Máster en

Ingeniería Computacional y Sistemas Inteligentes

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

Máster I.C.S.I.

Proyecto de Fin de Máster

Captura de movimiento multicámara mediante hardware de bajo coste

Haritz Garcia Oliden

Director

Dr. Alejandro García-Alonso Montoya

Departamento de Ciencias de la Computación e Inteligencia Artificial
Facultad de Informática

informatika fakultatea

facultad de informática

KZAA / CCIA

Septiembre 2011

Proyecto desarrollado en colaboración con

STT Ingeniería y Sistemas

Agradecimientos

A Jesús Calleja de STT, por sus explicaciones de las bases y fundamentos matemáticos del proceso de animación esquelética, gracias a las cuales he avanzado más rápido en esa fase del trabajo.

A José Manuel Jiménez de STT, por darme la oportunidad de hacer uso de sus equipos e instalaciones durante la realización del proyecto.

Al Dr. Alejandro García-Alonso de la UPV-EHU, por su orientación en el enfoque y objetivos del proyecto.

RESUMEN

Con el objetivo de investigar posibles métodos de captura de movimiento de bajo coste, en este proyecto se ha optado por analizar las capacidades de la cámara Microsoft Kinect desarrollando un proceso de captura mediante el uso de dos de estas cámaras. Al principio se introducen los conceptos básicos generales sobre técnicas de captura de movimiento, repasando brevemente las técnicas principales que se utilizan hoy en día. También se explican los detalles sobre el funcionamiento de la cámara Microsoft Kinect que se ha utilizado en esta ocasión, y se explican las razones por las que nos hemos decantado por esta tecnología. Después se expone detalladamente el proceso de captura de movimiento que se ha implementado en este proyecto. Se explica cada paso del proceso completo de captura y animación, comentando los problemas que presenta la tecnología utilizada y las soluciones que se ha decidido aplicar para intentar solventarlos. Al final, se recopilan las conclusiones que se han ido sacando a lo largo del proyecto sobre las limitaciones de la cámara y su posible utilidad práctica.

ABSTRACT

In order to research in new low cost motion capture methods, this project has chosen to analyze the capabilities of the Microsoft Kinect camera developing a simple capture process that uses two of these cameras. At first we introduce the general concepts of motion capture techniques, doing a brief review of the main techniques used today. We also explain the details about the Microsoft Kinect camera that we used, and expose the reasons that made us to try this technology. Afterwards we detail the motion capture process that has been implemented in this project. We explain every step of the complete capture and animation processes, discussing the problems we had with the used technology and talking about the solutions we decided to try to solve them. Finally, we collect the conclusions about the limitations of the camera and its potential practical use.

ÍNDICE

ÍNDICE DE LAS IMÁGENES

1 INTRODUCCIÓN

La animación 3D es, desde hace años, cada vez más utilizada en multitud de sectores. Desde los videojuegos, el cine, la televisión y la publicidad, cada año que pasa el uso de animaciones 3D de todo tipo se multiplica a lo largo del mundo. Y cuando se trata de animar personajes humanoides*, la captura de movimiento es sin duda la técnica más utilizada, gracias al gran ahorro de tiempo y el realismo de la animación que ofrece comparado con la clásica animación manual.

Se les llama captura de movimiento o mocap* a aquellas técnicas que consisten en registrar a un ser humano ejecutando una acción y utilizar esa información para animar de forma automática un modelo virtual humanoide. A pesar de sus evidentes ventajas, estas técnicas tienen un problema en el alto coste que tienen los sistemas profesionales, que hacen uso de dispositivos especiales que generalmente suelen ser muy caros. Por este motivo, es interesante analizar lo que nos ofrece la tecnología actual para conseguir un mocap de bajo coste. Concretamente, existe en la actualidad una cámara de captura de movimiento en el mercado con un precio increíblemente bajo: la Kinect de Microsoft.

A finales de 2005 Microsoft lanzó su segunda consola de juegos doméstica, Xbox 360. A pesar de que obtuvo unas buenas ventas iniciales, pronto se vieron eclipsados por el arrollador éxito de Nintendo y su Wii, que ofrecía un sistema de control por movimientos basado en un mando giroscópico. Para competir con Nintendo, los investigadores de Microsoft dedicaron cuatro años al llamado Project Natal, cuyo objetivo era elaborar un sistema de control por movimiento que fuera capaz de detectar todo el cuerpo del usuario, sin necesidad de utilizar ningún tipo de mando. El resultado del proyecto fue la cámara Kinect, un pequeño dispositivo de captura de movimiento óptico sin marcadores. Como es habitual, el hecho de que el hardware se fabrique de forma masiva abarató mucho su precio en el mercado (se dice que el primer prototipo costó unos 30.000$), y gracias a eso Kinect hace que hoy en día sea posible adquirir una cámara de captura de movimiento por 150$, algo impensable hace pocos años.

* A lo largo del presente documento, los términos que pueden resultarle desconocidos al lector se marcan de esta forma. Para leer una breve descripción, ver el *Capítulo 2, Definiciones y abreviaturas*.

1.1 Objetivos

Con el objetivo de desarrollar un sistema de captura de movimiento de bajo coste, en este proyecto se ha indagado en la utilidad de la cámara Microsoft Kinect para dicho fin. Se ha analizado cuáles son las restricciones de la cámara, para ver hasta donde se puede llegar con ella y determinar su utilidad potencial en la práctica.

Se ha desarrollado un proceso básico de captura de movimiento multicámara* mediante dos cámaras Kinect, con el fin de tener un sistema que sirva de modelo de análisis continuo. De esta forma, a lo largo de la implementación de este proceso se han podido ver los problemas que derivan del uso de esta clase de tecnología y se han buscado soluciones para dichos problemas.

1.2 ¿Por qué Kinect?

Básicamente, dos son los motivos principales que nos han llevado a decantarnos por esta tecnología.

Por un lado, gracias al sensor de profundidad que incorpora Kinect no hace falta más de una cámara para triangular un punto en el espacio. En comparación, en las técnicas de mocap tradicionales se utilizan un mínimo de cuatro cámaras (habitualmente entre 6 y 24) colocadas milimétricamente alrededor del actor, y se utilizan esas múltiples capturas para triangular los puntos. Por lo tanto, Kinect nos ofrece la posibilidad de utilizar menos cámaras abaratando el gasto en hardware y requiriendo un entorno menos restrictivo para su uso. En este proyecto se han utilizado dos Kinect, ya que con una única cámara se pueden dar oclusiones*.

Por otro lado, al ser un hardware relativamente reciente (fue lanzado en noviembre del 2010), a día de hoy no se ha trabajado tanto con él como con otros sistemas más antiguos, y eso lo hace muy atractivo como objeto de investigación.

1.3 Fases

Se ha optado por dividir todo el proceso de captura de movimiento -desde que el actor inicia su movimiento hasta que se produce una animación de su avatar* en el PC- en dos fases: La fase de captura y la fase de animación.

La fase de captura engloba la parte del proceso que se hace *online*, es decir, mientras el actor está siendo registrado por las cámaras. En esta fase se obtienen una serie de puntos 3D que corresponden a varias articulaciones del actor.

La fase de animación, por contra, recoge la parte del proceso que se hace *offline*, esto es, una vez se ha terminado con la captura. A partir de los datos que se han obtenido en la fase anterior, se aplican una serie de procesos matemáticos para obtener la animación del avatar.

2 DEFINICIONES Y ABREVIATURAS

6 grados de libertad: La máxima libertad de movimiento de un sólido, comprendiendo tanto el desplazamiento como la rotación en los ejes X, Y, Z.

Avatar: En este contexto, personaje virtual humanoide que se quiere animar para que sus movimientos reflejen aquellos que realizó el actor.

Cámara de Tiempo de Vuelo: Cámara de profundidad que emite una ráfaga de luz y calcula el tiempo que tarda en dar en un objeto y volver para determinar la distancia, de forma similar a lo que hace un sonar con el sonido.

Closing: Una de las operaciones morfológicas típicas para suprimir ruido. Elimina los agujeros de un objeto dilatándolo primero y erosionándolo después.

Euler, Leonhard: Matemático y físico suizo, uno de los más grandes de todos los tiempos y el mayor del siglo XVIII.

Exoesqueleto: Armadura mecánica que normalmente sirve para potenciar las capacidades físicas del ser humano (para levantar grandes pesos, etc.).

Filtro de color Bayer: También llamado GRGB, es la máscara de filtrado de color que se utiliza en la mayoría de sensores actuales.

Filtro pasa bajo: Filtro caracterizado por permitir el paso de las frecuencias más bajas y atenuar las frecuencias más altas.

Frame: Elemento individual que forma parte de una cadena que forma una animación. Normalmente, esos elementos son imágenes estáticas que al ser proyectadas en una rápida sucesión crean la ilusión de movimiento (el cine utiliza 24 frames por segundo, mientras que los videojuegos utilizan entre 30 y 60). Sin embargo, en nuestro contexto también se puede referir a una serie de datos (puntos, ángulos, etc.) que determinan la posición de un esqueleto en un momento dado.

Frecuencia de corte: En este contexto, el valor que limita las frecuencias que pasan por el filtro.

Humanoide: Figura cuyo aspecto corporal se asemeja a la de un humano. Debe ser bípedo, y tener una estructura ósea similar a la de los humanos/primates.

LED: Del inglés *Light-Emitting Diode*, es un diodo semiconductor que emite luz.

Material retroreflectivo: Material con gran capacidad para reflejar la luz.

Mocap: Del inglés *Motion Capture*, es el término que se suele emplear habitualmente para referirse a la captura de movimiento.

Monocromático: En este contexto, escala de grises.

Multicámara: Sistema que hace uso simultáneo de dos o más cámaras.

Oclusión: En este contexto, se da una oclusión cuando una parte del cuerpo tapa a otra, quedando la segunda fuera de la visión de la cámara.

Posición neutral del esqueleto: Posición inicial del esqueleto, antes de ser animado. En esta posición los sistemas de referencia de todos los huesos tienen la misma orientación que el sistema de referencia global, y el hueso raíz (la pelvis) se halla en el punto (0, 0, 0).

Ruido: Defectos e imperfecciones en la imagen que se deben eliminar para que no interfieran en su procesamiento posterior.

Segmentación: Es el proceso de dividir una imagen digital en varias partes o segmentos, con el objetivo de analizar uno o varios de ellos de forma independiente.

Thresholding: Técnica de procesamiento de imagen en el que se pasa una imagen a color binario definiendo un umbral (en inglés, *threshold*) y asignando blanco o negro a cada píxel en función de si su valor de color original está por encima o debajo de ese umbral.

Tracking: El motion tracking (o tracking a secas) es el proceso de seguir un punto u objeto a lo largo de la serie de imágenes que componen un video con el objetivo de determinar su movimiento.

Visual Studio 2010 Express: Son versiones gratuitas del IDE Visual Studio 2010, orientados a un único lenguaje de programación y sin algunas características avanzadas presentes en las ediciones comerciales.

