Universidad de la República
Facultad de Ingeniería

Análisis de video en Biomecánica

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Andréi Guchin, Gonzalo Pereira, Guillermo Ottado, Mauricio Ramos.

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

Juan Cardelino ..... Universidad de la República

Tribunal

Alvaro Gómez ..... Universidad de la República
Dario Santos ..... Universidad de la República
Federico Lecumberry ..... Universidad de la República

Montevideo
viernes 3 julio, 2015

Análisis de video en Biomecánica, Andréi Guchin, Gonzalo Pereira, Guillermo Ottado, Mauricio Ramos..

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 180 páginas.
Compilada el viernes 3 julio, 2015.
http://iie.fing.edu.uy/

Agradecimientos

Este proyecto de fin de carrera realizado en la Facultad de Ingeniería de la Universidad de la República es un trabajo en el cuál, directa o indirectamente, participaron muchas personas. No queremos dejar pasar la oportunidad de agradecerles a todas ellas.

A Patricia Polero y a Darío Santos, por todo el material y los conocimientos brindados en un área en la que no teníamos experiencia previa.

A los profesores de la carrera, por transmitirnos los conocimientos necesarios para realizar un trabajo de este tipo.

A nuestro tutor, Juan Cardelino, por su esfuerzo y dedicación. Sus enseñanzas, críticas, consejos, su manera de trabajar, y sobre todo la paciencia y motivación que han sido fundamentales para sacar adelante este proyecto.

Finalmente a nuestras familias y amigos, por toda la ayuda y apoyo brindados, por aguantar las llegadas tardes, las ausencias, el cansancio y lamentablemente algún que otro mal humor.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

El objetivo de este proyecto es desarrollar un sistema óptico de captura de movimiento basado en marcadores para facilitar la tarea en el análisis biomecánico del movimiento de las personas.

La propuesta inicial fue realizada por investigadores de biomecánica del Departamento de Biofísica de la Facultad de Medicina de la Universidad de la República, Uruguay, en busca de una herramienta de código abierto que le permita obtener datos y estadísticas específicas que las herramientas existentes no pueden ofrecer.

Se elabora una aplicación con los bloques fundamentales que componen un sistema de estas características, utilizando los lenguajes *C/C++*, *Python* y *Matlab*. Estos bloques son independientes unos de otros, lo que da la posibilidad de modificarlos o sustituirlos sin afectar el resto del sistema.

También se crea un prototipo de base de datos, con secuencias de videos sintéticas, y un conjunto de algoritmos para medir la performance de cada bloque y del sistema en su totalidad.

Las pruebas realizadas sobre el software implementado reflejaron que el mismo tiene una precisión del orden del centímetro. Estos resultados son buenos para ser una primera versión y teniendo en cuenta que los algoritmos utilizados en cada bloque son de complejidad baja y se pueden optimizar en todos sus aspectos.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 1

Introducción

Este documento presenta las tareas realizadas en el marco del proyecto de fin de carrera "Análisis de Video en Biomecánica" realizado en la Facultad de Ingeniería, Universidad de la República (UdelaR). A continuación, se hace una introducción a la problemática encontrada, en función de la cual se plantea el objetivo del proyecto y una breve descripción de las soluciones actualmente disponibles.

Especialistas de distintos ámbitos académicos o profesionales se encuentran habitualmente en la necesidad de realizar estudios del movimiento del cuerpo humano. Esta tarea implica registrar la posición de miembros o articulaciones en el espacio y su correspondiente evolución en el tiempo.

Algunos ejemplos correspondientes a distintas áreas que ilustran estas necesidades son:

- A *nivel asistencial*, en el área de *fisioterapia*. Para realizar un seguimiento de la evolución de un paciente resulta importante en muchos casos conocer en detalle el movimiento, posición, etc. de la articulación o miembro afectado para llevar un control del mismo durante la rehabilitación y así poder determinar con mayor exactitud el estado y la evolución del paciente.

- *Investigación académica en biomecánica*. Diversos proyectos se llevan a cabo en las universidades sobre esta temática, por ejemplo, en la Facultad de Ingeniería de la Udelar se realizó el estudio comparativo de la patada delfín y la patada "crawl" respecto al nado de los peces por ondulación, tratando de explicar cómo la misma puede ser propulsiva. Para ello se analizaron videos de nadadores olímpicos mediante los cuales se obtuvo una secuencia temporal de las posiciones de diferentes partes del cuerpo durante varios ciclos de patada[1].

- *Medidas de performance en el deporte de alto nivel*. Cuando se habla de entrenamiento deportivo se hace referencia tanto a la mejora del rendimiento

[1] http://www.fing.edu.uy/ingenieriademuestra2011/proyectos/proyectos-del-instituto-de-f%C3%ADsica. Accedido 20-12-14

del atleta como a la optimización de las capacidades en función del deporte en el que se desempeña. La técnica deportiva está relacionada directamente con la optimización de estas capacidades por lo que una buena técnica permite el mejor aprovechamiento de las posibilidades físicas del atleta garantizando mejores resultados. Las mejores soluciones para la optimización de la técnica de un deportista consisten en el análisis de videos donde se puede estudiar en detalle cada uno de los movimientos del mismo.

- *Animación 3D.* Probablemente el más conocido de estos ejemplos, tanto en el diseño de videojuegos como en películas, programas de televisión o comerciales, muchas veces se requiere capturar el movimiento de un actor para interpretar un personaje ficticio y que sus movimientos parezcan lo más natural posible. En el medio local existen empresas que se dedican a esto, tales como Mopix [1].

En este contexto, el análisis de video es una herramienta fundamental para la recolección y estudio de datos. El seguimiento de puntos de referencia se utiliza para el cálculo de posición y otras variables asociadas como son la velocidad, la aceleración y por ende desplazamientos. Trabajar con video permite además estudiar secuencialmente situaciones estáticas en el tiempo. El seguimiento de dichos puntos resultaría tedioso si se hiciera manualmente por lo que resulta necesario contar con una herramienta que realice esta tarea automáticamente.

A raíz de esto, se crean los *sistemas de captura de movimiento*, que se definen como un conjunto de dispositivos y software que a partir de la grabación del movimiento de una persona (o cualquier otra cosa), es capaz de trasladar dicho movimiento a un modelo digital para diferentes fines.

Los ejemplos mencionados anteriormente definen distintos casos de uso con características disímiles, de manera que la búsqueda de una solución única que abarque las necesidades particulares de todos ellos resulta compleja. Por ejemplo, en el ámbito deportivo la velocidad del movimiento es una variable importante a tener en cuenta para desarrollar una solución, de esta variable depende la elección tanto del sistema de adquisición como de los algoritmos más eficaces para el registro del movimiento. De igual manera definir la portabilidad del sistema depende de si la actividad a relevar es en condiciones de laboratorio controladas o al aire libre, debido a la protección y transporte de equipos o las variaciones en las condiciones de iluminación, ruido, etc..

Al día de hoy, las soluciones de software disponibles que podrían asistir al especialista en su tarea, son mayormente comerciales. Las pocas alternativas de código abierto, carecen de las características necesarias para el especialista o están enfocadas hacia otras áreas de aplicación. Contar con este tipo de herramientas es fundamental para las necesidades de los equipos de profesionales, cuya alternativa son productos comerciales de alto costo.

En virtud de estas necesidades, este proyecto busca relevar y caracterizar los distintos casos de uso para un sistema de captura de movimiento, seleccionar uno que sea representativo y suficientemente general para poder realizar una aplicación básica y funcional de código abierto de análisis de video, que proporcione solución a las necesidades que se describieron ya sea utilizando como base algún proyecto de software libre existente, o en su defecto, desarrollando un prototipo de software básico completo que abarque el problema en forma general, para luego estudiar extender la aplicación hacia otros casos de uso.

Es importante resaltar la importancia de tener un sistema de principio a fin, con todas las etapas implementadas, de forma tal de abarcar todos los procesos que la captura de movimiento implica. Esto tiene como ventaja que se tendrá un panorama general del estado del arte de cada etapa del sistema, y sentará las bases para que proyectos futuros puedan optimizar individualmente dichas etapas acorde a las necesidades del cliente.

Los requerimientos para la implementación de este sistema provienen de investigadores del Departamento de Biofísica de la Facultad de Medicina de la Udelar, quienes utilizan este tipo de sistemas para el estudio del movimiento de personas tanto con fines terapéuticos como de investigación.

De acuerdo a lo planteado por el cliente y a las hipótesis que se establecieron en conjunto, se plantean los siguientes supuestos relativos a la apariencia del laboratorio:

- Las capturas de movimiento del paciente se realizan en un ambiente controlado:
  - La iluminación es la adecuada para realizar las capturas correctamente.
  - El fondo sobre el que se hacen las capturas es estático, de color oscuro y opaco.
  - Se utiliza cámaras convencionales y la ubicación de las cámaras así como sus parámetros de interés son conocidos.
- La vestimenta que utiliza el paciente es oscura, opaca y lo más ajustada al cuerpo como sea posible.
- Los marcadores a detectar son esferas blancas colocadas convenientemente sobre el cuerpo del paciente.

En cuanto al movimiento se asume que:

- El paciente permanece dentro del área de captura.
- Existe oclusión de marcadores en vistas dadas.
- Las cámaras se mantienen fijas.

Por otro lado, como se muestra en la Figura 1.1 se requiere que la salida del sistema sea la posición en el espacio 3D de los marcadores, ubicados en el cuerpo del paciente, en cada instante de la captura, con la correcta identificación de dichos marcadores en cada cuadro. A estos requerimientos, se le suman otros de carácter académico, como la importancia de tener una base de datos con un cierto número de secuencias y una método de evaluación de performance establecido. Estos dos aspectos toman gran relevancia en etapas futuras, de ampliación de este proyecto, ya que al realizar pruebas de nuevas implementaciones sobre los mismos datos y evaluar los resultados con iguales métricas se tiene un punto de referencia sobre el cual evaluar posibles mejoras del sistema.

En resumen, el sistema creado pretende bajo ciertas condiciones controladas, obtener las coordenadas espaciales de un número de puntos de interés sobre un paciente. Una de las formas de obtener esto, y la estudiada en este trabajo, es colocar al sujeto con traje negro con marcadores en un ambiente con iluminación adecuada, filmar con varias cámaras a lo largo del tiempo, adquirir esta información en la computadora y mediante un posterior procesamiento obtener la posición 3D de cada uno de los puntos de interés (los marcadores) a lo largo de toda la secuencia.

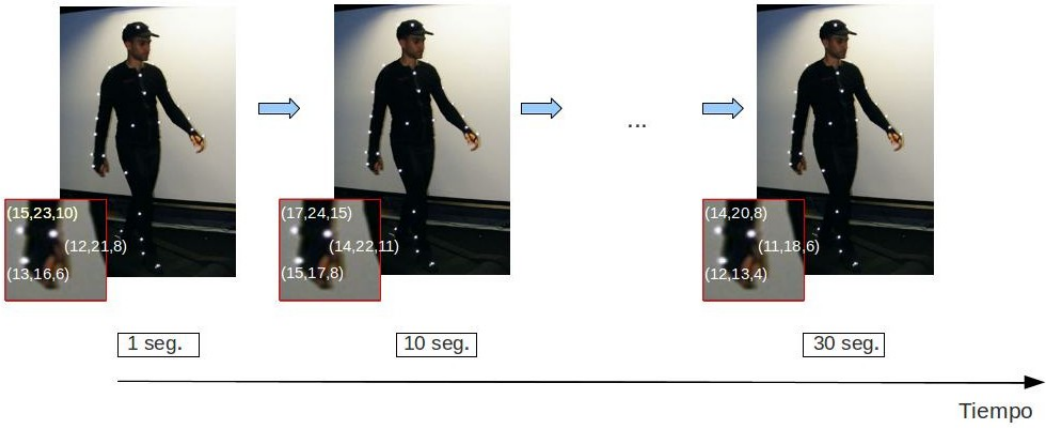


Figura 1.1: Posición de los marcadores a lo largo del tiempo.

La aplicación recibe como entrada imágenes de video provenientes de varias cámaras, las cuales capturan el movimiento de una persona desde distintos ángulos. Se implementan métodos que permiten obtener información sobre las cámaras y su disposición en el espacio mediante la calibración. Con esta información y a partir de las imágenes obtenidas se detectan distintos puntos de referencia del cuerpo según el estudio particular que se desee realizar. Posteriormente, a partir de la información de las múltiples cámaras se reconstruye la posición 3D de los puntos en cada cuadro, para luego efectuar la identificación de trayectorias de los puntos de interés a lo largo de la secuencia, ver Figura 1.2. A partir del procesamiento de la posición 3D de los puntos se obtendrán otros datos estadísticos de interés para el usuario.

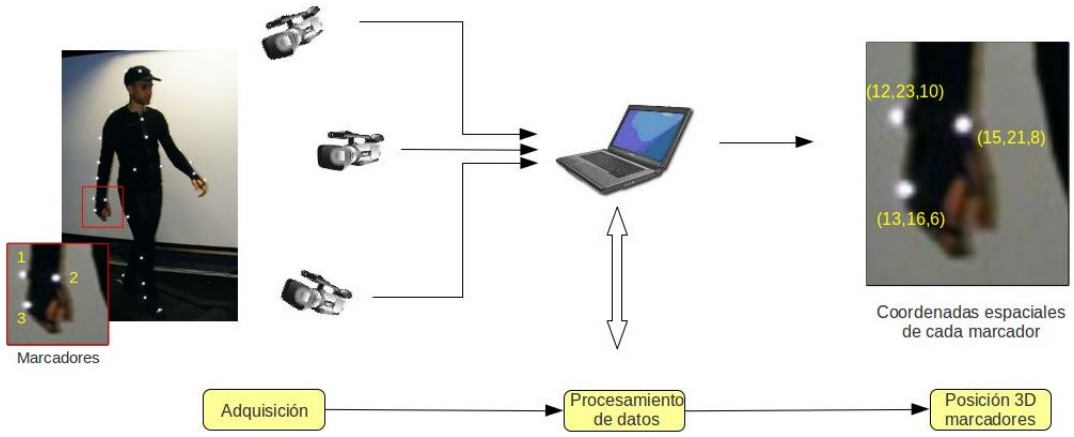


Figura 1.2: Funcionamiento de un sistema de captura de movimiento.

El proyecto se inicia con investigación y revisión bibliográfica sobre la temática, buscando diseños provenientes de sistemas de captura de movimiento ya implementados, así como detalles sobre la implementación parcial o total de cada funcionalidad de interés.

Se procura obtener una base de datos con secuencias, ya sean reales o sintéticas, sobre los cuales implementar la aplicación y sus distintos bloques. Si bien esta búsqueda no aporta los resultados esperados, debido a que no se encontraron bases de datos acorde a las necesidades del proyecto, se implementa un prototipo de base de datos sintética con un número reducido de secuencias como alternativa tomando algunos conceptos encontrados en dicho relevamiento, generando una base flexible, de fácil expansión y con potencial para futuros estudios.

Luego de analizar la bibliografía y definir la metodología, se opta por implementar el sistema con los 4 bloques principales mostrados en la Figura 1.3.

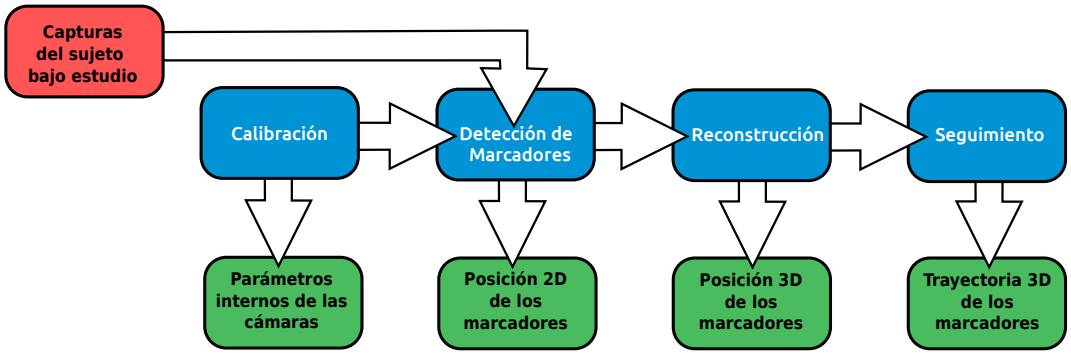


Figura 1.3: Diagrama de bloques del sistema a implementar.

Cada uno de estos bloques cumple un objetivo definido:

- El bloque de *calibración*, es el encargado de obtener los parámetros internos y externos de las múltiples cámaras utilizadas en la captura, permitiendo establecer la relación entre el espacio ambiente y la imagen sobre cada cámara. Una vez que se disponen el número y posición de las cámaras para efectuar la captura de movimiento, este bloque releva por única vez la información previa necesaria para realizar una correcta reconstrucción de los puntos.

- El bloque *detección de marcadores*, identifica los marcadores en las imágenes obteniendo la posición 2D de cada uno de ellos a lo largo de la secuencia de captura en cada cámara (ver Figura 1.4b).

- Con información de posición 2D proveniente de la detección de marcadores en múltiples vistas, y los parámetros relevados en calibración, el bloque de *reconstrucción* (ver Figura 1.4c) se encarga de obtener la posición 3D de cada marcador.

- Por último, el bloque de *seguimiento* (ver Figura 1.4d) (o "tracking") relaciona cada marcador en determinado cuadro de la secuencia con los que le siguen en el resto de los cuadros, obteniendo así la trayectoria de cada marcador a lo largo del tiempo.

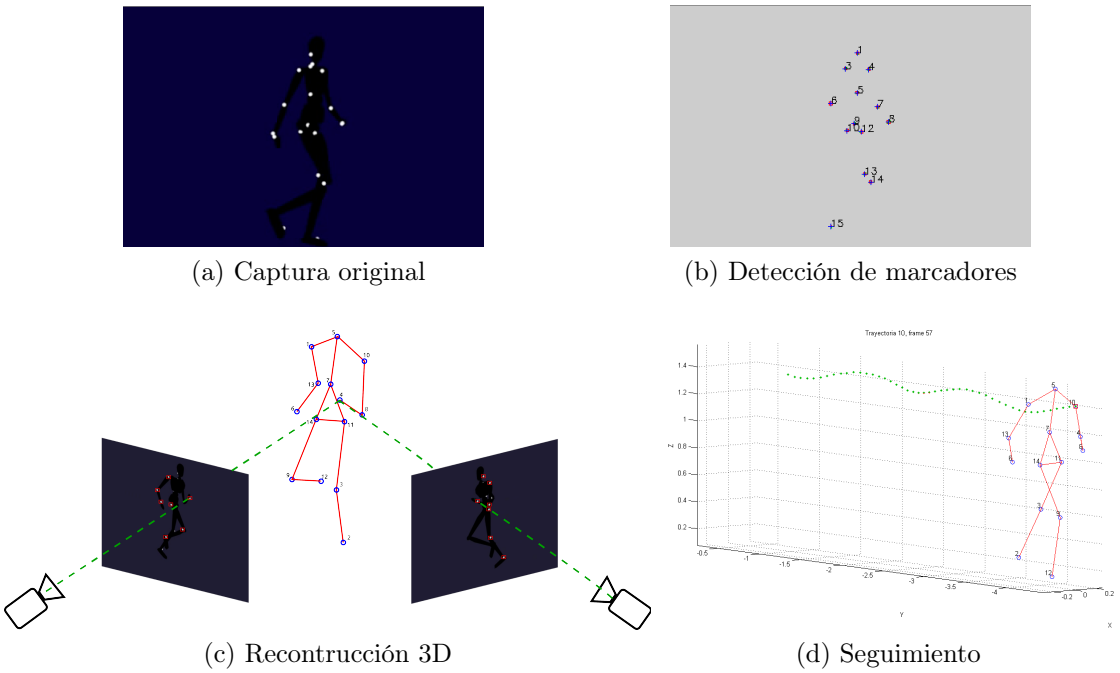
Cabe destacar que estos bloques se implementaron de manera independiente, con una interfaz claramente definida, de manera de poder realizar modificaciones sobre uno de ellos sin afectar el funcionamiento de los otros. Este aspecto cobra relevancia en etapas futuras, donde bajo la necesidad de optimizar algún bloque en particular o agregar alguna nueva característica que permita procesar datos provenientes de nuevos casos de uso, se deba modificar el sistema. De no presentar esta particularidad, la modificación de un bloque podría afectar el funcionamiento del sistema completo, requiriendo una posible re-ingeniería de la estructura.

Por otro lado, fue elaborada una interfaz gráfica básica de forma tal de facilitar la ejecución del software para los usuarios sin tener que trabajar directamente con el código.

A continuación se presenta una breve revisión de los sistemas de captura de movimiento disponibles. Una descripción más detallada y más técnica se realizará en la Sección 2, luego se define el objetivo del proyecto y se comenta la estructura de la documentación.

1.1. Revisión de soluciones existentes

Al día de hoy existen varios sistemas de captura de movimiento, sin embargo los mas usados debido a su buena performance y soporte presentan altos costos de licenciamiento.



(a) Captura original

(b) Detección de marcadores

(c) Recontrucción 3D

(d) Seguimiento

Figura 1.4: Salidas de los bloques principales del sistema.

Como se mencionó anteriormente, los más populares en la actualidad son los que se utilizan en la industria del cine o del diseño de videojuegos. En este contexto, se almacenan las acciones de actores humanos y se usa esta información para animar modelos digitales de personajes en animación 3D.

Algunos ejemplos de sistemas de captura de movimiento bajo licencia son:

- *Vicon* [2]. Es una empresa que vende sistemas de captura de movimiento, tanto software como hardware. Sus sistemas son muy conocidos y fuertemente utilizados en estudios clínicos y biomecánicos alrededor del mundo, así como para otras investigaciones científicas. En particular, este sistema está siendo utilizado en el Departamento de Fisiatría y Rehabilitación del Hospital de Clínicas. Presenta como ventajas frente a otros sistemas la velocidad con que realiza el procesamiento de los datos y la calidad de las cámaras para realizar las capturas. La gran desventaja que posee es el alto costo de sus equipamientos, bastante privativo en algunos ámbitos.

- *Qualisys* [3]. Junto con Vicon son los dos sistemas de captura de movimiento más utilizados en el ámbito de la investigación científica. Presenta como ventajas frente al anterior la utilización del modelo AIM[2], un identificador de marcadores que "aprende" de cada secuencia procesada ahorrando mucho tiempo y trabajo en identificar marcadores. Además presenta una interfaz de usuario más amigable y requiere menos tiempo de capacitación para su uso.

[2]Automatic Identification of Markers

- *OptiTrack* [4]. Es de los sistemas comerciales de menor costo, cerca de la cuarta parte de lo que cuesta un sistema Vicon. Otra de sus ventajas es que brinda acceso de bajo nivel a través de un conjunto de SDK's y API's tanto para el hardware como para los datos obtenidos, permitiendo manipular los mismos y procesarlos de la manera deseada por fuera de las aplicaciones que originalmente provee.

- *Massive* [5]. Es un software destinado a producir efectos especiales para cinematografía, programas televisivos y videojuegos entre otras cosas. El software implementa la captura de movimiento y cuenta con varios productos para generar distintos tipos de efectos y animaciones. Fue originalmente diseñado par utilizar en la trilogía de El Señor De Los Anillos de Peter Jackson y desde entonces se ha convertido en uno de los mejores software para realizar efectos visuales de muchedumbres y animación de personajes autónomos.

- *Motion Analysis* [6]. es el mayor fabricante mundial de sistemas de instrumentación óptica de alto rendimiento que permite testear y medir el movimiento de los objetos. Mantiene y comercializa el software con sus sistemas de hardware. Sus sistemas evalúan el movimiento en distintas aplicaciones: producción de animación, análisis de movimiento, y en el ámbito industrial. Posee un gran número de aplicaciones de pos-producción.

- *PhaseSpace* [7]. Posee un sistema de captura patentado basado en marcadores leds, de alta resolución temporal y espacial. Todo el sistema se compone de sólo unos pocos elementos robustos y portátiles. Sus aplicaciones funcionan tanto en entornos Linux como Windows. Ha desarrollado soluciones de captura de movimiento para la investigación, la industria y las comunidades de las artes gráficas, también para estar al alcance de las pequeñas empresas, universidades y particulares.

Los sistemas anteriores, y la mayoría de los sistemas modernos que efectúan capturas de movimiento, utilizan sensores infrarrojos o LEDs para efectuar el seguimiento de puntos. Estos dispositivos facilitan la etapa de detección de marcadores en cada cámara.

La gran desventaja de los sistemas anteriores es su alto costo. A modo de ejemplo, un sistema Vicon de 10 cámaras tiene un valor que ronda los U$S 70.000 [3].

Por otro lado, también existen programas de captura de movimiento de código libre, un ejemplo de ello es el programa Kinovea [8]. Enfocado principalmente en el ámbito deportivo, permite analizar y mejorar la técnica de los atletas a través del procesamiento de secuencias de video. Posee algunas características interesantes como la posibilidad de efectuar medición de distancias, ángulos y tiempos manualmente, así como la utilización de seguimiento semi-automático de puntos en

[3]Este precio depende fuertemente del tipo de cámara que se utilice, los precios de las mismas se encuentran a partir de los U$S 3.000 por unidad.

tiempo real para obtener su trayectoria. La gran desventaja que presenta frente a otros sistemas es que efectúa únicamente análisis sobre dos dimensiones y con una sola cámara.

Otra alternativa sobre la que se investigó es DVIDEOW [9]. Este software presenta varios conceptos teóricos como primer acercamiento al problema, pero la implementación presenta pérdidas de marcadores que requieren correcciones constantes. Además, no se tiene una actualización del mismo desde el 2009, desde esa fecha a la actualidad solamente se encontraron trabajos que lo presentan como fundamento teórico.

1.2. Objetivo del proyecto

Desarrollar un prototipo de software básico pero completo que permita tener un sistema de captura de movimiento de punta a punta.

Diseñar una base de datos que permita organizar la información recabada en las capturas para un posterior procesamiento.

Definir un protocolo para la adquisición de secuencias de movimiento basado en un sistema de captura óptico.

Contar con medidas de performance que permita evaluar y comparar los algoritmos desarrollados.

1.3. Estructura de la documentación

En este capitulo se presentó el objetivo general del proyecto y un breve estado del arte de la temática que será desarrollado profundamente en el Capítulo 2, junto a un planteo en la Figura 1.3 de los bloques que se estudiarán a lo largo del proyecto.

El Capítulo 3 presenta un estudio de bases de datos disponibles y como fueron generadas capturas experimentales propias sobre las cuales trabajar. Luego el Capítulo 4 presentará detalles sobre el procedimiento de la Figura 1.3 a partir de los datos experimentales generados.

En los capítulos siguientes se presentará cómo fue estudiado e implementado cada bloque: Segmentación y Filtrado en el Capítulo 5 para la detección de la posición de los marcadores en cada video, Calibración en el Capítulo 6 donde se mostrará como se genera la información del espacio de captura necesaria para luego en el Capítulo 7 sobre Reconstrucción, mostrar como se combina la información de los bloques anteriores para calcular la posición en el espacio de los marcadores.

Posteriormente, el Capítulo 8 sobre Seguimiento propone un método de identificación de marcadores reconstruidos.

Finalmente se presenta cómo fue evaluado cada bloque y múltiples pruebas sobre el sistema completo en el Capítulo 9 para luego presentar las conclusiones en el Capítulo 10.

Capítulo 2

Revisión bibliográfica

Como se menciona en la introducción, la primera etapa realizada en el proyecto fue la revisión bibliográfica de sistemas de captura de movimiento, de forma tal de obtener una idea de los distintos procesos que conforman un sistema de este tipo. Los procesos principales son:

- Calibración.
- Adquisición.
- Detección de marcadores (Segmentación y Filtrado).
- Seguimiento.
- Estimación de pose (Reconstrucción).

Al no encontrar detalles sobre el diseño de sistemas que verifiquen las hipótesis de este proyecto, o que puedan reutilizarse como base, se decide implementar uno de manera íntegra, realizando una búsqueda bibliográfica detallada y asignando cada artículo a una categoría de acuerdo a los procesos definidos anteriormente. En el Apéndice A, se muestra una Tabla con toda la bibliografía relevante, ordenada según las categorías antes mencionadas.

Se clasificaron una serie de artículos y publicaciones dentro de las cuales se destacan:

- Calibración
  - *Optical Motion Capture System with Pan-Tilt Camera Tracking and Realtime Data Processing* [10]. Seguimiento orientado a capturas en tiempo real, identificando marcadores por árboles de decisión, y calibración de cámaras.
- Detección de marcadores

- *Simple and robust hard cut detection using interframe differences* [11]. Esta investigación se propone un posible algoritmo de segmentación para utilizar en la etapa de detección de marcadores, utilizando un método de detección de bordes. Por motivos que se verán más adelante, estos tipos de algoritmos se descartaron para utilizar en segmentación (*ver Capítulo 5*).
- *Threshold survey* [12]. Se listan 40 algoritmos de segmentación mediante métodos de umbral. Se evalúan los mismos y se muestran ventajas y desventajas de cada uno. Entre ellos se encuentra el método de umbral de Otsu [13], el cual fue el elegido para implementar en este sistema (*ver Capítulo 5*).

- Seguimiento

  - *Skeleton-Based Motion Capture for Robust Reconstruction of Human Motion* [14]. Utilizando un modelo para el cuerpo durante la reconstrucción 3D, se presenta un método para robustecer los sistemas de captura ópticos basados en marcadores. Investigación sobre la que se decide basar este trabajo, presenta múltiples algoritmos accesibles, con un nivel de descripción aceptable.
  - *Modelling and Tracking Articulated Motion from Multiple Camera Views* [15]. Utilizan un sistema óptico con múltiples cámaras basado en marcadores, no se dan detalles del proceso de captura. Describen un modelo general que integra, un modelo cinemático basado en medidas angulares y un modelo para el proceso de medida. Manejan oclusiones y para el seguimiento utilizan técnicas basadas en filtro de Kalman extendido y filtros Bayesianos. Estimación para cada cuadro de los parámetros de un modelo de esqueleto. Como el modelo es 3D, implícitamente se reconstruye la pose al mismo tiempo.
  - *Skeletal Parameter Estimation from Optical Motion Capture Data* [16]. Se trabaja con sistemas Vicon [2] y PhaseSpace [7]. Se efectúa el seguimiento utilizando una distancia no euclídea y se ajusta con un modelo de esqueleto. Si bien el artículo no contiene explicaciones formales en su desarrollo, utiliza múltiples referencias a otros artículos.
  - *Optical Motion Capture System with Pan-Tilt Camera Tracking and Realtime Data Processing* [10]. Se trabaja con un modelo basado en marcadores, que contiene 34 grados de libertad. Los marcadores son ubicados formando poliedros en zonas de interés sobre el cuerpo, con el fin de utilizar algoritmos de búsqueda de poliedros para el etiquetado de marcadores. Utilizan múltiples servidores para efectuar el análisis en tiempo real. La posición de las cámaras no es fija a lo largo de la captura.
  - *Resolving Motion Correspondence for Densely Moving Points* [17]. Revisión de varios métodos de seguimiento.

- Estimación de pose
  - *Skeleton-Based Motion Capture for Robust Reconstruction of Human Motion* [14]. Descripto anteriormente en seguimiento.
  - *Modelling and Tracking Articulated Motion from Multiple Camera Views* [15]. Descripto anteriormente en seguimiento.
  - *Skeletal Parameter Estimation from Optical Motion Capture Data* [16]. Descripto anteriormente en seguimiento.
  - *Optical Motion Capture System with Pan-Tilt Camera Tracking and Realtime Data Processing* [10]. Descripto anteriormente en seguimiento.
  - *What can two images tell us about a third one?* [18]. Trabajo esencial sobre reconstrucción utilizando múltiples vistas. Gran desarrollo teórico, que busca robustecer algoritmos de reconstrucción.
- Sistemas completos
  - *Detección, rastreo y reconstrucción tridimensional de marcadores pasivos para análisis de movimiento humano* [19]. Trabajo de captura de movimiento, basado en el sistema de Lorna Herda [14], se utiliza hardware costoso.
  - *Marker Detection and trajectory generation algorithms for a multicamera based gait analysis system* [20]. Describe un sistema completo de captura óptico de movimiento basado en marcadores. Se gestiona el solapamiento de marcadores sobre una cámara en el proceso de segmentación. Las descripciones generales del resto del sistema son claras, si bien algunos detalles específicos de implementación no son mencionados. No utiliza información del esqueleto. El sistema necesita interacción con el usuario al final del proceso para efectuar correcciones.
  - *Análisis de video para estimación del movimiento humano* [21]. Describe globalmente distintas etapas que componen un sistema de captura de movimiento. Para introducirse en el tema es un buen resumen.

Se puede ver, que en varios casos un mismo artículo contiene información de valor sobre varias etapas del sistema. Además se encontraron algunos documentos con diseños de sistemas completos.

A partir de esta etapa se definen ciertos aspectos de vital importancia para el proyecto, en particular, se selecciona para el bloque de detección de marcadores el método de segmentación por umbral dinámico de Otsu [13], dado que presenta el mejor compromiso entre simplicidad y eficacia.

En los bloques de seguimiento y estimación de pose se decide basar el desarrollo en las ideas del sistema propuesto por Lorna Herda en sus tesis de doctorado [14]. Dicha elección se debe a que el estudio de Herda contiene las ideas principales para implementar un sistema de captura de movimiento de manera compacta y general,

posee una gran cantidad de menciones siendo un referente en el tema y la documentación disponible es amplia (dos papers, y tesis completa). Por otro lado, dicho trabajo se encuentra bajo las mismas hipótesis de uso que el estudio preliminar realizado por el grupo de proyecto (uso en fisioterapia, animación, biomecánica, entrenamiento en alto rendimiento), y es un punto de referencia habitualmente mencionado. Sin embargo, si bien en todas sus menciones la metodología de tratamiento de datos es la misma, tanto los bloques de adquisición como de detección van variando en equipamiento y métodos en los distintos proyectos.

Capítulo 3

Base de datos

3.1. Introducción

Con el fin de relevar distintas clases de actividades humanas y en particular la marcha, se pretende trabajar con una base de datos basada en marcadores. La misma debe contener un cierto número de sujetos realizando una variedad de movimientos predefinidos, donde para cada sujeto a estudiar, se tengan múltiples secuencias de videos 2D de movimiento obtenidas a partir de cámaras situadas en un entorno 3D cerrado, previamente acondicionado. Con el fin de medir el desempeño de los algoritmos generados, dicha base debe contar con el correspondiente ground truth 2D y 3D de los datos de movimiento disponibles. En nuestro caso dado que el sistema está basado en marcadores, la información relevante del ground truth consiste en las coordenadas espaciales a lo largo del tiempo para cada marcador de interés, así como la información de calibración de las cámaras utilizadas para efectuar las capturas.

La base de datos permitirá implementar, testear y comparar los distintos tipos de algoritmos desarrollados en el sistema. Por lo tanto es de interés primario relevar las bases de datos existentes o en su defecto realizar una.

En primera instancia se efectúa una búsqueda con el objetivo de encontrar secuencias de video de personas caminando, donde dada las características de este proyecto, las mismas deben poseer marcadores claramente distinguibles. Las secuencias tienen que ser tomadas con varias cámaras ubicadas alrededor del sujeto y para obtener información 3D como mínimo la cantidad de cámaras deben ser dos. Además es importante contar con el ground truth indicando la posición espacial de dichos marcadores para testear y comparar los distintos algoritmos de seguimiento aplicados sobre dichas secuencias. La información obtenida en el relevamiento de las bases de datos se ordena y describe en las Sección 3.2.

El análisis de la información recabada permite reunir un conjunto de conceptos de vital importancia a la hora de armar un laboratorio para la captura óptica basada en marcadores. Es por ello que en la Sección 3.3 se enumeran algunas variables de laboratorio y su impacto en el sistema de procesamiento posterior a la captura. Un ambiente controlado con cámaras ajustadas convenientemente provee un escenario

ideal para obtener información óptica de los marcadores. El sujeto es condicionado a moverse sobre un espacio de captura determinado con una vestimenta particular; la elección adecuada de estos parámetros junto con las condiciones de iluminación y el tipo de fondo facilitan la recolección de información, el posterior análisis y sobre todo, aumentan la performance del sistema completo.

Debido a que en la búsqueda no se encontraron bases de datos que cumplan los requerimientos mínimos del sistema a desarrollar, se decide implementar un prototipo de base de datos, que contenga un número reducido de secuencias útiles. Utilizando para ello la suite de animación 3D *Blender*, se genera un laboratorio de captura virtual y un modelo con marcadores animado sobre el cual se cargan secuencias de movimiento. Estas últimas se obtienen de sistemas de captura reales de alta performance al relevar las bases de datos. Los detalles de la implementación junto al análisis de esta base se pueden encontrar en la Sección 3.4. Se propone una estructura de datos para almacenar la información relevante del ground truth y se cuenta con código de soporte que facilita la generación de nuevas secuencias así como la gestión de la información en la estructura de datos, evaluar resultados y realizar el posterior análisis de performance.

Por último se hace un resumen con las conclusiones generadas en todo el capítulo

3.2. Revisión de base de datos

En esta sección se muestra resultados del relevamiento de bases de datos útiles para el análisis de la marcha. Inicialmente el objetivo de este proyecto era obtener un sistema funcional para el caso de la marcha humana, aunque luego de la implementación del mismo se generalizaron sus usos, las bases de datos con este tipo de movimiento fueron el eje central de la búsqueda.

Se encontraron principalmente tres páginas web que a manera de compendio agrupan bases de datos útiles en varias de las ramas de la visión por computadora.

CVonline [22]
Medianamente completa, no solo contiene enlaces a varias bases de datos sino reúne bibliografía e implementaciones útiles.

Computer Vision Papers [23]
Solo reúne enlaces de bases de datos.

Yet Another Computer Vision Index To Datasets (YACVID) [24]
Una característica importante es que la página contiene direcciones a bases de datos relativamente nuevas y resalta aquellas que son usadas con mayor frecuencia.

Dentro de estas páginas se encuentra una gran variedad de bases de datos que tratan el caso particular de la marcha, en la Tabla 3.1 se muestran algunas de las bases relevadas y sus características representativas, que permiten hacer una idea del panorama global encontrado a la hora de recopilar información en la web.

Tabla 3.1: Comparación de algunas bases de datos disponibles y empleadas por la comunidad.

Base de datos	Cantidad de sujetos	Nro. de secuencias	Entorno	Número de cámaras	Calibración disponible
M.C.L.[a]	3	299	Interior	1	No
C.M.U [b]	>100	2605	Interior	1	No
G.T [c]	20	∼100	Interior y exterior	3	Si
U.S. [d]	>100	∼ [e]	Interior	12	∼
Human ID	122	1870	Exterior	2	∼
HumanEva	4+2	56	Interior	4/7	Si
INRIA [f]	>11	>40	Interior	≥1	Si
CMU Mobo [g]	25	100	Caminadora	6	∼
CASIA	385	∼	Interior y exterior	>4	∼
MHAD [h]	12	660	interior	12	Si

Base de datos	Movimientos disponibles	Apariencia	Ground Truth	Tipo de acceso
M.C.L.	Varios	Traje MoCap	3D-Vicon [i]	Libre
C.M.U	Varios	Traje MoCap	3D-Vicon [j]	
G.T	Marcha fuera de régimen	Traje MoCap y Natural	3D en formato Maya	Libre
U.S.	Marcha	Natural	∼	Libre
Human ID	Marcha	Natural	∼	Bajo solicitud
HumanEva	Varios	Natural	3D - Vicon	Bajo solicitud
INRIA	Varios	Traje MoCap y Natural	2D - MoCap etiquetado manual	Libre
CMU Mobo	5 tipos de marcha	∼	Etiquetado Manual [k]	Bajo solicitud
CASIA	Varias velocidades de marcha	Natural	No	Bajo solicitud
MHAD	Varios	Traje MoCap	3D - Impulse [l]	Libre

[a] Motion Capture Lab
[b] Carnegie Mellon University Motion Capture Database
[c] Georgia Tech
[d] University of Southampton Database
[e] El símbolo ∼ indica que no se cuenta con información al respecto.
[f] INRIA Perception, Multicam Dataset.
[g] CMU Motion of Body.
[h] Berkeley Multimodal Human Action Database.
[i] Esta notación indica que la captura considerada ground truth, se hizo con un sistema de captura de movimiento (MoCap) de Vicon-Peak, http://www.vicon.com/
[j] Secuencias bvh de CMU, http://sites.google.com/a/cgspeed.com/cgspeed/motion-capture.
[k] Disponible en http://www.cs.cmu.edu/~zhangjy/#Data.
[l] Esta notación indica que la captura considerada ground truth, se hizo con un sistema de captura de movimiento (MoCap) Impulse (PhaseSpace Inc., San Leandro, CA), http://www.phasespace.com/

A continuación algunos comentarios que vale la pena resaltar sobre las bases

anteriores.

Motion Capture Lab. [25] Como describe el nombre, este laboratorio se centra en obtener capturas de movimiento tridimensionales a través del sistema Vicon. Cuentan con un sistema Vicon 8i de 14 cámaras, dos videocámaras digitales Sony DSR-PD150 que llegan a 30 fps[1], junto a un sistema de sincronización que permite integrar todas estas cámaras en el laboratorio. El video se utiliza únicamente como ayuda en los datos de captura para corregir y ver los marcadores, por lo que no cuenta con secuencias de video adecuadas. Cabe destacar que maneja múltiples formatos MoCap, inclusive el BVH[2].

Carnegie Mellon University Motion Capture Database. [26] Este laboratorio también se centra en obtener capturas de movimiento a través de un sistema Vicon, y no está implementado para evaluaciones ópticas de las capturas. Solo maneja videos monoculares de baja resolución, por lo que no cuenta con secuencias de video adecuadas para este proyecto. Sin embargo maneja múltiples formatos MoCap y posee herramientas para conversión a otros formatos, incluido el bvh. Posee descripción detallada de la ubicación de los marcadores, así como de la configuración del laboratorio. El sujeto a relevar se encuentra vestido con ropas finas y ajustadas, sobre la cual se colocan los marcadores, por lo tanto se puede despreciar las fluctuaciones de posición de marcadores debidas a la ropa, que si exhiben otros ambientes menos controlados Esta base de datos es bastante utilizada en el ámbito de la animación por computadora y es por lejos la que dispone de mayor cantidad de capturas de movimiento de acceso público de las bases relevadas en esta sección.

Georgia Tech. [27] Se encuentra desarrollando maneras de identificar a los seres humanos a distancia a través del reconocimiento de la marcha. También llevan a cabo trabajos relacionados en la localización y seguimiento de rostros, la detección de oclusiones y actividades específicas de sustracción de fondo. Lamentablemente las cámaras están todas sobre un costado del caminante, la resolución es baja y no todos los videos poseen marcadores. No se cuidan las condiciones de laboratorio para trabajar de manera óptica según las necesidades de este proyecto.

University of Southampton Database. [28] Interesados en el reconocimiento de personas a través del seguimiento de la marcha, relevantes sobre todo por ser pioneros en el estudio biomecánico de la marcha. Poseen dos bases de datos, HiD gait database (100 sujetos) y Biometric tunnel. El servidor donde se alojaba la base de datos está fuera de línea desde el 2004, el encargado de la página es Mark Nixon (msn@ecs.soton.ac.uk). Aparentemente se continúa actualmente el proyecto en http://www.cspc.ecs.soton.ac.uk/gait[3]. Si bien estas bases de datos no están basadas en marcadores, cabe resaltar que el fondo del espacio de captura del túnel biométrico contiene patrones asimétricos con colores saturados que facilitan no solo la extracción de fondo sino también la automatización del proceso de calibración.

Human ID Gait Challenge Dataset. [29] Contiene 1.2 Tera bytes de información. Utilizada para el reconocimiento de la marcha, no posee marcadores.

[1] http://www.bhphotovideo.com/c/product/197878-REG/Sony_DSRPD150_DSR_PD150_Professional_1_3_DVCAM.html. Accedido 29-11-14

[2] En la sección 3.4.2 se describe en detalle el formato BVH.

[3] Accedido 30-11-14

Al igual que en la base de Southampton contiene en la secuencia imágenes de dameros convenientemente dispuestos, útiles para calibrar las cámaras. Las sucesivas secuencias tomadas para un mismo sujeto modifican distintos tipos de factores, como la calidad del terreno a recorrer, si el sujeto lleva o no un maletín y el tipo de calzado utilizado. Tienen disponible junto a todas las secuencias de la base de datos las siluetas de los sujetos, obtenidas con un algoritmo base también propuesto y disponible. Lamentablemente el enlace que indica el protocolo seguido para obtener secuencias, junto con la especificación del equipamiento, está fuera de línea.

HumanEva. [30] Con 1.6 Gigabytes de información, el trabajo de este laboratorio es muy completo, incluye métricas de evaluación y un análisis importante de las necesidades actuales a la hora de generar una base de datos para relevar actividades humanas. El único inconveniente para cumplir los requisitos de este proyecto es que no está pensado para el seguimiento óptico de marcadores, por lo que los marcadores sobre los sujetos de estudio utilizados para recabar datos Mocap son escasos y demasiado pequeños, las condiciones de luminosidad y color de fondo no son las adecuadas para este propósito, incluso utilizan máscaras sobre las imágenes ópticas para cubrir los marcadores, estos últimos son utilizados solo para recopilar el ground truth a través de un sistema Vicon. Dado que su motivación es obtener una imagen "natural" del sujeto que contenga la complejidad generada por el movimiento de la ropa, el ground truth que presentan no es tan preciso como los obtenidos por métodos más tradicionales. Contiene código para efectuar sustracción de fondo y la implementación de un algoritmo base, con un filtrado de partículas utilizado para el seguimiento de la pose del sujeto. Para acceder a los datos se debe gestionar un permiso en la página de HumanEva.

INRIA Perception, Multicam Dataset. [31] Efectúan las capturas con múltiples cámaras y también ofrecen la secuencia de malla de los sujetos reconstruidos a partir de las imágenes. Ofrecen un software que permite navegar en 4D, es decir, el espacio y el tiempo, sobre los modelos disponibles en su base de datos. Por lo que estos modelos se pueden ver desde cualquier ángulo de visión e inspeccionar congelándolos en cualquier instante de tiempo. Lamentablemente su captura no está basada en marcadores.

CMU MoBo Dataset. [32] Estudia la marcha humana, enfocada en la identificación biométrica de humanos a partir de sus características individuales. Los sujetos bajo estudio efectúan 4 tipos de caminata sobre una caminadora: marcha lenta, marcha rápida, marcha sobre plano inclinado y marcha sosteniendo un balón. Se utilizan 6 cámaras de alta resolución ubicadas alrededor de la caminadora que adquieren imágenes a 30 cuadros/s. El acceso a la base de datos es restringido.

CASIA Gait Database. [33] Contiene 4 bases de datos sobre la marcha, la primera se efectúa en exteriores y posee una sola cámara; la segunda es en interiores y posee 11 cámaras sobre un lado del sujeto; la tercera utiliza cámaras infrarrojas y cada sujeto efectúa tres tipo de marcha, lenta, rápida y con mochila; por último la cuarta combina la información de cámaras de captura con una plataforma que registra la presión de la planta del pie a medida que el sujeto desarrolla el movimiento. Además de los archivos de vídeo, ofrecen las siluetas humanas a partir de sus secuencias de vídeo. Las capturas visuales no están basadas en marcadores.

Berkeley Multimodal Human Action Database (MHAD). [34] El objetivo de esta base de datos es reconocer el movimiento del cuerpo al realizar distintas actividades. Tiene la información necesaria para efectuar sustracción de fondo, devuelven información desde múltiples fuentes:

- sistema con 12 cámaras ópticas agrupadas en 4 conjuntos de cámaras estéreo
- dos sensores Kinect
- 6 acelerómetros sobre el sujeto
- 4 micrófonos a cada lado del espacio de captura

Todas estas fuentes se encuentran sincronizadas. Esta base de datos es bastante completa en cuanto a los tipos de captura que realiza, pero los marcadores utilizados son leds y únicamente se utilizan para recabar información para el ground truth, con lo cual no se cuida de obtener una buena información óptica de los mismos.

3.3. Características de Laboratorio

A continuación se enumeran algunas variables que es necesario tener en cuenta a la hora de diseñar un laboratorio adecuado para un sistema de captura óptica basado en marcadores. También se discute brevemente el posible impacto de estás variables en el tratamiento posterior de las secuencias.

3.3.1. Cámaras

Para la elección del tipo de cámaras es importante tener en cuenta los requerimientos de las secuencias a relevar. A continuación vamos a tratar dos variables básicas y generales que deben contemplarse, por un lado la cantidad de fotogramas y por otro la resolución de la cámara.

La cantidad de fotogramas por segundo de las secuencias de video condiciona la resolución temporal de los datos procesados a partir de dichas secuencias, así como también limita la velocidad de movimiento a realizar por el sujeto si se busca una captura aceptable. Basados en el relevamiento de las distintas bases de datos se puede afirmar que 30 cuadros por segundo es un valor normal para trabajar el caso de la marcha, pudiendo obtener en este caso información de la posición de los marcadores solo cada 1/30 segundos.
Hay que tener presente que si se efectúan movimientos de velocidades superiores a la marcha, manteniendo la cantidad de fotogramas anterior, aumenta la dificultad a la hora de vincular temporalmente los datos obtenidos, pudiendo bajar la performance del seguimiento de los marcadores.

Respecto a los tiempos de obturación, estos deben ser bastante cortos para no producir efectos de desplazamiento nocivos a la hora de reconocer los marcadores. Una regla habitualmente utilizada que sirve de orientación es que el tiempo mínimo de disparo que asegura que no salga movida una imagen, es la inversa de la distancia

focal. También se encuentran referencias que indican los tiempos de obturación recomendados según la actividad a relevar[4].

- 1/4000 s: se utiliza para tomar fotografías nítidas de sujetos en rápido movimiento, como los atletas o vehículos, en condiciones de buena iluminación.
- 1/2000 s y 1/1000 s: útil para fotografías nítidas de sujetos en movimiento moderadamente rápido, bajo condiciones normales de iluminación.
- 1/500 s y 1/250 s : para tomar fotografías nítidas de personas en movimiento en situaciones cotidianas.

La resolución del sensor también juega un rol fundamental a la hora de cumplir requerimientos de resolución espacial en los datos de salida. En la Figura 3.1 se hace un bosquejo que permite estimar la resolución espacial a medida que el sujeto se aleja de la cámara.

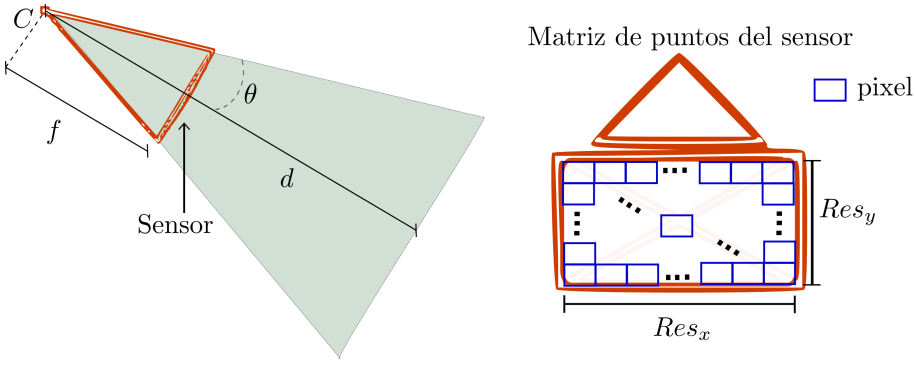


Figura 3.1: Estimación del la resolución espacial.

Sean x_s e y_s magnitudes sobre el sensor de la cámara, horizontal y vertical respectivamente, y sea X e Y las magnitudes a una distancia d del centro C de la cámara que producen respectivamente las proyecciones de magnitud x e y sobre el sensor. Se cumple la siguiente relación,

$$X = \frac{d}{f} x_s \ , \qquad Y = \frac{d}{f} y_s$$

Si el sensor[5] tiene un largo S_x y ancho S_y y resolución de $R_x \times R_y$ píxeles, resulta que el largo de un píxel es $l_x = S_x/R_x$ y el ancho $l_y = S_y/R_y$.

A modo de ejemplo, se considera una cámara con las siguientes características $S_x = 0{,}032\ m$ y $S_y = 0{,}010\ m$, resolución de 1600×600 píxeles y una distancia focal de 32 mm. Por lo tanto el error de un píxel se mapea sobre el sensor como $x = 0{,}032/1600\ m = 0{,}020\ mm$ a lo largo, e $y = 0{,}010/600\ m = 0{,}016\ mm$ a lo ancho. La Tabla 3.2 muestra como errores de 1, 2 y 3 píxeles en el sensor se propagan sobre el espacio a una distancia d del centro de la cámara.

