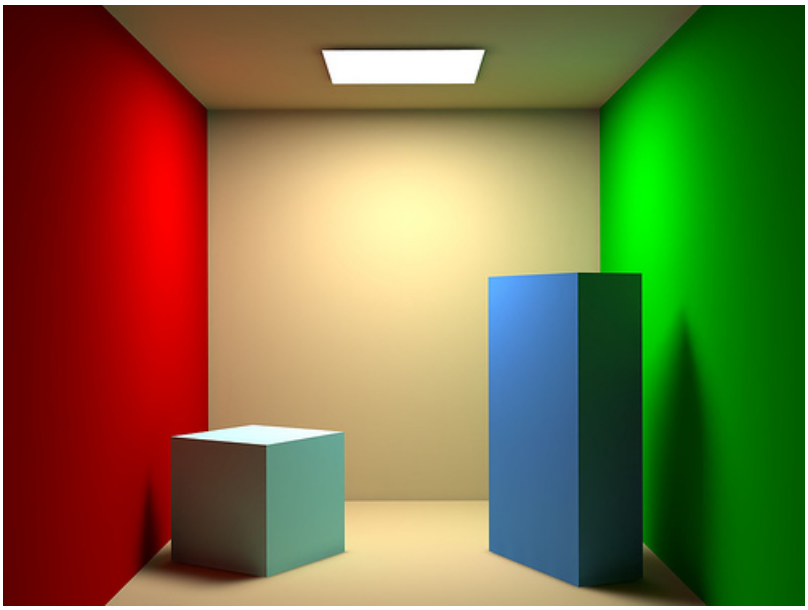

Figura 1: Escena renderizada con la herramienta Blender simulando una escena real, utilizando algoritmos de radiosidad para el cálculo de la luz ambiental

Existen muchas librerías gráficas que se proveen al desarrollador de funciones y métodos para estos fines. Las más conocidas hoy día pueden ser DirectX[1] y OpenGL [2], utilizadas en diversos entornos. Estas librerías dan al desarrollador la capacidad de crear figuras geométricas básicas. Combinando diferentes figuras básicas podemos generar figuras más complejas. Si sumamos estas figuras más complejas, podemos crear infinidad de formas, figuras y escenarios con diversos niveles de detalle, y representar estos en un monitor. Además de darles forma, se proveen métodos para aplicarle texturas a las superficies de estos objetos, lo que puede darles de un aspecto realista. También es posible aplicar diferentes iluminaciones y distribuciones de luz, lo que posibilita la correcta representación de una escena.

Por su parte, el hardware con el que se representan estas escenas en pantalla (mayormente la tarjeta gráfica del ordenador) utiliza vértices y triángulos para sus cálculos. De esta manera un ordenador actual puede generar casi cualquier escena o imagen 3D utilizando triángulos.

Además de representar objetos, podemos controlar la cámara virtual o punto de vista que los esta observando, especificando la posición, la orientación y la dirección, así como la perspectiva para representar la escena expuesta.

[1] Librería gráfica desarrollada por Microsoft. Más información en : '*http://msdn.microsoft.com/es-es/directx/default(en-us).aspx*'

[2] Librería libre multi-plataforma. Más información en : '*http://www.opengl.org/*'

Estas librerías trabajan con vértices, líneas y formas muy básicas, y aunque con ellas podemos crear casi cualquier cosa, es muy difícil modelar figuras complejas desde el mismo código. Por esta razón, existen otras herramientas, que utilizando estas librerías, nos ofrecen un entorno más amigable para generar todo tipo de modelos y escenas más o menos realistas.

Normalmente, se utilizan este tipo de aplicaciones para hacer los modelos por separado. Una vez tenemos los modelos, podemos hacer varias cosas con ellos.

Por una parte, podemos crear imágenes estáticas, con gran detalle y realismo. Estas imágenes son útiles no solo en el ámbito artístico, sino también en el industrial. En muchas ocasiones, además de la componente visual, es posible añadir a los modelos de datos e información como material, comportamiento, resistencia al calor, etcétera. Existen aplicaciones orientadas a delineación, con las cuales se pueden definir objetos que se van a producir en la empresa, manteniendo unas medidas concretas. Estas aplicaciones no solo dan como resultado un modelo 3D, sino que también pueden generar planos, diferentes perspectivas, proyecciones acotadas, y demás tipos de planos utilizados en la industria. La principal ventaja de este tipo de sistemas es la posibilidad de cambiar los planos fácilmente, hacer copias, calcular diferentes perspectivas o tipos de plano a partir de un solo modelo, o hacer diferentes versiones de un mismo modelo con gran facilidad.

Entre este tipo de aplicaciones, tenemos las aplicaciones CAD (del inglés *Computar Idead Designa*) que se orientan a modelar las piezas industriales que luego se podrán poner en producción.

Una de las más utilizadas en este aspecto, y por poner un ejemplo de solución CAD, tenemos Auto CAD (Página web oficial: '*www.autodesk.es*'). Esta herramienta se utiliza en muchos entornos industriales, y en muchas ocasiones es posible pasar estos modelos directamente a las máquinas para empezar a producirlas.

Por otro lado, podemos generar diferentes tipos de animaciones, haciendo que los modelos interactúen entre si. Ejemplos de este tipo de animaciones pueden ser las películas animadas que se proyectan hoy día en los cines, así como vídeos promocionales de edificaciones por construir enseñando en un vídeo como quedaría el trabajo terminado.

Estas soluciones están más orientada al mundo del diseño y animación, entre ellas podemos encontrar aplicaciones como Blender (Página web oficial: '*www.blender.org*')

o 3D Studio Max (Página web oficial: '*www.autodesk.es*'). Estas herramientas están orientadas a conseguir modelos más realistas de escenas, así como animaciones con movimientos de objetos y cámaras.

Además de estos entornos en los que el usuario puede crear representaciones tridimensionales de objetos reales o ficticios, existen herramientas y técnicas que hacen posible capturar objetos reales y realizar una representación virtual de estos, generando un modelo que los represente. Utilizando este tipo técnicas, se ha conseguido por ejemplo, generar representaciones tridimensionales de la superficie de la tierra utilizando como base fotografías capturadas desde satélites espaciales dotados de cámaras especiales.

A menor escala, existen escáners que con diferentes técnicas son capaces de escanear objetos pequeños con gran precisión.

Por lo tanto, utilizando aplicaciones y tecnologías como las mencionadas, podemos crear una representación realista de, pongamos por ejemplo, un moledor de café. También podemos realizar animaciones en las que los diferentes modelos interactúan entre sí de una forma específica. No obstante, si queremos que el usuario o espectador interactúe con este modelo, necesitamos generar un entorno de interacción. Para esto, podemos utilizar las mismas librerías antes mencionadas, y cargar los modelos generados con herramientas de más alto nivel, para poder interactuar con ellos. Esta interacción puede ser de diversos tipos, ya que podemos girar, acercar, mover, deformar, y un largo etcétera de acciones diferentes.

Un ejemplo de entornos de interacción son los vídeo-juegos. Estas aplicaciones introducen modelos tridimensionales en entornos en los que podemos interactuar con ellos como si estuviésemos en la escena. Además, los vídeo-juegos actuales incluyen motores gráficos que simulan las propiedades físicas del mundo real, como la gravedad, fricción, inercias, colisiones....

Este tipo de entornos interactivos también son utilizados para realizar simulaciones de situaciones o construcciones. En estos casos, se utilizan motores de simulación física muy potentes, con los que se puede simular de forma muy realista las leyes de la física que influirían en el comportamiento de estos objetos en el mundo real. De esta manera, podemos ver las posibles reacciones a diferentes acciones que queramos realizar sobre los modelos, así como el comportamiento de estos a raíz de estas acciones.

2.1.2. Utilización del 3D estereoscópico

Con un ordenador podemos conseguir representaciones tridimensionales de infinidad de objetos, pero al hacer la representación de estos, estamos ligados a dispositivos que lo representan en dos dimensiones. Esto es porque el papel, un proyector, o un monitor no pueden, en principio, representar una imagen en tres dimensiones reales, sino una proyección de estas en un solo plano.

Desde los orígenes de la fotografía y sobre todo el cine, muchos han intentado salvar esta barrera de la tercera dimensión, algunos con más éxito que otros.

El 3D estereoscópico es un método basado en proyectar una imagen diferente a cada ojo, para simular un entorno real, en el que cada ojo obtiene una perspectiva diferente de la misma escena. Una vez que cada ojo tiene su imagen, siendo estas *realistas*, el cerebro interpreta la profundidad según las coincidencias entre ambas, como lo haría en un entorno real.

Existen diferentes métodos para conseguir proyectar una imagen diferente a cada ojo. La mayoría se basan en representar las dos imágenes en la misma pantalla (juntas o alternativamente), y filtrarlas para que cada ojo solo vea una de ellas. Entre estas, las más utilizadas son las siguientes:

Filtrado anaglífico: esta tecnología se basa en imprimir las dos imágenes en la misma pantalla pero cada una con una mascara de color diferente. De esta manera, si tenemos una representación de color RGB (Red-Green-Blue o Rojo-Verde-Azul), una imagen se representaría con una máscara de rojo (solo se imprimiría la componente roja) y la otra con una máscara verde-azul. Para volver a separar las imágenes se utilizan filtros de color. El espectador debería tener un filtro rojo en un ojo (ver figura 2), y un filtro verde-azul en el otro. De esta manera, cada ojo solo vería la componente que le corresponde.

Figura 2: Gafas anaglíficas con filtros rojo para el ojo derecho y azul para el izquierdo (izquierda). Fotografía de una hoja de Alchemilla (ampliada 500 veces) representada en 3D estereoscópico utilizando filtrado anaglífico (derecha)

Filtros polarizados (Estéreo Pasivo): esta tecnología se basa en proyectar las dos imágenes a la vez en la misma pantalla, cada una filtrada con una lente polarizada en una dirección (para un filtrado eficiente, las direcciones de los filtros deben tener una diferencia de 90º). Se utilizan dos proyectores, uno para cada imagen, y filtros polarizados para cada proyector. El espectador filtra cada imagen con colocando filtros polarizados en cada ojo (figura 3) en la misma dirección en la que esta polarizada la imagen que le corresponde a ese ojo.

Figura 3: Gafas polarizadas

Gafas Activas: esta tecnología se basa en proyectar las dos imágenes alternadas en la misma pantalla. Para ello son necesarias gafas activas (figura 4), que deben estar sincronizadas con la representación. De esta manera, cuando se esta representando la imagen para el ojo derecho, y estando sincronizadas las gafas con la imagen, la lente del ojo izquierdo se vuelve opaca, y viceversa cuando la representación es para

el otro ojo[3]. La sincronización entre las gafas y la representación se suele hacer a través de una señal infrarroja, o sistema similar.

Figura 4: Gafas LCD activas modelo X102 DLP, de la compañía XpanD

HMD (Head Mounted Display): esta tecnología se basa en proyectar una imagen diferente a cada ojo directamente. Para esto, se utiliza un dispositivo parecido a un casco, que está equipado con dos pequeños proyectores, y unas lentes, que envían directamente la imagen a su respectivo ojo (ver figura 5).

Figura 5: Dispositivo HMD (Head Mounted Display)

Monitores Auto-estereoscópicos: Esta gama de monitores tuvo su entrada en el mercado hace pocos años. Este año, empresas como Samsung están anunciando nuevas pantallas con esta tecnología. En esencia, estos monitores son capaces de proyectar una imagen diferente a cada ojo sin necesidad de utilizar gafas especiales.

[3]Nótese que la imagen de una pantalla normal tiene como mínimo un índice de refresco de 60Hz. Si tenemos que alternar dos imágenes cada vez la frecuencia mínima de refresco del sistema de proyección deberá ser de 120Hz (60Hz por cada ojo). Por ello, es posible utilizar este tipo de representación estereoscópica en un monitor, siempre y cuando este tenga la velocidad de refresco suficiente.

Para ello se basan en un sistema de lentes lenticulares, que separan la los rayos de luz que refleja la imagen que tiene detrás. Este tipo de lentes requieren que las imágenes de detrás estén horizontalmente alternadas, como se ve en la figura 6.

Estas lentes son más comunes en otros productos, como las postales o folletos publicitarios, en las que la imagen cambia cuando se giran.

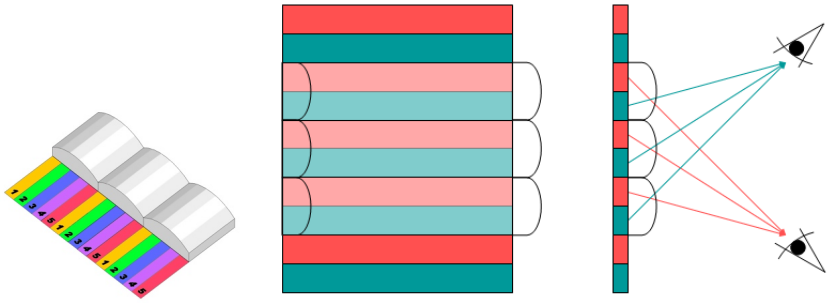

Figura 6: En la primera imagen (izquierda) podemos ver un esquema de una representación lenticular de cinco imágenes diferentes. Las lentes separan los haces de luz enseñando una de estas cinco imágenes según la posición del espectador. En la segunda (derecha), puede apreciarse un esquema de como estas lentes separan las imágenes que serán visualizadas por el espectador.

Hoy en día, se está dando un *boom* en torno a este tipo de tecnologías. Se pueden ver películas en 3D, vídeo-juegos en 3D, pantallas de ordenador y televisores preparados para la representación en 3D y cada vez más aplicaciones en diferentes campos como la medicina, diseño y construcción, meteorología, etc....

Tanto en escenas renderizadas como en escenarios reales, es necesario siempre tener al menos dos imágenes para representar la profundidad de la escena. Esto se puede hacer tanto con objetivos capaces de capturar dos imágenes o simplemente con dos cámaras y uniendo después las imágenes tomadas. Cámaras reales con este fin existen desde hace mucho tiempo atrás, y aunque se ha depurado mucho la técnica, en muchas ocasiones, con sacar dos imágenes con la misma cámara podemos tener el mismo resultado.

Sin embargo, en grabaciones y animaciones, necesitaremos sistemas especiales. En este aspecto, si se utilizan dos cámaras independientes, en post-producción es necesario sincronizar ambas capturas, cosa que requiere mucho tiempo. Para solventar este problema, se diseñaron películas de doble imagen, que imprimen ambas imágenes una al lado de la otra, para mantener las dos capturas siempre sincronizadas.

Con las cámaras digitales, no existe el mismo problema, ya que se pueden sincronizar ambas imágenes utilizando marcas de tiempo en el momento de capturar la escena.

En mundos virtuales, también se utilizan dos cámaras, pero de estas, se obtiene el resultado final directamente y sin ser necesario grabar la escena.

