Universidad Zaragoza
1542

Trabajo Fin de Grado

Reconstrucción 3D del entorno utilizando ORB-SLAM2 sobre un dispositivo Project Tango

3D Reconstruction of the environment using ORB-SLAM2 on a Project Tango device

Autor

Juan Luis Burillo Ortín

Director

Juan Domingo Tardós Solano

Escuela de Ingeniería y Arquitectura

2017

Resumen
Reconstrucción 3D del entorno utilizando ORB-SLAM2 sobre un dispositivo Project Tango

El proyecto realizado consiste en el desarrollo de una aplicación para el sistema operativo Android que permita la obtención de un modelo 3D del entorno utilizando técnicas de visión por computador. El dispositivo objetivo sobre el que se ha desarrollado la aplicación es una tableta Project Tango de Google. La peculiaridad de este dispositivo es que cuenta con una cámara RGB-D que permite obtener información de color y de profundidad.

Para poder crear la reconstrucción es necesario en primer lugar poder localizarse en el entorno. Para ello se ha utilizado la librería ORB-SLAM2 desarrollada en la universidad de Zaragoza, que permite construir un mapa del entorno y localizarse en el mismo de forma simultanea. La librería esta pensada para funcionar en un PC convencional, por lo que se ha tenido que realizar el portado a Android y el desarrollo de un nuevo visualizador gráfico. A continuación se desarrolló la aplicación de reconstrucción 3D, que utiliza las poses de la cámara calculadas por ORB-SLAM2 y las imágenes RGB-D para construir incrementalmente un modelo 3D denso del entorno.

Estas tareas requieren una gran capacidad de cálculo para funcionar en tiempo real. Al estar trabajando en un dispositivo móvil con capacidad de cálculo muy inferior a un PC convencional el rendimiento inicial de la aplicación no era el adecuado, por lo que se ha realizado un estudio del tiempo empleado por las funciones principales de la librería con el objetivo de encontrar los cuellos de botella y optimizarlos. Tras localizar las zonas más exigentes se han realizado optimizaciones basadas en la arquitectura de la CPU de la tableta.

Índice

1 Introducción y objetivos

1.1 Introducción

Las técnicas de *Simultaneous Localization and Mapping*(SLAM) consisten en utilizar un sensor para realizar la construcción del mapa de un entorno desconocido al mismo tiempo que se realiza el seguimiento de la trayectoria del sensor en dicho entorno.

Estas técnicas resultan especialmente útiles en los campos relacionados con la robótica, ya que permiten operar en un entorno sin tener previamente conocimiento sobre el mismo. Algunos ejemplos donde se pueden ver aplicaciones de técnicas de SLAM son los coches autónomos para realizar la navegación, drones para crear mapas, o robots para la inspección de terrenos desconocidos y potencialmente peligrosos, como por ejemplo una mina subterránea. Todos estos ejemplos están relacionados con la robótica pero también existen aplicaciones fuera de este campo, como la realidad aumentada o la realidad virtual, donde también es necesario conocer la posición del usuario respecto al entorno. También puede verse en algún campo menos común como el de la medicina para la asistencia en intervenciones quirúrgicas.

Para poder realizar SLAM se requiere una gran capacidad de cálculo para funcionar adecuadamente. Debido a esto, las áreas en las que se ha podido utilizar siempre han estado algo limitadas, pero con la velocidad a la que se mejoran las prestaciones de los procesador y con el desarrollo de nuevos algoritmos más eficientes, el rango de dispositivos capaces de llevarlo a cabo aumenta considerablemente, lo que genera nuevas ideas de aplicaciones.

Uno de los campos en los que comienza a introducirse es en el de los dispositivos móviles. Con el éxito de los smartphones en los últimos años el mercado de los procesadores móviles avanza a pasos agigantados por lo que poco a poco estas técnicas son cada vez más viables en este tipo de dispositivos, llegando a incluir algunos modelos de smartphones hardware específico para la realización de SLAM. Además, con los avances en este sector y el abaratamiento constante del hardware, el público objetivo crece considerablemente, incrementando significativamente las aplicaciones de esta tecnología y creando a su vez nuevos nichos de mercado todavía sin explotar. Algunos ejemplos donde ya se ha aplicado SLAM en dispositivos móviles son videojuegos que utilizan realidad aumentada o guías de museo mediante el móvil.

