Primer informe

Desarrollo e implementación de algoritmo de captura de puntos sobre un espacio 3D

MANUEL JHOBANNY MORILLO ORDOÑEZ

Bucaramanga, Santander
2016

Control de las cámaras Optitrack usando el SDK ofrecido por el fabricante (Natural Point)

La librería `ccameralib.h` se diseñó con el propósito de brindar características obtenidas directamente del SDK ofrecido por el fabricante de las cámaras, entre las que se destacan la activación y desactivación de las cámaras, visualización de los fotogramas de ambas cámaras, obtención de marcadores en la escena, control del tipo de video (modo segmentación y modo escala de grises) y detección de comandos introducidos por teclados.

Para hacer uso de esta librería, además de incluirla en las cabeceras del proyecto se deben llamar dos métodos que hacen parte de ella, los método `activateCameras()` y `ShowCameras()`, el primero se encarga de activar las cámaras; devuelve falso si las cámaras no pudieron ser activas, y el segundo inicia la visualización de los fotogramas de ambas cámaras en ventanas por separado.

Luego de iniciadas las cámaras están disponibles una serie de controles que se activan por teclado y permiten realizar diversas tareas programadas. Estas tareas se especifican dentro del método `GetKey(…)`, las tareas son: cambiar el modo de vídeo, tomar capturas de pantalla, activar el modo servidor, enviar datos a través de la red y activar el modo que permite la captura de puntos.

A continuación se describen las tareas más importantes.

Cambiar el modo de vídeo

- Modo segmentación: permite el rastreo de marcadores, consiguiendo así obtener el área y las coordenadas en pixeles de cada marcador que se detecta dentro de la escena.
- Modo escala de grises: Se utiliza para sacar instantáneas de alguna escena en concreto. Por ejemplo, para empezar un nuevo proceso de calibración de las cámaras. También es utilizado para rastrear objetos, obtener áreas y coordenadas en pixeles de los marcadores.

Activar el modo servidor

Crea un servidor en el `localhost` y queda en espera a que un cliente se conecte a él.

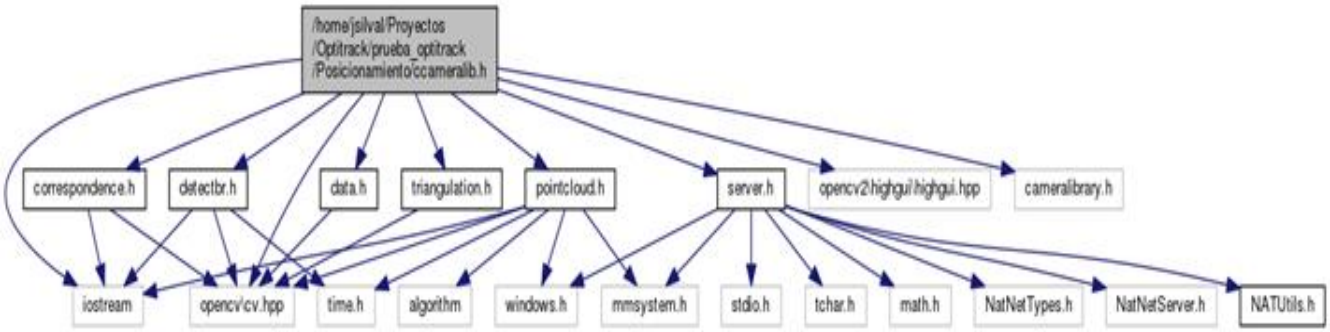
Enviar datos a través de la red

Activa el envío de datos a través de la red, los datos serán enviados sí y solo sí se están detectando objetos rígidos en la escena.

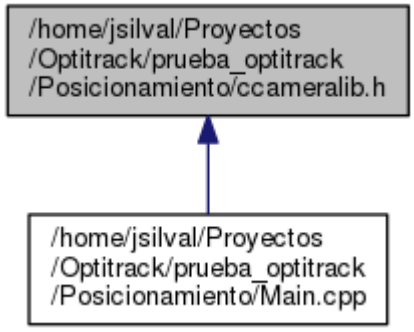
Activar el modo que permite la captura de puntos

Permite al sistema obtener puntos o coordenadas x, y, z, utilizando un puntero, captura el punto donde se ubique la punta del puntero. Más adelante se explica en detalle esta característica.

Para una vista más general, a continuación se muestra la dependencia de `ccameralib.h` y los archivos que directamente o indirectamente incluye este archivo de cabecera.