[4] http://es.wikipedia.org/wiki/Velocidad_de_obturación
[5] También referido como retina de la cámara.

número de píxeles	**d (metros)**				
	0,5	**1,0**	**3,0**	**5,0**	**10,0**
1	0,03	0,06	0,19	0,31	0,63
2	0,06	0,12	0,38	0,62	1,26
3	0,09	0,18	0,57	0,93	1.89

Tabla 3.2: Resolución espacial en centímetros como función de la distancia al centro de la cámara

3.3.2. Marcadores

A la hora de seleccionar el equipamiento se debe prestar especial atención y coordinar el tipo de marcador, la vestimenta del sujeto, el fondo del espacio de captura y el tipo de iluminación. Pues las características de estos elementos en conjunto pueden cambiar sensiblemente la performance del sistema que procesa computacionalmente la captura, sobre todo en sus primeras etapas tales como la detección de marcadores y la calibración. Si se quiere facilitar la tarea de detección automática de marcadores, estos deben ser fácilmente distinguibles del resto del entorno, y visibles la mayor cantidad de tiempo para las cámaras. Para ello se debe tener marcadores que generen un alto contraste con el resto de los elementos dentro de la captura de video. Su tamaño y forma también son importantes, por lo general se trabaja con marcadores esféricos, y un tamaño que dado cierto espacio de trabajo no condicione fuertemente la resolución necesaria de las cámaras. Una medida que se puede considerar aceptable con cámaras convencionales y movimientos a menos de 12 metros del centro de las cámaras, son marcadores de 3 *cm* de diámetro. Si el fondo del espacio de captura es negro, opaco, al igual que la vestimenta del sujeto a capturar, los marcadores pueden ser blancos, no pulidos de manera que su reflejo sea difuso, para no generar variación de tono sobre los mismos. Su disposición sobre el sujeto responde a los intereses de lo que se quiera observar, pero se debe tener presente que al ser una captura óptica, para que un marcador sea útil, debe estar visibles buena parte del tiempo en la secuencia.

3.3.3. Iluminación

La iluminación debería ser preferiblemente uniforme, para no generar sombras que modifiquen los tonos del espacio de captura. La luz natural es una buena alternativa, en espacios cerrados lo que habitualmente se utilizan son pantallas delante de los focos lumínicos para hacerlos más difusos, reduciendo así el riesgo de generar imágenes especulares en los objetos. La disposición de los focos depende de la forma del espacio de captura, pero por lo general rodean al mismo, cuidando que estos no sean capturados directamente por las cámaras, y generen falsos positivos en la detección de marcadores. Si bien esto último puede ser solucionado parcialmente en la etapa de post-procesamiento efectuando convenientemente sustracción de fondo, es recomendable tratarlo en las primeras etapas, a nivel del sistema de captura.

3.3.4. Vestimenta

Es preferible que la vestimenta del sujeto a relevar, consista en ropas finas y ajustadas para despreciar fluctuaciones de la posición de marcadores debido al movimiento de la ropa. Como se menciona anteriormente al igual que el fondo, la elección del color y material debe contrastar con los marcadores.

3.3.5. Espacio de captura

Ya se han mencionado algunas características cualitativas relacionadas con el espacio de captura, como ser que el fondo debe contrastar con los marcadores. Un caso interesante a tener en cuenta es el del túnel biométrico de la Universidad de Southampton mencionado en la Sección 3.2, donde para lograr dicho contraste utilizan patrones asimétricos con colores saturados para el fondo del espacio de captura, esto les permite agrupar dos etapas, logrando luego de un tratamiento sobre la secuencia, extraer el fondo del resto de la información y efectuar la calibración de las cámaras. De todas maneras si bien se mantienen relacionadas, las etapas mencionadas se manejan comúnmente por separado. Por lo que si se utilizan marcadores blancos, un fondo oscuro idealmente negro, y opaco es suficiente.

En cuanto a las dimensiones del espacio de captura, el mismo depende principalmente del tipo de movimiento a relevar y las características de las cámaras utilizadas. A continuación se analiza posibles configuraciones para dos caso, la marcha rectilínea y marcha libre, en el último caso se permite al sujeto movilizarse sobre un área circular.

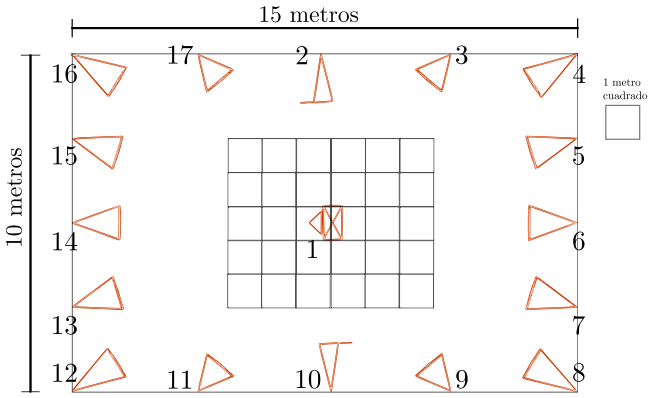
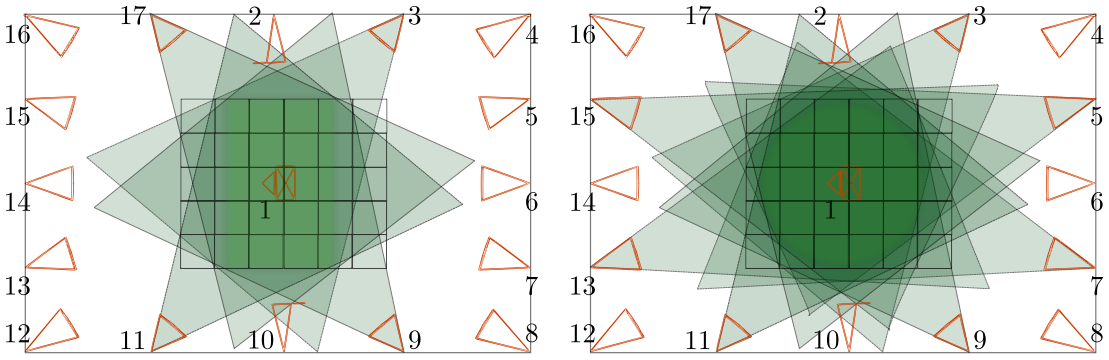


Figura 3.2: Toma superior del Laboratorio.
Todas las cámaras se encuentran a $1\ m$ del suelo.

En el caso de la marcha rectilínea puede verse que en principio con tan solo cuatro cámaras dispuestas como en la Figura 3.3a se genera un espacio de captura útil de aproximadamente $3\ m \times 5\ m$ en forma de pasarela. Si el sujeto restringe su movimiento sobre esta zona, dado que la misma se ubica dentro de la intersección de los campos visuales, será visto en su totalidad por todas las cámaras de manera simultánea. En el caso de marcha libre se disponen 8 cámaras intentando que el



(a) Espacio de captura para marcha rectilínea. (b) Espacio de captura para marcha libre.

Figura 3.3: Posibles espacios de captura.

espacio de captura no tenga direcciones privilegiadas, con lo cual se genera una circunferencia, que en este caso es de aproximadamente 5 m de diámetro.

Siguiendo el proceso discutido anteriormente en la Sección 3.3.1, se puede estimar la resolución esperada en los datos obtenidos a partir de las secuencias de video, si se cuenta en principio con información de la resolución de las cámaras y su distancia focal. Consideremos nuevamente cámaras con resolución de 1600×600 píxeles y distancia focal de $f = 32\ mm$, la mayor distancia sobre la cual una cámara debe tener cobertura en la pasarela de la Figura 3.3a es de 8 m, mientras que en la Figura 3.3b, caso de la marcha libre, es de 9 m con lo cual un error de un píxel en una cámara produce una incertidumbre de aproximadamente poco más de medio centímetro en ambos casos a la hora de efectuar la reconstrucción con las secuencias de video obtenidas. Debe tenerse presente que estos son casos particulares, pero que permiten hacer una idea de las consideraciones a tener en cuenta para cumplir algunos de los requerimientos del sistema. Aumentar el número de cámaras permite una mayor cobertura de volumen, lo que significa una mayor visibilidad del conjunto de marcadores. A su vez una mejor visibilidad del conjunto de marcadores está correlacionado con disminuir la posibilidad de que información importante se pierda en las etapas de análisis de datos posteriores.

3.3.6. Sincronización

Esto resulta esencial en las emisiones con conexiones en directo con varias cámaras, para garantizar que las imágenes de vídeo se mantienen estables incluso cuando se cambia de una cámara a otra. Se debe contar con una fuente central que envíe una señal de sincronización a todas las videocámaras que participan en una grabación, de esta manera se indica a cada una de ellas cuando deberán comenzar a grabar. Esta señal de sincronización puede consistir en eventos sonoros o visuales que permitan sincronizar en etapas de post-procesamiento, pero lo más recomendable si se dispone de equipamiento adecuado es directamente enviar una señal a las cámaras vía hardware. Si no se contara con una entrada de sincronización sería prácticamente imposible realizar una grabación con varias cámaras

simultáneamente.

3.4. Datos experimentales

3.4.1. Motivación

No se encontraron base de datos acorde a las necesidades de este proyecto. Como se menciona en la Sección 3.2 varios son los motivos, pero básicamente se debe a que ninguna se diseñó para el análisis óptico con marcadores y no ofrecen secuencias de video utilizables para estos fines. Por lo tanto se plantea la necesidad de construir una base de datos adecuada que permita desarrollar este sistema. Esto último no estaba contemplado inicialmente en los tiempos del proyecto, sin embargo dada la importancia que posee para el desarrollo del sistema se re-formula el cronograma y se ajustan los objetivos.

Una característica importante en muchas de las bases relevadas es la manera con la que habitualmente se obtiene el ground truth a partir de sistemas de captura de movimiento (MoCap), tales como Vicon [2] o Impulse [2], estos sistemas devuelven información muy precisa sobre la ubicación 3D de marcadores ubicados sobre el sujeto a estudiar. Este tipo de base de datos con capturas de movimiento, se utilizan ampliamente en el ámbito de la animación por computadora, pues se carga la información de la captura al modelo 3D de interés y este efectúa la acción esperada con un alto nivel de realismo.

Dada la complejidad material y logística de implementar una base de datos, los tiempos necesarios para desarrollar las herramientas de análisis del proyecto y su necesidad de secuencias útiles, junto con el potencial de la información obtenida en el relevamiento de dicha base y sus usos en otros ámbitos, se decide generar una base de datos sintética con el material MoCap utilizado como ground truth. Trabajando en esta dirección se generan secuencias en un ambiente totalmente controlado, donde se conoce el ground truth, simplificando el testeo de algoritmos sobre diferentes etapas del procesamiento y permitiendo incrementar gradualmente las dificultades en cada ensayo, por ejemplo efectuando variaciones en el número o posición de las cámaras. Estas características permiten a su vez generar *benchmark's* de algoritmos, de gran importancia para futuros desarrollos del sistema.

Continuando en esta dirección al día de hoy, se cuenta con herramientas de código libre para generar una base de datos sintética que sea robusta, completa y extensible. Particularmente se trabajo con la suite de animación 3D *Blender* [6], generando un laboratorio en dicho entorno virtual. Para obtener las secuencias de video, se asocia a un modelo virtual 3D la información de movimiento de interés obtenida a partir de sistemas MoCap, para luego generar secuencias animadas, las cuales por último, se renderizan sobre un cierto número de cámaras para generar secuencias de video conteniendo múltiples vistas del modelo original. Estas secuencias son las que se utilizaron para el desarrollo del sistema y permitieron generar una base de datos sintética.

[6] Blender Foundation http://www.blender.org/

3.4.2. Formatos Mocap

La captura de movimiento (Mocap), es una técnica de grabación del movimiento bastante popular en el cual se generan animaciones humanas o animales en ámbitos virtuales a partir de modelos reales. Este tipo de técnicas han sido utilizadas desde hace más de una década en el ámbito del entretenimiento, tales como el cine o los videojuegos e incluso en el ámbito deportivo y médico.

Importante en la mayoría de las bases de datos relevadas, incluso aquellas de análisis óptico, ya que los sistemas comerciales que efectúan este tipo de capturas devuelven datos con un alto nivel de precisión, por lo que se utilizan como ground truth y se comparan con datos obtenidos a partir de otros sistemas. La primera desventaja de todas estas prestaciones es su alto costo, bastante privativo en muchos ámbitos.

En la actualidad el éxito de las capturas de movimiento ha dado lugar a una serie de casos de producción que pueden registrar y proporcionar datos de movimiento, sin embargo muchas de las empresas han desarrollado su propio formato de archivo. Esto significa que los formatos de archivo de los datos de captura de movimiento están lejos de unificarse en un estándar, sin embargo la naturaleza ASCII de muchos de los formatos, hace que sea razonablemente fácil decodificar y entender la información por simple inspección de los datos.

Tabla 3.3: Formatos de archivo Mocap.

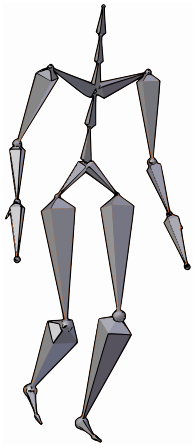
Formatos Mocap	**Compañía**
ASF & AMC	Acclaim [a]
BVA & BVH	BioVision [b]
BRD	LambSoft Magnetic Format [c]
C3D	Biomechanics, Animation and Gait Analysis [d]
CSM	3D Studio Max, Character Studio[e]
GTR, HTR & TRC	Motion Analysis [f]

[a] http://www.darwin3d.com/gamedev/acclaim.zip. Accedido 30-11-14
[b] http://www.biovision.com/bvh.html. Accedido 30-11-14
[c] http://www.dcs.shef.ac.uk/~mikem/fileformats/brd.html. Accedido 30-11-14
[d] http://www.c3d.org/c3d_format.htm. Accedido 30-11-14
[e] http://www.dcs.shef.ac.uk/~mikem/fileformats/csm.html. Accedido 30-11-14
[f] http://research.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/MoCapTOC.html. Accedido 30-11-14

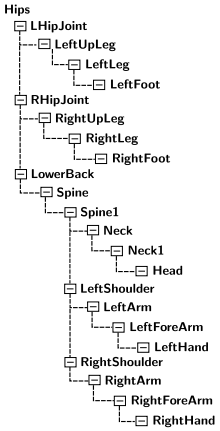
La aplicación que se realice condiciona cual de los formatos de la Tabla 3.3 es más conveniente. La mayoría de los formatos mostrados son soportados por *Blender*, sin embargo dado que en este caso el interés es la manipulación y fácil interpretación de la información, el formato elegido para manipular las capturas de movimientos es el BVH, por ser posiblemente el más compacto, conciso y ampliamente utilizado.

De todas maneras se cuenta con funciones en *Matlab* para llevar la información de movimiento contenida en estos archivos a varios de los formatos descriptos en la Tabla 3.3, así como también directamente a una matriz con las coordenadas 3D de cada marcador a lo largo del tiempo.

BVH (BioVision Hierarchical data). El formato BVH proviene del formato BVA de BioVision con la notable adición de una estructura de datos jerárquica que representa los huesos[7] del esqueleto[8]. Se compone de dos partes, la primera sección detalla la jerarquía y la pose inicial del esqueleto y la segunda sección el movimiento del mismo, describiendo la información temporal del esqueleto cuadro a cuadro.



(a) Esqueleto



(b) Estructura jerárquica

```
HIERARCHY
ROOT Hips
{
    OFFSET 0.000000 0.000000 0.000000
    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
    JOINT LHipJoint
    {
        OFFSET 0.000000 0.000000 0.000000
        CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
        JOINT LeftUpLeg
        {
            OFFSET 0.102356 -0.052561 -0.097592
            CHANNELS 3 Zrotation Yrotation Xrotation
            JOINT LeftLeg
            {
                OFFSET 0.139398 -0.000000 -0.382993
                CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
                JOINT LeftFoot
                {
                    OFFSET 0.007817 -0.118188 -0.021479
                    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotatior
                    End Site
                    {
                        OFFSET 0.000000 -0.060762 0.000000
                    }
                }
            }
        }
    }
....
....
....
}
```

(c) Sección de jerarquía

```
HIERARCHY
ROOT Hips
{
....
....
....
}
MOTION
Frames: 62
Frame Time: 0.041667
0.000000 0.000000 0.780000 0.020256 -0.020256 0.020249
0.020258 -0.020260 0.020251 0.021696 20.999995 0.028031
0.020256 -0.020258 0.020249 0.147010 -0.000000 -0.403906
0.007817 -0.118188 -0.021479 0.020256 -0.020257 0.020249
0.020254 -0.020258 0.020249 0.021700 -21.000001 0.012480
0.020256 -0.020259 0.020249 -0.147497 -0.000000 -0.405244
-0.008281 -0.113074 -0.022750 0.020256 -0.020256 0.020250
0.020256 -0.020256 0.020249 0.020256 -0.020256 0.020249
0.020256 -0.020256 0.020249 0.000000 -0.000000 0.000000
0.020256 -0.020256 20.999994 0.001065 0.004776 0.095661
-0.000000 -0.000000 0.000000 0.020257 -0.020255 0.020249
0.298616 0.000000 0.000000 0.020256 -0.020256 0.020249
0.020256 -0.020255 0.020249 0.000000 -0.000000 -0.000000
0.020256 -0.020255 0.020249 0.000000 -0.000000 -0.000000
0.000000 -0.000000 0.000000 0.020256 -0.020257 0.020249
-0.294284 0.000000 -0.000000 0.020256 -0.020257 0.020249
0.020256 -0.020256 0.020249 0.000000 -0.000000 -0.000000
0.020256 -0.020257 0.020249 -0.000000 0.000000 -0.000000
....
....
....
```

(d) Sección de movimiento

Figura 3.4: Información contenida en archivos BVH.

[7]Entidad básica en la representación del esqueleto. Cada hueso representa el segmento más pequeña dentro del movimiento sujeto a cambios de traslación y rotación.

[8]Entidad global que representa el movimiento. Un esqueleto está formado por huesos.

La sección jerárquica del archivo empieza con las dos primeras líneas conteniendo las palabras HIERARCHY y ROOT respectivamente, seguidas del nombre del hueso que es la raíz de la estructura jerárquica y a continuación la estructura restante del esqueleto conteniendo la información del origen de cada hueso y el orden en que las variables del hueso se modifican en la segunda sección. La sección de movimiento empieza con la palabra clave MOTION, contiene la cantidad de cuadros en la secuencia, el tiempo entre cuadros y la información completa de la posición de cada hueso del esqueleto en cada cuadro de la secuencia. Por más detalles del formato consultar el artículo de M. Meredith y S.Maddock [35].

Si bien en principio cualquier fuente de archivos BVH con capturas de movimiento es válida, se decide trabajar en este proyecto con las fuentes BVH de la base de datos *MotionBuilder-friendly version* ofrecidas por *cgspeed* [36], que provienen de las capturas de Carnegie Mellon University Motion Capture Database [26]. Como se menciona anteriormente en la Sección 3.2, CMU dispone de un gran número de capturas de movimiento de acceso público, varias utilidades de software que permiten llevar a otros formatos y es utilizado ampliamente en el ámbito de la animación por computadora.

3.4.3. Laboratorio Virtual

En esta sección se describe de que manera se puede utilizar *Blender* y Python para generar secuencias de video para una base de datos sintética.

Blender es una suite de animación 3D gratuita y de código abierto, la misma cuenta con las herramientas necesarias para trabajar en todas las etapas de desarrollo de un proyecto de animación 3D. Por otro lado posee una interfaz de programación de aplicaciones muy flexible y potente que permite extender su funcionalidad a través de programas en lenguaje Python. Estas cualidades son muy útiles para generar una base de datos por lo que se decidió desarrollar la misma utilizando las herramienta de esta aplicación.

Se ha generado un espacio de trabajo controlado, conteniendo entre otras cosas un cierto número de cámaras, iluminación, color de fondo determinado y un modelo con marcadores sobre el cual cargar la captura de movimiento de interés. También se han implementado programas en Python para facilitar múltiples tareas:

- Ayudar en el ajuste de las capturas provenientes de archivos BVH al esqueleto del modelo virtual.
- Generar las secuencias de video a partir de archivos .blend con el modelo virtual desarrollando el movimiento de interés.
- Exportar la información de calibración de las cámaras en un archivo .m para usar la misma desde *Matlab*.

Disposición del laboratorio

El laboratorio generado posee 17 cámaras dispuestas en un ambiente rectangular de 10×15 m como se muestran en las Figuras 3.2 y 3.5. La configuración de

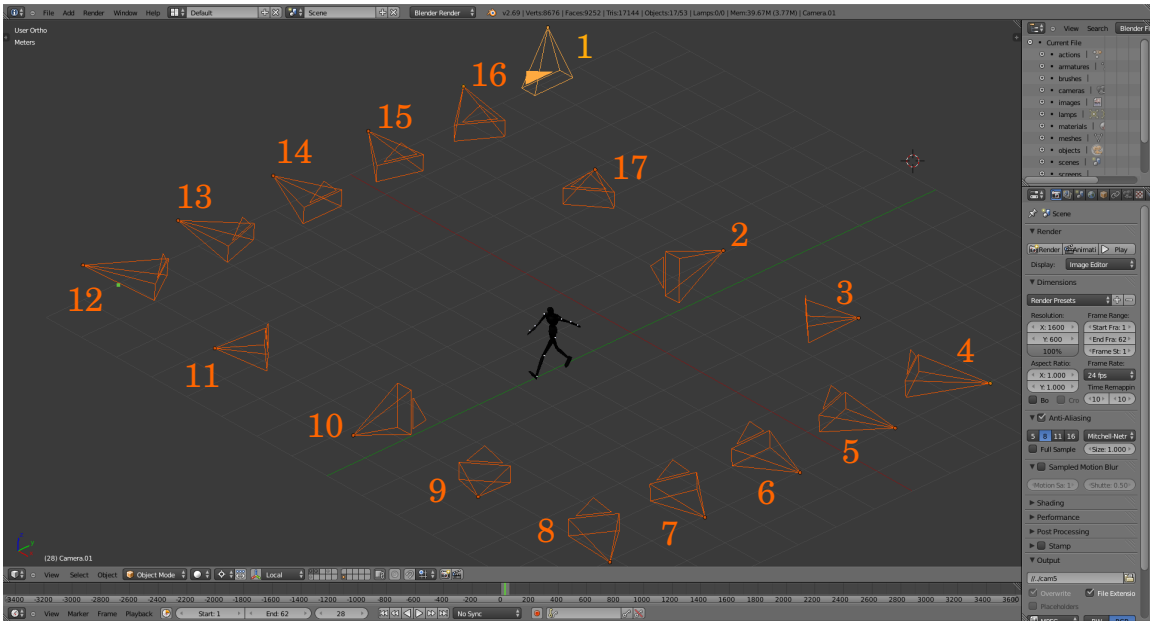


Figura 3.5: Entorno de trabajo en *Blender*.

las mismas permite probar diferentes combinaciones de captura a la hora de generar secuencias. Se diseñaron las cámaras para que sus parámetros fuesen similares a cámaras convencionales encontradas en el mercado y utilizadas en las bases de datos relevadas. Para iluminar el laboratorio se dispusieron 8 focos puntuales de luz omnidireccional sobres los límites del mismo, rodeando la escena y a un nivel de 3 metros de altura. De esta manera se garantiza que ninguno de los focos sea tomado directamente por las cámaras y los marcadores se iluminan correctamente.

Modelo virtual

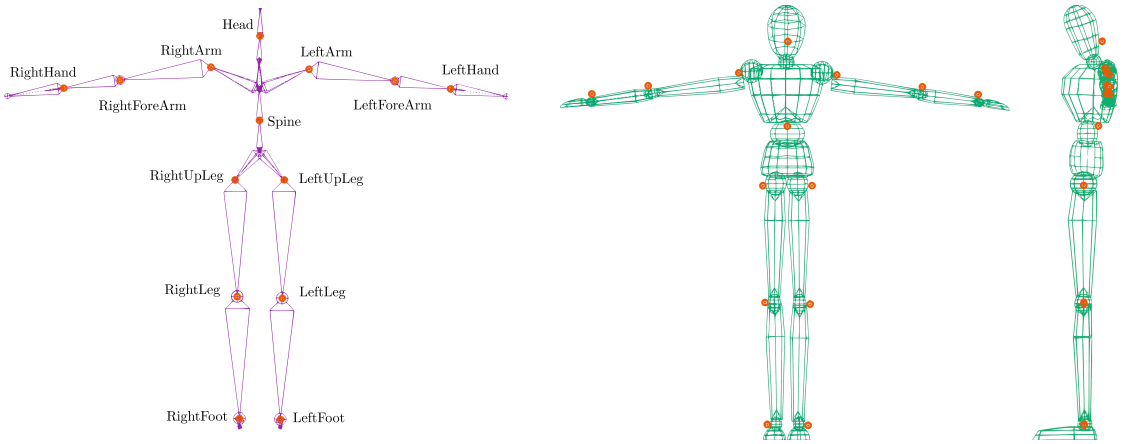
El modelo virtual utilizado se basa en un maniquí de madera convencional, la elección del mismo responde a la facilidad con la que se puede ajustar el mismo a una posición particular, siendo cada miembro fácilmente correlacionado con un hueso específico, sin perder las particularidades propia de un sujeto real. Cabe destacar que la relevancia del modelo en las capturas es simular las oclusiones de marcadores debida a los miembros del sujeto en una captura real.

Para la elección del material aplicado al modelo virtual, se toman en cuenta las consideraciones planteadas en la Sección 3.3, dado que los marcadores que se van a utilizar son de color difuso blanco, se decide aplicar sobre el modelo un material difuso negro con especularidad baja, de manera que no se generen brillos indeseados debido a la iluminación en el momento de la captura. De esta manera la configuración recomendada es utilizar un fondo también oscuro.

Esqueleto

En cuanto a el esqueleto con la información de movimiento, el mismo se obtuvo de la base de datos *MotionBuilder-friendly versión* (de aquí en más MotionBuilder) ofrecidas por *cgspeed* [36], donde se cuenta con las fuentes BVH que provienen de las capturas de movimiento de CMU (Sección 3.2). Si bien *cgspeed* ofrece otras bases más nuevas, donde la jerarquía del esqueleto en los archivos BVH a sido reducida con el fin de facilitar el ajuste a modelos virtuales, se ha trabajado con la base original antes mencionada. Aparentemente el procesamiento aplicado para producir las nuevas bases, introdujo ruido sobre algunos cuadros en las secuencias originales, esto se ha constatado al importar en *Blender* dichas secuencias.

Con el fin de normalizar las secuencias BVH provenientes de MotionBuilder se ha utilizado la herramienta de edición de archivos BVH *bvhacker* [37]. La misma permite centrar las secuencias sobre un mismo punto en el primer cuadro removiendo los offset globales. Una vez gestionado lo anterior se importa en *Blender* la secuencia BVH y se procede a generar los marcadores en las articulaciones del esqueleto, dado que se tiene la posición exacta del origen de cada hueso en el esqueleto y la articulación es la unión entre dos huesos consecutivos, se puede obtener la posición exacta de los marcadores a partir de la secuencia BVH. La generación de los marcadores y el enlazado del esqueleto al modelo se realiza a través de un código python, `acoplar_modelo.py`, realizado para tal fin. El mismo genera marcadores esféricos de material difuso blanco levemente poroso, con $2,8$ *cm* de diámetro y baja especularidad, centrados en las articulaciones definidas previamente en una lista, como se muestra en la Figura 3.6a y luego enlaza cada miembro del modelo a su correspondiente hueso. De está manera para finalizar la creación del modelo con marcadores solo resta que el usuario realice ajuste mínimos de tamaño y posición sobre el modelo. Es importante resaltar que los marcadores se ubican sobre puntos del esqueleto de los cuales se conoce la posición 3D en cada cuadro y que el modelo es necesario debido a que el esqueleto no se puede renderizar.



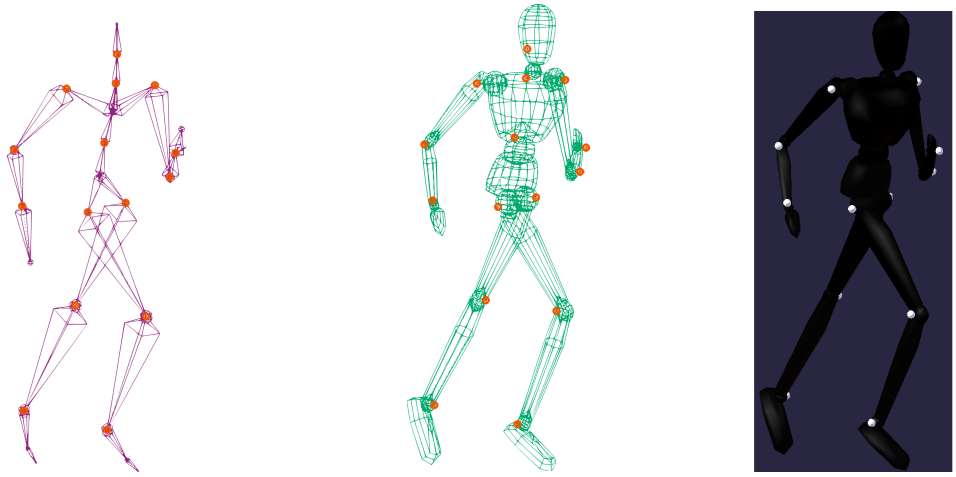
(a) Creación de marcadores (b) Ajuste de modelo

Figura 3.6: Generación de marcadores sobre el modelo virtual.

Una ves finalizado el proceso, Figura 3.6b, el modelo puede realizar el movimiento contenido en la secuencia BVH y solo resta renderizar.

Renderizado

Ya se dispone de la secuencia de movimiento en el entorno virtual de *Blender*, lo que resta es renderizar dicha secuencia sobre las cámaras del laboratorio. Para ello se implementó un código en python, `render.py`, que automatiza todo el proceso, dispone los videos obtenidos en cada render con nombres y ubicaciones predefinidas en la base de datos y genera un código *Matlab* con la información ground truth de cada cámara.



(a) Importación de secuencia BVH y generación de marcadores

(b) Modelo enlazado al esqueleto

(c) Imagen renderizada

Figura 3.7: Creación de secuencias en *Blender*.

Ground truth

Para contar con el ground truth final de la secuencia animada se debe exportar desde *Blender* la información del esqueleto en un BVH, cuidando que se mantenga la misma escala de tiempos que en el momento del renderizado. De esta manera se tienen sincronizados los videos y el ground truth 3D.

Mocap Tools

Existe una herramienta en *Blender* que permite manipular y copiar la información de movimiento entre dos esqueletos, denominada *Mocap Tools*. La misma no viene activada por defecto y se debe habilitar desde el menú de preferencias. Para la creación de secuencias utilizando esta herramienta, se debe tener un esqueleto ya enlazado al modelo virtual con marcadores, luego se importa la secuencia BVH de

interés que a su vez genera otro esqueleto. Utilizando las opciones de *Mocap Tools* se copia el movimiento del esqueleto BVH al esqueleto ya definido inicialmente, de esta manera se evitan los ajustes posteriores del modelo virtual y se tiene una secuencia lista para renderizar. El problema surge al momento de obtener el ground truth, pues la exportación del esqueleto que se utiliza para mover al modelo no es precisa en este caso, se detectaron fallas en este sentido por parte de *Blender*, solo se exportan los movimientos relativos de los huesos y no la traslación del centro de masa del esqueleto. El motivo fundamental es el mecanismo utilizado por *Mocap Tools* para enlazar los dos esqueletos, el BVH y el del modelo virtual. Dado que la información de traslación no se incluye en el esqueleto de destino sino que se condiciona a que este último se traslade según un hueso particular externo (stride bone) cuya información de movimiento no puede ser exportada en un BVH por *Blender*. Con lo cual, si bien al trabajar con esta herramienta se generan secuencias visualmente aceptables para un proceso de animación, no se obtiene el ground truth necesario para las etapas de performance y desarrollo del sistema, por lo que no se utiliza.

3.5. Estructura de la base de datos

El prototipo de base de datos, actualmente contiene cinco secuencias sintéticas descriptas en la Tabla 3.4, las mismas fueron generadas a partir de las secuencias BVH provenientes la base de datos *MotionBuilder-friendly version* ofrecidas por *cgspeed* [36], cuyo origen son las capturas de movimiento de CMU (Sección 3.2).

Tabla 3.4: Secuencias generadas para la base de datos sintética.

Acción	**Nro. de secuencia CMU**	**Longitud de secuencia (segundos)**	**Cuadros por segundo**	**Espacio de captura (metros)**
Marcha rectilínea	08_03	2.5	30	1,5× 5,5
Marcha rectilínea, zancada exagerada	08_07	2.6	24 y 48	1,5×5,5
Marcha rectilínea lenta, zancada ancha	08_11	3.8	30	1,5×5
Corriendo	09_07	1.2	24 y 48	1,5×5
Marcha libre	09_12	12.5	24	3×5

La estructura utilizada para los directorios de la base de datos se muestran en las figuras 3.8 y 3.9, donde las mismas sirven para soportar tanto datos sintéticos como datos reales. En dichas figuras dentro de cada tag se debe colocar la variable indicada.

Básicamente se tiene la información de cada secuencia de un sujeto separada en tres niveles: los datos de video, la información obtenida con los algoritmos al

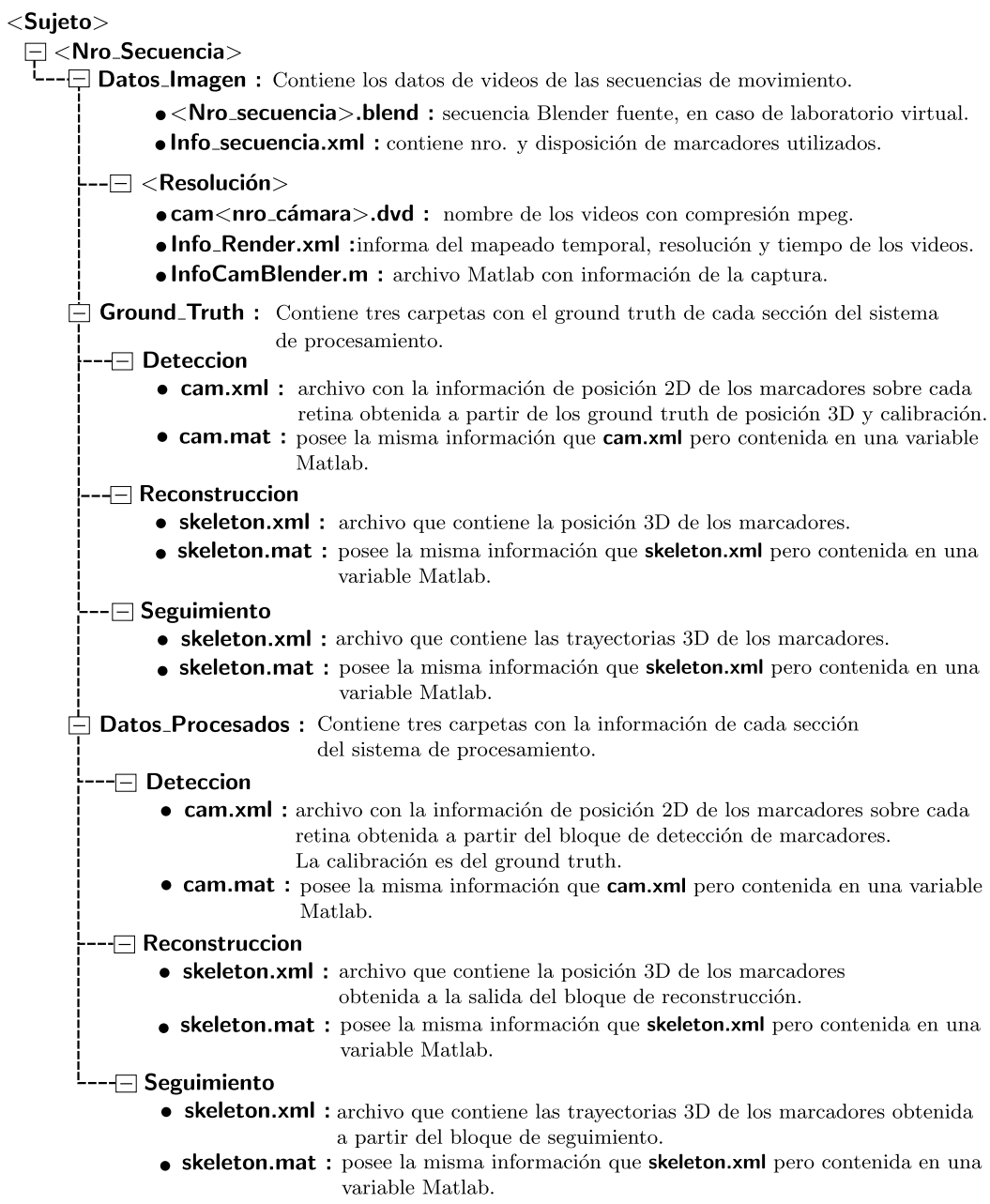


Figura 3.8: Estructura de directorios para almacenar información de capturas.

procesar los datos de video y por último, si se encuentra disponible, la información de ground truth del mismo.

En el caso de secuencias sintéticas, fuera del entorno virtual se debe contar con los videos de la secuencia y el ground truth tanto de la calibración como del movimiento del esqueleto. La generación y ubicación de los videos en la estructura se efectúa mediante el código `render.py`, luego la exportación del ground truth de la calibración a un archivo .m de *Matlab*, se efectúa a través de `obtener_InfoCam.py`.

Calibracion

- **Datos_imagen :** Contiene dos carpetas con imágenes disponibles para efectuar pruebas de calibración.
  - **Tablero**
    - **cam<Nro_cam>_<Nro_posición>.tiff :** archivo que posee la imagen de un tablero de ajedrez de $2\,m \times 2\,m$, con cierta posición particular en el espacio 3D numerada como <**Nro_posición**>, vista por la cámara <**Nro_cámara**>.
    - **tablero.blend :** archivo fuente con el modelo 3D del tablero y las cámaras utilizadas. Contiene código *Python* para posicionar el tablero automáticamente.
    - **CambioDeBase**
      - **cam<Nro_cámara>.tiff:** archivo que posee la imagen del calibrador del archivo **Calibrador.blend**, vista por la cámara <**Nro_cámara**>.
      - **Calibrador.blend:** archivo fuente con el calibrador 3D y las cámaras utilizadas, que permite ubicar el origen de coordenadas. Contiene código *Python* para renderizar automáticamente.
  - **Punto_Luminico**
    - **punto_luminoso.blend :** archivo fuente con el modelo 3D del punto luminoso y las cámaras utilizadas. Contiene código *Python* para renderizado.
    - **cam<Nro_cámara>**
      - **cam<Nro_cámara>_<Nro_cuadro>.tiff:** archivo que posee la imagen de un punto luminoso. La posición del punto luminosos en el espacio 3D en el cuadro <**Nro_cuadro**> es la misma para todas las cámaras.
- **Datos_Procesados :** Contiene dos carpetas con datos de calibración relevados a partir de las imágenes de la carpeta **Calibracion/Datos_Imagen/**.
  - **Tablero**
    - **calibracion.xml:** archivo con los resultados de la calibración a partir del método descripto en **Metodo.txt**.
    - **Metodo.txt :** archivo de texto que indica el método utilizado.
  - **Punto_Luminico**
    - **calibracion.xml:** archivo con los resultados de la calibración a partir del método descripto en **Metodo.txt**.
    - **Metodo.txt :** archivo de texto que indica el método utilizado.
- **Ground_Truth :** Contiene un archivo xml **calibracion.xml** con el ground truth de las cámaras utilizadas en la carpeta **Calibracion/Datos_Imagen/**.

Figura 3.9: Estructura de directorios para almacenar información de calibración.

Ambos códigos fueron implementados de manera independiente del número y disposición de las cámaras. El movimiento del esqueleto es exportado a un archivo BVH a partir de las herramientas disponibles en *Blender*.

Una vez finalizada la exportación de información del entorno *Blender* se implementaron rutinas de *Matlab* que permiten gestionar y ordenar dicha información en las secciones correspondientes de la estructura de directorios, conociendo el nombre del sujeto y el número de secuencia que distingue el movimiento desarrollado.

Armado el laboratorio y ubicadas las cámaras para la captura de varias secuencias, el proceso de calibración y almacenamiento de sus variables se efectúa una sola vez. Por lo que ambas estructuras de directorios, la de información de captura y la de información de calibración, se pueden almacenar en una única carpeta que identifique la configuración del laboratorio particular.

3.6. Estructura de datos.

Para poder trabajar a lo largo de todo el sistema de procesamiento se debe contar con una estructura de datos que mantenga y soporte toda la información de interés a lo largo proceso. También es deseable que los procesos utilicen dicha estructura sin quedar condicionados a futuras modificaciones de la misma, esta independencia requiere la existencia de funciones de acceso de entrada y salida a la estructura.

Con este fin se ha diseñado e implementado las estructuras de datos de las figuras 3.10 y 3.11. Las mismas permiten almacenar tanto la información de ground truth como la generada a la hora de trabajar en las etapas de análisis de video. Se han implementado funciones de acceso de entrada y salida en *Matlab*, así como varias herramientas de gestión que permiten entre otras cosas inicializar las estructuras, obtener o modificar cualquier parámetro, efectuar exportación e importación a xml y visualizar la información de manera interactiva generando para ello gráficas 2D o 3D.

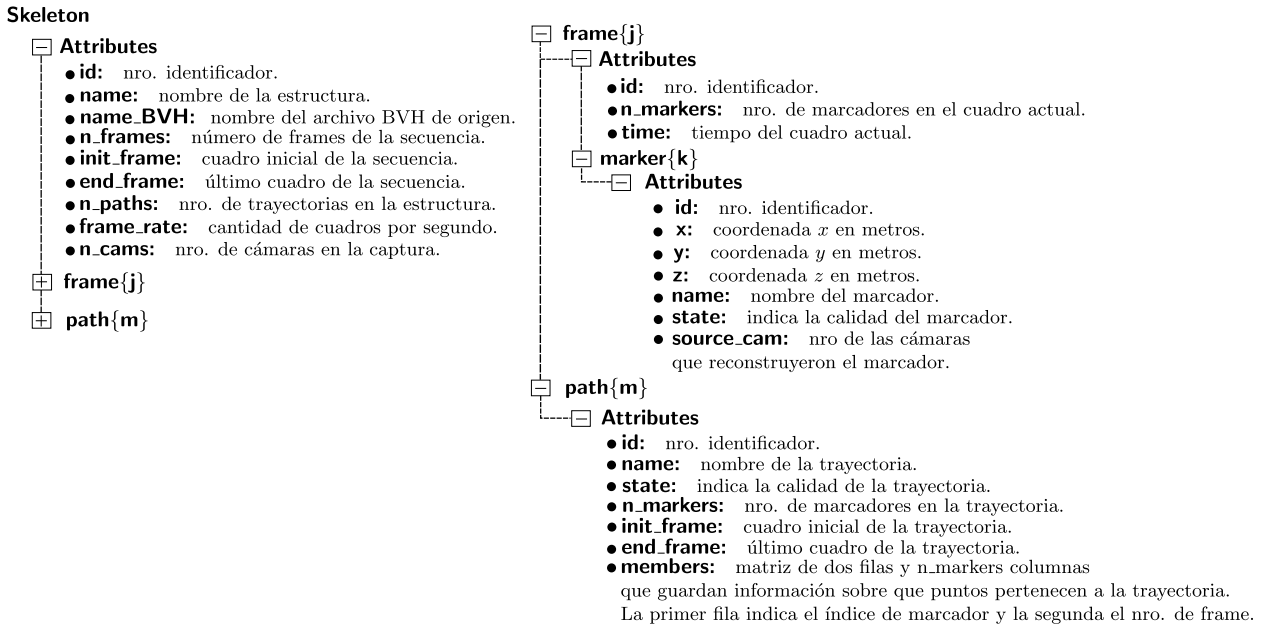


Figura 3.10: Estructura de datos para esqueletos.

Por ejemplo si se desea obtener una estructura para almacenar la información 3D del sujeto a lo largo del procesamiento, se inicializa una estructura `skeleton` a través de la función `init_structs()`, donde se indica entre otros parámetros el número de cuadros y número de marcadores. A partir de este momento se tiene una variable de nombre `skeleton` como la mostrada en la Figura 3.10 con sus campos inicializados. Luego se dispone las funciones `set_info()` y `get_info()`, que permiten modificar o acceder respectivamente a cualquier campo de la estructura. Las funciones que modifican la estructura lo hacen de una manera coherente, es decir si por ejemplo se introducen las coordenadas de un marcador en un cierto cuadro,

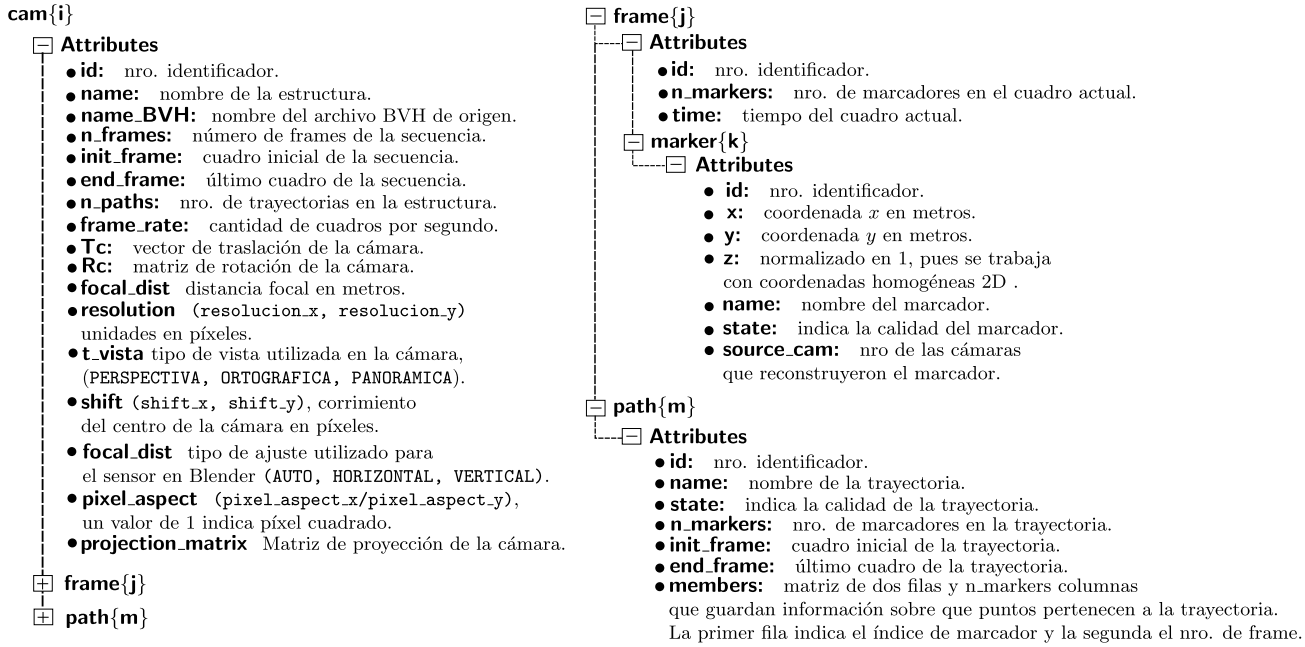


Figura 3.11: Estructura de datos para cámaras.

se incrementa automáticamente el atributo `n_markers` en el cuadro respectivo.

Básicamente se tienen dos estructuras principales, una para almacenar información del esqueleto asociado al modelo y otra para almacenar la información de las cámaras con las que se efectúa la captura de video.

Cabe destacar que este tipo de estructura es útil tanto para una base de datos sintética como una base de datos real.

3.7. Resumen

Si bien se encontraron numerosas bases de datos para la marcha, todas ellas terminan siendo descartadas por no ajustarse completamente a las hipótesis bajo las cuales se trabaja en este proyecto. Deficiencias tales como la inadecuada cantidad o posición de las cámaras, condiciones del laboratorio que dificultan y en algunos casos imposibilitan, una correcta segmentación a partir de la información óptica. Y por último la ausencia o tamaño inadecuado de los marcadores, son los factores más importantes.

Salvando diferencias, el problema principal está en que no se tienen imágenes ópticas para trabajar con marcadores, en general se tiende a trabajar sobre el cuerpo completo, y se contrasta el procesamiento con datos Mocap relevados con sistemas infrarrojos. Por otro lado esto último genera una rica fuente de material de trayectorias de puntos 3D en distinto tipo de movimientos, sobre todo en formatos Mocap.

Estas consideraciones impulsaron la idea de recorrer el camino inverso, crear una base de datos sintética a partir del material Mocap disponible, para luego

generar los videos en los cuales trabajar.

De todas maneras cabe destacar que se genera un relevamiento de bases de datos para el movimiento humano, se profundiza en las características usuales que presentan dichas bases de datos y se logra reunir un conjunto de conceptos y herramientas necesarios para introducirnos en el tema.

A la hora de configurar un laboratorio no solo hay que integrar las distintas variables, sino también cuidar que esta integración se gestione de manera compatible, de lo contrario se estaría desaprovechando el potencial de las secuencias generadas por el sistema de captura en cuanto al procesamiento posterior se trata, dificultando la tarea de obtención de datos a partir de las secuencias de video. En esta sección se logra tratar algunos parámetros claves y se discute las relaciones que estos deben tener para generar capturas de movimiento con información óptica basada en marcadores de bueno calidad.

A continuación se enumeran consideraciones de parámetros importantes para generar capturas en el caso de la marcha.

Es deseable para una buena resolución temporal poseer cámaras que permitan capturar 30 cuadros por segundo, con tiempos de obturación de al menos $1/2000\,s$, esto último permite evitar efectos de distorsión debidos a falta de nitidez. La resolución espacial de los datos en el procesamiento condiciona la resolución de la cámara. Se concluye que trabajando a 10 metros con resolución 1600×600 se obtienen incertidumbres de píxel de casi 1 cm.

El color de los marcadores debe contrastar claramente con la vestimenta y el fondo del espacio de captura, se recomienda una forma esférica. En capturas con cámaras ubicadas a menos de 12 metros del movimiento a relevar, un tamaño aceptable para el marcadores es de $3\,cm$ de diámetro.

La vestimenta debe ser ajustada, para despreciar fluctuaciones en la posición de los marcadores y preferiblemente de igual color que el fondo.

En cuanto a la iluminación, la misma debe ser uniforme, por ello si se utiliza iluminación artificial con focos puntuales es habitual colocar pantallas difusoras delante de los focos.

El espacio de captura debe contrastar con los marcadores, y sus dimensiones varían según el tipo de marcha a relevar. En caso de marcha rectilínea sobre una plataforma de $3\,m \times 5\,m$ se encuentra que 4 son el mínimo número de cámaras que permiten relevar el movimiento de manera satisfactoria. Mientras que en el caso de la marcha libre sobre una plataforma circular de $5\,m$ de diámetro, se recomienda la utilización de 8 cámaras. Una posible disposición de las mismas se muestra en la Figura 3.3.

Al investigar los diferentes formatos de captura de movimiento (MoCap) disponibles en las distintas bases de datos relevadas, surge naturalmente el formato BVH como el propicio para importar la información de la captura de movimiento al entorno virtual y a *Matlab*, su relativa sencillez y extendido uso facilitan sobremanera dicho pasaje.

Se decide trabajar en este proyecto con las fuentes BVH de la base de datos

MotionBuilder-friendly version ofrecidas por *cgspeed* [36], que provienen de las capturas de Carnegie Mellon University Motion Capture Database [26]. Alguno de los factores que justifican esta elección son que C.M.U. dispone de un gran número de capturas de movimiento de acceso público, varias utilidades de software que permiten llevar a otros formatos y es utilizado ampliamente en el ámbito de la animación por computadora.

Utilizando una suite de animación 3D gratuita y de código abierto como lo es *Blender*, se genera un laboratorio de captura de movimiento virtual, donde logra obtenerse cinco secuencias sintéticas dentro de los cuales cuatro son movimientos de marcha y una de corrida, con sus respectivos videos. Si bien las secuencias de video obtenidas son lo único necesario para el análisis posterior, al generar dichas secuencias a través de un entorno virtual controlado como lo es *Blender*, se puede obtener la información exacta del ambiente de captura. Información de las cámaras, iluminación, colores, ruido, posición de los marcadores en cada cuadro, etc. Permitiendo contar con un ground truth tanto del movimiento como de la calibración de las cámaras, sobre el cual validar los algoritmos en cada etapa del sistema de procesamiento. Esto último también permite obtener un benchmark de algoritmos, sumamente importante en futuros trabajos sobre el sistema.