1.2 Objetivos

El objetivo principal del proyecto es desarrollar una aplicación para el sistema operativo Android capaz de crear mapas 3D densos del entorno en tiempo real. Para ello nos basaremos en la librería ORB-SLAM2[1] desarrollada en el grupo de robótica de la Universidad de Zaragoza. ORB-SLAM2 es capaz de localizar la cámara al mismo tiempo que construye un mapa de puntos dispersos del entorno. Puede utilizar cámaras monoculares estéreo y RGB-D, que además de la imagen en color proporcionan una imagen de profundidad.

El dispositivo donde se ejecutará la aplicación es una tableta Project Tango diseñada por Google, que cuenta con una cámara RGB-D. El objetivo de nuestra aplicación es utilizar las poses de la cámara calculadas por ORB-SLAM2 y las imágenes RGB-D para contruir incrementalmente un modelo 3D del entorno.

La librería ORB-SLAM 2 está pensada para trabajar en un PC por lo que será necesario portarla para Android asegurando el correcto funcionamiento de sus dependencias. En su versión para PC utiliza la librería Pangolin para la visualización. Esta librería tiene problemas de compatibilidad con la versión de OpenGl usada en Android por lo que se eliminarán todas sus referencias de ORB-SLAM2 y se desarrollará un visualizador propio basado en OpenGL ES 2.0 para la aplicación de Android.

Además, la librería requiere una gran capacidad de cálculo y al portarla a un dispositivo móvil su rendimiento disminuye considerablemente, por lo que también será necesario realizar optimizaciones que garanticen un rendimiento adecuado.

En resumen, los objetivos del proyecto son:

1. Portar ORB-SLAM2 a Android y desarrollar un nuevo visualizador.
2. Desarrollar la aplicación de reconstrucción 3D del entorno usando la cámara RGB-D del dispositivo Project Tango.
3. Optimizar su funcionamiento para obtener una buena experiencia de usuario.

2 Herramientas

2.1 Project Tango

El dispositivo utilizado para el desarrollo del proyecto es una tableta Project Tango de Google. Esta tableta no se diseñó con el objetivo de llegar a un público general, se trata de un modelo pensado para que los desarrolladores de aplicaciones móviles puedan trabajar en aplicaciones que utilicen realidad aumentada o relacionadas con la visión por computador.

El hardware con el que cuenta este dispositivo es bastante peculiar, especialmente la gran cantidad de sensores que incluye, a continuación se muestra una lista de sus características técnicas:

Características técnicas

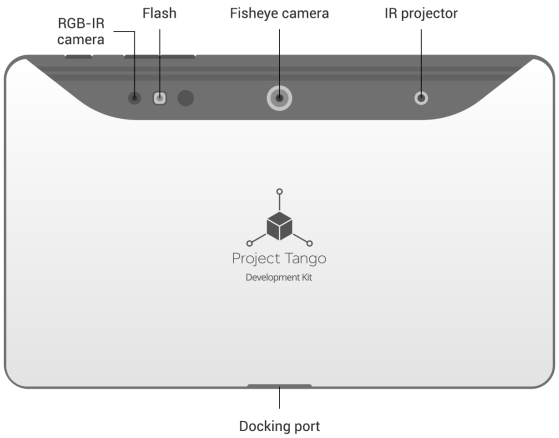

Figure 2.1: Cámaras Project Tango

- Cámara RGB-IR: Permite la obtención de imágenes a color a una resolución de 1280x720. Además también es la encargada de obtener la información de profundidad proporcionada por el emisor de infrarrojos a una resolución de 320x180.
- Emisor de infrarrojos: La cámara de profundidad funciona emitiendo un patrón de luz infrarroja que después es capturado por la cámara RGB-IR, permitiendo conocer a la distancia que se encuentran los objetos.

- Cámara gran angular: Permite la obtención de imágenes en blanco y negro con un gran ángulo de visión.
- Procesador NVIDIA Tegra K1: Este procesador está formado por una CPU ARM Cortex A-15 de cuatro núcleos a 2.3GHz y una GPU Nvidia Kepler con 192 nucleos CUDA.
- RAM: 4GB, suficiente para ejecutar cualquier tipo de aplicación.