1. Ilustración: Archivos que directamente o indirectamente incluyen `ccameralib.h`

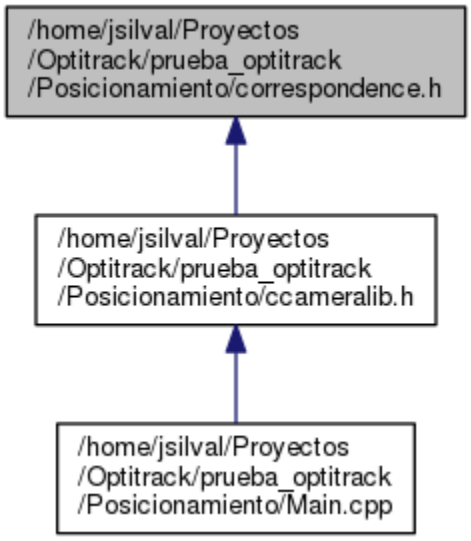


2. Ilustración: Archivos que directamente o indirectamente incluyen `ccameralib.h`

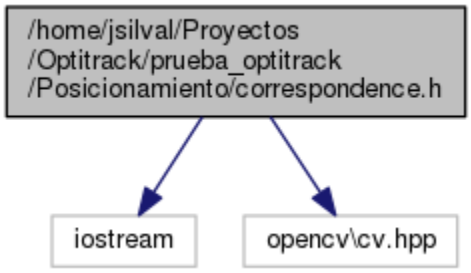
Solución al problema de la correspondencia de marcadores detectados en modo rastreo de objetos

El problema de la correspondencia consiste en detectar adecuadamente cada marcador sobre la escena, es decir, realizar la correspondencia 2D entre cada marcador. El algoritmo que realiza esta tarea se encuentra dentro de la librería `correspondence.h`. Para una primera versión del algoritmo se pensó en lograr detectar de forma adecuada la correspondencia si se cumplen las siguientes condiciones: no existe superposición de cuerpos rígidos en la escena, cada cuerpo rígido debe tener una esfera adicional y de menor tamaño que las esferas que lo conforman. Básicamente el algoritmo identifica la esfera más pequeña que hace parte del objeto rígido y localiza la esfera más cercana a ella, desde ese punto recorre en sentido horario las demás esferas hasta completar el ciclo, hace esto para cada objeto rígido que se encuentre en la escena.

Finalizado la detección de la correspondencia, la función `CorrespondenceDetection(…)` (la función principal de esta librería), devuelve un conjunto de esferas ordenados de forma correcta y listo para ser pasados junto con el otro set de esferas detectados de la segunda cámara como argumento a la función de triangulación.



3. Ilustración: Archivos que directamente o indirectamente incluyen `correspondence.h`.

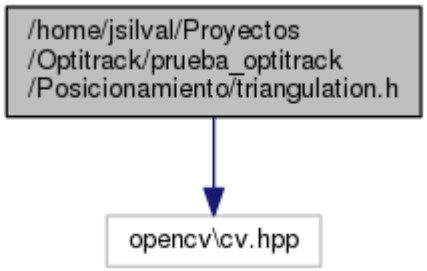


4. Ilustración: Dependencia de `correspondence.h`.

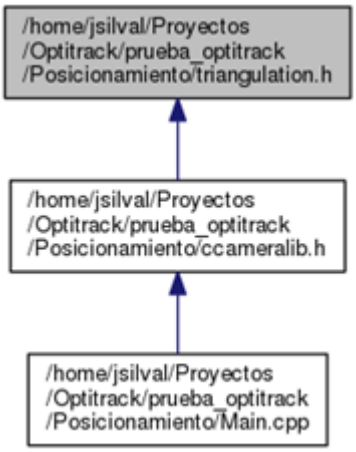
Triangulación de coordenadas

La triangulación es un método que se utiliza para calcular las coordenadas espaciales de una escena 3D utilizando la correspondencia 2D entre varias imágenes.
Para lograr este objetivo se tradujo la función de calibración que viene en el código fuente de "Camera Calibration Toolbox for Matlab", descargado de la página https://www.vision.caltech.edu/bouguetj/calib_doc/.

La función lleva por nombre `stereo_triangulation` y hace parte de la librería triangulation.h, además de recibir los conjuntos de puntos en 2D de la escena, recibe los parámetros intrínsecos y extrínsecos, resultado de la calibración de las cámaras, también recibe por referencia dos matrices vacías 3xN que serán llenadas con las coordenadas en 3D visto desde la cámara izquierda y visto desde la cámara derecha.



5. Ilustración: Dependencia de `triangulation.h`



6. Ilustración: Archivos que directamente o indirectamente incluyen `triangulation.h`.

A continuación se muestra el resultado que se obtuvo al comparar las distancias medidas sobre la barra de calibración brindada por el fabricante de las cámaras, utilizando los resultados de esta función de triangulación.

	[mm]
Barra de calibración	
Distancia Mayor	250
Distancia Media	162.5
Distancia Menor	87.5

Tabla 1: Distancias teóricas de la barra calibrada.