Blender cuenta con una interfaz de programación de aplicaciones flexible, que permite extender su funcionalidad a través de programas en Python. Con el objetivo de automatizar varias etapas en el desarrollo de nuevas secuencias, así como gestionar la exportación de información desde *Blender* a otros lenguajes, se han creado múltiples scripts. En particular se posee un programa Python que automatiza la extracción de parámetros de interés hacía un archivo xml con cierta estructura particular ya definida en la base de datos.

La estructura de datos implementada cuenta con lo necesario para mantener y dar soporte a la información de interés a lo largo del proyecto. Cabe destacar que también soporta información de secuencias reales.

Se logra implementar un prototipo de base de datos lo suficientemente general y útil para trabajar con sistemas de captura de movimiento basada en marcadores. Las características de las herramientas utilizadas permiten generar secuencias de movimiento sintéticas, con relativa facilidad. El potencial actual y la posible expansión de la base de datos permiten afirmar que la misma es una herramienta a tener en cuenta en futuros proyectos de captura de movimiento.

Capítulo 4

Implementación de bloques del sistema

En base a lo estudiado en la etapa de investigación, se pudo concluir que un sistema de captura de movimiento con las características necesarias para cumplir el objetivo del proyecto debe estar formado por cuatro bloques generales: *calibración*, *detección de marcadores*, *reconstrucción* y *seguimiento*. En la figura 4.1 se muestra un esquema del sistema a implementar.

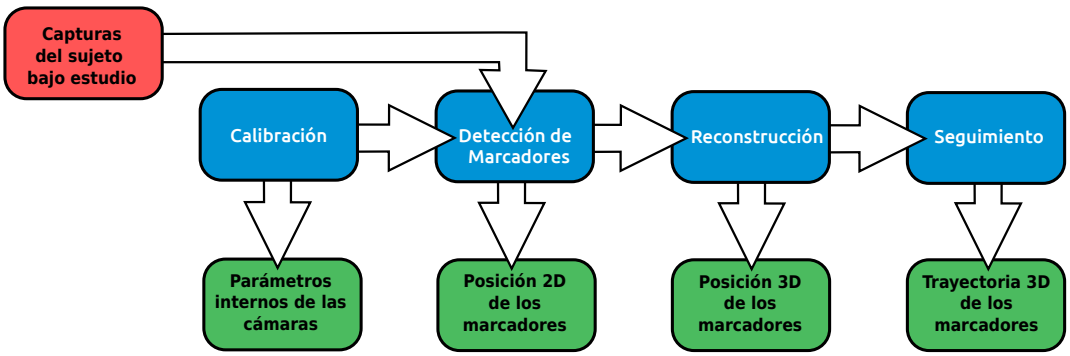


Figura 4.1: Diagrama de bloques del sistema completo.

A grandes rasgos, el sistema funciona de la siguiente manera:

1. Se *calibran* las cámaras. Esto es, determinar los parámetros de las mismas de forma tal de tener un mapeo del espacio 3D a las coordenadas 2D de las imágenes capturadas.

2. Se realiza la captura del paciente en movimiento desde todas las cámaras calibradas.

3. A partir de las secuencias, se realiza la *detección de marcadores* para cada cámara. Esto equivale a determinar la posición 2D de dichos marcadores en cada cámara donde están visibles, a lo largo de toda la secuencia.

4. Luego, con la posición 2D de determinado marcador en al menos dos cámaras se realiza la *reconstrucción* del mismo, es decir, obtener las coordenadas 3D

de dicho marcador. Se realiza para todos los marcadores, y para todos los cuadros de la secuencia.

5. Finalmente, se realiza el *seguimiento* (o *tracking*) de cada marcador en el espacio. Con esto se obtienen las trayectorias 3D de todos los marcadores en el cuerpo del paciente.

Debido a los tiempos disponibles para realizar el proyecto en su totalidad, y junto a la planificación realizada al comienzo del mismo, se tiene como idea principal el conseguir la implementación de un sistema con características similares, adaptando el mismo para cumplir los objetivos establecidos.

Sin embargo en la investigación bibliográfica se encuentra otra realidad, puesto que no hay muchos sistemas a disposición. Por un lado, la mayor parte de los sistemas encontrados poseen altos costos de licenciamiento (por ejemplo Vicon [2], OptiTrack [4], PhaseSpace [7], Qualisys [3] o MotionAnalysis [6]) y por otro lado, los sistemas de código abierto no se adaptaban a las necesidades presentes o el trabajo a realizar sobre los mismos era más costoso que hacer una implementación propia. En este último caso se encuentra el software Kinovea [8], que realiza únicamente el seguimiento en coordenadas 2D.

Se decide realizar una implementación propia de los bloques del sistema. Nuevamente, de acuerdo a la filosofía explicada en los párrafos anteriores, se prioriza la búsqueda de sistemas ya implementados cuyo diseño se corresponda a cada bloque de la Figura 4.1, antes de recurrir a la creación de uno.

Se tuvo presente en esta búsqueda el hecho de poder separar el sistema en bloques independientes. Esto asegura que la construcción de uno de ellos no dependa del correcto funcionamiento de otro. Por otro lado, da la posibilidad que en etapas futuras se pueda realizar el estudio de uno de los bloques de la Figura 4.1 individualmente y así poder modificarlo u optimizarlo sin afectar al resto. Esta forma de trabajo funciona adecuadamente siempre y cuando la salida de un bloque sea exactamente la entrada del siguiente, para los casos en que no se logra esto, se realizan algoritmos capaces de importar la salida de un bloque y convertirla al formato de entrada de otro (por ejemplo, *Xml2Struct* que convierte el xml que tiene como salida el bloque de detección de marcadores en estructuras de Matlab para realizar la reconstrucción).

Como se menciona anteriormente, en la búsqueda realizada se encuentra la tesis de doctorado de Lorna Herda [14], la misma plantea un sistema de captura de movimiento con las características buscadas para este proyecto. Al estudiar dicho sistema se encuentra que posee las mismas hipótesis de uso que el estudio preliminar realizado (utilización en fisioterapia, biomecánica, animación, deporte, etc.). Además, es un trabajo mencionado repetidas veces en otros artículos de la misma rama científica, encontrándose una documentación amplia respecto a la metodología implementada.

Debido a las ventajas que presenta esta metodología respecto a las otras encontradas, se decide implementar el sistema de Herda y elaborar los bloques de calibración y detección de marcadores aparte.

Estudiado el diseño propuesto por Herda para su sistema, se construye un diagrama de bloques completo, más detallado que el mostrado en la Figura 4.1, el cual se puede observar en la Figura 4.2. Es importante destacar que, si bien la mayor parte de los bloques de reconstrucción y seguimiento se realizaron con la metodología del sistema de Lorna Herda [14], en la documentación se presentan ciertas ambigüedades respecto a la descripción de algunos métodos y en la forma de resolver determinados casos de uso, que tuvieron que ser analizados y definidos por el grupo del proyecto en base a los conocimientos adquiridos.

A continuación se muestra el diagrama de bloques y se explica su funcionamiento.

4.1. Diagrama de bloques

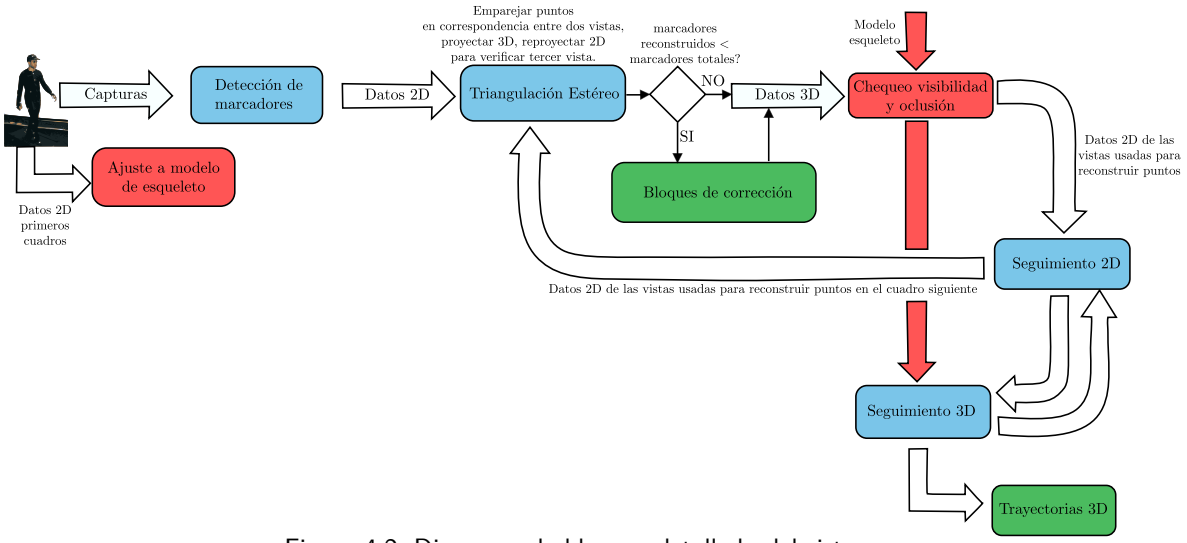


Figura 4.2: Diagrama de bloques detallado del sistema.

El sistema representado en este diagrama, funciona de la siguiente manera:

1. Se realiza la captura de movimiento del paciente desde múltiples vistas, en un entorno controlado. Estas capturas son la entrada principal al sistema.

2. A partir de las capturas, por un lado se ajusta el **modelo teórico de esqueleto** a utilizar, de acuerdo a las características del cuerpo del paciente, y por otro lado se realiza la **detección de marcadores** de cada cuadro para cada cámara. Este bloque es el mismo que el del diagrama de la Figura 4.1 y se explicará en detalle en el Capítulo 5.

3. Luego que se tiene la posición 2D de los marcadores en cada cuadro y en cada cámara, se realiza la **triangulación estéreo** para obtener la posición 3D de los mismos. A grandes rasgos, la *triangulación 3D* empareja dos puntos en correspondencia de dos vistas distintas y con ellos calcula la proyección 3D de ese punto en el espacio, luego se re-proyecta ese punto en las otras vistas

para verificar. Si verifica su posición en al menos una vista más, entonces la posición 3D se considera válida.

4. Si el número de marcadores reconstruidos en 3D es menor al número total de marcadores en el modelo de captura, se ingresa en el **bloque de corrección**, donde se utilizan varios métodos para recuperar la posición 3D de los marcadores restantes (ver Figura 4.3).

5. Cuando se tiene la posición 3D de todos los marcadores, se ingresa en el bloque de **chequeo de visibilidad y oclusión** donde se verifica que la posición reconstruida de cada marcador sea correcta y no se haya reconstruido alguno con datos erróneos.

6. Finalmente, se realiza el **tracking 3D y 2D** en simultáneo para reconstruir las trayectorias de cada marcador.

La explicación detallada de cada bloque y cómo fueron implementados se muestra en los capítulos siguientes.

A continuación, se explica como funciona el bloque de corrección:

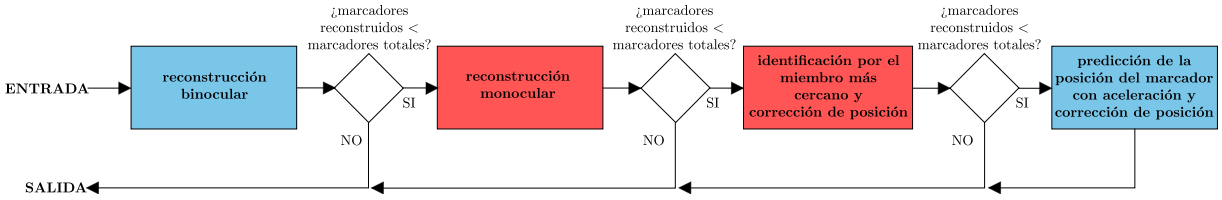


Figura 4.3: Detalle del bloque de corrección.

Una vez realizada la triangulación estéreo, se verifica que el número de marcadores reconstruidos sea igual a la cantidad de marcadores que efectivamente se estén usando en la adquisición de datos. Si esta condición no se verifica, se ejecuta un "bloque de corrección" para solucionarlo. Dicho bloque a su vez, contiene varios sub-bloques, los cuales pueden verse en la figura 4.3.

Primeramente se efectúa la *reconstrucción binocular* y si aún no se obtienen todos los marcadores se efectúa la *reconstrucción monocular*, se disminuye la exigencia sobre la reconstrucción al utilizar sucesivamente métodos menos precisos, con el fin de completar el número de *marcadores reconstruidos*.

Si en la salida de cada uno de los bloques anteriores la condición de *todos los marcadores reconstruidos* aún no se cumple, se pasa al siguiente bloque, donde se asocian los marcadores 3D reconstruidos que aún no fueron identificados con aquellas articulaciones del modelo de esqueleto (ajustado en la inicialización) que no tienen ningún marcador asociado. Para esto, se evalúan las distancias de los marcadores no identificados con la posición de las articulaciones del modelo en cuadros anteriores y se asocian aquellos marcadores que se encuentren a menor distancia a cada una de dichas articulaciones. Al tiempo que se realiza esto, se verifica que la distancia entre marcadores asociados a un mismo hueso del esqueleto

se mantenga aproximadamente constante (dado que los marcadores están fijos en los huesos y los mismos no varían su tamaño).

Si aún faltan marcadores sin reconstruirse, se utiliza como último recurso la estimación de la posición del marcador evaluando la aceleración del mismo en cuadros anteriores y verificando que dicha estimación sea coherente con el modelo de esqueleto.

Si bien se intentó reproducir el sistema propuesto por Herda [14] tal cual se especificaba en su documentación, se presentaron diversos obstáculos que impidieron poder implementar los bloques de reconstrucción y seguimiento como se detallan anteriormente. Un obstáculo importante son las ambigüedades presentadas en las especificaciones de los bloques en la documentación de Herda, dado que retrasaron la etapa de estudio del sistema al tener que investigar e implementar métodos para poder superar los vacíos presentes en la teoría. Esto último junto con la gran cantidad de módulos a implementar y el tiempo con que cuenta este proyecto generaron diferencias con lo propuesto por Herda.

A raíz de esto, se decide dar prioridad a los bloques principales del diagrama frente a los secundarios o a los que se implementan para casos de uso particulares. Con esto se intenta tener implementado un sistema de principio a fin, capaz de capturar la posición 3D de un sujeto realizando el movimiento de marcha a lo largo del tiempo con una performance aceptable. Estos bloques son:

- Detección de marcadores
- Triangulación estéreo (Reconstrucción)
- Seguimiento 3D
- Calibración

En los capítulos que siguen, se explicará el funcionamiento de estos bloques de forma detallada, así como su implementación y el análisis de resultados de cada uno de ellos.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Detección de marcadores

5.1. Introducción

Como se vio anteriormente, una vez realizada la captura de video del paciente, el primer paso en el procesamiento de las secuencias de video en un sistema de captura de movimiento es reconocer los marcadores en el cuerpo del sujeto, para luego tener la posición de cada uno de ellos en el espacio y a lo largo del tiempo.

La gran diferencia entre el sistema a implementar y los sistemas modernos es que, por temas de presupuesto, no se utilizarán sensores infrarrojos u otro tipo de tecnología más moderna que facilite (o evite) la etapa de segmentación. Además, cómo se mostró en el capítulo 3, las condiciones de captura de las secuencias son muy favorables lo que permite utilizar métodos simples de segmentación basados en el estudio de los píxeles de la imagen.

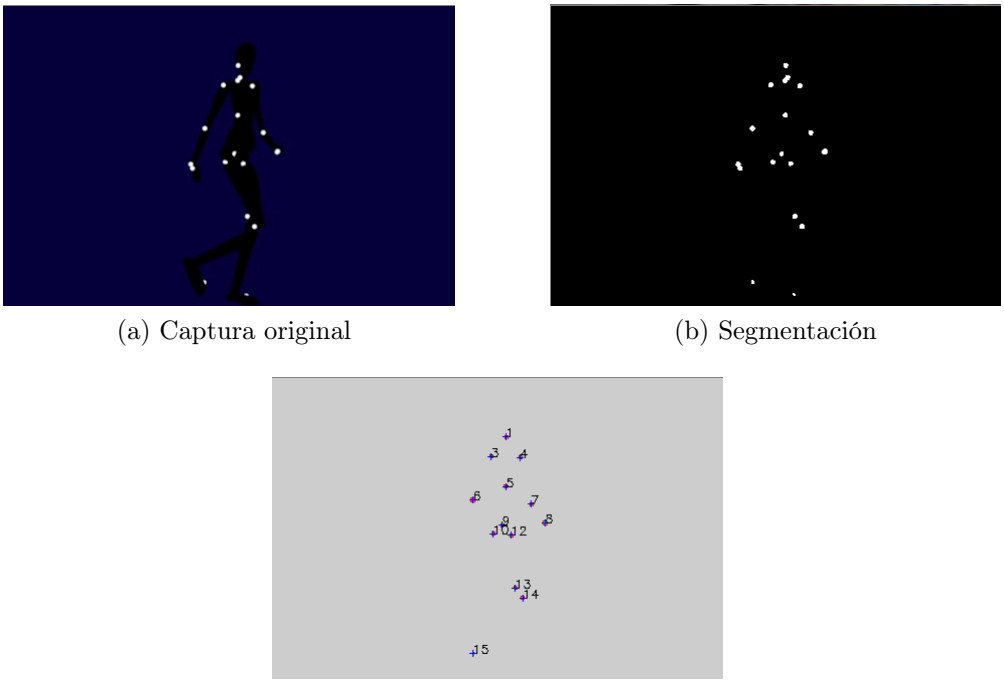
Debido a las condiciones del laboratorio donde se realizarán las capturas (esto es, marcadores blancos sobre un paciente con ropa oscura, fondo oscuro, e iluminación controlada), el problema de la detección de marcadores puede explicarse en líneas generales, como la detección de círculos blancos de un cierto tamaño sobre fondo oscuro.

El bloque de detección de marcadores, se puede dividir en dos partes bien definidas: la **segmentación** y el **filtrado de objetos** hasta obtener los marcadores. En la Figura 5.1 se muestra el resultado del procesamiento de cada parte.

El término segmentar hace referencia, en rasgos generales, a la división de una imagen en múltiples secciones u objetos para su posterior análisis. En otras palabras, la segmentación se encarga de identificar los objetos de importancia dentro de la imagen.

Algunas de las aplicaciones prácticas de la segmentación son:

- Pruebas médicas: localización de tumores, medida de volúmenes de tejido, cirugía guiada por computadora
- Localización de objetos en imágenes satelitales
- Sensor de huella dactilar



(a) Captura original

(b) Segmentación

(c) Detección

Figura 5.1: Ejemplo de funcionamiento del bloque.

- Reconocimiento facial
- Reconocimiento de iris
- Sistemas de control de tráfico
- Visión por computadora

El nivel de detalle de este proceso depende del problema a atacar. En este caso la segmentación juega un papel muy importante dentro del sistema ya que, en base a los datos obtenidos en ella se detectarán los marcadores en el cuerpo del paciente, que luego serán utilizados tanto en el tracking como en la reconstrucción 3D. Por otro lado, un error en la detección de la posición de los marcadores será imposible de detectar en etapas posteriores y generará un seguimiento 3D erróneo del mismo. Debido a esto, es recomendable tener la mayor exactitud posible, en especial si se tiene en cuenta que el sistema será utilizado al servicio de la medicina, por lo que deberá tener una precisión mayor que otros sistemas donde el nivel de detalle no juega un papel tan importante (como en el Kinect de Microsoft[1] por ejemplo).

Por otro lado, no hay que descuidar uno de los principales objetivos de este proyecto: *tener una primera versión del sistema de principio a fin con todos los bloques implementados*. En algunos casos, esto puede implicar la reducción de la complejidad del bloque y por ende, su precisión.

[1]http://www.xbox.com/es-ES/Kinect , Noviembre 2014

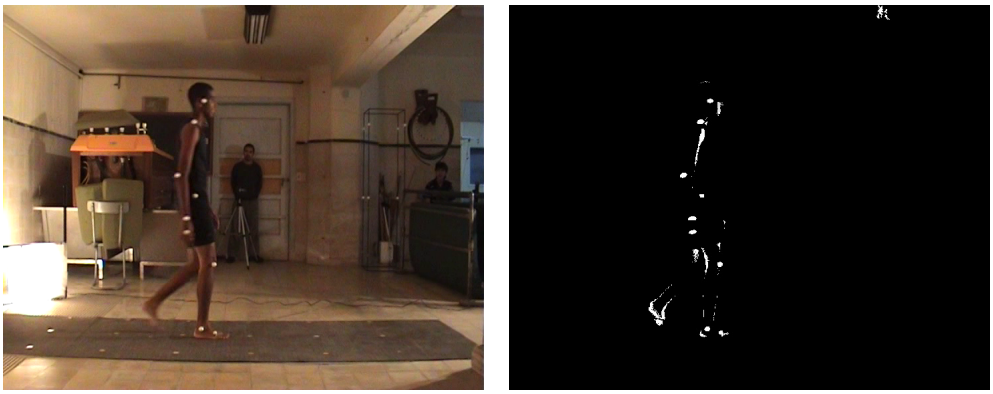
Existen varios métodos a aplicar para realizar la segmentación. Siguiendo lo aclarado en el párrafo anterior se comenzaron probando los más simples y se fue aumentando el nivel de complejidad hasta encontrar un método que se ajuste a los requerimientos. En la siguiente sección se describirán dos de los métodos más destacados, entre ellos la generación de umbrales con el método de Otsu [13], que fue el elegido para este sistema.

La etapa de filtrado de objetos es bastante más sencilla que la de segmentación. A fin de cuentas, este filtrado no es más que una clasificación de los objetos segmentados, y dado que los objetos a detectar tienen formas relativamente sencillas y las condiciones de laboratorio son controladas al realizar la captura, esta etapa no requerirá implementar algoritmos muy complejos. En particular, se implementó un detector de objetos circulares en base a momentos geométricos y un filtro según el área de los mismos.

A lo largo de este capítulo se explicarán los criterios de selección de los métodos utilizados y cómo funciona el algoritmo implementado. Por último, se mostrarán algunos resultados obtenidos al procesar imágenes sintéticas y reales con este bloque.

5.2. Fundamento teórico

En la Figura 5.2b se observa el resultado de segmentar el cuadro de la Figura 5.2a. Se confirma lo mencionado anteriormente sobre la simplicidad de los objetos a detectar, es decir, si se observan los objetos segmentados los marcadores no son difíciles de filtrar. Por este motivo no se realizó una investigación profunda del estado del arte de esta etapa, sino que se decidió implementar filtros sencillos y dedicarle más tiempo a la etapa de segmentación.



(a) Captura de video. (b) Segmentación.

Figura 5.2: Resultado de aplicar extracción de fondo y segmentación a una captura de video.

Para la etapa de segmentación, se realizó una revisión comenzando desde los algoritmos más básicos hasta encontrar alguno que cubra todos los requerimientos del sistema.

Existen varios métodos para realizar la segmentación, en particular los algoritmos para tratar imágenes monocromáticas generalmente se clasifican en dos grupos basados en la intensidad de los píxeles: discontinuidad y similaridad.

En los algoritmos basados en discontinuidad, se parte de la suposición que los límites de las regiones son suficientemente distintos unos de otros y a su vez del fondo, para permitir detectarlos en base a las discontinuidades de intensidad. Los algoritmos principales en esta categoría son los basados en **detección de bordes**.

Por otro lado, en los algoritmos basados en similaridad, se busca dividir la imagen en diferentes zonas donde los píxeles de cada una son similares entre sí y comparten ciertas características predefinidas. Los algoritmos más conocidos son los basados en identificación de regiones, como por ejemplo la **aplicación de umbral**.

A continuación se explican los algoritmos mencionados.

5.2.1. Detección de bordes

Consiste básicamente en la detección de líneas o transiciones en una imagen mediante el procesamiento de los píxeles que la componen. En lo que sigue se explica el detalle de este método, tal como se menciona en el libro *Digital Image Processing de Rafael Gonzalez y Richard Woods* [38].

Así como el difuminado en una imagen (que equivale a hacer un promedio de los píxeles en una zona) puede realizarse mediante la integración, los cambios de intensidad abruptos entre píxeles continuos pueden detectarse utilizando derivadas. Las derivadas de primer y segundo orden son en particular las más indicadas para este propósito.

Las derivadas de una función digital son siempre definidas en términos de diferencia. Hay varias formas de aproximar estas diferencias, pero para lograr detectar bordes de forma correcta es necesario que la aproximación usada para la derivada de primer orden cumpla los siguientes requisitos:

1. valga cero en áreas de intensidad constante
2. no valga cero al inicio de un escalón o rampa de intensidad
3. no valga cero en los puntos pertenecientes a una rampa de intensidad

De la misma forma, se requiere que la aproximación utilizada para la derivada de segundo orden verifique que:

1. valga cero en áreas de intensidad constante
2. no valga cero al inicio y al final de un escalón o rampa de intensidad
3. valga cero en los puntos pertenecientes a una rampa de intensidad

Considerando las propiedades de estas derivadas, se puede concluir que la de primer orden es la más adecuada para detectar bordes más “gruesos” y la segunda para detectar los más finos. Así mismo, para detectar puntos aislados la más adecuada es la derivada segunda, lo que no es de sorprender ya que la misma es

más sensible que la primera frente a cambios bruscos de intensidad. A raíz de esto, también se concluye que la derivada segunda es la más adecuada para detectar detalles finos (incluido el ruido). También es de destacar que mediante el signo de la derivada segunda se puede detectar si la transición en un borde (ya sea rampa o escalón) es de luz a oscuridad o viceversa.

Por otro lado, para realizar el procesamiento de las imágenes, se analizan las mismas como matrices numéricas. Una imagen en color, se traduce como tres matrices bidimensionales, una por cada componente cromática (por ejemplo rojo, verde, azul) siendo las filas y columnas de las matrices, el ancho y largo de la imagen. A efectos de simplificar el análisis, se estudian imágenes en escala de grises, lo cual implica trabajar con una sola matriz en vez de tres. La escala de grises en 8 bits va de 0 (negro), a 255 (blanco), para cada píxel de la imagen (ver Figura 5.3).

0	0	0	70	70	70
0	0	0	70	70	70
0	0	0	70	70	70
150	150	150	255	255	255
150	150	150	255	255	255
150	150	150	255	255	255

Figura 5.3: Imagen en escala de grises y su representación matricial.

La herramienta elegida para encontrar tanto la magnitud como la dirección de un borde en la posición (x, y) de la imagen f, es el gradiente denotado como ∇f y definido como:

$$\nabla f = grad(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{5.1}$$

Para detectar un borde en una imagen resta calcular el gradiente y luego su magnitud para cada píxel Si la superficie es uniforme, este vector será nulo (o muy pequeño) y si la superficie varía (por ejemplo, cuando hay un borde de por medio) el módulo de este vector será alto.

Por otro lado, el gradiente puede ser implementado para todos los valores de x e y pertinentes mediante el filtrado de la imagen $f(x, y)$ con las máscaras de la Figura 5.4.

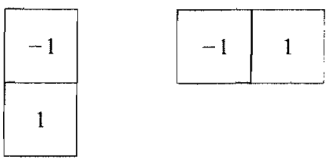


Figura 5.4: Máscaras de 1 dimensión.

Realizando esto se detectarán los bordes verticales y horizontales de la imagen. Para aumentar el detalle de la detección (por ejemplo para detectar bordes diago-

nales) es necesario aumentar las dimensiones de las máscaras, por ejemplo, a 2x2 o mejor aún a 3x3.

Existen variedades de máscaras aplicables, entre ellas la de Sobel (ver Figura 5.5), que presentan beneficios adicionales como la supresión de ruido, manteniendo la característica de detectar los bordes.

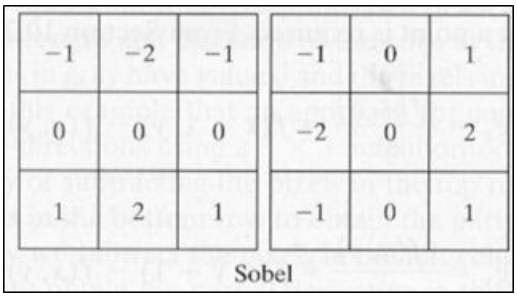


Figura 5.5: Máscara de Sobel.

En la Figura 5.3 se observaba una imagen de 6x6 píxeles y su representación matricial. Al aplicar la máscara de Sobel a la matriz de esta imagen, se detecta la transición entre los cuatro niveles de gris mientras que se igualan los niveles constantes. Si se aplica este procesamiento y luego se realiza una saturación de la intensidad de los pixeles se obtiene la matriz de la Figura 5.6.

0	0	255	255	0	0
0	0	255	255	0	0
255	255	255	255	255	255
255	255	255	255	255	255
0	0	255	255	0	0
0	0	255	255	0	0

Figura 5.6: Resultado de aplicar máscaras de Sobel sobre imagen original 6x6 y luego realizar una saturación de la intensidad de los pixeles.

Este método se puede combinar con otros para mejorar los resultados. Una posibilidad es someter la imagen a un proceso de *smooth* [39] -o suavizado- previo a la detección, de esta forma se descartan los bordes pequeños que en general son considerados como ruido. Otra posible combinación para realizar una detección más selectiva es la aplicación de un umbral luego del cálculo del gradiente. Cuando el interés recae tanto en destacar los bordes principales de una imagen como en obtener la mayor conectividad posible, es común que se aplique suavizado y umbral a la vez.

5.2.2. Métodos de umbral

Los métodos del valor umbral son un grupo de algoritmos cuya finalidad es segmentar los objetos de una imagen en función de un rango de valores. La pertenencia de un píxel a cierto segmento se decide mediante la comparación de alguna propiedad del mismo (por ejemplo su nivel de gris o nivel de luminosidad) con cierto valor umbral. Dado que esta comparación de valores se realiza individualmente para cada píxel, al método del valor umbral se le considera un método de segmentación orientado a píxeles.

Por lo tanto, mientras en los métodos de detección de bordes las regiones eran identificadas encontrando primero segmentos de borde y luego tratando de unir los mismos para formar bandas, en los métodos de umbral se trata de separar la imagen directamente en regiones basándose en la intensidad de estos píxeles y/o en otras propiedades, reduciendo el problema a encontrar el umbral correcto.

En la Figura 5.7 se observa el resultado de aplicar el método de umbral a la Figura 5.3 con un umbral de 200.

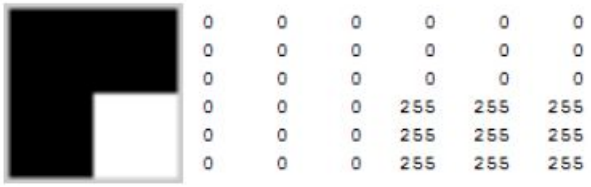


Figura 5.7: Resultado de aplicar un umbral de valor 200.

En este ejemplo se aplica el proceso más simple de generación de umbrales, sin embargo, en la mayoría de los casos ajustar el histograma de una imagen a esta forma no da tan buenos resultados. Para estas situaciones se recurre a la *umbralización múltiple*, donde un punto (x, y) se puede clasificar en varias clases dependiendo de la complejidad de la imagen.

Cuando la distribución de las intensidades entre objeto y fondo se distinguen suficientemente, es posible utilizar un solo umbral global aplicable en toda la imagen. Por otro lado, para aplicaciones donde el valor del umbral debe ir cambiando para una secuencia de imágenes, es recomendable utilizar algún método para calcular el valor del mismo automáticamente. En base a los factores que afectan la imagen y a las restantes características de la misma, se han implementado distintas formas de obtener el valor de umbral. El siguiente algoritmo iterativo muestra un ejemplo sencillo de como calcular el umbral:

1. Seleccionar un umbral inicial T estimado.
2. Segmentar la imagen utilizando T. Esto producirá dos conjuntos de píxeles, los que estén por encima del umbral (C_1), y los que estén por debajo (C_2).
3. Calcular el promedio de las intensidades (m_1 y m_2) de los píxeles en C_1 y C_2 respectivamente.

4. Calcular un nuevo umbral $T = \frac{1}{2}(m_1 + m_2)$.

5. Repetir los pasos 2 a 4 hasta que la diferencia entre los umbrales T de sucesivas iteraciones sea menor a un ΔT definido anteriormente.

Los principales métodos existentes para obtener el valor del umbral están listados en el survey de Mehmet Sezgin [12], en el cual se clasifica dichos métodos en las siguientes categorías:

- Basados en la forma del histograma.
- Basados en agrupamiento.
- Basados en la entropía de las regiones.
- Basados en los atributos de los objetos.
- Espaciales.
- Locales.

Entre los cuarenta métodos exhibidos en este paper, se encuentra el método de Otsu [13]. El mismo se encuentra dentro de los métodos basados en agrupamiento y es uno de los más utilizados en segmentación por umbral debido a su eficacia y simplicidad. Utiliza técnicas estadísticas para resolver el problema. En particular se utiliza la varianza que, como es sabido, es una medida de la dispersión de valores (en este caso se trata de la dispersión de los niveles de gris).

5.3. Justificación del algoritmo elegido

Siguiendo la metodología aplicada en todas las etapas de este trabajo, en principio se manejó la idea de tomar como base una implementación existente y adaptarla para el propósito de este proyecto. Sin embargo, en la etapa de investigación se encontró que no hay muchos sistemas de código abierto disponibles, y los que se encontraron necesitan muchos ajustes para llegar a cubrir las necesidades que se plantean aquí. En particular, para la segmentación se encontraron varias implementaciones de algoritmos, pero ninguna se ajusta a los requerimientos del proyecto. Por esta razón se decide implementar un algoritmo conocido, que aplique la teoría de la Sección 5.2.

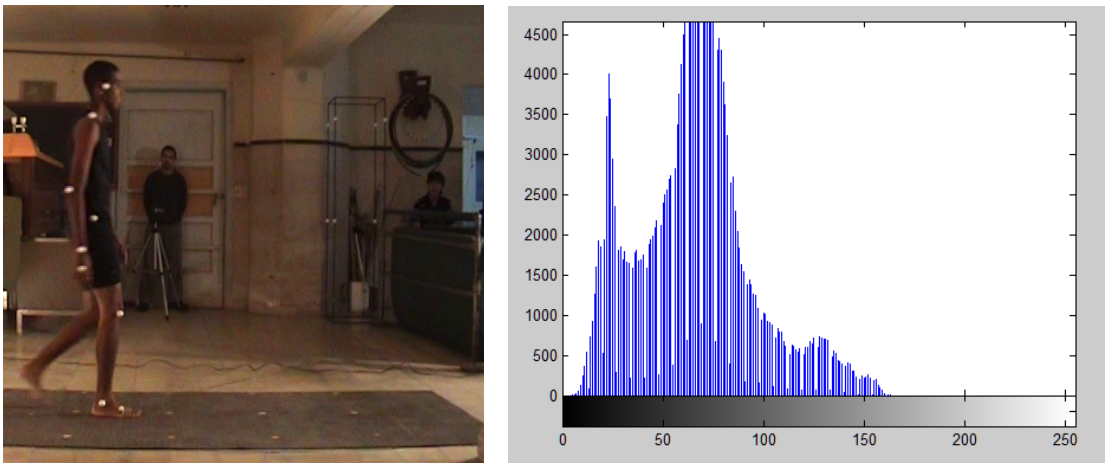
De los métodos nombrados en la sección anterior, si se piensa en los requerimientos del sistema, la detección de bordes presenta algunas desventajas frente a la detección por umbral. En primer lugar, si bien en los objetos a detectar hay bordes, los mismos no son tan nítidos debido al difuminado natural que impone la forma esférica de los marcadores. Este aspecto puede afectar negativamente la detección de bordes ya que el contraste entre un píxel y su vecino no es tan pronunciado, sin embargo en la segmentación por umbral, esta característica simplemente afectaría el tamaño de la esfera segmentada en función del valor del umbral impuesto. Además la detección de bordes presenta otros problemas que la umbralización no, y

que a efectos de este proyecto es mejor evitar, por ejemplo los bordes detectados usualmente quedan desconectados al aplicar detección de bordes e integrarlos es bastante costoso computacionalmente hablando. Por esto, y debido a sus propiedades intuitivas, simplicidad en la implementación y a su rapidez computacional, se eligió utilizar un método de umbral para implementar el bloque de *Segmentación*. En particular se eligió el método de Otsu [13] de tres clases ya que ofrece un buen compromiso entre simplicidad y eficacia, como se verá más adelante.

Para la elección del número de clases se tuvo en cuenta las características de la captura de video a procesar. Usar el número de clases incorrecto afecta en gran medida la segmentación. En un principio se podría suponer que cuánto mayor es el número de clases más precisa será la segmentación, sin embargo esto no es correcto. Un error típico al usar un número de clases mayor al adecuado es tener un único objeto segmentado en varios más pequeños. Por otro lado, utilizar un número menor de clases también es causa de errores. Por ejemplo, al utilizar 2 clases, se está asumiendo que el fondo es uniforme y esto no sucederá siempre en las capturas de video que se utilizarán en este sistema.

Para realizar esta elección se tuvo en cuenta que las capturas a procesar serán realizadas en un ambiente controlado y, siguiendo con la filosofía de priorizar la completitud del sistema frente a la eficacia del mismo, no se tuvo como prioridad para esta primera etapa implementar un método de mayor complejidad que sea más robusto frente a ciertos tipos de ruidos o características que se puedan dar en otro tipo de capturas (iluminación, fondo, ropa del paciente, etc.).

Con el método de Otsu [13] se pretende, a partir del histograma de la imagen, separar los píxeles de dicha imagen en tres niveles encontrando dos umbrales que los separen. Trabajar con tres clases permite ser un poco más flexible con los contrastes entre los marcadores y el resto de la imagen por lo que no sería estrictamente necesario, por ejemplo, que el traje del paciente y el fondo sean del mismo color.



(a) Original (b) Histograma

Figura 5.8: Captura original de un paciente real y su histograma de intensidad de pixeles.

En la Figura 5.8b se observa el histograma de intensidad de píxeles de la imagen

5.8a, que corresponde a una captura de un paciente real. Dicho histograma muestra 3 picos definidos, por lo que utilizar el método de Otsu de 3 clases parece una buena opción para determinar los umbrales de segmentación. Manteniendo esta relación de intensidades entre fondo, ropa del paciente y marcadores, este método funcionaría adecuadamente para el bloque.

El bloque de segmentación de este sistema fue implementado en el lenguaje C++, debido a que es uno de los lenguajes de programación que cuenta con mayor cantidad de recursos para procesamiento de imágenes. En particular, se utilizaron las librerías *OpenCV* [40] y CVBlob [41] ya que funcionan para las plataformas principales de PC y dispositivos móviles, y están diseñadas para tener una gran eficiencia computacional en las implementaciones. Además, estas librerías son bastante populares dentro de las librerías de código abierto con similares características, lo cual implica que poseen una comunidad activa de usuarios muy grande.

La idea inicial fue realizar la totalidad del sistema en C++, sin embargo, por razones prácticas[2] se decidió implementar los bloques de reconstrucción, calibración y tracking en *Matlab*.

Por otro lado, C++ es un lenguaje de una complejidad superior a otros disponibles, por lo que implementar el algoritmo con el mismo implicó mayor esfuerzo (por ejemplo, en el manejo de memoria[3]) que de haber utilizado un lenguaje que trabaje a más alto nivel.

Finalmente, la justificación de la realización del algoritmo para el bloque de segmentación puede entenderse desde dos puntos de vista:

- Por un lado, se elaboró un algoritmo de una complejidad relativamente baja como se estableció en los requerimientos, pero que cubre todas las características necesarias para esta clase de sistema. En la sección 5.6 se verá que los resultados obtenidos son buenos y el error cometido está dentro de los márgenes que aseguran el buen funcionamiento del sistema.

- Por otro lado, a pesar de que el estado del arte de segmentación presenta una gran cantidad de algoritmos, el estado del arte "industrial" está bastante más atrasado del estado del arte referente a investigaciones, debido al tiempo que implica realizar una nueva implementación o la dificultad para conseguir una que funcione significativamente mejor a las existentes. Por esto, implementar una primera versión del sistema con el algoritmo de segmentación elegido no está muy lejos de lo utilizado actualmente en la "industria". Nuevamente, es importante aclarar que este bloque (como el resto de los bloques) está implementado de forma tal que en un futuro se pueda optimizar tanto como se quiera, sin afectar el resto de los bloques del sistema. Esto da la posibilidad de modificar el sistema no solo para mejorar la segmentación sino también para robustecerla, permitiendo por ejemplo, realizar capturas fuera

[2] La mayoría de los integrantes del equipo cuenta con mayor experiencia en *Matlab* que en C++.

[3] C++ no cuenta con soluciones que realicen asígnación y limpieza de memoria automáticamente, como por ejemplo el *Garbage Collector* en Java, por lo que debe realizarse de forma manual.

de las condiciones del laboratorio, como usualmente es necesario en el ámbito deportivo.

5.4. Detalles técnicos de la implementación

A continuación, se presentan algunos conceptos básicos necesarios para entender la implementación de un algoritmo con segmentación por umbral. Además se explican los conceptos de momentos y excentricidad, herramientas que fueron aplicadas para el diseño del filtro circular.

5.4.1. Segmentación con umbral

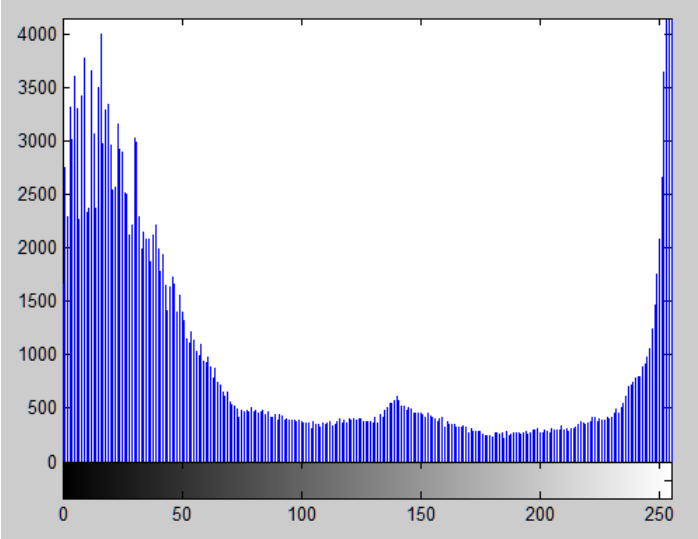


Figura 5.9: Histograma de intensidad de una imagen.

Considerando la Figura 5.9 como el histograma de intensidad de una imagen $f(x, y)$, se puede apreciar que la misma está compuesta por un objeto u objetos claros, iluminados, de aproximadamente la misma intensidad y un fondo oscuro. De esta manera, se definen en este histograma dos campanas bien determinadas. La manera más obvia de extraer los objetos del fondo es seleccionando un umbral T que separe estas dos campanas y por lo tanto cualquier punto (x, y) de la imagen que cumpla $f(x, y) > T$ será un punto perteneciente al objeto mientras que el resto son puntos pertenecientes al fondo. De acuerdo a lo anterior, la imagen segmentada puede definirse de la siguiente manera.

$$g(x,y) = \begin{cases} 1 & si \quad f(x,y) > T \\ 0 & si \quad f(x,y) \leq T \end{cases} \tag{5.2}$$

Si T toma un valor constante en toda la imagen, al proceso se le llama *umbralización global*. Por otro lado, si T cambia en una misma imagen el proceso es

llamado *umbralización variable*. A veces se utiliza el término *umbralización local o regional* en la umbralización variable cuando el valor de T en un punto (x, y) depende de las propiedades de los puntos de alrededor.

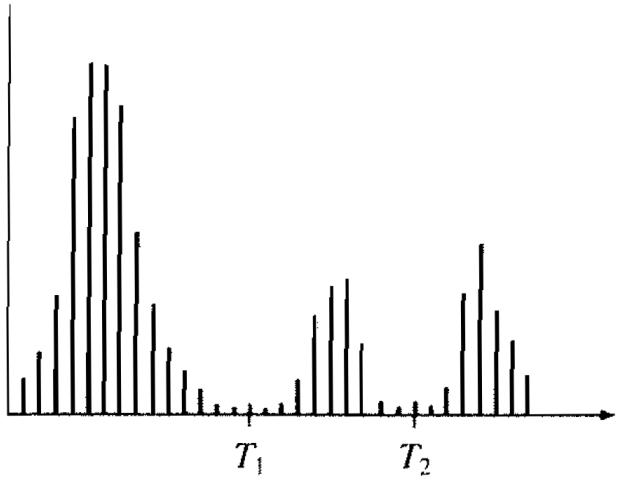


Figura 5.10: Histograma de intensidad de tres clases [42].

En la figura 5.10 se puede ver el histograma de una imagen con tres clases dominantes, que puede corresponder por ejemplo, a un objeto brillante, otro un poco menos brillante y un fondo oscuro. En este caso la umbralización de tres clases clasificará el punto (x, y) como perteneciente al fondo si $f(x, y) \leq T_1$, perteneciente a un objeto si $T_1 < f(x, y) \leq T_2$ y perteneciente al objeto más brillante si $f(x, y) > T_2$. Por lo tanto, la imagen segmentada será de la forma:

$$g(x,y) = \begin{cases} a & si \quad f(x,y) > T_2 \\ b & si \quad T_1 < f(x,y) \leq T_2 \\ c & si \quad f(x,y) \leq T_1 \end{cases} \tag{5.3}$$

donde a, b y c son tres valores distintos de intensidad.

Observando los histogramas anteriores, puede verse que la efectividad de la umbralización está directamente relacionada con el ancho y la profundidad de los valles que separan las distintas clases. Siguiendo esto, los factores claves que afectan directamente al tamaño de estos valles son:

- Separación entre picos. Cuanto más separados, mejor posibilidad de segmentar correctamente.
- Ruido de la imagen.
- La relación entre los tamaños de los objetos y el fondo.
- La uniformidad de la iluminación.
- La uniformidad de las propiedades de reflexión de la imagen.

Es de destacar que, si bien no resulta tan evidente como pasa con el ruido de la imagen, la iluminación y las propiedades de reflexión juegan un papel clave para obtener una segmentación efectiva y por lo tanto controlar estos parámetros debe ser prioridad si se quiere obtener una buena segmentación. Cuando no es posible controlarlos, existen tres aproximaciones básicas que se pueden realizar para mejorar los resultados: corregir el patrón de sombras directamente, corregirlo mediante algún proceso ya establecido (por ejemplo utilizando la transformada top-hat [43]) o aplicar un umbral variable tal como fue mencionado anteriormente.

Umbral de Otsu

El método de Otsu [13] calcula el valor umbral de forma tal que la dispersión dentro de cada segmento sea lo más pequeña posible, pero al mismo tiempo sea lo más alta posible entre segmentos diferentes. Para ello se calcula el cociente entre ambas varianzas (para el caso de dos clases) y se busca un valor umbral para que este cociente sea máximo.

Dicho de otro modo, se puede ver al proceso de umbralización como un problema estadístico cuyo objetivo es minimizar el error promedio que se produce al asignar los píxeles de la imagen a dos o más clases. La solución a este problema es conocida como *regla de decisión de Bayes [44]*. Sin embargo no es trivial aplicar esta regla, ya que estimar la densidad de probabilidad de cada clase no es simple. El método de Otsu es considerado una de las mejores aproximaciones a esta solución, ya que maximiza la "varianza intermedia entre clases" (*between class variance*[4]) que es una medida muy utilizada en problemas de discriminación estática, lo que permite obtener un umbral óptimo. A esto se le suma la ventaja de que todos los cálculos realizados en el método se realizan sobre el histograma de intensidades, que es muy fácil de obtener.

La "varianza intermedia entre clases" puede escribirse como

$$\sigma_B^2 = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 = P_1P_2(m_1 - m_2)^2 = \frac{(m_GP_1 - m)^2}{P_1(1 - P_1)} \quad (5.4)$$

En esta ecuación, P_1 y P_2 son las probabilidades de que un píxel sea asignado a la clase 1 o 2 respectivamente, m_1 y m_2 son las medias de las intensidades de cada una de estas clases. Además, $m(k)$ es la media (intensidad promedio) acumulada hasta el nivel k y m_G es la intensidad media (intensidad global promedio) de la imagen en su totalidad. Por lo que, para L posibles niveles de intensidad en una imagen digital se tiene:

$$m(k) = \sum_{i=0}^{k} ip_i \quad (5.5)$$

$$m_G(k) = \sum_{i=0}^{L-1} ip_i \quad (5.6)$$

[4]diferencia entre la varianza total y la suma de las varianzas de cada clase [45]

Considerando que k es el umbral que separa la clase 1 de la clase 2, P_1 puede escribirse como

$$P_1(k) = \sum_{i=0}^{k} p_i \tag{5.7}$$

donde p_i es la probabilidad para los píxeles de la imagen de tener intensidad i.

Por lo que la ecuación 5.4 también queda dependiendo del umbral k:

$$\sigma_B^2(k) = \frac{(m_G P_1(k) - m(k))^2}{P_1(k)(1 - P_1(k))} \tag{5.8}$$

Para el caso de umbralización con múltiples clases (K clases), la varianza intermedia vale:

$$\sigma_B^2(k) = \sum_{k=1}^{K} P_k(m_k - m_G)^2 \tag{5.9}$$

donde

$$P_k = \sum_{i \epsilon C_k} P_i$$

$$m_k = \frac{1}{P_k} \sum_{i \epsilon C_k} i p_i$$

y m_G es la ganancia global como se definió anteriormente. Esta umbralización implica tener $K - 1$ umbrales.

A modo de ejemplo, para tres clases (tres niveles de intensidades separadas por dos umbrales) la "varianza intermedia entre clases" queda:

$$\sigma_B^2(k) = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2 \tag{5.10}$$

donde

$$P_1 = \sum_{i=0}^{k_1} p_i$$

$$P_2 = \sum_{i=k_1+1}^{k_2} p_i$$

$$P_3 = \sum_{i=k_2+1}^{L-1} p_i$$

y

$$m_1 = \frac{1}{P_1} \sum_{i=0}^{k_1} i p_i$$

$$m_2 = \frac{1}{P_2} \sum_{i=k_1+1}^{k_2} i p_i$$

$$m_3 = \frac{1}{P_3} \sum_{i=k_2+1}^{L-1} ip_i$$

Además, como en el caso de dos clases, se dan la siguientes relaciones:

$$P_1 m_1 + P_2 m_2 + P_3 m_3 = m_G \tag{5.11}$$

y

$$P_1 + P_2 + P_3 = 1 \tag{5.12}$$

Luego, aplicando lo visto acerca del umbral de Otsu, se tiene que el umbral óptimo k^* es el valor de k que maximiza la Ecuación 5.8 (para el caso de múltiples clases, serían los valores de k_k^* que maximizan la 5.9). Para encontrar k^* basta con evaluar la Ecuación 5.8 para todos los valores de k válidos[5] y seleccionar el valor de k que maximiza dicha ecuación. Si el máximo $\sigma_B^2(k)$ se da para varios k, k^* se calcula como el promedio de los k que dan dicho valor.

Para el ejemplo del algoritmo de tres clases, se deberían encontrar los valores de k_1 y k_2 que maximicen la varianza entre clases. Para ello, se evalúa la Ecuación 5.9 para todos los pares (k_1, k_2) posibles, es decir: (k_1, k_2) tq $0 < k_1 < k_2 < L - 1$.

Algo importante a destacar es que este método es poco costoso en términos computacionales ya que el máximo numero de k's para los que hay que evaluar la Ecuación 5.8 es L, que corresponde a la cantidad de niveles de intensidad de la imagen.

En resumen, el *algoritmo de Otsu* se puede implementar de la siguiente manera:

1. Realizar el histograma de la imagen, donde cada componente corresponde a un nivel de intensidad (con un total de L niveles)
2. Calcular la probabilidad $P_1(k)$ con la Ecuación 5.7 para $k = 0, 1, 2, ..., L - 1$.
3. Calcular la media $m(k)$ con la Ecuación 5.5 para $k = 0, 1, 2, ..., L - 1$.
4. Calcular la media global m_G con la Ecuación 5.6.
5. Calcular la "varianza intermedia entre clases" (*between-class variance*), $\sigma_B^2(k)$, como se muestra en la Ecuación 5.8 (o 5.9) para $k = 0, 1, 2, ..., L - 1$.
6. A partir del punto anterior, obtener el umbral de Otsu k^* como el valor de k (o los valores de k_k para el caso de múltiples clases) que maximiza $\sigma_B^2(k)$.