2.2 Lenguajes y librerías

Librerías

- ORB-SLAM2: La librería en la que se basa el proyecto, utilizada para realizar SLAM. Esta a su vez utiliza otras librerías como DBoW2, g2o o Eigen.
- OpenCV: Se ha usado para el procesado de imágenes, además también es usado por ORB-SLAM2.
- Tango Support: Se trata de una librería proporcionada por Google para facilitar tareas relacionadas con sus dispositivos Project Tango.
- OpenGL ES 2.0: Utilizado en el visualizador desarrollado para la visualización de los modelos 3D.
- ARM Intrinsics: Son un conjunto de funciones similares al lenguaje ensamblador para la arquitectura de CPU ARM.

Lenguajes de programación:

- Java: El lenguaje principal de la aplicación, principalmente usado para gestionar los distintos componentes de la interfaz gráfica y para realizar las llamadas a las funciones de C++ de la aplicación.
- C++: Se ha utilizado en todo lo relacionado con la librería ORB-SLAM2 y con el procesamiento de datos de la aplicación

3 Diseño del sistema

3.1 Introducción

La función principal de la aplicación es la generación de un modelo 3D del entorno utilizando imágenes RGB-D. Para conseguirlo se ha usado la librería ORB-SLAM2, de la que se hablará en la siguiente sección. Además, una vez que se ha generado un modelo se da la opción de guardarlo en el almacenamiento interno del dispositivo. De esta forma el usuario tiene la posibilidad de visualizarlo más adelante.

3.2 Funcionamiento de ORB-SLAM2

Como puede verse en la figura 3.1, el funcionamiento de ORB-SLAM2 se divide en tres hilos principales, *tracking*, *local mapping* y *loop closing*.

El hilo de *tracking* se encarga del procesado de las imágenes. Su objetivo es la localización de la cámara en el entorno y la selección de un conjunto de imágenes clave(*keyframes*) que se utilizan en la construcción del mapa. Además también es el encargado de encontrar la localización de la cámara en el caso de que se pierda, debido por ejemplo a una oclusión o un movimiento demasiado brusco.

El *local mapping* es el encargado de procesar los *keyframes* seleccionadas por el hilo de *tracking* para construir el mapa. Los nuevos *keyframes* obtenidos son analizados respecto a los que ya han sido procesados para encontrar puntos en común y realizar optimizaciones mediante la técnica de ajuste de haces o Bundle Adjustment(BA) para conseguir una reconstrucción óptima del mapa. Además también se encarga de encontrar *keyframes* redundantes para eliminarlos y no sobrecargar el sistema con información repetida.

Por último, el hilo de *loop closing* está al cargo de detectar bucles en los nuevos *keyframes*. Un bucle se produce cuando tras haber iniciado el proceso de creación del mapa se vuelve a un lugar del que ya se ha obtenido información, pero esta vez siguiendo un camino distinto al original. Al detectar uno de estos bucles se corrige la deriva acumulada que se produce durante la creación del mapa, haciendo una nueva optimización del mapa(full BA).