2.1.3. Reconocimiento de gestos y trackeado

Desde la aparición de los ordenadores personales, las interfaces de interacción humano-maquina han sido complicadas, aunque se han ido simplificando poco a poco, generando nuevas formas de interacción entre ambos.

Primeramente, solo se disponía de un teclado convencional, y toda interacción entre el usuario y la maquina se daba por este medio. Además del teclado, se podían utilizar diferentes tipo de joystick y botoneras, que no tenían porque ser de ningún tipo estándar, y muchas veces eran específicos de una concreta aplicación.

La primera revolución la supuso el ratón, que ofrecía otra forma de interactuar, dando pie a las interfaces gráficas que hoy conocemos, basados en paneles y ventanas.

Según se desarrollaba la tecnología, fueron apareciendo nuevas formas de interacción, como paletas gráficas, trackballs, pantallas táctiles etc....

Hoy en día, la mayoría de dispositivos móviles tienen integradas pantallas táctiles con las que poder interactuar con el sistema. El reconocimiento de la escritura es muchas veces nombrado como ejemplo de reconocimiento de gestos, y en este tipo de entornos es fácil encontrar aplicaciones que utilicen este sistema como forma de interacción.

Un gesto es una acción corporal no verbal, que tiene un mensaje implícito y reconocible por otro. Estos gestos pueden implicar movimientos del tronco, extremidades o partes de la cara. Es el hecho de tener un mensaje lo que hace que un simple movimiento pase a ser un gesto. Decir adiós con la mano, afirmar o negar con la cabeza, un guiño para expresar que se esta bromeando, señalar un objeto, o incluso mover la mando para pedirle a alguien que se acerque pueden ser ejemplos válidos de gestos cotidianos que prácticamente todo el mundo sabría reconocer.

En esencia, el reconocimiento de gestos se basa en intentar reconocer estos utilizando algoritmos matemáticos. Los gestos pueden ser de diferentes tipos: gesticulación corporal general, escritura, gestos con las manos, gesticulación facial, etcétera. Algunos de estos tipos, se orientan a reconocimiento de ordenes a través de gesticulación , para poder interactuar directamente con máquinas o dispositivos. Otros de ellos, intentan reconocer el estado de ánimo de alguien a través de su posición, y sobre todo, de sus gestos faciales.

El desarrollo de este tipo de técnicas de reconocimiento posibilita al usuario a interactuar con máquinas de una forma más natural, y a veces sin necesidad de dispositivos mecánicos (teclados, ratones, botoneras,...) intermediarios para dar ordenes.

La forma de capturar información más común en la bibliografía actual es la de utilizar cámaras y visión artificial [4]. De esta manera, es posible enviar la información sobre el usuario sin que este tenga ningún dispositivo en las manos. En este tipo de implementaciones, las imágenes tomadas por las cámaras se post-procesa, para extraer de la imagen entera la imagen correspondiente al usuario. La imagen del usuario se analiza a lo largo del tiempo, y se genera una estimación del movimiento, que será la información que se tendrá en cuenta a la hora de clasificar el gesto.

Otras opciones para capturar los movimientos del usuario son los sensores de movimiento como por ejemplo acelerómetros y giroscopios [8, 6]. Este tipo de sensores es capaz de detectar su movimiento en el espacio (giros y translaciones) y los datos capturados pueden utilizarse para el reconocimiento de gestos.

Una vez capturados los datos, el gesto capturado se procesa utilizando un algoritmo diseñado con este propósito [3]. La mayoría de estos algoritmos, comparan la información procesada con una base de conocimiento almacenada con anterioridad y que contiene muestras ya procesadas de gestos previamente capturados. El algoritmo da como resultado que nivel de parecido tiene la muestra capturada, con las muestras almacenadas en la base de conocimiento. Dependiendo de cuanto se parezca este gesto con los de la base de datos, podremos decir o cual de los gestos es, o si no ha sido un gesto conocido.

El reconocimiento de textos es utilizado para problemáticas muy diversas. Entre otras, las más destacables pueden ser:

Reconocimiento del lenguaje de signos: utilizando técnicas de captura de gestos es posible traducir lo expresado a través del lenguaje de signos a texto [**?**].

Control a través de gesticulación facial: utilizando reconocimiento de gesticulación facial es posible crear formas de interacción con computadores para personas incapacitadas, o que no pueden utilizar las manos para interactuar con el ordenador.

Controladores para realidad virtual inmersiva: en varios casos, al utilizar dispositivos de realidad virtual como HMD (Head-Mounted-Display, ver figura 5), encontrar el teclado o el ratón puede ser un problema. Utilizando los gestos como forma de interacción, este tipo de dispositivos pueden ser prescindibles.

Reabilitación de pacientes: colocando acelerómetros y giroscopios en partes clave de un paciente a rehabilitar, es posible recoger información que posibilite a dispositivos automáticos ayudar su rehabilitación asignándole o ayudándole a realizar ejercicios específicos.

2.2. Problemática

Actualmente en las muchas empresas se generan representaciones 3D para sus proyectos. Muchas de estas son parte activa de estos proyectos, siendo usados para transmitir a otros miembros del equipo como va a ser el acabado final, o para representar aspectos del proyecto de forma visual.

Otra parte de estas representaciones tienen aspectos analizables, como una representación tridimensional de las precipitaciones producidas sobre una zona geográfica concreta para fines de análisis meteorológico por ejemplo.

En este ámbito, las estaciones meteorológicas capturan los datos en bruto, y los almacenan estas lecturas en tablas de bases de datos. Estos datos se analizan directamente, utilizando para el análisis cálculos numéricos. Este tipo de análisis es muy laborioso, y se necesita de mucha experiencia y conocimiento para obtener conclusiones de forma rápida y eficaz. Además, existen diferentes sistemas de captura de datos. Así como los radares dopler capturan la información en forma de conos concéntricos, otros tipos o tecnologías pueden capturar la información de otra manera. Esta diferencia de lecturas afecta directamente a la interpretación de los datos.

En los últimos años se han desarrollado aplicaciones que, a través de imágenes en dos dimensiones (áreas de colores, isobaras, etcétera), ayudan al experto a analizar los datos capturados. No obstante, estas representaciones siguen perdiendo una componente importante, ya que los datos capturados representan un volumen tridimensional.

En cualquier caso, y aun teniendo los modelos tridimensionales, estos tienen que ser vistos a través de dispositivos que generan una proyección bidimensional de los mismos, como proyectores o monitores, haciendo perder detalles de profundidad y percepción espacial de la escena a analizar.

2.3. Objetivos del proyecto

El objetivo del proyecto es desarrollar un sistema capaz de representar una escena en un entorno 3D, que simule una maqueta virtual. Esta maqueta deberá verse de una forma realista desde el punto en el que esté posicionado el espectador, pudiendo este último moverse al rededor de la pantalla.

Además de la representación, sera necesario desarrollar un sistema de interacción con el sistema que le de un valor añadido adicional al ser utilizado en monitores grandes, como proyectores.

Como aplicación demostrativa, se ha escogido una aplicación de análisis meteorológico. Esta aplicación representará un mapa tridimensional sobre el cual serán dibujadas las nubes de un día concreto a cierta hora.

El espectador podrá interactuar con el mapa y la representación de nubes pudiendo realizar las siguientes acciones:

- Cambiar la inclinación de la vista
- Cambiar la orientación del mapa
- Aumentar o disminuir el zoom
- Navegar por el mapa
- Poner en marcha o parar la animación
- Avanzar o retroceder en la animación más rápido
- Volver al principio de la aplicación
- Pasar una sola imagen hacia delante o hacia detrás

2.4. Retos tecnológicos

Para un mejor entendimiento de lo que supone llevar a cabo el proyecto, serán comentados brevemente los principales retos en el aspecto tecnológico.

Estos retos se centran en dos ámbitos distintos, la representación realista de el mundo 3D, y la interacción humano-máquina.

2.4.1. Representación 3D

Para cumplir con los objetivos del proyecto, es necesario poder relacionar la posición del espectador en el mundo real, con la posición de la cámara en el mundo virtual. Para ello será necesario ajustar la relación de distancias y posiciones de ambos sistemas de referencia. El sistema de referencia del mundo virtual estará determinado por lo definido en la aplicación a la hora de cargar los modelos. En el mundo real, sera el sistema de captura de datos el que determinará cual es el sistema de referencia utilizado.

Obtenidos los datos y habiéndolos ajustado, es necesario adecuar la vista de las cámaras a la del espectador, para generar un entorno inmersivo, y que parezca que el monitor o pantalla sea una ventana, y que podemos ver a través de ella cuando nos movemos.

2.4.2. Interacción Humano-Máquina

Al tener que utilizar marcos de representación como pantallas retroproyectadas, los interfaces comunes pueden no ser adecuados para interactuar con el entorno virtual. Por ello, será necesario desarrollar un sistema de interacción que posibilite que el usuario se mueva sin perder la interacción con el sistema.

Además del movimiento, la interacción debe ser mínimamente intuitiva, sin añadir dificultad a la navegación.

2.5. Fases del proyecto

El desarrollo del proyecto se ha orientado a funcionalidades, por lo que las diferentes funciones a desarrollar marcan una fase en si misma. Cada una de estas fases está compuesta por una serie de sub fases o hitos que marcan el desarrollo de cada una.

De esta manera, el esquema de fases de este proyecto quedaría así:

- Aprendizaje e investigación (tecnologías básicas y lenguajes)
  - Tracking y captura de datos
  - Lenguajes y herramientas
  - Librerías gráficas
- Reconocimiento de gestos
  - Aprendizaje e investigación (en torno al reconocimiento de gestos)
  - Implementación de clasificador
  - Desarrollo de aplicación de configuración y testeo
- Representación ajustada al usuario y estéreo
  - Aprendizaje e investigación (en torno a la representación estereoscópica)
  - Cálculos y ajustes de perspectiva
  - Desarrollo de aplicaciones de ejemplo y testeo
- Implementación de aplicación de análisis meteorológico
  - Representación de la escena
  - Implementación de la navegación
  - Integración de dispositivos de trackeo
  - Integración de reconocimiento de gestos
  - Documentación

3. Desarrollo del proyecto

En este punto, se analizará como se ha desarrollado el proyecto.

En primer lugar, ha sido necesario adquirir los conocimientos más básicos para la utilización de las herramientas y la forma de desarrollo de la empresa. Por ello, la primera fase del proyecto se ha centrado en el aprendizaje de estos conocimientos.

Una vez asimilados, se han analizado y desarrollado los dos pilares del proyecto, el reconocimiento de gestos y la representación gráfica.

Dominados estos aspectos, se ha pasado a implementar la aplicación demostrativa final, implementando versiones más elaboradas de las desarrolladas en las dos fases anteriores.

En este capítulo veremos más en detalle estas fases.

3.1. Aprendizaje e investigación (tecnologías básicas y lenguajes)

El objetivo de esta primera fase, ha sido hacer una introducción a los sistemas que se van a utilizar a lo largo de todo el proyecto.

Principalmente, se ha hecho incapié en las tecnologías nuevas, así como en los lenguajes y librerías.

3.1.1. Tracking y captura de datos

Un aspecto clave para el desarrollo es el que concierne a la captura de datos de posición. Esta «técnica» es conocida como *tracking* y se basa en capturar los datos de posición y orientación de un objeto o persona en el espacio.

Como ya se ha adelantado en secciones anteriores, para el desarrollo de la aplicación es necesario conocer en todo momento la posición del espectador, y capturar los movimientos realizados por el usuario para interpretarlos como gestos.

Como análisis en este aspecto, se han probado diferentes tipos de herramientas de *tracking* que posee la empresa. De esta manera se ha escogido la que se creía más adecuada.

Además de esto, se han creado diferentes aplicaciones de prueba para poder asimilar el funcionamiento de estos dispositivos.

Las herramientas seleccionadas para el proyecto han sido:

Flock of Birds: este tracker magnético comercial producido por la compañía *Ascension* [4], consta de un emisor y un sensor. El sensor detecta su posición respecto de la del emisor. Es una herramienta muy precisa y con un bajo consumo de recursos.

Nintendo Wiimote: Es un controlador de juegos sin cables, que se puede conectar con el ordenador a través de una conexion bluetooth. Existen apis orientadas al manejo de este controlador. Los datos son obtenidos tanto de los botones (datos de pulsación), como del acelerómetro triaxial montado tanto en el mando principal como en el auxiliar, llamado Nunchuck (datos de movimiento).

3.1.2. Lenguajes y herramientas

A la hora de desarrollar, es necesario conseguir una interoperabilidad entre los diferentes elementos hardware. De esta manera, es necesario utilizar idiomas y herramientas compatibles con todos ellos.

En el mundo de los dispositivos de seguimiento (*trackers*), es muy común que adjunten ejemplos y documentación orientada a utilizar herramientas como *Microsoft Visual C++* [5] o *Apple xCode* [6](en algunos casos). Por ello se ha elegido *C++* como lenguaje de programación y *Microsoft Visual C++* como entorno de desarrollo.

Además de esta razón, *C++* es un lenguaje muy extendido, y que ofrece control casi total en de las capas más bajas del diseño.

[4]Más información en : *'http://www.ascension-tech.com'*

[5]Herramienta de desarrollo programacional desarrollado por la compañía Microsoft, y disponible solo para plataformas Windows. Más infomración en «http://www.microsoft.com/express/windows/»

[6]Herramienta de desarrollo programacional desarrollado por la compañía Apple, y disponible solo para plataformas Mac OS X. Más información en «http://developer.apple.com/technologies/tools/xcode.html»

3.1.3. Librerías gráficas

Para generar representaciones gráficas es necesario utilizar librerías de programación orientadas a ello.

Como librería gráfica se ha escogido OpenGL[7], por su carácter abierto, y las posibilidades que nos ofrece.

No obstante, para la representación de escenas complejas, OpenGL sigue siendo de muy bajo nivel, por lo que esta tarea se puede volver muy complicada. Para estos casos, es necesario utilizar implementaciones de más alto nivel, como OpenSceneGraph [8]. Esta última está orientada a la representación de escenas con modelos complejos utilizando un grafo para definir la composición de la escena.

Evidentemente, el manejo de estas librerías no es sencillo, y adquirir el manejo y la comprensión necesarias para implementar una aplicación tan compleja con ellas requiere mucho esfuerzo y tiempo.

3.2. Reconocimiento de gestos

Esta fase se dedica exclusivamente a uno de los pilares del proyecto.

El reconocimiento de gestos es una forma más intuitiva de interactuar con el computador. Para ello, capturamos los datos de movimiento del usuario, y la máquina decide si ha sido un gesto o no, y si lo ha sido, a que gesto pertenece el movimiento capturado.

Esta es una disciplina muy estudiada estos días, y que esta sufriendo grandes avances últimamente.