Resultado de las pruebas

	Distancia Mayor [mm]	**Distancia Media [mm]**	**Distancia Menor [mm]**
Prueba #1	251.336	163.304	88.0342
Prueba #2	251.593	163.535	88.0386
Prueba #3	251.784	163.702	88.1115
Prueba #4	250.932	163.049	87.8941
Prueba #5	250.374	162.432	87.9213
Prueba #6	251.235	163.273	87.9618
Prueba #7	250.714	162.946	87.7674
Prueba #8	251.369	163.449	87.9211
Prueba #9	250.096	162.504	87.5923
Prueba #10	251.641	163.46	88.1807

Tabla 2: Distancias experimentales obtenidas después de utilizar los resultados de la función de triangulación.

MEDIA [mm]	251.1074	163.1654	87.9423
DIFERENCIA [mm]	1.1074	0.6654	0.4423
DESVIACIÓN [mm]	0.563878671	0.4294130878	0.1699236561

Tabla 3: Media, diferencia y desviación de los resultados.

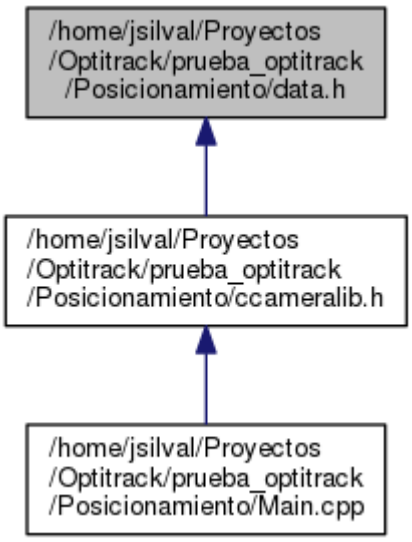
El resultado muestra que la precisión de la función de triangulación es buena, obteniendo un error máximo de 1.1 mm y una desviación de aproximadamente 0.5 mm en las medidas.

1. Foto: Barra calibrada utilizada para calcular las distancias entre las tres esferas (distancia mayor, menor y media)

Datos constantes usados por la función de triangulación y la función que detecta los cuerpos rígidos.

Adicionalmente, la función de triangulación hace uso de una librería que le proporciona las constantes de calibración o parámetros intrínsecos y extrínsecos de las cámaras, estas constantes se encuentran definidas en el archivo de cabecera `data.h`, además de estas constantes, también se definen las distancias conocidas de cada objeto rígido, con el fin de facilitar la detección de éstos.

A continuación se muestra la dependencia de `data.h` y los archivos que directamente o indirectamente incluye este archivo de cabecera.

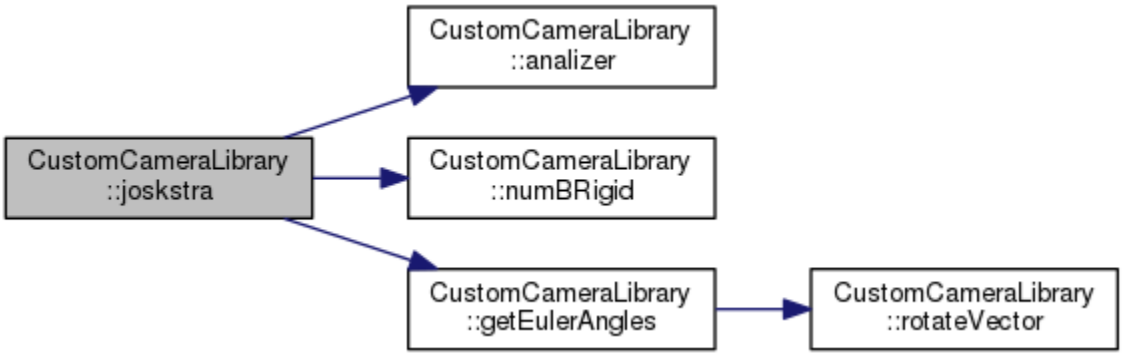


7. Ilustración: Archivos que directamente o indirectamente incluyen `data.h`.

Detección de cuerpos rígidos

La detección de los cuerpos rígidos se basa en la geometría de ellos, cada cuerpo rígido está formado por 4 esferas que lo definen y una quinta esfera que se utiliza para detectar la correspondencia. La distancia entre cada una de estas 4 esferas es bien conocida y no se repiten en ninguno de los demás cuerpos rígidos. Teniendo en cuenta estas características se diseñó una función para detectar cada uno de los cuerpos rígidos que entran en la escena, dicha función se encuentra dentro del archivo de cabecera `detectbr.h` y lleva de nombre `joskstra` y ha sido inspirada en el algoritmo de Dijkstra.

`Joskstra`, hace uso de varias funciones auxiliares que le permiten analizar cada distancia conocida y comparar con las distancias que se van calculando, identificar qué objeto rígido ha detectado y calcular los ángulos de Euler.



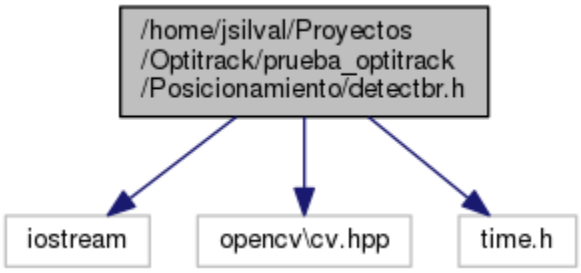
9. Ilustración: Llamadas para esta función.