3 CONCEPTOS BÁSICOS

En este capítulo se introducen los conceptos básicos generales sobre técnicas de captura de movimiento, mencionando brevemente las técnicas principales que se utilizan hoy en día. También se explican los detalles sobre la tecnología y funcionamiento de la cámara Microsoft Kinect que se ha utilizado en esta ocasión.

3.1 Técnicas de captura de movimiento

Las técnicas de captura de movimiento se dividen en dos grupos principales: Sistemas ópticos y sistemas no ópticos.

En el primer grupo se utilizan cámaras para captar imágenes de los actores y se hace uso de esa información para triangular la posición del cuerpo. Existen tres tipos principales de sistemas de captura de movimiento ópticos: Los que utilizan marcadores pasivos, los que utilizan marcadores activos y los que no utilizan marcadores. Los marcadores pasivos son piezas de material retroreflectivo* que se colocan en partes estratégicas del cuerpo del actor (pies, rodillas, cadera, manos, codos, cabeza, etc.), mientras este viste una malla de color oscuro (normalmente negro). Al iluminar los marcadores, las cámaras captan una serie de puntos blancos sobre un fondo negro, dando una imagen fácil de tratar mediante técnicas de procesamiento de imagen (ver imagen 1). Un sistema de este tipo suele contar entre 6 y 24 cámaras, para poder triangular correctamente la posición 3D de cada marcador. Hoy en día es el sistema más utilizado en capturas de cuerpo completo para animación. Los marcadores activos, a diferencia de los pasivos, incluyen su propia fuente de luz (generalmente un LED*). Son utilizados, principalmente, en captura de movimiento facial [1], ya que es un sistema mucho más cómodo de utilizar (ver imagen 2). Los llamados marcadores modulados por tiempo o los marcadores imperceptibles semi-pasivos también entrarían en esta categoría. Por último, los sistemas sin marcadores son una aproximación emergente que se está viendo respaldada por los últimos avances en la investigación en visión por computador. En los últimos años muchos estudios interesantes han perseguido el objetivo de conseguir una captura de movimiento en condiciones cada vez menos restrictivas [2][3][4][5]. Microsoft Kinect ofrece un sistema de captura de movimiento sin marcadores.

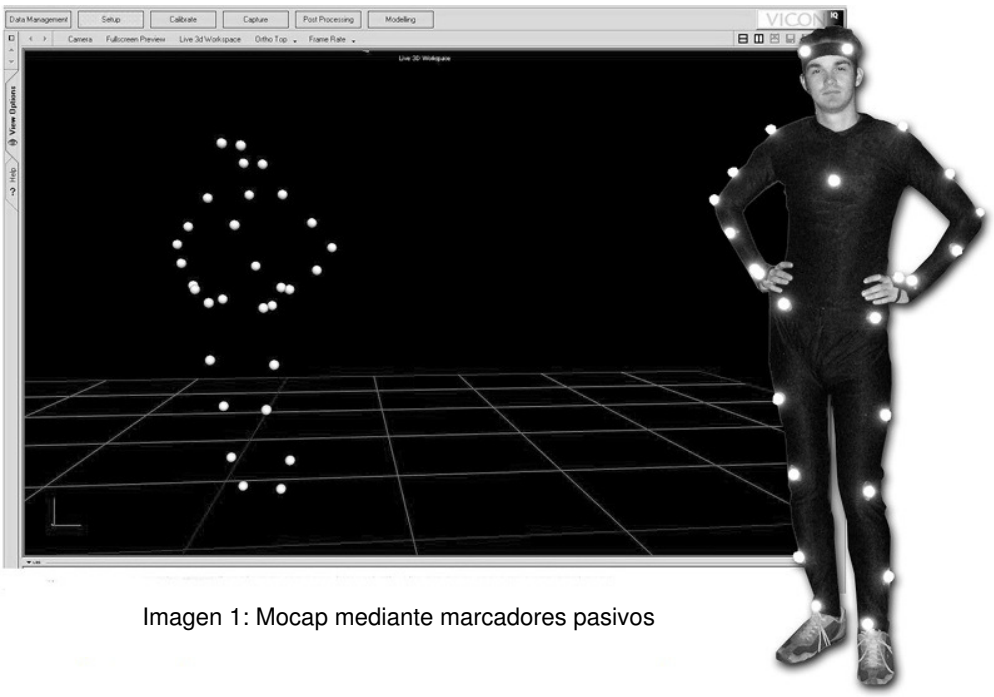

Imagen 1: Mocap mediante marcadores pasivos

En cuanto a los sistemas no ópticos, se dividen en tres grupos: Sistemas inerciales, sistemas de movimientos mecánicos y sistemas magnéticos. Los sistemas inerciales se basan en el uso de sensores de inercia para calcular los cambios en los seis grados de libertad* de un objeto. El exitoso mando de Wii utiliza este tipo de tecnología, pero con un nivel de precisión muy inferior a los sistemas profesionales. Un traje de captura de movimiento con este tipo de sensores oscila entre los 25000$ y los 80000$. Los sistemas de movimientos mecánicos se basan en algún tipo de exoesqueleto* que el actor tiene que llevar. Este exoesqueleto tiene sensores en cada articulación que le dicen al sistema su posición exacta. Sin embargo, su alto coste y limitada movilidad lo convierten en una alternativa poco viable. Los sistemas magnéticos utilizan tres bobinas para crear tres campos magnéticos ortogonales y mapean meticulosamente el volumen del objeto que está en el centro. Es una tecnología que ha propulsado varias investigaciones interesantes en los últimos años [6].

Imagen 2: Captura facial mediante marcadores activos

3.2 La tecnología de la cámara Microsoft Kinect

La cámara Microsoft Kinect utiliza una tecnología desarrollada por la empresa israelí *PrimeSense*, el sistema de escaneo *Light Coding*, que es capaz de interceptar la información 3D de la escena mediante la proyección continua de una luz infrarroja.

La 'cámara', por lo tanto, se divide en dos partes: La propia cámara RGB y el sensor de profundidad (en realidad hay una tercera, un micrófono para funciones de reconocimiento del habla, pero no es algo que nos interese en estos momentos). El sensor de profundidad consiste en un proyector de láser infrarrojo combinado con un sensor monocromático* CMOS, que captura la información 3D del vídeo independientemente de las condiciones lumínicas (ver imagen 3). El rango de sensibilidad del sensor de profundidad es ajustable entre 0,7 y 5 metros, y utiliza una escala de grises de 11 bits -2048 niveles de intensidad- desde el blanco (cerca) al negro (lejos). El rango de visibilidad ronda los 57º horizontales y 43º verticales, aunque la cámara está montada sobre un pivote motorizado que es capaz de rotar 27º tanto arriba como abajo. La cámara RGB envía un stream de vídeo de una resolución VGA (640x480 píxeles) de 8 bits con un filtro de color Bayer*, a una frecuencia de 30 frames* por segundo.

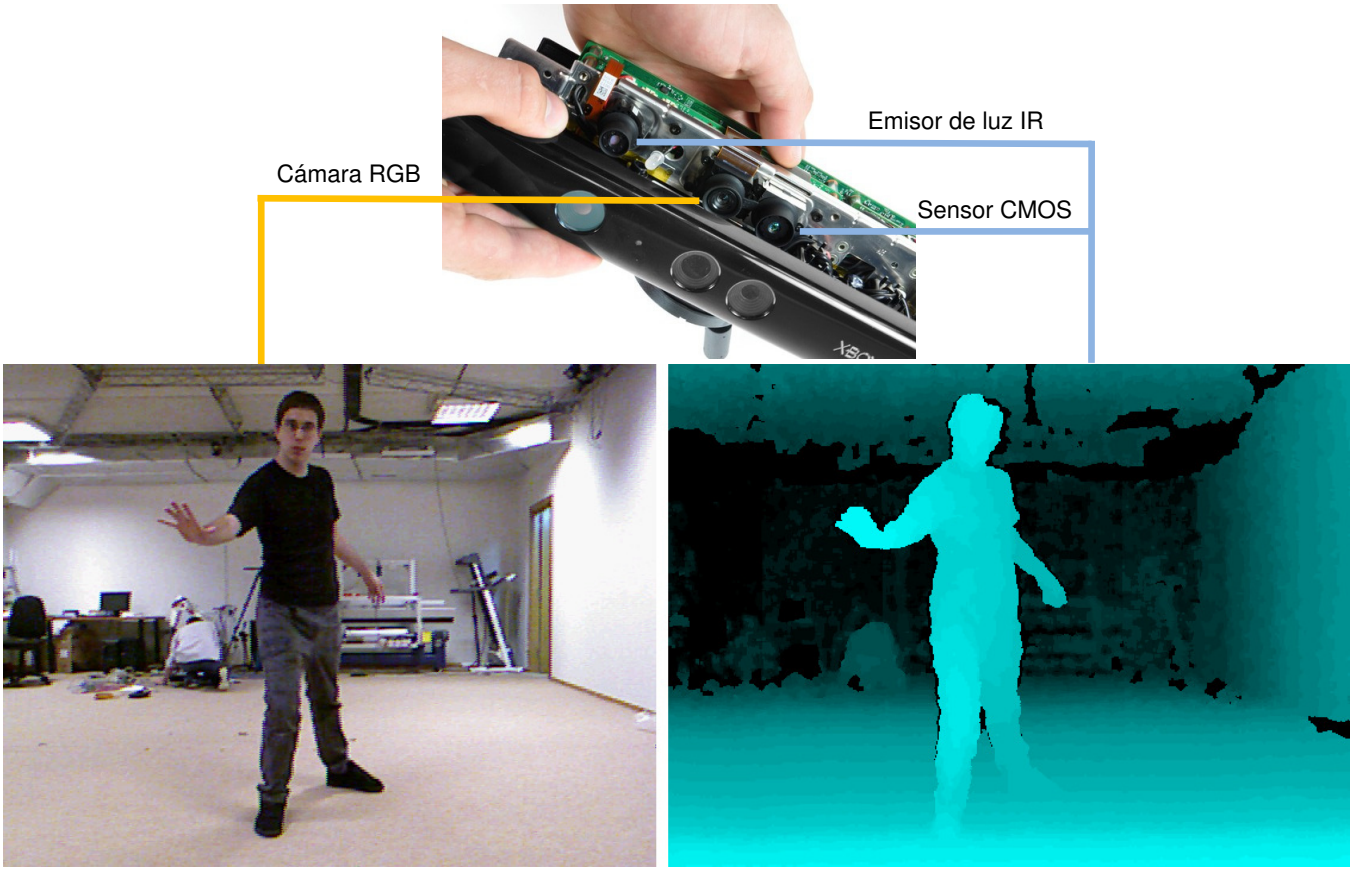

Imagen 3: Imagen RGB Vs Imagen de profundidad

De todas formas, aunque las características técnicas están bastante claras, ha habido cierta controversia sobre cómo se consigue la imagen de profundidad.

