Reconstrucción 3D Mediante un Sistema Stéreo de Cámaras Kinect

Anthony Alexis Caicedo Amorocho

Trabajo de Grado para Optar el título de Ingeniero de Sistemas

Director

Gabriel Rodrigo Pedraza Ferreira

PhD. En Informática

Codirector

Luis Eduardo Bautista Rojas

M Sc. En Ingeniería de Sistemas e Informática

Universidad Industrial de Santander

Facultad de Ingeniarías Físico-Mecánicas

Escuela de Ingeniería de Sistemas e Informática

2019

Dedicatoria

A mis padres, por su paciencia y apoyo a lo largo de todos estos años.

A la Psicóloga Olga Lucia Ordoñez, Por su guía, apoyo y por cuyo infinito amor a su profesión inspira y alegra la vida de quienes la rodean.

Anthony Alexis Caicedo Amorocho

Agradecimientos

A mis padres por su apoyo, paciencia y comprensión sobre todas las decisiones que he tomado.

A Omar Duarte, Alvaro Martinez, Carlos Rocha y Nitae Uribe, por darme la oportunidad de compartir su sueño de crear, innovar y transmitir nuevas experiencias a las personas a través de los videojuegos, permitiéndome enamorarme de la programación.

A mis profesores, cuyas clases más que enseñarme, me dejaron experiencias de vida.

A mi familia, amigos y compañeros, por su comprensión y apoyo incondicional en todas mis decisiones.

Tabla de contenido

Lista de figuras

Lista de tablas

RESUMEN

TÍTULO: RECONSTRUCCIÓN 3D MEDIANTE UN SISTEMA ESTÉREO DE CAMARAS KINECT[1]

AUTORES: ANTHONY ALEXIS CAICEDO AMOROCHO[2]

PALABRAS CLAVE: SIMULACIÓN, GPU, RENDERIZADO, REALIDAD VIRTUAL.

DESCRIPCIÓN:

Usualmente un sensor de profundidad no puede aportar información suficiente para hacer una reconstrucción de una escena sin moverse a través de la escena que está visualizando, debido a esto se ha optado por recurrir a la reconstrucción de una escena en 3D usando la cantidad de sensores Kinect suficientes para obtener una escena 3D semejante a la escena real que está siendo visualizada por los sensores estando en una posición fija cada uno.

Actualmente no es posible conectar más de un sensor Kinect a un solo computador es por eso que se usó el proyecto de código libre Live Scan 3D para que cada Kinect que esté conectado a un computador pueda enviar su información a un servidor que renderizara la información de todos los Kinects que estén conectados a esa red en una única escena y creando así una recreación en 3D de la escena real visualizada por los sensores.

Gracias a la tecnología de realidad virtual existente hoy en día es posible navegar a través de escenarios virtuales recreados por diversos motores gráficos. Para hacer más inmersiva está recreación de una escena real se desarrolló un módulo de realidad virtual usando el motor de juegos Unity el cual nos permite navegar a través de la escena reconstruida en una perspectiva de primera persona.

---

[1] Trabajo de Grado

[2] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Gabriel Rodrigo Pedraza Ferreira PhD. Codirector: Luis Eduardo Bautista Rojas M Sc.

ABSTRACT

TITLE: 3D RECONSTRUCTION USING A KINECT CAMERA STEREO SYSTEM [3]

AUTHOR: ANTHONY ALEXIS CAICEDO AMOROCHO[4]

KEYWORDS: SIMULATION, GPU, RENDERED, VIRTUAL REALITY.

DESCRIPTION:

Usually a depth sensor cannot provide enough information to do a reconstruction of a scene without moving through the scene you are viewing, because of this you have chosen to resort to the reconstruction of a 3D scene using the amount of Kinect sensors sufficient to obtain a 3D scene similar to the real scene that is being visualized by the sensors being in a fixed position each.

Currently it is not possible to connect more than one Kinect sensor to a single computer that is why the Live Scan 3D free code project was used so that each Kinect that is connected to a computer can send its information to a server that will render the information all the Kinects that are connected to that network in a single scene and thus creating a 3D recreation of the real scene visualized by the sensors.

Thanks to the virtual reality technology that exists today, it is possible to navigate through virtual scenarios recreated by various graphic engines. To make this recreation of a real scene more immersive, a virtual reality module was developed using the Unity game engine which allows us to navigate through the reconstructed scene in a first-person perspective.

---

[3] Bachelor Thesis

[4] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: Gabriel Rodrigo Pedraza Ferreira PhD. Codirector: Luis Eduardo Bautista Rojas M Sc.

Introducción

La simulación virtual de ambientes reales durante mucho tiempo se limitó únicamente al modelado a escala de los entornos que se quisieran simular en un mundo virtual en el cual se pudiera en interactuar de cierta manera con los elementos de dicho entorno que se está simulando, cuando la industria de los videojuegos dio un paso más adelante creando los visores de realidad virtual para dar una inmersión más completa al jugador en el entorno del video juego, se demostró que al recrear un ambiente real de manera virtual y poder visualizar dicha reconstrucción virtual desde la perspectiva de primera persona que ofrece un visor de realidad virtual. la sensación de realismo que percibe el cerebro humano aumenta considerablemente pero no es completa ya que el usuario no puede interactuar de manera real y natural con los objetos virtuales, debido a que los modelos 3D que recrean una escena real no se mueven de manera natural y gracias a estas inconsistencias de lo virtual con lo real es que se producen los efectos de mareo y sensaciones de vértigo una vez que el usuario ha estado usando algún visor de realidad virtual durante mucho tiempo.

El área de la simulación virtual de ambientes reales ha venido avanzando a pasos agigantados en los últimos años debido a los grandes avances en hardware que se han dado en la industria de la computación, GPUs (Graphic Procesing Unit) cada vez más potentes y veloces hacen que simular un ambiente real en un espacio virtual sea cada vez más sencillo y fiel a la realidad, debido a la gran capacidad de procesamiento con la que cuentan las GPUs de última tecnología.

En conjunto con sensores de profundidad que son capaces de capturar imágenes espaciales, las GPUs son capaces de renderizar de manera virtual el ambiente percibido por los sensores de

profundidad creando una representación virtual de lo que perciben los sensores de profundidad. El renderizado de ambientes de realidad virtual es uno de los procesos más exigentes a nivel computacional ya que exige que la GPU renderice dos veces la misma escena (una para cada ojo) y que hace el usuario tenga la sensación de inmersión en la escena renderizada. Debido a esto el frame rate de actualización de una escena en realidad virtual no suele ser demasiado alto, incluso consolas de videojuegos de última generación que vienen diseñadas para juegos en realidad virtual ven reducida la cantidad de frames que la consola renderiza por segundo cuando se está renderizando un juego en realidad virtual. Aun así, el ojo humano solo necesita unos 30 frames por segundo para percibir un movimiento fluido de las cosas por lo que, en realidad virtual, mantener una tasa de actualización de 30 frames por segundo se considera aceptable.

Con los nuevos protocolos de comunicación y los nuevos algoritmos de renderizado 3D se puede alcanzar una velocidad de actualización de imágenes en realidad virtual realmente alta. En los videojuegos, debido a que se usan modelos 3D a escala cuyas formas están definidas en un archivo que se puede leer a alta velocidad para renderizar dicha figura, la velocidad de renderizado suele ser bastante alta. Al momento de renderizar imágenes 3D obtenidas mediante sensores de profundidad juega un papel importante la velocidad a la que se transmite la información del sensor hasta el computador en cargado de realizar el renderizado de la escena. adicionalmente influye la velocidad del algoritmo de renderizado que usa la computadora para crear la representación virtual de la imagen obtenida por el sensor de profundidad, es aquí donde entra el balance de eficiencia que tienen que existir entre la velocidad de transmisión de las imágenes del sensor y la velocidad de renderizado del algoritmo que se usa para recrear la escena virtual.

El desarrollo del presente proyecto se dividió en cuatro fases que nos permitió evaluar la factibilidad técnica de realizar una reconstrucción 3D usando el proyecto Live Scan 3D para visualizar dicha reconstrucción mediante un dispositivo de realidad virtual. Para la fase 1 se implementó por completo el proyecto Live Scan 3D y gracias a que es un proyecto de código abierto y tenemos acceso a su código fuente se realizaron una serie de mejoras en cuanto al formato de archivos en el que se guardan las nubes de puntos, las cuales fueron claves para la cuarta fase en la que fue necesario reconstruir una malla a partir de la nube de puntos obtenida.

Para la fase 2 se usó el motor de videojuegos Unity con el cual se desarrolló una aplicación de realidad virtual para la plataforma Samsung Gear VRm de tal forma que este fue el dispositivo de realidad virtual seleccionado para el desarrollo del presente proyecto gracias a su poco peso y portabilidad, sumado a que no requiere estar conectado a otros dispositivos mediante cables. Al finalizar esta fase la aplicación fue capaz de renderizar la nube de puntos que estaba siendo captada por los sensores de una escena de realidad virtual.

En la fase 3 se realizaron pruebas de rendimiento, tanto en el proyecto Live Scan 3D como en la aplicación desarrollada, con el fin de optimizar el rendimiento de la aplicación en el celular, para que la escena mostrada por la aplicación se viera un poco más fluida, ya que debido a la poca capacidad computacional que tiene el celular que usa el dispositivo Samsung Gear VR. Al finalizar esta fase fue posible establecer algunas condiciones de funcionamiento optimas con las cuales la escena de realidad virtual mostrada por la aplicación se ve de manera muy fluida.

Para determinar qué tan parecida a la escena real es la escena reconstruida por la aplicación se llevaron a cabo una serie de mediciones en la fase 4, las cuales permitieron establecer cuál era el margen de error con el que el proyecto Live Scan 3D estaba reconstruyendo la escena que estaban visualizando los sensores Kinect.

1. Objetivos

1.1. Objetivo General

Establecer la factibilidad técnica del proyecto Live Scan 3D para la reconstrucción 3D de una escena usando múltiples sensores Kinect.

1.2.Objetivos Específicos

- Implementar el proyecto Live Scan 3D para reconstruir una escena física usando múltiples sensores Kinect.
- Evaluar la exactitud y precisión de la reconstrucción 3D realizada con el proyecto Live Scan 3D.
- Realizar un prototipo de visualizador para un dispositivo de realidad virtual.
- Establecer las condiciones de operación adecuadas para la reconstrucción de una escena usando el proyecto Live Scan 3D.

El cumplimiento de cada objetivo se llevó a cabo mediante las fases que se pueden observar en la tabla 1.

Tabla 1.

Metodología General.

	FASE 1: IMPLEMENTAR EL PROYECTO LIVE SCAN 3D	FASE 2: CREAR UN PROTOTIPO DE APLICACIÓN PARA REALIDAD VIRTUAL	FASE 3: ESTABLECER LAS CONDICIONES MINIMAS PARA EL OPTIMO FUNCIONAMIENTO DEL SISTEMA	FASE 4: MEDICIÓN DEL ERROR EN LA RECONSTRUCCIÓN REALIZADA
ACTIVIDADES	**Se compilo el código fuente para tener los ejecutables necesarios para poner en marcha el proyecto**	Se creó una escena en Unity en la cual poder navegar con un avatar en realidad virtual	Se investigaron las capacidades técnicas del equipo de realidad virtual usado y el protocolo de comunicación que usa	Se guardaron las nubes de puntos de varias escenas a las cuales se querían medir la precisión
	Se logró conectar múltiples sensores Kinect al servidor de Live Scan 3D	Se programaron y agregaron los scripts necesarios para conectar la aplicación al servidor de Live Scan 3D	Se realizaron pruebas de rendimiento con diferentes modems para la comunicación y diferentes densidades y tamaños para las nubes de puntos	Se reconstruyo una malla a partir de las nubes de puntos guardadas para medir su precisión
	Se logró renderizar una escena usando los sensores Kinect y el proyecto Live Scan 3D	Se programaron los scripts necesarios para renderizar la información recibida por el servidor de Live Scan 3D	Se estableció el tamaño mínimo de la escena y su densidad de puntos para que funcione correctamente con una red Ethernet	Se midió la precisión de las mallas obtenidas con el proyecto Live Scan 3D usando un software especializado

2. Planteamiento del Problema

En la actualidad conseguir personal altamente capacitado para una labor especializada no es tarea fácil, en la mayoría de los casos se recurre a contratar la asesoría de un especialista en determinadas áreas pero no siempre es fácil contar con la presencia física de dicho especialista en el lugar de trabajo en el que se requieren sus servicios, esto suele repercutir en gastos adicionales de viáticos del especialista para que pueda estar presente en el lugar que se le requiere para que así pueda prestar sus servicios.

Hoy en día la tecnología nos brinda innumerables herramientas que nos ayudan a borrar la barrera de la distancia a la hora de compartir información de algún tipo. Actualmente la manera más completa que tenemos de compartir información de un ambiente es mediante videos que nos brindan información gráfica y sonora de un sitio o tema en particular. Sin embargo, no nos permite sentir una inmersión completa en el lugar y momento en el que se tomó el video y nos limita a ver un escenario desde el ángulo de un sensor lo cual limita enormemente la información que se puede obtener de una escena mediante un video 2D. No solo por la perspectiva del sensor sino también debido al nivel de experticia que tenga la persona que está realizando la toma de la escena. Es evidente que una persona con un nivel de experticia alto en un determinado tema no grabaría una escena de su interés desde el mismo ángulo que la grabaría una persona que no cuente con un nivel de experticia alto en el contexto de la escena que se está grabando.

Actualmente, muchos simuladores para entrenamiento solo cuentan con un monitor en el cual presentar datos y situaciones al usuario el cual cuenta con algún tipo de dispositivo para interactuar con el simulador, y de esta forma realizar tareas propuestas por el simulador. Dichos simuladores

presentan al usuario mundos virtuales totalmente creados por computadora, ambientes ficticios que intentan imitar al mundo real en todos los aspectos pero que no lo son, tal es el caso de los simuladores de vuelo, los cuales ofrecen una experiencia de vuelo a los usuarios sobre un mundo demasiado poligonal, es por esto que el presente proyecto busca evaluar la factibilidad técnica de reconstruir un ambiente real de trabajo de manera virtual usando múltiples sensores Kinect y unas gafas de realidad virtual para recorrer dicha reconstrucción virtual ofreciendo así al usuario una experiencia diferente al momento de ver la reconstrucción 3D de su entorno más cercano al mundo real.

En búsqueda de borrar la barrera de la distancia y facilitar el intercambio de información, este proyecto busca evaluar la factibilidad técnica de realizar una reconstrucción 3D de una escena real usando 4 sensores Kinect V2. Para esto se usó el proyecto de código libre Live Scan 3D el cual permite conectar múltiples sensores Kinect V2 a un computador de manera independiente cada uno los cuales compartirán la información obtenida por cada Kinect V2 a través de la red, de esta manera fue posible reconstruir una escena especifica desde diferentes puntos de vista usando un motor gráfico de renderizado 3D el cual recreó un escenario virtual semejante al real que pudo ser recorrido en una perspectiva de primera persona por el usuario final mediante el uso de gafas de realidad virtual para moverse a través de la escena reconstruida.

3. Marco Teórico

3.1. Voxel

Es la mínima unidad tridimensional que compone un objeto de 3 dimensiones.

3.2. Realidad Virtual

Se refiere a un entorno generado por computadora mediante algoritmos de renderizado, que busca simular un ambiente del mundo real en el cual interactuar de alguna forma.

3.3. Reconstrucción 3D

Proceso llevado a cabo por computadores que reciben información del mundo real mediante sensores capaces de percibir la forma de los objetos, dicho proceso se lleva a cabo mediante algoritmos de reconstrucción especialmente diseñados para cada caso concreto de las reconstrucciones requeridas.

3.4. Algoritmo

Conjunto de instrucciones definidas en estricto orden, que permiten realizar con éxito una determinada tarea.

3.5. Nube de Puntos

Conjunto de vertices de un sistema de coordenadas de 3 dimensiones, generalmente se representan con valores X, Y y Z, y se usan para representar la superficie externa de algún objeto, ya sea real o generado por computadora.

3.6. Malla Poligonal

Es una superficie poligonal creada a partir de un sistema de vértices que se usa como base para la creación de los polígonos que tiene la malla, por lo general las mallas poligonales suelen estar hechas por triángulos ya que son los polígonos más fáciles de calcular en una nube de puntos.

3.7. Formato PLY

Poligon File Format o PLY, es un formato de archivos de computadora, que se usa para guardar la información de las coordenadas de los vertices de una malla y la información de los poligonos que se forman usando dichos vertices, lo cual da la forma de la malla poligonal.

Los archivos PLY suelen guardar la información en el siguiente formato:

```
ply

format ascii 1.0
format binary_little_endian 1.0
format binary_big_endian 1.0
comment This is a comment!
element vertex 12
property float x
property float y
property float z
element face 10
property list uchar int vertex_index
end_header
```

3.8. Antecedentes de la Situación de Estudio

Desde el lanzamiento de la versión para PC del Kinect V2 ha sido usado en diversos tipos desarrollo e investigaciones de nuevas tecnologías. En el presente documento nombraremos algunos proyectos que han usado el Kinect V2 como sensor de profundidad para realizar escaneos y reconstrucciones de escenas reales y otros proyectos que, aunque no usan el sensor Kinect V2 nos aportan los resultados de pruebas de reconstrucción de escenarios reales usando diferentes algoritmos de reconstrucción.

3.8.1. BundleFusion: Real-time Globally Consistent 3D Reconstruction Using On-the-fly Surface Re-integration. Este Proyecto desarrollado en la universidad de Stanford (California, Estados Unidos) usa un sensor de profundidad genérico para reconstruir superficies de una escena en tiempo real basándose en un sistema de estimación de posición del sensor de profundidad, es decir, para que este proyecto funcione correctamente el sensor de profundidad usado debe contar con un giroscopio incorporado con el cual pueda reportar su posición en todo momento al sistema, de esta manera el sistema pueda estimar que parte de la escena está siendo visualizada por el sensor en cada momento y poder llevar a cabo la reconstrucción de la escena en la posición que está reportando el sensor, de esta manera, cuando el sensor pasa varias veces por un mismo punto el sistema volver a reconstruir esa parte del escenario en caso de que haya habido algún cambio desde la última vez que fue escaneada dicha parte del escenario, en resumen, este proyecto nos permite escanear y crear la superficie de una escena real y en dado caso de que alguna parte de la escena sufra algún cambio basta con volver a escanear la parte que fue modificada para que el cambio se refleje también en la reconstrucción virtual.

3.8.2. KinectFusion: Real-Time Dense Surface Mapping and Tracking. KinectFusion es un Proyecto desarrollado por el departamento de investigación de Microsoft, en rasgos generales funciona de manera similar al BundleFusion pero con la diferencia de que en este proyecto el sensor usado es el Kinect V2 y debido a que este sensor de profundidad no cuenta con giroscopio para determinar su posición debe permanecer a una distancia fija del objeto a escanear, de esta manera el sistema puede saber a cuanta distancia se encuentra del objeto que está escaneando el sensor gracias al sensor de profundidad del sensor.

Con este sensor se pueden escanear objetos en 360° pero siempre manteniendo el objeto que está siendo escaneado a una distancia fija del sensor y girando el sensor a su alrededor para obtener una reconstrucción de una mayor superficie del objeto a escanear.

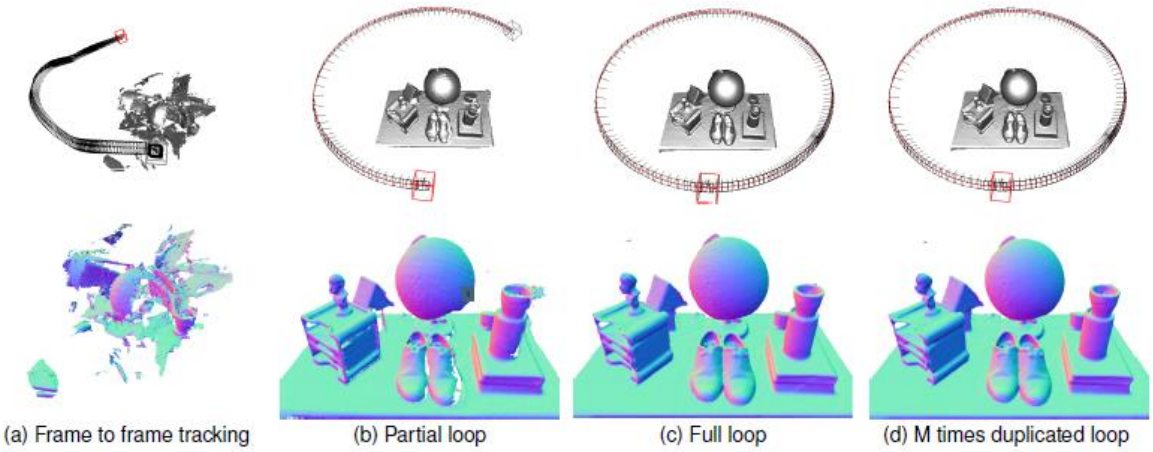

Figura 1. Resultados de escaneos parciales y escaneos de 360° obtenidos por KinectFusion.