10. Ilustración: Llamadas a esta función.

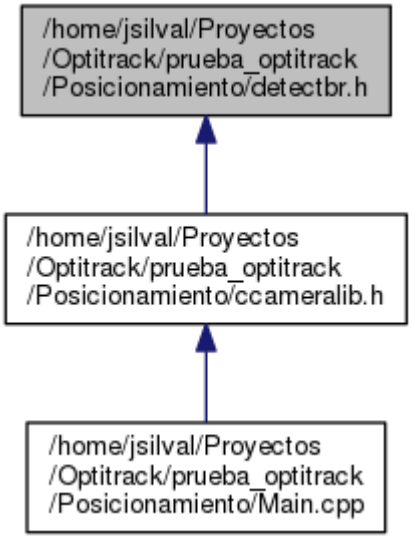
La función `josktsra` se utiliza dentro del archivo de cabecera `ccameralib.h`, dentro del método `ShowCameras(…)`, recibiendo así el set de coordenadas 3D visto por alguna de las cámaras y devolviendo un arreglo de estructuras, cada estructura tiene las características que identifican a un cuerpo rígido.

Características que identifican a un cuerpo rígido

- Identificador del cuerpo rígido.
- Nombre del cuerpo rígido.
- Centro del círculo que forma el cuerpo rígido.
- Cantidad de marcadores que detectó del cuerpo rígido.
- Ángulos de Euler.
- Punto proyectado (si se trata del pointer).
- Orientación (Cuaternios).



11. Ilustración: Dependencia de `detectbr.h`



12. Ilustración: Archivos que directamente o indirectamente incluyen `detectbr.h`.

Capturar nube de puntos

Para realizar la captura de puntos en el espacio se diseñó la clase Cloud, definida dentro del archivo de cabecera `pointcloud.h` y la definición de sus métodos se encuentran en el archivo `pointcloud.cpp`.
La captura de puntos se puede realizar de dos maneras o modos, en modo automático o en modo manual.

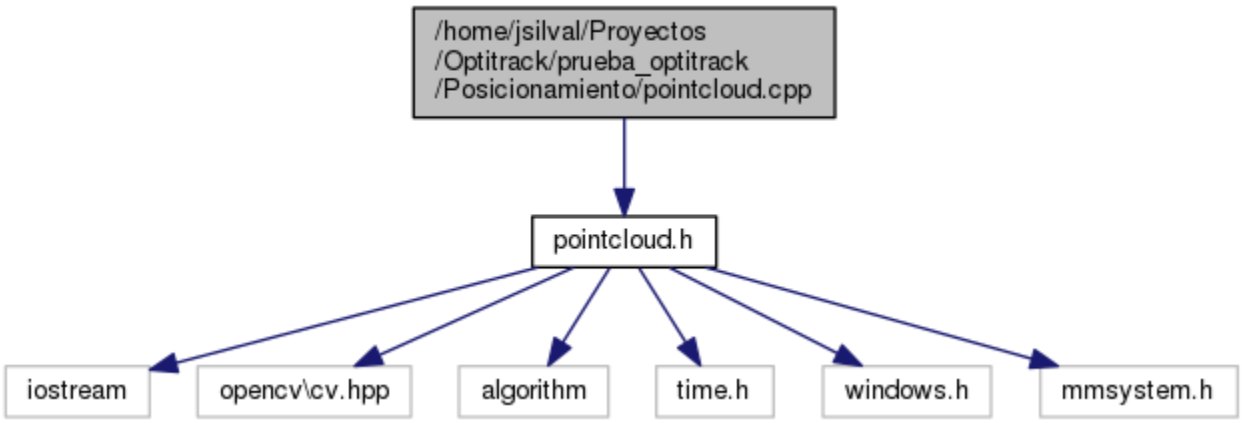
Modo automático

La toma de puntos se realiza cada cierto intervalo de tiempo, se puede dar pausa en cualquier momento y volver a reanudar. Ideal para tomar grandes volúmenes de puntos.