En un principio, Microsoft dio a entender que su cámara utilizaba tecnología de Tiempo de Vuelo*. Sin embargo, esto no parece muy creíble. Por una parte, se necesitan emisores y receptores de luz capaces de operar a nivel de nanosegundos, los cuales a día de hoy son muy caros (otras cámaras que usan esta tecnología rondan los 5.000 - 10.000$). Incluso así, los sensores actuales rara vez consiguen una imagen de profundidad de una precisión superior a 2 cm, mientras que Kinect ofrece una precisión de pocos milímetros y con menos ruido*.

Más tarde, se dio con la solución investigando la empresa desarrolladora de tecnología de la cámara, PrimeSense. Esta empresa tiene varias patentes, y todas están relacionadas con tecnología de proyección de patrones. Lo que hacen es proyectar un patrón conocido de puntos de luz sobre la superficie cuyas distancias se quieren calcular, determinando su geometría mediante el análisis de diferencias de la imagen obtenida con la del patrón original (que es lo que se habría obtenido de haberse proyectado en una superficie lisa). Esta técnica es muy inteligente y ofrece un resultado robusto en un hardware barato, por lo que es ideal para Kinect.

Con una cámara infrarroja común, podemos comprobar claramente el patrón que proyecta la cámara Kinect (ver imagen 4).

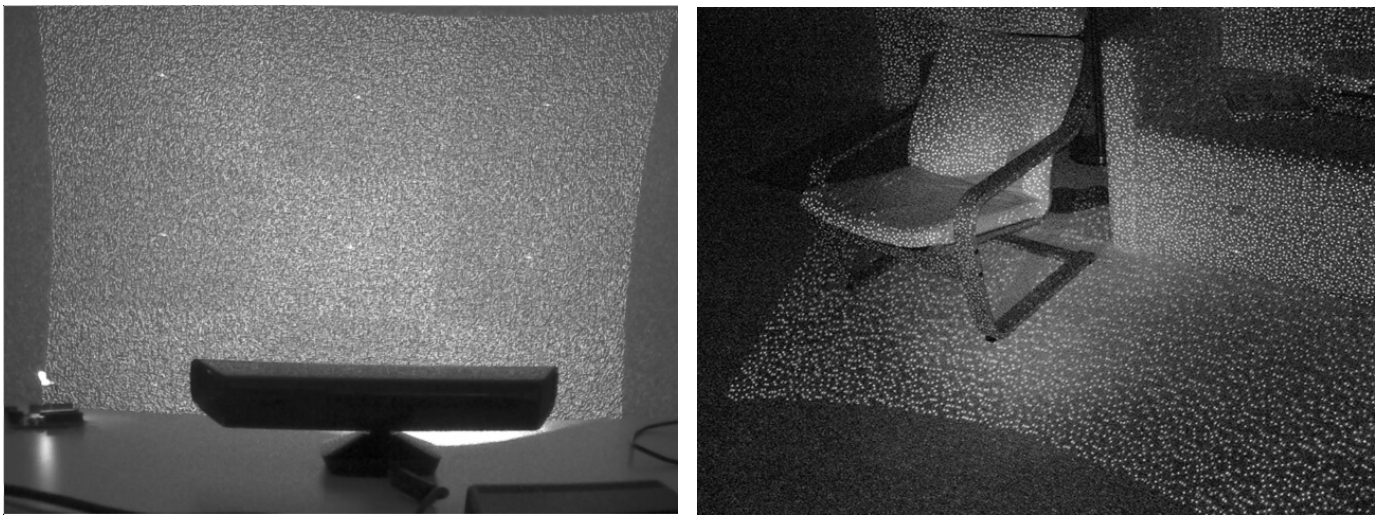

Imagen 4: Patrón de puntos de luz emitido por Kinect

4 FASE DE CAPTURA: CAPTURA CON OPENNI Y NITE

El objetivo de la fase de captura es conseguir un tracking* esquelético, que consiste en detectar unos puntos clave en el cuerpo del actor (usualmente articulaciones) y realizar un tracking de ellos de forma que después sea posible representar su movimiento con un esqueleto simplificado creado con esos puntos.

El primer paso, evidentemente, es instalar la cámara en el PC. Aunque se podría pensar que esto será complicado (ya que se trata de un accesorio para una consola doméstica), no lo es en absoluto. La cámara se conecta a la Xbox mediante el puerto USB, por lo tanto a nivel de hardware no hay ningún problema. A nivel de software, desde que la cámara salió a la venta han aparecido múltiples controladores no oficiales para hacerlo funcionar en PC. No es difícil encontrarlos en páginas web como kinecthacks.net. La mayoría ofrecen acceso a la imagen RGB y de profundidad, algunas también al motor del pivote y prácticamente ninguna al micrófono (que en todo caso no es algo que nos interese en este proyecto). Los sistemas operativos soportados son principalmente Ubuntu y Windows 7. Nosotros hemos optado por utilizar una versión no oficial liberada por PrimeSense, para sistemas operativos Windows Vista / 7.

4.1 El framework de OpenNI

OpenNI (Open Natural Interaction), es una organización sin ánimo de lucro creada en noviembre del 2010 que tiene entre sus miembros principales a PrimeSense. Han desarrollado un framework para Windows 7 y Ubuntu 10.10 que ofrece un set de APIs de código abierto que pretenden convertir en un estándar para aplicaciones de acceso a dispositivos de interacción natural (ver imagen 5). En muchos sitios al propio framework lo llaman OpenNI, lo cual puede crear confusión.

Para conseguir un tracking esquelético, existe un middleware libre para el framework de OpenNI llamado NiTE, también desarrollado por PrimeSense, que ofrece un tracking básico de 15 puntos (*cabeza, pecho, abdomen, hombros, codos, muñecas, caderas, rodillas* y *tobillos*), y que es capaz de capturar a dos personas simultáneamente. Es el que hemos utilizado como punto de partida en este proyecto. Todo el proceso de captura se ha implementado en C# como un proyecto

WPF (Windows Presentation Foundation), con un lienzo en el que se dibuja en tiempo real el esqueleto que está siendo capturado.

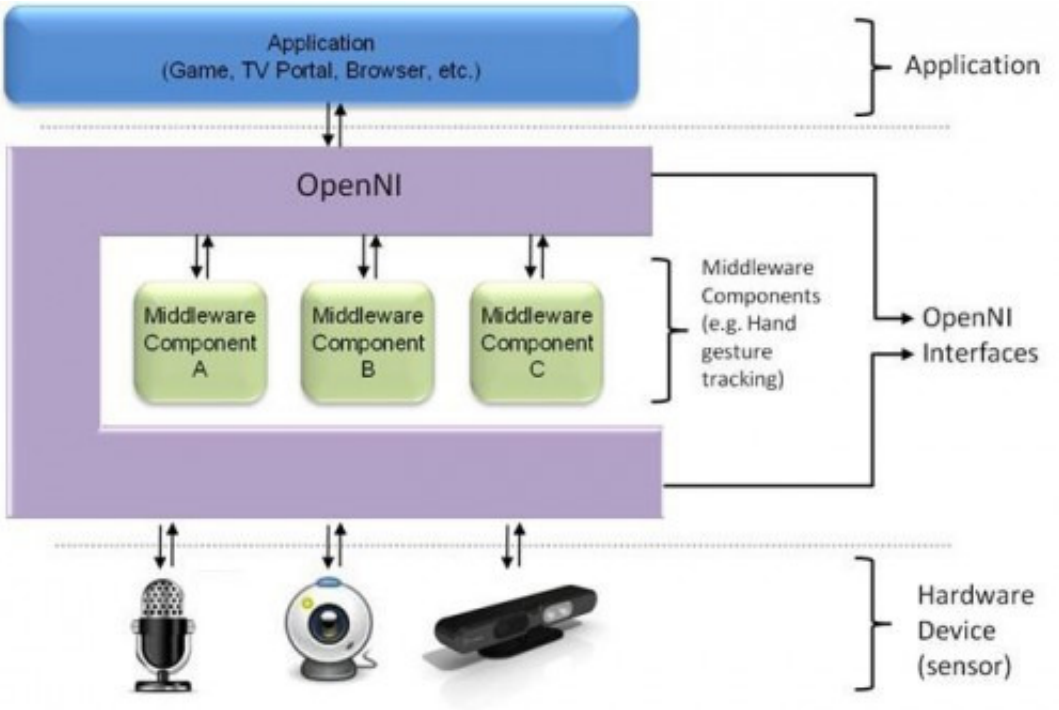

Imagen 5: Esquema del framework de OpenNI

En cuanto al tracking en sí, no está muy claro cómo funciona el tracking esquelético de NiTE, más allá de que lo hace basándose únicamente en la imagen de profundidad (es decir, no hace uso del RGB). Al estar desarrollado por PrimeSense, quienes han diseñado también el hardware de Kinect para Microsoft, es posible que utilicen una técnica parecida a la que ha publicado recientemente el grupo de investigación de Microsoft en Cambridge [7]. En la aproximación que proponen, se intenta predecir la próxima posición del cuerpo dada su posición actual utilizando un clasificador previamente entrenado con un variado dataset.

En todo caso, este tracking devuelve las coordenadas X, Y, Z de 15 puntos. Las coordenadas X e Y están en unidades de píxel y van de 0 a 640 para X y de 0 a 480 para Y, dependiendo de la posición en pantalla del punto. La coordenada Z, sin embargo, define en milímetros la distancia de la cámara a la que está el punto. Para poder trabajar con los puntos es necesario que las tres coordenadas compartan la misma unidad de medición, así que hemos calibrado la cámara hasta dar con la función que nos dice cuantos milímetros son un píxel a una profundidad determinada N (para así tener las tres coordenadas en milímetros):

$$\mathrm{mmPerPixel}_N = 1.15 \times \left(\frac{\sqrt{\mathrm{Depth}_N}}{10} + \frac{\mathrm{Depth}_N}{1000} - 3 \right)$$

4.2 El SDK Oficial

Durante el desarrollo de este proyecto, concretamente a mediados de junio de 2011 (siete mese después de que aparecieran los primeros hacks de Kinect) Microsoft liberó su propio SDK oficial, con APIs para C++, C# y Visual Basic. Se puede descargar y utilizar libremente, siempre que no sea con fines comerciales. Por supuesto, nosotros lo probamos inmediatamente, pero no resultó viable para nuestro proyecto. Veamos por qué.

El SDK oficial incluye sus propios drivers (solo compatibles con Windows 7) que ofrecen pleno acceso a la cámara, incluyendo la imagen RGB, la de profundidad, el motor del pivote y el micrófono. También incluye una extensa documentación sobre su uso así como el código de múltiples ejemplos listos para ser compilados y probados. A cambio, es necesario disponer del Visual Studio 2010 (puede ser una versión Express*) y el Microsoft .NET Framework 4.0.