Como se puede ver en los resultados de las reconstrucciones obtenidas con el proyecto KinectFusion es posible obtener una reconstrucción de 360° de una pequeña escena que este ubicada dentro del circulo trazado por el sensor Kinect V2 al momento de realizar un escaneo

desde todos los ángulos posibles siempre que esté dentro del ángulo de visión vertical del Kinect V2.

3.8.3. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. Este sistema de KinectFusion le permite al usuario usar un sensor Kinect estándar y moverse rápidamente dentro de una habitación para reconstruir con una alta calidad un Modelo 3D geométricamente preciso de la escena. Para lograr esto, el sistema rastrea continuamente la posición del sensor y fusiona los datos de profundidad en vivo del sensor en un solo Modelo 3D global en tiempo real. A medida que el usuario escanea la habitación se renderiza el modelo 3D de la escena física y se fusionan en el mismo modelo 3D global. La reconstrucción por lo tanto crece en Detalle a medida que se agregan nuevas medidas de profundidad. Los agujeros están llenos, y el modelo se vuelve más completo y refinado con el tiempo.

Incluso pequeños movimientos, como por ejemplo por sacudidas del sensor, dan lugar a nuevos puntos de vista de la escena y, por tanto, refinamientos al modelo. Esto crea un efecto similar a la superresolución de imágenes. agregando mayor detalle de lo que aparece visible, las reconstrucciones son de alta calidad, particularmente debido al ruido incluido en los datos de entrada y la velocidad de reconstrucción. En el modelo reconstruido también se puede mapear la textura usando el sensor RGB del Kinect.

4. Marco Tecnológico

El presente proyecto se apoyó en tecnologías existentes tanto en software como hardware que permitió llevar a cabo las tareas necesarias para alcanzar el cumplimiento de los objetivos planteados. A continuación, se describirá de manera concisa cada uno de los componentes tecnológicos que se usaran para el desarrollo del proyecto.

4.1. Live Scan 3D

Live Scan 3D es un proyecto de código abierto creado por Marek Kowalski[5] un estudiante de doctorado y asistente de investigación en la facultad de electrónica y tecnología de la información en la Universidad Tecnológica de Varsovia en Polonia.

Actualmente no es posible conectar más de un Kinect V2 a un mismo computador debido a la gran cantidad de datos que el Kinect V2 envia al computador ocupando todo el ancho de banda del bus de transferencia del computador, haciendo imposible que otro Kinect V2 pueda enviar más datos a través del bus de datos usado por los puertos USB. Es por eso que para el desarrollo de este proyecto se ha seleccionado el proyecto LiveScan3D debido a que permite conectar varios KinectsV2 a través de una conexión en red.

LiveScan3D tiene una arquitectura cliente servidor en la cual cada uno de los Kinects V2 que forman parte del sistema están conectados a un computador y cada uno de los computadores están

[5] Link del perfil personal de Marek Kowalski de la Universidad Tecnológica de Varsovia: http://home.elka.pw.edu.pl/~mkowals6/doku.php?id=wiki:about_me

conectados en red, al menos uno de los computadores debe actuar como servidor. El servidor puede realizar calibración, filtrado, sincronización de todos los frames enviados por los Kinects y renderizado de la nube de puntos obtenida de todos los Kinects conectados a la red.

4.2. Kinect V2

Kinect V2 es el sensor de movimiento creado por Microsoft para ser lanzado junto con su consola Xbox One y de esta manera competir con los sensores de movimiento de las demás empresas de videojuegos en el mercado (WiiMote y PlaystationMove) el cual cuenta con una cámara RGB la cual capta imágenes en full HD (1920 x 1080 P) a 30 FPS y una cámara infrarroja con una resolución de (512 x 424 P) la cual le permite detectar imágenes aun en la oscuridad y es la encarga de enviar la información de la profundidad de la imagen. Debido a que la calidad de la imagen detectada por el Kinect V2 a aumentado, también lo ha hecho el tamaño de la información enviada por el Kinect, como resultado de este incremento Kinect V2 utiliza el protocolo de comunicación de serie universal USB 3.0 ya que necesita un bus con suficiente ancho de banda para poder enviar toda la información captada por los sensores a una velocidad optima, es esta característica la que hace imposible conectar más de un dispositivo Kinect V2 a un mismo computador, ya que todo el ancho de transferencia de datos es ocupado por completo por el Kinect V2 cuando está transmitiendo datos. Esta versión de Kinect no posee motor de inclinación ya que su rango de visión ha sido mejorado y ahora tiene un campo de visión de 70° en horizontal y 60° en vertical, pero en caso de que se requiera la inclinación del Kinect puede ser graduada manualmente. También posee un micrófono para usar reconocimiento de voz y usar algunos comandos por voz.

Aunque lógicamente el Kinect V2 fue creado para funcionar con muchos juegos, especialmente conversiones de Xbox One, Kinect V2 para Windows aprovechará toda la versatilidad del PC para usarse en todo tipo de aplicaciones, desde interfases para personas con problemas de movilidad, a control de máquinas, sistemas de vigilancia, medicina, y otras tareas.

Algunos desarrolladores ya lo están probando con las gafas de realidad virtual Oculus Rift, recién adquiridas por Facebook lo cual Permitirá añadir detección de movimientos al entorno virtual, para que puedas coger objetos e interactuar con otras personas.

4.3. Samsung Gear VR

El Gear VR es un accesorio para realidad virtual creado por Samsung para ser usado en conjunto con sus celulares de alta gama que son capaces de renderizar entornos virtuales en modo VR, esta es una característica poco común en los celulares y aun los celulares que son capaces de renderizar escenarios para realidad virtual tienen sus limitaciones, por lo general las experiencias de realidad virtual ofrecidas por los dispositivos móviles suelen ser mayoritariamente videos prerrenderizados de una escena en concreto ya que para poder renderizar un escenario virtual completamente desde cero se requiere una gran capacidad de procesamiento gráfico, debido a que para crear la sensación de inmersión en un escenario de realidad virtual se requiere renderizar dos veces la misma escena pero con una pequeña diferencia en el ángulo de visión de cada escena que se está renderizando, esto para simular el ángulo de visión de cada ojo y de esta manera poder engañar al cerebro y crear una sensación real de inmersión en un mundo virtual. Aunque el renderizado de escenas de realidad virtual es bastante costoso a nivel computacional Samsung cuenta con celulares capaces de reconstruir escenarios virtuales, aunque con muy poca calidad grafica para no afectar el

rendimiento de la aplicación y de esta manera no dañar la experiencia de usuario en la simulación de realidad virtual.

4.4. Point Cloud Library (PCL)

La biblioteca de nubes de puntos (o PCL) es un proyecto abierto a gran escala para imágenes en 2D / 3D y procesamiento de nubes de puntos. El marco PCL contiene numerosos algoritmos de vanguardia que incluyen filtrado, estimación de características, reconstrucción de superficies, registro, ajuste de modelos y segmentación. Estos algoritmos se pueden usar, por ejemplo, para filtrar valores atípicos de datos ruidosos, unir nubes de puntos 3D, segmentar partes relevantes de una escena, extraer puntos clave y descriptores de cálculo para reconocer objetos en el mundo según su aspecto geométrico y crear superficies a partir de Apunte las nubes y visualícelas, por nombrar algunas.

5. Marco Legal

Para el desarrollo del sistema estéreo de cámaras Kinect V2 para la reconstrucción 3D se usaron diferentes tecnologías preexistentes, cada una de las cuales cuenta con su propia licencia de distribución legal para los proyectos que implementan sus tecnologías. A continuación, se describe la licencia de uso de cada uno de los componentes tecnológicos utilizados en el presente proyecto:

5.1. LiveScan3D

El proyecto LiveScan3D se encuentra bajo la licencia MIT[6] lo cual permite el uso del código de este proyecto incluso dentro de proyectos de software privativo.

5.2. Point Cloud Library

PCL está bajo los términos de la licencia BSD[7] de 3 cláusulas y es un software de código abierto. Es gratis para uso comercial y de investigación.

5.3. Samgun GearVR

Las gafas Samgung GearVR cuentan con dos versiones, una versión para desarrolladores que es la que se usara para el desarrollo del presente proyecto y otra versión para uso comercial en cuyo caso solo se permite su uso con aplicaciones registradas y descargadas directamente de la Oculus Store, para el caso de la versión de desarrollo puede usar igualmente aplicaciones de la Oculus Store y cualquier aplicación desarrollada para la plataforma contara en primera instancia con un Oculus Signature File (OSIG) que se encargara de firmar las aplicaciones desarrolladas para que puedan ser ejecutadas por el dispositivo de realidad virtual y una vez se desee comercializar la aplicación está deberá ser subida a la plataforma de Oculus Store en donde después de ser evaluada

[6] Esta licencia es una licencia de software libre permisiva lo que significa que impone muy pocas limitaciones en la reutilización y por tanto posee una excelente Compatibilidad de licencia. La licencia MIT permite reutilizar software dentro de Software propietario.
[7] Las licencias BSD son una familia de licencias de software libre permisivas, que imponen restricciones mínimas en el uso y distribución del software cubierto.

para verificar el cumplimiento de las medidas de seguridad se procedera a firmar la aplicación con un OSIG para uso comercial y se publicara la aplicación en la Oculus Store.

6. Fase 1. Puesta en Marcha del Proyecto Live Scan 3D

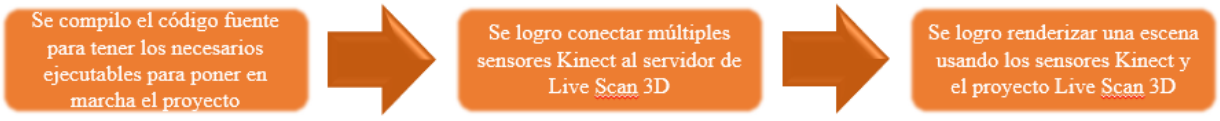

Figura 2. Actividades realizadas durante la fase 1.

En el repositorio de GitHub[8] del proyecto LiveScan3D se puede encontrar el código fuente del proyecto y la versión 1.0.0 del proyecto compilado, para la realizar una reconstrucción 3D se debe descargar los archivos compilados de la versión 1.0.0 pero no se tendrían las ultimas actualizaciones del código que es constantemente actualizado y corregido por la comunidad de desarrolladores de GitHub por consiguiente para el desarrollo del presente proyecto se compilaron y se crearon los archivos binarios necesarios para la puesta en marcha del sistema a partir del código fuente del proyecto LiveScan3D que se encuentra disponible en los repositorios de la cuenta de GitHub del estudiante de Doctorado de la universidad de Warsaw Marek Kowalski y de ser necesario se realizaran correcciones al código fuente según las necesidades del presente proyecto bajo la licencia de software MIT.

[8] GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora.

6.1. Descargando el Código Fuente del Proyecto Live Scan 3D

Para descargar el código fuente hay que ir al perfil de GitHub de Marek Kowalski[9], una vez en la página de GitHub hay dos opciones para descargar el repositorio del proyecto, la primera es copiando el link del repositorio que proporcionan en las opciones desplegables del botón "Clone or download."

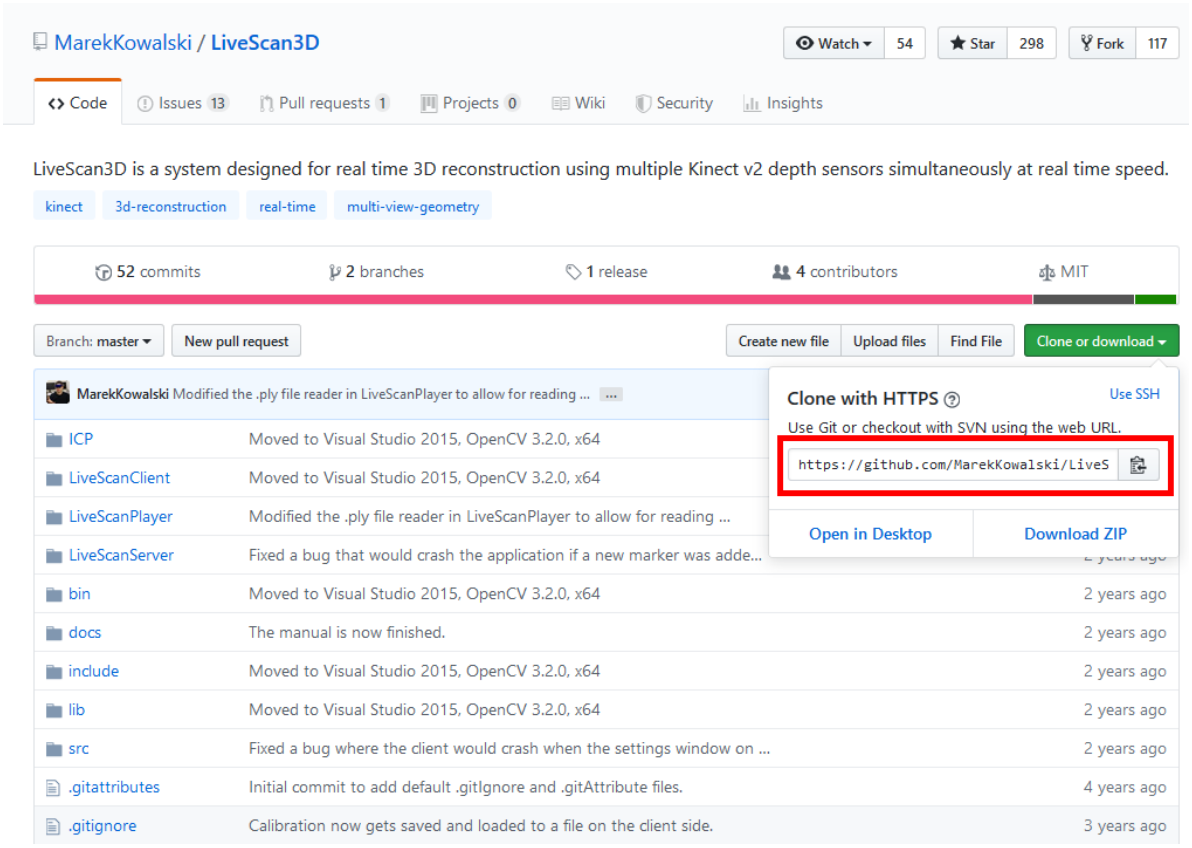

Figura 3. Link de descarga del repositorio

[9] Link del perfil de Marek Kowalski en GitHub: https://github.com/MarekKowalski/LiveScan3D

Con este link se podrá hacer checkout al repositorio del proyecto usando cualquier herramienta de control de versiones para obtener una copia local del código fuente del proyecto.

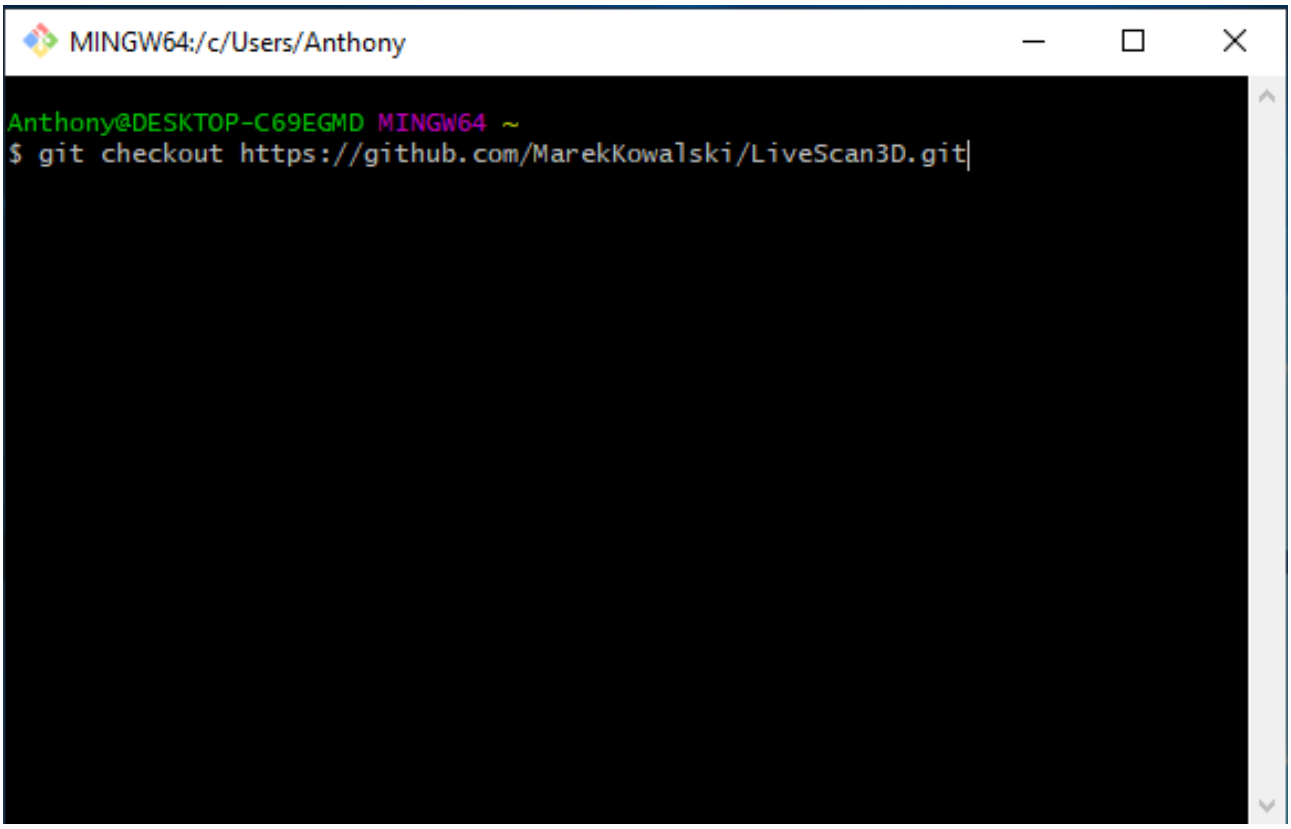

Figura 4. Checkout usando la consola de Git.

De esta manera la herramienta Git creara una copia del proyecto en la carpeta que nosotros especifiquemos. La segunda manera es dando clic en el botón “Download ZIP” de las opciones que nos muestra el botón “Clone or download”, de esta manera obtendremos una carpeta comprimida con el código fuente del proyecto.

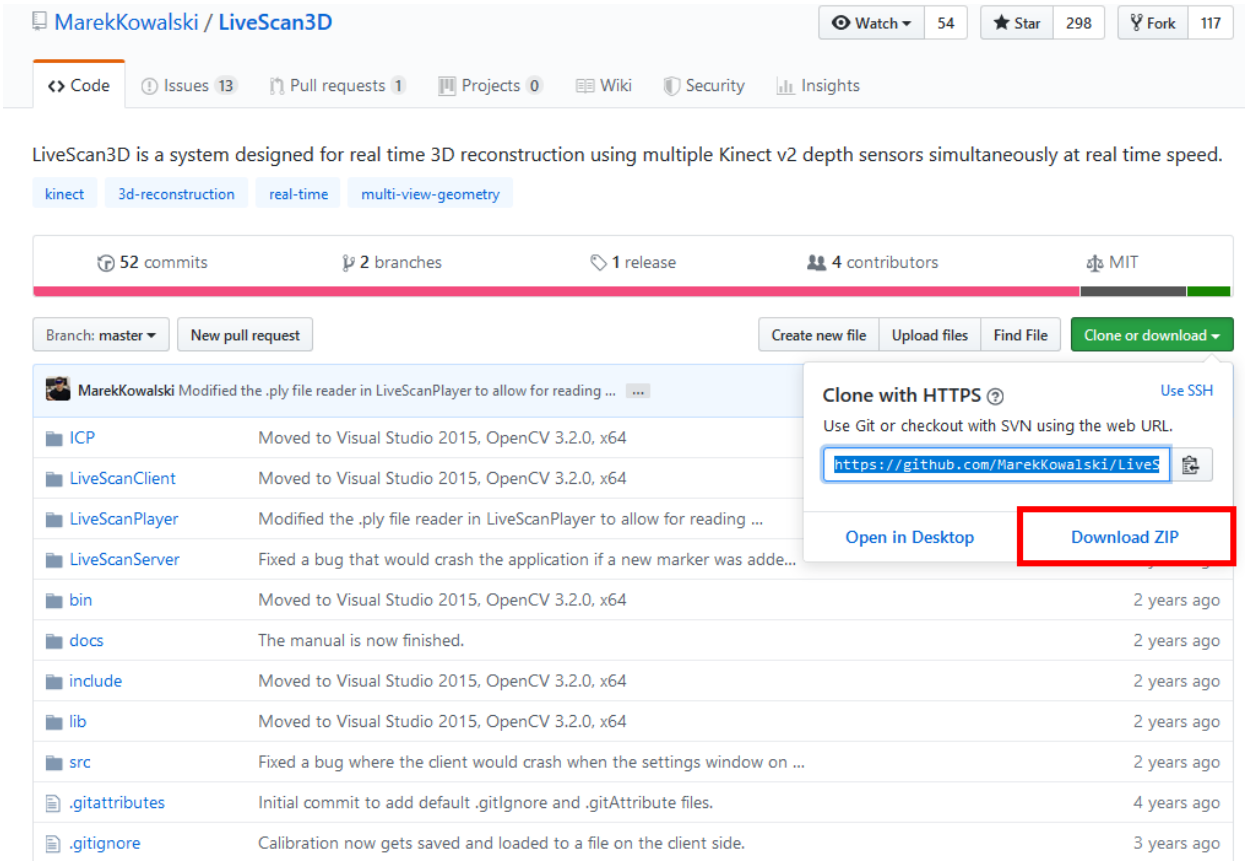

Figura 5. Opción para descargar la carpeta comprimida del código fuente del proyecto.