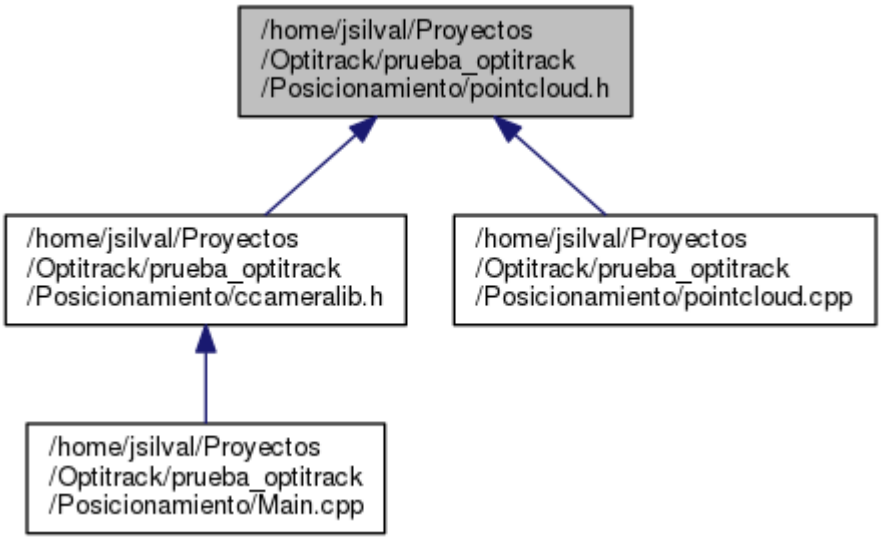
Modo manual

Los puntos se capturan cada vez que se presiona la tecla 'x', se puede dar pausa en cualquier momento y luego volver a reanudar. Ideal para capturar de manera controlada el número de puntos en una determinada zona.

Además, esta clase cuenta con funciones que permiten realizar modificaciones en la nube de puntos tomada; eliminar un punto en concreto, añadir un punto, añadir un conjunto de puntos, eliminar el último punto, encontrar un punto dentro de la nube y mostrar la nube de puntos recolectada.