En cuanto al tracking esquelético que proporciona, también hace uso solamente en la imagen de profundidad, como el de NiTE. En un principio parecía superior, ya que a los puntos que ofrece la versión de NiTE se le añaden los puntos correspondientes a las manos y los pies. Es capaz de mantener el tracking de dos usuarios simultáneos, pero además ya no se requiere que los actores adopten la clásica posición en cruz inicial, cosa que es necesario con el sistema de NiTE. Sin embargo, al probarlo vimos que no funcionaba como esperábamos.

El problema es que desde el principio Microsoft concibió Kinect como un dispositivo de interacción natural, no como cámara de captura de movimiento. Por lo tanto, su SDK (y su tracking esquelético) están claramente orientados a crear aplicaciones que se controlen con gestos del cuerpo. Es decir, se da por supuesto que el usuario estará en todo momento frente a la cámara, y el tracking funciona muy bien (mejor que el de NiTE) en estos casos. Sin embargo, al no estar contemplada la posibilidad de que el usuario se ponga de lado, o incluso le dé la espalda a la cámara (algo que ocurre constantemente en captura de movimiento), cuando esto ocurre el tracking del SDK oficial es absolutamente incapaz de seguir el movimiento. Por este motivo, continuamos utilizando el sistema de NiTE, que es capaz de mantener el tracking aunque el actor gire sobre sí mismo.

5 FASE DE CAPTURA: CAPTURA DE MANOS Y PIES

Siendo el del SDK oficial el único tracking libre disponible que ofrece las posiciones de las puntas de las manos y los pies, y teniendo en cuenta que no es suficiente para nuestro proyecto, es necesario añadir un módulo al tracking esquelético de NiTE para obtener esa información.

Como se verá en el capítulo 7, conocer estos puntos es totalmente necesario, sobre todo en el caso de los pies. En este capítulo se explica cómo se lleva a cabo la detección de las puntas de las manos y los pies en nuestro sistema. También se resumen brevemente otras aproximaciones que fueron descartadas por ofrecer resultados inferiores.

5.1 Captura de las manos

Lo primero que se hace es segmentar* la imagen para coger sólo la zona donde se encuentra la mano y aplicarle un thresholding* para aislar la mano del fondo, y después se le aplica un closing* para conseguir una silueta limpia.

Cabe destacar que en esta fase es muy importante aplicar técnicas muy sencillas que requieran de un coste computacional mínimo, pues todo se debe hacer online mientras la captura se está llevando a cabo y el tracking de NiTE consume bastantes recursos, incluso en procesadores de doble núcleo. Por ello, cuanto menos procesamiento extra se haga mejor, para así poder garantizar que la captura cumplirá la frecuencia de 30 frames estables que ofrece el output de la cámara.

Por suerte, en nuestro caso esto se puede hacer de forma muy sencilla, pues el tracking de NiTE nos da la posición de la muñeca, y para segmentar la parte de la mano nos limitamos a coger una ventana de W_{hand} x W_{hand} píxeles alrededor de ese punto. El valor de W_{hand} varía dependiendo de la distancia a la cámara a la que se encuentre la muñeca, de forma que esté siempre en concordancia con el tamaño de la mano. Para aplicar el thresholding utilizamos la imagen de profundidad. Usando como umbral el valor del píxel que corresponde a la muñeca, eliminamos todos los que sean significativamente más claros u oscuros. Después del thresholding se le aplica el closing para eliminar el ruido y conseguir una silueta limpia (ver imagen 6).

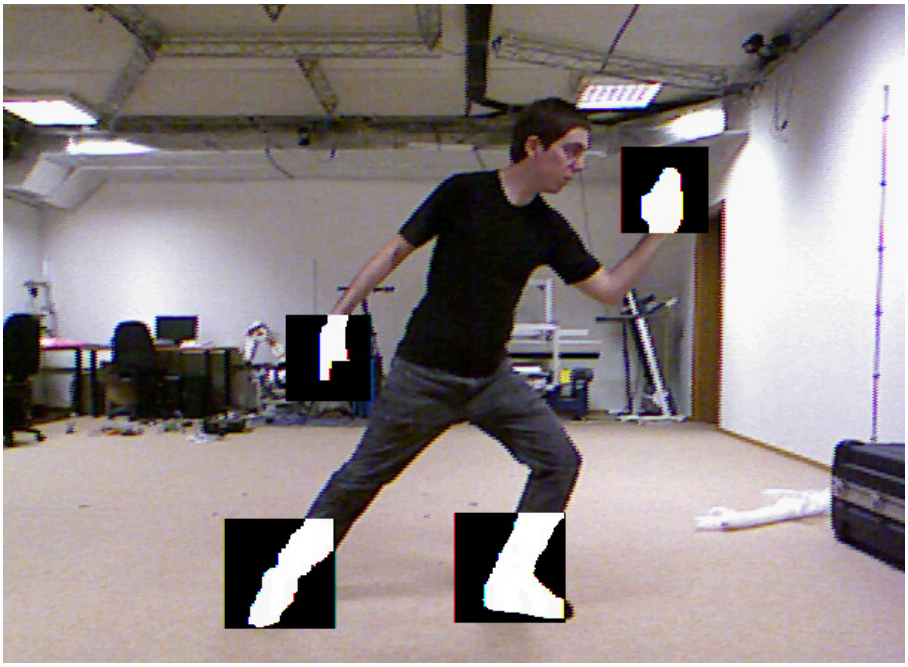

Imagen 6: Segmentación y thresholding de manos y pies

Ahora podemos proceder con la detección de la punta de la mano en ese segmento. Para determinar la posición de la punta de la mano, se parte de la base de la muñeca y se busca el punto de la mano que esté más alejado. Sin embargo, no se busca en todo el segmento de W_{hand} x W_{hand}, sino que se limita a una ventana de w'_{hand} x w'_{hand} alrededor del punto en el que fue detectada en el frame anterior (ver imagen 7). En las pruebas realizadas, los tamaños de W_{hand} y w'_{hand} (en píxeles) fueron los siguientes:

$$W_{hand} = \frac{7000}{Depth_{wrist}} \qquad w'_{hand} = \frac{W_{hand}}{8}$$

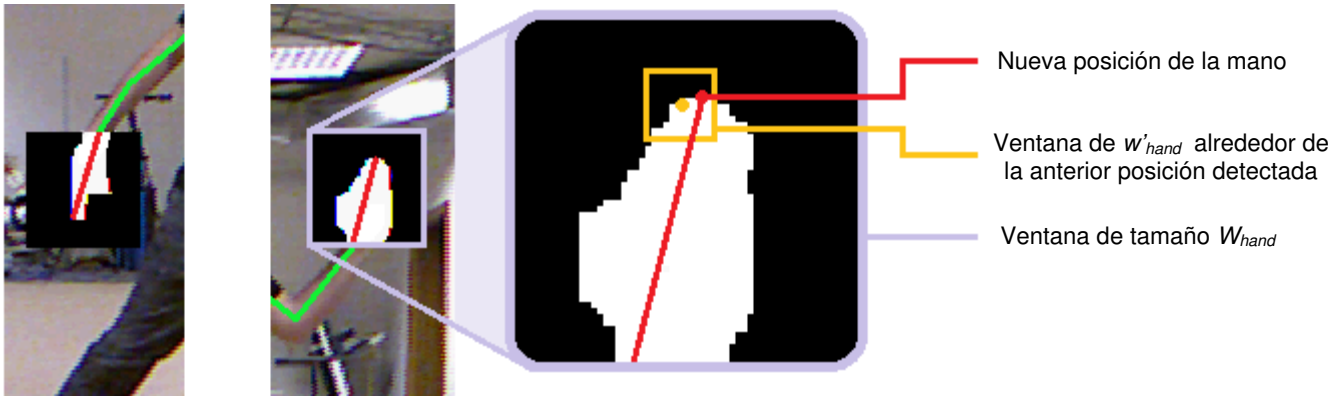

Imagen 7: Detección de las puntas de las manos

Una vez obtenidas las coordenadas X e Y, obtener la Z es muy sencillo. Sabiendo que la longitud de la mano es de 120 milímetros aproximadamente, tan solo hay que resolver esta ecuación:

$$Z_{wrist\text{-}hand} = \sqrt{120^2 - X_{wrist\text{-}hand}^2 - Y_{wrist\text{-}hand}^2}$$

Para obtener la Z de la mano hay que restarle o sumarle $Z_{wrist\text{-}hand}$ a la Z de la muñeca, dependiendo de si en la imagen de profundidad el píxel correspondiente a la mano es más claro o más oscuro del píxel correspondiente a la muñeca.

5.2 Captura de los pies

La captura de la orientación de los pies se hace de forma similar a la de las manos, pero con algunas diferencias. Como el tracking de NiTE nos proporciona el punto correspondiente al tobillo, podemos segmentar alrededor de él sin problemas. Sin embargo, a la hora de hacer el thresholding ya no podemos valernos de la imagen de profundidad, en el caso de los pies recurrimos a la imagen RGB. Esto requiere que el actor lleve un calzado de color lo más uniforme posible (y que este color sea distinto al del suelo del lugar en el que se vaya a realizar la captura) ya que antes de empezar se configura el programa para que utilice ese color como umbral para el thresholding. Por último, se aplica el mismo closing que a las manos, para obtener unas siluetas limpias (ver imagen 6).

En cuanto a la detección de la punta del pie, se hace igual que con la mano. Se busca el punto más alejado de la base del tobillo en una ventana de w'_{foot} x w'_{foot} píxeles alrededor del punto en el que fue detectado en el frame anterior (ver imagen 8). En las pruebas realizadas, los tamaños de W_{foot} y w'_{foot} (en píxeles) fueron los siguientes:

$$W_{foot} = \frac{9000}{Depth_{ankle}} \qquad w'_{foot} = \frac{W_{foot}}{8}$$

Aquí se ha comentado que la longitud de la mano es de unos 120mm. A lo largo del documento se mencionan varias proporciones del actor de pruebas (tamaño de manos y pies, estatura…). Es importante ajustar bien estos datos antes de empezar la captura para que esta salga lo mejor posible.