[7] Librería de procesamiento gráfico muy extendido y disponible para multiples plataformas. Más información en «http://www.khronos.org/opengl/»

[8] Librería de procesamiento gráfico orientado a representación de escenas complejas. Utiliza un arbol de grafos como forma de definir la disposición de la escena final. Mas información en «http://www.openscenegraph.org/projects/osg»

3.2.1. Aprendizaje e investigación (en torno al reconocimiento de gestos)

El reconocimiento de gestos es una disciplina muy compleja, que relaciona algoritmos matemáticos y tratamiento de señales.

Por ello, ha sido necesario un proceso de aprendizaje orientado a refrescar los conocimientos sobre el tema, así como a asimilar conocimientos nuevos.

En un principio, se ha hecho una investigación sobre los sistemas de reconocimiento de gestos más utilizados [1, 2, 3, 4, 5, 8]. Al final se a optado por utilizar el algoritmo presentado por Mena et al.[2], que ofrece una clasificación efectiva con un entrenamiento rápido y poca configuración, requiriendo también poca cantidad de recursos.

3.2.2. Implementación de clasificador

En esta fase, se ha modificado una implementación del algoritmo citado ya existente en la empresa.

Las modificaciones han estado orientadas a adaptar el algoritmo propuesto para que sea utilizado con varias personas diferentes, teniendo en cuenta los diferencias de estilos que se puedan encontrar entre un usuario y otro.

Para capturar estos movimientos, se ha utilizado un acelerómetro tri-axial[9] que proporciona la información suficiente para clasificar los gestos con una mano. En la figura 7 podemos ver una representación en forma de gráfica en la que se puede ver como se recogen los datos a través del acelerómetro. En la bibliografía se citan más tecnologías de captura de movimientos, pero se ha elegido esta por su sencillez de uso y su bajo coste.

[9]Este término se refiere a la cantidad de dimensiones en las que el sensor es sensible. Por lo tanto, en este caso, el sensor detecta la aceleración producida en las tres dimensiones, X, Y y Z.

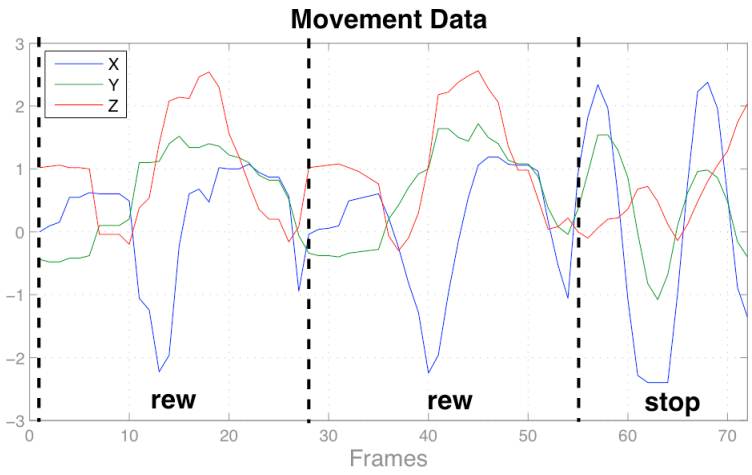

Figura 7: En esta figura se puede ver una representación en forma de gráfica de líneas de tres muestras de gestos capturadas consecutivamente. Cada una de las líneas corresponde a uno de los ejes de movimiento del acelerómetro.

En esta ocasión se han escogido 7 gestos, que se describen en la figura 8. A través de esos gestos, el usuario podrá interactuar con la animación de la escena, pudiendo poner en marcha y parar la animación, rebobinar y avanzar más rápido, pasar un solo modelo para adelante o para atrás, y volver a empezar.

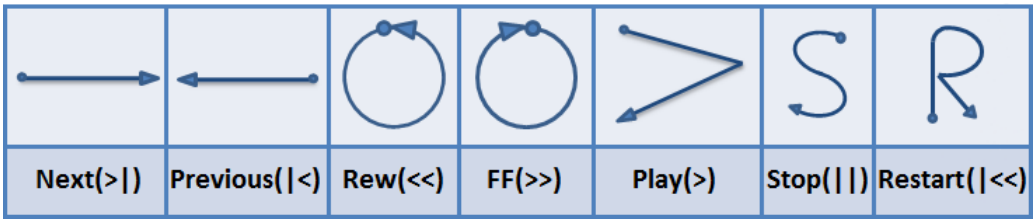

Figura 8: Lista de gestos y sus representaciones

Como ligera introducción, comentaremos brevemente el funcionamiento del algoritmo. Para el caso en el que sea necesaria una explicación más extensa, se ha añadido un anexo 5.1 en el que se explica un poco más en detalle el funcionamiento de este algoritmo.

Se carga la base de datos con gestos pregrabados y la etiqueta del gesto al que pertenece, y se almacena en memoria. Una vez tenemos la base de datos, cada gesto capturado es comparado con cada gesto almacenado. De esta comparación se obtiene un valor de distancia ponderada utilizando los datos de distancia euclídea entre las señales y su avance temporal, y que responde a la ecuación 1. El valor más pequeño de todas las comparaciones será el de aquel gesto que más se parezca al

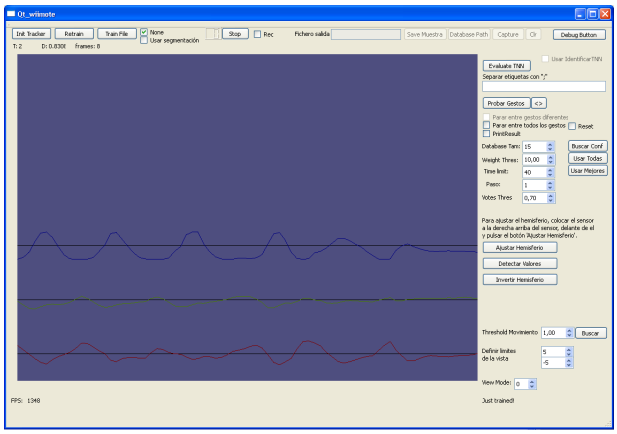

Figura 9: Captura de la aplicación de testeo

capturado. Con este valor mínimo, y utilizando un umbral de corte, se decide si se da el gesto como válido, o por lo contrario, no se parece lo suficiente como para admitir la clasificación.

$$distanciaPonderada = \frac{distancia}{avanceTemporal} \tag{1}$$

3.2.3. Desarrollo de aplicación de configuración y testeo

Para el desarrollo de este algoritmo, y para realizar las pruebas de rendimiento pertinentes, se ha desarrollado una aplicación adicional (figura 9). Esta aplicación monitoriza el funcionamiento del algoritmo, para así poder reconocer los posibles errores o problemas a la hora del desarrollo.

también se ha utilizado esta herramienta para conseguir muestras gráficas del proceso de clasificación, que han servido para entender mejor el funcionamiento interno de el algoritmo.

La captura de datos para la clasificación se ha llevado a cabo utilizando para ello el sensor tri-axial integrado en el mando auxiliar (Nunchuck) del controlador Nintendo Wiimote.

3.3. Estereoscopía y representación ajustada al usuario

Este es otro de los pilares del proyecto. Esta fase, se centra en adecuar la representación de un modelo tridimensional a la visión del espectador, teniendo en cuenta su posición respecto de la pantalla y utilizando 3D estereoscópico.

De esta manera, y como modelo de ejemplo, se ha utilizado un cubo de 9cm x 9cm x 9cm.

3.3.1. Aprendizaje e investigación (en torno a la representación estereoscópica)

La representación estereoscópica se basa en proyectar o ver con cada ojo una imagen diferente. Estas dos imágenes están capturadas de tal manera, que cuando cada ojo ve una de estas imágenes al mismo tiempo, el cerebro es capaz de interpretar la profundidad de la escena como si de una vista normal se tratase.

Estas imágenes deben de tener unas «propiedades» concretas para que el cerebro humano pueda interpretarlo correctamente. Por ello, en esta parte del desarrollo se ha investigado acerca de cuales son estas «propiedades», y como conseguir imágenes que puedan ser percibidas correctamente. El resultado de esta investigación se describe en el apartado 5.3.1 de los anexos.

3.3.2. Cálculos y ajustes de perspectiva

A la hora de representar la escena en el monitor o en la pantalla, esta es estática, y no se ajusta al movimiento del espectador. Por ello, esté donde esté el espectador, la pantalla sigue enseñando la misma perspectiva.

En esta fase, se ha desarrollado un algoritmo capaz de ajustar esta perspectiva a la posición del espectador, teniendo como dato de entrada la relación de distancias entre el espectador y el centro de la pantalla.

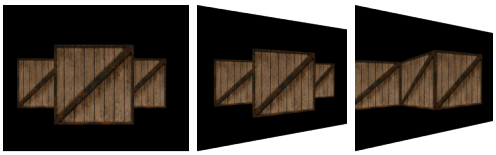

Figura 10: Imagen de varios cubos desde el frente (izquierda). Imagen de varios cubos en una pantalla vista desde un punto de vista diagonal(centro). Imagen de varios cubos en una pantalla vista desde un punto de vista diagonal con la vista ajustada(derecha).

En una representación normal, la pantalla actúa como si fuese un cuadro o una foto, que sigue siendo la misma se mire por donde se mire. Con este cambio de perspectiva, se trata de crear la ilusión de que se está mirando a través de una ventana, pudiendo ver las zonas que esconde el «marco de la ventana» moviéndonos hacia los lados o acercándonos a ella.

Respecto a esta idea, podemos ver en la figura 10, como se diferencian estos dos conceptos. En la primera imagen vemos la representación de la escena, que será la misma en las otras dos imágenes. En la segunda imagen, tenemos una representación de lo que se vería en pantalla si la viésemos desde una posición diagonal a ella, y sin adecuar la perspectiva a esta posición. En la tercera imagen, se ve la pantalla desde el mismo punto, pero esta vez con la perspectiva adecuada al espectador.

En esta última imagen se puede apreciar que se está mirando la escena desde la diagonal derecha del espectador.

Hay que decir, que en las tres la escena es la misma. No se han movido los cubos en ningún caso.

3.3.3. Desarrollo de aplicaciones de ejemplo y testeo

En este punto se ha creado una aplicación que utiliza estos algoritmos para representar una escena de prueba. Se implementa tanto el algoritmo de ajuste de perspectiva como el de estereoscopía.

Esta aplicación utiliza la librería OpenGL para representar las imágenes finales de la escena. Esta escena consta de tres cubos con textura de madera (ver figura 10).

Para comprobar la precisión de la representación se ha utilizado un cubo de las mismas dimensiones que el que se intenta representar (9cm x 9cm x 9cm). Para com-

probar la relación de perspectivas entre el mundo real y el virtual, se ha representado el cubo virtual como si estuviese justo saliendo de la pantalla, y se a colocado el cubo real justo donde se dibuja el virtual. Con el cubo real en posición, se ha observado el cubo desde diferentes perspectivas, y se ha comprobado como se ajusta la perspectiva de la representación virtual a la perspectiva real.

3.4. Implementación de aplicación de análisis meteorológico

En la última de las fases principales, se ha desarrollado una aplicación para la demostración de los algoritmos presentados. Se ha propuesto una aplicación de análisis meteorológico, que integra estos en un marco de trabajo real.

Esta implementación requiere cargar modelos muy complejos, por lo que el desarrollo de esta puede ser muy complicado si se utilizan funciones muy básicas para la representación gráfica. Por ello, se ha optado por utilizar una api gráfica de más alto nivel que OpenGL. La librería utilizada para en esta ocasión ha sido OpenSceneGraph (en adelante OSG).

OSG es una librería orientada a cargar modelos tridimensionales complejos, y da la posibilidad de definir una escena utilizando un esquema en forma de grafo. De esta manera, la representación de escenas es mucho más sencilla. Además incorpora clases de visualización y de manipulación de la escena, que ofrecen una interacción general con la escena visualizada de forma muy sencilla.

Además de su facilidad de uso, el echo de que su código sea accesible es un valor añadido que se ha tenido en cuenta.

3.4.1. Representación de la escena

Con el objetivo de que sirviese como introducción al manejo de OSG, para una primera toma de contacto, solo se ha cargado el mapa en un escenario normal. Primero cargando un cubo, se ha ido complicando la escena cargada para poder ir asimilando la forma de trabajar con la librería.

Una vez funcionan las versiones de prueba, se pasa a implementar la versión que cargará los datos de la aplicación objetivo.

La aplicación final carga un modelo tridimensional simulando el terreno de Euskadi como el que se puede ver en la primera imagen de la figura 11. Sobre este terreno, se puede ver un puntero azul que define cual es el punto de referencia actual, y al que la cámara esta mirando constantemente.

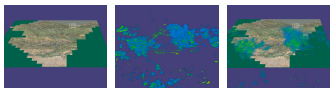

Figura 11: En esta imagen podemos ver el mapa de Euskadi (izquierda) con el puntero azul de referencia situado sobre el monte Kapildui, Álaba, y las coordenadas de este punto representadas en kilómetros. también vemos una representación de un modelo de nubes correspondiente al día 5 de julio del 2008 (centro), y una imagen de la combinación de las dos anteriores (izquierda). En esta última imagen, se les ha aplicado un grado de transparencia a las nubes para una mejor visualización.

3.4.2. Implementación de la navegación

A la hora de cargada la escena, se ha añadido un manipulador de escena genérico de OSG. Este manipulador nos brinda la posibilidad mover el objeto,acercarnos, etc utilizando el ratón.

En nuestro caso, no nos interesa utilizar el ratón como herramienta para la navegación, por lo que esta fase se ha centrado en desarrollar un manipulador de escena integrable en OSG, pero que además de la interacción típica de la librería, también nos de la posibilidad de utilizar otros controles, como en este caso del Nintendo Wiimote.

Con este manipulador personalizado podremos acercarnos y alejarnos de este punto de referencia utilizando tanto el ratón como el joystick integrado en el mando auxiliar del Wii-Remote (Nunchuck). Utilizando el acelerómetro tri-axial del mando principal o el ratón, podemos girar el mapa tomando como eje de giro el vector normal del terreno, y como punto de giro el de referencia (puntero azul). Igualmente podremos variar la inclinación del terreno rotándolo sobre el punto de referencia. Además, es posible navegar por el mapa (mover el punto de referencia) para ver en más detalle diferentes puntos del mapa.

De esta manera, podremos mirar cualquier punto sobre el mapa de la perspectiva que nos interese, para poder analizar los objetos que representemos en el.

Para ofrecer diferentes formas de navegar, se han desarrollado varias formas de navegación. La primera de ellas, que se ha denominado VIEW, representa el modelo

como si fuese una maqueta flotando en medio de la pantalla. A esta representación se le puede sumar un modelo que simula paredes y suelo de una abitación para ofrecer un contexto más realista. Esta forma de navecación esta pensada para que parezca una maqueta manipulable, pudiendo navegar por ella y hacer diferentes giros y zooms.