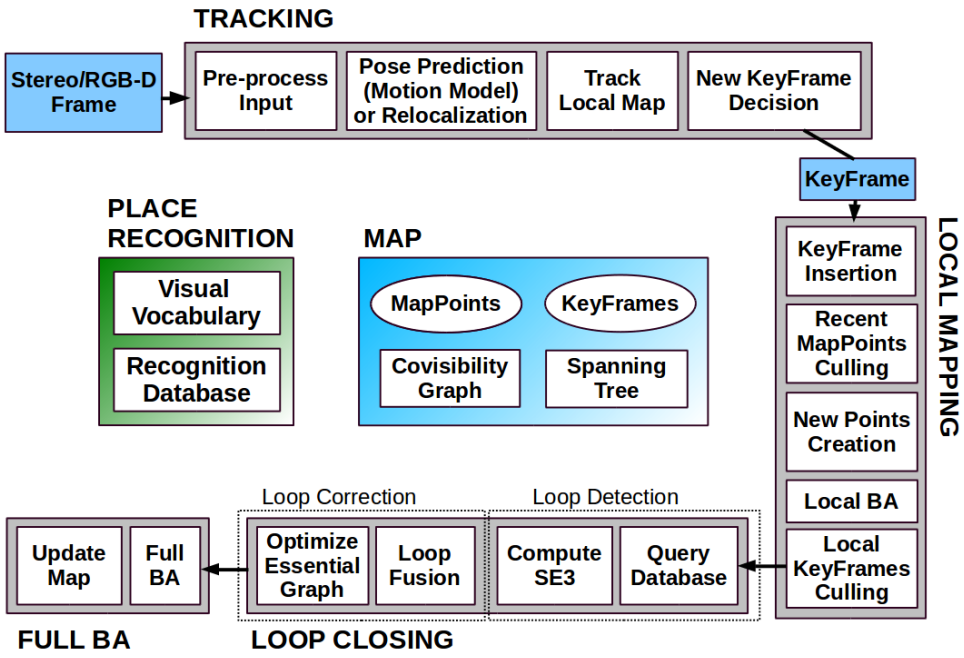

Figure 3.1: Esquema del funcionamiento de ORB-SLAM2[1]

3.3 Aplicación de reconstrucción 3D

Como puede verse en la figura 3.2, cuando el usuario inicia la reconstrucción se inician los hilos de ORB-SLAM2. Tras iniciar ORB-SLAM2, el hilo de *tracking* comienza a obtener las imágenes RGB-D de la cámara con las que se empieza a construir el mapa y el modelo 3D.

Un modelo 3D está formado por las poses de los *keyframes* y sus imágenes RGB-D, con las que se forman nubes de puntos de color. Como a cada *keyframe* le corresponde una nube de puntos, su construcción se ha integrado dentro de ORB-SLAM2 y de esta forma el modelo se va creando a medida que el hilo de *tracking* decide insertar nuevos *keyframes*.

El visualizador también es iniciado al principio de la generación del modelo. Este se encarga de mostrar los puntos del mapa que se va creando y la reconstrucción 3D, por lo que necesita acceder al mapa de ORB-SLAM2 para obtener los datos. Además cuando el usuario finaliza la generación del modelo, este puede guardarse en el almacenamiento del dispositivo para poder visualizarlo con posterioridad.

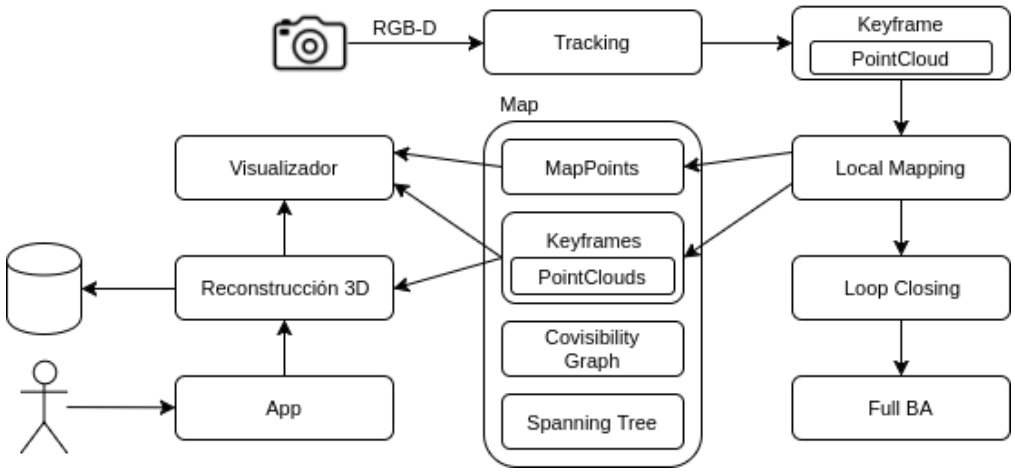

Figure 3.2: Diagrama de la generación de un modelo 3D

La aplicación también permite la visualización de modelos que hayan sido guardados previamente. En este caso ya no se depende de ORB-SLAM2, ya que los *keyframes* que forman el modelo ya se tienen almacenados en el disco. Tras cargar el modelo, este puede observarse por medio del visualizador y el usuario puede navegar por él utilizando la pantalla táctil del dispositivo. En la figura 3.3 se muestra el diagrama de la visualización de un modelo previamente almacenado.