Una vez finalizada la descarga del código fuente para el proyecto por cualquiera de los dos medios se tendrá un proyecto de Visual Studio listo para trabajar en él, hacer los cambios que necesiten y compilar los ejecutables del proyecto siempre que se necesite probar nuevas características.

6.2. Compilando el proyecto con Visual Studio

Una vez se haya descargado el repositorio del proyecto se tendrá una carpeta que se ve de la siguiente forma:

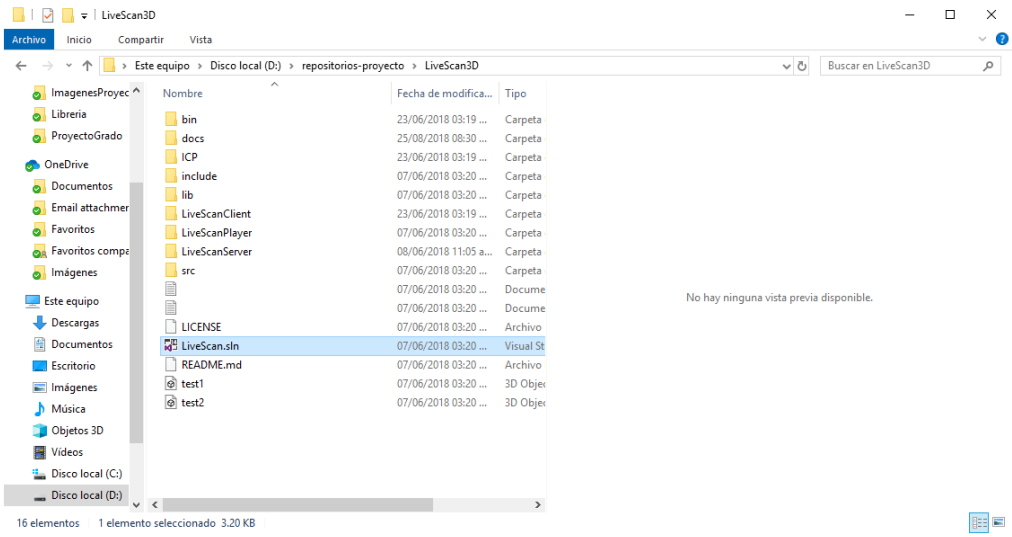

Figura 6. Estructura de la carpeta del proyecto LiveScan3D.

A continuación, se debe dar doble clic el proyecto de Visual Studio “LiveScan.sln” para abrirlo con el editor de código de Visual Studio.

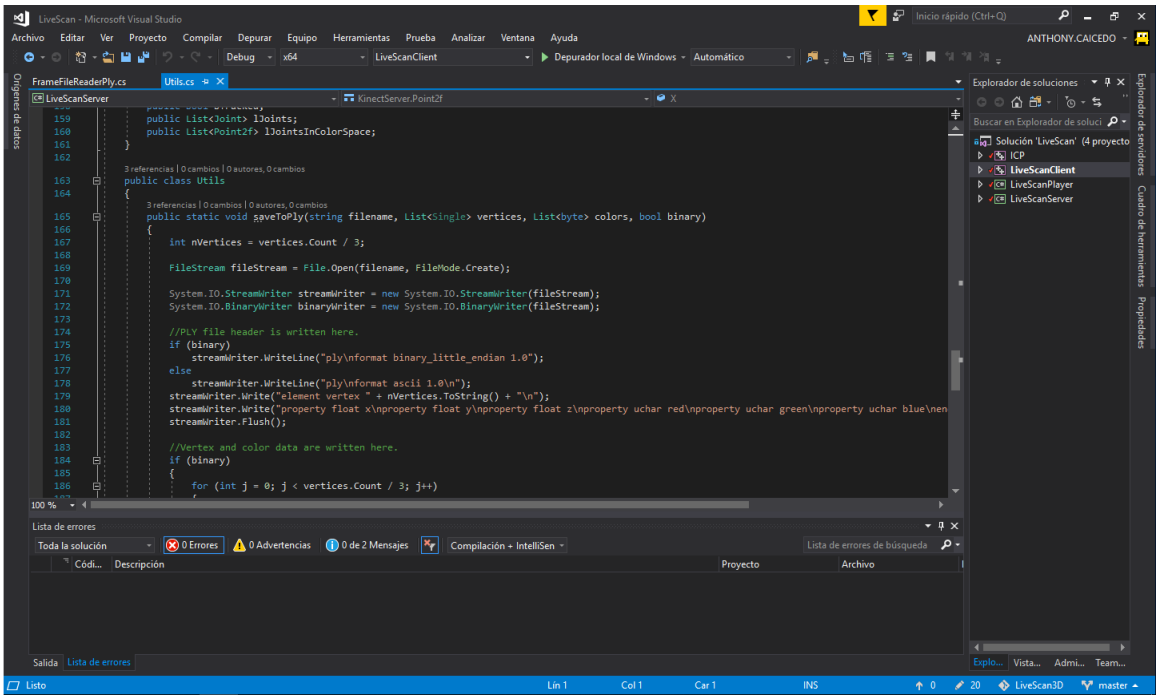

Figura 7. Proyecto LiveScan3D desde el editor de Visual Studio.

Una vez se tenga el proyecto abierto en Visual Studio se podrá editar o agregar código al proyecto según las necesidades. Como se aprecia en la Figura 7 la solución del proyecto consta de 4 proyectos independientes los cuales serán compilados cada vez que se compile la solución de Live Scan o podrán ser compilados individualmente si así se desea, de cualquier forma ya se podra agregar código y añadir funcionalidades o corregir errores que tenga el proyecto.

6.3. Conectando el Sensor Kinect V2

Antes de conectar el sensor Kinect V2 se tendrá que instalar el SDK del sensor Kinect V2 Kinect for Windows SDK 2.0 [10]en cada uno de los computadores en los que se vaya a conectar un Kinect V2, una vez que el SDK del Kinect este instalado el computador ya debería reconocer el Kinect cuando se conecte y ya se podría comenzar a enviar la nube de puntos capturada por el sensor Kinect V2.

[10] Link de descarga de la SDK de Kinect V2 para Windows: https://www.microsoft.com/en-us/download/details.aspx?id=44561

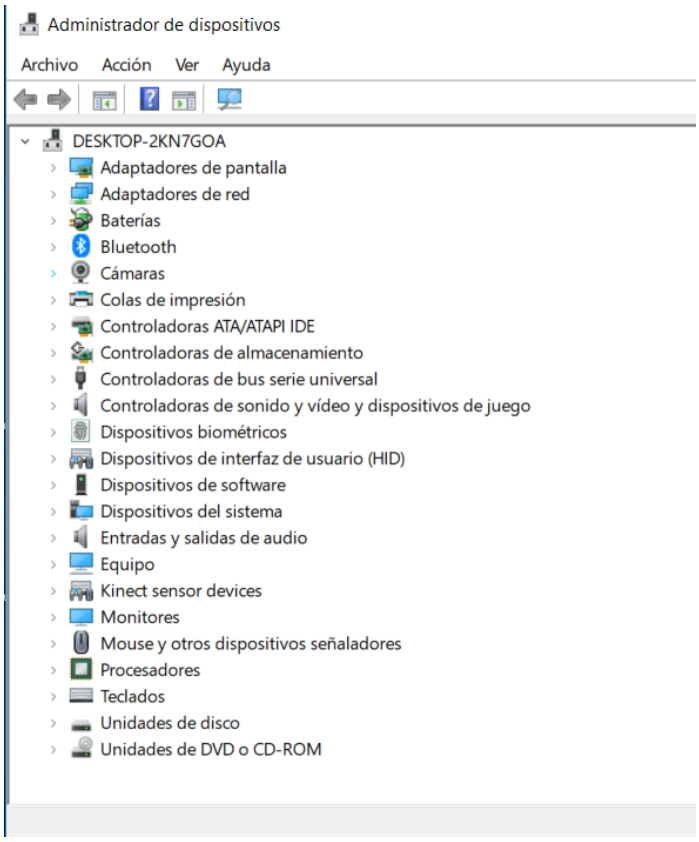

Figura 8. Kinect aparece en la lista del administrador de dispositivos del panel de control.

Cuando se conecte el Kinect y el SDK este correctamente instalado debería aparecer en la lista de dispositivos conectados tal como muestra la figura 8.

6.4. Iniciando el Cliente de LiveScan3D

Una vez que el computador ya reconozca el Kinect correctamente hay que ir a la carpeta en la que se encuentren los ejecutables del proyecto, ya sea que los que se hayan descargado o compilado a partir del código fuente, y ejecutar el cliente de LiveScan3D.

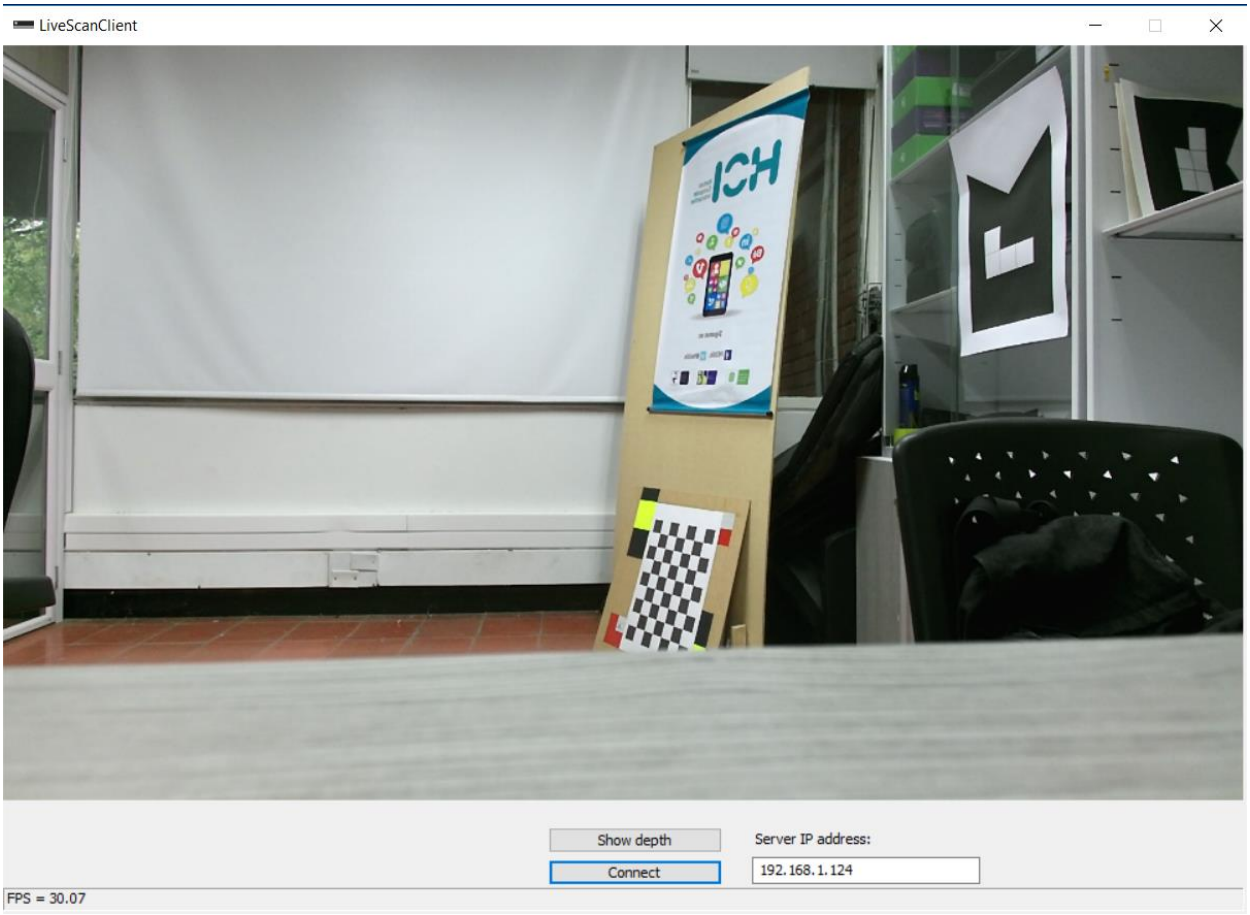

Figura 9. LiveScanClient mostrando la imagen del KinectV2.

Cuando el cliente se inicialice mostrara la imagen que está obteniendo del Kinect de lo contrario la ventana aparecerá en blanco y un mensaje en la parte inferior izquierda mostrara un mensaje de error. Una vez que el cliente este mostrando la imagen del Kinect correctamente se debe colocar la dirección IP del computador que actuara como servidor en la caja de texto que esta frente al botón “Connect”.

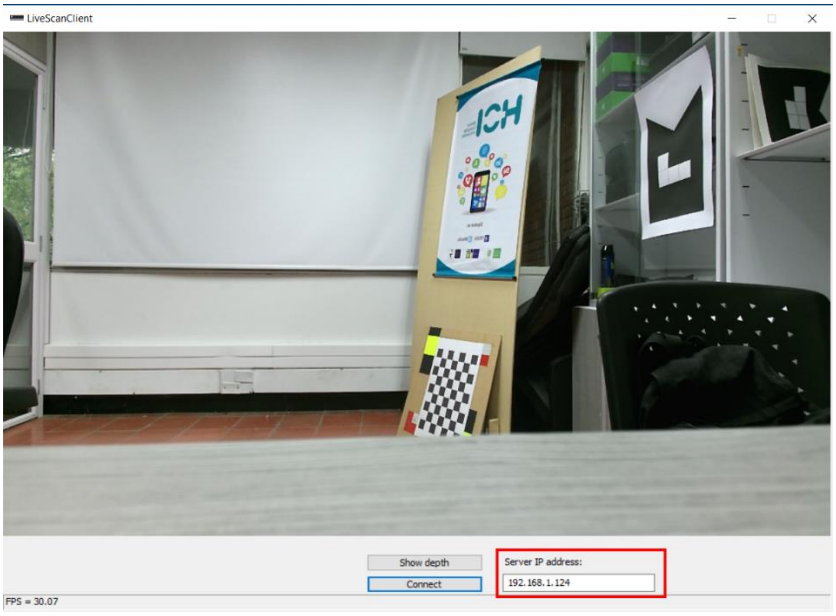

Figura 10. Caja de texto de la dirección IP del servidor.

Una vez se haya ingresado la dirección IP correcta hay que dar clic en “Connect” si el cliente no logra conectar con el servidor mostrara un mensaje de error en la parte inferior izquierda de lo contrario mostrara la cantidad de FPS que está enviando al servidor.

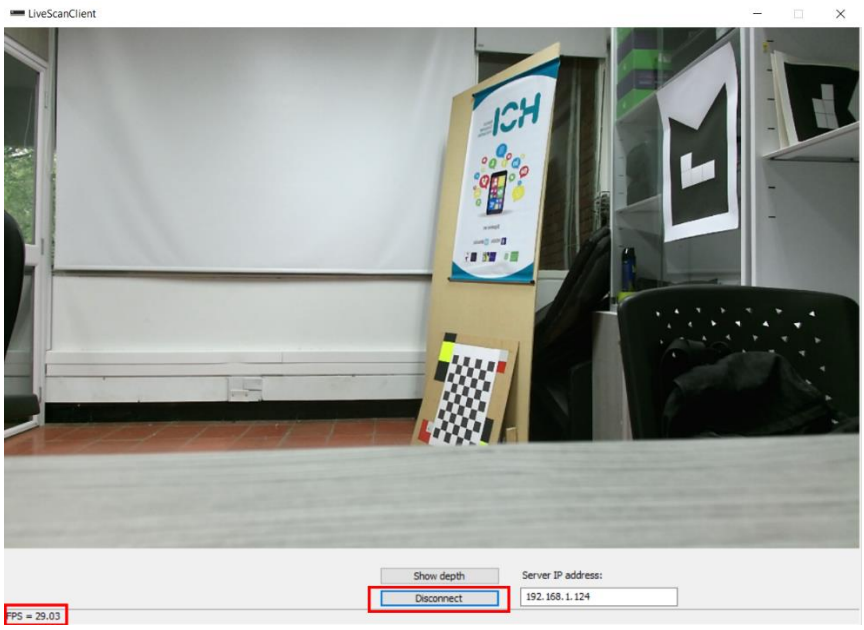

Figura 11. Cliente Live Scan conectado y enviando sus FPS al servidor.

6.5. Iniciando el servidor de Live Scan

Hay que ir nuevamente a la carpeta que contiene los ejecutables del proyecto, pero esta vez hay que dar doble clic en el ejecutable “LiveScanServer” y debería abrirse una ventana como la siguiente:

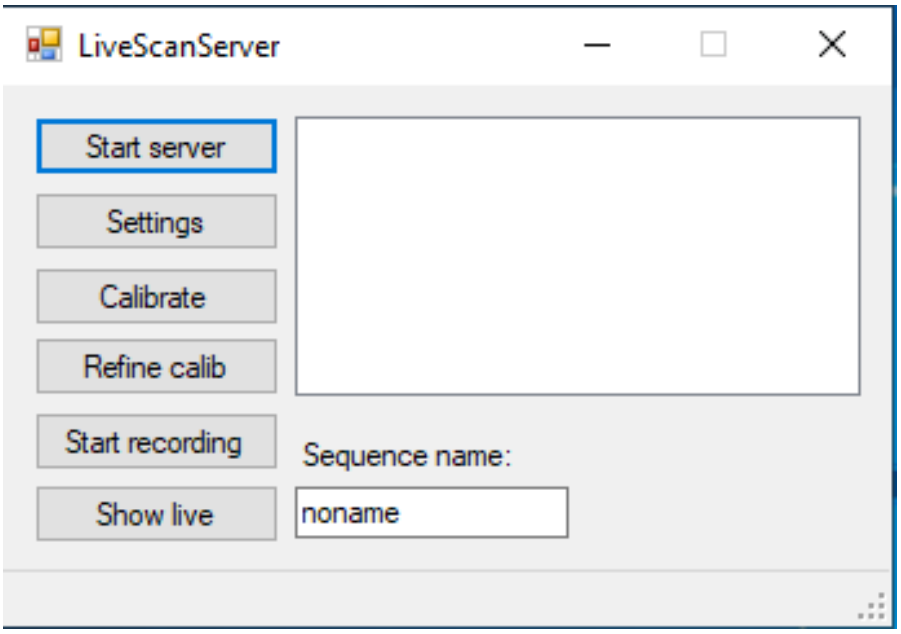

Figura 12. Interfaz del servidor de LiveScan.

Una vez se haya abierto la ventana del servidor de LiveScan hay que dar clic en el botón “Start server” para que el servidor comience a escuchar a los clientes que se conecten a él, después se debe abrir el Live Scan Cliente como se explicó en el punto anterior y colocar la dirección IP del equipo en el que se está ejecutando el servidor y dar clic en “Connect” una vez que el cliente logre conectar con el servidor este mostrara un listado de los clientes que estén conectados en el momento y de los cuales estará recibiendo la nube de puntos.