La otra forma de navegación, denominada FLY, simula estar flotando en el aire sobre la maqueta, con un nivel de zoom muy alto. De esta manera, se simula que el espectador se encuetra en un helicóptero, sobrevolando el modelo, del que puede mirar en torno a el con total libertad. Este modo de navegación esta orientado a analizar el comportamiento de las nubes de forma más cercana, pudiendo verlas desde debajo, colocandonos en cualquier punto del mapa.

3.4.3. Integración de dispositivos de trackeo

De forma independiente al manipulador, se ha integrado al sistema el tracker magnético Flock of Birds. Los datos de este dispositivo se utilizarán para determinar la posición del espectador respecto de la pantalla. Con estos datos, podremos alterar la perspectiva virtual para adaptarla a la perspectiva del espectador.

Una vez integrado el *tracker*, se diseñado e implementado un sistema de adaptación de perspectiva para la adaptación de esta.

3.4.4. Representación estéreo

OSG integra funciones para pasar de una representación convencional a una representación estereoscópica.

Al tener la posición de la cámara virtual representada en el mundo, se han utilizado las funciones internas de OSG para convertir esta en una representación 3D.

En este punto, se ha trabajado en que la representación final tuviese las propiedades necesarias para que el espectador pudiese percibir la profundidad de la imagen de una forma realista.

Como trabajo previo a la implementación de esta funcionalidad se ha realizado un pequeño análisis sobre cuales son los requisitos que deben tener este tipo de implementaciones, cuyas conclusiones se presentarán en el apartado de conclusiones.

3.4.5. Cargar datos meteorológicos

Utilizando varias aplicaciones existentes en la empresa, es posible crear representaciones tridimensionales de los datos de una estación meteorológica. En este caso, se utilizan los datos recogidos a través de un radar Dopler situado en el monte Kapildui, en la provincia de Araba. Un ejemplo de este modelo puede verse en la segunda imagen de la figura 11.

Al modelo del terreno ya cargado se le añaden los modelos de nubes (tercera imagen de la figura 11). Estos modelos de nubes son calculados con anterioridad, y cargados en la aplicación a la hora de iniciarla. En tiempo de ejecución es posible cambiar de modelo de nubes, pudiendo ver una representación gráfica de las nubes de un día concreto.

Evidentemente, las coordenadas en las que está renderizado el terreno tienen relación directa con las coordenadas en las que está renderizadas las nubes. De esta manera, los datos obtenidos de la estación meteorológica se posicionan en el espacio virtual justo donde se posiciona en el mundo real.

Al cargar los modelos de las nubes, se sigue un orden cronológico. Gracias a esto, cargando como primer modelo los datos de un día específico, podemos hacer que las representaciones de las nubes vayan pasando una detrás de otra como si fuese un pase de diapositivas, o una animación, viendo la progresión que han tenido las nubes durante un periodo de tiempo.

En esta parte de la fase de desarrollo, se han implementado las funcionalidades de animación y carga de modelos, posibilitando la interacción con la animación a través del teclado. Las posibles interacciones con la animación se listaron en el apartado 3.2.2, ya que en el próximo hito de esta fase es combinar esta interacción con el reconocimiento de gestos.

3.4.6. Integración de reconocimiento de gestos

En este punto, el objetivo ha sido integrar el reconocimiento de gestos con la interacción con la animación.

A través del interfaz por gestos, es posible controlar el cambio de modelos, controlando así la animación de las nubes. De esta manera, podemos interactuar con la animación como si se tratase de un reproductor de vídeo doméstico.

Es posible controlar estas animaciones, pasándolas adelante, atrás, rebobinando la animación, o pasando los modelos a mayor velocidad.

De esta manera, se crea una forma de interacción más intuitiva, facilitando el uso de la aplicación.

3.4.7. Documentación

4. Conclusiones

En muchas áreas de la industria, es interesante tener un entorno de representación en el que el nivel de detalle de lo representado sea mayor que una simple proyección en una pantalla. En aplicaciones en las que la percepción de la profundidad es un aspecto clave, nos puede interesar tener representaciones que expresen este rasgo, para poder percibir mejor lo se intenta representar.

Hasta ahora se han utilizado maquetas y réplicas a escala modeladas con diversos materiales para poder representar esta profundidad de forma adecuada. No obstante, esto requiere un gasto extra, y un trabajo adicional. Además, no es una herramienta con la que se pueda interactuar de forma activa.

Hoy en día, existen tecnologías que nos permiten representar esta profundidad de forma más interactiva y dinámica, a través de la visión 3D y otras tecnologías de interacción humano-máquina.

Estas representaciones se consiguen en centésimas de segundo, por lo que es posible realizar cambios de propiedades, animaciones, re-estructuraciones o cambiar distribuciones y escalas en tiempo real.

Este proyecto, se ha trabajado sobre diversas tecnologías que pueden ayudar a generar aplicaciones como la aquí descrita, pudiendo interactuar con una maqueta tridimensional de forma interactiva a través de gestos.

En los próximos apartados se comentaran, en forma de conclusiones, los resultados obtenidos, así como los principales problemas a lo largo del desarrollo. Además, se plantearan las impresiones personales del trabajo terminado, y se propondrán nuevos caminos a seguir para mejorar o desarrollar más aun la aplicación y el contexto en el que está.

4.1. Resultados obtenidos

Es el momento de hablar de cuales han sido los resultados del desarrollo.

Este capítulo está dividido en varias partes. Primero hablaremos de cuales han sido los resultados obtenidos a la hora de clasificar los gestos propuestos y su rendimiento

a la hora de realizar la clasificación. También haremos un repaso de como a resultado la implementación de las funciones de visualización.

4.1.1. Reconocimiento de gestos

En la parte de reconocimiento de gestos se han realizado pruebas de rendimiento a la aplicación desarrollada.

Como set de datos, se han grabado muestras de los diferentes gestos de 5 personas, que han realizado 20 repeticiones por gesto. Los gestos capturados son los propuestos en el apartado 3.2.2 de este documento. Como tenemos 7 gestos, la base de conocimiento general consta de 700 gestos (5 personas x 7 gestos x 20 repeticiones = 100 muestras). Las muestras se han redimensionado a 15 frames por gesto para la base de datos, pues este valor ha dado resultados satisfactorios. La segmentación de cada gesto se ha llevado a cabo pulsando un botón mientras se realizaba el gesto, por lo que el sistema almacena los datos solo mientras este botón está pulsado.

Para determinar el rendimiento del clasificador, se ha utilizado el método '*Left One Out*', al igual que en otras publicaciones como [6, 2]. El ratio de acierto en el proceso de pruebas es superior al 95 % en las pruebas por persona. En las pruebas por gesto, este valor supera el 96 % de acierto. La figura 12 representa los resultados de los experimentos.

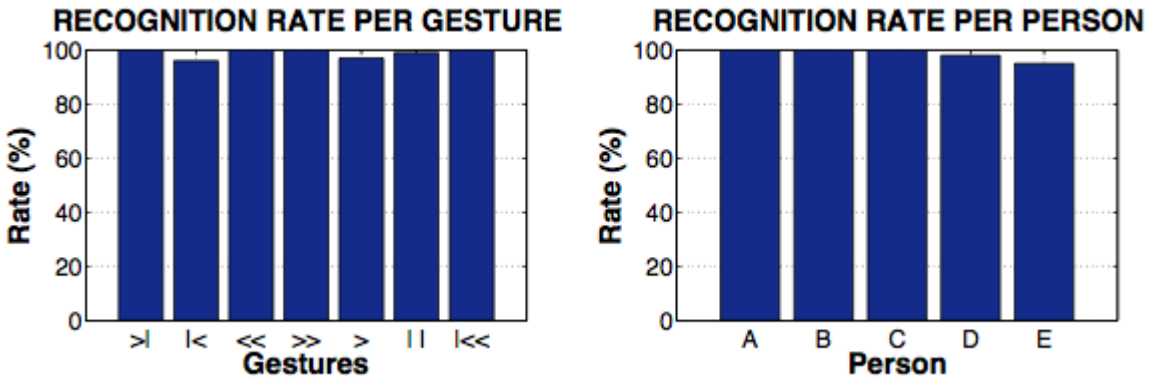

Figura 12: En la primera gráfica (izquierda) se representan los resultados de las pruebas realizadas en orden de gestos diferentes. En la segunda (derecha), se hace un análisis del reconocimiento por cada usuario

Figura 13: Imagen de la escena en modo maqueta, y con las paredes visibles (izquierda). Misma imagen pero con una perspectiva de lado (centro). Imagen de la escena en 3D anaglífico (derecha).

4.1.2. Representación Gráfica

Dentro de la representación gráfica, se definen dos objetivos tecnológicos diferentes. Uno de estos lo representa el ajuste de la perspectiva a la posición del espectador en tiempo real.

Para ajustar la perspectiva se ha desarrollado un algoritmo que, recogiendo como entrada el punto en el espacio en el que está situado el espectador, consigue reconfigurar la perspectiva de la cámara virtual de forma que la representación crea la ilusión de que la pantalla es una «ventana» al mundo virtual.

La aplicación del algoritmo de ajuste de perspectiva a demostrado la validez de los cálculos propuestos en el documento. La perspectiva se adecua perfectamente a la posición del espectador, siendo la experiencia del usuario muy realista.

En el anexo 5.2 se hace un análisis más extenso del cálculo de esta perspectiva, y hace una introducción a los diferentes aspectos a tener en cuenta a la hora de ajustar la perspectiva de esta manera.

En representaciones en las que solo se observa la maqueta suspendida en el aire, algunos usuarios han encontrado problemas a la hora de percibir el objeto. Esto se puede evitar añadiendo un contexto adecuado, como el de representar la escena con paredes y suelo. Se pueden ver una captura de esta configuración en la figura 13

La representación estereoscópica por su parte, ha funcionado de la manera esperada en la implementación escogida. En todo momento se representa la profundidad de manera perceptible, y con una profundidad de campo[10] muy extensa. Incluso con tamaños del mapa en el que la parte más profunda de la representación se encuentra muy lejana, la profundidad sigue siendo perceptible sin que se requieran grandes esfuerzos visuales.

A la hora de calcular las imágenes a representar, se han tenido que tomar ciertos detalles en cuenta. Estos detalles son la clave para que la percepción de la profun-

[10] El concepto «profundidad de campo» se refiere a la zona en la cual la imagen captada por el objetivo es nítida (es decir enfocada), de manera que en la imagen resultante, las personas y objetos que se encuentren dentro de esa zona aparecerán también nítidos.

didad sea posible, y que siendo esta posible, no produzca inconvenientes (mareos, dolor de cabeza, vista cansada, etc.). Una explicación de lo que es necesario tener en cuenta se encuentra en la sección 5.3, en el apartado de anexos. Además, en los apartados 5.3.2 y 5.3.3, se desarrolla más extensamente cuales son las bases en las que se apoya el 3D estereoscópico, y la forma de capturar imágenes para poder representarlas con profundidad perceptible.

4.1.3. Implementación utilizada para la representación estéreo

Para la representación, se calcula la posición virtual en la que estaría el espectador. Desde esta posición, y con una disparidad horizontal de 6 cm se renderizan dos imágenes, que después serán representadas por pantalla. Estas imágenes se representan en la pantalla una al lado de la otra. El resultado en una pantalla normal se vería parecido a lo expuesto en la figura 14.

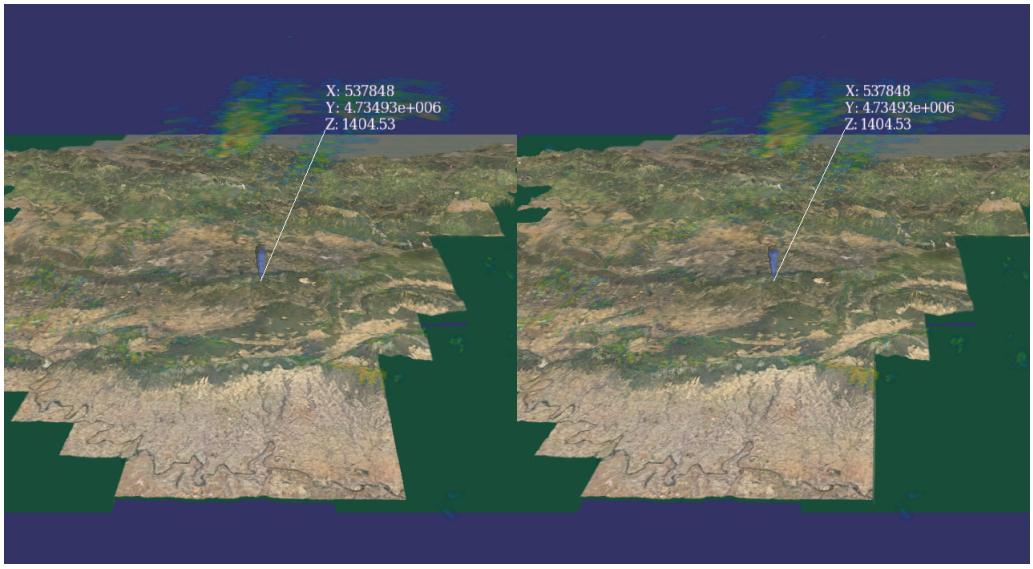

Figura 14: Representación de las dos imágenes renderizadas en una pantalla normal, configuradas con un split horizontal

Teniendo las dos imágenes existen diferentes formas de representarlas para que ofrezcan una sensación de profundidad. La escogida para el proyecto utilizar filtros y gafas polarizadas, como se ha explicado en el apartado 2.1.2 de este documento.

Para obtener las imágenes, se ha optado por utilizar un ordenador con dos tarjetas gráficas independientes. Se ha configurado el escritorio para que utilice las dos salidas, y que lo represente como escritorio extendido. De esta forma, el escritorio se

estira hasta ocupar el tamaño de dos pantallas, estirando también las ventanas de las aplicaciones a pantalla completa.

La salida de cada tarjeta se ha conectado a un cañón proyector, que proyectan la imagen en la misma pantalla, solapando las dos imágenes. Para mandar una imagen a cada ojo, y como se explicó en el capítulo 2.1.2, se utilizan filtros polarizados. El primer filtrado se realiza entre el cañón proyector y la pantalla, poniendo un filtro polarizado por el que pasa la imagen, con una diferencia de 90º entre el filtro de cada cañón. El segundo filtrado se realiza utilizando gafas con lentes polarizadas, también desfasadas 90º entre sí, coincidiendo la dirección de la polarización con la utilizada para filtrar la imagen de cada ojo (orientación del filtro del ojo derecho = orientación del filtro de la imagen para el ojo derecho, y viceversa).