5.4.2. Clasificación de objetos

Existe una gran cantidad de métodos para clasificar objetos en imágenes, la complejidad de los mismos varía dependiendo de la clasificación a realizar. Algunos se basan en las características de los objetos segmentados mientras que otros utilizan las características de la imagen completa.

A continuación se explican los conceptos de momentos y excentricidad, los cuales se utilizaron para elaborar el algoritmo de clasificación de círculos utilizado.

[5] todos los k enteros tal que $0 \leq k \leq L - 1$ (con $L - 1$ nivel de intensidad máximo de la imagen) que verifiquen $0 < P_1(k) < 1$

Momentos y excentricidad [46]

Los momentos geométricos son propiedades numéricas que se pueden obtener en una determinada imagen, los cuales proporcionan una alternativa interesante para la representación de la forma de un objeto. Tienen en cuenta todos los píxeles de la imagen, no sólo los bordes y se pueden clasificar en:

- Momentos simples
- Momentos centrales
- Momentos centrales normalizados

Los *momentos simples* se emplean para obtener otros momentos, pero también dan información por sí mismos. El momento de orden $(p+q)$ se calcula como

$$m_{pq} = \int\int x^p y^q f(x,y) dx dy$$

donde $f(x,y)$ es una función continua. En el espacio discreto se calculan:

$$M(p,q)) = \sum_x \sum_y x^p y^q f(x,y)$$

A partir de esta definición se puede ver, por ejemplo, que el momento simple de orden 0 representa el área de la figura en imágenes binarias, ya que es la suma de los valores de todos los píxeles:

$$M(0,0) = \sum_x \sum_y f(x,y)$$

Por otro lado, los *momentos centrales* son utilizados para reconocer figuras dentro de una imagen, independientemente de su posición. Se calculan como

$$\mu_{pq} = \int\int (x-X)^p (y-Y)^q f(x,y)$$

o para el espacio discreto como

$$MC_{pq} = \sum\sum (x-X)^p (y-Y)^q f(x,y)$$

donde (X,Y) corresponden al centro de masa de la figura (centroide) y se calcula con los momentos simples de orden 0 y 1 de la siguiente manera:

$$X = \frac{M(1,0)}{M(0,0)} \quad Y = \frac{M(0,1)}{M(0,0)}$$

Finalmente, los *momentos centrales normalizados*, permiten reconocer figuras dentro de una imagen, independientemente de su posición y de su tamaño. Para ello, se normalizan los momentos centrales con el momento de orden 0, obteniendo así figuras independientes de la escala:

$$MCN(p,q) = \frac{MC(p,q)}{MC^{\beta}(0,0)}$$

donde $\beta = \frac{p+q}{2} + 1$.

El reconocimiento de la forma de un objeto, se analiza a partir de los momentos de orden 2 (es decir, $p + q = 2$). En ellos, la densidad de la figura se multiplica por distancias al cuadrado desde el centro de masas o centroide. Así por ejemplo, con los tres momentos centrales de segundo orden ($MC(0,2)$, $MC(2,0)$, $MC(1,1)$) se forman las componentes del tensor de inercia o matriz de rotación. A partir de estas componentes, también se puede calcular el ángulo de rotación de la figura alrededor de su centro de masas y la excentricidad. Esta última, se calcula a partir de la Ecuación 5.13, donde n_{20}, n_{02} y n_{11} son los momentos centrales normalizados de orden 2.

$$e = \sqrt{1 - \frac{\frac{n_{20}+n_{02}}{2} - \sqrt{4n_{11}^2 + \frac{(n_{20}-n_{02})^2}{2}}}{\frac{n_{20}+n_{02}}{2} + \sqrt{4n_{11}^2 + \frac{(n_{20}-n_{02})^2}{2}}}} \tag{5.13}$$

La excentricidad es un parámetro asociado con cada sección cónica. Es una buena medida de cuánto se diferencia dicha sección de ser circular. Así, un círculo tendrá excentricidad 0, una elipse entre 0 y 1, para una parábola valdrá 1 y para una hipérbola será mayor a 1.

5.5. Implementación

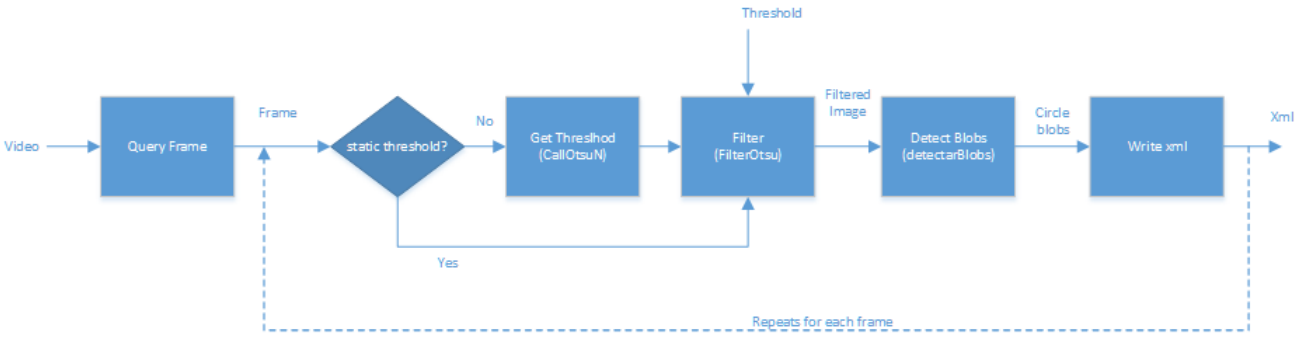


Figura 5.11: Diagrama de flujo del algoritmo de segmentación.

En la Figura 5.11 se presenta un diagrama dónde se observa el flujo del algoritmo de segmentación realizado. Los nombres que aparecen entre paréntesis dentro de algunos bloques son los nombres de las funciones dentro del código que implementan cada bloque[6].

El algoritmo realiza la segmentación siguiendo el siguiente proceso:

1. Se recibe como entrada un video y este es separado en cada uno de sus cuadros a través del bloque *Query Frame*.

[6]Como las funciones están en inglés, a diferencia de los diagramas mostrados anteriormente, estos están en inglés.

2. Se toma un cuadro, y se calcula el umbral de Otsu con el bloque *Get Threshold*. Si al comenzar la segmentación es ingresado un umbral fijo, este paso se saltea.

3. Con el umbral calculado (o ingresado), se filtra el cuadro en el bloque *Filter*.

4. A partir de la imagen filtrada, se identifican los marcadores con el bloque *Detect blobs*.

5. Se escribe la posición de los marcadores detectados para este cuadro en un archivo con formato XML.

6. Se toma el siguiente cuadro y se repite el proceso a partir del paso 2.

El bloque *Query Frame* es implementado mediante las funciones *cvCaptureFromAVI* y *cvQueryFrame*, las cuales pertenecen a la librería *OpenCV* [40].

Por otro lado, el bloque *Get Threshold* contiene una implementación del algoritmo de Otsu de N clases [47], con $N = 3$. Se utilizó esta implementación ya que la implementación que hay en la librería *OpenCV* es de dos clases y tampoco se encontraron otras implementaciones del algoritmo para 3 clases en internet que cumplieran los requerimientos necesarios para el bloque.

Como se vio en la Sección 5.2.2, este método devuelve 2 umbrales de los cuales se tomará el mayor de ellos, dado que las hipótesis del problema establecen que la adquisición de video debe realizarse sobre fondo oscuro y con el paciente utilizando ropa oscura, de forma tal que los marcadores (que son de color blanco) sean los elementos más claros en la imagen.

En la Figura 5.12, se observa un diagrama que describe el funcionamiento del bloque *Filter*. Este bloque es el encargado de filtrar la imagen según la intensidad de los píxeles y está implementado por la función *FilterOtsu*, que recibe como parámetros de entrada una imagen (uno de los cuadros de la secuencia) y el umbral a utilizar para el filtrado. Primero se le aplica a la imagen un difuminado (*smoothing*) con un filtro de mediana, con el objetivo de reducir el ruido. Luego se cambia el espacio de colores de la imagen de RGB a HSV ya que este último es más adecuado para realizar segmentación basada en la intensidad de píxeles [48]. Por último, se filtra la imagen (sobre el canal V) con el umbral ingresado utilizando la función *cvThreshold* de la librería *OpenCV*. Esta función compara la intensidad de cada píxel de la imagen con el valor del umbral estableciendo un nuevo valor de intensidad: 0[7] para los píxeles que originalmente tenían intensidad menor al umbral y 255[8] para los que originalmente presentaban intensidad mayor.

En la Figura 5.13, se puede ver un ejemplo de los resultados de procesar una imagen sintética de la base de datos, con el bloque *Filter*. Se puede ver que la intensidad de los píxeles azules y negros queda por debajo del umbral, mientras que la de los píxeles blancos queda por encima.

Como se ve en la Figura 5.13b, para el caso ideal todos los píxeles resultantes del filtrado corresponden a los marcadores en el paciente. En la práctica, no

[7] negro en el espacio HSV.

[8] blanco en el espacio HSV.

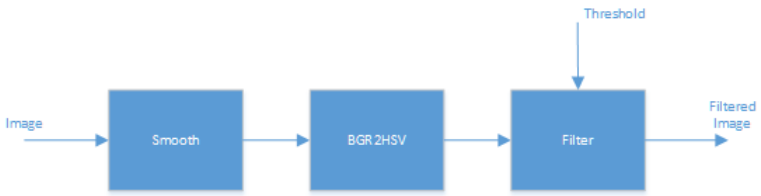
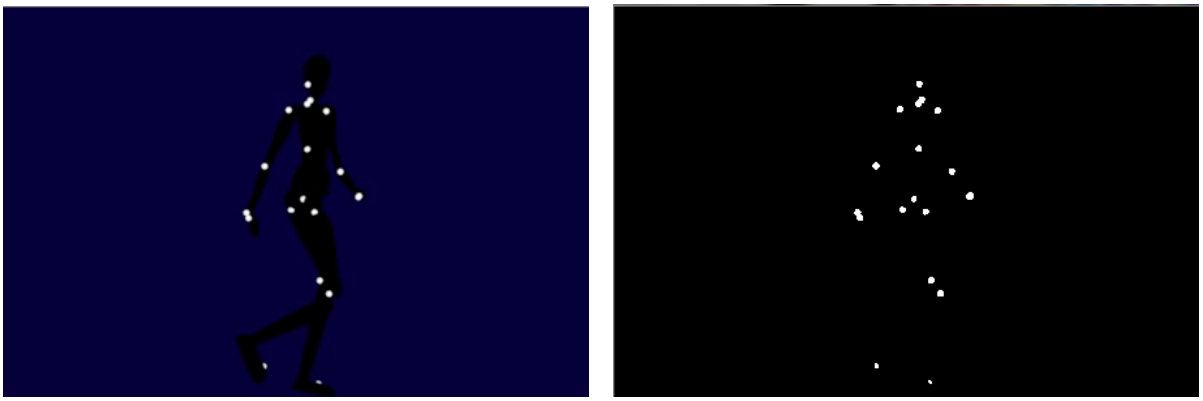


Figura 5.12: Diagrama de flujo del bloque de umbralización.



(a) Captura original de una secuencia sintética. (b) Imagen filtrada con el umbral de Otsu.

Figura 5.13: Entrada y salida del bloque de umbralización.

siempre sucede esto, por lo que luego de obtenidos los mismos debe hacerse una diferenciación entre los marcadores y el resto de los píxeles detectados.

El funcionamiento del bloque *Detect blobs* es descrito por el diagrama de la Figura 5.14. Este bloque recibe como entrada la imagen previamente filtrada por el bloque *Filter* y da como salida una imagen con los marcadores detectados e identificados. En primer lugar, se identifican todos los blobs[9] de la imagen filtrada con el bloque *Find blobs*, que es implementado por la función *cvLabel* de la librería CVBlob [41]. Cuando se hace referencia a "identificar todos los blobs", se refiere a identificar cada grupo de píxeles blancos continuos de la imagen filtrada como un objeto (un blob) único. En la Figura 5.16a, se ve el resultado de procesar la Figura 5.13b con el bloque *Detect blobs*.

Luego, si se ingresó la opción para filtrar por área, los blobs [49] detectados se filtran por área máxima y/o mínima mediante la función *cvFilterByArea* perteneciente a la librería CVBlobs [41].

Siguiendo el flujo, la imagen con blobs ingresa al bloque *Circular filter* se haya filtrado por área o no, donde se descartan los blobs que no tienen forma circular. Para ello, se realiza un cálculo basado en la excentricidad[10] de los objetos y en los

[9] Binary Large Objects [49]

[10] Parámetro asociado con cada sección cónica. Es una buena medida de cuánto se diferencia dicha sección de ser circular. Así, un círculo tendrá excentricidad 0, una elipse entre

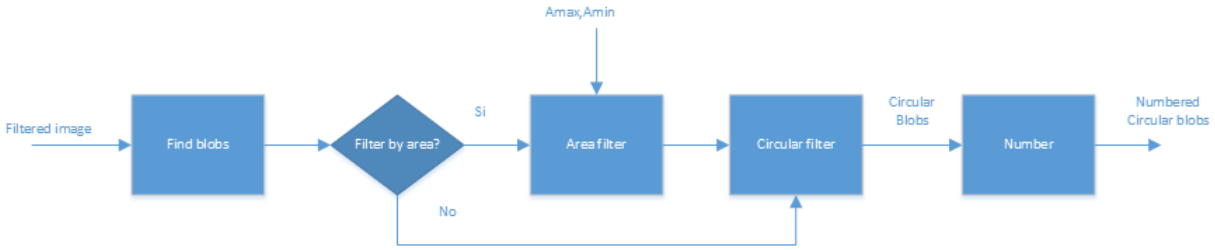


Figura 5.14: Diagrama de flujo del bloque de detección de blobs.

momentos de la imagen. Estas propiedades se explicaron en la Sección 5.4.

De los blobs detectados en un cuadro, los que tendrán forma aproximadamente circular serán los que tengan el valor de la excentricidad más cercana a 0. Observando la Ecuación 5.13, se puede ver que $e \to 0$ si

$$\sqrt{4n_{11}^2 + \frac{(n_{20} - n_{02})^2}{2}} \to 0$$

por lo que una buena medida de qué tan circular es un blob puede ser evaluar n_{11} y $n_{20} - n_{02}$ y ver qué tan chicos son sus valores en comparación con $n_{20} + n_{02}$.

Precisamente, para la implementación del bloque *Circular Filter*, se evalúa para cada blob

$$\frac{n_{11}}{n_{20} + n_{02}} < A$$

y

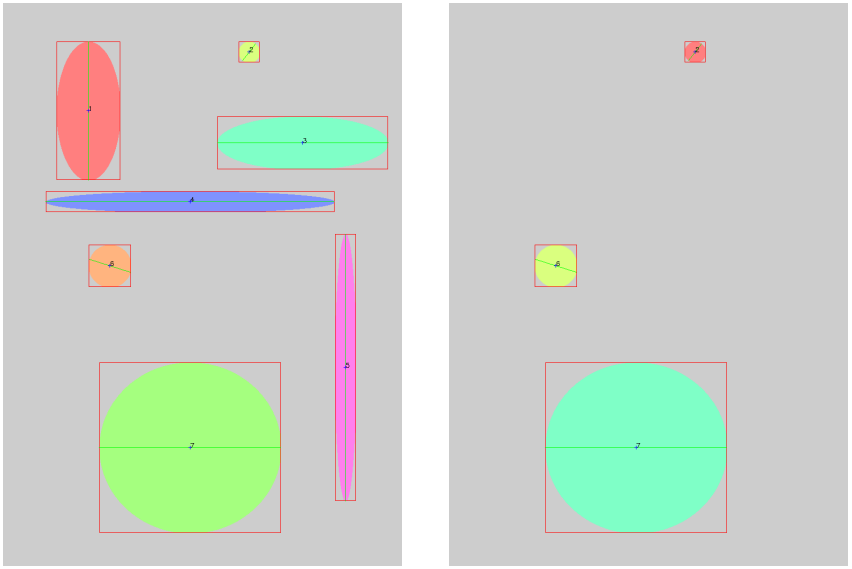
$$\frac{n_{20} - n_{02}}{n_{20} + n_{02}} < B$$

donde A y B son constantes, que cuanto más cercanas a 0 sean más preciso harán el filtro. En la Figura 5.15 se observa un ejemplo para las constantes $A = 0{,}1$ y $B = 0{,}3$.

Es importante destacar que si este filtro se realiza de forma muy selectiva (esto es, darle un valor muy pequeño a las constantes A y B), no será posible detectar algunos marcadores ya que debido a varios factores -entre los que se encuentran: características de la captura, tamaño de los marcadores en la imagen, iluminación, ruido, etc.- en la mayoría de los cuadros los marcadores no son círculos perfectos. Por esta misma razón se tuvo que descartar la utilización de un detector de círculos, cómo por ejemplo la *transformada de Hough* [51], para la implementación de este bloque.

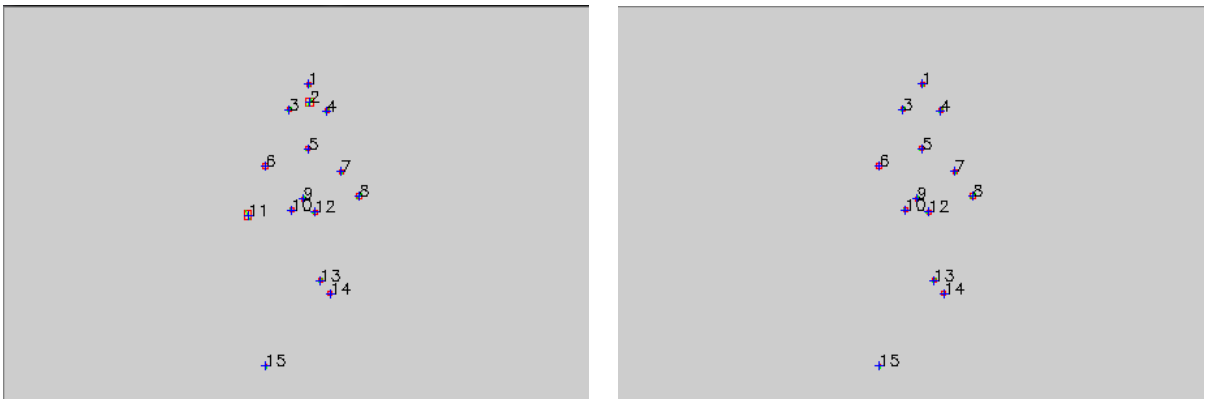
En la Figura 5.16b se ve el resultado de procesar la Figura 5.16a con el bloque *Circular Filter*.

0 y 1, para una parábola valdrá 1 y para una hipérbola será mayor a 1 [50].



(a) Entrada - Filtro circular. (b) Salida - Filtro Circular.

Figura 5.15: Entrada y salida del bloque *filtro circular* para las constantes $A = 0{,}1$ y $B = 0{,}3$.



(a) Objetos detectados - Salida del bloque *Detect blobs*. (b) Círculos detectados - Salida del bloque *Circular Filter*.

Figura 5.16: Resultado de procesar la imagen 5.13b con los bloques de detección de blobs y el filtro circular.

Luego del filtro circular, se identifican los blobs que resultaron del filtrado mediante el bloque *Number*, que utiliza la función *cvPutText* de la librería *OpenCV* para numerar cada blob de la imagen con su número de *id*. Esta numeración se hace de izquierda a derecha y de arriba hacia abajo.

Finalmente, volviendo al diagrama de flujo del bloque general (ver Figura 5.11), el bloque *Write xml* escribe la posición (x, y) de cada blob en el cuadro procesado, en un archivo XML [52]. Este bloque es implementado mediante funciones de C++

para escribir archivos, teniendo en cuenta la estructura de los archivos XML.

En la Figura 5.17, se presenta un ejemplo del archivo de salida del bloque *Write xml* que corresponde también a la salida del bloque de segmentación. Como se observa, es el resultado de analizar una secuencia de tres cuadros (frames 0,1 y 2) donde se detectan tres marcadores en cada uno de ellos.

```
<?xml version="1.0" encoding="UTF-8"?>
<Detected_Markers Version="1">
    <Frame id="0" >
        <Marker id="1" >
            <Centroid x="211.588235" y="179.882353" />
        </Marker>
        <Marker id="2" >
            <Centroid x="131.615385" y="185.615385" />
        </Marker>
        <Marker id="3" >
            <Centroid x="155.000000" y="185.500000" />
        </Marker>
    </Frame>
    <Frame id="1" >
        <Marker id="1" >
            <Centroid x="211.588235" y="179.882353" />
        </Marker>
        <Marker id="2" >
            <Centroid x="131.615385" y="185.615385" />
        </Marker>
        <Marker id="3" >
            <Centroid x="155.000000" y="185.500000" />
        </Marker>
    </Frame>
    <Frame id="2" >
        <Marker id="1" >
            <Centroid x="131.615385" y="185.615385" />
        </Marker>
        <Marker id="2" >
            <Centroid x="154.714286" y="185.500000" />
        </Marker>
        <Marker id="3" >
            <Centroid x="177.500000" y="185.500000" />
        </Marker>
    </Frame>
</Detected_Markers>
```

Figura 5.17: Salida del bloque *Segmentación*.

Se eligió el formato XML para exportar los resultados porque es un formato conocido universalmente y fácil de importar en cualquier lenguaje, en particular *Matlab* contiene librerías para trabajar con el mismo.

Resta comentar que, además de los videos a procesar, el algoritmo tiene como entradas opcionales un conjunto de argumentos que establecen distintos parámetros para el procesamiento. Estos argumentos se presentan a continuación:

- **t <valor>**, establece un umbral fijo para la umbralización. Debe ser un valor entre 0 y 255.

- **A <valor>**, área máxima para el filtro por área. Debe ser un valor positivo.
- **a <valor>**, área mínima para el filtro por área. Debe ser un valor positivo.
- **s**, guarda los videos que resultan de la salida de los bloques de umbralización y detección de blobs.

5.6. Resultados y análisis

Los resultados para cada marcador se pueden clasificar en 4 tipos:

- *Verdadero positivo*. Se detecta un marcador donde hay un marcador realmente.
- *Verdadero negativo*. No se detecta un marcador donde no hay un marcador en el cuadro original.
- *Falso positivo*. Se detecta un marcador donde en realidad no hay uno.
- *Falso negativo*. No se detecta marcador donde hay uno realmente.

A raíz de esto, se puede concluir que es deseable tener tanto verdaderos positivos como verdaderos negativos. El falso negativo y el falso positivo corresponden a errores que se deben evitar.

En el desarrollo de sistemas de este tipo, normalmente se priorizan algunos errores frente a otros y se optimiza el sistema para reducir la probabilidad de que ocurran los más prioritarios. Uno de los errores más importantes en segmentación es el *falso negativo*, lo que se traduce en una pérdida de un marcador, ya que si se pierde por muchos cuadros consecutivos es imposible recuperar su trayectoria.

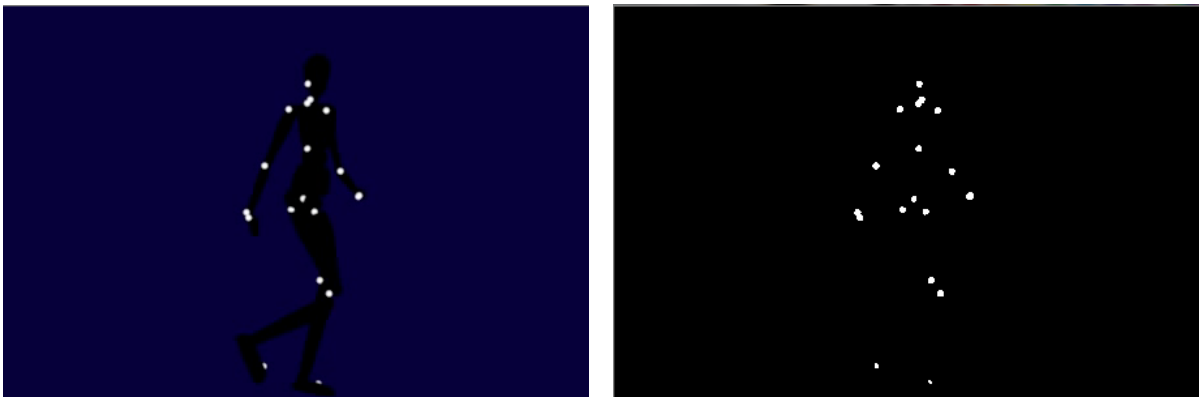
Otro de los errores a priorizar es el cálculo del centroide de un marcador de forma incorrecta. Esto en general sucede porque al segmentar es complicado detectar el total de los píxeles en un marcador debido a la mala iluminación, calidad de la filmación, etc.

A continuación, se presentan los resultados de distintas pruebas realizadas en los bloques más críticos que integran la detección de marcadores.

5.6.1. Umbralización

Para el bloque de umbralización de este sistema se observó que el resultado obtenido, como era de esperarse, depende fuertemente de las condiciones de captura y obviamente del umbral calculado.

Para el caso sintético, como el mostrado en la Figura 5.18, donde se imponen las condiciones de captura de manera deseable, los resultados fueron muy buenos. Esto es razonable ya que como se vio anteriormente, el método de Otsu calcula el umbral óptimo, por lo que mientras en la imagen se presenten los marcadores cómo los objetos más brillantes y se aprecie un alto contraste entre los mismos y el resto de los elementos (fondo, paciente, etc.), la segmentación será muy efectiva.



(a) Captura original de una secuencia sintética. (b) Imagen filtrada con el umbral de Otsu.

Figura 5.18: Entrada y salida del bloque de umbralización para una secuencia sintética.

Sin embargo, al probar con capturas reales los resultados cambian. En la Figura 5.19a se muestra un cuadro perteneciente a una captura real, cuyo resultado al pasar por el bloque de umbralización se observa en la Figura 5.19b.

(a) Captura original de una secuencia real. (b) Imagen filtrada con el umbral de Otsu.

Figura 5.19: Entrada y salida del bloque umbralización para un caso real.

Se puede observar que los objetos resultantes de la segmentación no solamente corresponden a los marcadores, sino que también se presentan otro tipo de objetos de igual o mayor intensidad luminosa. De todos modos, en principio esto no sería un problema, ya que con los filtros que se aplican a continuación de este bloque se podría eliminar el resto de los objetos.

Otra alternativa para eliminar los objetos detectados que no son marcadores, directamente en esta etapa, es eliminar el fondo mediante un extractor de fondo. En la Figura 5.20a se muestra el cuadro de la Figura 5.19a luego de extraerle el fondo

con un programa de la base de datos *Humaneva* [53] implementado en *Matlab*. En la Figura 5.20b se observan los resultados de aplicarle umbralización a dicha imagen.

(a) Captura con el fondo extraído. (b) Video Filtrado.

Figura 5.20: Segmentación para un caso real sin fondo.

Se observa que, si bien se siguen detectando objetos que no corresponden a los marcadores, el ruido en la imagen filtrada es mucho menor y los marcadores están mejor detectados que en la Figura 5.19b.

Es de destacar que los marcadores detectados en las figuras 5.19b y 5.20b no tienen el mismo tamaño. Esto se debe a que sus imagenes originales (5.19a y 5.20a respectivamente) son distintas entre ellas y por lo tanto el cálculo del umbral va a dar resultados distintos, esto provoca que en la imagen sin fondo se detecten más pixeles que en la imagen original. Dada la geometría de los marcadores, los pixeles más oscuros se encuentran sobre los bordes del marcador, y se aclaran al acercarse al centro.

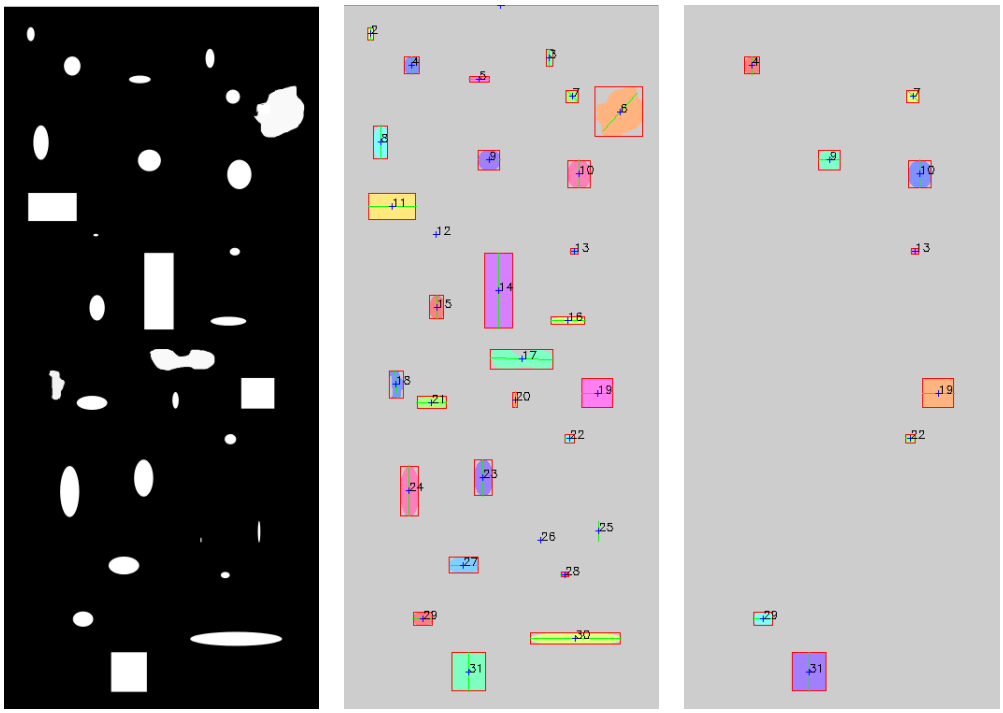
5.6.2. Detección de marcadores

Como se mencionó en la explicación del algoritmo, tanto la función para detectar objetos cómo la función para filtrar los mismos por su tamaño son funciones pertenecientes a la librería *OpenCV*, por lo que no sería necesario probarlas individualmente ya que las mismas están probadas y optimizadas por sus creadores. En consecuencia, para la detección de marcadores será necesario realizar pruebas más que nada en el filtro circular.

En la Figura 5.15 se muestra un ejemplo de funcionamiento de este filtro, dónde se puede ver que a grandes rasgos el filtro funciona correctamente.

En la Figura 5.21 se muestra un filtrado más complejo. Se puede observar que para las constantes $A = 0{,}3$ y $B = 0{,}1$ se detectan los círculos correctamente y se incluyen objetos - como el número 19, 29 o 31 - que si bien no son círculos,

podrían ser marcadores que por distintos factores (como ser iluminación, calidad de la cámara, ruido, etc.) quedaron segmentados con otra forma. Por otro lado, hay objetos en la Figura 5.21b (por ejemplo el número 28), que también podrían ser un marcador mal detectado, pero no es incluido en la Figura 5.21c.



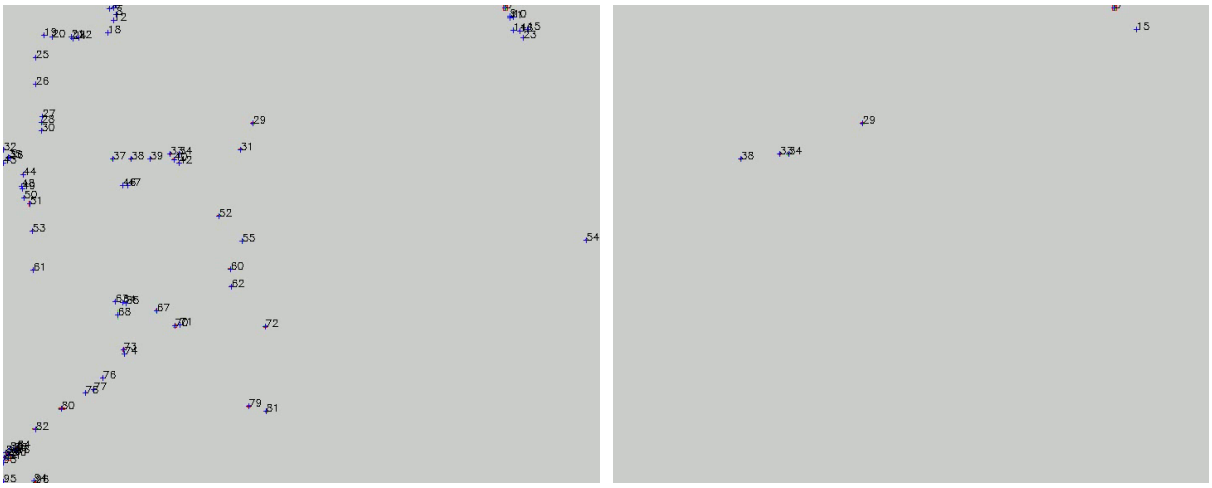
(a) Imagen de prueba. (b) Blobs detectados. (c) Circulos detectados.

Figura 5.21: Ejemplo de detección de blobs y filtro circular con $A = 0{,}3$, $B = 0{,}1$ para una imagen de prueba.

Algo parecido sucede con la detección de marcadores en la captura mostrada en la Figura 5.19. Cómo se comentaba anteriormente, en la salida de la umbralización se segmentaron otros objetos además de los marcadores, que se presentan como ruido en la imagen. Para eliminar los objetos más grandes que los marcadores se aplicó el filtro de área estableciendo un área máxima de 50 píxeles (ver Figura 5.22a). Por otro lado, en este caso no es recomendable aplicar un filtro de área mínima ya que los marcadores segmentados resultaron ser muy pequeños, por lo que de establecer esta cota inferior probablemente se eliminarán algunos. Lo mismo pasa con la aplicación de difuminado (o *smoothing*): si bien este proceso es muy efectivo para eliminar ruido en los bordes, resulta ser contraproducente cuando los marcadores son muy pequeños, ya que también los elimina.

Como se puede ver en la Figura 5.22b los resultados para esta captura no fueron muy buenos ya que, como los marcadores detectados presentan un área de unos pocos píxeles, el filtro circular no fue capaz de reconocerlos como círculos.

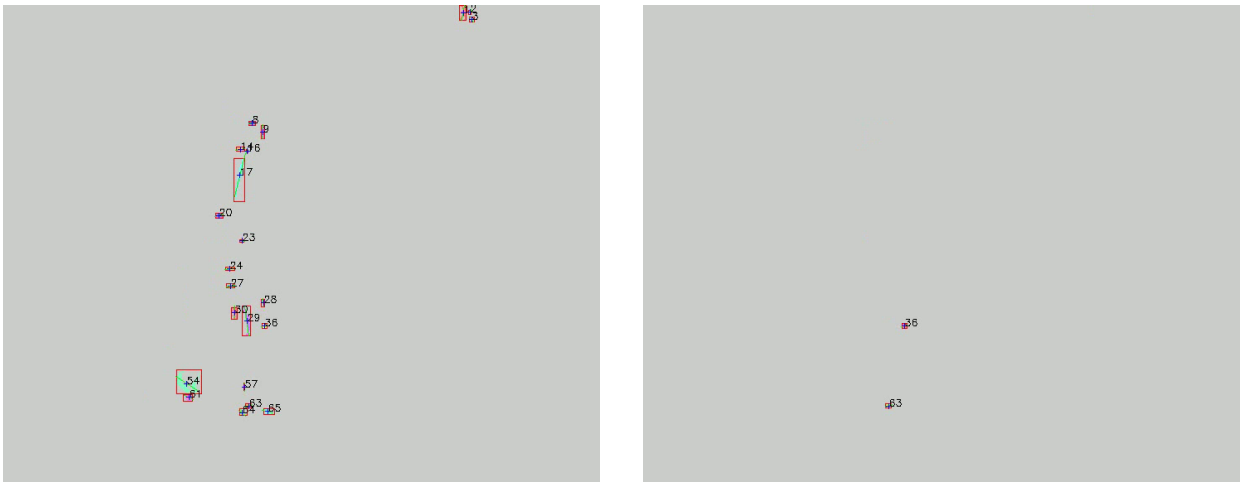
Por otro lado, para el caso real pero sin fondo (Figura 5.20), se puede ver en la Figura 5.23 que no se detecta tanto ruido como en el caso anterior. Sin embargo, los



(a) Blobs detectados. (b) Circulos detectados.

Figura 5.22: Detección de marcadores para un caso real, con $A = 0{,}3$, $B = 0{,}1$ y área máxima de 50 píxeles

círculos detectados en la Figura 5.23b vuelven a ser muy pocos, dejando marcadores fuera de la detección.



(a) Blobs detectados. (b) Círculos detectados.

Figura 5.23: Segmentación y detección de marcadores para un caso real sin fondo, con $A = 0{,}3$, $B = 0{,}1$ y área mínima de 10 píxeles

Como se comentaba anteriormente, en la Figura 5.20b se puede apreciar que los marcadores están segmentados con una superficie mayor a la de la Figura 5.19b. Esto da como indicio que para esta captura (con estas condiciones de iluminación, ambiente, paciente, etc.) el filtro circular con valores en sus constantes de $A = 0{,}3$ y $B = 0{,}1$ está siendo muy selectivo.

Aumentando estos valores a por ejemplo $A = 0{,}5$ y $B = 0{,}4$, se puede observar que la detección mejora, incluyendo más marcadores -*verdaderos positivos*- en la salida (ver Figura 5.24).

(a) Blobs detectados. (b) Circulos detectados.

Figura 5.24: Segmentación y detección de marcadores para el caso de la Figura 5.20a , con $A = 0{,}5$, $B = 0{,}4$, área mínima de 10 píxeles y área máxima de 50 píxeles

Para el caso sintético, hacer el análisis anterior no tiene el mismo provecho, ya que al obtenerse una segmentación muy buena los marcadores se detectaron en su totalidad (ver Figura 5.25a).

Por otro lado, en la Figura 5.25b puede verse que el filtro circular detecta todos los marcadores, a excepción de los que están superpuestos entre sí (como el 2 y el 11 de la Figura 5.25a), ya que al superponerse pierden la forma circular.

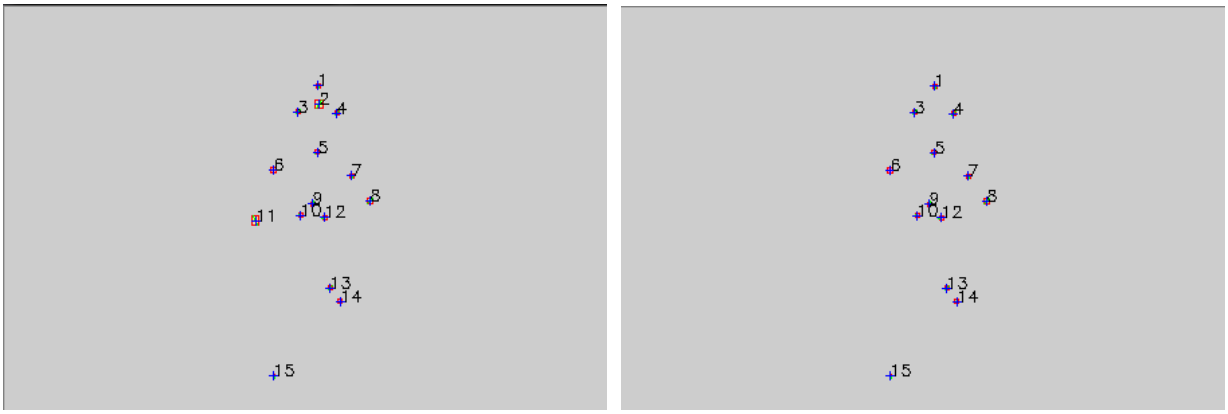
Queda cómo tarea pendiente para una etapa futura optimizar el bloque de segmentación para estos casos particulares, de forma tal de no perder los datos de los marcadores superpuestos. Una posible solución podría ser utilizar el método de erosión, para separar los marcadores y quedarse con la posición de sus centroides.

5.6.3. Medida de error

Como se mencionó anteriormente, una de las ventajas de tener una base de datos sintética, es poder tener una posición *ground truth* contra la cual comparar los resultados obtenidos con el sistema implementado. Esta comparación se realiza entre el centroide del objeto real y la posición *ground truth* del mismo objeto. A continuación se muestra un resumen del cálculo de error para el bloque de segmentación, el detalle del mismo se explicará en el Capítulo 9.

En la Figura 5.26, se observa la comparación de la segmentación de una secuencia contra su *ground truth* para un sujeto moviéndose de izquierda a derecha.

En la Tabla 5.1 se puede observar una medida de error para una secuencia sintética estándar de la base. En ella se observa, para cada cámara, el porcentaje de detección de marcadores en toda la captura, el error promedio en píxeles y el



(a) Blobs detectados. (b) Círculos detectados.

Figura 5.25: Resultado de procesar la imagen 5.18b con los bloques de detección de blobs y el filtro circular.

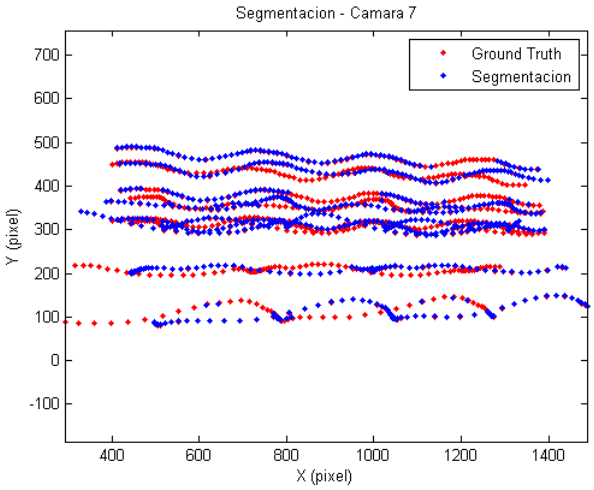


Figura 5.26: Segmentación de cámara 7 y ground truth.

percentil 99[11] también en píxeles. Se realizaron estas medidas para varias secuencias distintas y los valores se mantienen aproximadamente en este mismo rango.

El error promedio, cuyo cálculo se detalla en la Sección 9.1, se calcula como un promedio de la distancia entre el centroide de cada marcador **detectado** y la posición ground truth del mismo marcador en todos los cuadros. Por otro lado, el porcentaje de detección es calculado únicamente sobre los marcadores visibles, para todos los cuadros.

En la Figura 5.27 se observa un histograma que contempla datos de todas las cámaras a lo largo de la secuencia completa, donde se muestra la cantidad de marcadores por rango de error en píxeles. Observando la figura se puede ver que la curva tiene una media en el rango de 1 píxel de error.

[11]Valor del error en píxeles por debajo del cual se encuentra el 99 % de los marcadores.

Id de cámara	Porcentaje de detección	Error promedio (en píxeles)	Percentil 99 (en píxeles)
1	73.31 %	0.9121	2.2751
2	86.22 %	0.9477	3.6592
3	67.36 %	1.0278	3.4376
4	65.79 %	1.0343	3.5561
5	63.97 %	0.9965	2.529
6	63.85 %	0.9822	2.4508
7	59.59 %	1.0388	3.4561
8	57.21 %	1.0713	3.7559
9	59.09 %	1.0206	2.927
10	77.44 %	1.0441	3.3186
11	57.83 %	1.0316	3.8255
12	56.83 %	1.0561	4.3085
13	60.28 %	1.0121	2.6783
14	63.97 %	0.9931	2.536
15	64.91 %	0.9948	2.5073
16	64.6 %	0.9994	3.126
17	64.97 %	1.0094	2.9885

Tabla 5.1: Medida de error en captura estándar de la base de datos sintética.

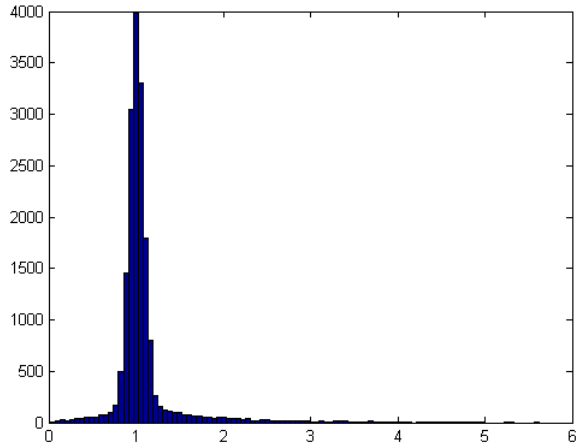


Figura 5.27: Histograma de cantidad de marcadores por rango de error en píxeles

Este cálculo es realizado para una secuencia de 110 cuadros, con 17 cámaras y 13 marcadores en el cuerpo del paciente. Sin embargo, el total de marcadores en el histograma no son $17 * 13 * 110 = 24310$ debido a que en cada cámara no se visualizan todos los marcadores. Para ver el total de los marcadores analizados hay que tener en cuenta el porcentaje de detección de cada cámara, esto da un total de 15671 marcadores.

Finalmente, a modo de conclusión, se puede decir que el bloque de segmentación y detección de marcadores funciona correctamente para secuencias sintéticas como las de la base de datos, teniendo errores promedio de 1 píxel y errores máximos de 4 píxeles. Estos errores se mantienen en el rango de los errores de los otros bloques que componen el sistema, por lo que podría decirse que, si bien se utilizaron métodos de segmentación sencillos y no tan robustos como otros métodos, los resultados se mantienen acordes a la performance general de esta primer versión del sistema.

Por otro lado, en las pruebas con secuencias reales no se dieron resultados tan buenos, sin embargo ajustando los parámetros de forma adecuada (esto es, elegir correctamente los valores de área máxima, área mínima y umbral) se puede ver que se logran detectar un número razonable de marcadores por cámara. Además, no hay que pasar por alto que no fue fácil conseguir capturas reales para probar este bloque, y las que se obtuvieron no se adaptan del todo a las hipótesis del problema, por lo que fue difícil realizar un verdadero testeo de este bloque.

Queda pendiente como posible mejora a futuro robustecer este bloque, por ejemplo teniendo en cuenta el color de las capturas y no solo los niveles de gris, de forma tal de poder detectar marcadores en otros contextos no tan amigables como las condiciones del laboratorio establecidas para esta primera etapa de desarrollo.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Calibración

6.1. Introducción

Para poder obtener la posición en tres dimensiones de los marcadores a partir de las imágenes capturadas, es necesario que las cámaras estén previamente calibradas. El objetivo de la calibración consiste en determinar un conjunto de parámetros tal que pueda establecerse una relación entre el espacio 3D y las coordenadas 2D de las imágenes.

Los puntos en el espacio pueden ubicarse respecto a un sistema de coordenadas 3D. A su vez, los puntos capturados en las imágenes pueden referenciarse respecto a un sistema de coordenadas 2D en píxeles. En la Figura 6.2 se observa un punto M en el espacio respecto a un sistema de coordenadas 3D y su correspondiente proyección en la cámara, el punto m, de coordenadas 2D. Si se quiere determinar la posición de un punto en el espacio en función de las correspondientes proyecciones de dicho punto en las imágenes capturadas por las cámaras, es necesario determinar las ecuaciones que vinculan al sistema de coordenadas del espacio con el sistema de coordenadas en píxeles de las cámaras.

De la relación entre estos sistemas de coordenadas se obtienen los parámetros de las cámaras. Dichos parámetros se clasifican en intrínsecos y extrínsecos. Los primeros son aquellos que describen las propiedades geométricas y ópticas de la cámara, es decir, las características internas de la cámara. Por otra parte, los parámetros extrínsecos son los que describen la posición y orientación de la cámara respecto al sistema de coordenadas del espacio.

Para realizar la calibración es necesario establecer un modelo que describa el sistema óptico de las cámaras. Esto es, el modelo por el cual una cámara es capaz de transformar el espacio 3D en imágenes de dos dimensiones. Un modelo simple y que describe estos sistemas adecuadamente es el modelo *pinhole*.

Este modelo se basa en la implementación mas simple de una cámara real, la

cámara estenopeica. En dicha cámara, la imagen capturada esta conformada por la proyección del espacio 3D a través de un orificio situado delante de la pantalla o retina, que representan el material fotosensible de la cámara. Como se muestra en la Figura 6.1, la imagen capturada se proyecta en la retina en forma invertida. Por otro lado, el modelo de la cámara *pinhole* se describe en la Figura 6.2.

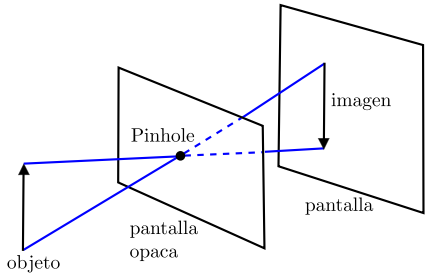


Figura 6.1: Cámara estenopeica. Imagen extraída de [54].

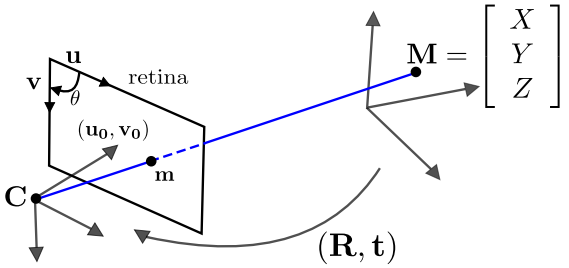


Figura 6.2: Modelo *pinhole* de una cámara, con el que se representa la proyección de un punto en el espacio M sobre la cámara. Imagen extraída de [55].

En dicho modelo, una cámara está representada por un punto C, foco de la cámara, y un plano, al que se le llama retina de la cámara. La imagen que se proyecta en la retina corresponde a la imagen capturada por la cámara. Dado un punto M en el espacio, su correspondiente proyección en la retina, el punto m, se encuentra en la intersección de la retina y la recta formada por los puntos C y M de manera que C, M y m son colineales.

6.2. Matriz de Proyección

En esta sección se sigue lo explicado en [55]. Sean las coordenadas del punto M y las del punto m las siguientes:

$$m = \begin{bmatrix} u \\ v \end{bmatrix}, \quad M = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Notando con $\tilde{x}$ a los vectores en coordenadas homogéneas se tiene:

$$\tilde{m} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \quad \tilde{M} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Tomando el modelo *pinhole* de la cámara, la relación entre un punto M y su proyección m es:

$$s\tilde{m} = A[R \quad t]\tilde{M} \tag{6.1}$$

$$\text{siendo } A = \begin{bmatrix} \alpha & c & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

Por lo tanto estos puntos se relacionan a través de la matriz $P = A[R \quad t]$, a menos de un factor de escala s. A dicha matriz se le denomina *Matriz de Proyección* de la cámara.

La matriz P se compone a su vez de la matriz A que representa los parámetros intrínsecos de la cámara, y la matriz $[R \quad t]$ que representa los parámetros extrínsecos.

La matriz $[R \quad t]$ está formada por la rotación y traslación que relaciona el sistema de coordenadas del espacio con el sistema de coordenadas de la cámara. La matriz A está formada por los parámetros intrínsecos de la cámara:

(u_0, v_0)	coordenadas del punto principal.
α y β	factores de escala en los ejes de imagen u y v.
c	grado de oblicuidad de los ejes imagen

El punto principal (u_0, v_0) se define como el punto formado por la intersección de la retina y la recta perpendicular a dicha retina que pasa por el punto C. La distancia entre el punto C y el punto principal se define como la distancia focal de la cámara f. Los factores α y β se relacionan con la relación de aspecto de los píxeles de la cámara. El parámetro c se relaciona con el ángulo θ formado por los ejes u y v.

Por lo tanto, los pasos para efectuar la proyección de un punto 3D sobre la retina de una cámara son los siguientes:

- Se pasa del sistema de coordenadas del espacio 3D (X_w, Y_w, Z_w) al sistema de coordenadas de la cámara (X, Y, Z)

$$[X \ Y \ Z]^T = \mathbf{R}.[X_w \ Y_w \ Z_w]^T + \mathbf{t}$$

- Se proyecta el punto 3D respecto a las coordenadas de la cámara sobre las coordenadas imagen de la retina (x, y).

$$x = f\frac{X}{Z} \qquad y = f\frac{Y}{Z}$$

- En algunos casos puede ser necesario modelar además ciertas distorsiones introducidas por el lente de la cámara.

$$\breve{x} = x + \delta_x \qquad \breve{y} = y + \delta_y$$

Siendo $(\breve{x}, \breve{y})$ las coordenadas distorsionadas y (δ_x, δ_y) las distorsiones aplicadas a (x, y).

- Por último se pasa de la coordenadas imagen $(\check{x}, \check{y})$ a las coordenadas en píxeles $(\check{u}, \check{v})$.

$$\check{u} = d_x^{-1}\check{x} + u_0 \qquad \check{v} = d_y^{-1}\check{y} + v_0$$

  Siendo d_x y d_y las distancias entre píxeles adyacentes en las direcciones horizontal y vertical, respectivamente.