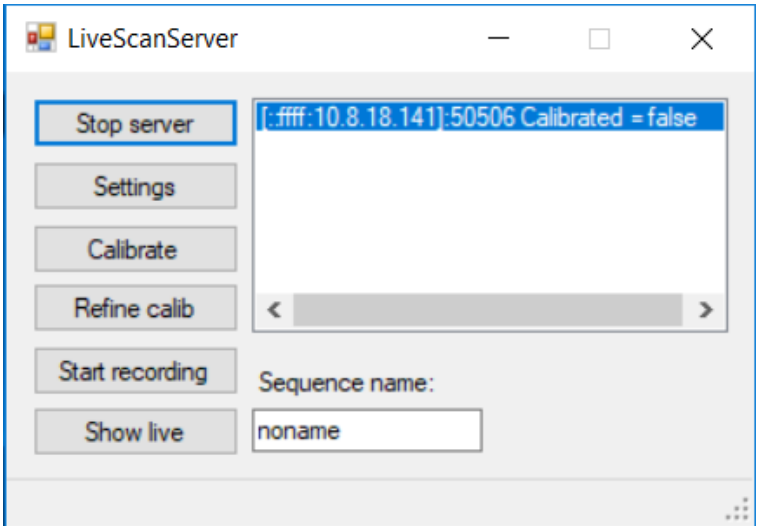

Figura 13. LiveScanServer con clientes funcionando.

Una vez que al menos un cliente esté conectado al servidor, este ya se encontrara procesando la nube de puntos, aunque la ventana de renderizado de la escena no esté abierta, para poder ver la escena que el Kinect está visualizando hay que dar clic en el botón “Show live”.

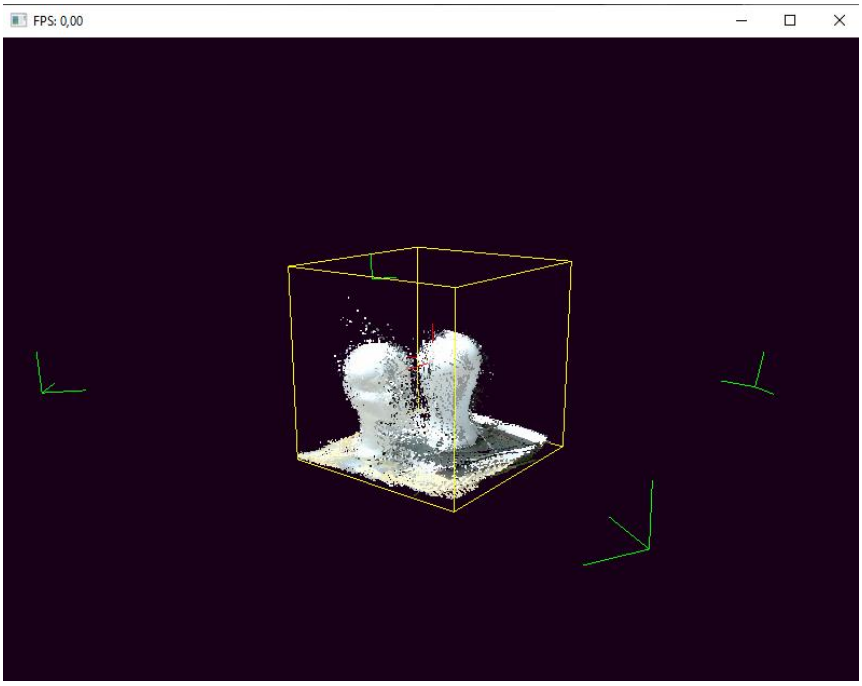

Figura 14. Ventana de renderizado de la escena.

Una vez que la ventana de renderizado este mostrando la nube de puntos que está recibiendo de los distintos clientes habrá que calibrar los sensores para obtener una reconstrucción más fiel a la escena real, para este procedimiento de calibración remitiré al lector al manual de calibración que viene junto con el proyecto Live Scan 3D.[11]

7. Fase 2. Prototipo de Aplicación para Realidad Virtual

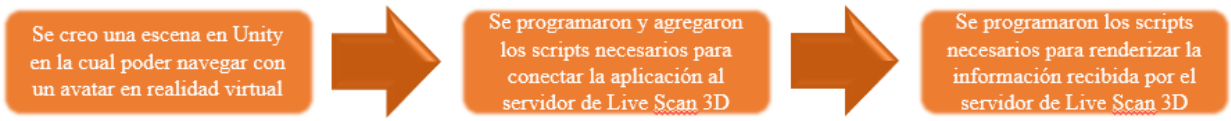

Figura 15. Actividades realizadas durante la fase 2.

El prototipo de la aplicación para realidad virtual se desarrolló con el motor para videojuegos Unity de esta manera es posible crear builds de la aplicación para diferentes dispositivos de realidad virtual, ya que todos ofrecen un rendimiento diferente, para el presente proyecto la aplicación fue creada para dispositivos soportados por las gafas de realidad virtual Samsung Gear VR ya que es un dispositivo cómodo de usar debido a que no necesita el uso de cables externos conectados a un computador encargado de procesar el renderizado grafico de la aplicación, en consecuencia a esto el dispositivo móvil usado para ejecutar la aplicación de realidad virtual será el encargado de realizar el renderizado de la escena 3D debido a esto no se podría llegar a tener una fluidez de imagen como la que se tendría si usáramos el procesamiento grafico de una GPU

[11] Dentro de la carpeta “docs” que viene en la carpeta del proyecto Live Scan 3D se encuentra el manual del software en el cual explican más detalladamente como calibrar los sensores.

pero por fortuna la tecnología avanzada cada día a pasos agigantados y muy probablemente pronto se produzca un dispositivo móvil con el poder de procesamiento de una GPU.

Para el desarrollo del presente proyecto el dispositivo móvil usado fue Samsung Galaxy S6 Edge por lo tanto el renderizado de la escena en la aplicación de realidad virtual está sujeto a las limitaciones técnicas de este equipo.

7.1. Especificaciones Técnicas del Samsung Galaxy S6 Edge

- Procesador. Velocidad de la CPU 2.1GHz, 1.5GHz. Tipo CPU Octa-Core.
- Pantalla. Tamaño 5.1" (129.2mm) Resolución 2560 x 1440 (Quad HD)
- Cámara principal - resolución CMOS 16.0 MP.
- Memoria. RAM 3 GB.
- Red / Plataforma. Multi-SIM SIM individual.
- Conectividad. ANT+ Sí
- OS Android.

7.2. LiveScanVR

LiveScanVR es el proyecto de aplicación para realidad virtual que nace a partir del proyecto LiveScan creado por Marek Kowalski para llevar a cabo la prueba de concepto propuesta por este proyecto en el que se demuestra que es posible visualizar la reconstrucción 3D de puntos de una escena real de manera remota mediante un dispositivo de realidad virtual, en este caso las gafas de

realidad virtual Samsun Gear VR. El proyecto LiveScanVR se puede descargar desde su repositorio de GitHub[12].

7.2.1. Estructura del proyecto LiveScanVR. En la siguiente imagen se puede apreciar la estructura de carpetas del proyecto LiveScanVR en las cuales se encuentran organizados todos los componentes necesarios para la correcta visualización de la reconstrucción en las gafas Samsung Gear VR.

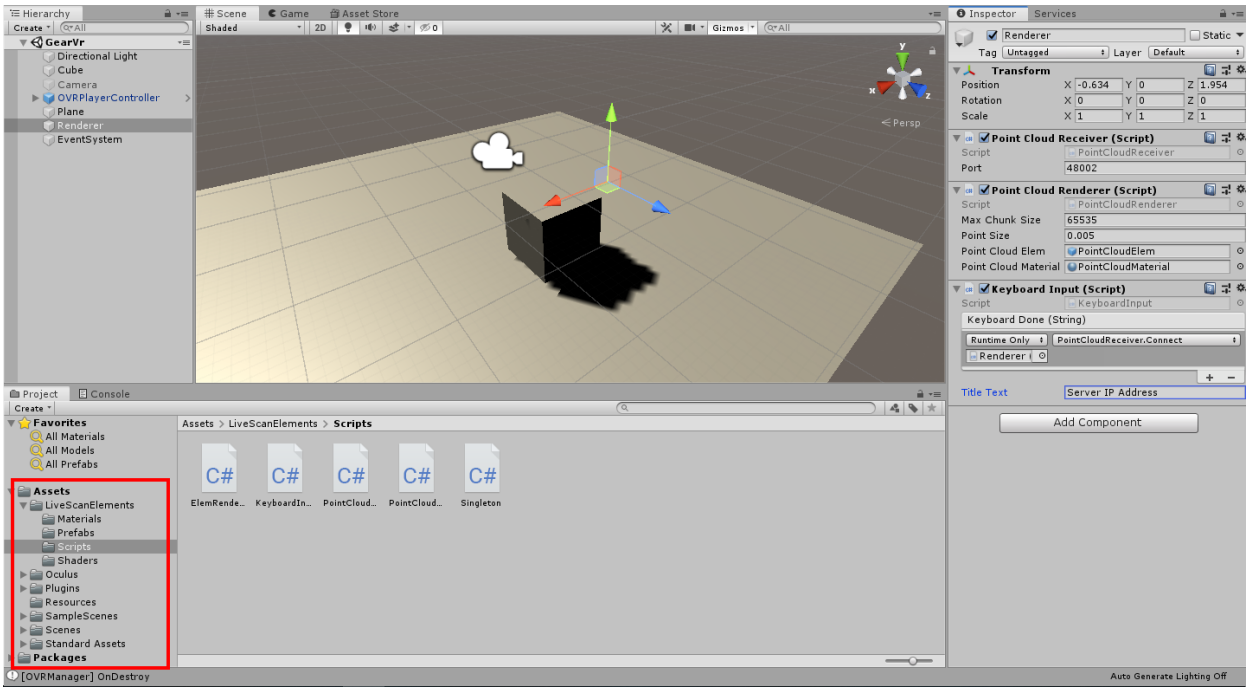

Figura 16. Proyecto LiveScanVR en Unity.

[12] Link de descarga del proyecto LiveScanVR: https://github.com/Hexx2bin/LiveScanVR

Dentro del proyecto LiveScanVR se encuentra una carpeta llamada “LiveScanElements” dentro de esta carpeta se encuentran todos los elementos desarrollados para el correcto funcionamiento del proyecto LiveScanVR:

- Materials: esta carpeta contiene el material que se le aplicara a la textura de la nube de puntos.
- Prefabs: dentro de esta carpeta se encuentra el prefab que se encargara de renderizar la nube de puntos.
- Scripts: aquí se encuentran los scripts que contienen el código necesario para iniciar la comunicación con el servidor y renderizar la información recibida.
- Shaders: en esta carpeta se encuentra el script del shader que se encargara de dibujar cada punto de la nube de puntos.
-

Dentro de la carpeta “Oculus” se encuentra la SDK de Oculus para todos sus dispositivos de realidad virtual, esta SDK puede ser descargada gratuitamente desde la asset store de Unity y se agregara automáticamente si lo descargamos desde la pestaña “Asset Store” de Unity.

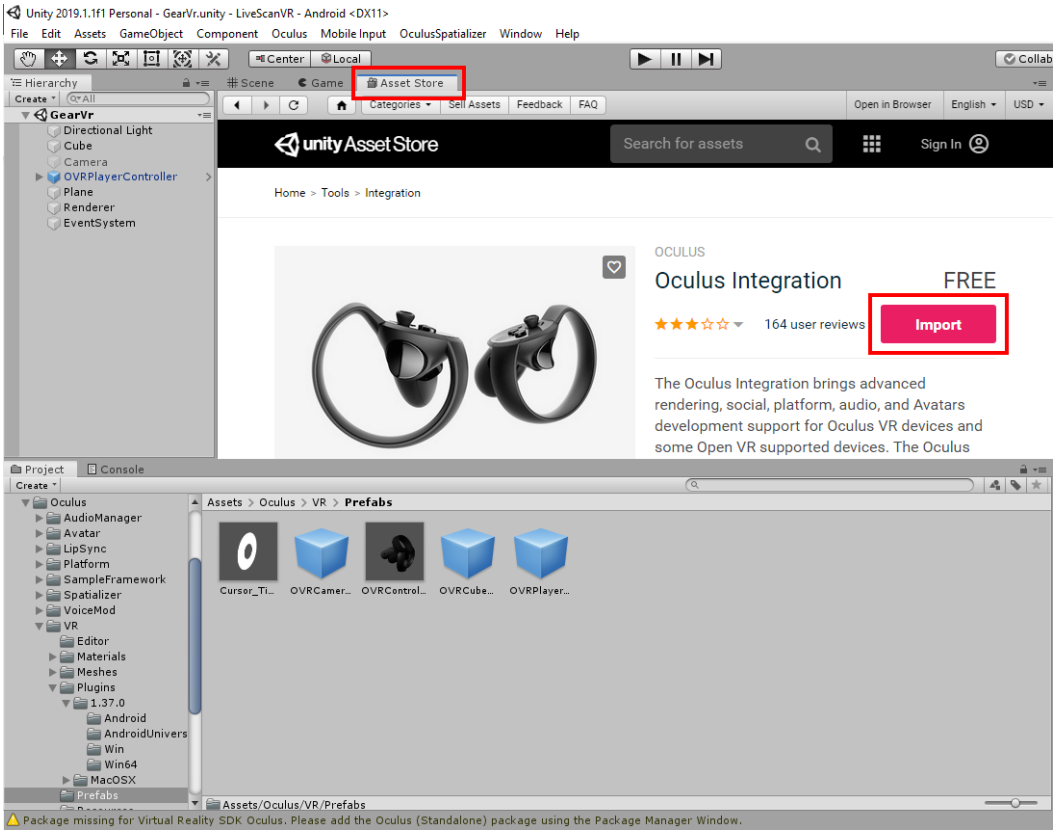

Figura 17. Asset Store de Unity.

Al dar clic en el botón “Import” desde la pestaña de la Asset Store los paquetes se agregarán automáticamente al proyecto, en futuras actualizaciones de la SDK de Oculus será necesario importar las nuevas versiones para trabajar siempre con las ultimas características de los dispositivos de realidad virtual.

En la carpeta “Plugins” se encuentran las librerías externas necesarias para que la aplicación funcione correctamente.

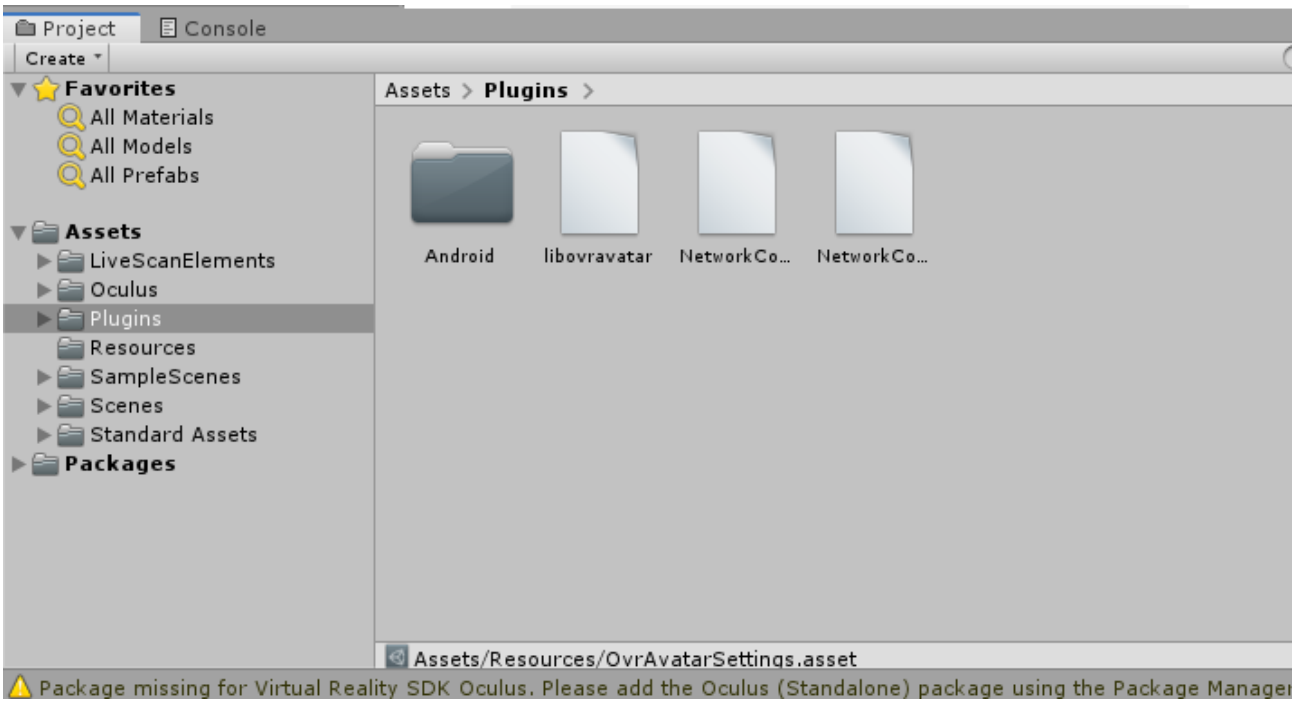

Figura 18. Plugins del proyecto LiveScanVR.

Las librerías que usamos en el proyecto son:

- Libovravatar: Es la librería para el correcto funcionamiento del avatar que actuara como personaje virtual en la escena. Contiene todas las funciones que controlan el funcionamiento de la visión en realidad virtual y el control del movimiento del avatar en la escena.
- NetworkCommunication: Está librería creada por Marek Kowalski, creador del proyecto LiveScan3D, permite a la aplicación comunicarse con el servidor del proyecto LiveScan3D para que la aplicación de realidad virtual reciba la nube de puntos que el servidor está visualizando.

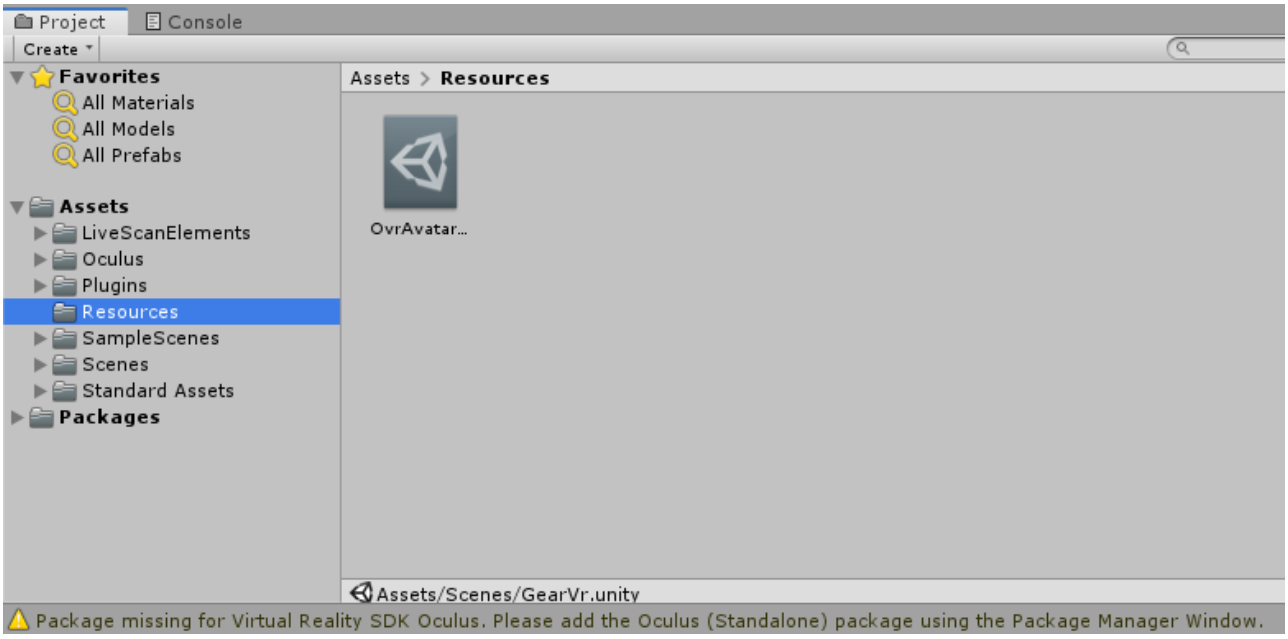

Figura 19. Resources del proyecto LiveScanVR.

La carpeta “Resources” contiene todos los assets usados en el proyecto, en este caso solo usamos el archivo de configuración del avatar de realidad virtual.