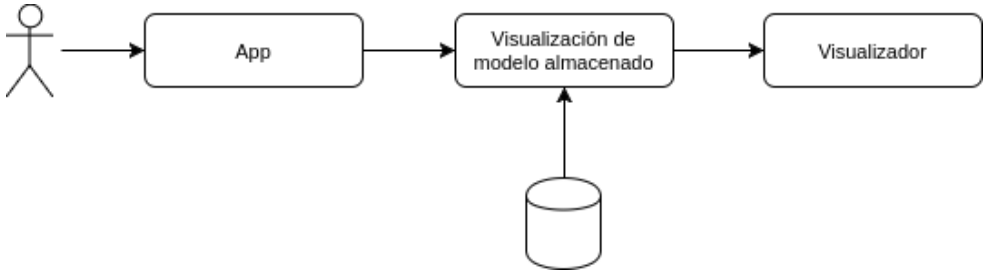

Figure 3.3: Diagrama de la visualización de un modelo almacenado en disco

4 Portado de ORB-SLAM2 a Android

4.1 Java y C++

Como norma general las aplicaciones de Android son programadas en Java utilizando el kit de desarrollo de software(SDK)[2] de Android, pero también existe la posibilidad de usar el kit de desarrollo nativo(NDK)[3] que permite usar C++ como lenguaje de programación.

La mayor de las ventajas que ofrece el NDK es la posibilidad de reutilizar librerías que ya han sido programadas en C ó C++ sin tener que volver a programarlas para Java, por lo que su uso es imprescindible en este proyecto ya que ORB-SLAM2 está programado en C++. Además su uso también ofrece la posibilidad de generar un código más optimizado en aplicaciones exigentes ya que C/C++ permite tener un mayor control sobre la memoria del dispositivo.

Aunque se decida utilizar el NDK la aplicación debe seguir teniendo parte de su código en Java para gestionar como mínimo las acciones del usuario, la interfaz gráfica y las llamadas a las funciones de C++ desarrolladas.

4.2 Compilación

La librería ORB-SLAM2 está preparada para ser compilada y ejecutada en un ordenador con sistema operativo Linux, pero en este proyecto el dispositivo que va a utilizar la librería es distinto, por lo que es necesario realizar algunos ajustes en el proceso de compilación.

Como la mayoría de los dispositivos móviles la tableta Project Tango cuenta con una arquitectura de procesador ARM, por lo que hay que tener en cuenta que la librería va a ser compilada por un procesador con una arquitectura distinta a la del dispositivo que la ejecutará. Al tener una arquitectura de procesador diferente es necesario realizar un proceso de compilación cruzada para conseguir que la librería pueda ejecutarse en el dispositivo móvil. Para ello se han utilizado las herramientas proporcionadas en el NDK de Android, que permiten realizar la compilación cruzada.

ORB-SLAM2 depende de otras librerías para funcionar y al igual que ocurre con ORB-SLAM2 estas también deben ser preparadas para funcionar en nuestro dispositivo. Debido a esto, antes de poder compilar ORB-SLAM2 es necesario preparar las librerías de las que depende.

4.3 Dependencias

ORB-SLAM2 tiene varias dependencias de librerías que hay que resolver para poder portar la librería con éxito. Las dependencias son las siguientes:

- Pangolin: se utiliza para la interfaz de usuario y para la visualización de la librería. Al tratar de compilarla para Android se han encontrado problemas de compatibilidad con la versión de OpenGL utilizada en Android, por lo que se ha decidido eliminar esta dependencia por completo para sustituirla por un visualizador propio. En la sección 4.4 se habla sobre el visualizador desarrollado.
- OpenCV: puede obtenerse directamente compilada para funcionar en Android en forma de librería dinámica. En el proyecto se ha decidido utilizar una versión proporcionada por Nvidia que cuenta con optimizaciones para los procesadores Tegra, que es el que tiene nuestro dispositivo.
- DBow2 y g2o: de la misma forma que ORB-SLAM2, estas librerías debe ser compilada mediante compilación cruzada para que puedan funcionar en nuestro dispositivo.
- Eigen: esta librería no es una dependencia directa de ORB-SLAM2, pero si que lo es de la librería g2o, por lo que también hay que tenerla en cuenta. Se trata de una librería *header-only*, por lo que no es necesario compilarla.