En la figura 15 podemos ver una representación esquemática del montaje final.

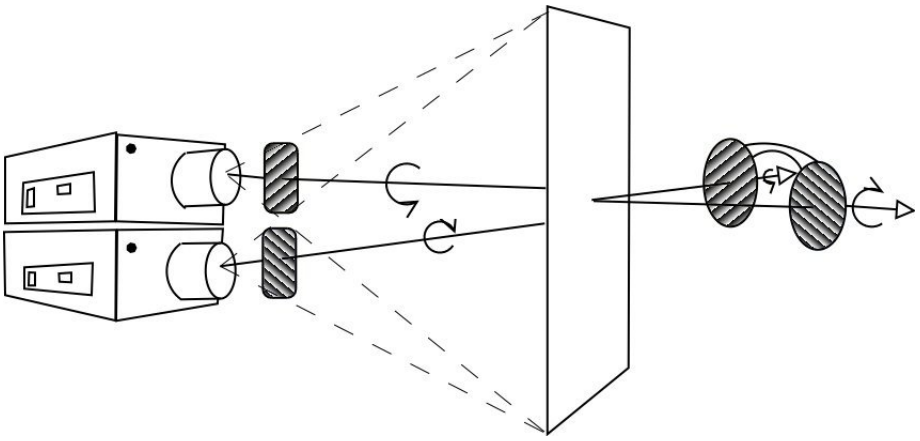

Figura 15: Esquema que representa el montaje final del sistema de proyección

4.2. Problemas durante el desarrollo

A lo largo del proyecto se han encontrado diferentes tipos de problemas en las diferentes áreas. Algunos de estos problemas han sido solventados, y otros se han analizado para una posible futura revisión.

4.2.1. Problemas con la representación 3D

El mayor problema ha sido el de relacionar el mundo físico con el mundo virtual.

Para interactuar con el mundo que se encuentra al otro lado de la pantalla utilizamos mandos, controles o capturadores de movimiento, que capturan nuestras pulsaciones o movimientos y los transforman de alguna manera en acciones.

El problema reside, que cuando se trata de convertir movimientos en el mundo real a movimientos en el mundo virtual, existe una barrera que se es necesario de atravesar. Esta barrera la componen las diferencias entre los sistemas de referencia del mundo real y el mundo virtual.

Esto que puede parecer de poca importancia, es vital para poder interactuar de forma correcta, porque si nosotros nos movemos hacia delante (si utilizamos coordenadas cartesianas para definir el espacio, esto sería por ejemplo avanzar en el eje X), el sistema del mundo virtual puede creer que nos movemos hacia arriba (en el caso de que el sistema de coordenadas del mundo virtual tuviese el eje X como componente vertical).

Para ello es necesario transformar los sistemas de referencia del sistema de captura al sistema de referencia del mundo virtual.

Un posible proceso para realizar este cambio se desarrolla en el punto 5.4 del apartado de anexos.

4.2.2. Problemas con el estéreo

La representación estereoscópica ofrece varios problemas no evidentes, que pueden entorpecer la representación de la profundidad.

Uno de estos problemas es la distancia entre la cámara y la representación de la escena. En este documento se detalla una forma de relacionar la distancia de separación escena-cámara del mundo real con esta misma distancia en el mundo virtual (ver sección 5.3.3).

Esto es posible cuando se tiene un punto u objeto al que se quiere mirar. En nuestro caso, este objeto observado es la maqueta, pero es importante saber cual es el punto de la maqueta que se quiere observar, ya que si la distancia es muy grande entre diferentes puntos de la maqueta, esto podría perturbar la visualización.

Lo mismo pasa si en vez de estar observando la maqueta desde un punto lejano, configuramos la vista para navegar por la maqueta como si fuésemos en un

helicóptero. En este caso, podemos decidir utilizar el suelo como punto observado, pero según inclinemos la vista hacia arriba, y nos acerquemos más al horizonte esta distancia aumenta considerablemente.

Este problema puede ser trivial en entornos donde lo que importa es solo representar la profundidad, pero si lo que necesitamos es analizar visualmente un objeto suspendido en el aire (en este caso las nubes) el control de este tipo de distancias es importante.

Para analizar este problema se han creado dos modos de visualización diferentes. Por una parte, en el modo normal, la escena se describe como una maqueta suspendida en el aire, a la que podemos acercarnos o de la que podemos alejarnos. En el segundo modo, la cámara actúa como un helicóptero ficticio, que nos da la libertad de mirar a cualquier dirección desde un punto en el aire, e incluso desplazarnos por la maqueta.

En el primer modo, se ha optado por marcar un punto de referencia en el mapa (como se ha explicado anteriormente con un puntero azul). La distancia espectador pantalla será reflejada en el mundo virtual como la distancia entre la cámara y este punto. Moviendo el puntero, podremos hacer que cualquier punto del mapa sea el punto de referencia.

En el segundo modo se ha comprobado que utilizar el suelo como punto de referencia para la distancia cámara-objeto funciona satisfactoriamente, siempre y cuando sea posible controlar tanto la altura como la posición de la cámara para poder ajustar la vista al objeto que queramos analizar. Para que la distancia no sea excesiva cuando miramos más allá del horizonte, configuramos una distancia máxima, que será de 30 kilómetros en nuestro caso.

Por otra parte, la adaptación adecuada de estas distancias entre el mundo real y el mundo virtual también han sido un problema. En la implementación realizada, se ha optado por generar un frustum de perspectiva que siguiese la relación expuesta en el apartado 5.3.3 del anexo. Una vez obtenida esta perspectiva, si queremos hacer zoom para aumentar el nivel de detalle, escalamos el frustum para obtener una perspectiva con la misma relación que la original. Este método ofrece resultados muy satisfactorios, y no es necesario calcular la relación cuando se hace zoom.

4.2.3. Problemas con reconocimiento de gestos

Uno de los mayores problemas del reconocimiento de gestos es la segmentación.

La segmentación consiste en diferenciar los frames que pertenecen al gesto, de los que pertenecen a otro gesto o a ninguno[11]. En este caso, se ha segmentado el gesto manualmente, utilizando un botón como indicador de principio y final del gesto. De esta manera, el sistema capturara los frames solo cuando el gesto esté pulsado, e interpretará la pulsación como comienzo de gesto y el soltar el botón como final del mismo.

Por otra parte, el rendimiento de la máquina también a resultado ser un problema a la hora de recoger los movimientos que componen el gesto. Esto es porque las animaciones de las nubes requieren una gran carga de trabajo para los procesadores de la máquina (CPU y GPU) y esto puede llegar a interferir en el tiempo entre lecturas de los frames para detectar el gesto.

Para solucionar esto, se ha disgregado el trabajo de captura y el de renderizado, creando un hilo de ejecución diferente apara la captura de los datos del gesto, dándole a este prioridad sobre el procesado de las imágenes.

4.3. Conclusiones finales

En este apartado se comentarán las conclusiones de forma más personal, incluyendo interpretaciones del trabajo realizado.

4.3.1. Aplicación desarrollada

La versión final de la aplicación representa las lecturas obtenidas por un radar Dopler en forma de modelo tridimensional, sobre un mapa en 3D. Es necesario comentar también, que ambas representaciones tridimensionales están modeladas utilizando para ello coordenadas reales, lo que quiere decir que las coordenadas tanto del mapa como de las nubes, coinciden en el mundo virtual y en el mundo real. De esta manera,

[11] Los movimientos que no pertenecen a ningún gesto, denominado también ruido, se generan cuando estamos preparando el gesto (levantar la mano para decir adiós, cuando el levantar la mano no es parte del gesto, sino el posterior balanceo lateral de la mano), o cuando nos movemos para hacer algo que no es un gesto (rascarnos la nariz, por ejemplo).

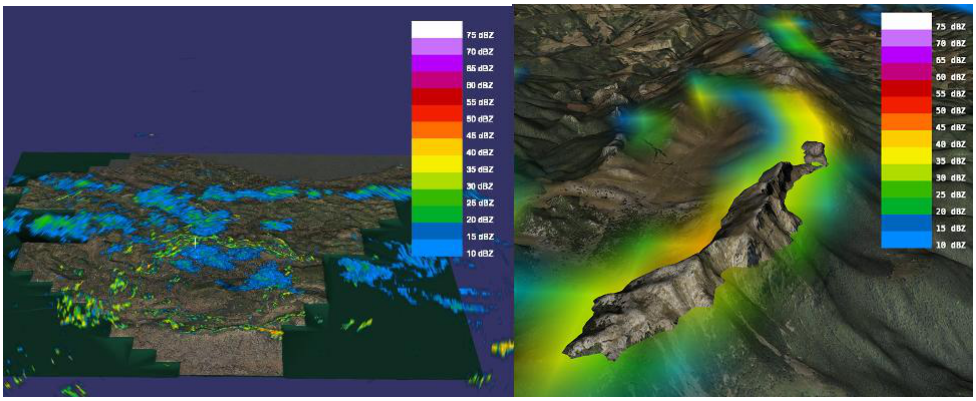

Figura 16: Captura de la aplicación representando mapa y nubes (izquierda). Zoom sobre zona en la que se puede ver una lectura correspondiente a una colisión con el terreno (derecha).

si el radar está situado en el monte Kapildui, y este tiene las coordenadas en kilómetros (X, Y, Z), el mapa representará este monte en las coordenadas virtuales (X, Y, Z). Esto mismo sucede con las coordenadas de las nubes.

De esta manera, los dos modelos (mapa y nubes) se relacionan en el mundo virtual al igual que lo harían en el mundo real. Este dato es interesante, porque a la hora de hacer la implementación final, se han conseguido lecturas que no corresponden a nubes, sino a colisiones no controladas con el terreno.

Un ejemplo de esto lo podemos ver en la figura 16. En esta podemos ver claramente como esta lectura no corresponde a la de una nube ni precipitación, sino a un monte que cruza la zona de barrido del radar. Si la imagen estática puede dejar dudas de si pueden ser bancos de nubes acumulados al rededor del monte, al poner en marcha la animación se ve perfectamente que esta lectura es constante, por lo que se puede afirmar que es una colisión.

Este es un perfecto ejemplo de lo que una aplicación de este tipo aporta a los procesos de análisis, ya que no es necesario ser experto para percibir ciertos detalles que de otra manera serían muy difíciles de identificar.

4.3.2. Conclusiones en torno al desarrollo

Los entornos virtuales como este, con propiedades inmersivas, mejoran considerablemente la experiencia de usuario en entornos donde lo representado son escenas

o modelos tridimensionales. De esta manera, la perspectiva tridimensional aporta un valor añadido que es muy importante en entornos analíticos.

En la empresa actual, este tipo de aplicaciones ofrecen grandes ventajas para diseñadores, tanto arquitectos como diseñadores industriales a la ora de presentar sus trabajos, gracias a que se pueden realizar los cambios in-situ y ofrecer diferentes perspectivas de un mismo modelo. también pueden ser de especial interés en marcos de trabajo orientados a investigación de todo tipo desde, biólogos para realizar simulaciones de comportamientos microscópicos, hasta meteorólogos, pasando por disciplinas como la medicina para representar escáneres tridimensionales, roturas, o la exploración espacial.

Por ejemplo, en la figura 2 se puede ver una imagen en 3D anaglífico, que representa una hoja de Alchemilla ampliada 500 veces (a través de unas gafas anaglíficas es posible percibir la profundidad de esta imagen). De esta manera, podemos percibir de una forma mucho mejor el resultado de una imagen aumentada, siendo más perceptible para cualquier persona la composición microscópica de lo analizado.

Por su parte, la interfaz basada en gestos es muy útil a la hora de interactuar con el sistema. Esto nos da la posibilidad de prestar atención a la representación sin que la búsqueda de controles (como botones en el teclado) desvíe nuestra atención del modelo.

4.3.3. Conclusiones personales

4.4. Líneas futuras

La interfaz de gestos funciona a base de acelerómetros. Esto requiere que el usuario tenga que utilizar dispositivos en los que van montados estos sensores. Además de esto, es necesario pulsar un botón para definir el comienzo del gesto, y soltarlo para definir el final del mismo.

Para solventar el primer problema, se recomienda pasar de un sistema de captura de movimiento basado en acelerómetros, a uno basado en visión-artificial, con la que el usuario no necesita de ningún tipo de dispositivo.

La definición del comienzo y el final del gesto, conocida en la bibliografía como segmentación del gesto, se podría hacer automáticamente, utilizando algoritmos de

segmentación de gestos, evadiendo así el problema de pulsar un botón. también sería conveniente utilizar algoritmos de filtrado para mejorar la captura de datos, sobre todo a la hora de entrenar el sistema, ya que, aunque los resultados obtenidos son más que satisfactorios, la precisión de la clasificación podría mejorar considerablemente.

En la parte visual, también es posible cambiar de un sistema de detección utilizando campos magnéticos a uno utilizando visión artificial. Esto también evitaría que el espectador tuviese que llevar con sigo el sensor del tracker.

La visión artificial es un campo que avanza muy deprisa, y aunque su precisión es aun menor que, por ejemplo, en el caso del seguimiento por campos magnéticos, esto es un aspecto que está siendo mejorado. Por otra parte, estos algoritmos requieren de mucho procesamiento, por lo que sería necesario tener una máquina más potente, o en su defecto, utilizar una segunda máquina exclusivamente para manejar estos algoritmos.

A pesar de esto, trabajos actuales han demostrado el buen funcionamiento de estas técnicas en aplicaciones varias.

Referencias

[1] Maria Karam. A taxonomy of Gestures in Human Computer Interaction. pages 1–45.

[2] Oscar Mena, Luis Unzueta, Basilio Sierra, and Luis Matey. Temporal Nearest End-Effectors for Real-Time Full-Body Human Actions Recognition. *Proceedings of the IEEE*, pages 269–278.

[3] S. Mitra and T. Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(3):311–324, 2007.

[4] G. Rigoll, A. Kosmala, and S. Eickeler. High Performance Real-Time Gesture Recognition Using Hidden Marcov Models. *Gesture and sign language in human-computer interaction*, pages 69–80, 1997.

[5] Thomas Schl, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture Recognition with a Wii Controller. *Gesture*, pages 18–21, 2008.

[6] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a Wii controller. *Proceedings of the 2nd international conference on Tangible and embedded interaction - TEI '08*, page 11, 2008.

[7] Ian Stavness, Billy Lam, and Sidney Fels. pCubee : A Perspective-Corrected Handheld Cubic Display. *Design*, 2010.

[8] Thomas Stiefmeier, Daniel Roggen, and Gerhard Tröster. Gestures are Strings : Efficient Online Gesture Spotting and Classification using String Matching. *Training*.

5. Anexos

En esta sección final se incluyen los apartados que, siendo parte del proyecto, tienen un nivel técnico y de detalle superior al que requiere la mayor parte del documento presentado.