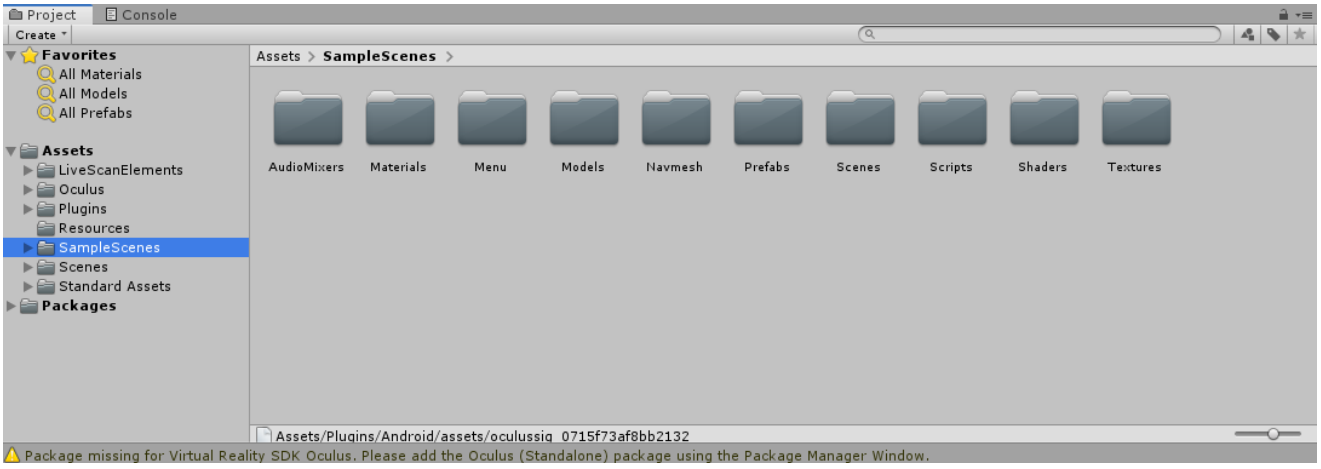

Figura 20. SamplesScenes del proyecto LiveScanVR.

Esta carpeta es agregada por el SDK de Oculus y contiene Scenes de muestra para utilizar con los dispositivos de realidad virtual de Oculus y todos los recursos usados en dichas escenas.

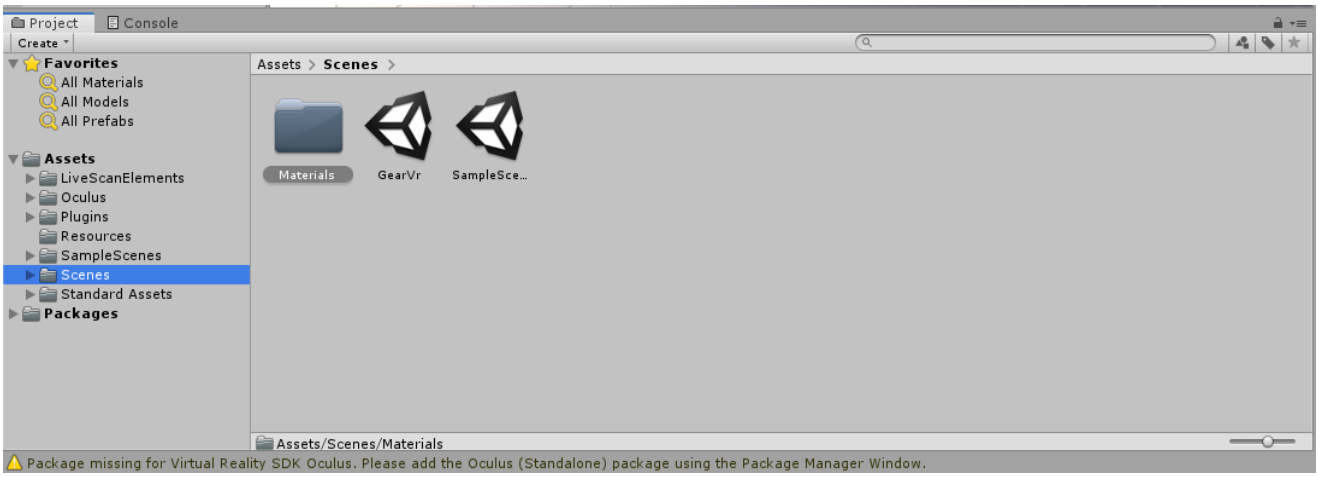

Figura 21. Scenes del proyecto LiveScanVR.

La carpeta “Scenes” contiene las escenas creadas para este proyecto y dentro de la carpeta “Materials” se encuentran los materiales utilizados en estas escenas.

Las demás carpetas “Standard Assets” y “Package” son carpetas que Unity crea por default para todos los proyectos y no contienen información relevante para el desarrollo de la aplicación LiveScanVR.

7.2.2. Escena GearVR. Esta es la escena principal que se visualiza en la aplicación de realidad virtual al usar las gafas Samsung Gear VR.

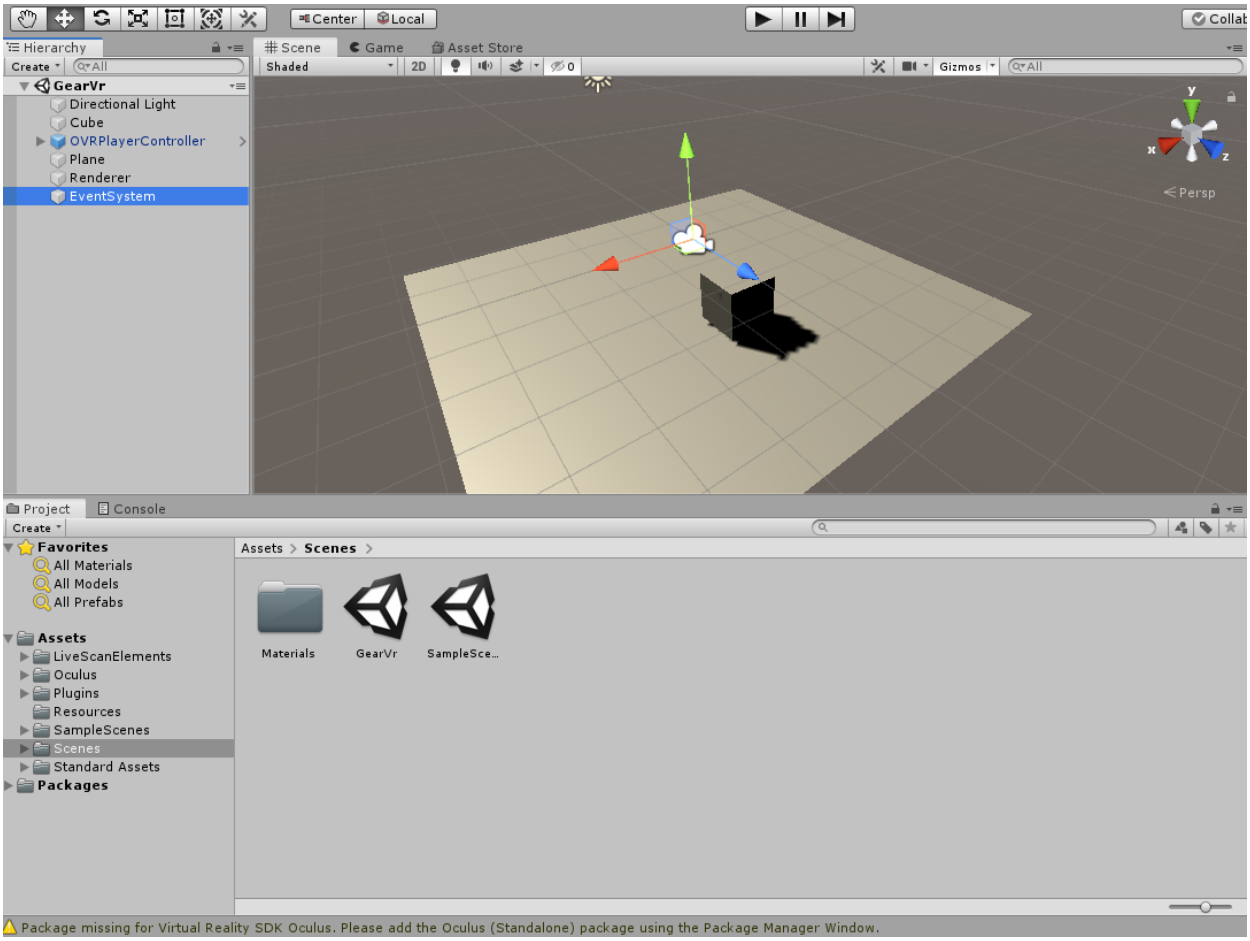

Figura 22. Escena GearVR.

Esta es una escena sencilla que contiene los elementos necesarios para renderizar la nube de puntos que el servidor de LiveScan3D envía a la aplicación y un plano que permite al avatar de realidad virtual caminar sobre ese plano junto con un cubo que nos sirve de referencia en el mundo virtual.

7.2.3 Componentes y su funcionamiento. Dentro de la escena varios componentes son esenciales para crear la experiencia de realidad virtual, los cuales son:

- OVRPlayerController: Este prefab viene con la SDK de Oculus y contiene las cámaras necesarias para visualizar la escena en realidad virtual.
- Renderer: Este componente es el encargado de renderizar la nube de puntos que recibe del servidor de LiveScan3D, para este trabajo el renderer tiene tres scripts ensamblados los cuales son:

    Point Cloud Receiver: Este es el script encargado de establecer la conexión con el servidor de Live Scan 3D y de mantener el flujo de envío y recepción de los mensajes que contienen la información de la nube de puntos.

    Point Cloud Renderer: En este script se procesa la información recibida mediante el Point Cloud Renderer y con esa información se crean dublicados del prefab Point Cloud Elem cuyo tamaño se puede especificar en la variable PointSize y se le agrega el color especifico usando el shader Point Cloud Material.

    Point Cloud Elem: Este prefab es el que actuara como Voxel en el proceso de renderizado que hace el Point Cloud Renderer, por cada punto que el Point Cloud Renderer necesite mostrar, creara un duplicado de este prefab y le asignara un tamaño, posición y color correspondiente en la escena.

    Elem Renderer: Este Script solo se encarga de actualizar el tamaño y el color del Point Cloud Elem con la información que reciba en su método UpdateMesh.

Keyboard Input: Este Script se usa como interfaz para inicializar la conexión con el servidor usando la ip que se establezca en el método Start del script.

7.2.4 Build del .apk de la Aplicación LiveScanVR. Para el desarrollo del presente proyecto se produjo un ".apk" (Android Package) debido a que el dispositivo de realidad virtual que se usara para la prueba de concepto propuesta en este proyecto es el Samsung Gear VR y este dispositivo usa celulares Samsung como dispositivos de visualización para la realidad virtual y gracias a esto se debe generar una aplicación que funcione en el sistema operativo Android que sea usado por el celular Samsung que se use para ejecutar la aplicación, en el desarrollo del presente proyecto se usa el Samsun Galaxy S6 Edge. Es posible generar el build final para otro dispositivo de realidad virtual, pero en este proyecto se usa las gafas Samsung Gear VR solo se explicara cómo generar un build para este dispositivo de realidad virtual.

Las aplicaciones para dispositivos de realidad virtual deben llevar una firma electrónica que las habilita para funcionar en dichos dispositivos, si se compila el ".apk" sin incluir la firma electrónica proporcionada por Oculus la aplicación se cerrara sola al momento de abrirla en el dispositivo ya que no tendrá la autorización para funcionar, para firmar la aplicación se debe incluir un archivo de firma digital proporcionado por Oculus a través de su página para desarrolladores, dicha firma digital es única para cada dispositivo ya que se genera a partir del código de identificación de cada dispositivo, los pasos para generar el archivo de firma digital de Oculus se pueden encontrar en su página en el apartado de herramientas para los desarrolladores de Oculus[13].

[13] Link de la pagina de Oculus para generar el archivo de firma electrónica para aplicaciones del Samsung Gear VR: https://dashboard.oculus.com/tools/osig-generator/

Una vez que tengamos el archivo de firma digital se debe guardar dentro de la carpeta "Assets/Plugins/Android/assets" que está en la jerarquía del proyecto de Unity, una vez que se tenga el archivo de firma digital en el proyecto, el compilador de Unity agregara la firma digital a la aplicación cada vez que se compile un ".apk" nuevo.

Para crear el ".apk" de la aplicación solo hay que crear un build con Unity. En la sección de tutoriales de Unity se explica de manera detallada como crear un build [14]para la plataforma de su preferencia, para este caso creara un build para la plataforma Android.

7.2.5 Ejecución y Puesta en Marcha de la Aplicación en las Gafas Samsung Gear VR. Una vez se haya instalado la aplicación en el celular solo hay que abrir la aplicación y cuando la aplicación lo indique hay que proceder a insertar el celular en las gafas de realidad virtual, tan pronto como el celular detecte que ha sido conectado a las gafas de realidad virtual la aplicación se iniciara, en caso de que la aplicación no tenga la firma digital correcta se cerrara automáticamente.

[14] Link de la pagina de Unity en el que explican como crear el instalador para aplicaciones Android: https://unity3d.com/es/learn/tutorials/projects/space-shooter-tutorial/building-game

Figura 23. Celular conectado a las gafas.

Para que la aplicación funcione el celular debe estar conectado a la misma red que el computador que está actuando como servidor.

7.2.6 Errores Conocidos de la Aplicación LiveScanVR. Debido a que la aplicación LiveScanVR es solo un prototipo para demostrar la viabilidad de la reconstrucción 3D de manera remota puede que la aplicación tenga algunos errores a la hora de funcionar, es posible que más errores aparezcan con el tiempo a medida que los usuarios utilicen la aplicación, pero en la actualidad los únicos errores conocidos son los siguientes:

- Desconexión por suspensión de la aplicación: las gafas Samsung Gear VR pueden detectar cuando el usuario tiene las gafas puestas o no y debido a que las aplicaciones de realidad virtual consumen demasiada batería del celular todas las aplicaciones de realidad virtual del celular entran en modo suspensión cuando el usuario no tiene las gafas puestas, por lo tanto la aplicación LiveScanVR pierde la conexión con el servidor una vez que el celular entra en modo suspensión pero cuando el celular vuelve a activar la aplicación la conexión con el

servidor no se reestablece, esto debido a que la conexión con el servidor se hace en el evento de inicio de la aplicación y en ninguna otra parte del código se hace la conexión, por eso una vez que un usuario se ha quitado las gafas es necesario volver a iniciar la aplicación para que haga la conexión con el servidor nuevamente y funcione correctamente.

8. Fase 3. Requerimientos Mínimos para el Óptimo Funcionamiento del Sistema y la Aplicación Live Scan VR

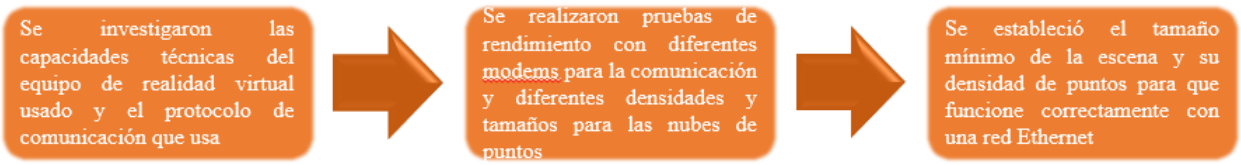

Figura 24. Actividades realizadas durante la fase 3.

El sistema Live Scan 3D es bastante costoso computacionalmente debido a que el algoritmo de renderizado de la nube de puntos que usa no es muy eficiente, pero cuenta con opciones de configuración que nos permite optimizar un poco su funcionamiento dependiendo de las necesidades y de las características del computador que estamos usando para renderizar la nube de puntos. En esta sección explicaremos los parámetros que se usaron en el desarrollo del presente proyecto los cuales ayudan a optimizar el rendimiento de la aplicación de realidad virtual.

8.1. Especificaciones Mínimas para un Computador que Actúa como Cliente

El computador que actúa como cliente no requiere mucha potencia computacional, pero si necesita una tarjeta de red que pueda enviar los datos de la nube de puntos a una velocidad aceptable para que el servidor reciba la información de manera oportuna.

Especificaciones Mínimas del Cliente

- Tarjeta Gráfica: 2GB de memoria DDR6
- Procesador: Intel® Core™ i5
- Memoria RAM: 8 GB
- Sistema Operativo: Windows 10 (64bit)
- Tarjeta de red con estándar 802.11 N

8.2. Especificaciones Mínimas para un Computador que Actúa como Servidor

Se recomienda que el computador que alojara el servidor de la aplicación sea el que tenga la tarjeta gráfica más potente y al menos 16 GB de memoria RAM ya que este computador se encargara de procesar y renderizar la nube de puntos en la ventana de visualización.

Especificaciones mínimas del servidor:

- Tarjeta Gráfica: 4GB de memoria DDR6
- Procesador: Intel® Core™ i7

- Memoria RAM: 16 GB
- Sistema Operativo: Windows 10 (64bit)
- Tarjeta de red con estándar 802.11 N

8.3. Requerimientos Mínimos de Red

Debido a la gran cantidad de información que el sensor Kinect V2 transmite es necesario contar con una red de alta velocidad que pueda transmitir esa gran cantidad de información de manera rápida y para mayor eficiencia del sistema se recomienda que todos los computadores que forman parte del sistema estén conectados mediante la misma tecnología de red para que la recepción de paquetes en el servidor sea eficiente y así evitar la latencia alta a la hora de renderizar cada frame en el servidor.

El sensor Kinect V2 se conecta al computador a través de un puerto USB 3.0 y no funciona cuando se conecta a un puerto USB 2.0 esto es debido al tamaño de los datos que transmite el sensor Kinect V2 es muy grande y no puede ser enviado a la velocidad del protocolo USB 2.0 (480 Mbps) mientras que el protocolo USB 3.0 permite la transferencia de datos a velocidades de hasta 4.8 Gbps, diez veces más que su antecesor. Es debido a esta velocidad de transferencia de datos tan alta que usa el sensor Kinect V2 para comunicarse con el computador, que la red también necesita un ancho de banda que permita poder transmitir datos a esta misma velocidad o superior, de esta manera se concluye que para un óptimo rendimiento del sistema se debería implementar una red LAN de fibra óptica ya que este soporta un ancho de banda de hasta 10 Gbps lo cual permitiría al servidor recibir los datos de los clientes de manera rápida y así la visualización de la escena sería mucho más fluida.

8.4. Requerimientos mínimos de una escena para mantener un flujo estable de datos

Debido a las limitaciones técnicas del Samsung Gear VR este no puede renderizar una escena de gran tamaño por esto es necesario establecer unas condiciones ideales en la escena para que el Samsung Gear VR pueda renderizar la escena a una velocidad aceptable en la que se evidencie el cambio constante de forma en la escena, para esto se usa la ventana settings en la que se configura el tamaño de la escena que se visualizara en la aplicación de realidad virtual y la densidad de puntos que se usara para renderizar la escena.

Lo primero que se debe entender es que para que la aplicación renderice más rápido la escena lo que se necesita es que la nube de puntos no contenga demasiados puntos, entre menor sea la cantidad de puntos a renderizar más rápido terminara la aplicación de renderizar cada frame, para lograr una nube de puntos más pequeña y con suficientes puntos para poder dar forma a los objetos presentes en la escena solo deben tenerse en cuenta dos factores dentro de los settings del proyecto Live Scan 3D, el tamaño del cubo en el que se deben encontrar los objetos de la escena a renderizar y el filtro que se aplican a los puntos para determinar la densidad de puntos.

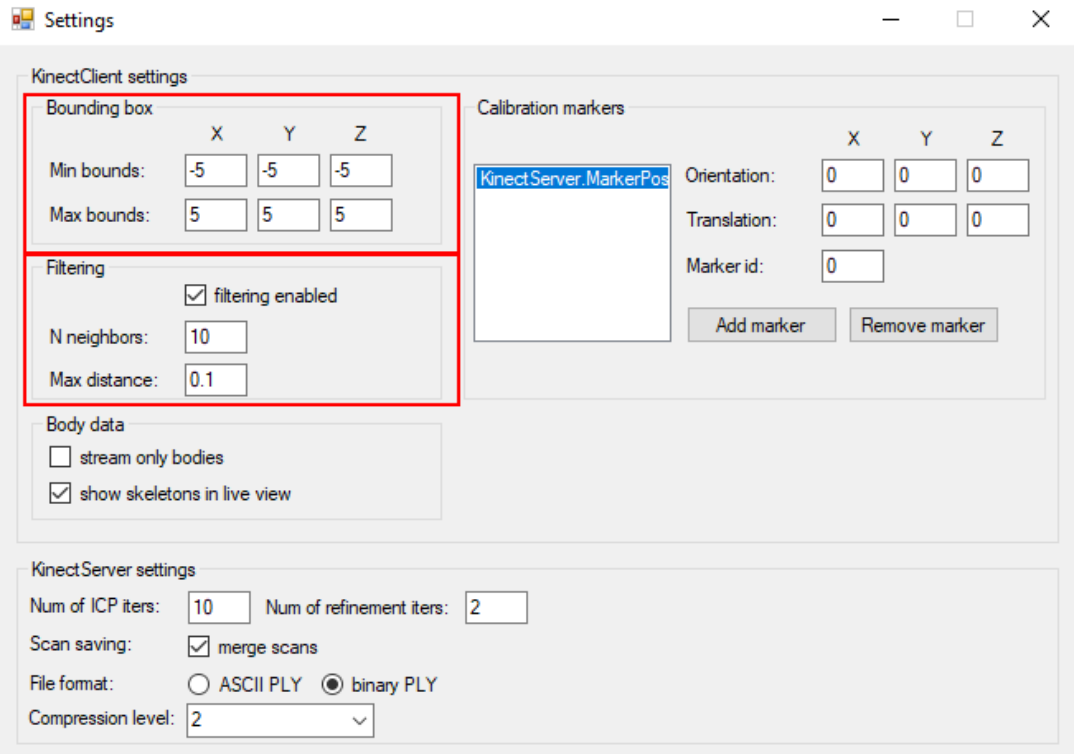

Figura 25. Opciones de configuración de la escena

Antes de comenzar a establecer el tamaño mínimo de la escena se debe tener en cuenta que las unidades que usa Live Scan 3D están dadas en metros por lo que en la configuración que trae Live Scan 3D por defecto se está visualizando una escena de diez metros cúbicos, ya que al establecer las longitudes en el eje negativo y positivo se toma dos veces la longitud que coloquemos en los settings. El tamaño del cubo de visualización de la escena se establece de esta manera ya que Live Scan 3D toma como el origen del sistema de coordenadas el punto central en el que se encuentra el marcador con el que fue calibrado el sistema y es a partir de ese origen de coordenadas que se crea el cubo de visualización usando los parámetros que establezcamos en la tabla de settings, así que para establecer el tamaño correcto del cubo que necesitamos se debe dividir la longitud del lado del cubo entre dos y ese valor agregarlo al lado negativo y positivo de cada eje.