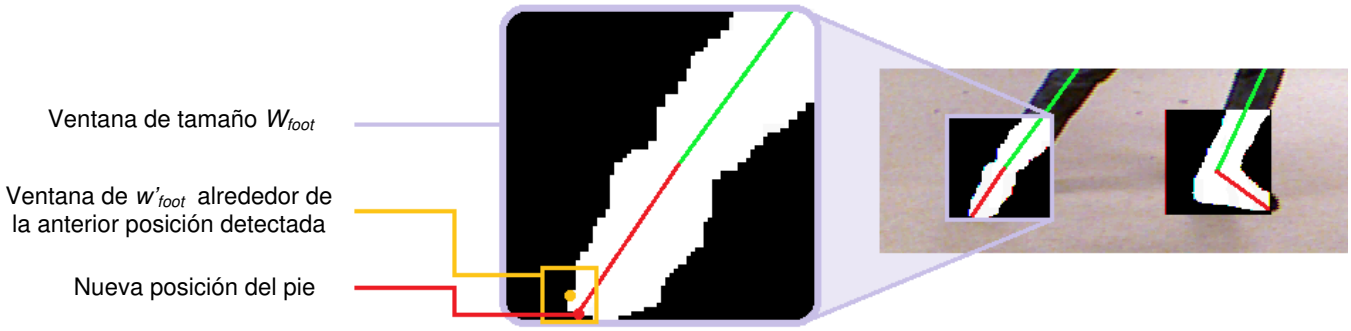

Imagen 8: Detección de las puntas de los pies

El cálculo de la coordenada Z también se hace igual que con la mano. Sabiendo que la longitud del pie es de 180 milímetros aproximadamente, tan solo hay que resolver esta ecuación:

$$Z_{ankle\text{-}foot} = \sqrt{180^2 - X_{ankle\text{-}foot}^2 - Y_{ankle\text{-}foot}^2}$$

Para obtener la Z del pie hay que restarle o sumarle $Z_{ankle\text{-}foot}$ a la Z del tobillo, dependiendo de si en la imagen de profundidad el píxel correspondiente al pie es más claro o más oscuro que el píxel correspondiente al tobillo.

Un problema común a la hora de animar pies es que, debido a que las mediciones no son exactas, estos pueden atravesar el suelo. Una solución sencilla para no encontrarnos con este problema es calcular el valor Y del suelo para la profundidad a la que está el pie, y darle ese valor a la coordenada Y del pie si el suyo es inferior (ver imagen 9). En nuestras pruebas las cámaras se encontraban a una altura de 89 cm respecto al suelo. Con la cámara a esa altura, hemos calculado que la función que para una profundidad determinada, nos da la Y (en píxeles) a la que está el suelo es la siguiente:

$$Y_{ground} = 346 + 2 \times \frac{4000 - Depth}{250} - 30 \times \frac{Depth - 2000}{2000}$$

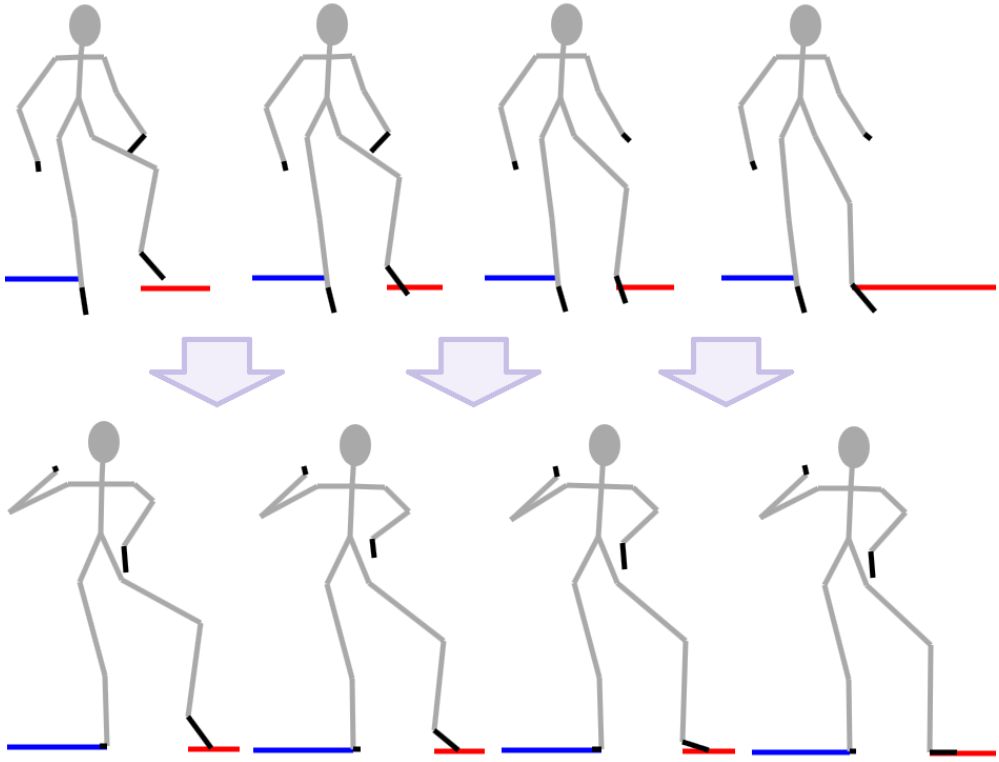

Imagen 9: Corrección del pie para que no atraviese el suelo

5.3 Otras experiencias

Evidentemente, el proceso que se ha narrado ha sido el resultado de constantes pruebas con distintas ideas y parámetros en busca del mejor resultado. A continuación se mencionan brevemente un par de aproximaciones que daban inferiores resultados.

Al principio se intentó realizar el tracking de los pies basándose en la imagen de profundidad, como con las manos. Sin embargo, la falta de definición del sensor de profundidad de Kinect hace que sea muy difícil la detección del pie si éste se encuentra en contacto con el suelo (ver imagen 3), por lo que al final optamos por utilizar la imagen RGB. Esta decisión fue complicada, ya que añade la restricción de que el actor debe llevar un calzado de distinto color al del suelo y además es la única vez en todo el proceso en el que se hace uso de esta cámara. Sin embargo, los resultados obtenidos fueron mucho mejores, así que se decidió hacerlo así.

También se intentó utilizar la imagen de profundidad para calcular el valor Z de las puntas de manos y pies. Por ejemplo, en el caso de las manos se utilizaba la diferencia de color entre el píxel correspondiente a la punta de la mano y el píxel correspondiente a la muñeca para intentar deducir la distancia en milímetros. Sin embargo, no fuimos capaces de definir una buena ecuación que nos trasladase de color de profundidad a milímetros, así que se nos ocurrió probar la solución que finalmente utilizamos.

6 FASE DE ANIMACÍON: COMBINANDO MÚLTIPLES CAPTURAS

Como se ha comentado en el primer capítulo, gracias a su sensor de profundidad Kinect proporciona las coordenadas X, Y Z de cualquier píxel de la imagen, por lo que no hace falta más de una cámara para triangular un punto.

Sin embargo, con una única cámara se dan constantes oclusiones. Estas se pueden dividir en dos grupos, las que se dan cuando una parte del cuerpo cubre a otra (el tronco puede ocultar el brazo si se está de lado, por ejemplo) y cuando una persona cubre a otra, al realizar captura con múltiples actores (en nuestro sistema dos como máximo, que es el número de personas que es capaz de seguir el tracking de NiTE).

Por lo tanto, se ha añadido una segunda cámara para intentar cubrir los puntos ciegos de la primera.

Todo el proceso de animación se ha implementado en C++.

6.1 El sistema multicámara

Montar un sistema que haga uso simultáneo de más de una cámara Kinect es problemático. Recordemos que Kinect detecta la profundidad mediante la emisión de un patrón de puntos de luz infrarroja, por lo que si se emplean varias Kinect para registrar una misma escena, lo más probable es que los patrones emitidos se mezclen haciendo que los sensores de profundidad den lecturas erróneas.

La única solución para evitar este problema es ubicar las dos cámaras con un ángulo de 90º entre ellas, de manera que se minimice al máximo el solape de patrones sobre el cuerpo del actor (ver imagen 10). En la práctica esta medida ha resultado ser bastante aceptable, con un número de errores de lectura razonablemente pequeño.

Por otra parte, hemos determinado que el rango de 2 a 4 metros de la cámara es donde el tracking de NiTE da mejores resultados, pues a una distancia menor a los 2 metros apenas es posible captar todo el cuerpo del actor, y a partir de los 4 metros las lecturas de profundidad son menos fiables. Por ello se ha fijado una superficie de captura de 2x2 metros (ver imagen 10).

En cuanto a la altura, las cámaras se han colocado a 89 cm del suelo, con una inclinación de 0º. Se ha optado por esta altura porque el actor de las pruebas medía unos 178 cm, por lo que así las cámaras lo tenían siempre centrado en el eje Y.

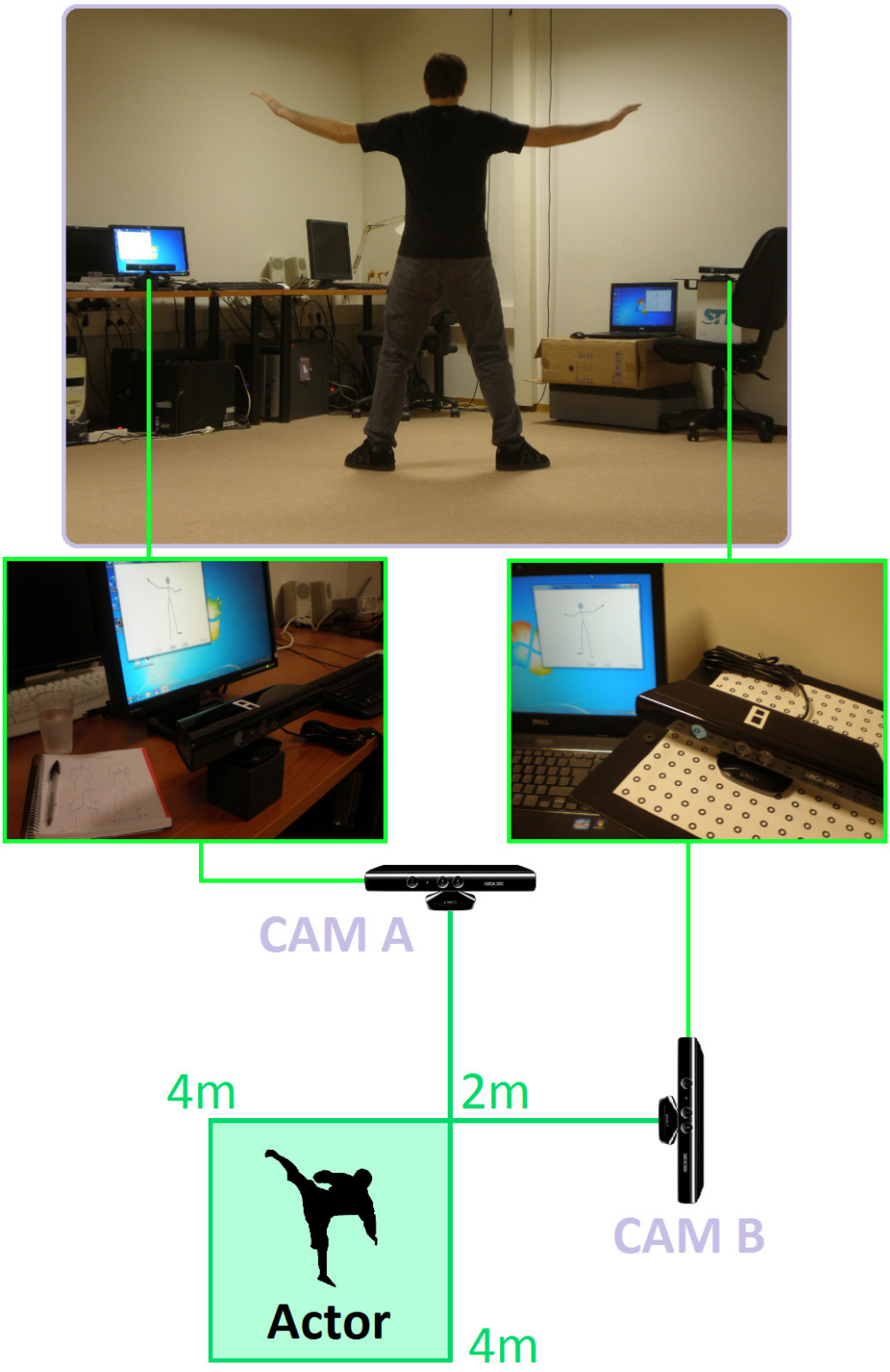

Imagen 10: El sistema multicámara

6.2 Combinando capturas

El primer paso de la fase de animación es unificar las capturas de las dos cámaras para crear una 'captura definitiva' que sirva para calcular los sistemas de referencia del esqueleto.