Por ello, y a titulo introductorio, se listará cuales son los apartados incluidos en este capítulo:

- Algoritmo de clasificación
  - Fase de entrenamiento
  - Fase de clasificación
  - Fase de decisión
- Adaptación de la perspectiva al espectador
  - Definiendo la perspectiva
  - Calcular la perspectiva del espectador
  - Ecuaciones para calcular la perspectiva
- Implementación del 3D estereoscópico
  - Conceptos generales
  - Representación estereoscópica
  - Conseguir las imágenes
- Relación entre el mundo real y el mundo virtual

5.1. Algoritmo de clasificación

En este proyecto, y para la clasificación, se ha escogido el un algoritmo de reconocimiento y clasificación de gestos sencillo, y que no consume demasiado recurso. El algoritmo escogido se presentó en el trabajo de Mena et al. [2] titulado *Temporal Nearest End- Effectors for Real-Time Full-Body Human Actions Recognition*, en el que se describe su funcionamiento. No obstante, para representar el funcionamiento, se dará un repaso de este sistema de clasificación.

En los siguientes apartados se entra en más detalle en lo expuesto en este último párrafo.

5.1.1. Fase de entrenamiento

Este clasificador requiere de una base de conocimiento inicializado previamente. Para esta fase, se tiene una cantidad n de muestras pre-grabadas, y se especifica a que gesto pertenece cada muestra. El entrenador, lee todas las muestras, y las almacena en un objeto en memoria recordando a que gesto pertenece cada muestra, para que el proceso de clasificación sea más rápido.

Antes de almacenar cada muestra, esta se redimensiona para que todas las muestras tengan la misma cantidad de frames. Para el proceso de redimensión se utiliza un '*Spline Cúbico*'. Utilizando esta función matemática, conseguimos reducir la cantidad de frames de una señal sin que esta apenas se deteriore.

5.1.2. Fase de clasificación

La clasificación se basa en buscar coincidencias de avance temporal entre dos señales, una de ellas, la capturada en el momento de realizar el gesto, y otra, una de la serie de ejemplos almacenados en la base de datos procesada anteriormente. Esta búsqueda devolverá como resultado un índice de distancia entre ambas, en el que nos basaremos para determinar cual de las muestras de la base de datos coincide mejor con el gesto.

Un gesto está representado por una serie de lecturas a lo largo del tiempo (en nuestro caso, una cada 30 milisegundos más o menos), y podemos decir que cada

una de estas lecturas (en adelante frames), será un valor en un momento determinado. Pondremos como ejemplos de gesto almacenado las señales [A-B-C-D-E-F-G] y [A-B-C-D-C-B-A]. Cada uno de los caracteres representa una lectura diferente, y están ordenados cronológicamente.

De la misma manera podemos definir una señal de gesto capturada al momento como (A-C-B-D-E-F-G). Comparemos la señal capturada con la primera de la base de datos.

En el primer caso, los estados coinciden ($A = A$) en la primera posición, por lo que decimos que existe coincidencia en la posición '0'. Buscamos el segundo estado de la muestra (C) con la misma señal de la base de datos, y esta se encuentra en la tercera posición. Como la coincidencia anterior estaba en la posición 0 y esta está en la posición 2 (y $2 > 0$), deducimos que, aunque no de una manera idéntica, ambas señales avanzan igual (o al menos de manera parecida). Como avanzan igual, sumamos uno al contador de avances temporales coincidentes.

En la siguiente iteración, en este caso, buscamos el tercer estado de la muestra recién capturada (B) con la primera señal almacenada. Vemos que la coincidencia se encuentra en la posición 1, que en este caso es menor a la de la anterior lectura, que estaba en la posición 2. Ambas señales avanzan de manera diferente esta vez, por lo que no sumamos ninguna coincidencia esta vez.

El algoritmo continua de esta manera hasta cotejar la muestra recién capturada con todas las almacenadas en la base de datos. De esta manera, conseguimos un valor de avance temporal entre esta señal y cada una de las almacenadas.

Volviendo a las señales de ejemplo propuestas, si las comparamos con la muestra capturada, la primera devolvería un valor de temporalidad de 6 ($A+C+D+E+F+G$), mientras que la segunda, devolvería un valor de 3 ($A + C + D$).

Además de este valor, se mide la distancia euclídea entre ambas señales (capturada y almacenada), y cuando se tienen ambas, se ponderan (como describe la ecuación 2), para obtener un valor que combine estas dos lecturas.

$$distanciaPonderada = \frac{distancia}{avanceTemporal} \tag{2}$$

La señal que menor distancia ponderada de como resultado será la que se tomará como la coincidente.

5.1.3. Fase de decisión

Para saber si la señal es realmente coincidente, o simplemente ruido, se define un valor de corte. Si la distancia ponderada es mayor que este valor, será tratado como desconocido.

Este valor de corte se configura dependiendo de factores como tamaño de las señales en la base de datos, amplitud máxima y mínima de las señales, y otros varios valores. también es posible definirlo por experimentación.

5.2. Adaptación de la perspectiva al espectador

Si representamos una escena en una pantalla o en un monitor plano, la vista representada es estática, por lo que puede compararse con un cuadro o una foto, que también son estáticas. De esta manera, si el espectador se mueve, la imagen seguirá siendo la misma, aunque la perspectiva del espectador sobre la imagen (ahora la ve desde un lado) no es la misma.

Figura 17: Imagen de varios cubos desde el frente (izquierda). La misma representación vista desde un punto de vista diagonal(derecha).

Si representamos una escena utilizando 3D estereoscópico, la visión tridimensional aporta una componente de profundidad, pero cuando el espectador se mueve, sigue viendo la misma imagen. Por esta razón, podemos decir que la perspectiva sigue siendo estática, y el estéreo no le aporta dinamismo.

Para entender este concepto, podemos poner como ejemplo la representación de un cubo que sobresale de la pantalla. Imanemos en frente de un monitor, con las gafas polarizadas viendo una escena en la que aparece un cubo de frente. Este cubo parece estar saliendo de la pantalla hacia nosotros. Ahora, si nos desplazamos hacia nuestra izquierda, nuestra mente espera ver el costado izquierdo del cubo, con otro lado de este visible. No obstante, seguimos viendo la misma cara frontal del cubo, y aunque sigue habiendo profundidad, la perspectiva no es realista.

Esto funciona bien en sistemas en los que el espectador está quieto (cine, televisión, etc...), pero en ambientes más interactivos (vídeo-juegos, representaciones analíticas para medicina, ingeniería, etc...) la percepción de profundidad puede no ser suficiente.

Con el sistema que se presenta, se busca que cuando el espectador se mueva hacia su izquierda, vea parte de la cara izquierda del cubo tal y como la vería si el cubo estuviese en esa posición.

Figura 18: Imagen de varios cubos desde el frente (izquierda), seguida de la misma escena con la perspectiva adaptada al desplazamiento del usuario hacia la izquierda (centro) y la misma representación vista desde un punto de vista diagonal (derecha)

Para esto, el sistema debe ser sensible a la posición del espectador en todo momento, y si está a un lado de la pantalla, se debe representar el lado de la representación que vería el espectador. De la misma manera, si el espectador viese algo que está por detrás de la pantalla y quisiese ver lo que supuestamente esconde el marco del monitor, al moverse podría ir viendo lo que se esconde, como si se tratase de una ventana.

Este tipo de cambios de perspectiva han demostrado dar una sensación de profundidad suficientemente convincente como para no tener que utilizar sistemas estereoscópicos [7].

A continuación veremos como podemos calcular la perspectiva del espectador, y como podemos implementarla. En las siguientes líneas de este capítulo, y a no ser que se especifique lo contrario, no trataremos con representaciones estereoscópicas, ya que esto añadiría complejidad innecesaria al concepto de perspectiva.

5.2.1. Definiendo la perspectiva

En nuestro caso, el termino perspectiva se refiere a la forma que tiene el espectador de ver una escena concreta desde un punto conocido del espacio. En el mundo real, cuando observamos algo, tenemos un campo de visión concreto, fuera de lo cual no vemos. Además, el punto desde el que observamos es determinante, ya que las luces, las formas, y lo que vemos esta determinado por desde donde, y hacia donde miramos.

Tanto en fotografía como en diseño 3D, se define la perspectiva de una cámara a través de una figura geométrica denominada *frustum*.

El *frustum* es una figura similar a una pirámide con base cuadrada, pero con la punta cortada. Como se puede ver en la figura 19, solo lo que este dentro del volumen generado por esta figura será visto por la cámara.

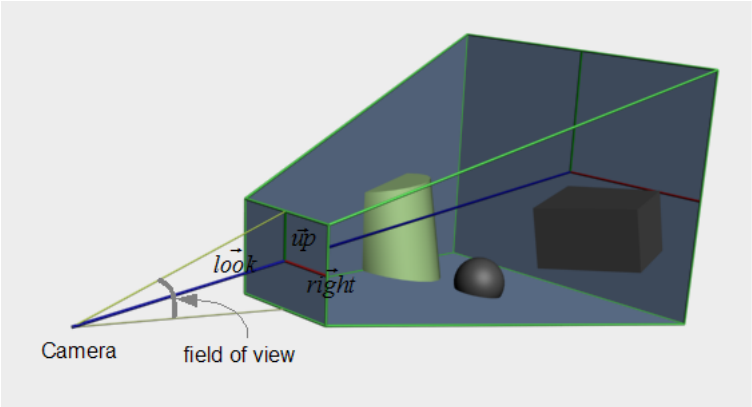

Figura 19: Representación del frustum de una escena virtual, y lo que estaría dentro de esta

Existen dos tipos de frustum, el simétrico y el asimétrico (ver figura 20). El tipo utilizado normalmente es el simétrico, en el que el ángulo formado por una pared y la base es en todos los casos el mismo. No obstante, en nuestro caso queremos simular un cambio de perspectiva, y para eso podemos alterar esta figura, generando un frustum asimétrico.

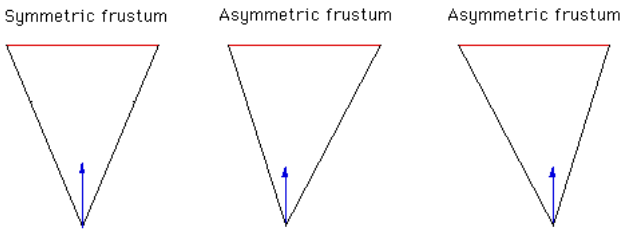

Figura 20: Representación visual de un frustum simétrico (izquierda) y dos frustum asimétricos (centro y derecha)

Alterando de esta manera la perspectiva, podemos hacer que la representación en pantalla actúe como si fuese una ventana en vez de un cuadro o foto estática. De esta forma, al cambiar la posición del espectador, la parte de la escena que el espectador ve es diferente. Por esto, si queremos ver algo que, supuestamente, está detrás de la pantalla a la derecha pero la pantalla es muy estrecha para verlo, podemos movernos hacia la izquierda para intentar desplazar el campo de visión hasta que entre (ver figura 21).

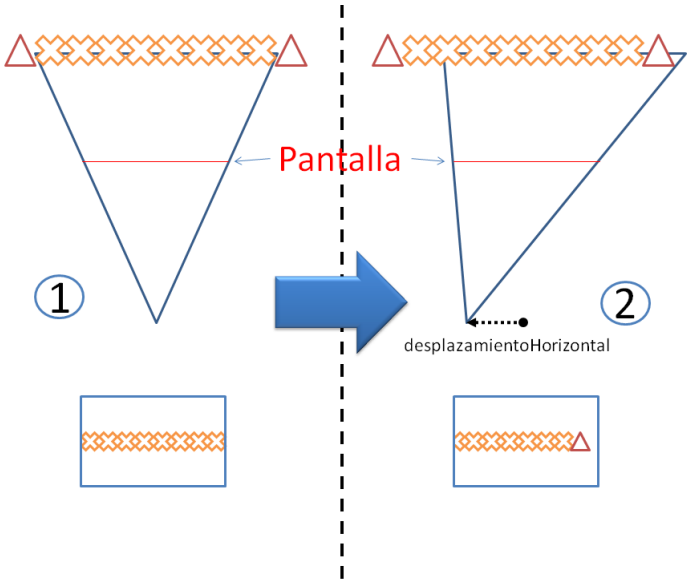

Figura 21: En la imagen 1 vemos el esquema de la perspectiva de un espectador que estaría delante de la pantalla, y debajo vemos una representación de lo que se vería en la pantalla. En la imagen 2 el espectador se desplaza hacia la derecha, y vemos una representación de lo que cambia en la perspectiva. En este segundo caso, vemos debajo, como la figura triangular que en el primero quedaba fuera de la pantalla, es ahora visible.

5.2.2. Calcular la perspectiva del espectador

Existen varias maneras de definir un frustum. Por ejemplo, para definir la forma simétrica, podemos hacerlo especificando el ángulo de apertura en horizontal y en vertical, seguido de la distancia entre el espectador (o la cámara) y el fondo de la figura. En cambio, para la forma asimétrica, necesitamos definir cada uno de los lados por separado.

En nuestro caso, utilizamos la librería gráfica OpenGL como soporte para el renderizado de la escena. Esta librería provee de funciones para el cálculo, almacenamiento y representación de la perspectiva de la cámara.

Con OpenGL, podemos definir el frustum utilizando seis datos: coordenadas del plano superior (*top*), inferior (*bottom*), izquierdo (*left*) y derecho (*right*), distancia al plano cercano (*near*) y distancia al plano lejano (*far*). Podemos ver un esquema de estos datos en la figura 22. Nótese que los datos *near* y *far* son escogidos, y no calculados. Por ello, estos dos datos son conocidos en todo momento.

Por su parte, la librería se encarga de guardar los datos en forma de matriz, para después aplicarla cuando se calcule la escena final.

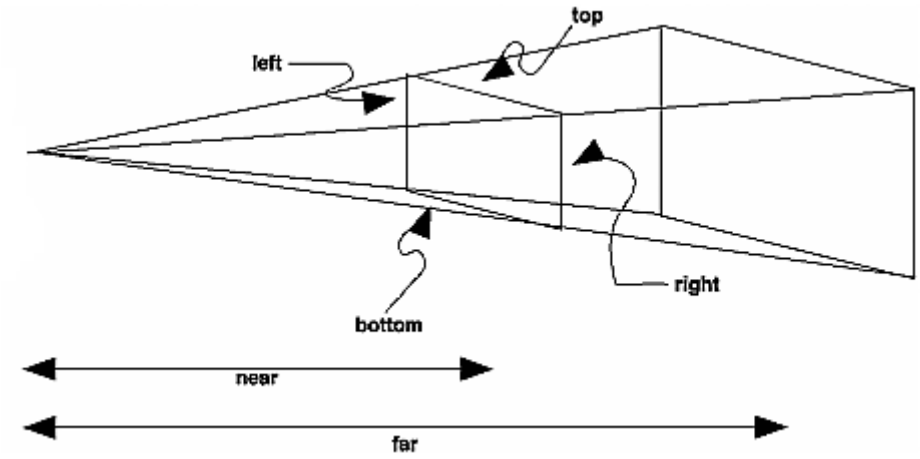

Figura 22: Esquema de definición de frustum

Por tanto, tenemos que calcular las coordenadas de los planos. Al ser los cálculos muy similares para los desplazamientos horizontal y vertical, se explicarán solo para el caso de un desplazamiento horizontal, y después se presentaran las formulas finales para el desplazamiento en horizontal y vertical.