6.3. Métodos de calibración

De acuerdo a los objetos utilizados para realizar la calibración, los métodos pueden clasificarse como se describe en [55]:

- **Calibración mediante objetos 3D**

  La calibración mediante este método es realizada capturando la imagen de un objeto de calibración cuyas dimensiones y geometría son conocidas. Los objetos de calibración suelen ser planos colocados ortogonalmente. También pueden utilizarse estructuras con marcadores de dimensiones conocidas. A estos objetos se les aplica traslaciones en el espacio logrando una mayor cantidad de puntos de referencia para calibrar. La ventaja de este método es su precisión aunque se requiere de objetos más costosos y procedimientos más elaborados.

- **Calibración mediante objetos 2D**

  En este caso se utilizan objetos planos con figuras de patrones determinados, por ejemplo dameros. Estos objetos son colocados en varias posiciones delante de la cámara de manera de capturar varias imágenes del objeto. Esta metodología ofrece más flexibilidad para calibrar.

- **Auto calibración**

  Este método consiste en obtener la información necesaria a través de la captura de varias imágenes de un escena estática, prescindiendo de objetos para calibrar. Esta método es aún más flexible que los anteriores aunque suele ser menos preciso.

Algunas soluciones comerciales, como por ejemplo *Vicon* [2], utilizan como objeto de calibración una vara con leds (ver Figura 6.3).

En este caso la calibración se realiza moviendo el objeto de calibración a través del espacio de trabajo. Este método resulta muy flexible para la calibración de un conjunto de varias cámaras simultáneamente.

6.4. Calibración en el entorno Blender

En el capítulo 3 se describió la metodología con la que se simuló un laboratorio real utilizando el programa de modelado 3D *Blender* y con el cual, a su vez, se simularon secuencias de marcha. Una vez que se tiene establecida la configuración

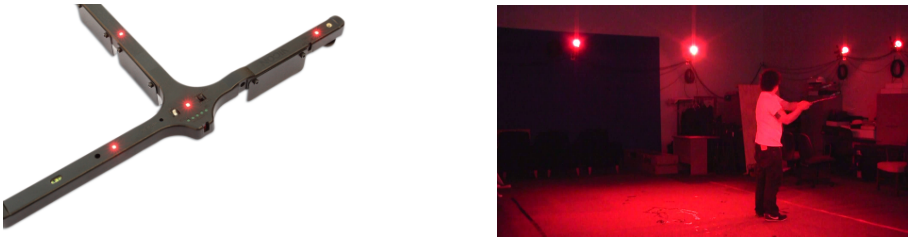


Figura 6.3: Calibración Vicon

de las cámaras, es posible conocer las matrices de proyección de cada una de estas cámaras a través de la información proporcionada por el propio programa. De esta manera se puede obtener la calibración *real* de las cámaras. Esto permite medir el desempeño de los métodos de calibración que puedan ser simulados en *Blender*. A su vez, al tener la calibración *real*, se puede evaluar el desempeño de otros bloques del sistema aislando el error que pueda aportar el bloque de calibración al error final del sistema.

Si se conocen las coordenadas de un punto 3D en el espacio de trabajo de *Blender* y además se poseen las matrices de proyección de las cámaras, es posible determinar las coordenadas 2D de dicho punto proyectado en cada una de las cámaras. De esta manera se obtiene el *ground truth* de la detección de marcadores. Teniendo estos datos se pueden medir el desempeño de los bloques siguientes del sistema. Por ejemplo, es posible evaluar de manera aislada el desempeño del bloque de segmentación sabiendo la posición 2D real de los marcadores, sin introducir errores provenientes de otras etapas del sistema.

A continuación se describe cómo se obtienen las matrices de proyección a partir de ciertos parámetros proporcionados por el entorno *Blender*.

La posición de una cámara se establece respecto a un sistema de coordenadas determinado, al que se le puede llamar el sistema de coordenadas del mundo. Por lo cual es posible saber la rotación y traslación que es necesario aplicar al sistema de coordenadas del mundo para obtener el sistema de coordenadas solidario a la cámara. Los parámetros que se pueden obtener de *Blender* son los ángulos de rotación (θ, ϕ, ψ) respecto a los ejes (x, y, z) del sistema de coordenadas del mundo, así como el orden en que se ejecutan dichas rotaciones. Si el orden establecido es *XYZ Euler*, lo que implica que primero se rota respecto al eje x, luego al y y al z, se obtiene la matriz de rotación:

$$R = R_{(z,\psi)} R_{(y,\phi)} R_{(x,\theta)} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ R_3 \end{pmatrix}$$

Donde,

$$R_{(x,\theta)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sen\,\theta \\ 0 & sen\,\theta & cos\,\theta \end{pmatrix}, \; R_{(y,\phi)} = \begin{pmatrix} cos\,\phi & 0 & sen\,\phi \\ 0 & 1 & 0 \\ -sen\,\phi & 0 & cos\,\phi \end{pmatrix}, \; R_{(z,\psi)} \begin{pmatrix} cos\,\psi & -sen\,\psi & 0 \\ sen\,\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

A su vez también es posible obtener el vector de traslación T que es necesario aplicar al sistema de coordenadas del mundo para obtener el sistema solidario a la cámara. De esta manera se obtiene la matriz de parámetros extrínsecos:

$$M_{ext} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & -R_1^t T \\ r_{21} & r_{22} & r_{23} & -R_2^t T \\ r_{31} & r_{32} & r_{33} & -R_3^t T \end{pmatrix}$$

Para hallar la matriz de parámetros intrínsecos se necesitan saber los siguientes parámetros que son proporcionados por *Blender*:

- f, distancia focal.
- (o_x, o_y), coordenadas del punto principal.
- (s_x, s_y), tamaño efectivo de los píxeles de la retina en píxeles/milímetro.

utilizando estos parámetros la matriz de parámetros intrínsecos resultante es:

$$M_{int} = \begin{pmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{pmatrix}$$

6.5. Calibración para secuencias reales

Se ha podido tener acceso a secuencias de video de análisis del movimiento de la marcha que se han realizado en el Hospital de Clínicas. Dichas secuencias han sido obtenidas por un grupo de médicos e investigadores con el objetivo de analizar el movimiento de pacientes en el ámbito terapéutico o con un objetivo esencialmente de investigación desde el punto de vista de la biomecánica.

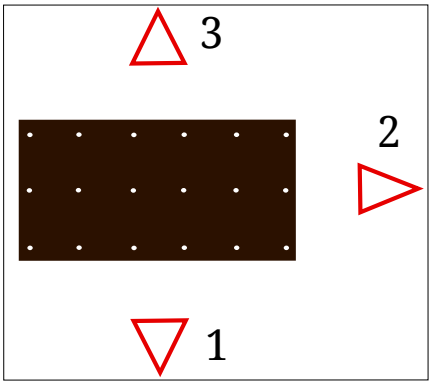


Figura 6.4: Laboratorio del Hospital de Clínicas.

La metodología utilizada consiste en el uso de tres cámaras de video convencionales de 25 fps y resolución 720 x 576 píxeles. Dichas cámaras se disponen como se muestra en la Figura 6.4, alrededor de una alfombra o cinta sobre la que se le pide al paciente que camine.

La persona que se desea evaluar debe tener colocados marcadores, fundamentalmente en las articulaciones del cuerpo, y debe desplazarse sobre la alfombra dispuesta para esto. En la Figura 6.5a, se observa a un individuo caminando con los marcadores colocados.

(a) Persona con marcadores (b) Objeto de calibración

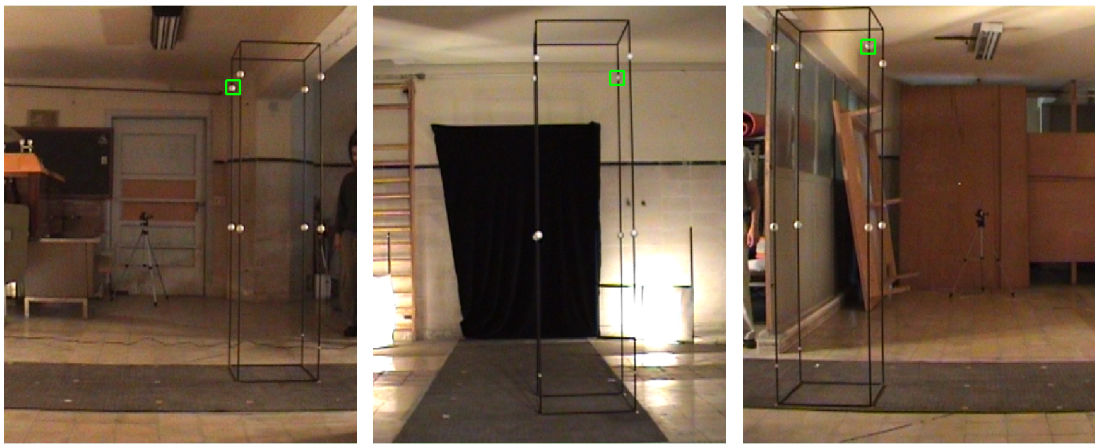
Figura 6.5: Captura de secuencia real

Antes de la captura del movimiento se emite una señal acústica con la cual se sincronizan las tres vistas una vez obtenidos los videos. El software con el cual este grupo de médicos ha analizado el movimiento es el *Dvideow*. Dicho software realiza la detección, seguimiento y reconstrucción de los marcadores. Por otro lado, no se ha podido tener acceso a este software por lo cual tampoco se pudo evaluar su desempeño y compararlo con el desarrollado en este proyecto.

El método que utilizaron para calibrar las cámaras requiere de un objeto de calibración como el que se muestra en la Figura 6.5b, al que se le llama *fantoma*. Este objeto se compone de marcadores colocados sobre una estructura cuyas dimensiones son conocidas. Dicho objeto se lo coloca en posiciones determinadas que se encuentran marcadas sobre la alfombra, ver Figura 6.5b. De esta manera se obtiene un mayor número de puntos ubicados en el volumen de trabajo.

Los videos proporcionados por los especialistas del Hospital de Clínicas incluyen tanto las secuencias de marcha de diferentes individuos como el proceso con el que se calibran las cámaras del laboratorio. A partir de dichos videos se desarrolló un algoritmo tal que permite extraer las matrices de proyección de cada una de las cámaras. Tomando un cuadro del video en el que se encuentre el *fantoma* en una de las posiciones, el algoritmo implementado requiere que cada marcador del

fantoma, en cada una de las vistas, sea seleccionado manualmente, y en un orden tal que permita establecerse una correspondencia entre las proyecciones de un mismo marcador en las tres vistas. En la Figura 6.6 se muestra a uno de los marcadores seleccionado en cada una de las vistas.



(a) Vista cámara 1 (b) Vista cámara 2 (c) Vista cámara 3

Figura 6.6: Uno de los marcadores de calibrador seleccionado en cada una de las cámaras

Por otra parte la posición de los marcadores en el espacio 3D es conocida ya que se saben las medidas del *fantoma*, ver Figura 6.7. Dichas posiciones pueden ser referidas un punto que puede elegirse arbitrariamente, así como los ejes de coordenadas (x, y, z).

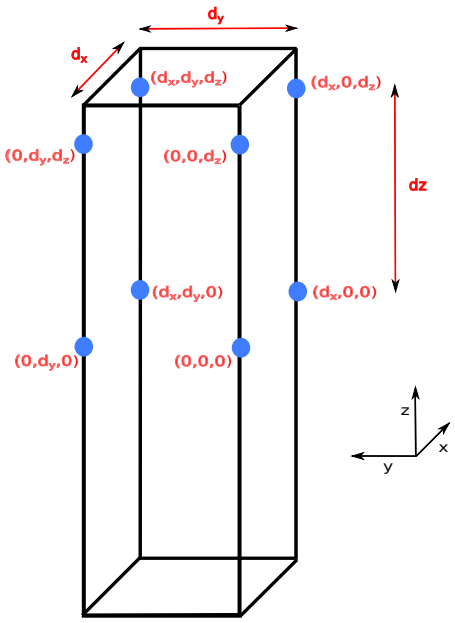


Figura 6.7: Coordenadas de los marcadores del *fantoma*

De esta forma se tiene asociadas las coordenadas 3D de los marcadores en el espacio con sus correspondientes coordenadas 2D en píxeles en cada una de las cámaras, $\mathbf{X_i} \leftrightarrow \mathbf{x_i}$. Si se tiene una cantidad suficiente de puntos, entonces las matrices de proyección P pueden ser estimadas tal que $\mathbf{x_i} = P\mathbf{X_i}$. Para esto se utiliza el algoritmo *Direct Linear Transformation (DLT)*.

Según lo demostrado en [56], para cada asociación de puntos $\mathbf{X_i} \leftrightarrow \mathbf{x_i}$ se cumple que :

$$\begin{pmatrix} 0^T & -w_i X_i^T & y_i X_i^T \\ w_i X_i^T & 0^T & -x_i X_i^T \end{pmatrix} \begin{pmatrix} P^1 \\ P^2 \\ P^3 \end{pmatrix} = 0$$

siendo P^{iT} las columnas i-ésimas de P y $\mathbf{x_i}$ de coordenadas homogéneas (x_i, y_i, w_i). Dicha matriz se obtiene resolviendo un conjunto de ecuaciones del tipo $Ap = 0$. Dado que por cada punto se tiene dos ecuaciones y que la matriz P tiene doce entradas y once grados de libertad, ignorando el factor de escala, resulta que son necesarios conocer al menos seis correspondencias $X_i \leftrightarrow x_i$.

Para anular los efectos de la selección arbitraria del origen y la escala del sistema de coordenadas se aplica una normalización tanto a los puntos imagen 2D como a los 3D, logrando de esta manera mejorar los resultados finales. Para los puntos 2D de cada vista se traslada el origen de coordenadas de dicha vista al centroide de los puntos y se aplica un escalado tal que la distancia promedio de los puntos al origen sea $\sqrt{2}$. Para los puntos 3D el mismo procedimiento excepto que el escalado que se aplica es tal que la distancia promedio al origen es $\sqrt{3}$. De esta manera se tiene dos matrices que realizan esta transformación, la matriz T_{3D} tal que $\tilde{X}_i = T_{3D} X_i$ para los puntos en el espacio, siendo $\tilde{X}_i$ los puntos normalizados. Análogamente para los 2D imagen se tiene la matriz T_{2D} tal que $\tilde{x_i} = T_{2D} x_i$.

Dado que las coordenadas de los puntos 2D están afectadas por el ruido y que se tienen más de seis correspondencias $X_i \leftrightarrow x_i$, no existe una solución exacta a las ecuaciones $Ap = 0$. Por lo tanto la solución se obtiene minimizando un error, en este caso se busca p tal que minimice $||Ap||$. Para esto se utiliza la descomposición en valores singulares (SVD), donde se obtiene del vector singular asociado al menor valor singular. De esta manera se obtiene la matriz de proyección $\tilde{P}$. Por último debe descomponerse la normalización, por lo tanto la matriz de proyección P queda $P = T_{2D}^{-1} \tilde{P} T_{3D}$.

6.6. Calibración simulada en *Blender*

Con el objetivo de establecer una metodología de calibración que fuera válida para la configuración de cámaras con las que se diseñó la base de datos elaborada en *Blender*, se probaron implementaciones existentes que pudieran calibrar dicha base. Para esto se evaluaron dos toolbox elaborados en *Matlab*:

- *Automatic Multi-Camera Calibration Toolbox (amcctoolbox) [57].*
- *Multi-Camera Self-Calibration Toolbox [58].*

La metodología de calibración fue simulada en *Blender* y las imágenes obtenidas como resultado se procesaron con los toolbox mencionados.

6.6.1. Automatic Multi-Camera Calibration Toolbox (amcctoolbox)

Este toolbox implementado en *Matlab*, automatiza ciertas funciones del toolbox *Camera Calibration Toolbox*[1], también implementado en *Matlab*. De esta manera se omite toda intervención del usuario que es necesaria en este último toolbox, facilitando la calibración de un conjunto de varias cámaras.

El método implementado en el *amcctoolbox* es uno de los más utilizados para la calibración de cámaras. Dicho método utiliza un objeto de calibración de dos dimensiones con figuras de patrones determinados. En este caso la figura que se utiliza es la de un damero como el que se muestra en la Figura 6.8. Para la calibración se toma como dato conocido el largo y ancho de los cuadrados blancos y negros que componen el damero.

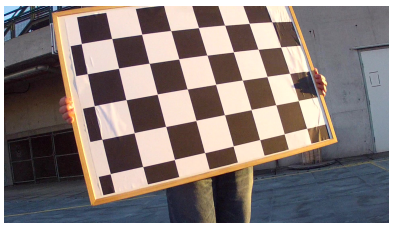


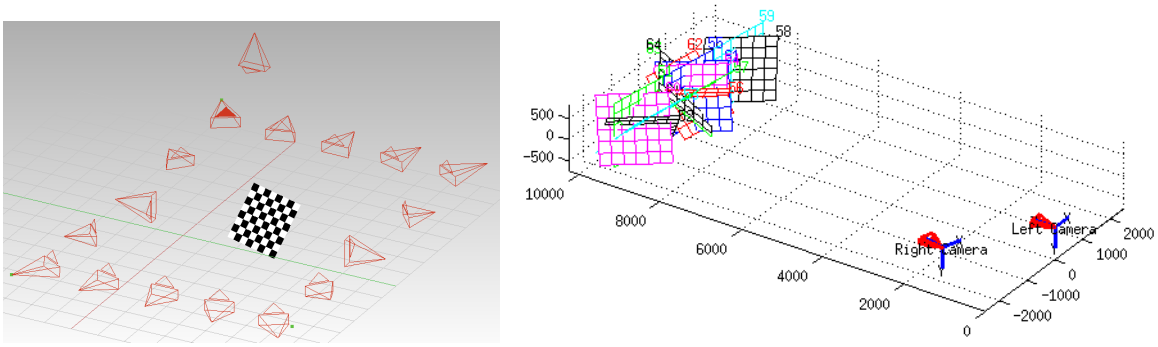
Figura 6.8: Objeto de calibración utilizado en el *amcctoolbox*

El procedimiento para calibrar una cámara consiste en capturar imágenes del damero colocado en distintas orientaciones, al menos dos, como se menciona en [59]. Dichas orientaciones del damero relativas a la cámara no necesariamente deben conocerse.

Para la calibración de conjuntos de varias cámaras el damero debe ser colocado en múltiples ubicaciones y orientaciones tal que cada cámara capture algunas de las mismas, al tiempo que en cada una de estas posiciones el damero pueda ser visto por más de una cámara simultáneamente. Para que esto pueda lograrse debe existir una suficiente superposición de los ángulos de visión entre cámaras adyacentes. Por otra parte, para una correcta calibración el damero debe ser colocado en distintas ubicaciones tales que se logre cubrir el mayor volumen de trabajo.

[1] http://www.vision.caltech.edu/bouguetj/ . Accedido 3-12-14.

Para calibrar el conjunto de cámaras del laboratorio virtual elaborado en *Blender*, descrito en la sección 3, se simula un damero real como se muestra en la Figura 6.9a. Para cada par de cámaras adyacentes se coloca el damero en algún punto dentro del volumen de trabajo, con una orientación tal que la figura plana del damero sea visible en ambas cámaras. Luego, a partir de dicha posición, se toman capturas del damero cambiando ligeramente la ubicación y orientación del mismo. Para cada par de cámaras adyacentes se consideran entre 15 y 20 posiciones distintas del damero. La simulación se realiza a través de un código Python implementado en *Blender* para tal fin. En la Figura 6.10 se muestran algunas de las capturas realizadas por una de las cámaras.



(a) Damero en el entorno *Blender* (b) Calibración estéreo

Figura 6.9: Simulación del damero en *Blender* y calibración estéreo de un par de cámaras adyacentes

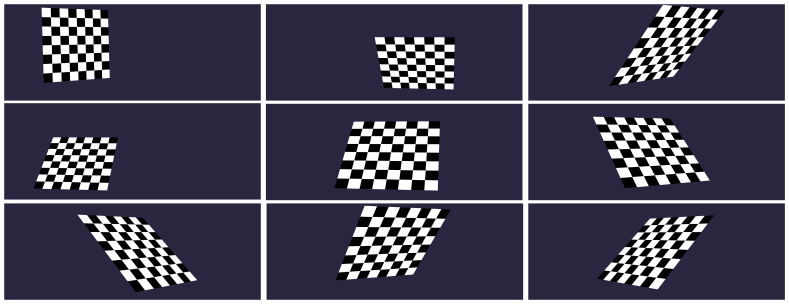


Figura 6.10: Capturas del damero en distintas posiciones y orientaciones realizadas por una de las cámaras.

Una vez realizadas las capturas se procesan las imágenes mediante el toolbox, obteniéndose los parámetros intrínsecos de cada una de las cámaras y la posición relativa de cada uno de los pares de cámaras adyacentes. Si se tiene un par de cámaras, se define a una de ellas como la cámara izquierda y a la otra como la cámara derecha. Luego de realizada la calibración se obtiene para cada par de cámaras la matriz de rotación R y el vector de traslación T tal que si las proyecciones de un

punto P en el espacio 3D son los puntos x_L en la cámara izquierda y x_R en la cámara derecha, la relación entre dichos puntos es

$$x_R = R * x_L + T$$

De esta forma se obtiene también la rotación y traslación que relacionan el sistema de coordenadas de una de las cámaras con el de la cámara adyacente a esta. A la calibración particular de un sistema de dos cámaras se le denomina calibración estéreo. En la Figura 6.9b se muestra el resultado de la calibración estéreo para un par de cámaras del laboratorio virtual. En dicha figura se observa además, la reconstrucción de cada una de las posiciones del damero consideradas para ese par de cámaras.

Una vez obtenidas las posiciones relativas de las cámaras adyacentes es posible hallar la posición relativa de cada una de las cámaras en el espacio. En la Figura 6.11 se muestra la posición relativa de cuatro de las cámaras.

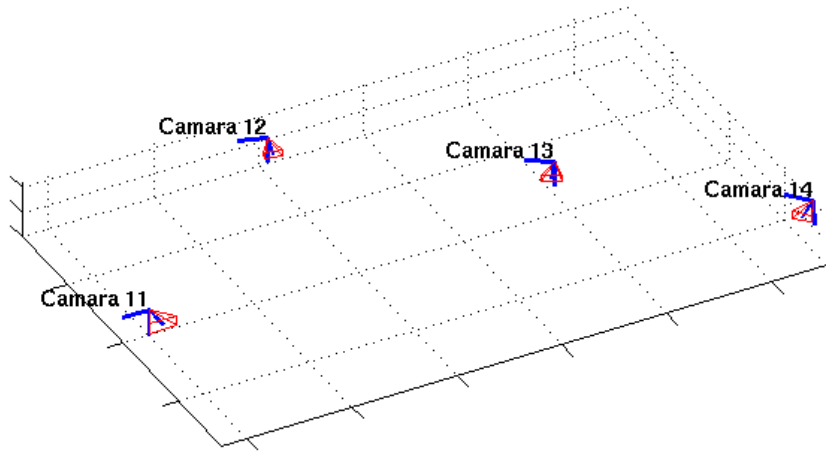


Figura 6.11: Reconstrucción de la posición de las cámaras mediante el toolbox

6.6.2. Multi-Camera Self-Calibration Toolbox

Este toolbox está especialmente pensado para la calibración simultánea de un conjunto de varias cámara. El procedimiento consiste en capturar con todas las cámaras, el movimiento de una fuente puntual de luz recorriendo todo el volumen de trabajo. Por tanto para cada cuadro se tiene un punto 3D en el espacio en una posición distinta y en cada una de las cámaras su correspondiente proyección si dicho punto es visible desde esa cámara. Para esto debe existir un contraste suficiente de la fuente puntual de luz respecto al laboratorio. Puede usarse, por ejemplo, una lámpara led y debe haber poca o nula luz ambiente en el laboratorio.

Si se tienen m cámaras y n puntos 3D $\mathbf{X_j} = [X_j, Y_j, Z_j, 1]^T$ con $j = 1, \ldots, n$, la proyección de dichos puntos en los puntos imagen u_j^i queda:

$$\lambda_j^i \begin{bmatrix} u_j^i \\ v_j^i \\ 1 \end{bmatrix} = \lambda_j^i u_j^i = P^i \mathbf{X_j}$$

siendo P^i la matriz de proyección de la i-ésima cámara y u, v las coordenadas en píxeles de los puntos imagen. Las coordenadas de dichas proyecciones deben ser detectadas para cada cuadro y para cada cámara. El objetivo de la calibración es hallar los factores de escala λ_j^i y las matrices de proyección P^i. Pueden expresarse todos los puntos y las matrices de proyección en una sola matriz W_s:

$$W_s = \begin{bmatrix} \lambda_1^1 \begin{bmatrix} u_1^1 \\ v_1^1 \\ 1 \end{bmatrix} & \ldots & \lambda_n^1 \begin{bmatrix} u_n^1 \\ v_n^1 \\ 1 \end{bmatrix} \\ \vdots & \ddots & \vdots \\ \lambda_1^m \begin{bmatrix} u_1^m \\ v_1^m \\ 1 \end{bmatrix} & \ldots & \lambda_n^m \begin{bmatrix} u_n^m \\ v_n^m \\ 1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} P^1 \\ \vdots \\ P^m \end{bmatrix}_{3m\text{x}4} [\mathbf{X_1} \ldots \mathbf{X_n}]_{4\text{x}n}$$

Por lo tanto se puede expresar:

$$W_s = PX$$

donde $P = [P^1 \ldots P^m]^T$ y $X = [\mathbf{X_1} \ldots \mathbf{X_n}]$

Si son detectados una cantidad suficiente de puntos no ruidosos (u_j^i, v_j^i) y se conocen los λ_j^i, entonces W_s tiene rango 4 y se puede factorizar en P y X.

El número mínimo de cámaras para una correcta calibración depende del número de parámetros que se conocen para cada una de las cámaras o del número de parámetros que no se conocen pero que son iguales en todas las cámaras. Por ejemplo, tres cámaras son suficientes si se conocen todos los puntos principales o si los parámetros internos de las cámaras no se conocen pero son los mismos para todas ellas.

La simulación en *Blender* se logra creando un punto 3D que toma para cada cuadro, distintas posiciones en forma aleatoria dentro del volumen de trabajo. Para cada cuadro se *renderiza* su posición en las 17 cámaras. En este caso se han tomado 500 posiciones distintas. En la Figura 6.12 se muestra en *Blender* la distintas posiciones que toma en un punto dentro del volumen de trabajo. La simulación es implementada en lenguaje Python.

En la Figura 6.13a se muestra la configuración de las 17 cámaras en el espacio 3D de *Blender*. En la Figura 6.13b se muestra, con círculos en azul, la posición de las cámaras obtenidas luego del proceso de calibración. Se observa además, en rojo, la reconstrucción de las posiciones del punto 3D en el espacio.

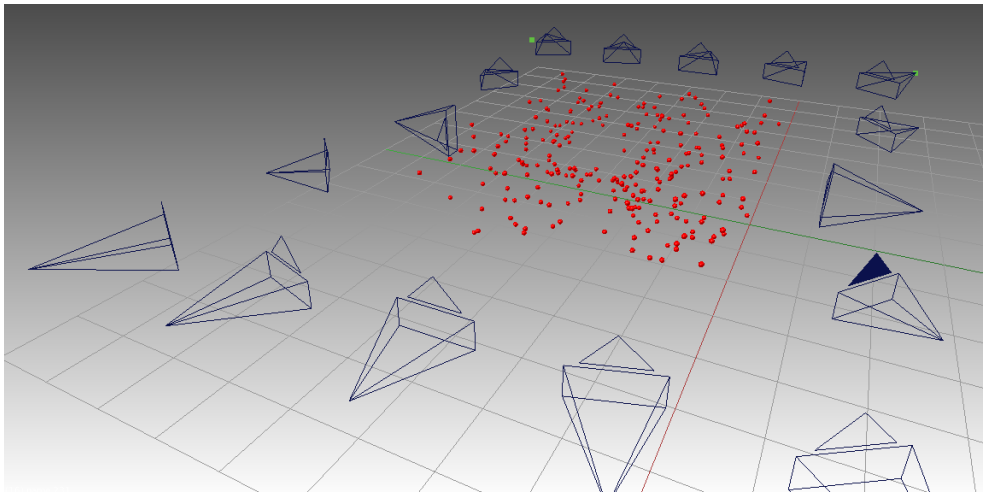
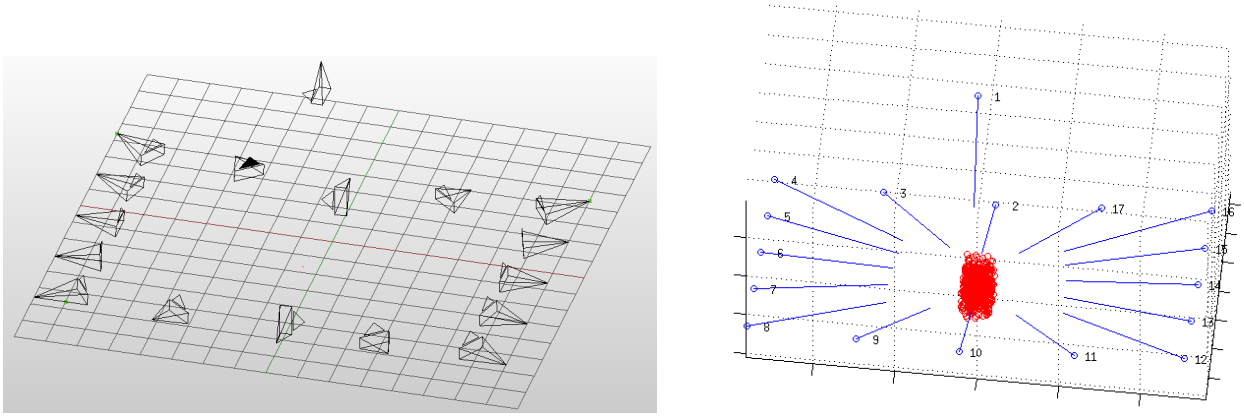


Figura 6.12: Distintas posiciones de un punto 3D a lo largo del volumen de trabajo



(a) *Blender* (b) Calibración

Figura 6.13: Configuración de las cámaras en el espacio 3D de *Blender* y la misma configuración hallada mediante la calibración

En la Figura 6.14 se muestra en azul el promedio y en rojo la desviación estándar del error de re-proyección obtenido para cada una de las cámaras. Sea X un punto 3D en el espacio cuya posición es a priori desconocida y x su correspondiente proyección en un cámara. A partir del resultado de la calibración se obtiene la matriz de proyección P de dicha cámara. A su vez, es posible obtener la estimación del punto X, siendo $\hat{X}$ dicha estimación. La proyección de $\hat{X}$ sobre la cámara es $\hat{x} = P\hat{X}$. El error de re-proyección de $\hat{X}$ se define como la distancia euclídea $d(x, \hat{x})$ entre los vectores x y $\hat{x}$.

Como se observa de la Figura se obtienen errores menores a 0.13 píxeles para todas las cámaras.

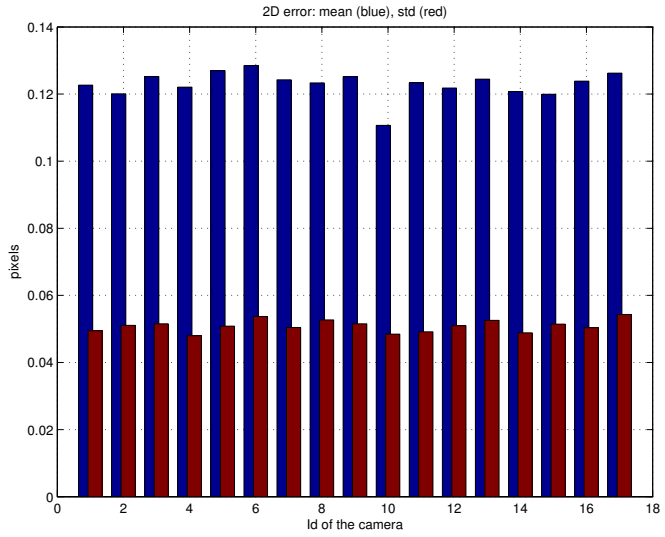


Figura 6.14: Promedio y desviación estándar del error de re-proyección en todas las cámaras

6.6.3. Ajuste del sistema de coordenadas.

De los dos toolbox vistos anteriormente se obtuvieron los parámetros intrínsecos y extrínsecos de las cámaras. En el *Multi-Camera Self-Calibration Toolbox* las parámetros extrínsecos de las cámaras son referidos a un sistema de coordenadas desconocido que se establece en el proceso de calibración del toolbox. El origen de coordenadas de ese sistema se ubica en el centroide de todas las posiciones capturadas de la fuente de luz. En el *amcctoolbox* por otra parte, los parámetros extrínsecos de las cámaras son referidos al sistema de coordenadas de la cámara que se definió como el origen del sistema de coordenadas.

En algunos casos puede desearse que el sistema de coordenadas del espacio 3D sea definido por el usuario. En nuestro caso para poder comparar el resultado de la calibración con el ground truth, el sistema de coordenadas al que están referidos los parámetros de la calibración, debe coincidir con el sistema de coordenadas definido en el entorno *Blender*.

Una opción para realizar esto, es colocar en el espacio marcadores en posiciones conocidas y capturar imágenes de dichos marcadores en cada cámara. Se asume que de la calibración se obtuvieron las matrices de proyección para cada una de las cámaras referidas a un sistema de coordenadas del espacio, *a priori*, desconocido. Sea entonces, S_1 dicho sistema de coordenadas, y $P_i|_{S_1}$ las matrices de proyección referidas a este sistema, siendo i la i-ésima cámara. Sea S_2 el sistema de coordenadas que se desea establecer como el nuevo sistema de referencia del espacio 3D, y $P_i|_{S_2}$ las matrices de proyección referidas a este sistema, que se deben encontrar.

Si se colocan marcadores en determinadas posiciones del espacio, entonces son

conocidas sus coordenadas respecto al sistema S_2. Sean $X_j|_{S_2}$ sus coordenadas 3D en el sistema S_2, siendo j el j-ésimo marcador. Dichos puntos se proyectan en las cámaras en los puntos de coordenadas 2D x_j^i. Con dichos puntos y las correspondientes matrices de proyección $P_i|_{S_1}$ es posible conocer las coordenadas de los puntos X_j respecto al sistema S_1, esto es, $X_j|_{S_1}$. Con lo cual se tienen las coordenadas de los j marcadores en ambos sistemas de coordenadas. De esta manera es posible inferir la transformación Tr tal que $Tr(X_j|_{S_1}) = X_j|_{S_2}$. Si se aplica el cambio de base a las matrices de proyección se obtiene $Tr(P_i|_{S_2}) = P_i|_{S_1}$. Dicha transformación es una composición de una traslación y una rotación.

6.7. Conclusiones

En el presente capítulo se han tratado varios métodos de calibración y se ha expuesto porqué dicho proceso es necesario para conocer los parámetros de las cámaras en sistemas de captura. Se ha visto además que de la calibración se obtienen ciertos parámetros que son intrínsecos y otros extrínsecos a las cámaras, basados en el modelo *pinhole* de cámara. Por otra parte se describe cómo obtener los parámetros de calibración a partir de la información proporcionada por el entorno *Blender*, esto último permite conocer el *ground truth* de la calibración para un conjunto de cámaras así como el *ground truth* de la detección de marcadores en las imágenes 2D.

Se implementó un algoritmo que realiza la calibración de cámaras, para la metodología particular utilizada en la captura de movimiento realizada a pacientes en el Hospital de Clínicas. Dicha metodología plantea la utilización de una estructura tridimensional con marcadores ubicados convenientemente. Una característica de este método es que permite abarcar todo el volumen de trabajo. Asimismo el uso de objetos 3D para la calibración ofrece mayor precisión [55]. Sin embargo, tiene como desventaja que requiere la intervención del usuario en el proceso de calibración, con lo cual resulta en un metodología poco adecuada para calibrar un sistema de muchas cámaras.

Por otra parte se han utilizado dos toolbox elaborados en *Matlab*, para calibrar las cámaras del laboratorio virtual implementado en *Blender*. Para esto se ha simulado en el mismo entorno *Blender* una calibración real utilizando estos toolbox.

El *amcctoolbox*, plantea un método muy utilizado para la calibración de cámaras a través de objetos planos, en nuestro caso un damero, cuyos resultados poseen buena precisión [55]. Sin embargo presenta la desventaja de que los parámetros extrínsecos de las cámaras se vinculan en relación a sus adyacentes, por lo que debe definirse a una de las cámaras como el origen del sistema de coordenadas del espacio, respecto a la cual se halla la posición del resto de las cámaras. Por lo tanto el error se acumula incrementándose en aquellas cámaras que están más alejadas de aquella que se define como el origen del sistema. Este toolbox por otra parte, requiere para la calibración de un par cámaras, establecer en forma automática una correspondencia entre la imagen del damero detectado en la cámara izquierda con

la imagen capturada por la cámara derecha. Esto en algunos casos no se logra y la correspondencia entre dichas imágenes debe realizarse en forma manual.

Por otra parte el otro de los toolbox evaluados, el *Multi-Camera Self-Calibration Toolbox* plantea una forma sencilla de calibrar un conjunto de muchas cámaras y que, a priori, parece más adecuado para el sistema de 17 cámaras del laboratorio virtual desarrollado en *Blender*.

Debe considerarse para trabajos futuros, además de los aspectos cualitativos expuestos, una evaluación de performance de los métodos considerados, teniendo en cuenta las diferencias implicadas en la calibración de un sistema de pocas o muchas cámaras.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Reconstrucción

7.1. Introducción

A la salida del bloque de detección de marcadores se tiene, para cada cámara y para cada cuadro de una secuencia adquirida, un conjunto de coordenadas en dos dimensiones (x, y) que ubican la posición en la imagen de aquellos marcadores que fueron detectados. El proceso de reconstrucción consiste en obtener las coordenadas en tres dimensiones de la posición de los marcadores en el espacio a partir de las coordenadas en dos dimensiones obtenidas en el bloque anterior. En la Figura 7.1 se muestra un bosquejo de la reconstrucción de un marcador usando para esto la detección de marcadores en dos cámaras.

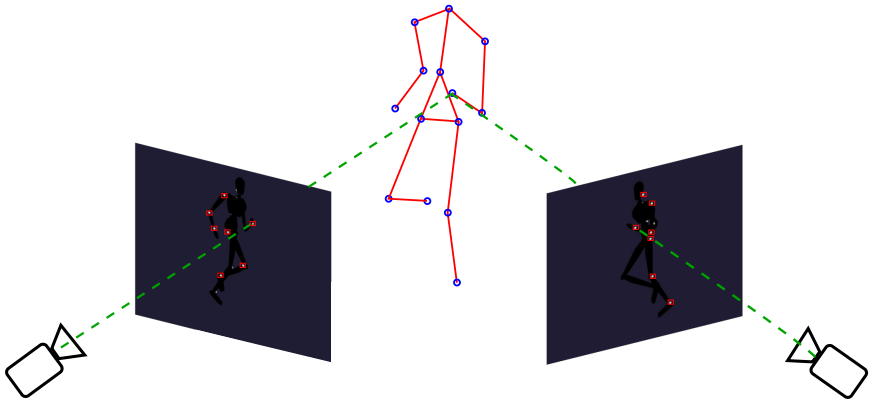


Figura 7.1: Reconstrucción con dos cámaras.

El proceso de reconstrucción implementado fue inspirado en el trabajo de Herda [14] y consiste en tres pasos fundamentales:

1. Determinar aquellos marcadores detectados en las distintas cámaras que corresponden a un mismo marcador en el espacio. De esta manera se establece

una correspondencia entre los marcadores detectados en las distintas vistas. Dichas asociaciones se establecen de a pares de cámaras.

2. Una vez establecidas estas correspondencias se selecciona aquella que bajo algún criterio, pueda considerarse con mayores posibilidades de ser una asociación correcta. Luego se determina la posición en el espacio del marcador a partir de la asociación seleccionada.

3. Una vez reconstruido uno de los marcadores debe verificarse si dicho marcador fue detectado en el resto de las cámaras.

7.2. Geometría epipolar

Para explicar el algoritmo de reconstrucción es necesario describir brevemente algunos conceptos fundamentales de la geometría epipolar. Dicha geometría es la que se presenta cuando dos cámaras en distintas posiciones se encuentran capturando el mismo espacio 3D. El análisis de esta situación permite obtener relaciones entre los puntos 3D con sus correspondientes proyecciones en las cámaras, así como las relaciones entre los propios puntos proyectados en las distintas cámaras [60] [56].

Como se muestra en la Figura 7.2, se tiene el caso en que dos cámaras observan un mismo punto 3D en el espacio, el punto X. Para esto se considera el modelo *pinhole* de la cámara descrito en el Capítulo 6. En ese caso los puntos O_1 y O_2 son los centros o focos de las cámaras.

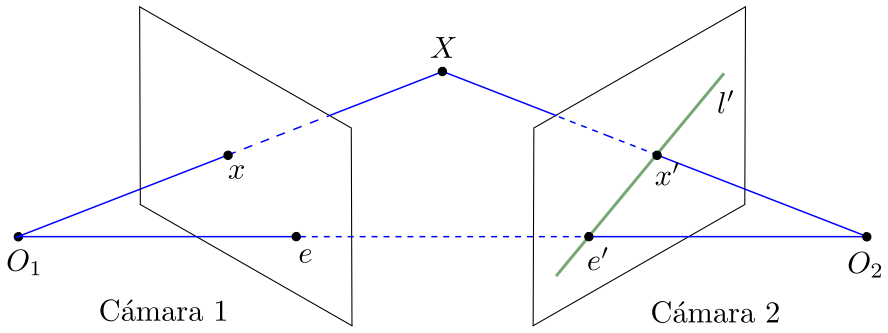


Figura 7.2: Geometría epipolar.

El punto X se proyecta en dichas cámaras en los puntos x y x'. Es decir, x pertenece a la intersección de la retina de la cámara 1, con el rayo de proyección $\overline{O_1X}$. Análogamente el punto x' pertenece a la intersección de la retina de la cámara 2 con el rayo de proyección $\overline{O_2X}$. A su vez, los focos de ambas cámaras O_1 y O_2 definen un rayo que corta a las retinas de las cámaras en los puntos e y e' llamados epipolos.

Si el punto X varía sobre el espacio 3D, se tienen múltiples rayos de proyección que pasan por dicho punto y el foco de la cámara 1, el punto O_1. Dado que O_1 se proyecta en la cámara 2 como el punto e' en el plano imagen, todas los rayos $\overline{O_1X}$ se proyectan en dicha cámara como rectas que se intersecan en el punto e', a estas rectas se les denomina rectas epipolares. Análogamente los sucesivos rayos de proyección $\overline{O_2X}$ se proyectan sobre la cámara 1 como rectas epipolares que se intersecan en el punto e. De esta manera se tiene que los puntos X, O_1 y O_2 forman un plano, llamado plano epipolar, que interseca a los planos imagen en las rectas epipolares.

De lo anterior se observa que teniendo la proyección x de un punto X sobre la cámara 1, también se conoce la recta epipolar l' y se puede ver que el punto X se proyecta en la cámara 2 en un punto x' situado en dicha recta epipolar $l' = \overline{e'x'}$. Esto implica que por cada punto visto en una retina en la otra se observa una línea.

Si los puntos x y x' son conocidos, sus rayos de proyección son también conocidos y si estos puntos corresponden a un mismo punto 3D sus rayos de proyección deben interceptarse en X. Por lo tanto las coordenadas del punto X pueden derivarse a partir de las coordenadas de sus puntos imagen.

Para el caso de cámaras reales se tienen distorsiones de ruido, imperfecciones en las lentes de las cámaras, etc. que producen que las proyecciones sean tal que el punto 3D, su proyección en la cámara y el foco de la misma no sean colineales, por lo tanto la proyección real de un punto 3D va a ser aproximadamente el punto ideal de su proyección.

7.2.1. Matriz Fundamental

La matriz fundamental es la representación algebraica de la geometría epipolar. Si se tiene el punto X y sus correspondientes proyecciones x y x' en las cámaras, se verifica la siguiente condición :

$$(x')^T F x = 0 \tag{7.1}$$

siendo F la matriz fundamental y x, x' las proyecciones en coordenadas homogéneas.

Supongamos que se tienen para la cámara 1, la matriz de proyección P y el punto x mencionado anteriormente. El rayo de proyección $\overline{O_1X}$ puede describirse en forma paramétrica como:

$$X(\lambda) = P^+ x + \lambda O_1 \tag{7.2}$$

donde P^+ es la pseudo-inversa de P, tal que $PP^+ = I$. En particular se toman dos puntos de esa recta, el punto P^+x ($\lambda = 0$) y el punto O_1 ($\lambda = \infty$). Si se proyectan estos puntos en la cámara 2 se obtienen los puntos $P'P^+x$ y $P'O_1$ respectivamente,

siendo P' la matriz de proyección de la cámara 2. Estos puntos pertenecen a la recta epipolar:

$$l' = (P'O_1) \times (P'P^+x) \tag{7.3}$$

El punto $P'O_1$ es el epipolo e' de la cámara 2. Por lo tanto se tiene que $l' = e' \times (P'P^+)x$, donde se define la matriz fundamental como:

$$F = e' \times P'P^+ \tag{7.4}$$

Si los puntos imagen se corresponden $x \leftrightarrow x'$, entonces x' se encuentra en la recta epipolar $l' = Fx$ correspondiente al punto x, lo cual implica que se cumple $0 = (x')^T l'$ y por lo tanto se verifica la condición de la Ecuación 7.1.

Como se explica anteriormente las igualdades de las ecuaciones mostradas se cumplen en condiciones ideales pero no para cámaras reales en las que aparecen los efectos del ruido, etc., por lo tanto cabe esperar que si se tiene un punto x y su correspondiente x', la Ecuación 7.1 no se verifique exactamente, aunque si debe devolver un valor próximo a cero.

Una propiedad básica de esta matriz es que si F es la matriz fundamental del par de cámaras con matrices de proyección (P, P'), entonces F^T es la matriz fundamental del par de cámaras en sentido opuesto: (P', P). Por otra parte si se tiene el punto x en la imagen de una cámara, su recta epipolar correspondiente en otra cámara es $l' = Fx$. Análogamente $l = F^T x'$ representa la línea epipolar asociada con x' en la segunda cámara.

7.3. Algoritmo propuesto por Herda

Como se explica en el Capítulo 4, se inicia la implementación de este bloque a partir del algoritmo propuesto por Herda [14]. Este algoritmo en algunos aspectos no es descripto con el detalle suficiente tal que pueda ser replicado fielmente. Por esta razón su implementación se ha elaborado realizando algunos supuestos en los casos donde su interpretación presenta ambigüedades.

Por otra parte, el sistema diseñado por Lorna Herda pretende utilizar la información del esqueleto (ver Capítulo 4), como por ejemplo la posición de articulaciones, distancias entre marcadores, etc. Dado que esta etapa del algoritmo no fue implementada resultó necesario robustecer el proceso de reconstrucción en las partes del algoritmo donde no se utiliza esta información.

Además, este algoritmo utiliza la triangulación estéreo para reconstruir un punto 3D a partir de los puntos 2D detectados en las distintas vistas. La correspondencia entre distintas vistas se establece mediante la matriz fundamental, cuyos conceptos fueron explicados en la sección anterior. Esta matriz se puede obtener luego del proceso de calibración.

Las ideas que sigue el algoritmo se describen a continuación:

- Se toman dos vistas y se realiza el test de la condición epipolar (Sección 7.4.2). Si existe una asociación no ambigua se reconstruyen las coordenadas 3D a partir de las coordenadas 2D.
- Las coordenadas 3D reconstruidas se re-proyectan en las cámaras restantes. Los puntos 2D encontrados en esa re-proyección son asociados al punto 3D. De esta forma se tiene por cada punto 3D, sus correspondientes puntos 2D asociados en las distintas vistas.
- Se considera que un marcador 3D está correctamente reconstruido si se re-proyecta en al menos una cámara. A este tipo de reconstrucción se le llama *reconstrucción trinocular*.
- Si el número de marcadores reconstruidos es inferior al número de marcadores colocados sobre la persona, entonces se realiza una segunda asociación entre dos vistas. En este caso la reconstrucción se realiza solamente con dos vistas (*reconstrucción binocular*).

Posteriormente el algoritmo plantea realizar ciertos chequeos sobre los puntos reconstruidos de manera de validarlos. Dichos chequeos utilizan información del esqueleto (Figura 7.3).

Lo que se busca verificar en dichos chequeos, es que para un determinado cuadro los marcadores sean efectivamente visibles por aquellas cámaras que los reconstruyeron y no están ocultos por alguna parte del cuerpo, lo que evidenciaría que la reconstrucción es incorrecta. Si esto llegara a suceder, el algoritmo busca otras vistas de las cuales reconstruir el marcador.

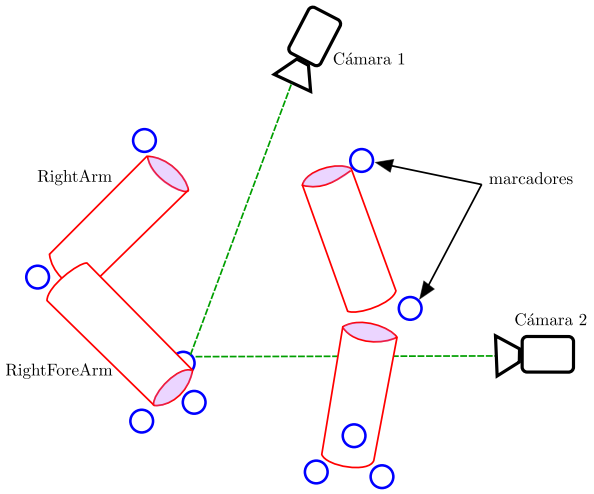


Figura 7.3: Chequeo de visibilidad y oclusión [14].

7.4. Algoritmo implementado

En la Sección anterior se describió el algoritmo de reconstrucción propuesto en el sistema de Herda, sin embargo, el algoritmo implementado no es exactamente el mismo. Como se aclara sobre el final de la Sección 4 , la cantidad de módulos a implementar es grande, presentando ambigüedades importantes que dificultan en muchos casos su reproducibilidad, considerando los tiempos con los que cuenta el proyecto, se decide dar prioridad a los bloques principales, comenzando por lo básico e implementando módulos hasta lograr la performance necesaria. A continuación se describe la implementación realizada para este proyecto y las diferencias con el algoritmo de reconstrucción propuesto por Herda.

El algoritmo implementado recibe como entrada los puntos 2D de los marcadores detectados y devuelve como salida los puntos 3D reconstruidos. El primer paso consiste en establecer una asociación entre ciertos puntos 2D de distintas cámaras. Luego, se pasa a un conjunto de bloques que se ejecutan de manera iterativa hasta que no queden marcadores para reconstruir. En dicho bloque se busca la mejor asociación entre puntos, bajo determinado criterio, luego se reconstruye un punto 3D y se realiza un proceso de validación de dicha reconstrucción. En la iteración siguiente se actualizan las asociaciones que habían sido establecidas previamente. Cuando no hay más marcadores para reconstruir se detiene el proceso iterativo y se devuelven aquellos marcadores que fueron reconstruidos en cada iteración. En la Figura 7.4 se presenta un diagrama del algoritmo.

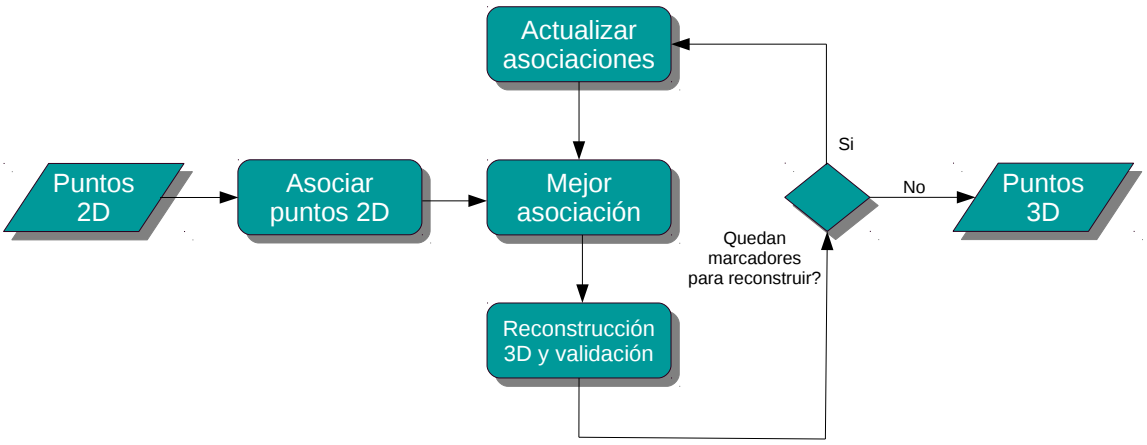


Figura 7.4: Diagrama del algoritmo implementado.