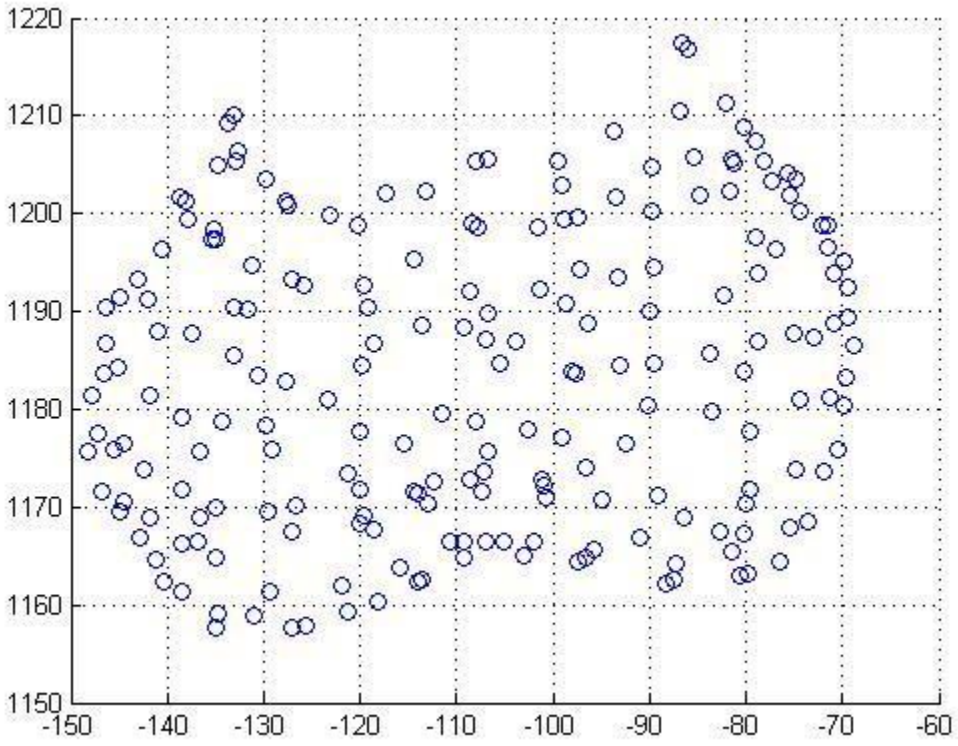


13. Ilustración: Dependencia de `pointcloud.cpp`

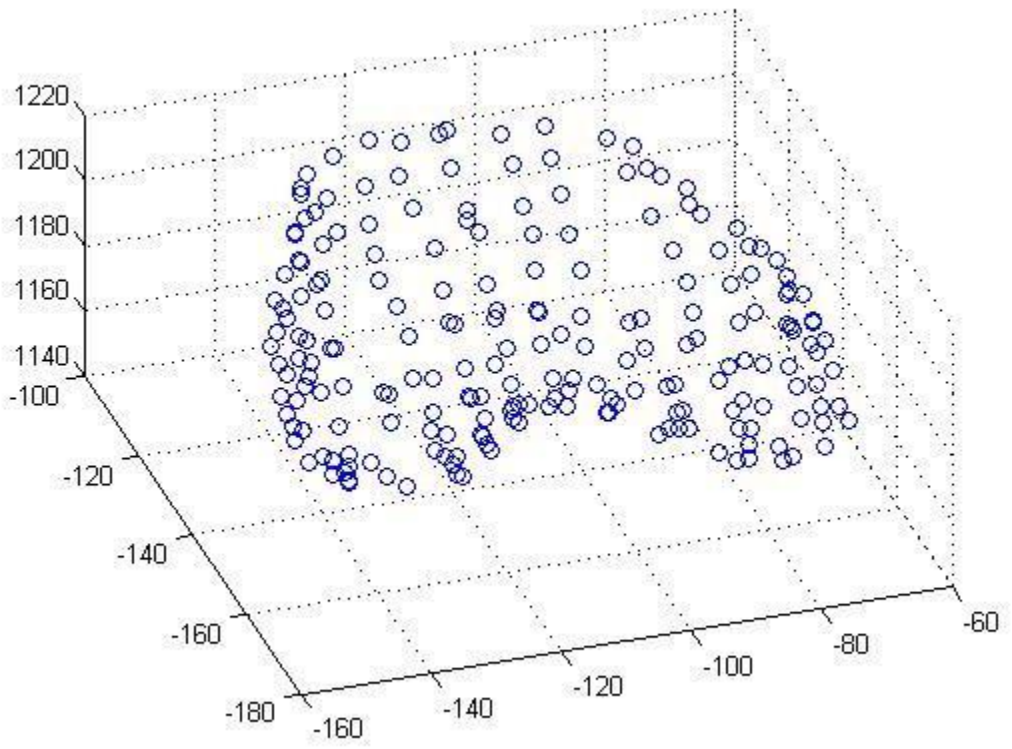


14. Ilustración: Archivos que directamente o indirectamente incluyen `pointcloud.cpp`

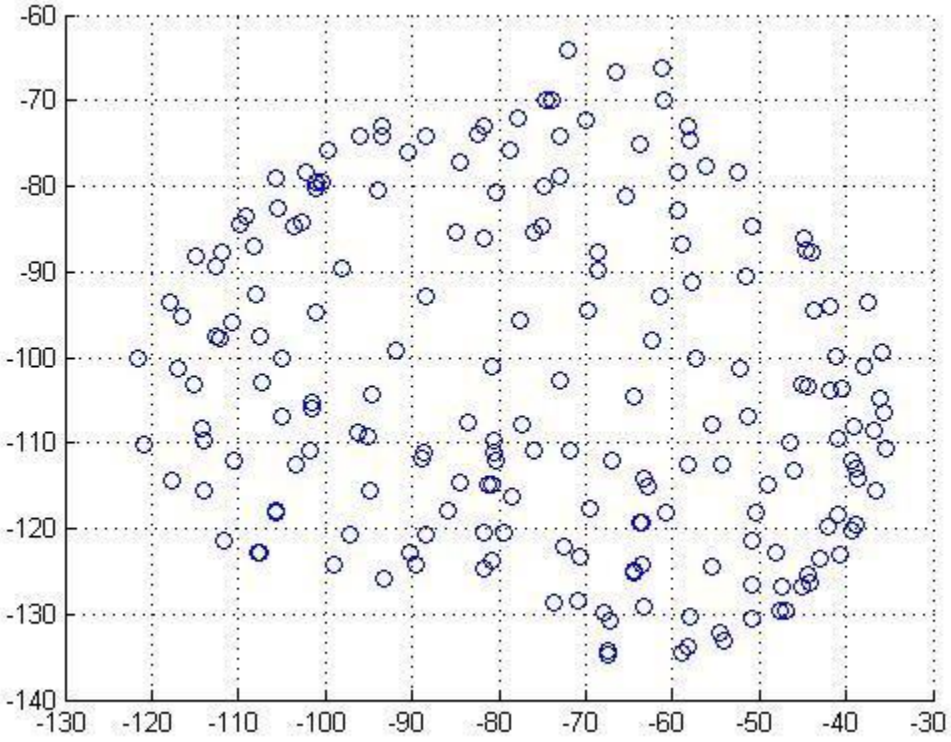
A continuación se muestra el resultado de capturar una serie de puntos sobre un modelo de fémur y un modelo de tibia, la nube de puntos se graficó utilizando el software Matlab.