Una vez resueltas todas las dependencias ya se puede compilar ORB-SLAM2 mediante compilación cruzada utilizando el NDK de Android.

4.4 Visualizador

Para realizar la visualización en el proyecto original de ORB-SLAM2 se utiliza la librería Pangolin, como puede verse en la figura 4.1. En este proyecto se ha decidido no utilizar dicha librería por razones de compatibilidad. En lugar de utilizar la librería Pangolin se ha implementado un visualizador para el que se han utilizado los componentes de interfaz de usuario que ofrece el SDK de Android y también OpenGL ES 2.0 para la visualización del mapa generado por ORB-SLAM2. El visualizador desarrollado se muestra en la figura 4.2.

En el visualizador se ha replicado el funcionamiento del sistema original, en el que se pueden observar los puntos del mapa que va creando ORB-SLAM2 junto con la posición de la cámara respecto a estos. Además se permite la navegación por el mapa utilizando gestos táctiles. Aparte de la visualización del mapa, también se pueden observar las imágenes procesadas a color y de profundidad, situadas en las esquinas superior e inferior derechas de la figura 4.2 respectivamente. En la figura 4.3 también puede verse la reconstrucción del entorno, de la que se hablará en el siguiente capítulo.

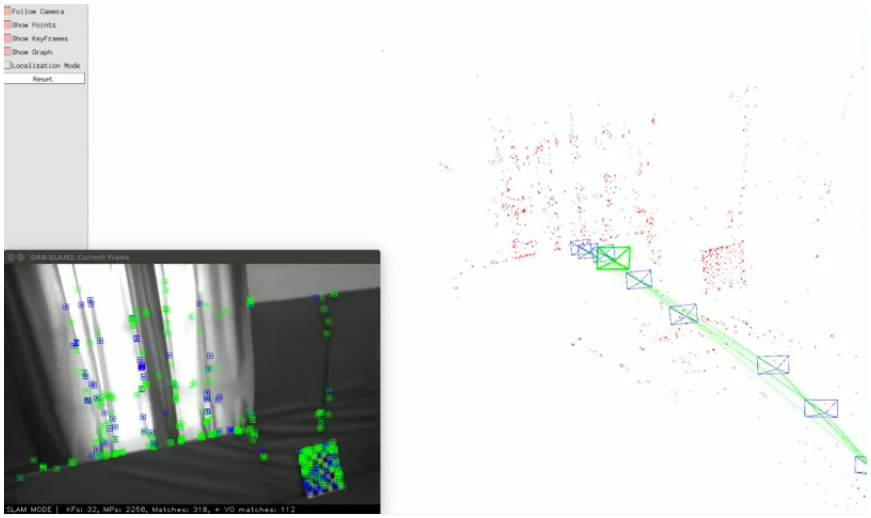

Figure 4.1: Visualización original de ORB-SLAM2 utilizando Pangolin sobre Linux.