Como se mencionó antes lo que se busca es tener una nube de puntos pequeña, por lo que se establecerá un cubo de medio metro para la visualización de la escena, así que en los límites del cubo colocaremos valores de -0.25 y 0.25 metros en las casillas de cada eje.

Ahora para reducir aún más la cantidad de puntos que se verán dentro del cubo de visualización, será necesario establecer unos valores de filtro de puntos adecuados para que no se tomen puntos que se encuentren demasiado cercanos, de esta manera solo se envían a la aplicación de realidad virtual los puntos suficientes para que la forma de los objetos se vea definida.

En esta parte al igual que en la anterior, la unidad de distancia usada es el metro y el número de vecinos se refiere a cuantos puntos como máximo se mostraran alrededor de cada punto que se va a mostrar en la escena, los valores que trae el Live Scan 3D muestran un punto por cada centímetro ya que busca diez puntos cercanos en una distancia de diez centímetros lo cual proporciona suficientes puntos para renderizar una forma bastante definida de los objetos, pero está no es la única forma en la que se puede darle definición a los objetos, para beneficiar la velocidad de renderizado de la escena en la aplicación de realidad virtual vamos a bajar la densidad de puntos a dos puntos por cada cinco centímetros y usando el script Point Cloud Renderer del proyecto Live Scan VR debemos aumentar el tamaño del punto que se renderiza en la escena para poder los objetos con una superficie más completa, de esta manera tendremos una buena definición usando pocos puntos.

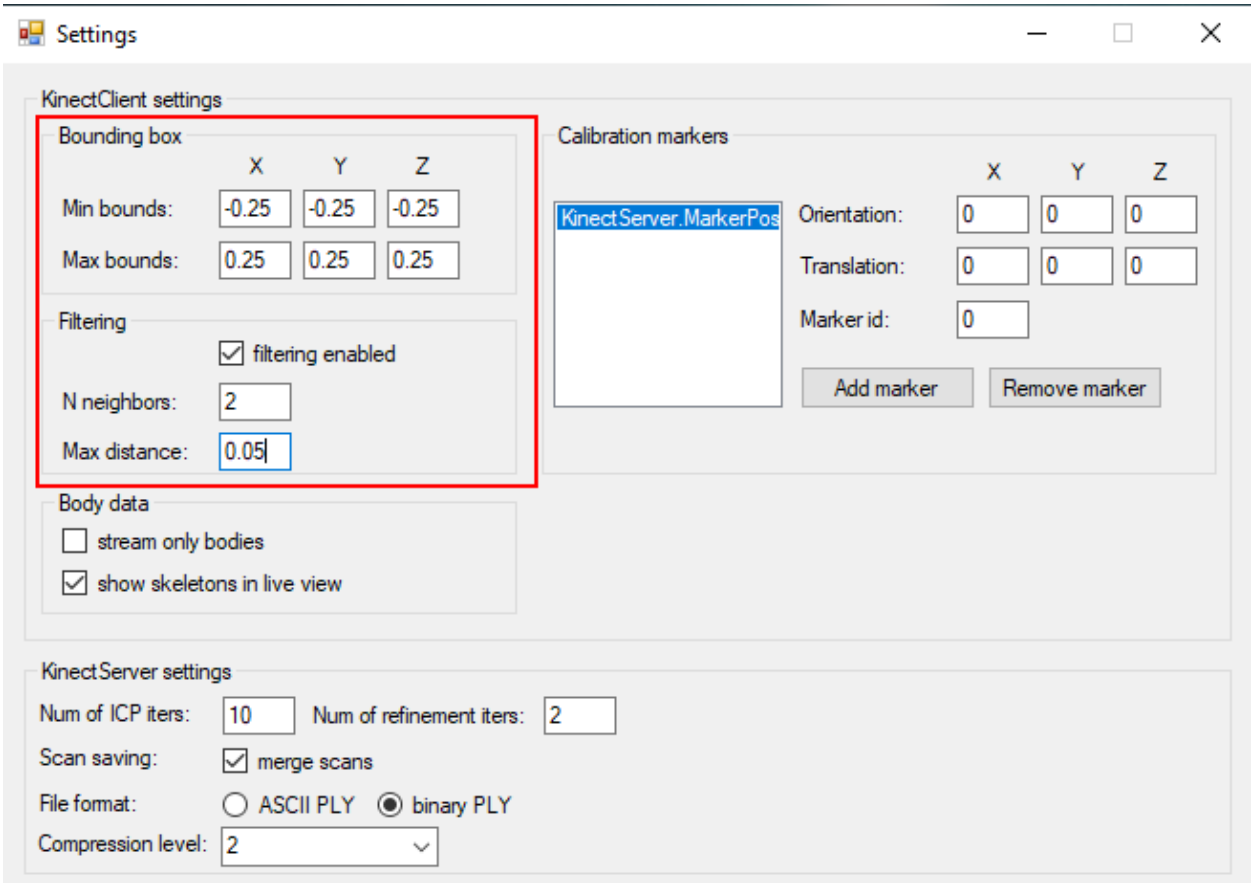

Figura 26. Parámetros finales para un flujo estable de datos.

8.5. Latencia en los clientes de Live Scan 3D

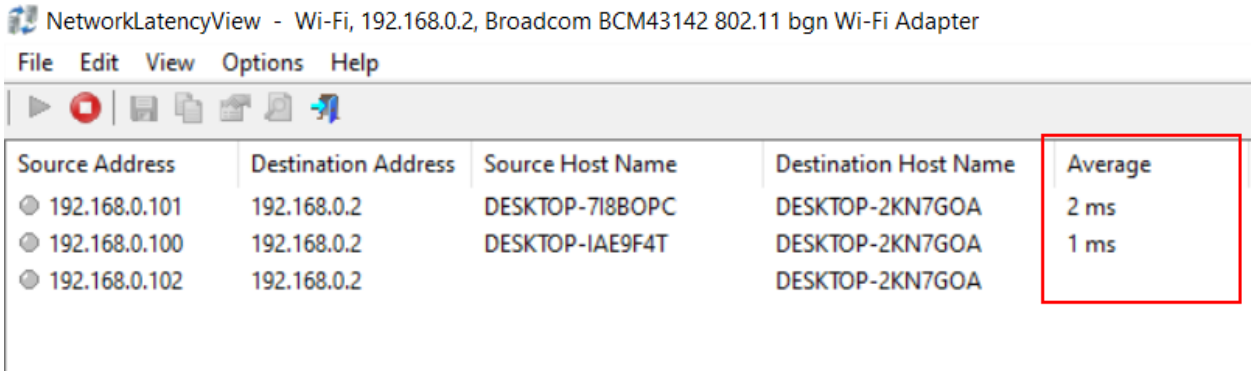

Figura 27. Latencia medida de los clientes de Live Scan 3D

Los resultados de usar una configuración especifican para este proyecto y computadoras que cumplen los requerimientos mínimos para el correcto funcionamiento del proyecto Live Scan 3D, es la baja latencia observada en cada uno de los clientes que envían la información al servidor de Live Scan 3D para ser reconstruida.

9. Fase 4. Evaluación de la Exactitud de la Reconstrucción del Proyecto Live Scan 3D

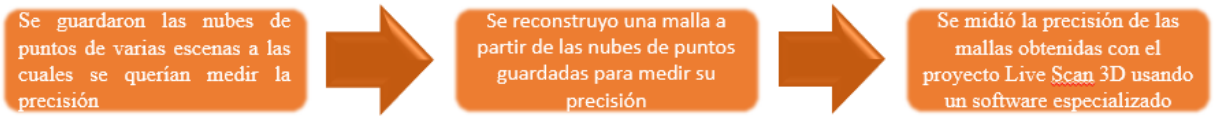

Figura 28. Actividades realizadas durante la fase 4.

Para evaluar la exactitud con la que los sensores Kinect V2 reconstruyen la escena primero se debe convertir la nube de puntos obtenida por los sensores en una malla poligonal que será comparada con la malla poligonal obtenida mediante un método de reconstrucción poligonal del cual si conocemos su margen de error.

9.1. Calibración de Live Scan 3D

El proyecto Live Scan 3D usa un algoritmo de calibración que funciona con los marcadores predefinidos que vienen incluidos en el proyecto.

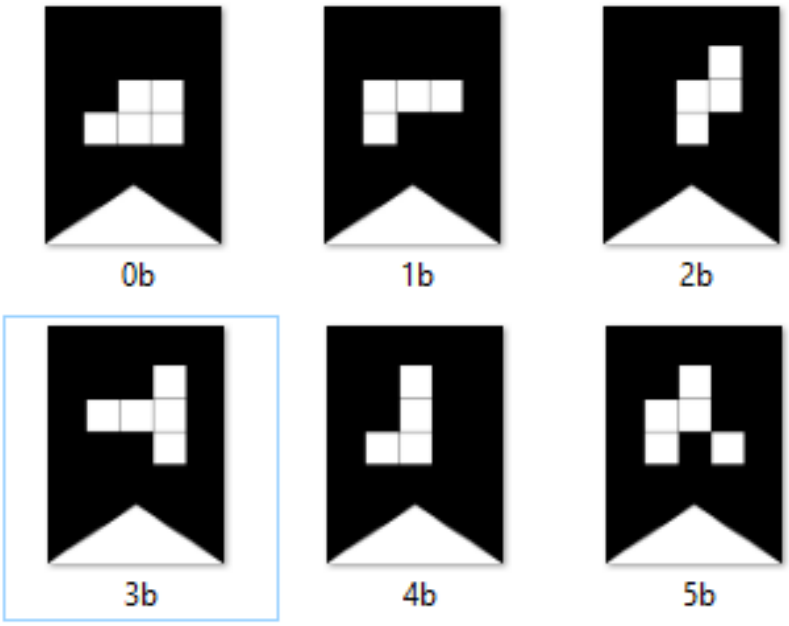

Figura 29. Marcadores de calibración del proyecto Live Scan 3D

Para calibrar los sensores se debe entrar en las opciones de calibración del servidor de Live Scan 3D ya que es el servidor quien calibrara los sensores en cada uno de los clientes hay que agregar en las opciones de calibración los marcadores que se vayan a usar para la calibración. Para calibrar los sensores es necesario que los marcadores que se estén usando se coloquen en posiciones en las que sean visibles para todos los sensores que estemos usando en el sistema por lo que dependiendo de la disposición espacial de los sensores que estemos usando tendremos que usar varios marcadores.

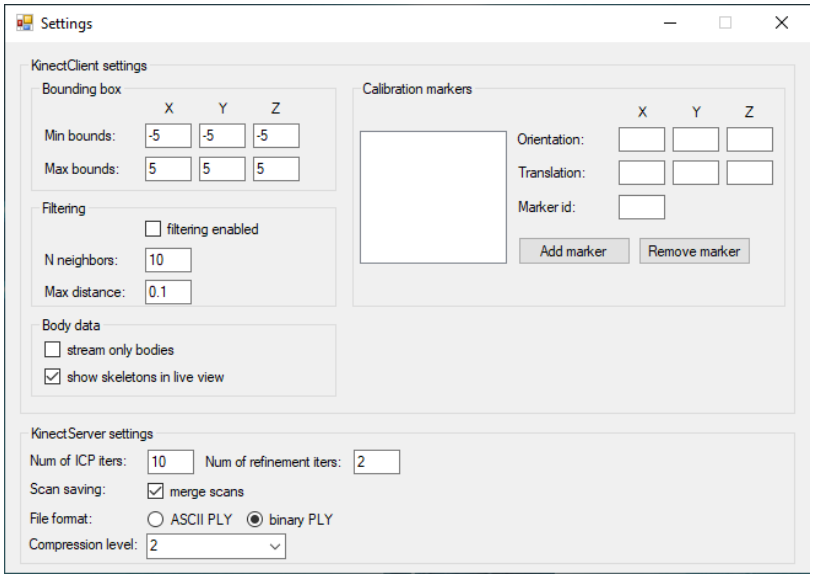

Figura 30. Opciones de calibración de Live Scan Server.

Una vez que tengamos la ventana de configuración abierta le daremos clic en agregar marcador.

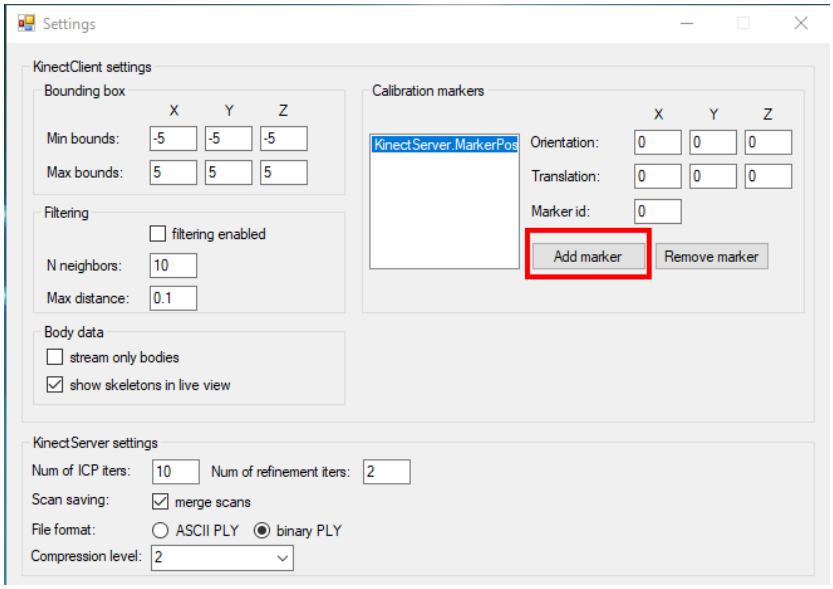

Figura 31. Marcadores usados para la calibración.

En la casilla “Marker id” hay que colocar el número correspondiente al marcador que usara para calibrar los sensores que vienen asignados a cada uno de ellos en la carpeta “calibration markers” por ejemplo, si se quiera usar el marcador “0b” para calibrar, en el “Marker id” pondremos 0.

En las opciones “orientation” y “translation” hay que dejar todos los valores en ceros para el primer marcador y en caso de usar más de un marcador hay que agregar la respectiva rotación (en grados) y traslación (en metros) que tengan los demás marcadores agregados con respecto del marcador que tenga posición y traslación en cero.

Una vez configurados los marcadores en la ventana “Settings” hay que iniciar el servidor y conectar los clientes al servidor.

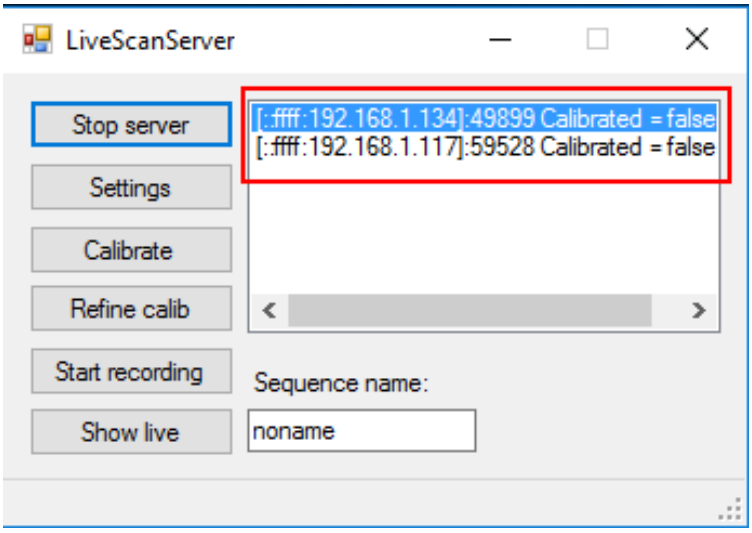

Figura 32. Clientes conectados al servidor sin calibrar.

Cuando ya estén conectados todos los clientes que se unirán al servidor hay que colocar los marcadores en una parte en que cada sensor pueda ver al menos uno de los marcadores, primero se debe colocar el marcador que tiene rotación y traslación cero ya que este marcador proporcionara el origen del plano de coordenadas que usara el servidor para colocar los puntos y los demás marcadores se colocan con la rotación y traslación con que fueron definidas con anterioridad en la ventana “Settings” con respecto al marcador que actuara como el origen del plano.

Después de que los marcadores ya se encuentren en una posición visible para cada sensor hay que dar clic en el botón calibrar de la ventana de Live Scan Server.

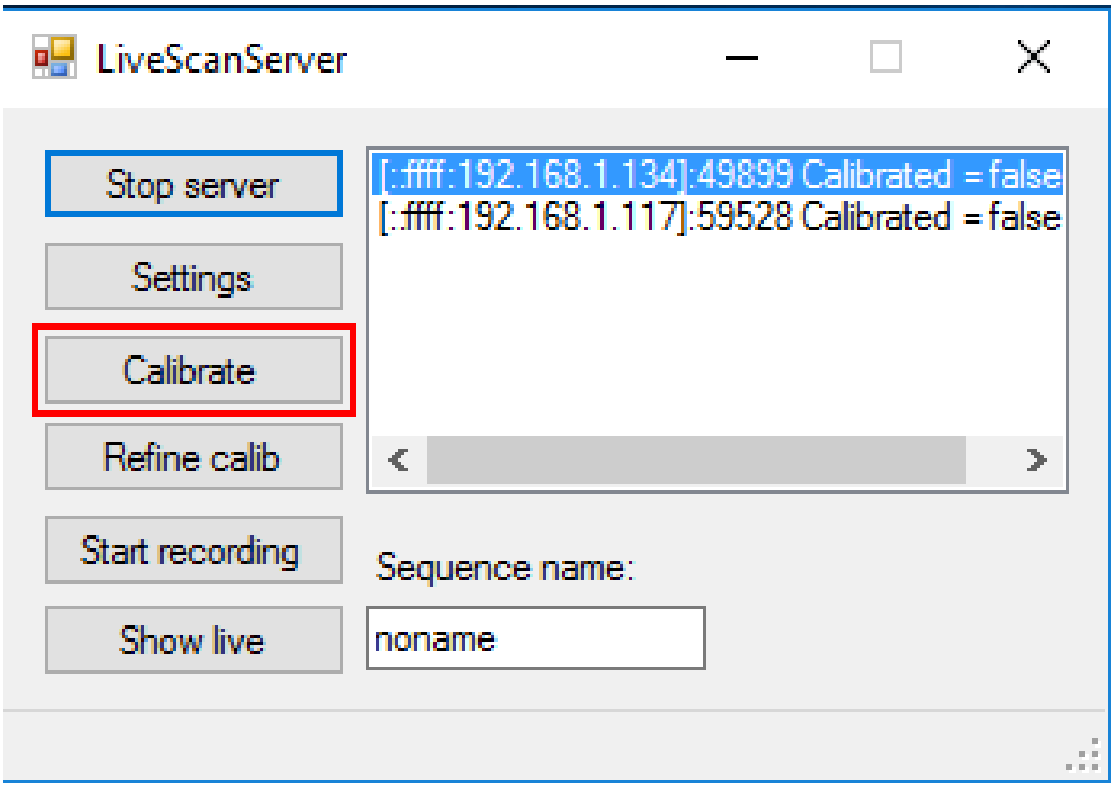

Figura 33. Botón calibrar.