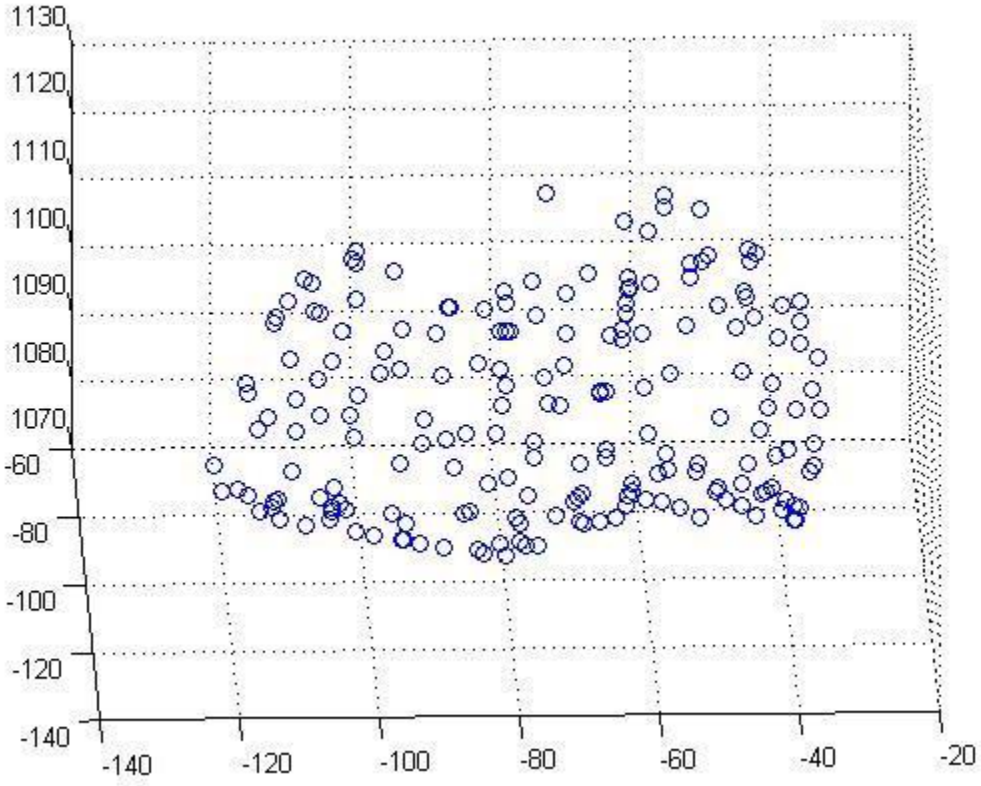
15. Ilustración: Nube de puntos tomadas sobre la cabeza de un fémur.



16. Ilustración: Nube de puntos tomadas sobre la cabeza de un fémur.



17. Ilustración: Nube de puntos tomadas sobre la cabeza de una tibia.



18. Ilustración: Nube de puntos tomadas sobre la cabeza de una tibia.

Para probar la precisión en que se estaban capturando la nube de puntos se diseñó una prueba que consistió en utilizar una esfera de material reflectivo de radio conocido y coordenada del centro conocido, tocar con la punta del pointer su superficie y obtener ese punto en coordenadas 3D para luego intentar calcular el radio. Cabe destacar que el centro de la esfera se obtuvo mediante triangulación, utilizando la función destinada para ese propósito. Teniendo en cuenta la propagación de error que esto implica, la toma de datos se realizó en 3 posiciones ubicadas a 2m, 1m y 0.6m de las cámaras, por cada posición se movió la esfera en 9 direcciones distintas (arriba, abajo, derecha o izquierda) y por cada una de esas posiciones se sacaron 30 puntos sobre la superficie de la esfera para luego analizarlos.

Algunos de los resultados de dicha prueba se muestran a continuación; r es el radio de la esfera r_teórico es el radio teórico de la esfera.

Los resultados corresponden a una de las distancias más cercanas a la cámara donde los errores en el cálculo del radio no superaron los 2.3 mm y hubo una desviación de 1.056637312 mm.

Si desea ver el resultado completo de la prueba de precisión del pointer puede hacerlo ingresando acá.

Y si bien los resultados no son muy precisos para el propósito de obtener errores menores a un milímetro, hay que tener en cuenta que el pointer presenta defectos de fabricación que hacen muy difícil llegar a tener una buena precisión y además, ocurre una propagación de error debido al uso de la función de triangulación, el algoritmo se seguirá mejorando hasta obtener mejores resultados.

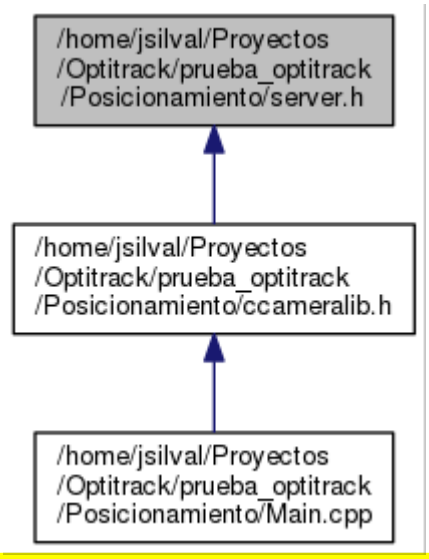
2. Foto: Modelo de tibia utilizado para la toma de nube de puntos.