Por lo tanto, tenemos que escoger cada punto entre dos candidatos, el de la cámara A y el de la cámara B, en todos los frames. En nuestra solución, sin embargo, esta elección no se hace por cada punto, sino que definimos cuatro grupos de puntos:

Tronco: Incluye los puntos *cabeza, pecho, abdomen, hombro izquierdo, hombro derecho, cadera izquierda* y *cadera derecha*.

Brazo Derecho: Incluye los puntos *codo derecho, muñeca derecha,* y *punta de la mano derecha*.

Brazo Izquierdo: Incluye los puntos *codo izquierdo, muñeca izquierda,* y *punta de la mano izquierda*.

Piernas: Incluye los puntos *rodilla derecha, rodilla izquierda, tobillo derecho, tobillo izquierdo, punta del pie derecho* y *punta del pie izquierdo*.

Para seleccionar los puntos del grupo Tronco debemos saber qué cámara está captando más frontalmente el tronco del actor. Esto lo podemos averiguar calculando la distancia entre los dos hombros y entre las dos caderas, la cámara que tenga las mayores distancias será la que mejor esté captando el tronco.

Para seleccionar los puntos de los grupos Brazo Derecho y Brazo Izquierdo, por defecto se escogen los de la misma cámara que los puntos del grupo Tronco, a no ser que se dé una oclusión en el que el tronco tape el brazo. Esto se puede comprobar mirando si algún punto del brazo está en el área X, Y que delimitan los hombros y las caderas, y tiene un valor Z mayor a estos. Si se da este caso, se escogen los puntos proporcionados por la otra cámara.

En el caso del grupo Piernas no se hace selección alguna, ya que siempre se utilizan los puntos proporcionados por la cámara A. Tras haber probado varios criterios de selección, sorprendentemente este ha sido el que ha dado mejor resultado para la mayoría de los casos.

7 FASE DE ANIMACÍON: ANIMACÍON DEL ESQUELETO

Con las capturas de ambas cámaras unificadas en una sola, se procede a la animación en sí.

La animación esquelética se basa en formar un esqueleto de huesos que se definen por los puntos que se han obtenido en la captura. En nuestro caso, los 19 puntos de captura nos proporcionan 14 huesos (ver imagen 11).

Cada hueso tiene su propio sistema de referencia que determina dónde está y hacia donde apunta, dentro del sistema de referencia global. Para animar el esqueleto lo que se hace es calcular las rotaciones y traslaciones que hay que aplicar a los sistemas de referencia de cada hueso. En este capítulo veremos cómo se calculan esas rotaciones y traslaciones.

Los huesos se organizan de manera jerárquica, siendo en nuestro caso el hueso raíz la pelvis. Todas las traslaciones y rotaciones de un hueso también se aplican a sus hijos.

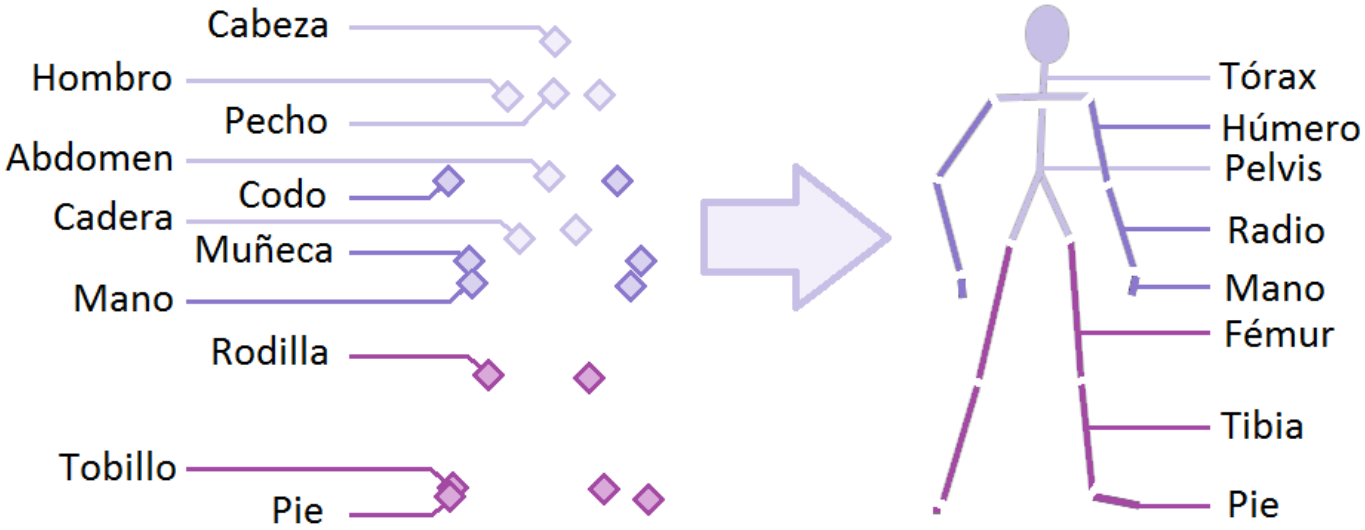

Imagen 11: Esqueleto formado por los puntos capturados

7.1 Filtrado de los puntos

Un problema habitual cuando se trata con capturas sin marcadores es la poca precisión de los puntos capturados. Por ejemplo, el punto correspondiente al abdomen no es el mismo punto preciso en cada frame, sino un punto aproximado. Lo mismo pasa con las caderas, hombros, etc. Como consecuencia, una animación basada en estos puntos presenta considerables vibraciones.

Además, también existe el problema de las lecturas erróneas. Ya sea porque el patrón emitido por una Kinect se ha superpuesto con la otra, o por algún otro motivo, a veces las lecturas de profundidad tienen picos que, de no solventarse, producen movimientos bruscos erróneos.

Una solución a estos dos problemas es el filtrado de puntos.

En procesamiento de señal, se denomina filtro a un proceso que elimina de una señal una característica determinada. Hay muchos tipos de filtros dependiendo del objetivo que se persiga [8], pero uno de los más clásicos es el filtro Butterworth, publicado por el británico Stephen Butterworth en 1930 [9].

El filtro Butterworth es un filtro diseñado para producir la respuesta más plana hasta la frecuencia de corte*. Por lo tanto, se puede utilizar como filtro pasa bajo*. En nuestro caso, se pueden interpretar los valores de los puntos capturados a lo largo del tiempo como una señal, y filtrarlos como tal. Se realizan tres filtrados paralelos (uno para los valores X, otros para los valores Y, y el último para los valores Z) de cada punto capturado, obteniendo como resultado unos puntos mucho más uniformes que nos proporcionarán una animación más suave, aparte de eliminar los picos puntuales correspondientes a errores de lectura.

Sin embargo, hay que tener precaución con este filtro, porque si se utiliza una frecuencia de corte muy pequeña se pueden eliminar movimientos que eran buenos. En nuestro caso, por ejemplo, hemos utilizado una frecuencia bastante baja porque los puntos proporcionados por el tracking de NiTE son poco precisos y cuanto más se suavicen, mejor queda la animación. Por lo tanto, en este sistema no se permiten movimientos muy bruscos, ya que son eliminados por el filtro.

En este proyecto se ha utilizado una implementación del filtro Butterworth como librería C++, que nos ha sido cedida por la empresa STT Ingeniería y Sistemas.

7.2 Cálculo de los sistemas de referencia

Cada hueso del esqueleto tiene su propio sistema de referencia, definido por el punto ($desp_x$, $desp_y$, $desp_z$) y los vectores R, U, D. Este sistema de referencia representa la ubicación y la dirección del hueso dentro del sistema de referencia global formado por el punto (0,0,0) y los vectores (1,0,0), (0,1,0) y (0,0,1). En la posición neutral* del esqueleto, los sistemas de referencia de cada hueso coinciden con el sistema de referencia global.

En la mayoría de los software de animación, un frame de animación se define como un punto *p* y tres ángulos (a_x, a_y y a_z) para cada hueso. Estos ángulos se conocen como ángulos de Euler*. El software, cuando anima, calcula el sistema de referencia de cada hueso partiendo del sistema de referencia neutral y aplicándole una rotación de a_x sobre el eje X, a_y sobre el eje Y y a_z sobre el eje Z. Después se traslada al punto *p*. Nosotros debemos hacer el trabajo contrario, es decir, calcular los sistemas de referencia a partir de los puntos que hemos obtenido de la captura, para luego calcular las rotaciones que hay que aplicarle al sistema de referencia neutral y obtener así los ángulos de Euler que le pasaremos al software de animación.

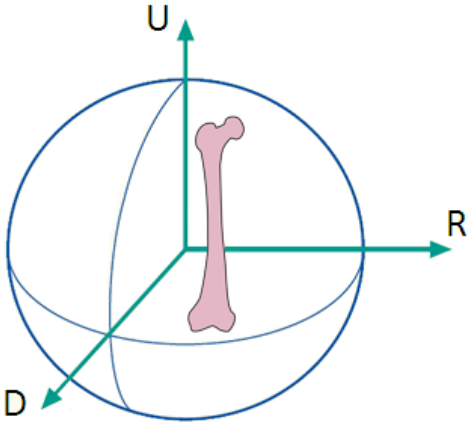

Imagen 12: Sistema de referencia de un hueso

Empezamos calculando los sistemas de referencia de cada hueso que se ajuste a los puntos capturados. Un sistema de referencia lo componen los vectores R, U, D, que son perpendiculares entre ellos (ver imagen 12). Hay dos cosas que se deben tener en cuenta para calcular estos vectores: La primera es que la multiplicación de dos vectores da como resultado un tercer vector que es perpendicular a los dos multiplicados. La segunda, que no todos los huesos pueden rotar libremente. Los codos, las rodillas y los tobillos están limitados a un único eje de rotación. A continuación se expone cómo calculamos esos vectores para cada hueso.