Como se puede apreciar en la figura 32. Cuando los clientes se conectan al servidor este muestra en qué estado de calibración se encuentra cada cliente, en caso de que el sensor no esté calibrado en su respectivo cliente mostrara “false” en la bandera de calibración.

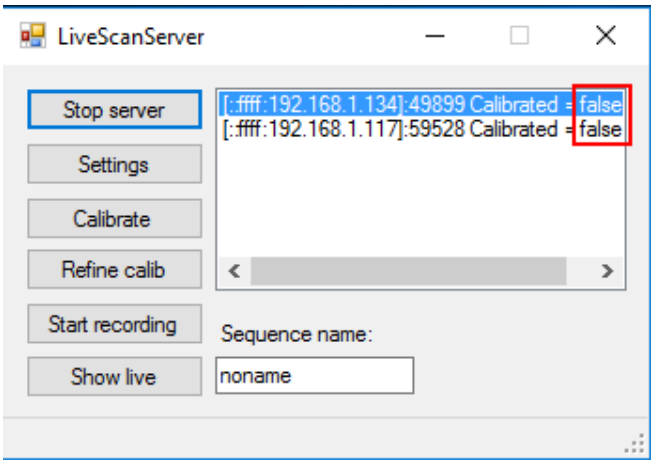

Figura 34. Clientes con sensores sin calibrar.

Una vez termine la calibración de los sensores la bandera de calibración de cada cliente deberá aparecer como “true”.

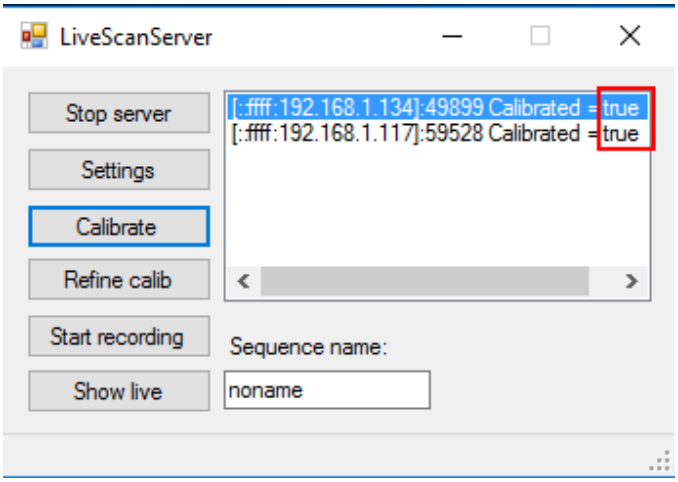

Figura 35. Clientes con sensores calibrados.

Cuando todos los sensores queden correctamente calibrados se debería poder ver que la nube de puntos proveniente de cada uno de los sensores concuerda en la ventana de “ShowLive” en la que se muestra la suma de todas las nubes de puntos que está recibiendo el servidor en ese momento.

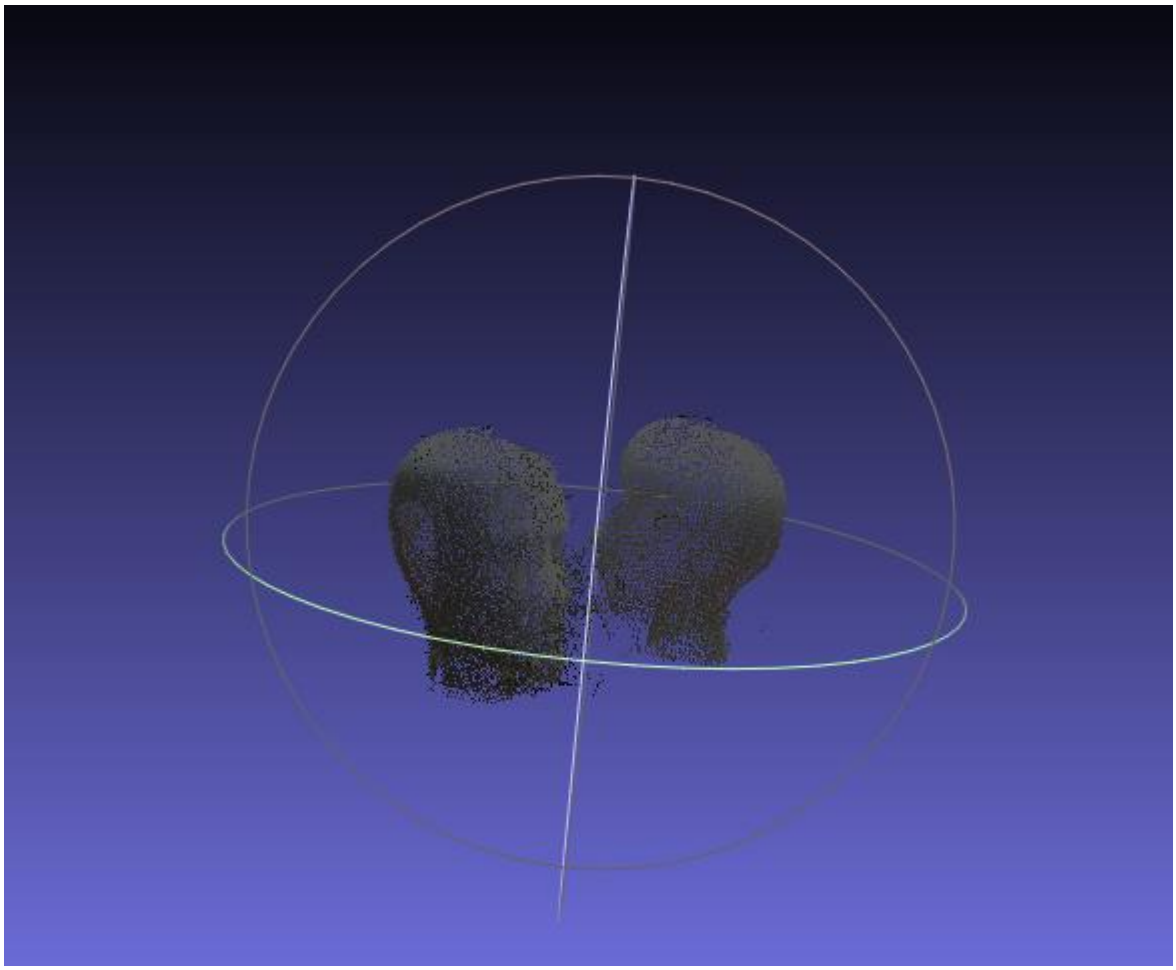

Figura 36. Nube de puntos obtenida de todos los sensores calibrados.

9.2.Guardado de la Nube de Puntos que Muestra el Servidor.

Para obtener una malla a partir de la nube de puntos que nos muestra Live Scan Server primero se debe guardar la nube de puntos en un formato en el cual se pueda trabajar sobre la información obtenida, para esto se usara la función de grabación que trae Live Scan Server.

Lo primero que hay que hacer es darle un nombre a la carpeta donde se guardaran los archivos de las nubes de puntos que va mostrando el servidor en cada “fps” para esto hay que dar clic en la casilla “Sequence name” y colocar el nombre de la carpeta donde se guardaran los archivos.

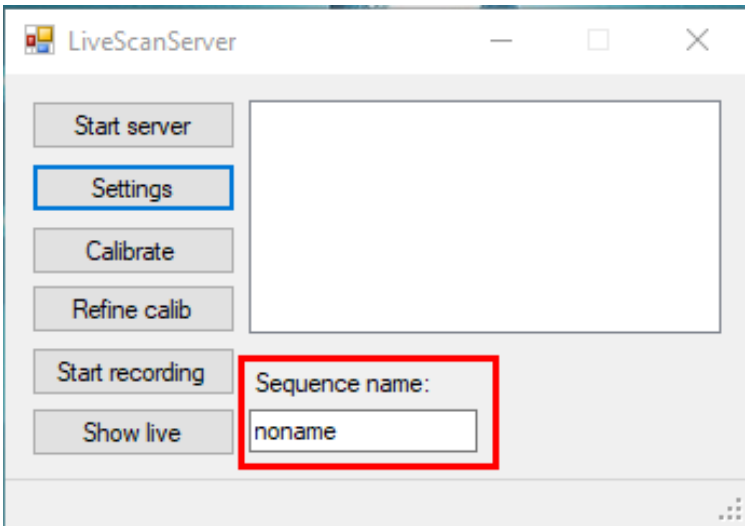

Figura 37. Casilla “Sequence name”.

Una vez que ya esté el nombre puesto en la casilla de “Sequence name” hay que dar clic en el botón “Start recording” para comenzar a grabar cada frame que el servidor renderice.

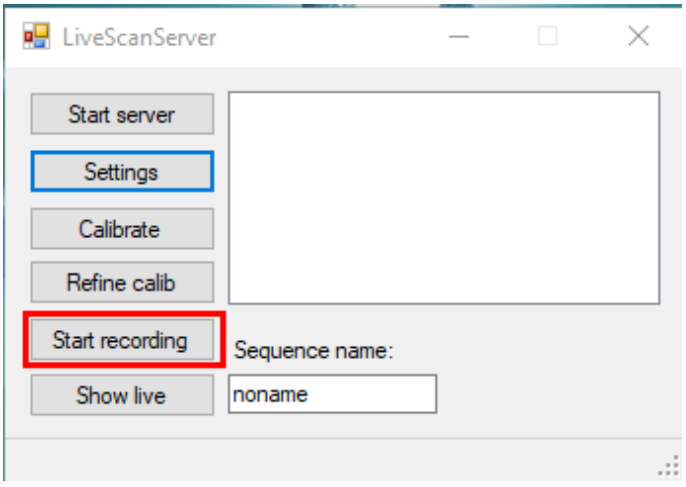

Figura 38. Botón “Start recording”.

Cuando se de clic en este botón se comenzaran a almacenar las nubes de puntos mostradas en cada frame en la memoria del computador por lo cual es necesario que el computador que esté actuando como servidor cuente con suficiente memoria RAM para almacenar varios frames de la nube de puntos sin llegar a colapsar por falta de memoria, aun así para obtener una malla de la escena solo se necesita un archivo de la nube de puntos.

En cuanto se de clic en el botón “Start recording” dicho botón se convertirá en “Stop recording”.

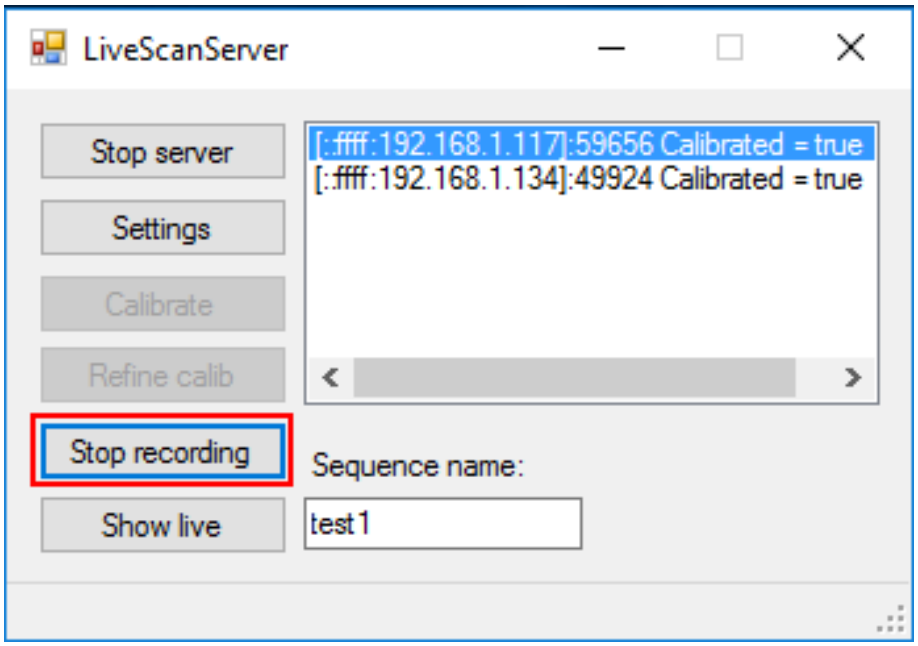

Figura 39. Botón “Stop recording”.

Si se da clic en el botón “Stop recording” el servidor dejara de almacenar la información de las nubes de puntos en la memoria del computador y comenzara a guardarla en archivos en el disco duro, el servidor guardara estos archivos dentro de la carpeta “out” que se crea por defecto en la carpeta en la que se ejecuto el servidor y dentro de esa carpeta “out” se crea una carpeta con el

nombre que se encuentre puesto en la casilla “Sequence name” y comenzará a enumerar los archivos de manera ascendente partiendo de “0001”.

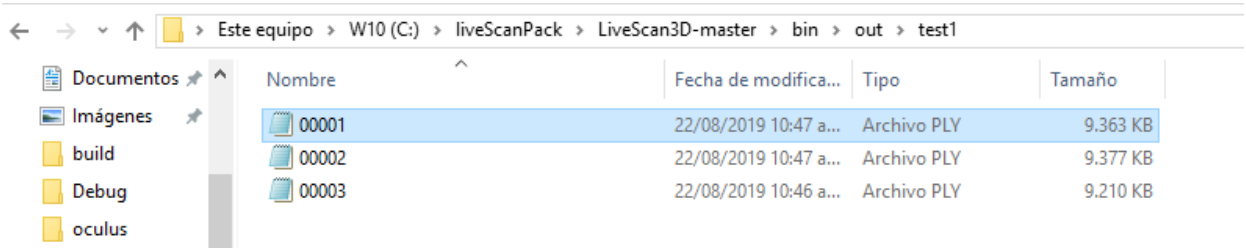

Figura 40. Archivos de nube de puntos guardados por el servidor.

El formato de los archivos es .PLY y en la ventana “Settings” se puede definir si el archivo .PLY se guardara en formato ASCII o en binario.

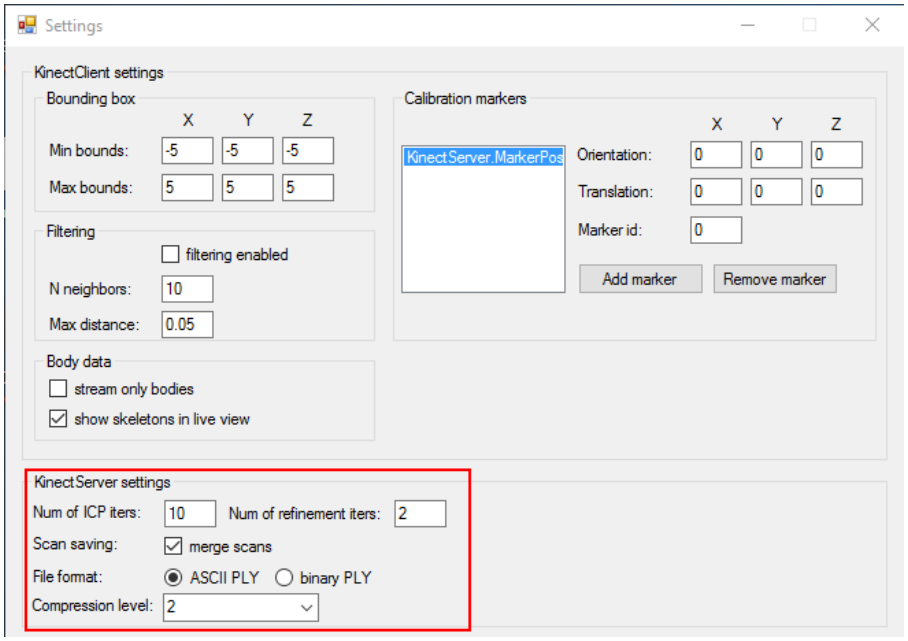

Figura 41. Sección de selección del formato .PLY.

9.3.Obtención de la Malla a Partir de la Nube de Puntos

Para la medición de la exactitud y la precisión del proyecto se tomaron cincuenta reconstrucciones de treinta frames consecutivos de una escena específica, y usando el programa MeshLab[15] se promediaron las treinta nubes de puntos obtenidas, para crear una nube de puntos que tuviera la información promedio de las treinta capturas realizadas.

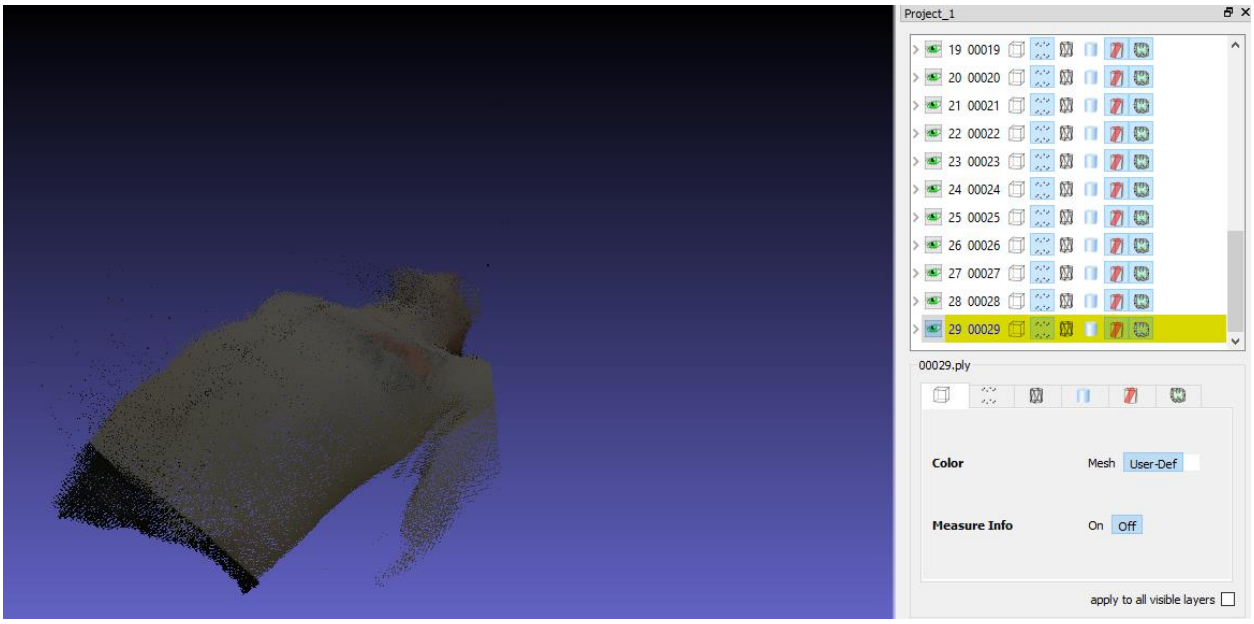

Figura 42. Muestra de treinta frames cargados en MeshLab

Una vez cargadas las treinta nubes de puntos, se unificaron todos los puntos en una solo nube de puntos y luego se procedió a promediar la cantidad de puntos para crear una nube que tuviera información sobre como cambia la forma captada por los sensores Kinect a lo largo del tiempo.

[15] MeshLab es un software para el análisis y tratamiento de archivos mallas tridimensionales, que cuenta con diversos algoritmos matemáticos disponibles para procesar y modificar mallas.

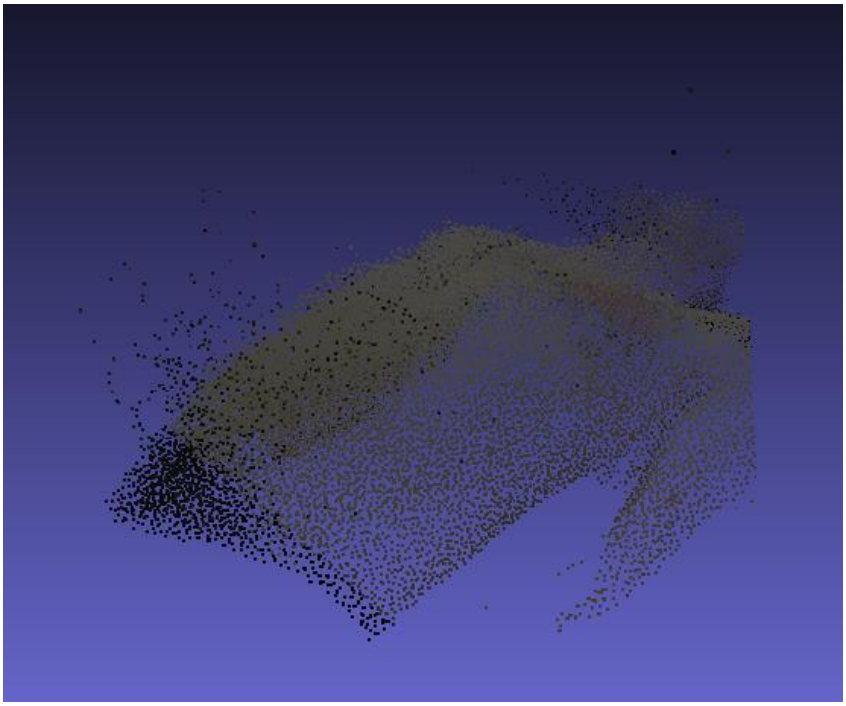

Figura 43. Nube de puntos promedio.