5.2.3. Ecuaciones para calcular la perspectiva

Empezaremos por calcular el ángulo de visión que tendrá la cámara. Este ángulo, debe hacer coincidir la apertura del campo de visión con los márgenes de la pantalla en la que se verá la imagen final como se puede observar en la figura 23. Es necesario ajustar la perspectiva tanto en horizontal como en vertical, por lo que, y suponiendo que conocemos las dimensiones de la pantalla, calcularemos la relación de tamaños entre estos dos ángulos utilizando la ecuación 3.

$$aspectRatio = \frac{alturaPantalla}{anchuraPantalla} \tag{3}$$

De esta manera, podemos pasar del ángulo horizontal al vertical y viceversa con la relación descrita en las ecuaciones 4 y 5.

$$ángulo Vertical = ángulo Horizontal * aspect Ratio \tag{4}$$

$$ángulo Horizontal = \frac{ángulo Vertical}{aspect Ratio} \tag{5}$$

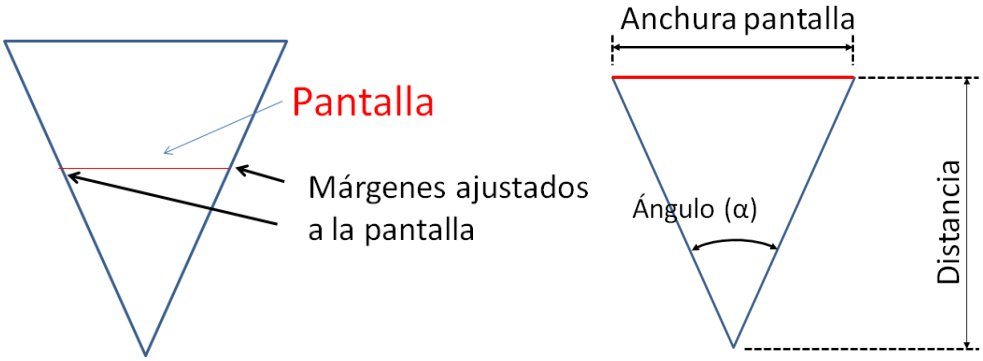

Figura 23: Esquema del frustum ajustado en los márgenes de la pantalla (izquierda). Representación de datos para el cálculo del ángulo de visión (derecha)

Para calcular el ángulo que utilizaremos, empezaremos suponiendo que el espectador se encuentra en frente de la pantalla. Siguiendo el esquema de datos expuesto en la figura 23, podemos definir el ángulo, ya que:

$$\tan(\frac{\alpha}{2}) = \frac{(\frac{AnchuraPantalla}{2})}{Distancia} \tag{6}$$

y si desarrollamos esta ecuación, podemos despejar αde la siguiente manera:

$$\alpha = 2\ \arctan\left(\frac{(\frac{AnchuraPantalla}{2})}{Distancia}\right) \tag{7}$$

En este punto, podemos calcular las coordenadas de anchura de este caso inicial.

$$coorAnchura = \frac{near}{\tan\left(\frac{\alpha}{2}\right)} \tag{8}$$

Hasta este punto, estamos suponiendo que el espectador esta delante de la pantalla, pero en realidad, queremos que el espectador pueda moverse de esta posición. Este ajuste, lo hacemos con un índice de posición que se define en la ecuación 9.

$$índiceMov = \frac{near}{distancia} \tag{9}$$

De esta manera, podremos calcular la perspectiva según la distancia en la que se haya desplazado respecto a esta situación inicial.

Teniendo estos datos, podemos calcular las coordenadas del frustum asimétrico con las ecuaciones 10, 11, 12 y 13:

$$left = -coorAnchura + (desplazamientoHorizontal * índiceMov) \tag{10}$$

$$right = coorAnchura + (desplazamientoHorizontal * índiceMov) \tag{11}$$

$$top = ratio * coorAnchura \tag{12}$$

$$bottom = -ratio * coorAnchura \tag{13}$$

Al aplicar estos valores al frustum, desplazamos el plano de proyección de este, en este caso, hacia la derecha. Hemos visto anteriormente en la figura 21, como moviendo el espectador, el campo de visión de este cambia siguiéndole. No obstante, en este caso, es la perspectiva la que cambiamos, tal y como se puede ver en la figura 24, por lo que es necesario mover el frustum para que siga al espectador.

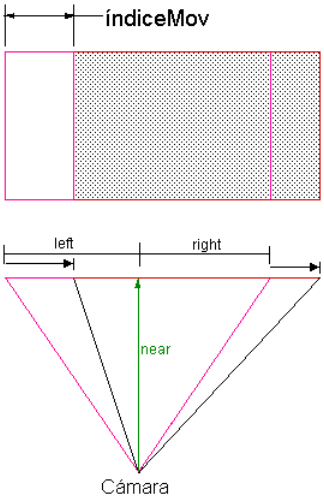

Figura 24: Diagrama que explica como ajustamos la perspectiva teniendo en cuenta el desplazamiento

De esta manera, hemos ajustado la perspectiva del espectador a su posición teniendo en cuenta solo el desplazamiento horizontal de este. Para tener en cuenta también el vertical, solo es necesario cambiar los valores de *top* y *bottom* utilizando el índice de desplazamiento utilizado anteriormente. Las ecuaciones finales para este fin serian las ecuaciones 14 y 15 para los nuevos *top* y *bottom*, mientras que para left y right seguiremos utilizando las ecuaciones 10 y 11.

$$top = ratio * coorAnchura + desplazamientoVertical * índiceMov \tag{14}$$

$$bottom = -ratio * coorAnchura + desplazamientoVertical * índiceMov \tag{15}$$

5.3. Implementación del 3D estereoscópico

Como se ha comentado anteriormente, la representación estereoscópica requiere proyectar o ver con cada ojo una imagen diferente. Pero, ¿como controlar la profundidad? ¿Cual es la parte que esta por fuera de la pantalla y cual por detrás? Este apartado intenta explicar brevemente los conceptos más generales de esa técnica y detallar como se a abordado esta problemática en el desarrollo del proyecto.

5.3.1. Conceptos generales

Como ya se ha nombrado anteriormente, el 3D estereoscópico es un método basado en proyectar una imagen diferente a cada ojo, para simular un entorno real, en el que cada ojo obtiene una perspectiva diferente de la misma escena. Una vez que cada ojo tiene su imagen, siendo estas *realistas*, el cerebro interpreta la profundidad según las coincidencias entre ambas, como lo haría en un entorno real.

Cuando nos referimos a realista, es porque estas imágenes deben tener ciertas 'propiedades' para que el cerebro pueda interpretar la escena de forma correcta.

Estas propiedades se pueden dividir en dos grupos principales: propiedades de la escena y propiedades del estéreo.

Las propiedades de la escena se aplican aun cuando no utilizamos la representación estereoscópica, ya que esta no es necesaria para tener una mínima sensación de profundidad. Por ejemplo, en animaciones en las que tenemos movimiento de la cámara, podemos percibir la profundidad de los objetos.

Propiedades de la escena:

- *Perspectiva*: Los objetos más alejados son más pequeños que los que están cerca. Además, las líneas paralelas convergen en el horizonte.
- *Tamaño de objetos conocidos*: Inconscientemente, esperamos que ciertos objetos sean más grandes o pequeños que otros. Por ejemplo, si ponemos un elefante (la representación del animal, no la de una figura pequeña) al lado de una taza de té, teniendo ambas el mismo tamaño, esperamos que el elefante esté más lejos.
- *Detalle*: Los objetos más cercanos tienen más detalle que los lejanos.

- *Oclusión*: Si un objeto ocluye a otro, el ocluido debe de estar más alejado que el otro.
- *Luces y sombras*: La forma en la que la luz se refleja en los objetos debe ser realista. Las sombras pueden ser vistas como objetos de oclusión, por lo que debe tenerse en cuenta si la sombra ocluye algo o es ocluida por algo.
- *Movimiento relativo*: Al mover la cámara, los objetos más lejanos se mueven más despacio que los que están más cerca.

Propiedades del estéreo:

- *Disparidad binocular*: Esta propiedad hace referencia a la diferencia entre las perspectivas de cada ojo. Los ojos están separados por cierta distancia horizontalmente (normalmente se toma la medida de 6cm como media global). Es necesario tener esto en cuenta a la hora de saber cuanto mover la cámara para sacar cada imagen, ya que si la distancia es muy pequeña, no habrá sensación de profundidad, pero si es muy grande, los ojos no podrán interpretar la profundidad.
- *Acomodación*: acomodación se refiere a el esfuerzo que necesita el músculo ocular para adecuar la lente del ojo a la profundidad representada.
- *Convergencia*: se refiere a la tensión necesaria para girar el ojo y alinear las dos imágenes para poder ser representadas.

Respecto al estéreo, la disparidad binocular se considera la más importante que las demás, pero si las otras dos están mal representadas, la visualización empeora considerablemente.

5.3.2. Representación estereoscópica

El ser humano interpreta la distancia de un objeto según cuanto tengan que girar el globo ocular en cada ojo para centrar la vista en el objeto. Teniendo en cuenta esto, si controlamos el giro de los ojos, controlamos a que profundidad lo estamos viendo.

La estereoscopía se basa en enseñar a cada ojo una imagen diferente, cada una con una perspectiva diferente del mismo objeto o escena (la que corresponde al ojo). Al

ser representaciones de la misma escena, habrá puntos que más o menos coincidan en ambas representaciones, como el borde de un cubo por ejemplo. Podemos separar estas dos imágenes para controlar cuanto giramos el globo ocular para mirar a un mismo punto en ambas imágenes.

De esta manera, si el objeto que intentamos representar debería de estar detrás de la pantalla, la linea de los ojos se juntaría detrás de la pantalla en la que estamos proyectando la imagen, por lo que cortarán el plano de proyección (la pantalla) estando la línea del ojo izquierdo a la izquierda y la del derecho a la derecha (ver figura 25).

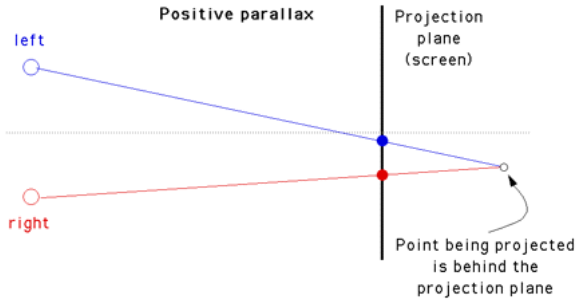

Figura 25: Diagrama de representación de imagen 3D detrás del plano de la pantalla

Nótese que realmente, a donde estamos mirando es al plano de proyección (al punto de corte), siendo el punto blanco un punto que nuestro cerebro interpreta.

Podemos mover la posición de estas imágenes a izquierda y derecha, modificando la distancia entre los puntos de corte. Si separamos más los puntos, el objeto estará cada vez más alejado, mientras que si los juntamos, el objeto irá acercándose.

Si acercamos estos puntos, acabarían juntándose, haciendo que el punto interpretado estuviese a la par de la pantalla, y si seguimos moviendo las imágenes en la misma dirección, las lineas de los ojos se cruzarán delante de la pantalla. Esto crea la ilusión de que el objeto sale de la pantalla (ver figura 26).

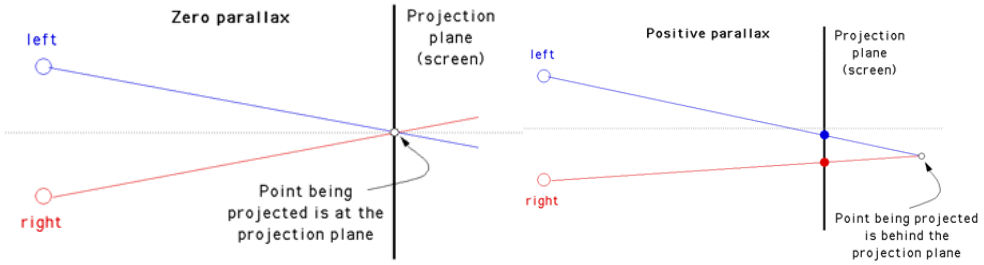

Figura 26: Diagrama de representación de una imagen 3D justo en el plano de la pantalla (izquierda). Diagrama de representación de una imagen 3D delante del plano de la pantalla (derecha).

La libertad de movimientos de nuestro globo ocular es limitada. Por ello, llegará un punto en el que la imagen representada esté demasiado cerca como para que nuestros músculos puedan cruzar la vista lo suficiente como para poder enfocarlo de forma cómoda. En este punto es cuando las propiedades de convergencia empezaran a fallar, y la sensación se vuelve incomoda y muchas veces molesta.

5.3.3. Conseguir las imágenes

Para representar una imagen, primero es necesario capturarla. En esta ocasión tenemos dos en vez de una, y ambas deben cumplir unas propiedades para poder representarse bien.

Teniendo en cuenta que las propiedades generales se cumplen, lo siguiente sería conseguir dos imágenes que representen la escena o el objeto en 3D de forma correntie.

Lo primero sería la disparidad. Para mantener la disparidad, sería necesario separar una toma de imagen de la otra 6 cm (medida tomada como media).

No obstante, no podemos tomar la imagen apuntando directamente a un punto concreto, ya que el resultado al intentar representar las imágenes no seria correcto (ver figura 27).