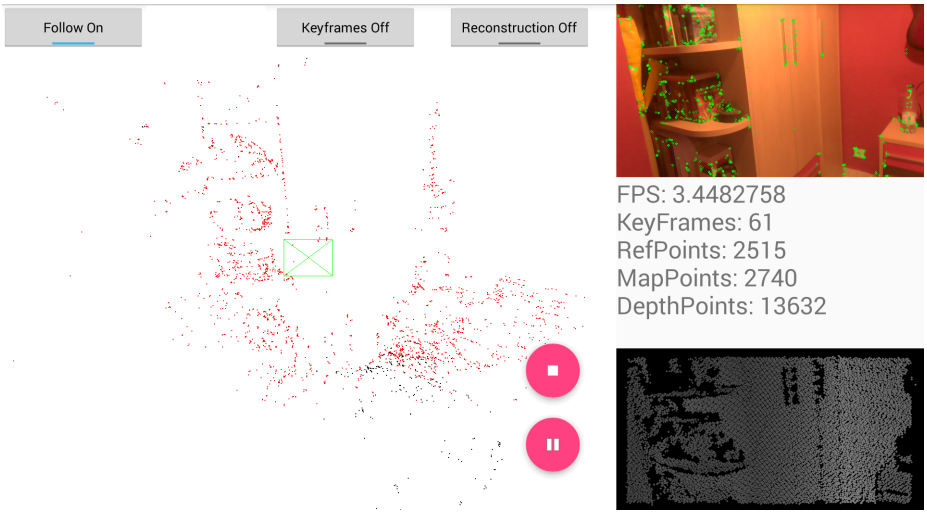

Figure 4.2: Visualización desarrollada para Android.

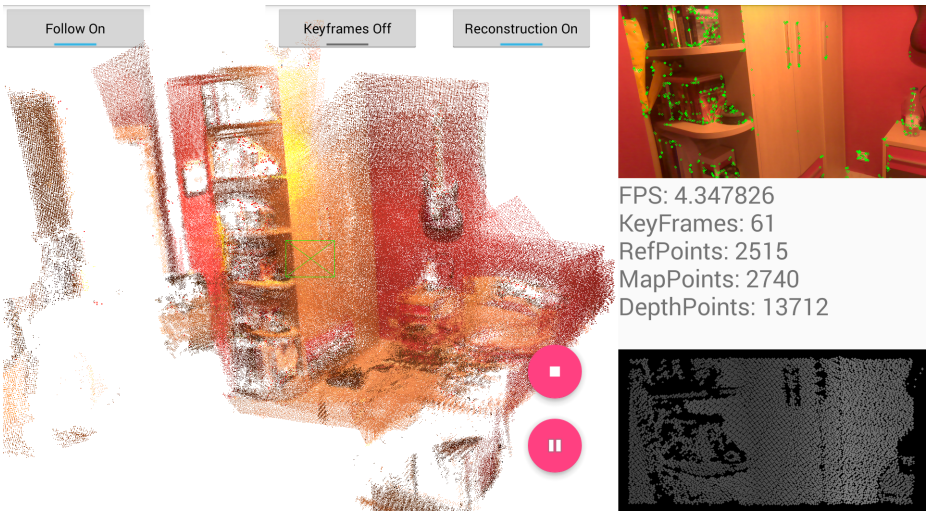

Figure 4.3: Visualización de la reconstrucción del modelo.

5 Reconstrucción 3D del entorno

La finalidad del proyecto es la creación de una aplicación que obtenga un modelo 3D del entorno, para conseguirlo se han utilizado las imágenes de RGB-D y se ha integrado su generación en ORB-SLAM2.

5.1 Obtención de las imágenes

Para obtener las imágenes RGB y las de profundidad se ha utilizado la API de Project Tango[4] proporcionada por Google, en concreto su versión para C/C++. La API permite el acceso a los datos capturados por los sensores de la tableta aunque no todos ellos se encuentran disponibles, por ejemplo no es posible obtener las imágenes de la cámara gran angular.

Imagen RGB

Las imágenes a color se han obtenido mediante la cámara RGB-IR, esta cámara obtiene las imágenes a una resolución de 1280x720 y trabaja a una frecuencia máxima de 30Hz. El espacio de color en el que se representan las imágenes capturadas por la cámara es YUV420.

En ORB-SLAM2 se trabaja con imágenes en escala de grises, como las imágenes originales están en formato YUV puede extraerse únicamente el canal Y que corresponde a la luminancia y así ORB-SLAM2 puede utilizarla sin realizar ningún tipo de conversión. A la hora de construir los modelos del entorno se necesita tener información sobre el color, por lo que se ha realizado un cambio del espacio de color a RGB. Se ha elegido este formato por ser el más adecuado con el que trabajar en OpenGL, que se utiliza en la visualización de los modelos.

Por otra parte, las imágenes tienen una resolución demasiado alta para utilizarlas con ORB-SLAM2, ya que a una resolución mas grande, mayor es el tiempo requerido en su procesado. Por eso antes de usarlas se ha realizado un reescalado disminuyendo su resolución hasta un total de 640x360, una resolución adecuada para realizar SLAM en tiempo real contando con las limitaciones en la capacidad de cálculo que tiene el dispositivo objetivo.