Pelvis:

Vector U = Punto *Pecho* – Punto *Abdomen*
Vector R = Punto *Cadera Derecha* – Punto *Cadera Izquierda*
Vector D = Vector U **x** Vector R

Tórax:

Vector U = Punto *Cabeza* – Punto *Pecho*
Vector R = Punto *Hombro Derecha* – Punto *Hombro Izquierdo*
Vector D = Vector U **x** Vector R

Húmero:

Vector U = Punto *Hombro* – Punto *Codo*
Vector R = Vector U **x** Vector U del Radio
Vector D = Vector U **x** Vector R

Radio:

Vector U = Punto *Codo* – Punto *Muñeca*
Vector R = Vector R del Húmero
Vector D = Vector U **x** Vector R

Mano:

Vector U = Punto *Muñeca* – Punto *Punta de la Mano*
Vector R = Vector U **x** Vector U del Radio
Vector D = Vector U **x** Vector R

Pie:

Vector U = Punto *Tobillo* – Punto *Punta del Pie*
Vector R = Vector U **x** Vector U de la Tibia
Vector D = Vector U **x** Vector R

Tibia:

Vector U = Punto *Rodilla* – Punto *Tobillo*
Vector R = Vector R del Pie
Vector D = Vector U **x** Vector R

Fémur:

Vector U = Punto *Cadera* – Punto *Rodilla*
Vector R = Vector R del Pie
Vector D = Vector U **x** Vector R

Como se puede ver, la manera de restringir las rotaciones de las rodillas y los tobillos es hacer que los vectores R del fémur, la tibia y el pie sean iguales (lo mismo pasa con los codos, en su caso el vector R del húmero y el radio deben coincidir). La dificultad está en calcular bien ese vector R. Si multiplicásemos los vectores U de la tibia y el fémur tendríamos problemas cuando la pierna esté extendida, pues los vectores U de la tibia y el fémur serán muy parecidos. El resultado es mucho mejor si multiplicamos los vectores U de la tibia y el pie, porque es mucho menos probable que esos dos huesos estén alineados. Por este motivo comentábamos en el capítulo 5 que conocer la orientación de los pies es muy importante.

Otro problema que nos encontramos a menudo son los movimientos corporales imposibles debido a errores de captura. En el caso del tracking de NiTE, es habitual que al tener la pierna extendida haya errores de profundidad en las rodillas, y por lo tanto parezca que estas están ligeramente flexionadas hacia el interior. Estos errores deben ser arreglados, ya que influyen muy negativamente en la credibilidad de la animación resultante.

En este caso también es necesario conocer la dirección del pie. De esta forma, si el ángulo entre el pie y la tibia es menor a *pi*, y el ángulo entre el fémur y la tibia es también menor a *pi* (ver imagen 13), sabemos que hay un movimiento de rodilla imposible y podemos corregirlo alineando la pierna (se le asigna a la tibia el sistema de referencia del fémur).

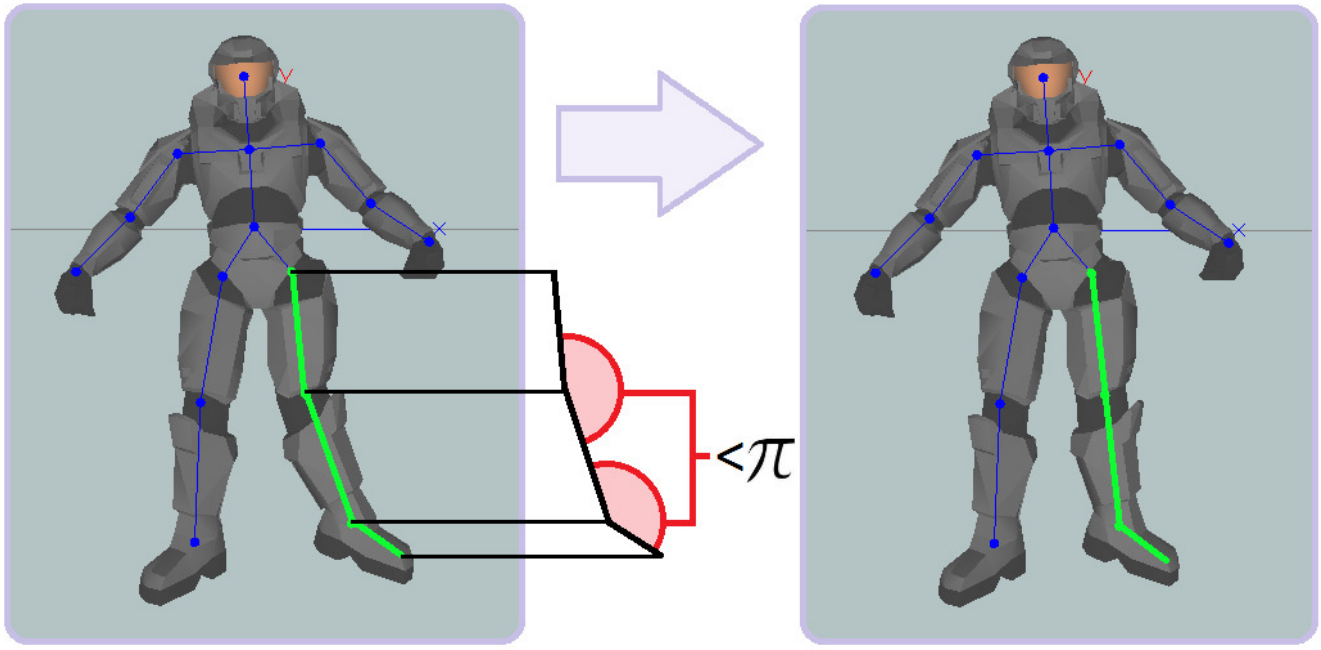

Imagen 13: Corrección de las rodillas para evitar movimientos imposibles

En este punto tenemos los sistemas de referencia de cada hueso calculados. Sin embargo, recordemos que los huesos están organizados jerárquicamente y al animar, las rotaciones que se le aplican a un hueso también se le aplican a sus hijos. Eso quiere decir que por cada hueso lo que nos interesa no es tener un sistema de referencia relativo al sistema de referencia global, sino un sistema de referencia relativo al sistema de referencia de su padre.

Conseguir la trasformación de los sistemas de referencia es muy simple. El primer paso es normalizar los vectores R, U, D de cada hueso.

Después, se procede al cálculo de los sistemas de referencia relativos. Para esto, se crean dos matrices, la primera con los vectores R, U, D del padre y la segunda con los del hijo. El sistema de referencia relativo se calcula multiplicando la matriz del hijo por la inversa de la matriz del padre:

$$\begin{pmatrix} R_x^{relat} & R_y^{relat} & R_z^{relat} \\ D_x^{relat} & D_y^{relat} & D_z^{relat} \\ U_x^{relat} & U_y^{relat} & U_z^{relat} \end{pmatrix} = \begin{pmatrix} R_x^{hijo} & R_y^{hijo} & R_z^{hijo} \\ D_x^{hijo} & D_y^{hijo} & D_z^{hijo} \\ U_x^{hijo} & U_y^{hijo} & U_z^{hijo} \end{pmatrix} \times \begin{pmatrix} R_x^{padre} & R_y^{padre} & R_z^{padre} \\ D_x^{padre} & D_y^{padre} & D_z^{padre} \\ U_x^{padre} & U_y^{padre} & U_z^{padre} \end{pmatrix}^{-1}$$

Como en este caso contamos con la ventaja de que trabajamos con matrices ortonormales, podemos simplificar el cálculo utilizando la traspuesta en vez de la inversa.

7.3 Cálculo de los ángulos de Euler

Una vez calculados los sistemas de referencia de cada hueso como sistemas de referencia relativos a sus padres, se procede a calcular los ángulos de Euler. Para ello basta con utilizar las siguientes ecuaciones:

$$a_y = \operatorname{asin}(R_z)$$

$$a_x = \operatorname{atan2}\left(\frac{-U_z}{\cos(a_y)}, \frac{D_z}{\cos(a_y)}\right)$$

$$a_z = \operatorname{atan2}\left(\frac{-R_y}{\cos(a_y)}, \frac{R_x}{\cos(a_y)}\right)$$

7.4 Cálculo de los desplazamientos

El sistema de referencia de cada hueso, como se ha dicho previamente, está formado por un punto además de los vectores R, U, D. Este punto nos indica el desplazamiento del hueso, es decir, su posición respecto al origen.

Igual que con las rotaciones, las traslaciones de un hueso se aplican también a todos sus hijos, Por lo tanto, es suficiente calcular la traslación de la pelvis, nuestro hueso raíz, para que todo el cuerpo se mueva en consecuencia.

Para ello, cuando se accede a la información del primer frame se guarda el punto correspondiente al abdomen, que servirá como punto inicial para calcular los desplazamientos posteriores.

Los desplazamientos sobre los ejes X y Z se calculan de forma intuitiva. Para conocer el desplazamiento en X se mide la diferencia en Z entre el punto inicial de la pelvis y el actual, según la cámara lateral (cámara B). En cambio, para conocer el desplazamiento en Z se mide la diferencia en Z según la cámara frontal (cámara A).

Los desplazamientos en Y son menos comunes (se dan cuando el actor salta o se agacha) pero un poco más complejos de calcular. Nosotros hemos utilizado de nuevo la función que, dada una profundidad, calcula el valor Y al que está el suelo. Conociendo la altura a la que está el suelo y la estatura del actor, podemos calcular si se ha agachado o ha saltado.