3. Foto: Modelo de tibia utilizado para la captura de la nube de puntos.

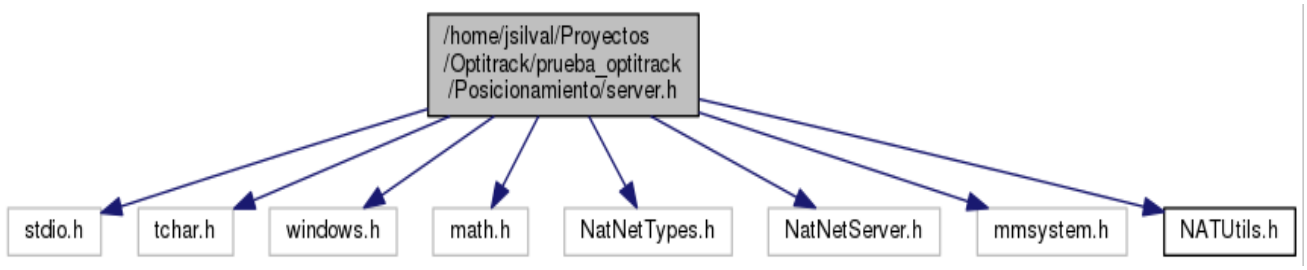
Envío de datos a través de la red

Se desarrollaron funciones que permiten crear un servidor local que pone a disponibilidad de un cliente datos que envía a través de la red, utilizando las librerías brindadas por el fabricante de las cámaras, este servidor envía los datos usando el protocolo UDP y se crea en un nuevo hilo de ejecución.

Los datos son enviados hasta un cliente, que a su vez, envía los datos recibidos a una escena de Unity. Los datos enviados corresponden a los obtenidos de los objetos rígidos que se detecten en el campo de visión de las cámaras; se agrupan por nombre del objeto rígido, marcadores que hacen parte del objeto rígido, orientación y posición del mismo.



19. Ilustración: Archivos que directamente o indirectamente incluyen `server.h`

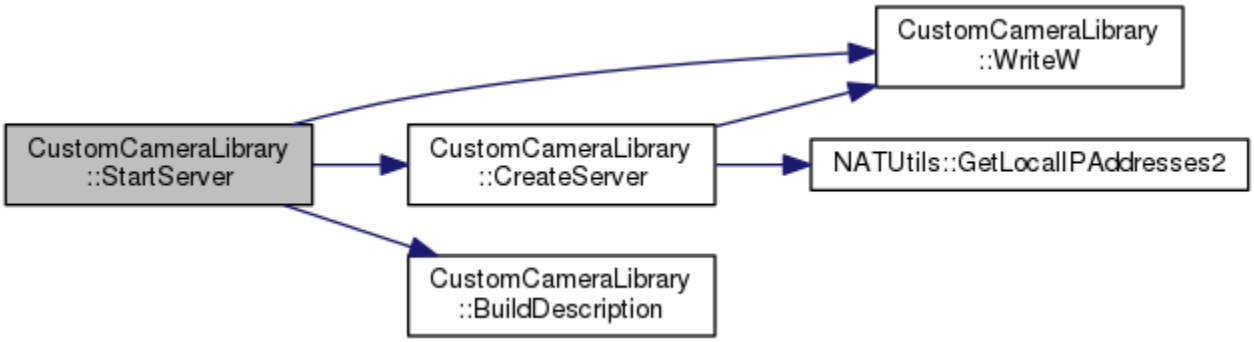


20. Ilustración: Dependencia de `server.h`

El servidor cuenta con las siguientes opciones principales programadas:

Crear servidor

Crea el servidor utilizando una conexión de tipo unicast.
Función: `StartServer()`, devuelve un mensaje de error si ocurre un fallo al momento de intentar crear el servidor.



21. Ilustración: Llamadas que realiza la función `StartServer()`

22. Ilustración: Llamadas a la función `StartServer()`

Reiniciar servidor

Reinicia el servidor si está activo y nuevamente lo inicia.
Función: `resetServer()`.

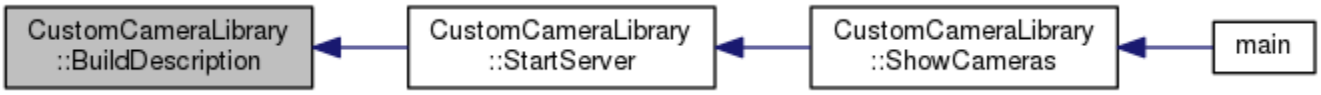
Detener envío de datos

Detiene el envío de datos a través de la red. La función verifica si el servidor está enviando datos, de ser así, cierra la conexión.
Función: `StopPlayingThread()`.

23. Ilustración: Llamadas a la función `StopPlayingThread()`

Crear descripción de objetos rígidos

Al momento de crear el servidor se realiza esta acción, se trata de una función que recoge la información necesaria para identificar los objetos rígidos que están en escena, los empaqueta y los mantiene a espera de que el cliente los solicite.



24. Ilustración: Llamadas a la función `BuildDescription(...)`

También cuenta con muchas otras opciones para lograr su correcto funcionamiento, pero las descritas anteriormente son las básicas para poner el servidor en marcha.

Marcadores y pointer utilizados

3. Foto: Pointer y objetos rígidos utilizados durante el desarrollo e implementación de los algoritmos.