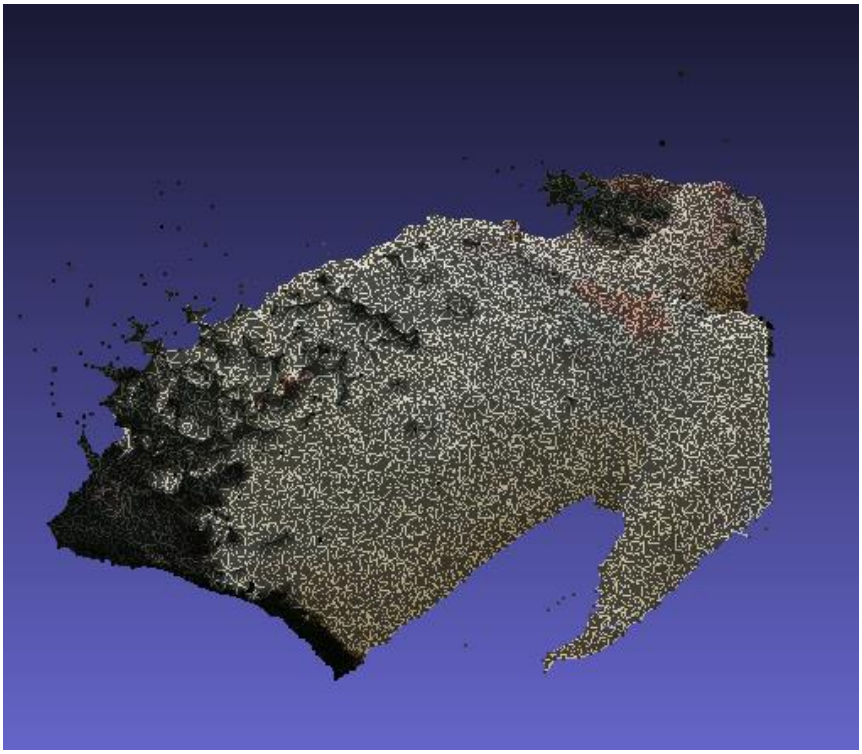

Figura 44. Malla obtenida a partir de la nube de puntos mediante el algoritmo Ball Pivoting.

9.4. Medición de la Precisión y Exactitud de la Malla

Usando el software Cloud Compare,[16] se registró la distancia media entre cada de las mallas y la malla de referencia obtenida mediante el escáner laser GoScan[17]

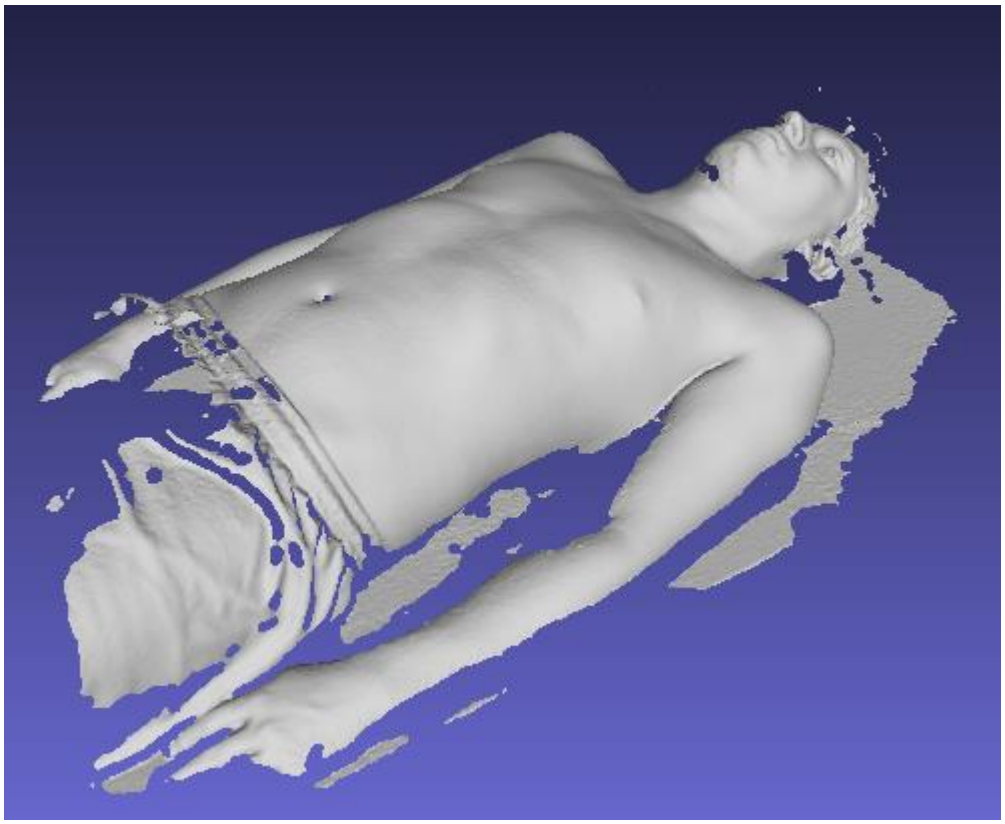

Figura 45. Malla de referencia, obtenida mediante el GoScan.

[16] CloudCompare es un software de procesamiento de nube de puntos 3D (y malla triangular). Originalmente se diseñó para realizar una comparación entre dos nubes de puntos 3D (como las adquiridas con un escáner láser) o entre una nube de puntos y una malla triangular. Se basa en una estructura de octree específica dedicada a esta tarea.
[17] GoScan es un scaner laser de alta resolución y nivel de detalle.

Usando la malla de alta resolución obtenida por el GoScan como malla de referencia, se realizaron las comparaciones con cada una de las cincuenta reconstrucciones obtenidas por el proyecto LiveScan 3D.

Figura 46. *Resultado de la comparación hecha con el software CloudCompare.*

En la figura 45 se puede apreciar el resultado de comparar la malla de referencia con la malla reconstruida del proyecto Live Scan 3D (vistas en el panel que se encuentra en la parte superior izquierda), el programa muestra los resultados de la medición hecha entre las dos mallas los cuales se pueden ver en la parte inferior izquierda de la figura 45, en donde se muestra la distancia media que hay entre las caras de ambas mallas y como se ve en los resultados de la figura 45 es bastante alta, seguida de la desviación estándar para el muestreo realizado entre las dos mallas, por último, a la derecha de la figura 45 podemos ver el histograma de colores que nos muestra la distancia que hay en las caras de la malla que se tomó como referencia para la comparación entre las dos mallas,

como se puede apreciar en la figura 45, el total de los polígonos de la malla de referencia se encuentra a una distancia bastante reducida.

Tabla 2.

Resultados de las comparaciones realizadas con CloudCompare.

No. MALLA	MEAN D	STN DEV.	MIN DIST.	MAX DIST.	AVR DIST.	SIGMA	MAX ERROR
1	0.004995	0.029095	0	0.443596	0.0199816	0.0184409	0.0038616
2	0.004264	0.029800	0	0.434219	0.0207611	0.0182151	0.0038616
3	0.008664	0.029319	0	0.445006	0.0211139	0.0183795	0.0038616
4	0.010557	0.028821	0	0.366017	0.0213506	0.0183293	0.0038616
5	0.013028	0.029051	0	0.308251	0.0225082	0.0183911	0.0038616
6	0.011687	0.030357	0	0.442721	0.0225631	0.0197467	0.0038616
7	0.011004	0.028495	0	0.364957	0.0210431	0.0183008	0.0038616
8	0.010763	0.028064	0	0.386854	0.029853	0.0177329	0.0038616
9	0.009949	0.029400	0	0.444805	0.0210657	0.01932	0.0038616
10	0.005944	0.022835	0	0.101583	0.0158352	0.0141476	0.0038616
11	0.009480	0.028012	0	0.44861	0.0199781	0.0182512	0.0038616
12	0.009591	0.027916	0	0.453043	0.0202349	0.0179337	0.0038616
13	0.010817	0.030044	0	0.452252	0.022258	0.0192058	0.0038616
14	0.008204	0.028503	0	0.454751	0.020158	0.0183611	0.0038616
15	0.009051	0.028317	0	0.408746	0.0203635	0.0179105	0.0038616
16	0.011773	0.027587	0	0.17096	0.0211197	0.0173838	0.0038616
17	0.011387	0.029683	0	0.452252	0.0219587	0.0190497	0.0038616
18	0.011306	0.028877	0	0.36618	0.0217766	0.0180699	0.0038616
19	0.010041	0.027992	0	0.304062	0.0206803	0.0174157	0.0038616
20	0.010911	0.027532	0	0.452252	0.0203712	0.017861	0.0038616
21	0.008281	0.028337	0	0.44029	0.0201824	0.0179147	0.0038616
22	0.009910	0.028220	0	0.456975	0.0202627	0.184939	0.0038616
23	0.010106	0.027690	0	0.407284	0.0198811	0.018158	0.0038616
24	0.008764	0.028245	0	0.462232	0.0200727	0.0182136	0.0038616
25	0.007984	0.028179	0	0.429246	0.0196433	0.0182311	0.0038616
26	0.011320	0.028068	0	0.447479	0.0200953	0.0192846	0.0038616
27	0.010668	0.028134	0	0.439137	0.020091	0.0188688	0.0038616
28	0.010775	0.028589	0	0.452582	0.0211332	0.0183813	0.0038616
29	0.013225	0.29129	0	0.417695	0.0225082	0.0187169	0.0038616
30	0.012872	0.028613	0	0.43934	0.0218245	0.0188096	0.0038616
31	0.012171	0.028335	0	0.0442182	0.0214873	0.0181714	0.0038616
32	0.011001	0.028974	0	0.44198	0.0213306	0.0185098	0.0038616

Continuación de la Tabla 2.

No. MALLA	MEAN D	STN DEV.	MIN DIST.	MAX DIST.	AVR DIST.	SIGMA	MAX ERROR
33	0.009925	0.027013	0	0.43934	0.0199374	0.0170876	0.0038616
34	0.009220	0.027818	0	0.437163	0.0200803	0.0177813	0.0038616
35	0.009554	0.027833	0	0.389007	0.0200294	0.0179462	0.0038616
36	0.010319	0.027130	0	0.30061	0.0196728	0.0177151	0.0038616
37	0.008584	0.027358	0	0.387778	0.0192552	0.017789	0.0038616
38	0.009002	0.027505	0	0.388162	0.019512	0.0178806	0.0038616
39	0.009365	0.028084	0	0.38847	0.0201557	0.0181635	0.0038616
40	0.009429	0.028493	0	0.437504	0.02044	0.0183983	0.0038616
41	0.011503	0.028671	0	0.445341	0.0216633	0.0182209	0.0038616
42	0.010871	0.028070	0	0.446211	0.0205661	0.0184301	0.0038616
43	0.010447	0.028467	0	0.455341	0.020849	0.0183676	0.0038616
44	0.007855	0.028788	0	0.438117	0.0202108	0.0186221	0.0038616
45	0.007774	0.027012	0	0.33567	0.0193485	0.0169923	0.0038616
46	0.010638	0.027688	0	0.445408	0.0198169	0.0188879	0.0038616
47	0.009452	0.027432	0	0.401457	0.019385	0.0182212	0.0038616
48	0.008819	0.027868	0	0.432292	0.0199704	0.0180535	0.0038616
49	0.008336	0.028657	0	0.462038	0.0194263	0.0197009	0.0038616
50	0.007935	0.028018	0	0.450931	0.0190497	0.0190523	0.0038616

Con los datos obtenidos en la tabla 2. Se procedió a hacer el análisis estadístico de la reconstrucción hecha por el proyecto Live Scan 3D.

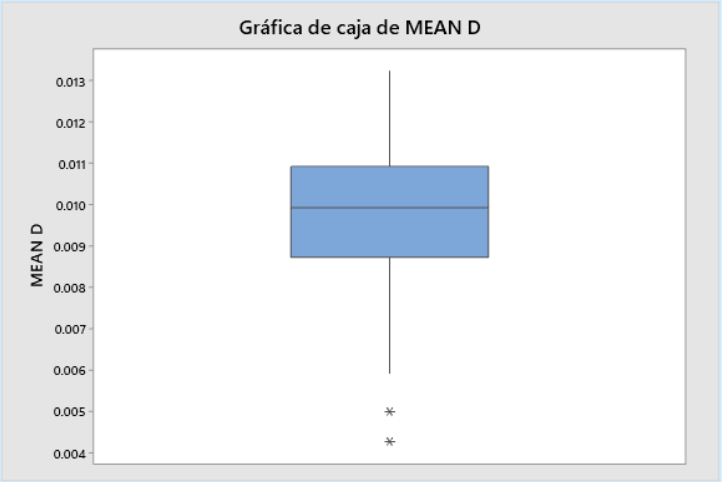

Figura 47. Caja de bigotes para los datos de MEAN DISTANCE.

Usando los datos de mean distance se determinó que la precisión de la reconstrucción hecha por el proyecto Live Scan 3D es de:

$$\pm 0.001824$$

En el artículo "Comparison of Kinect V1 and V2 Depth Images in Terms of Accuracy and Precision" se analiza la precisión y exactitud del sensor de profundidad del Kinect V2 y nos dice que su error máximo es de 0.0025 metros asumiendo este como el error teórico aceptable para hallar la exactitud, tenemos que la exactitud del proyecto Live Scan 3D es:

$$E_{det} = 0.00979 - 0.0025 = 0.00729$$

$$\%\ de\ E_{det} \frac{0.009790 - 0.0025}{0.0025} \times 100 = 291.6$$

Lo cual es un error porcentual alto, por lo cual se puede inferir, que el proyecto Live Scan 3D realiza reconstrucciones precisas, pero no muy exactas.

9.5.Posibles Causantes de Error en la Medición de los Datos

Dentro de las posibles causas de error a la hora de hacer las mediciones entre las mallas, sin duda la que mayor error introduce a la toma de los datos es la diferencia de tamaño que hay entre la malla de referencia y las mallas a evaluar. Para la toma de mediciones se hizo necesario escalar la malla de referencia para que tuviera un tamaño parecido al de las mallas a evaluar, dicho proceso se llevó a cabo de manera manual usando la herramienta Cloud Compare, dado que, es extremadamente difícil hacer coincidir el tamaño de las mallas usando el mouse del computador y

el ojo humano se generaba una diferencia de tamaño entre las mallas que hacía que la distancia entre ellas fuera más grande de lo que sería originalmente.

El ruido guardado en las nubes de puntos que se guardaron del proyecto Live Scan 3D contribuyo a que el algoritmo de reconstrucción de malla generara una malla con una forma diferente en ciertas zonas con demasiado ruido, dado que no se tiene un algoritmo específico para limpiar el ruido de las mallas antes de reconstruirlas, esta diferencia de formas entre las mallas a comparar y la malla de referencia pudo aumentar aún más la distancia media que se calculaba entre ambas mallas.

El espacio reconstruido también pudo haber generado que el error de las mediciones creciera, dado que el GoScan se maneja con la mano y reconstruye la forma de lo que entra en su campo de visión, es más difícil hacer una reconstrucción de una zona delimitada, ya que también entra en juego el error con el que la persona encargada de hacer la reconstrucción con el GoScan escanea la escena específica, mientras que los Kinects al estar montados sobre bases fijas siempre reconstruyen la misma escena a menos que se muevan de sus posiciones.

10. Conclusiones

Según el análisis estadístico realizado se considera que el proyecto es factible técnicamente en términos de precisión y latencia, pero no de exactitud.

Usando conexiones de fibra óptica para comunicar la aplicación de realidad virtual con el servidor de Live Scan 3D a través de internet será posible la visualización de manera remota.

El uso de dispositivos de realidad virtual con mucha más capacidad de procesamiento gráfico permitiría la reconstrucción de escena mucho más grandes y con una tasa de actualización de frames mucho más alta.

La reconstrucción observada en la escena de realidad virtual a través de las gafas Samsung Gear VR es bastante precisa, aunque no muy exacta, esto debido tal vez a la baja resolución de los sensores Kinect, ya que estos no fueron hechos expresamente para hacer reconstrucciones 3D realistas.

11. Recomendaciones

El presente proyecto es susceptible de mejorarse en cuanto al hardware usado se refiere y debido a que el software usado para su desarrollo es de código libre y tenemos acceso a su código fuente también es posible realizar mejoras a nivel de software. A continuación, se enunciarán algunos aspectos del funcionamiento del sistema desarrollado que deberán ser tenidos en cuenta a la hora de querer realizar mejoras en el rendimiento del mismo:

Como se enuncio en el apartado 7.3 el crear una red adecuada, única y exclusivamente para la transmisión de datos del sistema mejoraría en cierta medida el rendimiento del mismo.

Debido a que los mejores algoritmos de reconstrucción y renderizado en 3D son privativos el proyecto Live Scan 3D usa un algoritmo de renderizado funcional pero no tan eficiente como para funcionar en dispositivos móviles. Sustituir el algoritmo de renderizado en el proyecto Live Scan 3D contribuiría en gran medida a que el renderizado de la escena 3D en el visor de realidad virtual fuera más fluido.

Aunque el SDK de Kinect solo funciona en sistemas operativos de Windows, hoy en día es posible instalar cualquier sistema operativo en microcomputadoras que son bastante económicas, usando una microcomputadora que tenga un puerto USB 3.0 y un sistema operativo Windows, es posible reducir costos usando las microcomputadoras como clientes del sistema.

Usando el código fuente del proyecto Live Scan VR es posible crear un build de la aplicación que funcione en diferentes plataformas de realidad virtual, si se crea un build para una plataforma

como Oculus Rift o Valve HTC es posible que se vea una mejora en el rendimiento de la aplicación ya que estos visores de realidad virtual usan la tarjeta de video independiente del computador al que estén conectados ó simplemente se podría cambiar el celular que se está usando en conjunto con las gafas Samsung Gear VR por uno que tenga mayores prestaciones.

Referencias bibliográficas

3D, R. (s.f.). *Universidad Politecnica de Madrid*. Obtenido de http://www.elai.upm.es/webantigua/spain/Investiga/GCII/personal/lrodriguez/web3D/reconstruccion_3d.htm

Algoritmo. (2017). *Wikipedia*. Recuperado el 2019, de https://es.wikipedia.org/wiki/Algoritmo

Angela, D., Matthias Niebner, Michael Zollhöfer, Shahram Izadi, & Christian Theobalt. (2017). *Standfor Computer Graphics Laboratory*. Recuperado el 2019, de http://graphics.stanford.edu/projects/bundlefusion/

Format, P. (2018). *Wikipedia*. Recuperado el 2019, de https://en.wikipedia.org/wiki/PLY_(file_format)

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., . . . Fitzigbbon, A. (2011). *CiteSeer*. Recuperado el 2019, de http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.229.2346

Kowalski, M., & Naruniec, J. (2015). LiveScan3D: A Fast and Inexpensive 3D Data Acquisition System for Multiple Kinect v2 Sensors. *International Conference.* Lyon, France.

Microsoft. (2013). *KinectForDevelopers*. Recuperado el 2019, de http://www.kinectfordevelopers.com/es/2014/01/28/caracteristicas-kinect-2/

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., . . . Fitzgibbon, A. (2011). *Microsoft*. Recuperado el 2019, de https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=2ahUKEwiJrpOryIPjAhWLtlkKHR8gAJMQFjAAegQIBBAC&url=https%3A%2F%2Fwww.microsoft.com%2Fen-us%2Fresearch%2Fwp-

content%2Fuploads%2F2016%2F02%2Fismar2011.pdf&usg=AOvVaw3uHY0TJIr3p57KW4p52rtC

Nube de, P. (2017). *Wikipedia*. Recuperado el 2019, de https://es.wikipedia.org/wiki/Nube_de_puntos

Oculus. (2014). *GafasOculus*. Recuperado el 2019, de https://www.gafasoculus.com/gear-vr-samsung/

Poligonal, M. (2019). *Wikipedia*. Recuperado el 2019, de https://es.wikipedia.org/wiki/Malla_poligonal

Virtual, R. (2018). *Wikipedia*. Recuperado el 15 de 6 de 2019, de https://es.wikipedia.org/wiki/Realidad_virtual

Voxel. (2018). *Wikipedia*. Recuperado el 10 de 6 de 2019, de Wikipedia: https://es.wikipedia.org/wiki/V%C3%B3xel