Ahora que tenemos los ángulos de Euler y los desplazamientos de todos los huesos, podemos guardarlos en uno de los muchos formatos de animación (mdl, smd, directx, etc.) y utilizar un software de animación 3D como 3DStudio MAX, Blender o fragMOTION para ver el resultado (ver imagen 14). Nosotros hemos optado por el formato smd, porque es probablemente el más simple de todos. A continuación se puede ver la estructura de un archivo smd.

```
version 1
nodes
   0 "Pelvis" -1
   1 "Torax" 0
   2 "Humero_Der" 1
   3 "Radio_Der" 3
   4 "Mano_Der" 4
   5 "Humero_Izq" 1
   6 "Radio_Izq" 6
   7 "Mano_Izq" 7
   8 "Femur_Der" 0
   9 "Tibia_Der" 9
   10 "Pie_Der" 10
   11 "Femur_Izq" 0
   12 "Tibia_Izq" 12
   13 "Pie_Izq" 13
end
skeleton
time 1
   0 1.400000 0.000000 11.800000 1.680700 0.027143 0.054225
   1 0.000000 55.000004 0.000000 0.002379 0.011543 -0.004587
   2 50.000000 0.000000 -5.000000 -1.819901 1.871594 0.124399
   3 0.000000 -55.000000 0.000000 -1.408730 0.000000 0.000000
   4 0.000000 -60.000000 0.000000 0.274839 0.234689 -0.000000
   5 -50.000000 0.000000 -5.000000 -1.786599 -1.831277 -0.014100
   6 0.000000 -55.000000 0.000000 -1.524630 -0.000000 0.000000
   7 0.000000 -60.000000 0.000000 0.518515 2.942881 -0.000000
   8 25.000000 -35.000000 0.000000 -0.156998 1.149418 -0.057381
   9 0.000000 -90.000000 0.000000 -0.000000 -0.000000 0.000000
   10 0.000000 -100.000000 0.000000 0.109903 0.004617 -0.036436
   11 -25.000000 -35.000000 0.000000 -0.266313 -0.711148 -0.068435
   12 0.000000 -90.000000 0.000000 -0.000000 -0.000000 0.000000
   13 0.000000 -100.000000 0.000000 0.375205 0.000000 -0.010242
time 2
                                        .
                                        .
                                        .
end
```

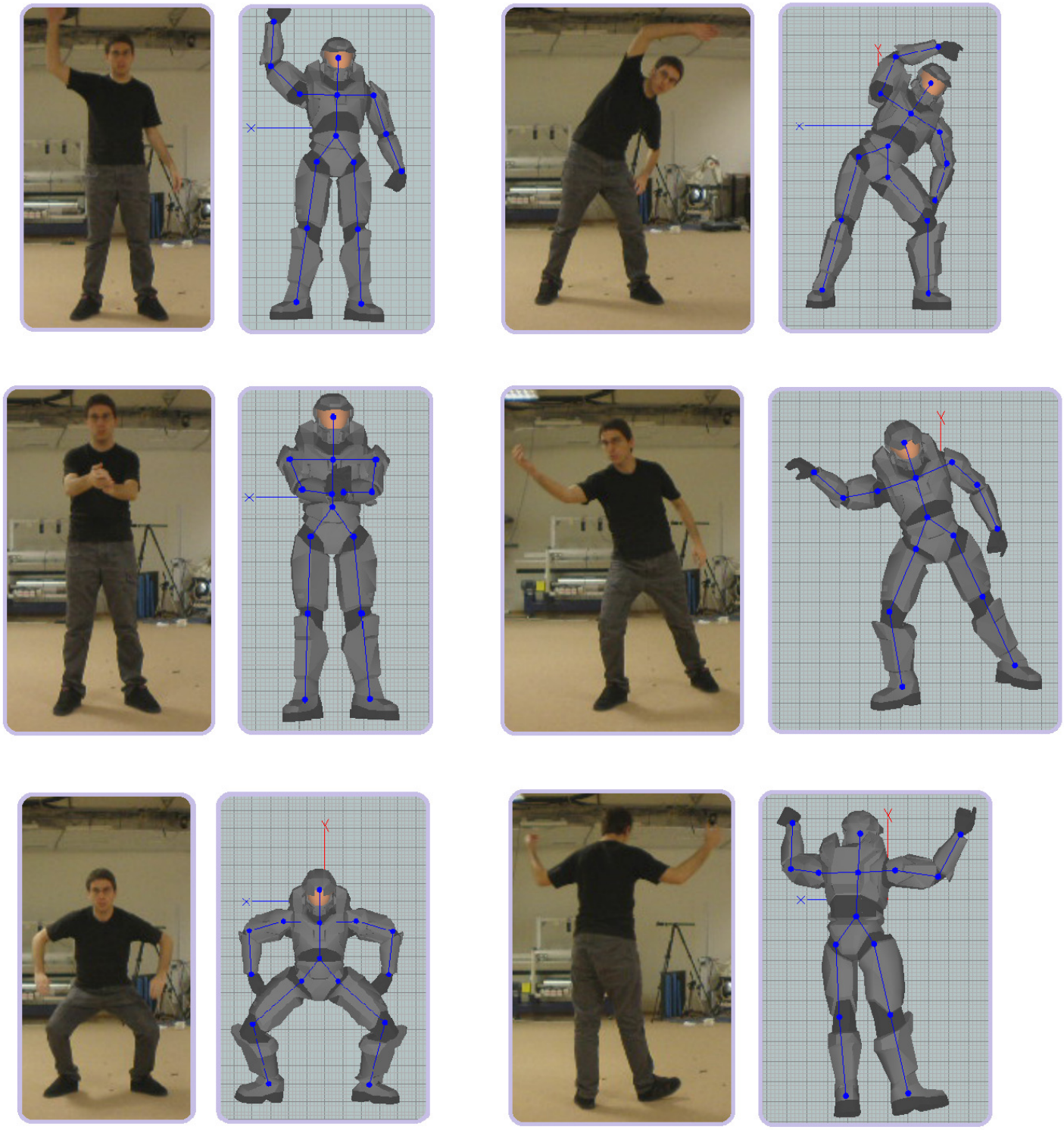

Imagen 14: Ejecución de la animación sobre un modelo 3D

8 CONCLUSIONES

Creemos que las conclusiones generales que podemos sacar de este trabajo son positivas. Se ha cumplido con el objetivo de realizar un sistema de captura de muy bajo coste, que funciona relativamente bien. Evidentemente, la precisión que ofrece no es la misma que la de un sistema de varios miles de euros, pero como solución de bajo coste podemos afirmar que cumple con nuestros objetivos.

Sin embargo, es cierto que tanto la cámara Kinect como el sistema de tracking de NiTE tienen ciertas limitaciones que no hemos podido solventar completamente, las cuales se comentan a continuación.

En cuanto a la cámara, su mayor limitación es el alcance de su sensor de profundidad. Teniendo en cuenta que más allá de 4 metros las lecturas empiezan a ser menos fiables, y más cerca de 2 metros el campo de visión de la cámara apenas capta todo el cuerpo del actor, contamos con un espacio de acción útil de 2 metros. Al ser este un sistema multicámara que emplea dos Kinect, nuestra área total de captura es de 2x2 metros, lo cual tenemos que admitir que es un espacio bastante limitado, sobre todo si se hacen capturas con dos personas (ver imagen 15). Recientemente la empresa Nyko ha lanzado un periférico llamado Nyko Zoom que disminuye la distancia focal de la cámara, ampliando su ángulo de visión (ver imagen 16).Teóricamente esto podría ayudar a ampliar nuestro área de captura, sin embargo durante el desarrollo de este proyecto no hemos tenido la oportunidad de probar este hardware.

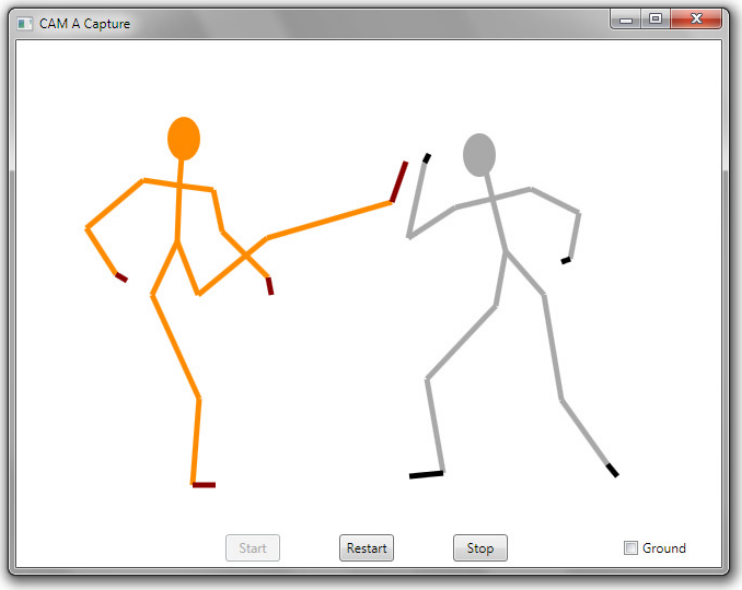

Imagen 15: Captura simultánea de dos actores

Imagen 16: Cámara Kinect con el periférico Nyko Zoom

El tracking de NiTE, al igual que la mayoría de sistemas de captura sin marcadores, tiene su defecto en la poca precisión de los puntos capturados. Si bien el filtrado de puntos nos ha ayudado a minimizar este problema, tampoco lo corrige completamente, y la animación final suele tener algunas imperfecciones y vibraciones que no se han podido eliminar al 100%. Además, disponer de tan solo cuatro puntos para definir la posición del brazo (la punta de la mano, la muñeca, el codo y el hombro) hace que sea imposible calcular correctamente algunas cosas, como por ejemplo las rotaciones de la mano y el hombro cuando el brazo está en extensión.

Como resultado de lo dicho anteriormente, tenemos una animación que no es lo suficientemente fina como para ser utilizada tal cual en una producción (como un videojuego, por ejemplo) pero que sí puede volverse utilizable puliéndolo un poco manualmente. Creemos que este tipo de solución puede ser interesante, pues es muchísimo más barato que un sistema de captura profesional, y fácilmente puede suponer un ahorro del 90% del trabajo si lo comparamos con la animación manual pura.

9 BIBLIOGRAFÍA

[1] Midori Kitagawa y Brian Windsor, *Facial Motion Capture*, MoCap for Artists, 2008.

[2] Roland Kehl y Luc Van Gool, *Markerless tracking of complex human motions from multiple views*, Computer Vision and Image Understanding, 2006.

[3] Beiji Zou, Shu Chen, Cao Shi y Umugwaneza Marie Providence, *Automatic reconstruction of 3D human motion pose from uncalibrated monocular video sequences based on markerless human motion tracking*, Pattern Recognition, 2009.

[4] E. Ceseracciu, Z. Sawacha, S. Del Din, S. Ceccon, S. Corazza y C. Cobelli, *Comparison of markerless and marker-based motion capture technologies through simultaneous data collection during gait*, Gait & Posture, 2009.

[5] Vijay John, Emanuele Trucco y Spela Ivekovic, *Markerless human articulated tracking using hierarchical particle swarm optimization*, Image and Vision Computing, 2010.

[6] Masaharu Toyoda, Shin Yabukami, Kazushi Ishiyama, Yasuo Okazaki, Ken Ichi Arai y Hiroyasu Kanetaka, *Wireless magnetic motion capture system using multiple LC resonant magnetic markers with high accuracy*, Sensors and Actuators A: Physical, 2008.

[7] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman y Andrew Blake, *Real-Time Human Pose Recognition in Parts from Single Depth Images*, Microsoft Research in Cambridge, 2011 (incluido como Anexo 1).

[8] Giovanni Bianchi y Roberto Sorrentino, *Electronic filter simulation & design*, ISBN 9780071494670, 2007.

[9] Stephen Butterworth, *On the Theory of Filter Amplifiers*, Experimental Wireless & the wireless engineer vol. 7, 1930 (incluido como Anexo 2).

Aparte de la documentación aquí expuesta, a lo largo del desarrollo del proyecto se ha consultado varios documentos internos de STT Ingeniería y Sistemas.

Aparte de la documentación aquí expuesta, a lo largo del desarrollo de este proyecto se ha tenido acceso a varios documentos internos de proyectos previos de la empresa de STT Ingeniería y Sistemas.

Universidad del País Vasco

Euskal Herriko Unibertsitatea

STT Ingeniería y Sistemas