A continuación se describe el funcionamiento de cada uno de los bloques.

7.4.1. Asociar puntos 2D

Este bloque recibe como entrada las coordenadas de los puntos detectados en cada una de las cámaras, parámetros de las mismas tales como sus matrices de

proyección y devuelve para cada punto una lista ordenada por relevancia, de las asociaciones existentes con puntos en otras cámaras. Basándose en lo explicado anteriormente el proceso se puede ejemplificar en la Figura 7.5.

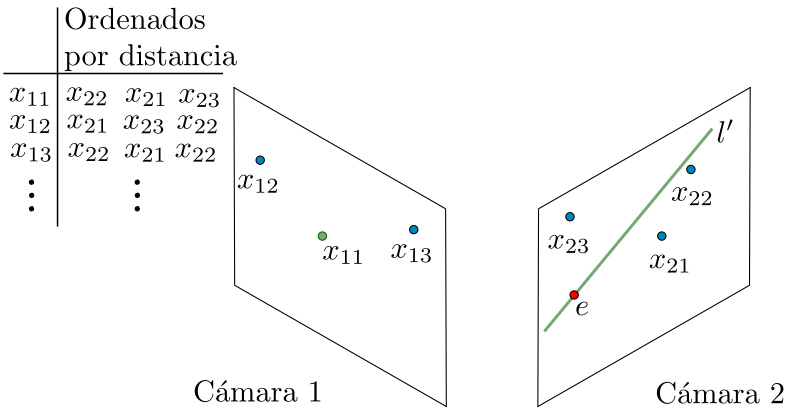


Figura 7.5: Asociación de puntos 2D en dos cámaras.

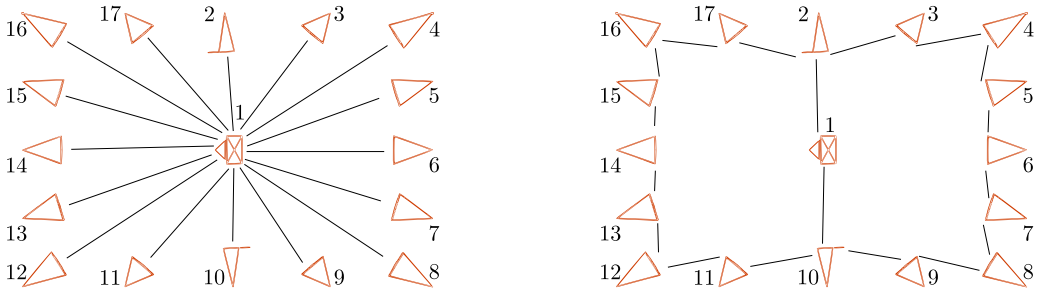
En primer lugar se seleccionan dos cámaras y se considera un punto en una de ellas, por ejemplo el punto x_{11} de la cámara 1, y se evalúa la Ecuación 7.1 para cada punto en la cámara 2. Esto equivale a proyectar la recta epipolar l' correspondiente al punto x_{11} sobre la cámara 2 y tomar las distancias de los puntos detectados en la cámara 2 a la recta l'. Se demuestra que dicha distancia difiere a menos de un factor de escala respecto al valor obtenido al evaluar la Ecuación 7.1. Se asume que los puntos de la cámara 2 que tengan mayor posibilidad de corresponder con el punto x_{11}, son aquellos que al ser evaluados por la ecuación obtienen valores próximos a cero. De esta manera se obtiene para cada punto en la cámara 1 un conjunto de puntos en la cámara 2 ordenados según su distancia a la recta epipolar correspondiente. Repitiendo el procedimiento de manera inversa, esto es, de la cámara 2 a la cámara 1, se obtiene igualmente para cada punto de la cámara 2 los puntos de la cámara 1 ordenados según su proximidad a la recta epipolar correspondiente. A continuación se toman otros pares de cámaras y se vuelve a repetir el proceso.

Es importante resaltar que para la elección de los pares de cámaras se han considerado dos casos. El primero de ellos evalúa cada cámara respecto a todas las restantes y el segundo considera la disposición de las cámaras en el espacio y empareja las cámaras adyacentes de manera consecutiva (ver Figura 7.6).

7.4.2. Mejor asociación

A partir de la lista con asociaciones entre puntos de dos vistas generada anteriormente, es necesario elegir aquella que posea mayor probabilidad de conformar la pareja de imágenes correspondiente a la proyección de un marcador 3D sobre dichas vistas.

Recordando que todas las asociaciones de puntos entre pares de cámaras se encuentran ordenadas por distancia, se toma aquella asociación que posea la me-



(a) Cada cámara con las restantes. Ejemplo de emparejamiento de la cámara 1.

(b) Cada cámara con su adyacente

Figura 7.6: Métodos utilizados para generar pares de cámaras.

nor distancia y contenga puntos válidos, descartando las restantes. Un punto se considera válido si en iteraciones anteriores no se a podido asociar a ningún punto 3D reconstruido (ver Sección 7.4.4). De esta forma cada punto de una cámara es asociado, si existen puntos válidos, con un punto en otra de las cámaras.

Supongamos en la Figura 7.2 que los puntos x y x' son la mejor asociación entre la cámara 1 y la cámara 2, y efectivamente son imágenes de un mismo punto 3D, idealmente los rayos de proyección se interceptarían en X, pero debido a incertidumbres en los procesos de calibración y segmentación la distancia entre dichos rayos no es nula, en el mejor de los casos solo es próxima a cero. Para evaluar cuál es entre los pares de puntos asociados disponibles el par que posee mayor posibilidad de corresponder a las proyecciones de un punto 3D, se proyectan todos los rayos de proyección y se toma aquella pareja de puntos que genere rayos de proyección con la menor distancia entre sí.

7.4.3. Reconstrucción 3D y validación

Luego de encontrar las mejores asociaciones entre dos cámaras se procede a reconstruirlas. Estas reconstrucciones deben ser validadas, para ello en principio se utiliza el criterio propuesto por Herda [14]: se considera que una reconstrucción proveniente de dos cámaras es válida, si al proyectar sobre una tercera cámara existe al menos un punto de esta última que diste menos de un cierto valor umbral. Si efectivamente se encuentra al menos un punto, se asocia con los dos puntos que generaron la reconstrucción y se repite el proceso con el resto de las cámaras. Una vez finalizada la iteración, se retira a la pareja que genera la reconstrucción así como también a los puntos que lograron validarla, y se itera nuevamente repitiendo el proceso con la siguiente mejor pareja asociada entre dos cámaras.

El algoritmo desarrollado, si bien conceptualmente es similar, se implementa desde una óptica diferente computacionalmente más eficiente. En lugar de llevar en cada iteración la información 3D a cada cámara, se procede a llevar la información de las cámaras una sola vez al espacio 3D y trabajar en cada iteración sobre el

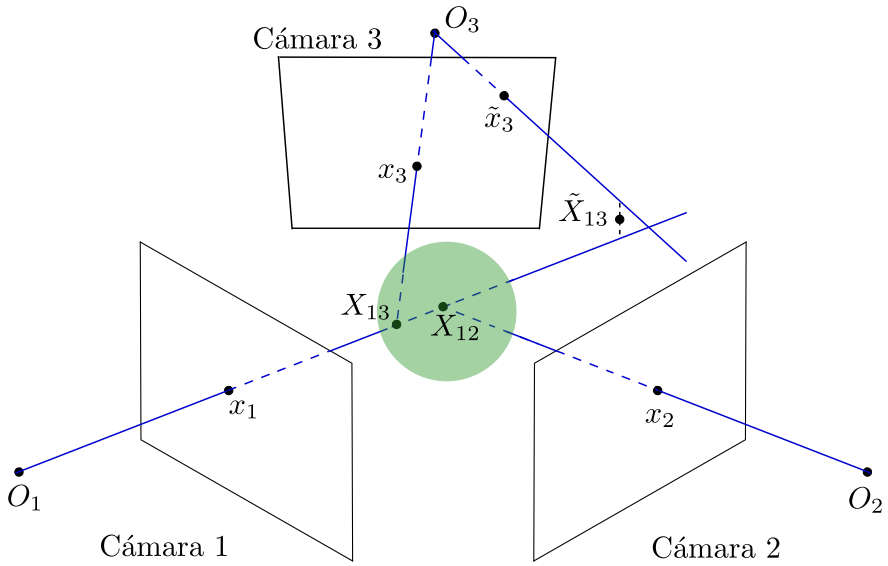


Figura 7.7: Reconstrucción entre cámaras 1, 2 y validación con cámara 3. El punto x_3 de la cámara 3 valida la reconstrucción, no así el punto $\tilde{x}_3$.

mismo.

La información que contiene un punto en una retina se mapea en el espacio 3D sobre el rayo de proyección que contiene a dicho punto y al centro de la cámara correspondiente. Supongamos que se tienen los rayos de proyección en el espacio 3D de todos los puntos contenidos en las retinas y que x_1 es un punto en la cámara 1 de centro O_1 y x_2 es un punto en la cámara 2 de centro O_2 se encuentran asociados y reconstruyen al punto X_{12} (ver Figura 7.7).

Idealmente X_{12} se genera al interceptar los rayos de proyección de los puntos x_1 y x_2, pero debido a incertidumbres en la detección de marcadores o la calibración, comúnmente los rayos se van a cruzar. La reconstrucción, se estima como el punto del espacio de menor distancia a ambos rayos, por lo que X_{12} se encuentra en el punto medio del segmento perpendicular a ambos rayos.

El algoritmo implementado asume que un punto en una cámara, valida a X_{12} si junto a x_1 reconstruye un punto 3D que se encuentra dentro de la esfera $B(X_{12}, \delta)$ de centro X_{12} y radio δ, donde δ es un cierto valor umbral. Notar que la elección anterior de x_1 sobre x_2 es indiferente para los propósitos de este proyecto.

Si bien este criterio difiere del postulado original propuesto por Herda, en cierta manera se considera más robusto, pues en el postulado original basta con que dos puntos en una retina se encuentren lo suficientemente cerca para obtener una validación, pero esto no implica necesariamente que provengan de puntos 3D cercanos, sin embargo bajo el nuevo criterio si dos puntos 3D se encuentran cerca, entonces es válido afirmar que sus proyecciones sobre las distintas retinas también deben estarlo. Otra ventaja es que los umbrales bajo los cuales se considera que dos puntos están "cerca" difieren en cada una de las cámaras debido al distinto mapeo de distancias, sin embargo en el espacio 3D, manejar un único umbral resulta suficiente.

7.4.4. Actualizar asociaciones

Del bloque anterior se tiene, un punto X reconstruido y sus correspondientes proyecciones en cada una de las cámaras. Dichas proyecciones no deben ser consideradas nuevamente, por tanto estos puntos 2D que reconstruyen X, no se consideran como puntos válidos en las siguientes iteraciones.

Finalmente el proceso iterativo se detiene cuando no hay más marcadores para reconstruir, lo cual implica que se cumple alguna de las siguientes condiciones:

- el número de marcadores reconstruidos es igual al número de marcadores que tiene colocada la persona, o igual al número máximo de marcadores reconstruidos que se haya indicado.
- No existen puntos 2D válidos tal que pueda establecerse una asociación entre puntos de distintas vistas.

7.5. Resultados de reconstrucción sobre secuencias sintéticas

Se ha probado el algoritmo descrito anteriormente con secuencias capturadas en Blender. Para la configuración de 17 cámaras se han obtenido resultados aceptables, con errores promedio en torno a los 0,5 cm. En la Sección 9.2, se define la medida de error y se evalúa el desempeño del algoritmo respecto a dicha medida. En dichas pruebas se observan resultados aceptables aún utilizando hasta 6 cámaras.

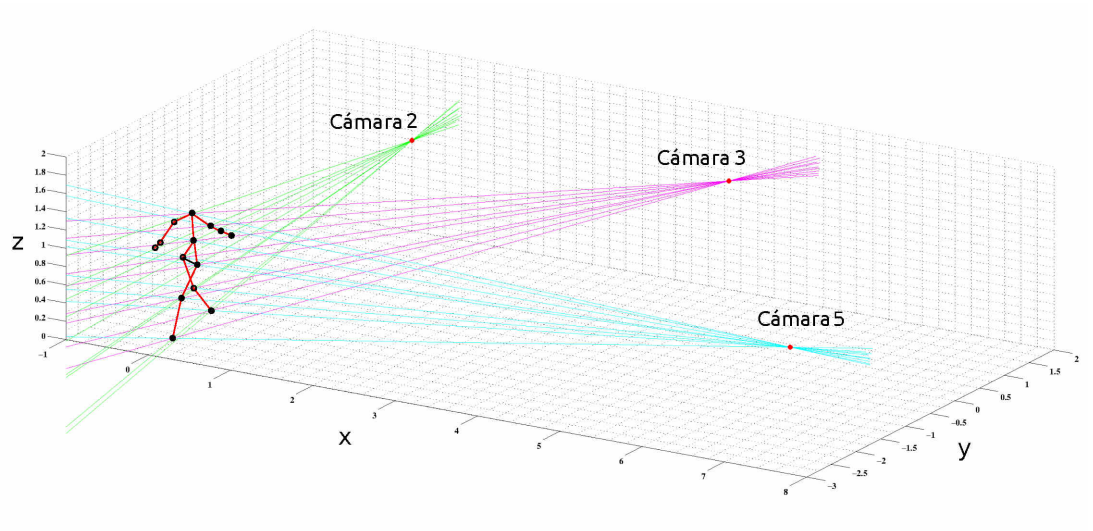


Figura 7.8: Ejemplo del proceso de reconstrucción. Los marcadores totalmente negros son los que se pudieron reconstruir utilizando información de las cámaras 2, 3 y 5.

Como se menciona previamente se han realizado dos implementaciones distintas del bloque *Asociar puntos 2D* (Sección 7.4.1), que se diferencian en el criterio para seleccionar los pares de cámaras:

- La primera implementación considera las asociaciones de marcadores posibles entre pares de cámaras, evaluando cada una de las cámaras respecto a todas las restantes.
- La segunda implementación toma en cuenta la configuración circular de las cámaras en el espacio y se evalúa cada cámara con la que se encuentra inmediatamente más cerca, recorriendo las cámaras en algún sentido.

		Primera implementación		**Segunda implementación**	
Marker Ground	**Name Ground**	**Error Promedio (cm)**	**Percentil 99 % (cm)**	**Error Promedio (cm)**	**Percentil 99 % (cm)**
1	LeftUpLeg	0.4182	**3.7197**	0.3671	0.5158
2	LeftLeg	0.3586	0.5449	0.3670	0.5411
3	LeftFoot	0.3862	1.0379	0.3720	0.5580
4	RightUpLeg	0.3963	1.7542	0.3714	0.5879
5	RightLeg	0.3830	0.9840	0.3780	0.5860
6	RightFoot	0.4192	1.5364	**0.4212**	**1.8483**
7	Spine	0.4075	0.9092	0.4040	0.6043
8	Head	0.4130	1.2853	0.3867	0.9063
9	LeftArm	0.3626	0.6394	0.3666	0.7997
10	LeftForeArm	**0.4774**	2.8511	0.3873	0.9056
11	LeftHand	0.4217	2.0700	0.4007	1.1722
12	RightArm	0.4756	2.4363	0.4025	1.4771
13	RightForeArm	0.4590	3.1711	0.3844	0.7810
14	RightHand	0.4545	2.4782	0.3816	0.7728
	Secuencia	**0.38556**	**1.7907**	**0.35686**	**0.81266**

Tabla 7.1: Performance de los algoritmos implementados en reconstrucción.

En la Tabla 7.1 se muestra una comparación de resultados. En la misma se observa que en la segunda implementación el error promedio mejora ligeramente respecto a la primera implementación. Se observa también el error considerando el 99 % de los marcadores, donde la mejora de la segunda implementación es más significativa. Por esta razón se ha elegido la segunda de las implementaciones.

Una de las razones por las cuales puede suceder esto, es que en el primer algoritmo se realizan búsquedas de asociaciones entre marcadores detectados, en pares de cámaras donde la probabilidad de que un mismo marcador sea visto por ambas puede ser muy baja. El ejemplo más representativo es cuando se consideran cámaras que están ubicadas en posiciones diametralmente opuestas. En este caso las asociaciones que se encuentren tienen mayores posibilidades de ser asociaciones incorrectas, influyendo negativamente sobre el desempeño global del algoritmo.

7.6. Resultados de reconstrucción sobre secuencias reales

Para probar la reconstrucción con secuencias reales se realizó la calibración según lo explicado en la Sección 6.5, por otra parte fue necesario robustecer el algoritmo de detección de marcadores como se explica en la Sección 5.6. Las pruebas del algoritmo presentado anteriormente realizadas sobre esta secuencia no han dado resultados aceptables, en este caso la mayoría de los marcadores se reconstruyen erróneamente.

Con el fin de encontrar donde falla el sistema se genera una secuencia sintética donde se utilizan tres cámaras para la captura. Dichas cámaras están ubicadas en posiciones relativas que simulan el caso real, de esta manera se procede a probar el sistema con dichas secuencias. En este caso los resultados tampoco han sido aceptables. Tras varias pruebas se llega a la conclusión de que la deficiencia del sistema es que el algoritmo de reconstrucción, no es lo suficientemente robusto para el caso en que se utilizan pocas cámaras.
Esta situación ha motivado el desarrollo de mejoras sobre este algoritmo. Los cambios se realizan sobre la implementación disponible del sistema, con el fin de lograr un desempeño aceptable sin modificar la estructura global del sistema.

7.7. Mejoras del algoritmo de reconstrucción

Se hicieron pruebas sobre la base sintética, nuevamente simulando la situación del caso real, con el fin de aislar los errores provenientes de la calibración y de la detección de marcadores. El algoritmo que se implementa en este caso, con el fin de solventar dichos errores, tiene una estructura similar al considerado anteriormente (ver Figura 7.4), aunque algunos bloques modifican su funcionamiento, fundamentalmente el primero (*Asociar puntos 2D*).

7.7.1. Asociar punto 2D

En esta versión modificada, el resultado de *Asociar puntos 2D* continúa siendo una lista ordenada de asociaciones entre puntos de distintas cámaras, pero el procedimiento para generar dicha lista es diferente.

Para cada punto de una retina se genera una lista ordenada que asocia dicho punto con los puntos de las restantes retinas. El orden de la lista se establece según la distancia euclídea entre los rayos de proyección de los puntos a asociar.

De esta forma se obtiene una matriz de distancias entre rayos de proyección 3D, donde las columnas de la matriz son una combinación posible de dos rayos de proyección y las filas se componen de la siguiente manera:

- Número de cámara α.
- Índice del punto 2D cámara α.

- Número de cámara β.
- Índice del punto 2D cámara β.
- Distancia entre los rayos de proyección de los puntos considerados.

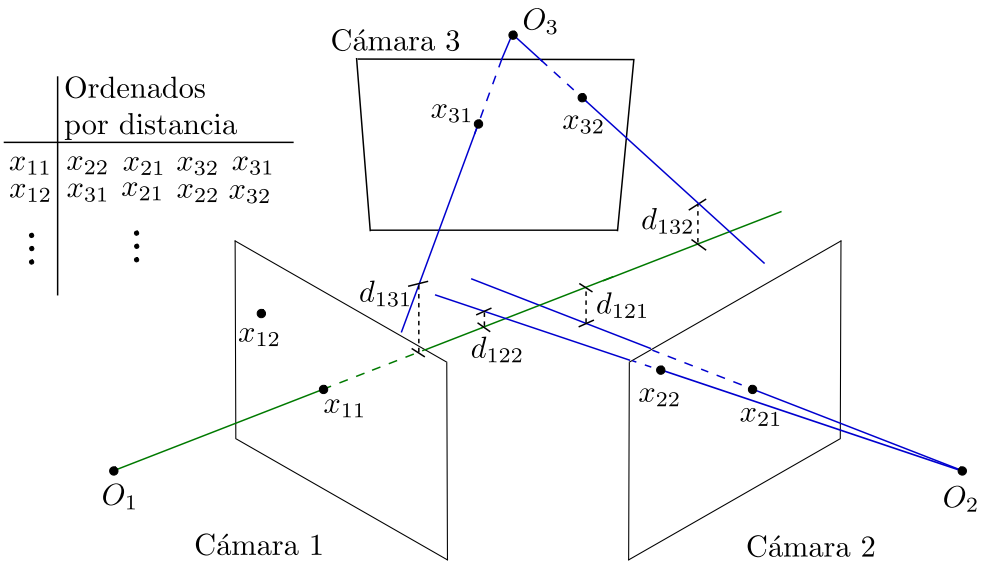


Figura 7.9: Nueva asociación de puntos entre pares de cámaras.

7.7.2. Mejor asociación

Este bloque mantiene su funcionalidad, encontrar la pareja de puntos entre dos vistas, con mayor probabilidad de corresponder a la proyección de un punto 3D. La diferencia fundamental con su implementación anterior es la gestión de las matrices de entrada, pues el ordenamiento por distancia de dichas matrices es diferente.

En la primera implementación las distancias que producen el ordenamiento son generadas entre rectas epipolares y puntos, o sea distancias sobre un plano, mientras que en esta implementación las distancias son entre rayos en el espacio 3D.

Manteniendo el criterio original para implementar este bloque, el cual consiste en tomar la pareja de puntos que genere rayos de proyección con la menor distancia entre sí. La nueva entrada proporcionada por *Asociar puntos 2D* simplifica la implementación a desarrollar.

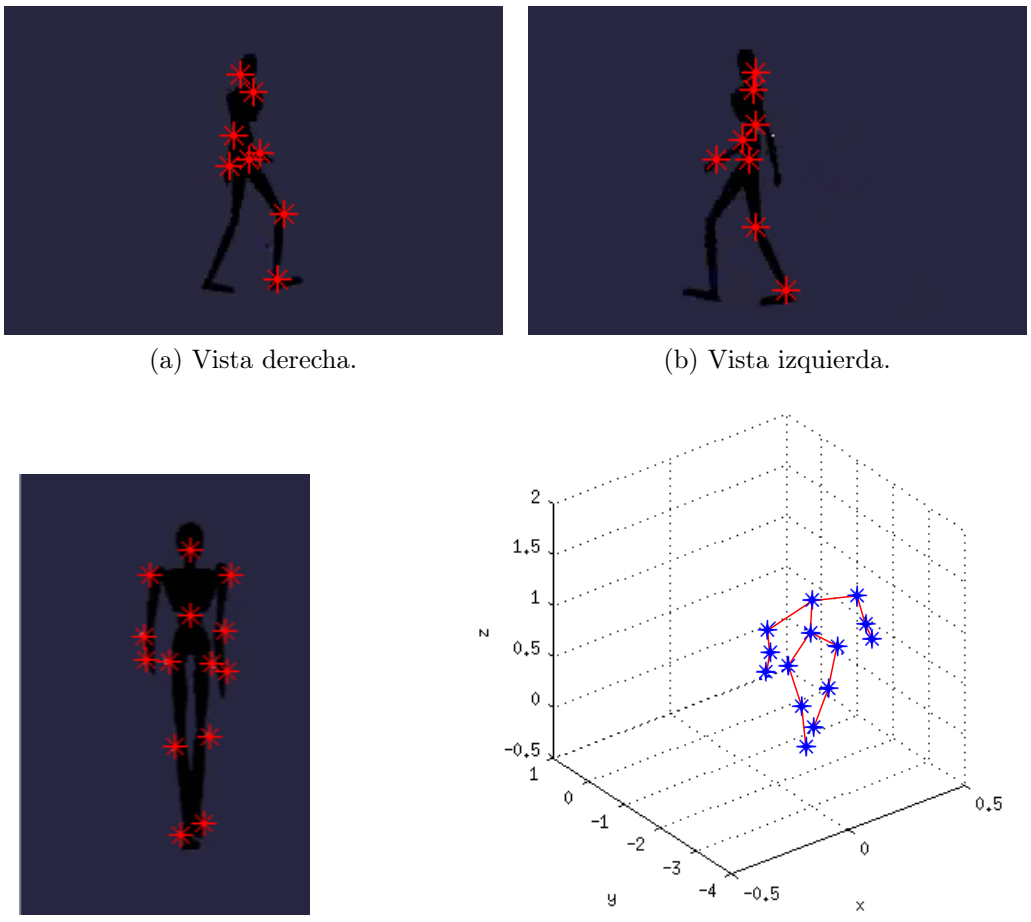
Una vez que se tiene la matriz de distancias con todas las asociaciones de puntos entre los pares de cámaras considerados, este bloque simplemente toma de entre todas las asociaciones disponibles de puntos válidos, aquel par de puntos que posea menor distancia asociada. Recordar que un punto se considera válido si en iteraciones anteriores no se ha podido asociar a ningún punto 3D reconstruido.

7.7.3. Resto de los bloques

Los bloques de *Reconstrucción 3D y Validación*, y el de *Actualizar Asociaciones* realizan el mismo funcionamiento y tiene igual implementación que los utilizados en el algoritmo anterior (ver Secciones 7.4.3 y 7.4.4).

Asimismo la condición que debe cumplirse para detener el proceso iterativo es la misma.

7.8. Resultados del nuevo algoritmo



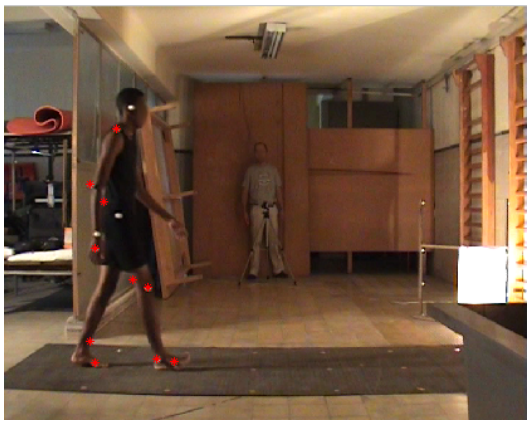
(a) Vista derecha. (b) Vista izquierda.

(c) Vista de atras. (d) Resultado de la reconstrucción.

Figura 7.10: Reconstrucción de una secuencia sintética. Los asteriscos rojos indican los marcadores detectados.

Se prueba el nuevo algoritmo para una secuencia sintética de tres cámaras en las que se intenta simular el caso real. En este caso el sujeto cuenta con catorce marcadores. Los resultados muestran una mejora significativa respecto al algoritmo original aunque no se llega a tener un desempeño ideal. De 20 cuadros relevados,

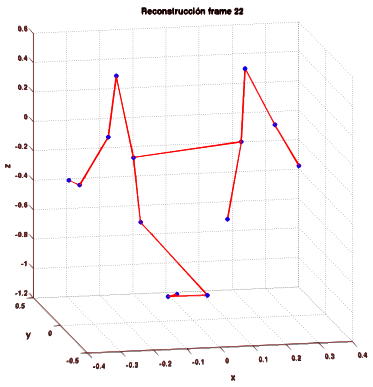
en 11 de ellos se logra reconstruir de manera aceptable los 14 marcadores, en seis de los cuadros se tiene un marcador incorrectamente reconstruido y en los tres restantes se tiene dos marcadores incorrectamente reconstruidos. En la Figura 7.10 se muestra el resultado de la reconstrucción para un cuadro determinado de la secuencia sintética.



(a) Vista derecha.

(b) Vista de atrás.

(c) Vista izquierda.



(d) Resultado de la reconstrucción.

Figura 7.11: Reconstrucción de una secuencia real. Los asteriscos rojos indican los marcadores detectados.

Por último el nuevo algoritmo fue probado con secuencias reales. Si bien se reconstruyen varios marcadores mejorando la performance que se tenía con el algoritmo de reconstrucción inicial, los resultados no son tan alentadores como los obtenidos para las secuencias sintéticas. En cierta manera es esperable dado que en este caso se adicionan errores provenientes de las etapas de detección de marcadores y de calibración. En la Figura 7.11 se muestra el resultado obtenido para un cuadro de la secuencia real.

7.9. Conclusión

Tomando como punto de partida las ideas propuestas por Herda, referentes a la reconstrucción de puntos 3D a partir de múltiples imágenes de puntos sobre distintas cámaras, se han diseñado e implementado dos variantes de algoritmo principal para la etapa de reconstrucción. Logrando probar dichos algoritmos sobre secuencias tanto sintéticas como reales se hace un análisis cualitativo de los resultados.

Inicialmente se genera un algoritmo con etapas bien definidas (ver Figura 7.4), que obtiene resultados aceptables para secuencias sintéticas y configuraciones de más de seis cámaras. Como es de esperar la configuración espacial de las cámaras influye en el desempeño del algoritmo, por lo que la afirmación anterior es válida asumiendo una distribución de cámaras uniforme alrededor del espacio de captura.

Las pruebas realizadas sobre secuencias reales dejan presente el problema de desempeño que tiene el algoritmo cuando se utilizan pocas cámaras, produciendo resultados poco satisfactorios. En este caso, las secuencias reales obtenidas manejan tan solo tres cámaras y las condiciones de laboratorio no son las más favorables para efectuar un proceso de detección adecuado. Se ha visto que es necesario robustecer las etapas tanto de detección de marcadores como de calibración, pues las incertidumbres producidas en dichas etapas tienen un efecto crítico sobre el desempeño de la reconstrucción. Con las secuencias reales disponibles, la falta de marcadores sobre las cámaras en la etapa de detección, es la mayor dificultad a enfrentar en el proceso de reconstrucción.

Con el fin de independizar la reconstrucción de los efectos de etapas anteriores, se introduce una secuencia sintética que simula el caso real, tanto en la calidad del movimiento como en el número y disposición de cámaras, donde los errores de calibración y detección de marcadores no son significativos.

Utilizando esta secuencia se implementa un nuevo algoritmo que mejora el desempeño en secuencias con pocas cámaras. Este nuevo algoritmo modifica parcialmente algunas etapas del algoritmo inicial, sobre todo la etapa de *Asociar puntos 2D*, logrando obtener resultados significativamente mejores al tratar con pocas cámaras sobre el caso sintético. Utilizando las secuencias reales, se obtiene mejoras cualitativas sobre el primer algoritmo, si bien el desempeño es inferior al caso sintético y la reconstrucción es apenas aceptable.

De lo expuesto se desprenden fundamentalmente dos cosas:

- Las dificultades que ofrece el caso real, básicamente en la detección de marcadores, y cómo influye significativamente en la etapa de reconstrucción perjudicando el resultado final del sistema. Con lo cual se desprende que mejoras en la detección deberían tener un gran impacto en la reconstrucción.
- El número de cámaras cambia significativamente el problema:
  - Por un lado se genera un algoritmo que gestiona de manera aceptable la redundancia de información que existe cuando se utiliza un número

de cámaras superior al necesario para cubrir cierto espacio de captura (dadas las características de los movimientos tratados en este proyecto, básicamente de marcha rectilínea, el límite inferior para una reconstrucción aceptable son seis cámaras). Cabe recordar en este punto que los sistemas comerciales poseen generalmente como mínimo 8 cámaras e intentan trabajar con redundancia de información.

- Por otro lado efectuando modificaciones al algoritmo anterior se genera un nuevo algoritmo, que permite trabajar con pocas cámaras pero no gestiona adecuadamente la redundancia de información.

El análisis en biomecánica exige restricciones sobre precisión espacial en los resultados de la reconstrucción, por lo que resulta necesario optimizar esta etapa.

De lo visto en la bibliografía existente podría proponerse el uso de la información de esqueleto, teniendo presente que esto último, impone restricciones sobre las posiciones de los marcadores y particulariza los casos de uso del sistema. Si bien este tipo de análisis se menciona en el algoritmo propuesto por Herda, se decide en este trabajo generar un algoritmo en principio más general, con entradas y salidas bien determinadas y no se incorpora la información de esqueleto.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 8

Seguimiento

8.1. Introducción

La salida del bloque de reconstrucción estudiado en el Capitulo 7 son los puntos 3D generados a partir de los múltiples vídeos de cada vista, presentados en el orden que fueron validados en cada cuadro (Figura 8.1). A su vez los marcadores fueron reconstruidos tomando como entrada los puntos obtenidos en las imágenes de la segmentación, sin tener estos información alguna sobre el cuerpo más que el orden en el que son generados al segmentar.

Estas reconstrucciones, son presentadas sin orden definido para cada cuadro de la secuencia, y el objetivo del tracking o seguimiento es identificarlas temporalmente, asignándoles una etiqueta constante a lo largo de la secuencia para cada marcador y obtener las trayectorias 3D.

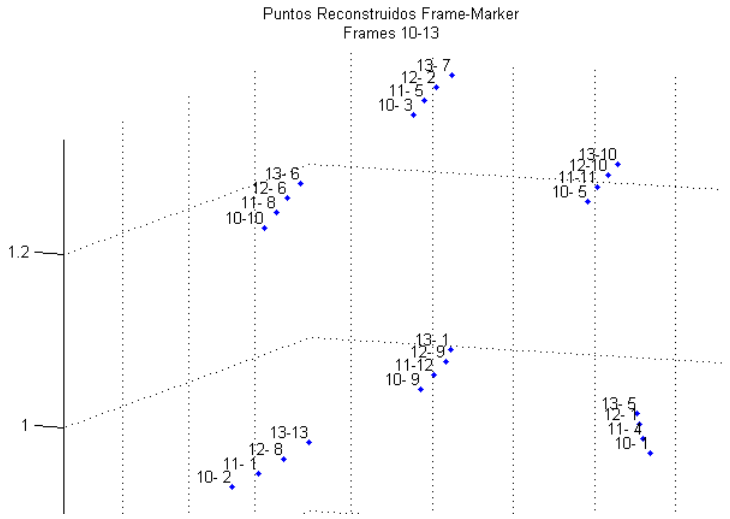


Figura 8.1: Salida de Reconstrucción para 4 cuadros. La etiqueta para cada marcador es el cuadro y el índice en la reconstrucción.

8.2. Estado del Arte

El problema del seguimiento de marcadores espaciales a lo largo del tiempo es un tema de estudio habitual con distintos enfoques, entre ellos:

- Aplicación de restricciones al movimiento, teniendo información previa de los marcadores a estudiar, como se mueven, y como se relacionan entre ellos.
- Predictores Lineales o Estadísticos, teniendo información previa de una trayectoria para estimar el próximo elemento y confirmar o corregir la continuación de la secuencia.

Estos enfoques no son mutuamente excluyentes, pudiendo ser ambos complementarios para robustecer la generación de las trayectorias.

Si se tiene acceso durante la adquisición de los datos, es posible relevar las distancias entre marcadores que se colocan en miembros que se mueven solidariamente o dependen directamente uno del otro, por ejemplo al colocar marcadores en articulaciones de los brazos hay elementos sobre los huesos que siempre tendrán la misma distancia entre ellos. Esto se cumple con una cierta tolerancia ya que los marcadores no se encuentran en la articulación misma sino sobre un lugar representativo del cuerpo como se observa en el ajuste presentado en la Figura 8.2 como es planteado en el trabajo de Lorna Herda [14].

Con estas distancias relevadas, y definidas las relaciones entre marcadores, es posible definir el esqueleto como una serie de restricciones que se mantienen a lo largo del tiempo (distancias entre marcadores solidarios) o que varían de forma continua (ángulo de huesos), donde la identificación es realizada inicialmente mediante un ajuste de los datos relevados en el mundo físico a los puntos obtenidos en la reconstrucción.

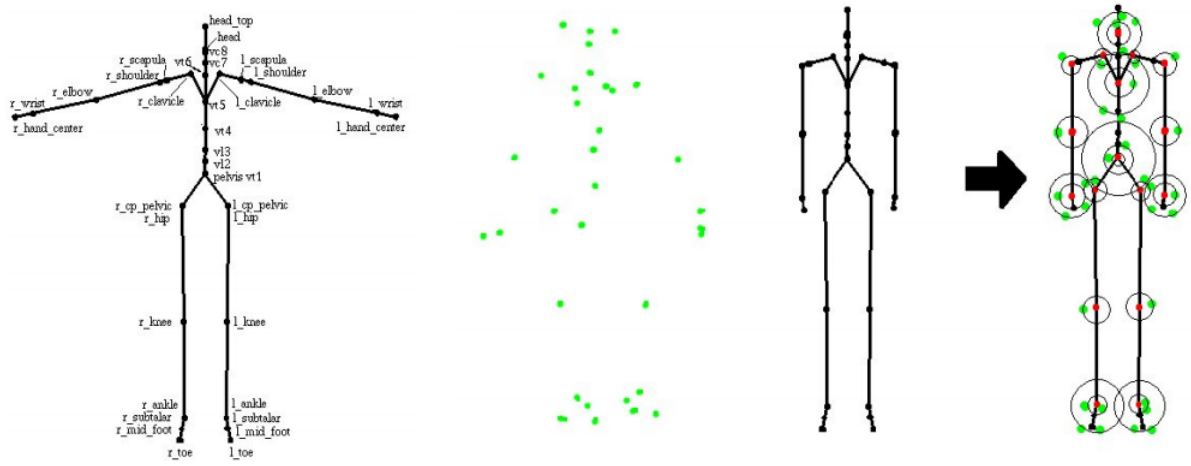


Figura 8.2: Inicialización de esqueleto, y ajuste a los marcadores encontrados. Cada marcador representa una articulación y se ajusta en una esfera de cercanía centrada en la articulación del esqueleto. Tomado de Herda [14].

Uno de los algoritmos más simples [17] es continuar una secuencia buscando el marcador más cercano al último punto conocido, imponiendo como restricción que el traslado es mínimo ("greedy matching"). Sin embargo, esto es susceptible a errores para casos muy simples, como se observa en el ejemplo de la Figura 8.3 .

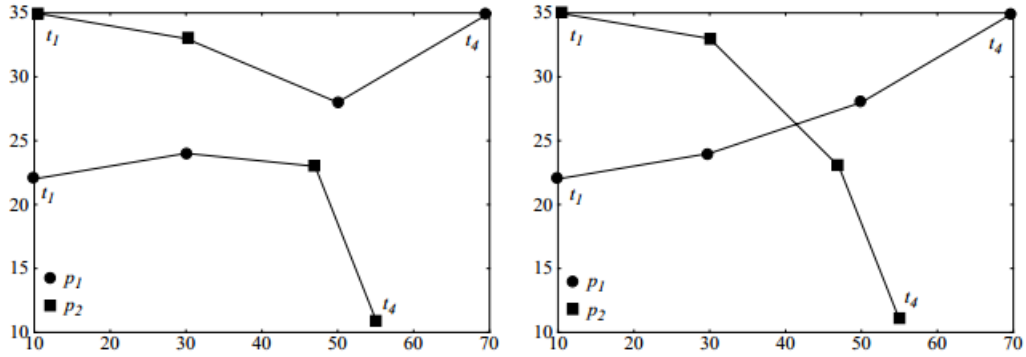


Figura 8.3: Dos marcadores moviéndose en cuatro cuadros, Izquierda corresponde al resultado de aplicar enlazado por elemento más próximo [17], Derecha al elegir el camino con variación más suave.

Para el caso de algoritmos de predicción, algunas soluciones implementadas para tracking de marcadores en desarrollos similares (como el DVIDEOW [9]) son el filtro de Kalman [61] y sus variantes, los cuales requieren inicializar y ajustar parámetros internos. Estos procedimientos fueron implementados en las primeras instancias de estudio del problema, pero se decidió implementar otro procedimiento capaz de corregir los errores de enlazado de forma más simple, de manera tal que el mismo no requiera estudio de parámetros internos para el filtrado pero ingrese alguna restricción a las trayectorias enlazadas.

Como se comentó en los capítulos anteriores, el procedimiento elegido es el presentado por Lorna Herda [14]. El mismo consiste en aplicar el tracking de partículas esbozado por Malik, Dracos y Papantoniou [62] al seguimiento de marcadores. Para ello, se busca el desplazamiento de un marcador desde el cuadro [f] al cuadro siguiente [f+1], sobre una ventana de cuatro cuadros.

La hipótesis principal de este algoritmo, es que el muestreo del movimiento capturado es suficiente para que el desplazamiento entre cuadros sea mínimo en distancia, y la idea para predecir y buscar el desplazamiento entre [f] y [f+1], es utilizar la información que se tiene de la secuencia entre [f-1] y [f], y utilizar una segunda proyección entre [f+1] y [f+2] para confirmar el enlace encontrado en el caso que exista más de una posibilidad (Figura 8.4).

Para poder confirmar una trayectoria de cuatro puntos, se debe cumplir que la misma presenta la menor variación de aceleración para la opción elegida entre todas las posibles. Para ello se realiza el siguiente cálculo:

$$\begin{aligned}\Delta a &= \left|\overrightarrow{\boldsymbol{a}}_{[f][f+1][f+2]} - \overrightarrow{\boldsymbol{a}}_{[f-1][f][f+1]}\right| \\ &= \left|\left(\overrightarrow{\boldsymbol{v}}_{[f+1][f+2]} - \overrightarrow{\boldsymbol{v}}_{[f][f+1]}\right).\Delta t - \left(\overrightarrow{\boldsymbol{v}}_{[f][f+1]} - \overrightarrow{\boldsymbol{v}}_{[f-1][f]}\right).\Delta t\right| \\ &= \left|\left(x_{[f+2]} - 3.x_{[f+1]} + 3.x_{[f]} - x_{[f-1]}\right).\Delta t^2\right|\end{aligned} \tag{8.1}$$

donde $x_{[f+1]}$ a su vez, es aquel punto de [f+1] que mejor se aproxima al desplazamiento en cuadros anteriores, minimizando la Ecuación 8.2.

$$\begin{aligned}\overrightarrow{\boldsymbol{v}}_{[f][f+1]} &= \overrightarrow{\boldsymbol{v}}_{[f-1][f]} \\ x_{[f+1]} - x_{[f]} &= x_{[f]} - x_{[f-1]}\end{aligned} \tag{8.2}$$

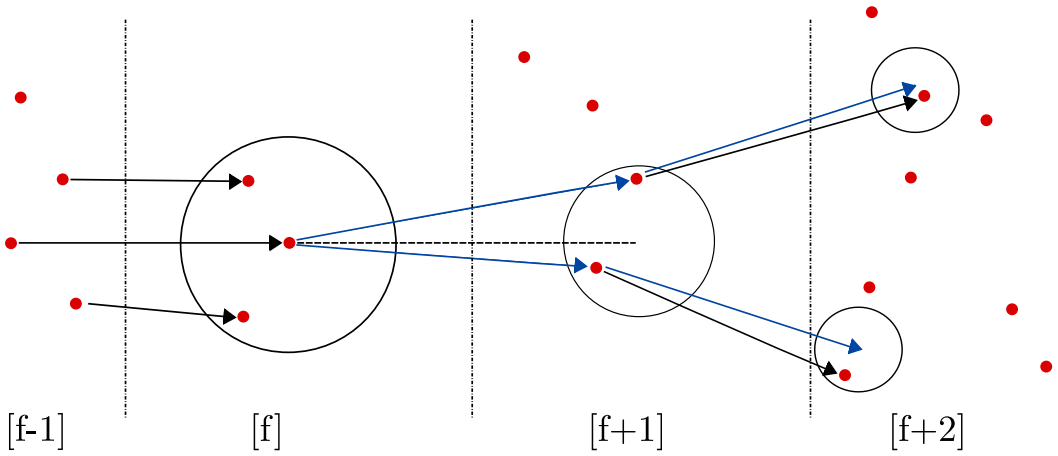


Figura 8.4: Seguimiento en cuatro cuadros, siendo [f] el cuadro actual que queremos seguir en [f+1]. La línea punteada es la continuación del movimiento previo, las líneas azules son las obtenidas buscando la mínima variación de aceleración para el punto elegido en [f+1]. De las dos posibles trayectorias, se elige aquella con menor variación de aceleración, como es visto en Herda [14].

En su trabajo, Lorna Herda afirma que realizar el seguimiento sobre la reconstrucción 3D presenta menos continuidad en las trayectorias, con respecto al seguimiento realizado sobre el conjunto de segmentación sumada a la proyección de los puntos reconstruidos 3D en cada vista 2D. Sin embargo, a raíz de las pruebas realizadas por el grupo, se decidió trabajar con el seguimiento en los puntos reconstruidos en 3D, ya que en caso de trayectorias que se cruzan en una vista 2D, las mismas son fácilmente separables en 3D debido a la geometría presentada.

Además, como se vio en el Capítulo 7, la reconstrucción fue implementada de forma distinta a lo propuesto por Herda. Si bien es posible obtener los puntos proyectados en cada vista, estos no cumplen el mismo rol. Sin implementar la re-proyección, en el tracking sobre una vista individual, se pierden trayectorias

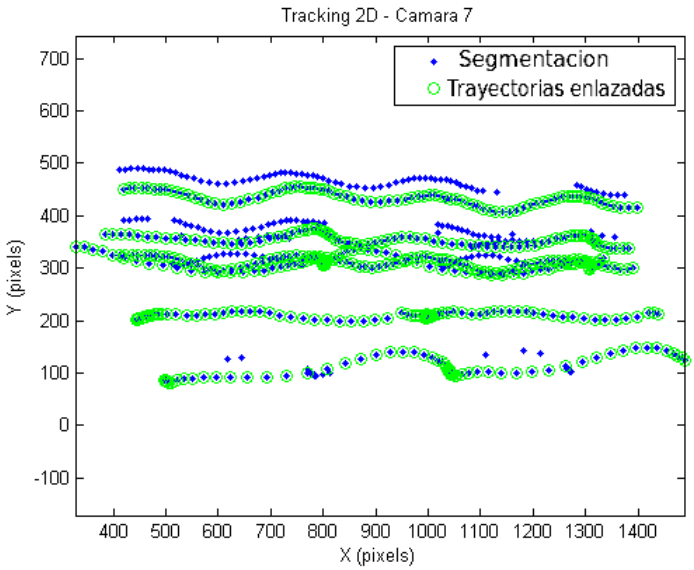


Figura 8.5: Resultado de aplicar el seguimiento de marcadores a una vista 2D sin re-proyección de puntos reconstruidos. Verde, las trayectorias que se enlazaron completamente, Azul los puntos segmentados en la vista 2D de la cámara 07 de la captura sintética.

cuando no se tiene la imagen de un marcador particular durante un cierto número de cuadros, como se observa en la Figura 8.5.

Asumiendo que los puntos obtenidos directamente de las animaciones y proyectados en cada cámara son el mejor caso de puntos 2D, se pudo ver que no presenta grandes ventajas el trabajar el enlace en cada una de estas vistas 2D por separado, frente a trabajar en 3D posteriormente a la reconstrucción y no volver a las vistas 2D, ya que la geometría entre vistas cumplió su cometido.

Esto último deja de ser cierto en el caso que se evalúen medidas adicionales para robustecer la salida del tracking (por ejemplo, en la validación por visibilidad, un punto observado en la segmentación de una vista es descartado en caso de no ser físicamente posible ver ese punto entre el cuerpo y la cámara).

8.3. Implementación

Para un cuadro [f] y un marcador $m_i^{[f]}$, se desea encontrar en el cuadro [f+1] el marcador $m_j^{[f+1]}$ que continúe la trayectoria que se tiene hasta el momento, cumpliendo las ecuaciones de continuidad 8.2 y 8.1. El elemento que traslada la información de un cuadro al siguiente y desde el anterior, es la matriz de enlaces, donde cada línea es un enlace, y cada enlace tiene los siguientes elementos:

$$\left[m_h^{[f-1]},\quad m_i^{[f]},\quad m_j^{[f+1]},\quad m_k^{[f+2]},\quad \left|\overrightarrow{\boldsymbol{a}}_{[f-1][f][f+1]}\right|,\quad \left|\overrightarrow{\boldsymbol{v}}_{[f][f+1]}\right|\right] \tag{8.3}$$

Para el ejemplo presentado en la Figura 8.1, en el cuadro $f = 11$ el marcador $m = 8$ presenta el siguiente enlace hacia $[f+1] = 12$, donde el índice en $[f+2] = 13$

no aplica debido a que no fue necesario para establecer en enlace entre $[f] = 11$ y $[f + 1] = 12$.

$$\left[m_3^{[10]}, \quad m_8^{[11]}, \quad m_3^{[12]}, \quad N/A, \quad \left|\overrightarrow{\boldsymbol{a}}_{[10][11][12]}\right|, \quad \left|\overrightarrow{\boldsymbol{v}}_{[1][12]}\right|\right] \tag{8.4}$$

Al final de cada iteración, la matriz de enlaces es consolidada para asignarle a cada nuevo marcador en [f+1] la etiqueta definida en el primer cuadro del enlazado, la cual puede ser inicializada como texto si se le presenta la opción al usuario (en caso contrario, se utiliza el orden de los marcadores en el primer cuadro). Esta asignación será la que se utilice a lo largo de todo el seguimiento por lo que todo marcador que no sea reconstruido en los primero cuadros, salvo que se tomen medidas adicionales, no aparece en las trayectorias finales.

El procedimiento que se estudia en esta sección para el seguimiento de marcadores es el presentado en la Figura 8.6. A continuación se irán detallando cada uno de los bloques.

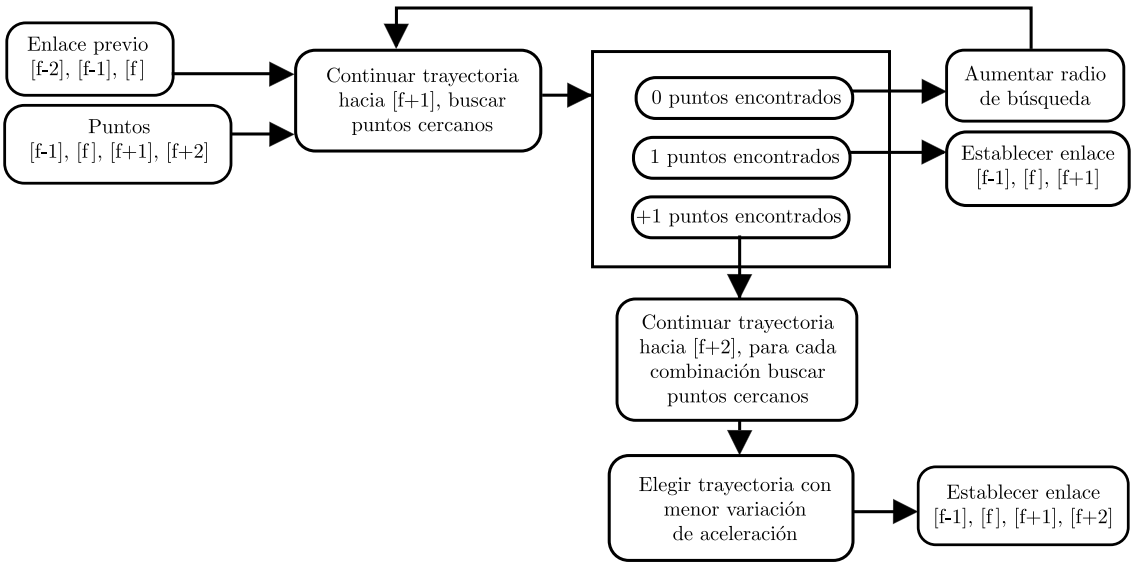


Figura 8.6: Diagrama de Seguimiento de Marcadores.

8.3.1. Enlazado en régimen, inicial y final

Dado un cuadro [f], se cargan todos los marcadores de los cuadros [f-1],[f],[f+1] y [f+2], los cuales son puntos en coordenadas cartesianas X, Y, Z en unidades correspondiente al plano donde se trabaja (píxeles para vistas 2D, metros para espacio 3D). Si el seguimiento desea hacerse para una vista 2D, se establece la tercer coordenada de todos los cuadros en valor predefinido unitario. Dentro de cada cuadro, los marcadores son identificados según su índice en la salida del algoritmo correspondiente, y los enlaces de cada marcador [f-1][f], son cargados a partir de la matriz de enlace del cuadro anterior. La segunda y tercera columna de la matriz de enlace

de [f-1] presenta los mismos datos que la primera y segunda columna de la matriz de enlace en [f], salvo el orden en que es presentada que es asociado al cuadro en curso.