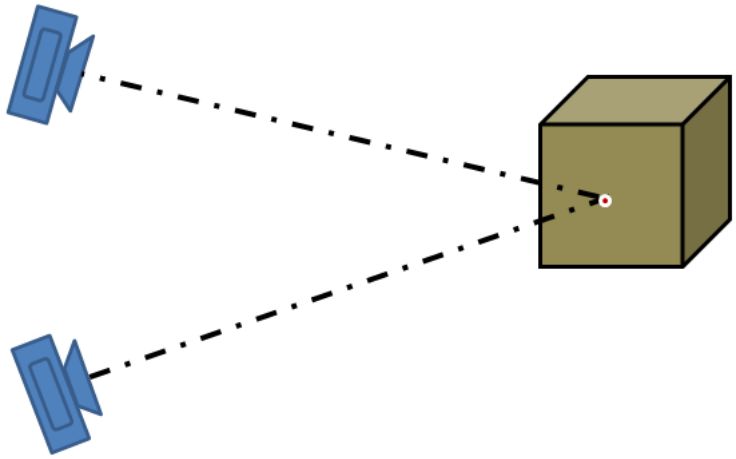

Figura 27: Esquema de toma de imagen girando las cámaras

La cuestión esta, en que el plano formado por una de las cámaras no concuerda con el sacado con la otra, porque al apuntar al mismo punto hemos rotado la cámara, y con eso la representación de la escena. Al intentar enviar la imagen a cada ojo, el punto concreto al que apuntábamos encajaría, pero el resto de la imagen no, haciéndola incoherente para nuestro cerebro. Esta idea se representa gráficamente en la figura 28 en la que vemos como captan las cámaras una escena, y cuales son los planos que capturarían.

Si nos fijamos en la figura 28, podemos ver que si miramos a la parte izquierda de la pantalla, un mismo punto tiene dos profundidades diferentes según en cual de las dos imágenes capturadas está..

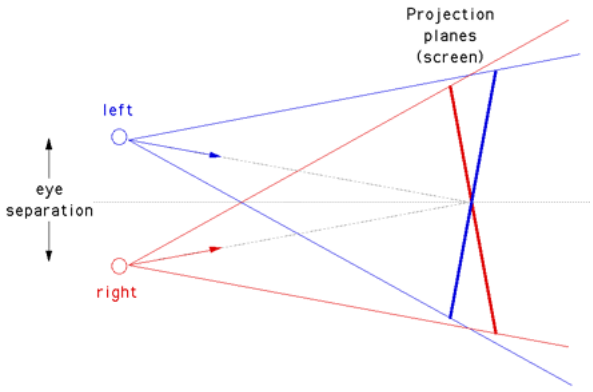

Figura 28: Representación incorrecta de escena con 3D estereoscópico

La forma correcta (o al menos la más fiel a la realidad) sería utilizar perspectivas asimétricas para capturar la escena, y obtener las imágenes en perpendicular a la escena.

Para esto, cada una de las cámaras, separadas entre sí 6 cm, toma una imagen paralela de la escena. En la figura 29 vemos un esquema de como sería la captura. En este esquema podemos percibir como en un lado de cada cámara tenemos un trozo de imagen que no se refleja en la otra.

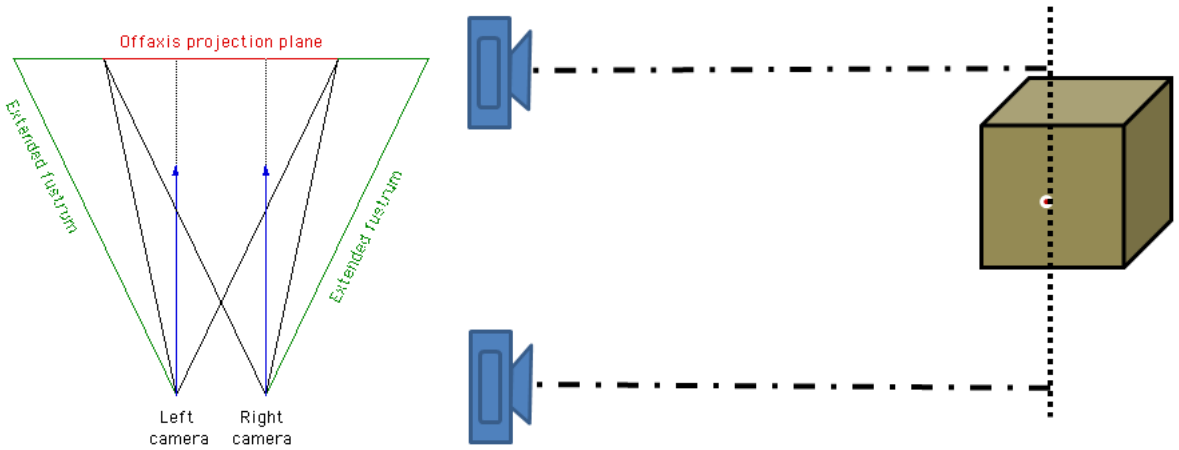

Figura 29: Esquema del resultado de obtener dos imágenes paralelas de una misma escena con un desplazamiento horizontal entre las imágenes (izquierda). Captura de dos imágenes de la misma escena de forma paralela (derecha)

Esto es lo mismo que pasa cuando utilizamos nuestros ojos para mirar. La diferencia reside en que nuestro cerebro está preparado para esto, por lo que post-procesa las imágenes capturadas por los ojos para difuminar o ignorar las partes no coincidentes. En nuestro caso las podemos quitar de la representación, ya que lo que se reflejará en la pantalla será solo la parte coincidente.

De esta manera, quitamos las partes que no nos interesan, y podemos representar esto mismo en la pantalla para tener una vista en tres dimensiones (siempre si utilizamos una de las tecnologías antes mencionadas para este fin). En la figura 30 podemos ver un esquema para los últimos dos pasos que hemos definido[12], así como la representación de como quedarían las imágenes para nuestros ojos en una implementación final con estas imágenes.

[12]Para la implementación se ha utilizado OpenGL, que nos provee las funciones necesarias para poder capturar las imágenes sin hacer la parte de cortar el sobrante. En vez de eso, podemos definir la cámara de cada ojo con un frustum asimétrico como el expuesto en el apartado 4.2, con lo que conseguimos la imagen directamente sin sobrante.

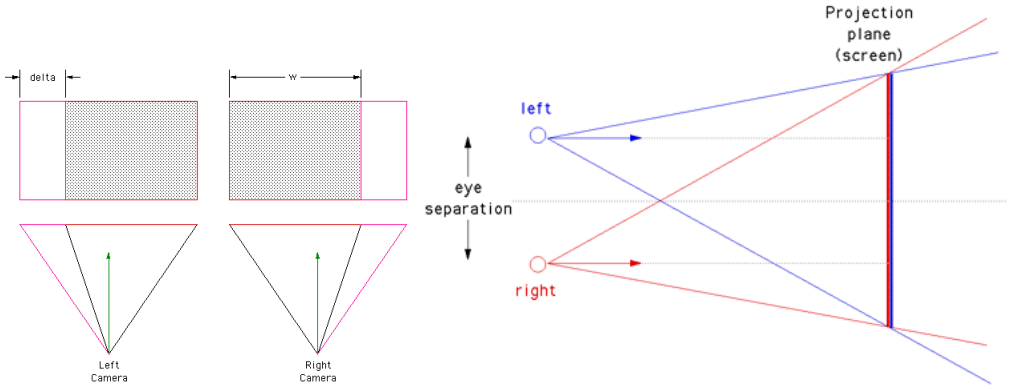

Figura 30: Recorte de imagen para obtener parte coincidente (izquierda). Representación final de las imágenes en la pantalla (derecha)

Podemos definir *delta* como el sobrante de la imagen. Es posible calcular este valor si tenemos en cuenta que *w* es el tamaño final de la imagen, *f* es la distancia entre la cámara y el plano de proyección, *e* es la distancia entre las cámaras, y conocemos el ángulo de apertura de la cámara como *a* [13]. Conocidos estos datos, la ecuación 16 define la forma de calcular *delta*.

$$delta = \frac{w\ e}{2\ f\ \tan(\frac{a}{2})} \tag{16}$$

La última cuestión que queda por tratar, es la distancia entre el espectador y la pantalla. En este aspecto, debemos tener en cuenta que las imágenes han sido tomadas desde una distancia determinada, y que el espectador estará viendo la escena de otra distancia diferente.

Por otra parte, podemos querer hacer una foto a un objeto pequeño y ampliarlo (un insecto por ejemplo) o a algo grande y reducirlo (una montaña). No podemos poner al espectador a dos centímetros de la pantalla, ni a tres kilómetros.

Para entender como se hace esta adaptación de tamaño, debemos tener en cuenta que tendremos una parte virtual (la parte en la que sacamos la foto), y otra parte real (el espectador mirando la pantalla). Lo que nos interesa ahora es, relacionar la parte virtual con la parte real, y para eso adaptaremos las diferentes distancias que aparecen definidas en la figura 31.

[13] Las cámaras virtuales, así como las reales, tiene un ángulo de apertura que se define en su matriz de perspectiva. Por ello, tanto si se trata de una fotografía convencional, como una imagen renderizada, este dato puede ser conocido o definido explícitamente.

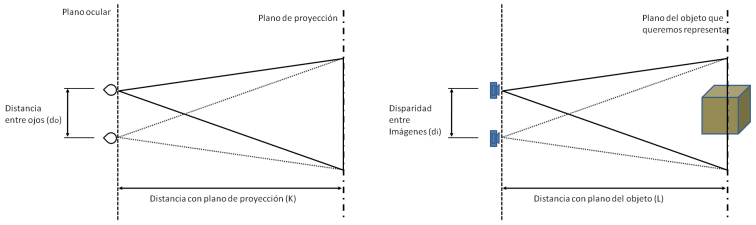

Figura 31: Representación del mundo real (izquierda) y representación del mundo virtual (derecha). En ambas imágenes se representan las distancias que tendremos que relacionar entre el mundo virtual y el mundo real.

Como podemos ver en las dos imágenes, es necesario mantener la relación de distancias en los dos esquemas. Si esto lo ponemos en como representación matemática, podemos decir que la relación de distancias del mundo real es:

$$\frac{d_o}{K} \tag{17}$$

y la correspondiente al mundo virtual es:

$$\frac{d_i}{L} \tag{18}$$

Para mantener la relación, entonces igualamos la ecuaciones 17 y 18:

$$\frac{d_o}{K} = \frac{d_i}{L} \tag{19}$$

Siguiendo esta sencilla ecuación, podemos saber cual es la disparidad necesaria entre las imágenes tomadas para poder representarla de forma adecuada.

$$d_i = \frac{d_o L}{K} \tag{20}$$

5.4. Relación entre el mundo real y el mundo virtual

En ocasiones en el que utilicemos sistemas de captura de movimientos como método de interacción entre el usuario y la aplicación, deberá tenerse en cuenta que entre los dos sistemas puede haber diferencias que hagan imposible que esta comunicación se realice de forma correcta.

El sistema que captura la posición del usuario está presente en el mundo real, pero los datos recogidos por este dispositivo tienen repercusión en el mundo virtual. Esto crea la necesidad de tener que adaptar los datos obtenidos en un sistema de referencia para ajustarlos a un segundo, y que las translaciones de un sistema tengan el mismo efecto (dirección y sentido de la translación) tanto en el mundo real como en el virtual.

Esta barrera se cruza utilizando una matriz de transformación de sistema. Con esto, adaptamos el sistema de referencia A al sistema de referencia B.

Supongamos que el sistema de captura (en adelante *tracker*) genera una matriz de transformación para tres dimensiones (una matriz de 4x4). Esta matriz está definida por el sistema de referencia del tracker (ver figura 32).

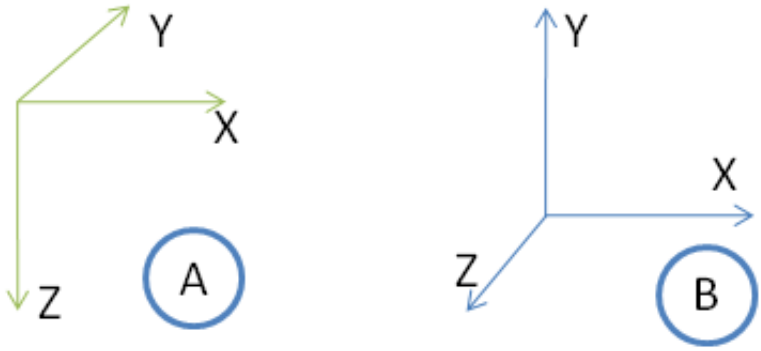

Figura 32: Representación del sistema de referencia del tracker (izquierda) y del mundo virtual (derecha)

Comparando el los dos sistemas, tracker y mundo virtual, podemos obtener los vectores equivalentes a un eje en un sistema en el otro. Es decir, cuales son las coordenadas que representan el eje X del sistema A, por ejemplo, en el sistema B. Con esta comparación podemos deducir:

$$u_x = (1, 0, 0) \tag{21}$$

$$u_y = (0, 0, -1) \tag{22}$$

$$u_z = (0, -1, 0) \tag{23}$$

Donde u_x, u_y y u_z son los vectores de transformación de A a B. Estos se pueden escribir en forma de matriz como se puede ver en la ecuación 24.

$$T_{sis} = \begin{bmatrix} u_{xx} & u_{xy} & u_{xz} & 0 \\ u_{yx} & u_{yy} & u_{yz} & 0 \\ u_{zx} & u_{zy} & u_{zz} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

Que aplicando los vectores anteriores, quedaría definido como:

$$T_{sis} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{25}$$

Podemos comprobar el resultado de aplicar esta transformación pasando la definición de un vector cualquiera definido en el sistema A y multiplicándolo por la matriz de transformación. Este vector debería estar en la misma posición relativa en ambos sistemas de referencia.

Para comprobarlo, definiremos el vector $v_0 = (2, 1, 0)$, y lo multiplicaremos por la matriz de transformación.

$$v_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 2 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -1 \\ 1 \end{bmatrix} \tag{26}$$

En la figura 33 podemos comprobar que tanto el vector v_0 como el v_1 son equivalentes, pero en diferentes sistemas de referencia.

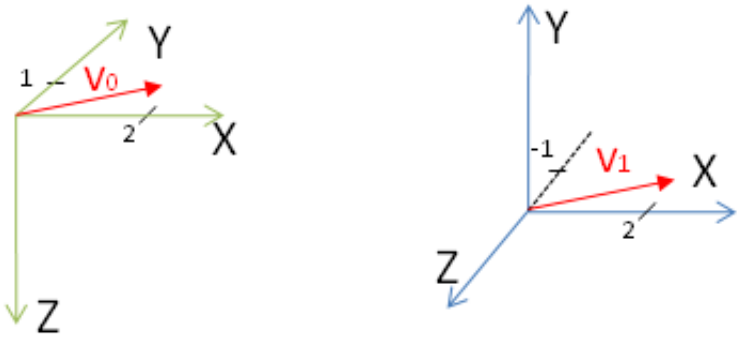

Figura 33: El vector $v_0$representado en el sistema de referencia A (izquierda) y el vector v_1 representado en el sistema de referencia B (derecha). En la figura se puede ver que los dos vectores son equivalentes.

Multiplicando la matriz de transformación de sistema (25) con la matriz obtenida del tracker, obtenemos la matriz de transformación del mundo virtual. Esta transformación será necesaria cada vez que tengamos que pasar de un sistema de coordenadas a otro, y como en este caso, solo tenemos dos sistemas, solo tendremos que hacer la conversión una vez.