Para el i-esimo marcador en [f], se relevan los índices que componen el enlace [f-1][f] para obtener el traslado previo, y aplicarlo para obtener el centro de búsqueda para el marcador en [f] cumpliendo la Ecuación 8.2 .

$$\begin{aligned} \overrightarrow{\boldsymbol{v}}^{i}_{[f][f+1]} = \overrightarrow{\boldsymbol{v}}^{i}_{[f-1][f]} \Rightarrow C^{i}_{[f+1]} &= x^{i}_{[f]} + \overrightarrow{\boldsymbol{v}}^{i}_{[f-1][f]}.\Delta t \\ &= 2.x^{i}_{[f]} - x^{i}_{[f-1]} \end{aligned} \tag{8.5}$$

La norma de este traslado es utilizada para evaluar los puntos cercanos al centro de búsqueda, donde pueden surgir tres casos:

- Se encuentra un solo punto dentro del radio de búsqueda, en este caso se agrega el índice del punto encontrado en [f+1] a los que se utilizaron de [f-1] y [f], calculando la aceleración y velocidad resultante para establecer el enlace. En este caso, el cuarto elemento de la línea de la matriz de enlace no fue necesario.

- Se encuentra más de un punto, por lo cual se tiene que usar algún criterio para elegir entre todas la posibilidades encontradas. Para cada punto encontrado dentro del radio de búsqueda, se calcula un segundo centro de búsqueda para [f+2], esta vez minimizando la Ecuación 8.1 :

$$\begin{aligned} \overrightarrow{\boldsymbol{a}}_{[f][f+1][f+2]} = \overrightarrow{\boldsymbol{a}}_{[f-1][f][f+1]} \Rightarrow C^{i}_{[f+2]} &= x^{i}_{[f+1]} + \overrightarrow{\boldsymbol{v}}^{i}_{[f][f+1]}.\Delta t + \overrightarrow{\boldsymbol{a}}^{i}_{[f-1][f][f+1]}.\Delta t^2 \\ &= 3.x^{i}_{[f+1]} - 3.x^{i}_{[f]} + x^{i}_{[f-1]} \end{aligned} \tag{8.6}$$

  Siendo el radio de búsqueda en [f+2] la distancia [f][f+1]. Con los puntos encontrados en cada búsqueda, se evalúa la variación de aceleración para los puntos [f-1][f][f+1][f+2], y elige la menor de todas estableciendo el enlace de cuatro puntos. Finalmente, se calcula la aceleración y velocidad del enlace [f-1][f][f+2], y se guardan los índices que permitieron la decisión esta vez con cuatro elementos para indicar que se procedió con la segunda búsqueda.

- No se encuentra ningún punto, tanto para la búsqueda en [f+1] como en [f+2]. En este caso, se presentan múltiples alternativas en distintas etapas. La más inmediata durante el enlazado cuadro a cuadro es aumentar el radio de búsqueda (por ejemplo puede suceder con un punto acelerando fuera del radio de búsqueda). Este aumento puede ser limitado o indefinido hasta encontrar un punto, aunque en una distancia mucho mayor a la esperada, por lo que debe ser validado posteriormente. Sin embargo, aunque el enlace pase la validación dentro del cuadro puede no pasar una validación posterior de trayectoria, esto se discute más adelante.

El enlazado final es similar a lo presentado para régimen, pero en caso de múltiples marcadores candidatos en el cuadro final [f+1], no se puede extender el movimiento, por lo que se elige mediante menor aceleración en los últimos tres cuadros con una sola búsqueda desde [f] a [f+1]. Otra alternativa igualmente válida puede ser buscar sobre los últimos cuatro cuadros; sin embargo, se eligió la primera opción debido a ser un caso simplificado del enlazado en régimen ya implementado.

Por otro lado, el enlazado inicial presenta una situación diferente. No se tiene información previa y deben establecerse las bases para el inicio del enlazado, para ello se cargan todos los marcadores en los primeros tres cuadros y calculan todas las posibles combinaciones de trayectorias entre elementos. En caso de existir combinaciones forzadas inválidas, las mismas presentan una aceleración resultante notoriamente mayor a la de enlaces, pero si corresponden a algún punto erróneo este se perderá naturalmente durante el enlazado.

Una vez enlazado el último cuadro de las secuencia se procede a limpiar aquellas trayectorias que perdieron marcadores en el camino estableciendo un umbral de perdida aceptables sobre el largo de las secuencias obtenidas. Adicionalmente, se descartan todos los puntos reconstruidos que no fueron asignados a ninguna trayectoria.

8.3.2. Validación e Inventario de trayectorias

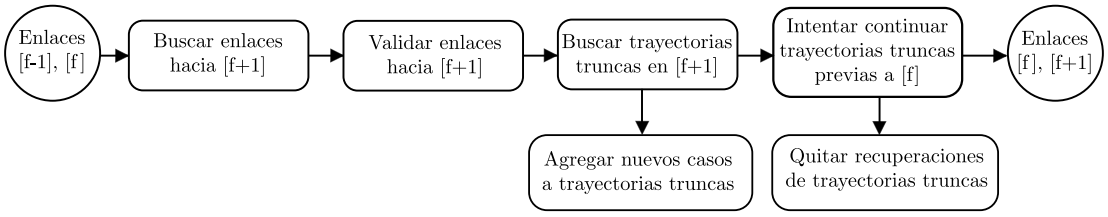


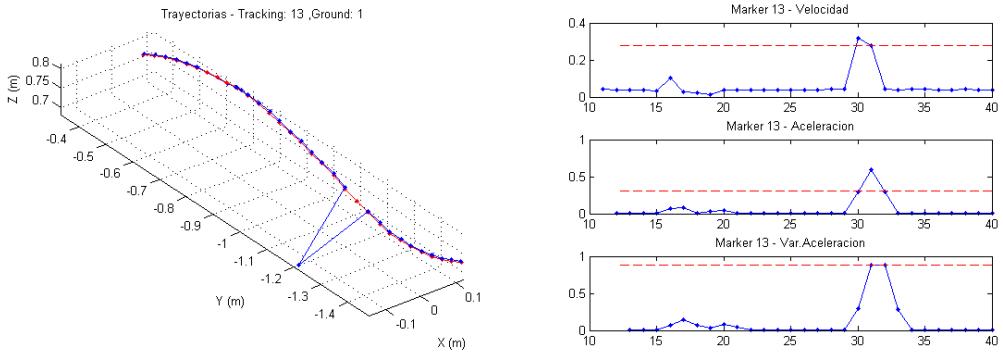
Figura 8.7: Diagrama Seguimiento e Inventario de Marcadores.

Antes de consolidar la matriz de enlaces y proceder a que los marcadores en [f+1] hereden las etiquetas de las trayectorias a las que pertenecen, los enlaces deben ser validados en múltiples instancias y luego de la evaluación, se debe estudiar aquellos casos donde el seguimiento no fue posible. Todo el procedimiento es resumido en la Figura 8.7.

- **Validación dentro del cuadro**. Se verifica que los índices de la matriz de enlaces en [f+1] son únicos, asociados a una sola trayectoria. Si dos o más trayectorias se enlazaron a un mismo punto en [f+1], se compara la aceleración con la que fueron enlazadas, validando aquella con menor aceleración y descartando las otras. En este punto existe la posibilidad de haber agregado todas las trayectorias involucradas en la decisión, en vez de una sola trayectoria por marcador en [f+1] que mejor cumpla las condiciones para ese

marcador. De haber elegido esta opción, el descartar una trayectoria que podría haber sido elegida por menor variación daría paso a la siguiente mejor. Este caso sucede de forma más frecuente al aplicar el seguimiento para trayectorias 2D (no sucede lo mismo en 3D) por lo cual se procedió con solo una trayectoria por marcador la cual, en caso de ser inválida, debe ser evaluada en la etapa de inventario de trayectorias.

- **Validación en trayectoria**. Pueden existir puntos que fueron reconstruidos con leves errores que deben detectarse, corregirse y estimar un reemplazo. Estos puntos son detectados como enlazados correctamente pero con posición, velocidad o aceleración que presenta discontinuidad notoria como puede observarse en el ejemplo de la Figura 8.8.



(a) Trayectoria.

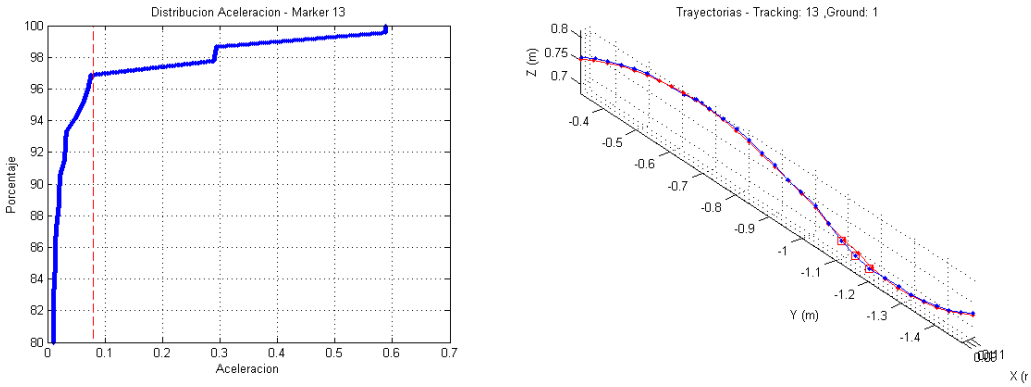
(b) Velocidad, Aceleración, Variación Aceleración.

Figura 8.8: Ejemplo de discontinuidad en un marcador, por punto mal reconstruido. La Figura 8.8a presenta en azul la trayectoria reconstruida, en rojo el ground truth.

Se establece un umbral a partir del estudio de la distribución de la aceleración de cada marcador (Figura 8.9a) para encontrar la aceleración que presenta un salto abrupto y detectar los cuadros donde se supera dicho umbral para corregir el marcador. El procedimiento para estimar los marcadores en estos casos es el mismo que se verá al momento de explicar el inventario de trayectorias.

Como opción general se implementó una alternativa que en vez de calcular un umbral para cada trayectoria establece un umbral global, el cual es útil para detectar aceleraciones notoriamente altas sin tener que estudiar cada trayectoria. Esta opción se logra mediante una ejecución en dos instancias del seguimiento, una primera instancia sin límites globales de aceleración en el enlazado, y una segunda con un límite establecido a un porcentaje de la distribución de la aceleración asumiendo un nivel para pérdidas. A efectos de las pruebas, se verifica que filtrar con el nivel sobre el cual cumplen el 99 %

de las aceleraciones de todos los marcadores enlazados, arroja buenos resultados. Si durante la segunda instancia un enlace global supera este umbral, se descarta y se procede con la recuperación de trayectoria.



(a) Función de distribución de la Aceleración. (b) Trayectoria Corregida.

Figura 8.9: Ejemplo Resultado de Umbral y Corrección En Trayectoria.

- **Inventario de Trayectorias**. Durante el enlazado en un cuadro se tienen marcadores en [f] que no lograron enlazarse en [f+1] pero cuya trayectoria podría estar enlazada hasta este cuadro. Estos puntos deben ser identificados mediante la comparación de la segunda columna de la matriz de enlaces [f][f+1] (actual) y la tercer columna de la matriz de enlaces [f-1][f] (anterior) ya que ambas tienen información de los marcadores en el mismo cuadro pero enlazan hacia atrás y adelante en el tiempo. Los elementos que se encuentren en la matriz [f-1][f] pero no hayan sido enlazados en [f][f+1] son agregados a una lista que contiene el momento y cuadro hasta los cuales se mantiene el enlace continuo (es decir, la última vez que se tuvo información de la trayectoria a la cual pertenece) y qué índice de marcador tiene en ese momento. Pasar de índice en un cuadro a trayectoria a la cual pertenece es trivial, consultando las etiquetas de marcador que se van heredando de cuadro a cuadro. Una vez actualizada la lista de las trayectorias perdidas en [f][f+1] se deben tratar de enlazar los puntos de [f+1] que no entraron en el enlazado regular con las trayectorias perdidas hasta [f-1]. Todos los puntos que sobraron en el enlazado regular son evaluados contra la lista de trayectorias perdidas y se revisan dos condiciones:

  1. **Distancia con respecto a una estimación lineal**. Prolongando el movimiento, se toma el último punto como inicio y los últimos cuadros de la trayectoria conocida como desplazamiento. El desplazamiento es repetido tantas veces como tiempo perdido tiene el marcador: si una trayectoria estaba definida hasta [f-1] y se buscan enlaces con elementos de [f+1], el desplazamiento es repetido dos veces (ver Figura 8.10b).

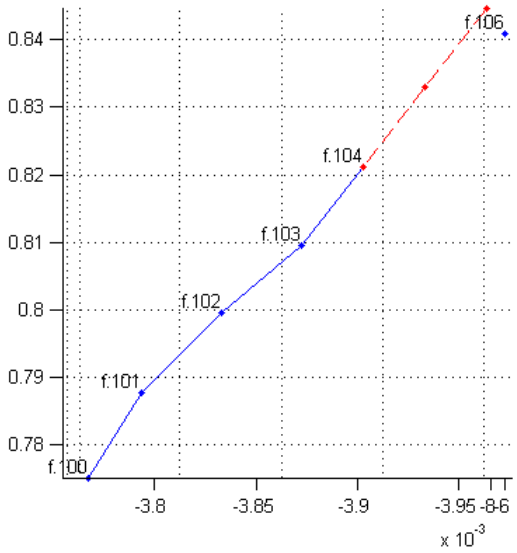
```
************************************************
-----------------------------------------
@(f+1) sobran: 14  19  20
@(f) sobran: 7  19  20
   ---> Perdi marker: 7 @(104), path: 8
-----------------------------------------
(f)(f+1) = (104)(105), enlaces = 17
************************************************
-----------------------------------------
@(f+1) sobran: 3  15  20
trayectoria: 8 ,punto 15@ (106) [-0.00579     -3.96     0.841]
--->dist a estimacion_lineal: 0.00417 ,step: 0.039 ,ratio_direccional: 0.107
trayectoria: 8 ,punto 15@ (106) [-0.00579     -3.96     0.841]
--->dist a ultimo conocido: 0.071 ,step: 0.039 ,ratio_radial: 1.82

posibles_rescates =

    8.0000   15.0000    0.0710

@(f) sobran: 14  19  20
-----------------------------------------
(f)(f+1) = (105)(106), enlaces = 18
************************************************
```

(a) Salida de Código durante perdida y recuperación de trayectoria.



(b) Trayectoria.

Figura 8.10: Trayectoria perdida en cuadro 104, con posible candidato de recuperación en cuadro 106 por cercanía con la estimación por desplazamiento.

2. **Distancia radial con respecto al último punto conocido**. En algunos casos la perdida del enlazado puede suceder cuando la trayectoria está en reposo (por ejemplo, un pie apoyado antes de volver a despegarse del suelo como puede observarse en la cantidad de puntos de estos marcadores en la Figura 8.11) o en transición (en el medio de una curva) por lo cual la estimación lineal anterior es inválida. Para ello es útil

relevar la distancia entre el último punto conocido de una trayectoria trunca y los puntos que no se enlazaron en [f+1]. Esta distancia en caso de una posible recuperación deberá ser proporcional al último traslado conocido por la diferencia temporal de cuadros entre el último punto y el posible candidato.

El procedimiento de validación indica qué marcadores son inválidos tanto usando el umbral global como el individual, pero esta validación puede truncar una trayectoria durante el enlazado que luego debe ser estimada en caso de recuperación o debe ser filtrada para el caso de post-procesamiento. Se debe establecer un procedimiento para tratar de recuperar estos marcadores tratando de cumplir con las restricciones impuestas.

8.3.3. Estimación de Marcadores Perdidos Durante Enlazado

Si se encuentra algún punto que cumpla alguna de las condiciones de recuperación de trayectorias, el mismo es considerado para continuar la trayectoria y deben estimarse los puntos intermedios para completar la misma.

Esta estimación es realizada mediante mínimos cuadrados donde las condiciones iniciales dependen de cuántos puntos de la trayectoria se tienen hasta el momento de la pérdida y tal que la condición final es el punto encontrado en [f+1], las incógnitas son los puntos intermedios que se quieren encontrar y las múltiples ecuaciones a cumplir son las planteadas en (8.1) y (8.2).

Sea $X_{[f+1]}$ un punto candidato por las condiciones anteriores, y se tiene que el último punto conocido en [f-1] de un trayectoria que pretende continuar verifica [f-1]≥3. Entonces, las ecuaciones que el punto a estimar $\tilde{\boldsymbol{X}}_{[\boldsymbol{f}]}$ deberá cumplir son

$$\begin{cases} X_{[f-3]} & -3.X_{[f-2]} & +3.X_{[f-1]} & -\tilde{\boldsymbol{X}}_{[\boldsymbol{f}]} & & = 0 \\ & X_{[f-2]} & -3.X_{[f-1]} & +3.\tilde{\boldsymbol{X}}_{[\boldsymbol{f}]} & -X_{[f+1]} & = 0 \\ & X_{[f-2]} & -2.X_{[f-1]} & +\tilde{\boldsymbol{X}}_{[\boldsymbol{f}]} & & = 0 \\ & & X_{[f-1]} & -2.\tilde{\boldsymbol{X}}_{[\boldsymbol{f}]} & +X_{[f+1]} & = 0 \end{cases} \tag{8.7}$$

Observar que las ecuaciones de variación de aceleración son combinación lineal de las ecuaciones de aceleración.

Escribiendo de forma matricial, separando los datos en incógnitas y condiciones se tiene

$$\begin{pmatrix} 1 & -3 & 3 & -1 & 0 \\ 0 & 1 & -3 & 3 & -1 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} X_{[f-3]} \\ X_{[f-2]} \\ X_{[f-1]} \\ \tilde{\boldsymbol{X}}_{[\boldsymbol{f}]} \\ X_{[f+1]} \end{pmatrix} = \begin{pmatrix} 1 & -3 & 3 \\ 0 & 1 & -3 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_{[f-3]} \\ X_{[f-2]} \\ X_{[f-1]} \end{pmatrix} + \begin{pmatrix} -1 \\ 3 \\ 1 \\ -2 \end{pmatrix} \tilde{\boldsymbol{X}}_{[\boldsymbol{f}]} + \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} X_{[f+1]} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\Rightarrow \underbrace{\begin{pmatrix} -1 \\ 3 \\ 1 \\ -2 \end{pmatrix}}_{A_{[f]}} \tilde{\boldsymbol{X}}_{[\boldsymbol{f}]} = \underbrace{- \begin{pmatrix} 1 & -3 & 3 \\ 0 & 1 & -3 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X_{[f-3]} \\ X_{[f-2]} \\ X_{[f-1]} \end{pmatrix} - \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} X_{[f+1]}}_{B_{[f]}}$$

$$\Rightarrow \tilde{\boldsymbol{X}}_{[\boldsymbol{f}]} = \left(A_{[f]}^t . A_{[f]} \right)^{-1} . A_{[f]}^t . B_{[f]} \tag{8.8}$$

El procedimiento presentado para aproximación de puntos es análogo para el caso de tener n cuadros a estimar entre el último momento f_i de una trayectoria y su posible continuación en f_j con $n+1 = f_j - f_i$. Si $f_i \geq 3$. Se plantearán $n+1$ ecuaciones para variación de aceleración y $n+1$ ecuaciones para variación de velocidad (aceleración) las cuales se resuelven de la misma forma.

$$\begin{cases} X_{[f_i-2]} & -3.X_{[f_i-1]} & +3.X_{[f_i]} & -\tilde{\boldsymbol{X}}_{[\boldsymbol{f_i+1}]} & & & & = 0 \\ & X_{[f_i-1]} & -3.X_{[f_i]} & +3.\tilde{\boldsymbol{X}}_{[\boldsymbol{f_i+1}]} & -\tilde{\boldsymbol{X}}_{[\boldsymbol{f_i+2}]} & & & = 0 \\ & & X_{[f_i]} & -3.\tilde{\boldsymbol{X}}_{[\boldsymbol{f_i+1}]} & +3.\tilde{\boldsymbol{X}}_{[\boldsymbol{f_i+2}]} & -\tilde{\boldsymbol{X}}_{[\boldsymbol{f_i+3}]} & & = 0 \\ & & & \ddots & \ddots & \ddots & \ddots & \\ & & & \tilde{\boldsymbol{X}}_{[\boldsymbol{f_j-3}]} & -3.\tilde{\boldsymbol{X}}_{[\boldsymbol{f_i-j}]} & +3.\tilde{\boldsymbol{X}}_{[\boldsymbol{f_j-1}]} & -X_{[f_j]} & = 0 \end{cases} \tag{8.9}$$

En caso que la cantidad de información previa a la pérdida es menor con $f_i \geq 2$ se plantea una ecuación menos, ya que la que plantea la variación de aceleración con el primer elemento incógnita precisa tres elementos previos mientras las otras ecuaciones no.

Esta metodología de estimación de puntos intermedios es retomada para la validación en trayectorias donde las incógnitas pasan a ser lo puntos que deben ser regularizados debido a presentar aceleración excesiva y discontinua para el marcador (como fue establecido al principio de esta sección).

8.4. Resultados y Análisis

El objetivo del bloque de seguimiento es identificar los puntos reconstruidos, presentarlos como trayectorias, y corregirlos si es necesario. Por esta razón es esperable que para una misma secuencia los resultados sean similares a la reconstrucción, siendo las únicas diferencias leves las correcciones realizadas sobre trayectorias particulares y la posibilidad de medir el error por marcador individual debido a que, a esta altura del proceso, ya están identificados las distintas trayectorias. Los siguientes resultados fueron probados sobre una captura sintética de 113 cuadros, 14 marcadores, con un conjunto de 17 cámaras.

El filtrado y validación por trayectorias permite reducir el error máximo y mejorar el promedio de trayectorias específicas (marcador 8 en la Figura 8.11) pero

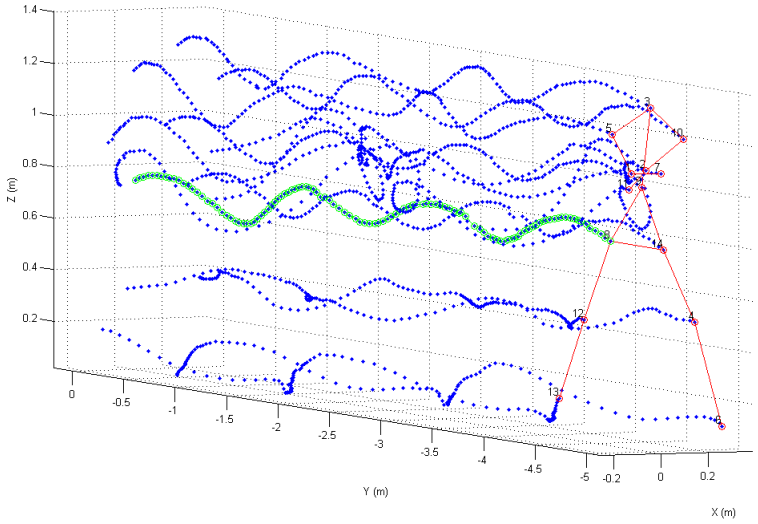


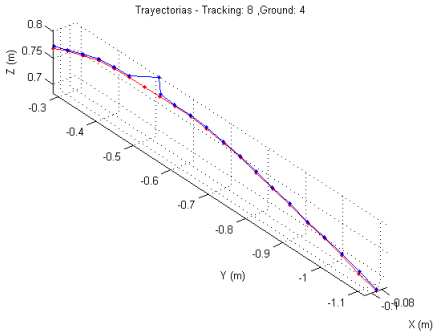
Figura 8.11: Tracking sobre captura sintética, 113 cuadros, 17 cámaras, 14 marcadores.

Marker Track	Marker Ground	Name Ground	Error Promedio(cm)	Percentil 99 %(cm)	Error Promedio (filtrado)(cm)	Percentil 99 % (filtrado)(cm)
12	1	LeftUpLeg	0,3671	0,5158	0,4042	2,0371
3	2	LeftLeg	0,367	0,5411	0,5456	2,7576
2	3	LeftFoot	0,372	0,558	0,377	0,7558
6	4	RightUpLeg	0,3714	0,5879	0,4033	1,6849
4	5	RightLeg	0,378	0,586	0,446	2,1396
14	6	RightFoot	0,4212	1,8483	0,4447	1,8483
9	7	Spine	0,404	0,6043	0,4103	0,7652
7	8	Head	0,3867	0,9063	0,3923	1,2422
8	9	LeftArm	0,3666	0,7997	0,3617	0,594
13	10	LeftForeArm	0,3873	0,9056	0,3961	1,1897
10	11	LeftHand	0,4007	1,1722	0,4128	1,4842
1	12	RightArm	0,4025	1,4771	0,3866	0,944
11	13	RightForeArm	0,3844	0,781	0,3945	0,8591
5	14	RightHand	0,3816	0,7728	0,411	1,1087
		Secuencia	**0,35686**	**0,81266**	**0,38305**	**1,6747**

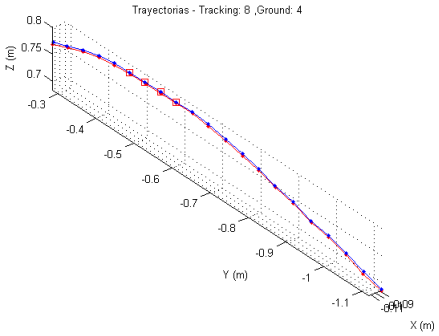
Tabla 8.1: Medida de error de tracking en captura de la base de datos sintética.

al aplicar globalmente a todas las trayectorias puede perjudicar a aquellas que tengan discontinuidades naturales en aceleración debido a variaciones abruptas de movimiento (Tabla 8.1). Para todos los marcadores el error promedio se encuentra por debajo de 0.5 cm para la reconstrucción sobre videos sintéticos de la secuencia CMU-8-07-100-200, con 17 cámaras y con el filtrado por aceleración individual de marcadores. En otras capturas los comportamientos son similares con promedios inferiores a 0.5 cm y errores máximos por debajo de 3 cm, para el caso de reconstrucción con 17 cámaras. Más adelante en el Capítulo 9 se estudiará el impacto de cambiar los conjunto de cámaras en marcadores.

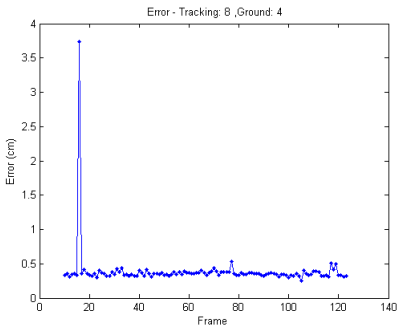
Al tener las etiquetas constantes para las trayectorias es finalmente posible visualizar la relación entre marcadores dejando pendientes medidas adicionales para



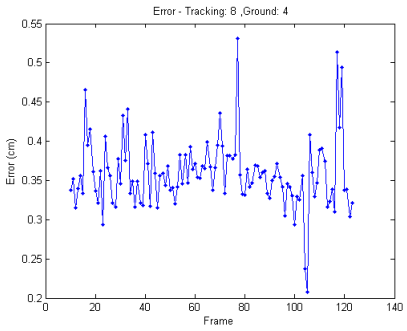
(a) Trayectoria sin Corregir Por Aceleración.



(b) Trayectoria Corregida Por Aceleración.



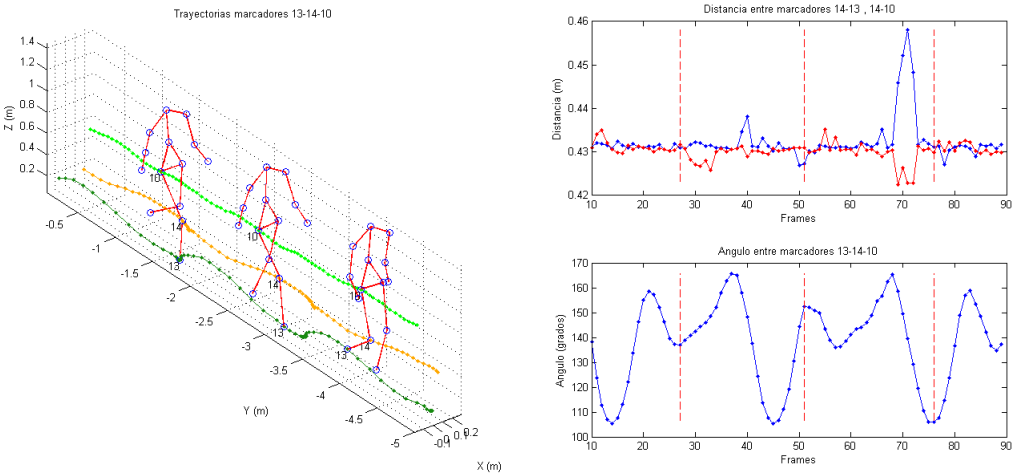
(c) Error de Marcador, sin Corregir.



(d) Error de Marcador, Corregido.

Figura 8.12: Corrección por variación de aceleración, medidas de error antes y después de corregir.

robustecer el sistema como por ejemplo, imponer restricciones al movimiento entre marcadores en un mismo miembro al establecer no solo continuidad en aceleración de marcadores, sino ángulos de articulaciones y huesos. Esta continuidad debe tener cierta tolerancia debido a que los marcadores son representativos de articulaciones y no se mueven totalmente solidarios. Debido a esto, existen leves variaciones en las distancias entre marcadores, observable en la Figura 8.13b correspondiente a los marcadores asociados a la pierna en la captura 8_03_100_100. En este caso se observa la continuidad y periodicidad del ángulo entre marcadores así como la constancia de la distancia, excepto en momentos ocasionales asociados a casos específicos de la captura (cuando el pie rompe el reposo por ejemplo, antes de dar otro paso), sin ser mayor a los pocos centímetros en momentos puntuales y recuperables en régimen.



(a) Trayectorias de marcadores de pierna.

(b) Distancia y Angulo entre marcadores de la pierna.

Figura 8.13: Ejemplos de Posibles restricciones en ángulo y distancia, para el caso de la pierna en marcha.

Capítulo 9

Evaluación

9.1. Medida de Error

Tanto para el caso de la detección como para la reconstrucción se tiene los verdaderos marcadores generados para las secuencias creadas en Blender. A partir de ellos, surge la posibilidad de establecer una medida de error sobre la salida de cada algoritmo y para el sistema en su totalidad. Se elige aplicar una metodología basada en la distancia euclidiana promedio entre los marcadores de ambos conjuntos que también fue utilizada en la base de datos HumanEva [63].

Asumiendo que los conjuntos de datos de salida de algoritmos y ground truth tienen sus cuadros sincronizados temporalmente, en un cuadro [f] dado se tienen M marcadores en un conjunto $\boldsymbol{x}$, $\{m_i(x)\}$, con $i = 1, \ldots, M$ donde $m_i(x) \in \mathbf{R}^3$ (o $\mathbf{R}^2$ para el caso de las vistas de cámaras individuales) y $\tilde{\boldsymbol{x}}$ es el conjunto ground truth con la misma cantidad de marcadores alineados (M) con $\boldsymbol{x}$ (el i-esimo marcador de $\boldsymbol{x}$ se corresponde con el i-esimo marcador de $\tilde{\boldsymbol{x}}$). La distancia se calcula como

$$D(\boldsymbol{x_f}, \tilde{\boldsymbol{x}}_{\boldsymbol{f}}) = \frac{1}{M} \sum_{i=1}^{M} \|m_i^{[f]}(\boldsymbol{x}) - m_i^{[f]}(\tilde{\boldsymbol{x}})\| \tag{9.1}$$

Sin embargo el cálculo debe ser modificado para contemplar el caso en que la cantidad de marcadores a la salida del procesamiento es menor a M. Por lo tanto, se define un conjunto binario en cada cuadro $\Delta = \{\delta_1, \delta_2, \ldots, \delta_M\}$, donde δ_i indica con 1 si el i-ésimo marcador de $\tilde{\boldsymbol{x}}$ fue detectado o 0 en caso contrario. La Ecuación (9.1) queda entonces

$$D(\boldsymbol{x_f}, \tilde{\boldsymbol{x}}_{\boldsymbol{f}}, \Delta_f) = \frac{1}{\sum_{j=1}^{M} \delta_j} \sum_{i=1}^{M} \delta_i . \|m_i^{[f]}(\boldsymbol{x}) - m_i^{[f]}(\tilde{\boldsymbol{x}})\| \tag{9.2}$$

En una secuencia con F cuadros, la performance promedio es calculada como el promedio de los errores en cada cuadro, de la siguiente manera

$$\mu_{secuencia} = \frac{1}{F}\sum_{f=1}^{F} D(\boldsymbol{x_f}, \tilde{\boldsymbol{x}}_{\boldsymbol{f}}, \Delta_f) \tag{9.3}$$

Para poder trabajar con los datos a la salida de la detección y reconstrucción de las secuencias es necesario obtener información adicional para entonces aplicar la Ecuación (9.3). Específicamente, las parejas entre marcadores obtenidos y aquellos en el ground truth que permite alinear y comparar los marcadores y definir cuales fueron detectados.

El emparejamiento es realizado cuadro a cuadro calculando en una matriz la distancia euclidiana entre todos los pares $\{i, j\}$, donde $i = 1, \ldots, M_x$ es un marcador del conjunto $\boldsymbol{x}$ obtenido mediante algoritmo en un cuadro [f], y $j = 1, \ldots, M_{\tilde{x}}$ es un marcador del conjunto $\tilde{\boldsymbol{x}}$ de ground truth:

$$d_{i,j}^{[f]} = \{\|m_i^{[f]}(\boldsymbol{x}) - |m_j^{[f]}(\tilde{\boldsymbol{x}})\|\} \tag{9.4}$$

Una vez calculadas todas las distancias para un cuadro, se buscan aquellas parejas que presenta la menor distancia relevando la pareja (i, j) para la cual se verifica. Luego de obtenida esta distancia, los marcadores (i, j) quedan descartados de la matriz volviendo a buscar la siguiente pareja con distancia mínima hasta que ya no queden elementos para emparejar. Esto sucede en el caso que se generen menos marcadores que los presentes el ground truth, o si todos los marcadores de ground truth ya fueron emparejados y sobran puntos como sucede en la reconstrucción con exceso de marcadores. Trabajar con las parejas en todos los cuadros de la secuencia es análogo a trabajar con la Ecuación (9.2).

Para las medidas de los distintos bloques se trabajará no solo con el promedio de error de secuencia sino también con el percentil 99 (magnitud que cumple que el 99 % de los puntos o marcadores se encuentran por debajo de dicho valor). Se decidió elegir esta representación en vez de la desviación estándar debido a que el error en la distancia no presenta una distribución gaussiana, sino otra distribución donde todos los errores están entre cero e infinito (en este caso no se tiene error infinito debido a que los coeficientes binarios de detección se anulan). El error promedio es el percentil que la mayoría de los puntos detectados presenta y el 99 % es afín al error máximo esperado.

9.2. Performance

9.2.1. Capturas Sintéticas

Las múltiples secuencias sintéticas generadas en Blender presentadas en la Sección 3.4 son ingresadas al sistema utilizando los parámetros de calibración de las cámaras simuladas y los vídeos generados para todas las cámaras. Cada conjunto de datos fue procesado por cada uno de los bloques establecidos en el Capítulo 4

y comparado con el ground truth obtenido por Blender a partir de los vídeos, utilizando la metodología establecida en la Sección 9.1 para medir los resultados a la salida de cada bloque.

Acción	**Nombre Secuencia**	**Longitud de secuencia (segundos)**	**Cuadros por segundo**	**Espacio de captura (metros)**
Marcha rectilínea	08_03_100_100	2,5	30	1,5 x 5,5
Marcha rectilínea, zancada exagerada	08_07_100_100	2,6	24	1,5 x 5,5
	08_07_100_200		48	
Marcha rectilínea lenta, zancada ancha	08_11_100_100	3,8	30	1,5 x 5
Corriendo	09_07_100_100	1,2	24	1,5 x 5
	09_07_100_200		48	
Marcha libre	09_12_100_100	12,5	24	3 x 5

Tabla 9.1: Resumen de secuencias de la base de datos sintética utilizadas para análisis de performance.

Se realizaron pruebas para las 5 secuencias de la Tabla 9.1, los resultados de dichas pruebas se muestran en la Tabla 9.2.

				Segmentación		**Reconstrucción**		**Seguimiento**	
captura	markers	cuadros	n.cams	Promedio (px)	99 % (px)	Promedio (cm)	99 % (cm)	Promedio (cm)	99 % (cm)
08_03_100_100	14	89	17	1,11	**3,68**	**0,41**	2,64	**0,42**	**2,90**
08_07_100_100	14	62	17	1,09	3,12	0,34	1,81	0,34	1,81
08_07_100_200	14	**123**	17	1,09	3,31	0,39	1,27	0,36	1,27
08_11_100_100	13	94	17	1,07	2,90	0,39	**2,87**	0,39	2,87
09_07_100_100	14	29	17	1,13	3,32	0,28	2,04	0,28	2,04
09_07_100_200	14	57	17	**1,14**	3,45	0,35	1,99	0,35	1,99
09_12_100_100	13	**300**	15	1,04	2,12	0,40	1,50	0,40	1,50

Tabla 9.2: Resultados de los bloques, para distintas capturas sintéticas

Las pruebas para todas las cámaras arrojan resultados similares en distintas capturas para el caso de amplia disponibilidad de cámaras, además confirman los resultados y rendimientos presentados para los distintos bloques en las hipótesis planteadas para las capturas sintéticas. La detección de marcadores presenta errores promedios de alrededor del píxel y errores máximos del entorno de los 4 píxeles. La reconstrucción y el seguimiento presentan resultados similares siendo la diferencia conceptual entre ambas el etiquetado de trayectorias. El error medio obtenido es menor a los 0.5 cm y el error máximo se encuentra por debajo de los 5 cm para el caso de reconstrucción con 17 cámaras, teniendo exceso de marcadores (esto es, establecer como parámetro de puntos reconstruidos una cantidad mayor a la esperada).

9.2.2. Ruido En Segmentación

Esta prueba consiste en relevar el error promedio en reconstrucción contra el ground truth 3D para distintas instancias de información de múltiples vistas bajo ruido aleatorio de distintas magnitudes en píxeles en las coordenadas X, Y, evaluando de esta forma como se comporta el sistema frente a estos tipos de ruido.

El ruido es introducido a la entrada del bloque de reconstrucción sobre dos conjuntos de datos análogos correspondientes a la segmentación: las proyecciones 2D sobre cada vista que corresponden al ground truth en esta etapa, y los puntos obtenidos efectivamente al segmentar cada vista individual. Recordando que la diferencia entre estos dos conjuntos es la visibilidad de los marcadores estimada en un 70 % por cámara y el error de detección estimado en 1 píxel, el ruido es inyectado entre la detección y la reconstrucción, evaluando la salida de reconstrucción contra el ground truth 3D utilizando el procedimiento de la Sección 9.1. En la Tabla 3.2 de la Sección 3.4 se estimó el impacto en metros del error en píxeles cuya magnitud es relevada en el caso de la secuencia sintética.

		14 marcadores reconstruidos		**18 marcadores reconstruidos**		**30 marcadores reconstruidos**	
Conjunto	**Ruido (pixels)**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % (cm)**
Ground Cam	0	1,27E-10	1,01E-09	1,27E-10	1,01E-09	1,27E-10	1,01E-09
Segmentación	0	0,38553	1,6915	0,38553	1,6915	0,38305	1,6747
Ground Cam	0,5	1,0404	3,0757	0,5535	2,0504	0,53616	2,1104
Segmentación	0,5	0,76172	2,8548	0,70514	2,096	0,69771	2,0234
Ground Cam	1	1,6511	13,184	1,1398	5,1215	1,0356	3,7874
Segmentación	1	1,3948	9,2686	1,1158	3,8027	1,145	3,8481
Ground Cam	2	4,8746	59,8172	3,4818	53,3151	1,7094	7,4023
Segmentación	2	6,3302	49,8491	3,348	45,4941	2,0409	8,0023
Ground Cam	4	12,7279	131,2929	13,9006	121,5673	3,17	21,153
Segmentación	4	10,7759	67,2878	11,2504	102,6039	4,5432	40,5608
Ground Cam	8	22,6934	136,0204	21,8413	143,0939	8,7161	42,9549
Segmentación	8	25,7256	187,2483	24,2908	158,6828	8,6034	41,6941

Tabla 9.3: Resultados de Error en reconstrucción para distintos caso de ruido agregado a la segmentación, tanto resultados sobre vídeos como sobre ground truth.

Las pruebas confirman los resultados estimados en la Tabla 3.2 para la secuencia dada con cámaras no alejadas más de 10 metros del sujeto. Hasta los 2 píxeles el error promedio se mantiene en el orden de los pocos centímetros con instancias de error máximo mayores (recordar que el bloque de detección ya presenta cierto error propio al cual se le agrega más ruido). Con el agregado que la reconstrucción es implementada con una cantidad variable de marcadores, lo cual permite reconstruir una mayor cantidad de marcadores que los dispuestos sobre el sujeto, en busca de generar la mayor cantidad de posibilidades. A mayor cantidad de marcadores reconstruidos, se obtienen mejores resultados en los casos de mayor ruido, al costo de mayor cantidad de operaciones en reconstrucción.

9.2.3. Variación Cantidad de Cámaras

Hasta ahora todas las pruebas realizadas para los bloques presentados en el Capítulo 4 fueron realizadas sobre el conjunto completo de cámaras presentadas en la Sección 3.4, con 17 puntos de vistas distintos en el espacio de captura de la Figura 3.3. Las pruebas de rendimiento con conjuntos reducidos de cámaras son realizadas sobre los últimos dos bloques (reconstrucción y seguimiento) debido a que no afectan el error en el bloque de detección.

CONJUNTO	**CÁMARAS**
C17	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17
C15	3,4,5,6,7,8,9,11,12,13,14,15,16,17
C8	3,5,7,9,11,13,15,17
C6.1	4,6,8,12,14,16
C6.2	3,6,9,11,14,17
C6.3	3,5,9,11,13,17
C6.4	3,7,9,11,15,17
C5.1	1,3,9,11,17
C5.2	1,4,8,12,16
C4.1	4,8,12,16
C4.2	3,9,11,17

Tabla 9.4: Distintas combinaciones de cámaras utilizadas en el laboratorio sintético, numeradas según la Figura 3.3 que se presenta en la sección Sección 3.4

Las pruebas son realizadas sobre dos secuencias en particular de distintas características de uso del espacio de captura de movimiento.

Captura de Marcha, Secuencia 8_07_100_200. En el movimiento de marcha de la captura 8_07, muestreada al 200 % (48 cuadros por segundo), el sujeto se mueve en el espacio de captura de la Figura 3.3, de la cámara 2 hacia la cámara 10, caminando de forma rectilínea.

Con la máxima cantidad de cámaras, todos los marcadores son reconstruidos y sus trayectorias seguidas a lo largo de la secuencia, se mantienen las medidas de error encontradas hasta el momento, error de marcador menor al centímetro y errores máximos puntuales del orden de los pocos centímetros.

Si se reduce el conjunto de cámaras, manteniendo una "visión doble" desde cada una de las esquinas en el espacio de captura, se conserva un buen rendimiento del sistema con errores similares a los obtenidos anteriormente, con leves mejoras en algunos marcadores pero también leves incrementos de error para otros. Ambas situaciones equilibran el promedio de la secuencia. Estos casos son presentados en la Tabla 9.5.

Reduciendo el conjunto a 6 cámaras comienza a influir no solo la cantidad sino la ubicación de las mismas. En la Tabla 9.6 el conjunto 6.1 posee 6 cámaras, 3 cámaras laterales sobre una misma pared, a izquierda y derecha de la "pasarela"

	CONJUNTO	**C17**		**C8**	
MARKER	**NOMBRE**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % (cm)**
1	LeftUpLeg	0,3671	0,5158	0,3551	0,6197
2	LeftLeg	0,367	0,5411	0,3491	0,4464
3	LeftFoot	0,372	0,558	0,3582	0,5339
4	RightUpLeg	0,3714	0,5879	0,368	0,6825
5	RightLeg	0,378	0,586	0,3627	0,6065
6	RightFoot	0,4212	1,8483	0,3597	0,5667
7	Spine	0,404	0,6043	0,3819	0,5122
8	Head	0,3867	0,9063	0,3676	0,5208
9	LeftArm	0,3666	0,7997	0,3607	0,516
10	LeftForeArm	0,3873	0,9056	0,3621	0,6793
11	LeftHand	0,4007	1,1722	0,3667	0,7895
12	RightArm	0,4025	1,4771	0,3602	0,5078
13	RightForeArm	0,3844	0,781	0,36	0,5199
14	RightHand	0,3816	0,7728	0,3637	0,4884
	Secuencia	**0,35686**	**0,81266**	**0,33603**	**0,53412**

Tabla 9.5: Error de Marcadores en Tracking para conjuntos de 17 y 8 cámaras en el caso de Marcha

	CONJUNTO	**C6.1**		**C6.2**	
MARKER	**NOMBRE**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % cm)**
1	LeftUpLeg	5,821	37,6612	3,6182	45,3281
2	LeftLeg	0,3513	0,816	PERDIDO	PERDIDO
3	LeftFoot	0,3605	1,2945	0,3667	0,8624
4	RightUpLeg	4,1048	58,1895	1,2155	27,1109
5	RightLeg	0,3472	0,4227	0,3412	0,41
6	RightFoot	0,5627	9,3183	0,7806	19,5713
7	Spine	0,664	17,1032	0,4985	5,5806
8	Head	0,3559	0,4546	0,38	0,54
9	LeftArm	0,5959	6,4694	0,6544	9,2929
10	LeftForeArm	0,3472	0,4888	0,3608	0,656
11	LeftHand	0,3519	0,6833	0,3717	0,7004
12	RightArm	0,5879	6,7102	0,5145	9,4856
13	RightForeArm	0,3511	0,6307	0,3653	0,6726
14	RightHand	0,4799	7,4024	0,6051	14,0673
	Secuencia	**1,0095**	**24,3404**	**0,71806**	**17,1131**

Tabla 9.6: Error de Marcadores en Tracking para algunos conjuntos de 6 cámaras en el caso de Marcha

donde se mueve el sujeto. Para este caso se pueden reconstruir la mitad de los marcadores sin problemas, sin embargo la otra mitad presenta errores promedios altos y picos de errores más altos aún. El conjunto de cámaras 6.2 presenta una distribución "hexagonal", tal que se rodea al sujeto en su movimiento, sin embargo

los resultados son similares al conjunto anterior.

	CONJUNTO	**C6.3**		**C6.4**	
MARKER	**NOMBRE**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % (cm)**
1	LeftUpLeg	1,4455	22,4893	PERDIDO	PERDIDO
2	LeftLeg	0,3948	1,7799	0,5296	2,74
3	LeftFoot	0,3824	0,8816	0,3844	0,8658
4	RightUpLeg	1,1137	12,3685	1,4861	15,8273
5	RightLeg	0,4018	1,5785	0,3846	0,6913
6	RightFoot	0,3969	1,2771	0,393	1,3017
7	Spine	0,4152	1,4769	1,2313	10,2902
8	Head	0,384	0,7113	0,384	0,6934
9	LeftArm	0,5909	7,9405	0,3908	0,8389
10	LeftForeArm	0,4258	1,7149	0,3895	0,9638
11	LeftHand	0,3909	1,2141	0,3756	1,0439
12	RightArm	0,3756	0,7621	0,386	0,8189
13	RightForeArm	0,3968	0,9704	0,516	5,8124
14	RightHand	0,617	14,0623	0,7186	11,8042
	Secuencia	**0,51182**	**6,9322**	**0,5512**	**7,4726**

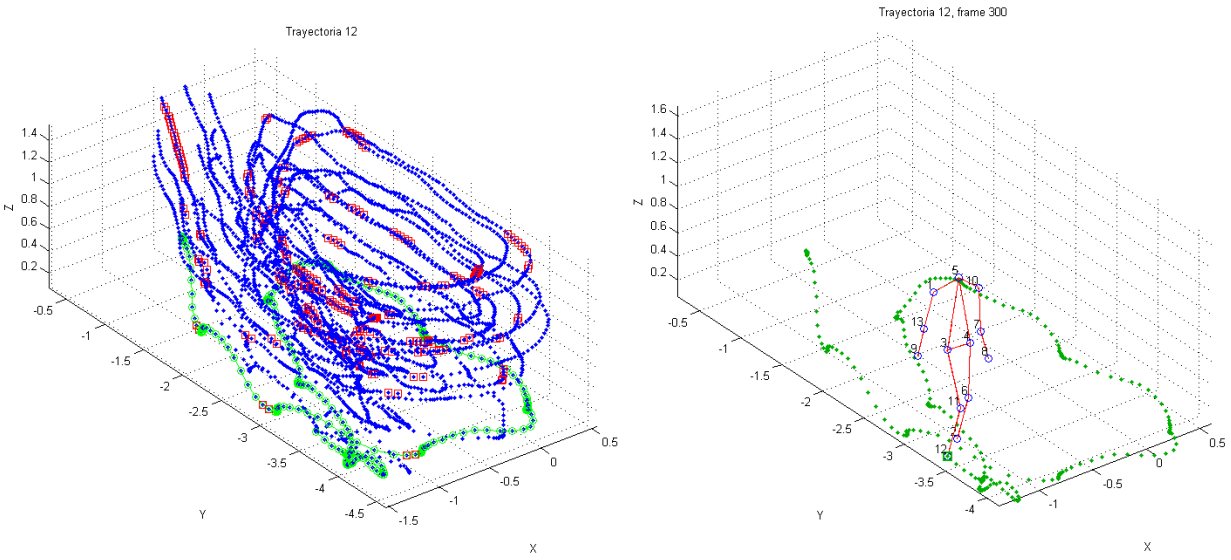
Tabla 9.7: Error de Marcadores en Tracking para algunos conjuntos alternativos de 6 cámaras en el caso de Marcha

En la Tabla 9.7, se presentan los resultados utilizando combinaciones de cámaras, usualmente manejadas en la bibliografía relevada ([64], [63]). Las mismas cuidan en colocar cámaras de manera que no se tenga una fuerte simetría espacial, como ocurre con las configuraciones 6.1 y 6.2. Combinaciones de cámaras con esta geometría presentan resultados notoriamente mejores a las otras combinaciones de 6 cámaras ensayadas anteriormente.

Los conjuntos de 5 cámaras (Apéndice, Tabla C.1) se comportan de la misma forma que los conjuntos de 4 cámaras (Apéndice Tabla C.2), donde solo los marcadores de los pies, hombros y cabeza logran buenos resultados. La única diferencia entre ambos conjuntos es la inclusión o no de la cámara superior.

Captura de Movimiento Libre, 9_12_100_100. La secuencia 9_12 muestreada al 100 % (24fps), presenta un sujeto en movimiento libre caminando en círculos hacia adelante y atrás desplazándose sobre todo el espacio de captura. Debido a que en algunos momentos de la captura se perdían todos los marcadores en algunas cámaras (1, 2, 10) y la reconstrucción no puede procesar una cámara cuando la misma no tiene marcadores en un cuadro, se decide solamente trabajar con las cámaras que tiene marcadores en todos los cuadros de la secuencia dejando como pendiente contemplar este caso en un futuro desarrollo. Las trayectorias obtenidas con el seguimiento y reconstrucción se observan en la Figura 9.1.

Para el caso en el cual se utilizan todas las cámaras, o dos cámaras por esquina (ver Tabla 9.8), se confirman los resultados obtenidos en la captura de marcha,



(a) Trayectorias Movimiento Libre. (b) Trayectoria Marcador, y Esqueleto reconstruido.

Figura 9.1: Trayectorias obtenidas para Movimiento Libre.

	CONJUNTO	C15		C8	
NUMERO	**NOMBRE**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % (cm)**
1	LeftUpLeg	0,4695	2,7781	0,4232	1,2907
2	LeftLeg	0,4176	1,4028	0,4232	2,3317
3	LeftFoot	0,3935	0,714	0,4045	0,6123
4	RightUpLeg	0,4451	2,9214	0,3882	1,4679
5	RightLeg	0,3726	1,0633	0,3475	0,7875
6	RightFoot	0,3633	0,8745	0,3858	2,204
7	Head	0,4068	1,5204	0,369	0,5278
8	LeftArm	0,3951	0,7051	0,4139	1,4362
9	LeftForeArm	0,467	2,9049	0,3852	0,5232
10	LeftHand	0,4259	1,5519	0,3923	0,5664
11	RightArm	0,3845	0,8526	0,4395	1,7055
12	RightForeArm	0,3987	1,696	0,3441	0,5617
13	RightHand	0,3866	1,1216	0,3528	0,7263
	Secuencia	**0,39741**	**1,4952**	**0,37824**	**1,2867**

Tabla 9.8: Error de Marcadores en Tracking para algunos conjuntos de 15 y 8 cámaras en el caso de Movimiento Libre

donde los mismos son buenos para todos los marcadores.

Utilizando los conjuntos de 6 cámaras laterales asimétricas ya estudiado, la Tabla 9.9 muestra que es posible tener un buen rendimiento para casi todos los marcadores. El único marcador con problemas, si bien presenta un buen promedio

	CONJUNTO	**C6.3**		**C4.2**	
NUMERO	**NOMBRE**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % (cm)**
1	LeftUpLeg	0,8081	9,8783	3,9761	23,5115
2	LeftLeg	0,5145	3,5296	0,959	9,4765
3	LeftFoot	0,4198	0,6535	0,9796	17,8756
4	RightUpLeg	0,4243	2,0492	6,5918	36,9034
5	RightLeg	0,3553	1,0051	0,8372	15,4254
6	RightFoot	0,3852	2,2009	1,1164	30,4735
7	Head	0,3803	0,5388	0,378	0,526
8	LeftArm	0,4288	1,3562	1,5435	20,0225
9	LeftForeArm	0,3955	0,5965	0,8621	12,847
10	LeftHand	0,398	0,5513	1,5922	17,1708
11	RightArm	0,5743	3,208	5,5887	25,7882
12	RightForeArm	0,358	0,6621	PERDIDO	PERDIDO
13	RightHand	0,3492	0,6748	PERDIDO	PERDIDO
	Secuencia	**0,43212**	**2,4391**	**1,8061**	**25,7332**

Tabla 9.9: Error de Marcadores en Tracking para algunos conjuntos de 6 y 4 cámaras en el caso de Movimiento Libre

de error, también presenta un pico de error máximo que es necesario corregir.

Finalmente el caso de 4 cámaras presenta resultados bastante malos, con promedios aceptables, pero de gran error máximo y trayectorias que no se logran seguir en la totalidad de la secuencia.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 10

Conclusiones

Se obtuvo en forma íntegra un sistema óptico de captura de movimiento basado en marcadores, que a partir de las capturas de video de una persona en un ambiente de laboratorio con las condiciones adecuadas, obtiene la posición 3D de los marcadores presentes en el cuerpo de dicha persona, logrando representar su movimiento con una precisión del orden del centímetro.

Si bien inicialmente el objetivo era que el sistema funcionara por lo menos para el caso de uso de la marcha, se han probado otros movimientos con resultados aceptables. La aplicación desarrollada permite a partir de múltiples capturas de vídeo de un sujeto en movimiento, detectar los marcadores en cada toma de video. Junto a la información de las cámaras, luego reconstruye la posición de los marcadores en el espacio y finalmente logra identificar cada marcador a lo largo de la secuencia temporal. Cabe destacar que el sistema implementado no es solo óptico, sino que es lo bastante general para funcionar con cualquier sistema de adquisición que genere imágenes, por ejemplo con las imágenes infrarrojas de un sistema Vicon [2].

La revisión bibliográfica inicial permite encontrar los procesos principales que normalmente conforman un sistema de captura de movimiento: *Adquisición*, *Calibración*, *Detección de marcadores*, *Reconstrucción* y *Seguimiento*; obteniendo a su vez una idea del estado del arte de dichos procesos. En dicha etapa se realiza una clasificación de los documentos de acuerdo a la relevancia que prestan y se definen los métodos que posteriormente se utilizaron para implementar los distintos procesos que componen el sistema.

Al relevar las aplicaciones existentes, no se encuentra software de código abierto que se adapte a las características necesarias para utilizar como base sobre la cual montar este proyecto. Por esto se decide implementar los distintos procesos por cuenta propia, teniendo esta etapa por momentos un carácter más de investigación científica que de proyecto ingenieril.

Se efectúa una búsqueda de secuencias de movimiento sobre las cuales desarrollar el sistema, si bien se encontraron numerosas bases de datos, las mismas

terminan siendo descartadas por no ajustarse completamente a las hipótesis que se plantearon en este trabajo. De todas maneras cabe destacar que se genera un relevamiento de bases de datos para el movimiento humano y se profundiza en las características usuales presentes en dichas bases de datos, logrando obtener un conjunto de conceptos y herramientas importantes para el proyecto. Previo análisis de los parámetros involucrados, se enumeran consideraciones a tener en cuenta a la hora de generar un laboratorio de captura. Con ayuda de la suite de animación 3D *Blender* y utilizando fuentes BVH de captura de movimiento disponibles en las distintas bases de datos relevadas, se obtiene un laboratorio de captura de movimiento virtual que permite generar secuencias sintéticas de movimiento con sus respectivos videos. Esto último permite contar con un ground truth sobre el cual validar los algoritmos en cada etapa del sistema así como obtener un benchmark para medir el rendimiento de los mismos. Cabe destacar que el sistema de prueba realizado, permite probar futuros algoritmos. Se implementa una estructura de datos que soporta la información de interés tanto de secuencias sintéticas como reales y un prototipo de base de datos, lo suficientemente general y útil para trabajar con sistemas de captura de movimiento basada en marcadores.

La detección de marcadores en tomas individuales (para cada cámara) se logra mediante segmentación y filtros de objetos de complejidad baja. El error medido para calcular los centros de los marcadores según nuestros requerimientos es bajo, por lo que se considera exitosa la implementación del bloque para el caso sintético. Para el caso real los resultados no fueron tan alentadores, sobre todo porque las secuencias reales que se consiguieron no poseen condiciones de laboratorio favorables para el procesamiento óptico y no se encuentran completamente en las hipótesis planteadas en este proyecto. Sin embargo se pudo realizar una detección aceptable trabajando en conjunto con un extractor de fondo. El ajuste de los parámetros internos a la segmentación permite variar los resultados para poder descartar figuras extrañas a los marcadores que se quieren detectar.

En la etapa de calibración se ha desarrollado un algoritmo que permite realizar la calibración de cámaras a partir de la metodología de captura utilizada por un grupo de investigadores y médicos del Hospital de Clínicas. Por otra parte, de manera independiente, se han analizado dos implementaciones (toolbox) existentes como posibles métodos para calibrar un conjunto de cámaras de un laboratorio de captura de movimiento. Dichos métodos se probaron sobre el laboratorio virtual implementado en el entorno *Blender*. Se concluye que uno de estos toolbox, el *Multi-Camera Self-Calibration Toolbox* [57], presenta mayor flexibilidad de uso en los laboratorios de captura que utilicen un número elevado de cámaras. Se plantea como trabajo futuro realizar una serie de pruebas de *performance* sobre dichas implementaciones y medir sus impactos en las restantes etapas del sistema.

La reconstrucción permite generar los puntos en el espacio siempre que se tenga información para emparejar puntos en dos vistas y confirmar la relación. En las condiciones adecuadas, la reconstrucción cumple con su cometido de generar

los puntos con errores del orden del centímetro, lo cual se considera aceptable para un sistema de captura de movimiento de estas características. Las secuencias reales disponibles inicialmente no tuvieron una performance aceptable, por lo que se tuvo que re-formular el algoritmo realizado inicialmente para las secuencias sintéticas. Si bien los problemas de detección de marcadores y calibración influyen considerablemente en la performance de la reconstrucción, un análisis posterior y pruebas realizadas en secuencias sintéticas que simulaban el caso real, revelaron que el número de cámaras cambia significativamente el problema, volviendo obsoleto el algoritmo inicialmente propuesto cuando se trabaja con menos de 6 cámaras. Por lo que se implementa un nuevo algoritmo que permite trabajar con las 3 cámaras de la secuencia real. Finalmente en conjunto ambos algoritmos solucionan los problemas de reconstrucción al variar el número de cámaras, permitiendo trabajar con capturas de al menos tres cámaras de manera aceptable. Actualmente los algoritmos de reconstrucción propuestos son generales y no introducen restricciones sobre los marcadores. Se propone utilizar en futuras revisiones, la relación presente entre marcadores adyacentes en capturas de movimiento de personas, con el fin de incrementar la performance.

Se implementó un algoritmo de seguimiento basado en restricciones de velocidad y trayectorias, asumiendo la hipótesis de correcto muestreo de la secuencia y el buen desempeño del proceso de reconstrucción. Si las pérdidas son puntuales y no se mantienen en el tiempo, se implementaron medidas para recuperar trayectorias continuando el movimiento. Los resultados de performance de seguimiento son los mismos que los de reconstrucción pero se pueden medir los errores de manera más específica, teniendo para cada marcador el mismo rendimiento satisfactorio que para el conjunto entero y toda posible discontinuidad puede ser detectada y reparada para mantener la continuidad tanto de velocidad como de aceleración.

Es importante tener presente los pasos que restan para tener una base de datos con secuencias reales. En este trabajo se exploran en detalle distintos factores a tomar en cuenta a la hora de armar un laboratorio de captura y se efectúan una serie de recomendaciones o definiciones, tanto para su armado como para la realización de capturas en condiciones óptimas para un posterior procesamiento. También se define una estructura que almacena toda la información recabada en dichas capturas y permite trabajar a lo largo de los diferentes procesos que componen el sistema de procesamiento. Por lo que para obtener una base de datos real con secuencias ópticas, solo resta efectuar las capturas de acuerdo a las condiciones que se definen y calibrar las cámaras. Con estas nuevas secuencias y una posterior evaluación se pueden generar mejoras en los algoritmos del sistema.

Cabe destacar que se está aumentando la reproducibilidad en el área. Pues se cuenta con un sistema completo y estructurado donde fácilmente se pueden modificar, ingresar y probar distintas partes, comparando sus desempeños con las métricas ya definidas en el Capítulo 9.

El sistema actual devuelve toda la información de los marcadores relevante para el usuario final en una matriz de Matlab, esto no es un problema para los investigadores en Biomecánica, pues los mismos efectúan habitualmente sus cálculos en dicho programa. Fácilmente se pueden efectuar cálculos de centro de masa, velocidades y aceleraciones, tanto parciales como totales. También se actualiza y devuelve toda la información generada a lo largo del procesamiento en estructuras de Matlab así como en archivos .xml. Si bien se desarrollaron herramientas de visualización de secuencias de movimiento en Matlab, las mismas son básicas. Por lo que en futuros trabajos se pueden generalizar sus prestaciones de manera más acorde al especialista. Otra alternativa con bastante potencial es exportar la información de salida del sistema nuevamente al entorno Blender, de manera tal que el especialista realice un análisis más interactivo sobre un modelo virtual.

Como trabajo a futuro queda explorar las opciones para robustecer el sistema en condiciones más exigentes asociadas al caso real, por otro lado implementar las medidas adicionales con restricciones para mantener coherencia asociada a un modelo físico, ampliar la interfaz gráfica para cada módulo, haciendo la misma más amigable al usuario y/o orientada al experto (calibración interactiva, bloque de visualización de resultados, entre otros). Si bien esta primera versión del sistema no está a la altura de los sistemas comerciales relevados en este documento, se logra dar el primer paso, sentando las bases y condiciones necesarias para poder continuar con el proyecto. El potencial actual tanto del sistema como de la base de datos y su posible expansión permiten afirmar que se generaron herramientas a tener en cuenta en futuros proyectos de captura de movimiento.

Apéndice A

Clasificación de la bibliografía recopilada

A continuación se muestra una tabla con la clasificación de los documentos recopilados durante la etapa de investigación en este proyecto.

Los mismos están clasificados según las etapas que componen el sistema y tienen un código de color de acuerdo a la relevancia que tienen para el proyecto:

- Verde: el documento contiene información de valor para el proyecto, por lo que debe tenerse en cuenta.
- Amarillo: la información contenida en el documento, si bien está relacionada con la categoría en la cual está ubicada, no contiene la metodología elegida para la implementación, o no aporta información de valor.

Además, se agregan comentarios sobre el contenido de los documentos, el año en que fueron publicados, cantidad de citas en otras publicaciones, así como ventajas y desventajas de la metodología aplicada.

		Mayor relevancia para el proyecto			
		Menor relevancia para el proyecto			
ARTÍCULO	**CALIBRACIÓN**	**ADQUISICIÓN**	**SEGMENTACIÓN**	**TRACKING**	**ESTIMACIÓN DE POSE**
Skeleton-Based Motion Capture for Robust Reconstruction of Human Motion		Sistema Propietario Elite	Sistema Propietario Elite	3/8 vistas para tracking. usa puntos anteriores para estimar/verificar el siguiente, filtro de particulas para enlazar. Esqueleto Pre-Definido, a inicializar	Pasa 2D a reconstruccion 3D
Aplicación de histogramas para detección de cambios en imágenes			OTSU		
Método de valor umbral			Muy buen algoritmo		
Detección y seguimiento de objetos en entornos dinámicos mediante estimación predictiva del flujo óptico.			USAN method	emparejan marcadores de 2 frames consecutivos por proximidad geometrica y predictor de Kalman, usan flujo optico tambien	
Investigación e implementación de un sistema de seguimiento de objetos basado en métodos de segmentación a través de level sets.			level sets method	nombra varios algoritmos de predicción pero no explica ninguno	
Detección, rastreo y reconstrucción tridimensional de marcadores pasivos para análisis de movimiento humano.	-->Grupo 3 cámaras (matriz fundamental, geometría epipolar y reproyección). -->Grupo 2 cámaras (matriz fundamental y geometría epipolar) --> DLT (Direct Linear Transformation)	--> 4 cámaras Fastec Imaging (250 frames por segundo)	-->Algoritmo detección movimiento (3 pasos) -->Algoritmo extracción de características(3 pasos)	-->Suponen velocidad y aceleración de particulas limitada (generar busqueda cónica de proxima posición) -->Ventana móvil de 4 frames	
Development of affordable optical Based Gait Analysis Systems	--> DLT (Direct Linear Transformation)	-->Usan Leds como marcadores -->Cámaras de 25 y 90 fps --->Utilizan filtro optico para manejar iluminación en las cámaras		-->Distancia Euclidiana para predicción -->Interpolación lineal cuando ocurre oclusión	
Marker Detection and trayectory generation algorithms for a multicamera based gait analysis system	-->corrige distorsión del lente -->calibración con 4 varillas colgantes con 6 marcadores cada una	-->6 cámaras CCD a 50hz -->2 plataformas de fuerza -->flash infrarojo -->EMG -->unidad de sincronización	-->agrupa y detecta pixeles con nearest neighbourhood -->remueven "pixeles ruidosos" -->trabaja solapamiento de marcadores	-->Primero enlazado espacial 2D (al terminar ese proceso obtienen nube 3D de todos los puntos frame a frame) -->Luego enlazado temporal 3D (generan trayectorias)	DLT
Analisis de video para estimacion del movimiento humano una revision					
Seguimiento en tiempo real de objetos sobre secuencia de imágenes empleando dispositivos de lógica programable.		analisis de la iluminacion, camaras LB100, etc	filtros matched ,segmentacion por color, metodos para reducir el ruido, level set	flujo optico, filtros de particulas, redes neuronales, algoritmos de transformacion afin	
Simple and robust hard cut detection using interframe differences.			pixeles, histogramas, block matching, object segmentation,	tracking and feature tracking based methods of shot boundary detection	
Treshold survey			evalúa 40 métodos de umbral sacando conclusiones para cada uno. Muy bueno		
Modelling and Tracking Articulated Motion from Multiple Camera Views		3 cámaras		Extended Kalman Filter, estimación para cada frame de los parámatros de un modelo de esqueleto. Como el modelo es 3D, implícitamente se reconstruye la pose al mismo tiempo	
Optical Motion Capture System with Pan-Tilt Camera Tracking and Realtime Data Processing	DLT	262[fps](512x512pixels, 8bit) monochrome CCD camera		Polyhedra Search Algorithm	
Skeletal Parameter Estimation from Optical Motion Capture Data	Vicon , PhaseSpace			Se hace el tracking mediante una distancia no eucliadiana. Luego se ajusta un un esqueleto a los marcadores detectados	
Optical Motion Capture System with Pan-Tilt Camera Tracking and Realtime Data Processing		2 camaras, res 752×480, 30 Hz		Extended Kalman filter + geometria epipolar	
Resolving Motion Correspondence for Densely Moving Points				Tracking 2D, 3 metodos clasicos, contra uno nuevo propio original	
What can two images tell us about a third one?					Paper Elemental de Reconstruccion 3D

ARTÍCULO	**Año**	**Citas (Citerseer)**	**Citas (Google)**	**Pros?**	**Contras?**
Skeleton-Based Motion Capture for Robust Reconstruction of Human Motion	31/01/2000	124	Using Skeleton-Based Tracking to Increase the Reliability of Optical Motion Capture	Sistema cuasi completo, filtro particulas para enlazar puntos. Muy mencionado en varios papers. Optimizado posteriormente en 2001 "Using Skeleton-Based Tracking to Increase the Reliability of Optical Motion Capture" 99 citas	Sistema propietaio de Adquisicion
Aplicación de histogramas para detección de cambios en imágenes					
Método de valor umbral					
Detección y seguimiento de objetos en entornos dinámicos mediante estimación predictiva del flujo óptico.					
Investigación e implementación de un sistema de seguimiento de objetos basado en métodos de segmentación a través de level sets.					
Detección, rastreo y reconstrucción tridimensional de marcadores pasivos para análisis de movimiento humano.	11/12/2009	0	2	Utilizan un método para medición de incertidumbre. Usa referencias conocidas	-Problemas cuando se cruzan marcadores y en periodos de oclusión largos
Development of affordable optical Based Gait Analysis Systems	01/05/2012 -Present	0	1	Este paper muestra lo que los autores estuvieron investigando desde hace algunos años, Han liberado papers que tratan el problema de manera incremental, incluso tratamiento y analisis de datos para especialistas en biomecanica una vez obtenida los datos 3D	Hacen referencia a una base de datos pero no se encuentra disponible
Marker Detection and trayectory generation algorithms for a multicamera based gait analysis system	2001		14	-->Describe sistema completo para detección de marcadores,Orientado a adquisicion de marcha, ver ultimas figuras similares a los datos generados en Blender, mucho énfasis en segmentacion	-->No introducen información del esqueleto .Puede llegar a introducir interacción con el usuario al final del proceso para correcciones. Sincronizacion temporal es confusa. Dificultad para marcadores cercanos
Analisis de video para estimacion del movimiento humano una revision	15/04/2009			-->Buen resumen para introducirse en el tema que nos atañe	-->Solo es un resumen
Seguimiento en tiempo real de objetos sobre secuencia de imágenes empleando dispositivos de lógica programable.				Expone varias referencias de libros que parecen ser buenos, tiene metodos para reducir el ruido,técnicas de estimacion de campo de movimiento, documentaron las pruebas que hicieron, información bastante útil.	
Simple and robust hard cut detection using interframe differences.				muestra varias opciones	no dice mucho de cada una de ellas.
Treshold survey					
Modelling and Tracking Articulated Motion from Multiple Camera Views	2000	6	36	- sistema orientado a pocas cámaras. Usando la información de todas las cámaras para reconstruir resolvería las oclusiones	
Optical Motion Capture System with Pan-Tilt Camera Tracking and Realtime Data Processing	2002	0	70		- orientado fuertemente a real-time, robustez tracking? pan- tilt camera ?
Skeletal Parameter Estimation from Optical Motion Capture Data	2005	35	120		-escueto en explicaciones formales aunque hay algunas referencias a otros articulos
Optical Motion Capture System with Pan-Tilt Camera Tracking and Realtime Data Processing	2008	0	1		
Resolving Motion Correspondence for Densely Moving Points	2001		293	Tracking 2D, no usa informacion de esqueleto. Estaditico, costos de asociar puntos a secuencia de tracking	Comparacion con otros algoritmos, años 90, performance, complejidad, costo operacional, tracking 2D, sin mencion 3D
What can two images tell us about a third one?	1993		131	Mencionado en varios papers	

Apéndice B

Interfaz gráfica (GUI)

Se implementó una interfaz gráfica de forma tal de ejecutar el sistema de forma práctica. La misma permite probar los procesos de manera aislada o encadenada con el resto.

Esto último permite obtener los datos de salida de cada bloque sin necesidad de ejecutar el proceso en su totalidad, ahorrando tiempo de procesamiento.

Como era de esperar, la interfaz gráfica tiene, entre otras cosas, todos los parámetros que se le ingresan al proceso en cada módulo. Por ejemplo: umbral fijo y filtro de área en el bloque de detección, marcadores totales, cámaras a utilizar en la reconstrucción, o el rango de frames donde se procesarán los marcadores.

En la Figura B.1 se observa una captura de pantalla de la interfaz implementada.

Cabe destacar que esta interfaz no pretende ser la interfaz final de la aplicación, sino un bosquejo, ya que el estado actual del sistema no permite que sea definido como "aplicación de usuario" sino como el estudio e implementación de un sistema de captura de movimiento diseñado previamente.

Queda como pendiente, preferiblemente para un proyecto de ingeniería de sistemas, diseñar una interfaz de usuario completa, amigable y con mejor usabilidad para los especialistas que utilizarán el sistema. Sin embargo en la Figura B.2 se propone la estructura final de la misma.

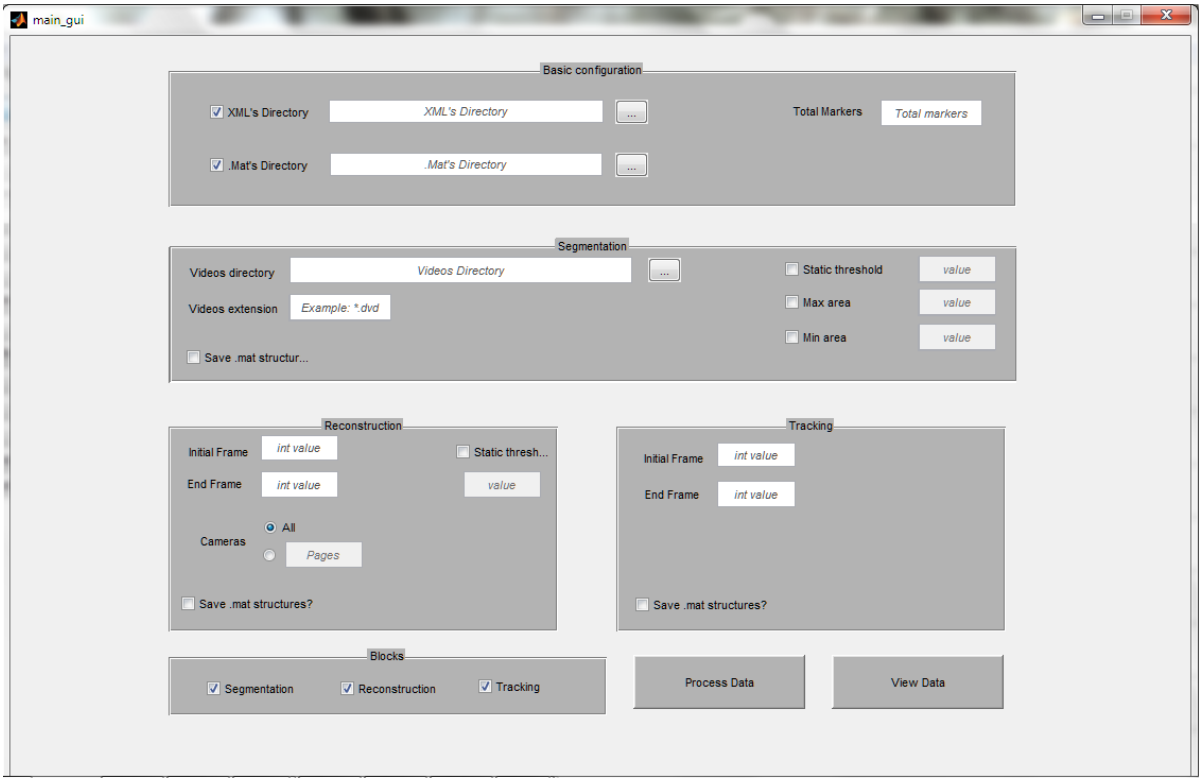


Figura B.1: Vista de la interfaz gráfica implementada.

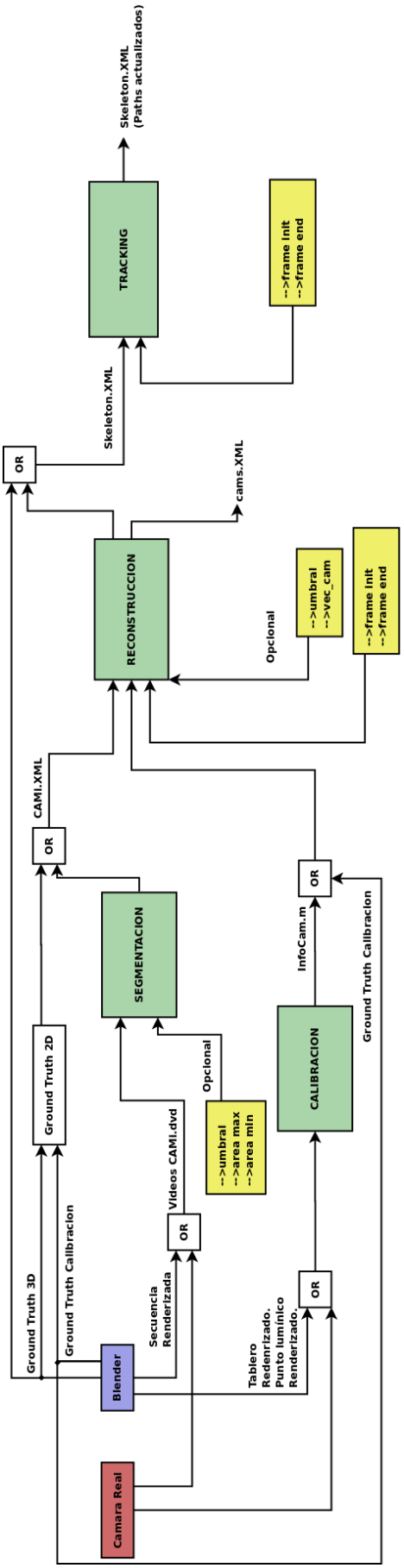


Figura B.2: Estructura propuesta para la interfaz gráfica. Los bloques amarillos son parámetros de entrada y los verdes procesos del sistema.

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Performance en conjuntos de 5 y 4 cámaras

	CONJUNTO	**C5.1**	**C5.1**	**C5.2**	**C5.2**
MARKER	**NOMBRE**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % (cm)**
1	LeftUpLeg	PERDIDO	PERDIDO	PERDIDO	PERDIDO
2	LeftLeg	0,4295	2,3917	0,8971	22,988
3	LeftFoot	0,4147	0,7458	0,5518	8,6042
4	RightUpLeg	4,2922	58,0209	PERDIDO	PERDIDO
5	RightLeg	3,3777	61,5332	0,3934	1,5824
6	RightFoot	PERDIDO	PERDIDO	0,493	4,8055
7	Spine	18,9827	50,6854	16,6915	35,0544
8	Head	9,2659	22,7929	PERDIDO	PERDIDO
9	LeftArm	0,5064	4,9531	0,3648	0,5347
10	LeftForeArm	0,6575	10,2007	0,7586	13,8037
11	LeftHand	0,6831	13,8355	1,5395	25,0238
12	RightArm	0,7107	14,7915	0,6703	7,409
13	RightForeArm	2,3882	34,3833	PERDIDO	PERDIDO
14	RightHand	1,906	38,9179	PERDIDO	PERDIDO
	Secuencia	**3,0121**	**40,035**	**1,0513**	**20,5912**

Tabla C.1: Error de Marcadores en Tracking para conjuntos de 5 cámaras en el caso de Marcha

	CONJUNTO	**C4.1**		**C4.2**	
MARKER	**NOMBRE**	**Promedio (cm)**	**99 % (cm)**	**Promedio (cm)**	**99 % (cm)**
1	LeftUpLeg	PERDIDO	PERDIDO	2,167	31,4114
2	LeftLeg	0,565	9,7164	1,1854	26,0476
3	LeftFoot	0,38	0,9696	0,4273	0,7269
4	RightUpLeg	PERDIDO	PERDIDO	0,8181	13,8336
5	RightLeg	0,6517	13,5	PERDIDO	PERDIDO
6	RightFoot	0,4682	4,8055	0,4841	1,1391
7	Spine	3,1591	30,8635	3,7266	29,8362
8	Head	0,3595	0,4278	0,388	0,54
9	LeftArm	0,3609	0,5417	0,4155	0,5588
10	LeftForeArm	2,8277	25,0512	2,8636	17,1966
11	LeftHand	0,6116	9,755	0,4814	3,8581
12	RightArm	0,6862	11,1969	0,6188	12,4351
13	RightForeArm	2,0736	21,7146	2,2073	25,8873
14	RightHand	0,9646	22,845	PERDIDO	PERDIDO
	Secuencia	**1,3662**	**23,0206**	**1,1976**	**22,5326**

Tabla C.2: Error de Marcadores en Tracking para conjuntos de 4 cámaras en el caso de Marcha

Apéndice D

Manual de usuario

D.1. Requerimientos

Si bien el sistema desarrollado se ha probado en sistemas Linux y Windows, a continuación se enumera los programas y librerías necesarias para una instalación en el sistema operativo Ubuntu 14.04.

- Instalar *Matlab R2011b*.
- Desde el repositorio local los siguientes paquetes:
  - `matlab-support`
  - `matlab-support-dev`
- *OpenCV* desde librerías del repositorio local:
  - libcv-dev
  - libopencv-dev
- Instalar *cvblob* [41]. Descargar `cvblob-0.10.4-src.tgz` y seguir el instructivo de la página para una instalación Linux.
- Copiar la carpeta del proyecto `Proyecto_Biomecanica`.
- Compilar el programa que efectúa la segmentación. Para ello ir a la carpeta `Proyecto_Biomecanica/segmentacion` y crear una carpeta `/build`. Dentro de ella abrir una terminal y ejecutar:

  ```
  cmake ..
  make
  ```

  A continuación en `Proyecto_Biomecanica/segmentacion/build/client` se tiene el ejecutable con el nombre `Source`, el mismo se debe llevar a la carpeta `Proyecto_Biomecanica/SISTEMA/ProgramaC` con el nombre `Source_GLNX86` si se instalo un sistema de 32 bits o `Source_GLNXA64` en caso contrario.
- Instalar *Blender* desde los repositorios locales.

D.2. Generación de secuencias sintéticas

A continuación se resumen los pasos a seguir para generar una secuencia en el laboratorio virtual a partir de archivos BVH de la base de datos *MotionBuilder-friendly version* ofrecidas por *cgspeed* [36], que provienen de las capturas de Carnegie Mellon University Motion Capture Database [26]. La secuencia generada se almacena en la base de datos del proyecto.

1. Seleccionar el movimiento que va a tener la secuencia, se tiene una lista disponible en *cgspeed*[1].

2. Preparar la secuencia. Para ello se utiliza el programa bvhacker[2] [37], al abrir la secuencia se quita el offset y se efectúa un centrado de la misma, con las opciones `NoOffset` y `Center` respectivamente, luego se guarda convenientemente `File/Save`.

3. Activar la importación de archivos BVH en el menú de preferencias del entorno *Blender*, `File/User_Preferences/Addons/Import-Export_bvh`.

4. Crear las carpetas correspondientes en la base de datos. Para ello correr el script `Base_de_datos/nueva_secuencia.sh` . Dicho script de `Bash` ejecuta dentro de *Blender* los script `import_bvh.py` y `acoplar_modelo.py`, que permiten respectivamente importar el archivo BVH previamente preparado y acoplarlo con el modelo y los marcadores.

5. Abrir el archivo .blend de la nueva secuencia y efectuar los ajustes finos del modelo virtual sobre el esqueleto. El modelo virtual se encuentra en el `layer 1`, los marcadores en el `layer 2` y el esqueleto en el `layer 3`. Solo se debe modificar el modelo virtual.

6. Por último para renderizar la secuencia del laboratorio virtual, se debe seleccionar apropiadamente las propiedades de las cámaras en el menú `Properties/Render`, seleccionar únicamente el `layer 1` y el `layer 2` con el modelo virtual y los marcadores para que sean visibles en pantalla y luego abrir en el editor de texto de *Blender* el script `render.py` y ejecutarlo. De esta manera automáticamente se generan y almacenan los videos en la estructura de datos de la base de datos.

D.3. Procesamiento de datos utilizando interfaz gráfica

Una vez abierto *Matlab* con el workspace en `/Proyecto_Biomecanica/SISTEMA`, se debe ejecutar el script `main.m`. El mismo se encarga de cargar los `paths` necesa-

[1] https://sites.google.com/a/cgspeed.com/cgspeed/motion-capture/cmu-bvh-conversion/bvh-conversion-release---motions-list. Accedido 4-03-15

[2] Se ha probado la versión 1.8.0.6 sobre wine 1.6.2 simulando este último a Windows XP

rios, abrir `matlabpool`[3] para utilizar múltiples procesadores y mostrar la interfaz de usuario.

La interfaz se divide en 6 zonas principales:

1. Basic configuration
2. Segmentation
3. Reconstruction
4. Tracking
5. Blocks
6. Process Data, View Data

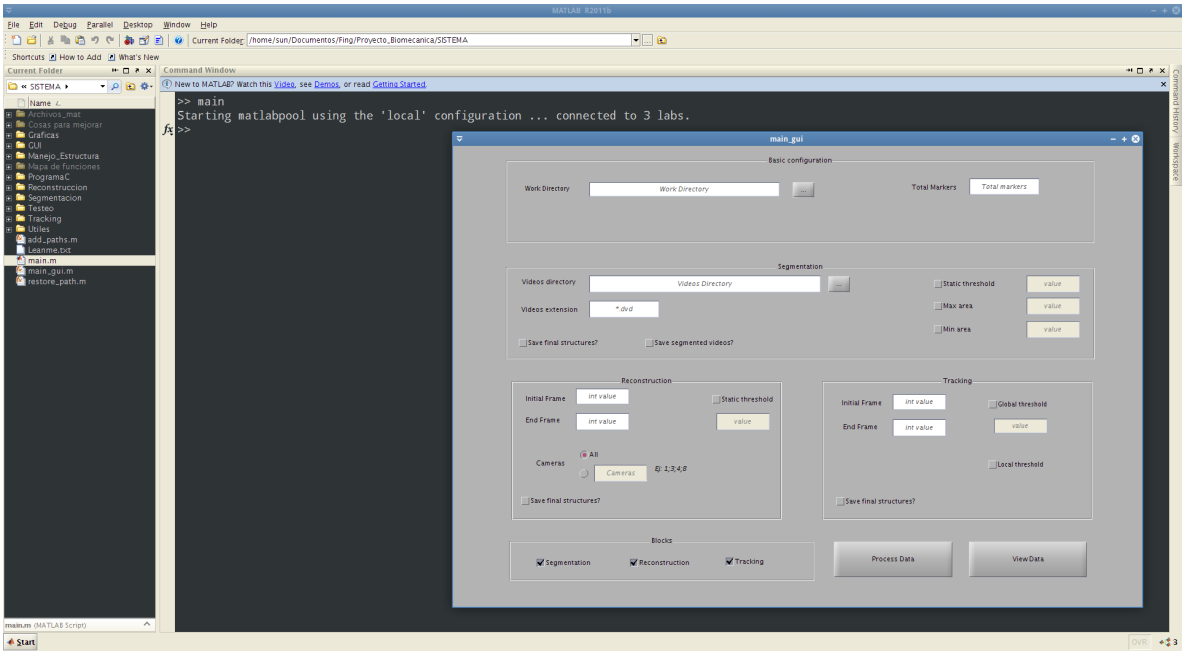


Figura D.1: Resultado de `main.m` en *Matlab*.

D.3.1. Basic configuration

Los parámetros a ingresar son:

- `Work Directory` es el directorio donde se almacena la información de los datos procesados. En el caso de trabajar con la base de datos por ejemplo para la secuencia sintética del sujeto 09_07 con videos de resolución 1600x600 y los cuadros por segundo del bvh original, la carpeta a seleccionar es: `Base_de_datos/Sujeto_CMU_09/09_07/Datos_Procesados/1600_600-100-100`.

[3]Por defecto Matlab utiliza 2 núcleos del procesador, si se desea configurar un número mayor de núcleos se debe acceder a `/Parallel/Manage_Configurations`

- `Total Markers` indica al sistema el número de marcadores utilizados. Dicho número se utiliza sobre todo en las etapas de reconstrucción y seguimiento. En la versión actual del sistema es recomendable, si la secuencia tiene n marcadores, ingresar un valor de al menos $n + 2$, para evitar perdida de marcadores en reconstrucción.

D.3.2. Segmentation

- `Videos Directory` es el directorio donde se deben encontrar los videos a procesar. En el caso de trabajar con la base de datos por ejemplo para la secuencia sintética del sujeto 09_07 con videos de resolución 1600x600 y los cuadros por segundo del bvh original, la carpeta a seleccionar es: `Base_de_datos/Sujeto_CMU_09/09_07/Datos_Imagen/1600_600-100-100`.
- `Videos extension`, casilla que permite ingresar la extensión de los videos a procesar, por defecto se utiliza la extensión .dvd que es un tipo de archivo mpeg.
- `Save final structures`. Se debe seleccionar esta casilla si se desea obtener archivos .mat y .xml de las estructuras cam y skeleton, dentro de la carpeta de trabajo ingresada (`Work Directory/<Resolución>/Segmentacion`).
- `Save segmented video`. Esta casilla permite guardar los videos que resultan de la salida de los bloques de umbralización y detección de blobs del último video segmentado. De esta manera si puede visualizar las distintas etapas de la segmentación y ajustar manualmente los parámetros de entrada en caso de ser necesario. Los videos de salida se almacenan en la carpeta de los videos a segmentar (`Videos Directory`).
- `Static threshold` es un parámetro opcional. El mismo establece un umbral fijo para la umbralización, debe ser un valor entre 0 y 255.
- `Max area` es un parámetro opcional que modifica el área máxima del filtrado por área. Debe ser un valor positivo.
- `Min area` es un parámetro opcional que modifica el área mínima del filtrado por área. Debe ser un valor positivo.

D.3.3. Reconstruction

- `Init Frame`, se ingresa el cuadro a partir del cual se efectúa la reconstrucción.
- `End Frame`, se ingresa el cuadro final hasta el cual se efectúa la reconstrucción.
- `Cameras`. Por defecto se reconstruye con todas las cámaras disponibles en la estructura cam sobre la cual se está trabajando, si se desea hacer reconstrucciones con cierto conjunto de cámaras se debe seleccionar la segunda casilla e ingresar el conjunto de cámaras.

- `Save final structures`. Se debe seleccionar esta casilla si se desea obtener archivos .mat y .xml de las estructuras cam y skeleton, dentro de la carpeta de trabajo ingresada (`Work Directory/<Resolución>/Reconstruccion`).

- `Static threshold` es un parámetro opcional. El mismo establece un umbral máximo dentro del cual buscar validaciones de una posible reconstrucción, el valor se debe ingresar en metros.

D.3.4. Reconstruction

- `Init Frame`, se ingresa el cuadro a partir del cual se efectúa el seguimiento.

- `End Frame`, se ingresa el cuadro final hasta el cual se efectúa seguimiento.

- `Save final structures`. Se debe seleccionar esta casilla si se desea obtener archivos .mat y .xml de las estructuras cam y skeleton, dentro de la carpeta de trabajo ingresada (`Work Directory/<Resolución>/Tracking`).

- `Global threshold` es un parámetro opcional para efectuar pruebas sobre el algoritmo de seguimiento.

- `Local threshold` es un parámetro opcional para efectuar pruebas sobre el algoritmo de seguimiento.

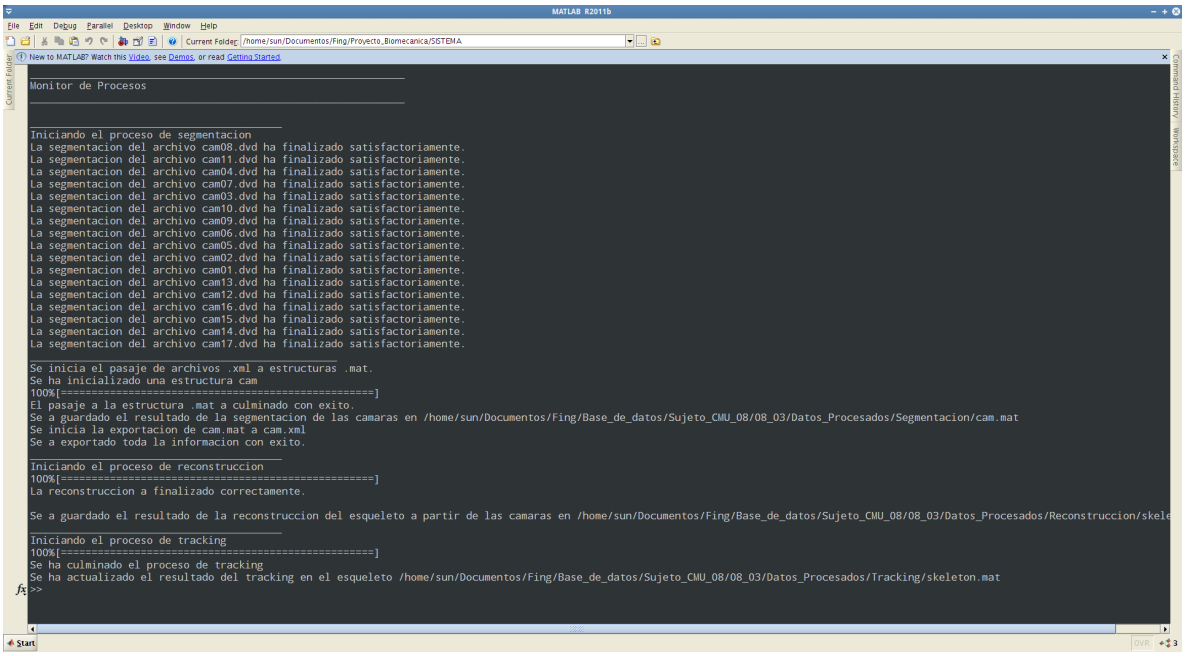


Figura D.2: Resultados del procesamiento en la consola de *Matlab*.

D.4. Bugs

En Ubuntu 14.04 de 32 bits, cuando se ejecuta la segmentación desde la interfaz gráfica la primera vez que se abre *Matlab*, en algunas oportunidades no se carga correctamente el ejecutable `Source_GLNX86`. Los pasos a seguir para solucionar este inconveniente son:

- Verificar que el ejecutable funcione fuera de Matlab. Abrir una terminal donde se encuentre el ejecutable y ejecutar:

  ```
  ./Source_GLNX86 <vidpath>
  ```

  donde `<vidpath>` es la dirección del video a segmentar.

- Si en el paso anterior no funciona la segmentación, compilar nuevamente el programa que efectúa la segmentación, revisar si dicha compilación finaliza correctamente y colocar el ejecutable en `Proyecto_Biomecanica/SISTEMA/ProgramaC` con el nombre `./Source_GLNX86`. En caso que si funcione la segmentación en la terminal, entonces el problema es de *Matlab*. Una solución es ejecutar tres veces la segmentación desde la interfaz gráfica, cuidando antes de cada ejecución de tener el workspace *Matlab* en `/Proyecto_Biomecanica/SISTEMA`. A la tercera vez el sistema funcionará, y de ahí en más mientras se tenga abierto *Matlab* se ejecuta todo normalmente.

Referencias

[1] Mopix. http://mopix.com.uy/. Accedido 20-12-2014.

[2] Vicon. http://www.vicon.com. Accedido 30-11-2014.

[3] Qualisys. http://www.qualisys.com/. Accedido 30-11-2014.

[4] Optitrack. http://www.naturalpoint.com/optitrack/. Accedido 30-11-2014.

[5] Massive. http://www.massivesoftware.com/. Accedido 27-12-2014.

[6] Motion analysis. http://www.motionanalysis.com/index.html. Accedido 30-11-2014.

[7] Phasespace systems. http://www.phasespace.com/index.html. Accedido 30-11-2014.

[8] Kinovea. http://www.kinovea.org/. Accedido 30-11-2014.

[9] Pascual J Figueroa, Neucimar J Leite, and Ricardo ML Barros. A flexible software for tracking of markers used in human motion analysis. *Computer methods and programs in biomedicine*, 72(2):155–165, 2003.

[10] Kazutaka Kurihara, Shin'ichiro Hoshino, Katsu Yamane, and Yoshihiko Nakamura. Optical motion capture system with pan-tilt camera tracking and realtime data processing. In *ICRA*, pages 1241–1248, 2002.

[11] Alvaro Pardo. Simple and robust hard cut detection using interframe differences. In *Progress in Pattern Recognition, Image Analysis and Applications*, pages 409–419. Springer, 2005.

[12] Mehmet Sezgin et al. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic imaging*, 13(1):146–168, 2004.

[13] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[14] Lorna Herda, Pascal Fua, Ralf Plänkers, Ronan Boulic, and Daniel Thalmann. Using skeleton-based tracking to increase the reliability of optical motion capture. *Human movement science*, 20(3):313–341, 2001.

[15] Maurice Ringer and Joan Lasenby. Modelling and tracking articulated motion from multiple camera views. In *BMVC*, volume 2000, pages 172–181, 2000.

[16] Adam G Kirk, James F O'Brien, and David A Forsyth. Skeletal parameter estimation from optical motion capture data. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 782–788. IEEE, 2005.

[17] C. J. Veenman, M.J.T. Reinders, and E. Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:54–72, 2001.

[18] Olivier Faugeras and Luc Robert. What can two images tell us about a third one? *International Journal of Computer Vision*, 18(1):5–19, 1996.

[19] Christian Andrés Diaz, Maria Luisa Toro, Johana Carolina Forero, and Andrés Torres. Detección, rastreo y reconstrucción tridimensional de marcadores pasivos para análisis de movimiento humano. cinemed iii-detection, tracking and 3d reconstruction of passive markers for human gait analysis. *Revista ingeniería Biomédica*, 3(6):56–68, 2009.

[20] M Shahid Shafiq, S Turgut Tümer, and H Cenk Güler. Marker detection and trajectory generation algorithms for a multicamera based gait analysis system. *Mechatronics*, 11(4):409–437, 2001.

[21] Fabio Martínez, Francisco Gómez, and Eduardo Romero. Análisis de vídeo para estimación del movimiento humano: una revisión. *Revista Med*, 17(1):953–106, 2009.

[22] Robert Fisher. Cvonline: The evolving, distributed, non-proprietary, on-line compendium of computer vision. http://homepages.inf.ed.ac.uk/rbf/CVonline. Accedido 30-11-2014.

[23] Gilles Mazars. Cvpapers - computer vision resource. http://www.cvpapers.com/index.html. Accedido 30-11-2014.

[24] Hayko Riemenschneider. Yet another computer vision index to datasets (yacvid). http://riemenschneider.hayko.at/vision/dataset/. Accedido 30-11-2014.

[25] The Ohio State University. Accad, motion capture lab. http://accad.osu.edu/research/mocap/mocap_home.htm. Accedido 30-11-2014.

[26] Carnegie Mellon University. Cmu graphics lab motion capture database. http://mocap.cs.cmu.edu/. Accedido 30-11-2014.

[27] Georgia Tech. Human identification at a distance. http://www.cc.gatech.edu/cpl/projects/hid/index.html. Accedido 30-11-2014.

[28] University of Southampton ISIS research group. Southampton human id at a distance database. http://www.gait.ecs.soton.ac.uk/database/. Accedido 30-11-2014.

[29] University of South Florida. Baseline algorithm and performance for gait based human id challenge problem. http://marathon.csee.usf.edu/GaitBaseline/. Accedido 30-11-2014.

[30] Max Planck Institute for Intelligent Systems Perceiving Systems. Humaneva dataset. http://humaneva.is.tue.mpg.de/. Accedido 30-11-2014.

[31] National Institute for Research in Computer Science INRIA and Control. 4d repository. http://4drepository.inrialpes.fr/pages/home. Accedido 30-11-2014.

[32] Ralph Gross and Jianbo Shi. The cmu motion of body (mobo) database. Technical Report CMU-RI-TR-01-18, Robotics Institute, Pittsburgh, PA, June 2001. Accedido 30-11-2014.

[33] Center for Biometrics and Security Research. Casia gait database. http://www.cbsr.ia.ac.cn/english/Gait%20Databases.asp. Accedido 30-11-2014.

[34] Lab at University of California at Berkeley and Johns Hopkins University Center for Imaging Science. The berkeley multimodal human action database (mhad). http://tele-immersion.citris-uc.org/berkeley_mhad#dl. Accedido 30-11-2014.

[35] Maddock Meredith and S Maddock. Motion capture file formats explained. *Department of Computer Science, University of Sheffield*, 211, 2001.

[36] cgspeed. http://www.cgspeed.com/. Accedido 6-12-2014.

[37] David Wooldridge. bvhacker: The free bvh file editing tool. http://davedub.co.uk/bvhacker/. Accedido 30-11-2014.

[38] Rafael C Gonzalez and Richard E Woods. *Digital image processing*. Prentice hall Upper Saddle River, NJ:, 2002.

[39] Rafael C Gonzalez and Richard E Woods. *Digital image processing*, chapter 3.5, page 174. Prentice hall Upper Saddle River, NJ:, 2002.

[40] Opencv. http://opencv.org/. Accedido 29-11-2014.

[41] Cvblob-blob library for opencv. https://code.google.com/p/cvblob/. Accedido 29-11-2014.

[42] Rafael C Gonzalez and Richard E Woods. *Digital image processing*, chapter 10.3, page 760. Prentice hall Upper Saddle River, NJ:, 2002.

[43] Rafael C Gonzalez and Richard E Woods. *Digital image processing*, chapter 9.6.3, page 692. Prentice hall Upper Saddle River, NJ:, 2002.

[44] Rafael C Gonzalez and Richard E Woods. *Digital image processing*, chapter 12.2.2, page 894. Prentice hall Upper Saddle River, NJ:, 2002.

[45] B Morse. Thresholding. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/thresho Accedido 29-11-2014.

[46] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179–187, 1962.

[47] M Tailanian and J Cardelino. Kernel density estimator. https://github.com/martin-etchart/kde, 2013. Accedido 24-11-2014.

[48] Shamik Sural, Gang Qian, and Sakti Pramanik. Segmentation and histogram generation using the hsv color space for image retrieval. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 2, pages II–589. IEEE, 2002.

[49] Wikipedia. Binary large object. http://en.wikipedia.org/wiki/Binary_large_objec. Accedido 29-11-2014.

[50] Ayoub B Ayoub. The eccentricity of a conic section. *College Mathematics Journal*, 34(2):116–121, 2003.

[51] Richard O Duda and Peter E Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[52] Wikipedia. Extensible markup language. http://es.wikipedia.org/wiki/Extensible_Markup_Language. Accedido 29-11-2014.

[53] M. Black and L. Sigal. HumaEva dataset. http://humaneva.is.tue.mpg.de/. Accedido 29-11-2014.

[54] Olivier Faugeras. *Three-dimensional computer vision: a geometric viewpoint*. MIT press, 1993.

[55] Gerard Medioni and Sing Bing Kang. *Emerging topics in computer vision*. Prentice Hall PTR, 2004.

[56] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[57] Michael Warren, David McKinnon, and Ben Upcroft. Online Calibration of Stereo Rigs for Long-Term Autonomy. In *International Conference on Robotics and Automation (ICRA)*, Karlsruhe, 2013.

[58] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A convenient multicamera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[59] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 666–673. IEEE, 1999.

[60] Boguslaw Cyganek and J Paul Siebert. *An introduction to 3D computer vision techniques and algorithms*. John Wiley & Sons, 2011.

[61] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45, 1960.

[62] N Malik, T. Dracos, and D. Papantoniou. Particle tracking in three-dimensional turbulent flows - part ii: Particle tracking. *Experiments in Fluids*, 15:297–294, 1993.

[63] Leonid Sigal, Alexandru O Balan, and Michael J Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International journal of computer vision*, 87(1-2):4–27, 2010.

[64] Ralph Gross and Jianbo Shi. The CMU motion of body (mobo) database. 2001.

[65] Dwayne Phillips. *Image processing in C: analyzing and enhancing digital images*. R & D Publications, Inc., 1994.

[66] Bobby Bodenheimer, Chuck Rose, Seth Rosenthal, and John Pella. *The process of motion capture: Dealing with the data*. Springer, 1997.

[67] Richard D Seely, Sina Samangooei, M Lee, John N Carter, and Mark S Nixon. The university of southampton multi-biometric tunnel and introducing a novel 3d gait dataset. In *Biometrics: Theory, Applications and Systems, 2008. BTAS 2008. 2nd IEEE International Conference on*, pages 1–6. IEEE, 2008.

[68] Prem Kuchi, Raghu Ram V Hiremagalur, Helen Huang, Michael Carhart, Jiping He, and Sethuraman Panchanathan. Drag: a database for recognition and analasys of gait. In *ITCom 2003*, pages 115–124. International Society for Optics and Photonics, 2003.

[69] Sincronización. http://www.canon.es/. Accedido 29-11-2014.

[70] Todo fotografía. http://todo-fotografia.com/. Accedido 2-12-2014.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Índice de figuras

Esta es la última página.
Compilado el viernes 3 julio, 2015.
http://iie.fing.edu.uy/