Imagen de profundidad

Para obtener la imagen de profundidad se ha utilizado el sensor de infrarrojos integrado en la tableta. La resolución obtenida es de tan solo 320x180, así que es necesario procesar la imagen para obtener una con la misma resolución que la utilizada en la imagen monocromática y así poder utilizarlas en ORB-SLAM2.

La API no permite obtener la imagen de profundidad directamente, en cambio proporciona una nube de puntos que contiene la información de profundidad del entorno. La nube es representada como un vector que contiene las coordenadas de cada uno de los puntos de los cuales se ha podido obtener información, que tienen que ser procesados para construir la imagen de profundidad.

La imagen se representa en escala de grises. Como el rango que pueden tomar los píxeles en este tipo de imagen se encuentra en el rango (0,255), se ha de procesar el valor de profundidad obtenido por la cámara para poder representarlo. Por lo que se ha dividido el rango de distancia en el cual el sensor funciona correctamente, que son unos 4 metros y medio, para asignar un valor en la escala de grises. La nube de puntos tiene una resolución menor que la imagen monocromática utilizada por ORB-SLAM2, por lo que a la hora de construir la imagen de profundidad se ha realizado un reescalado de las coordenadas de la nube de puntos y se ha aumentado el radio de los puntos para obtener una imagen similar a la que se obtiene con la resolución original. Se muestra un ejemplo en la figura 5.1. Dado que la imagen obtenida no necesita de un detalle excesivo este reescalado simple es suficiente, ya que se si realizara algo más sofisticado se penalizaría el rendimiento de la aplicación.

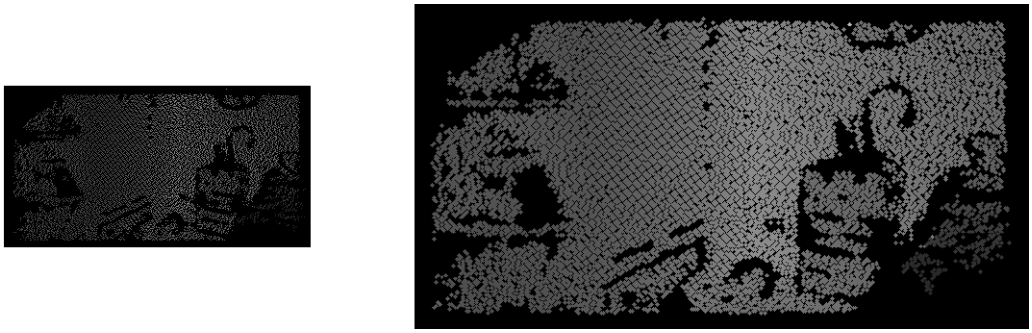

Figure 5.1: Imágenes construidas original y reescalada.

Existen limitaciones en el sensor de profundidad, ya que al ser un sensor integrado en una tableta este no puede ser demasiado potente. Esto afecta al rango de visión en el que el sensor es capaz de obtener información precisa, en concreto, el rango de visión se encuentra entre los 0.5 y los 4.5 metros, por lo que el sensor no funcionará correctamente cuando nos encontremos muy cerca o lejos del medio que se este procesando en el momento. Además la obtención de imágenes de profundidad está limitada a 5Hz, por lo que obtenemos una limitación en la frecuencia máxima a la que se podrán procesar los datos en la aplicación. Por último también es necesario tener en cuenta la tecnología con la que funciona el sensor, al estar basado en emisión de infrarrojos la información obtenida puede no ser correcta en situaciones en las que la iluminación sea alta en luz infrarroja como la luz solar, y que algunos materiales no reflejan dicha frecuencia de luz, por lo que no es posible obtener su información de profundidad.

5.2 Generación de modelos 3D

Para la generación del modelo se han utilizado las imágenes a color y las de profundidad utilizadas para realizar SLAM. Estas imágenes corresponden a un mismo lugar en el mismo instante de tiempo, por lo que si se combina la información de cada una de ellas se puede obtener una nueva nube de puntos con información del color, como se puede observar en la figura 5.2.

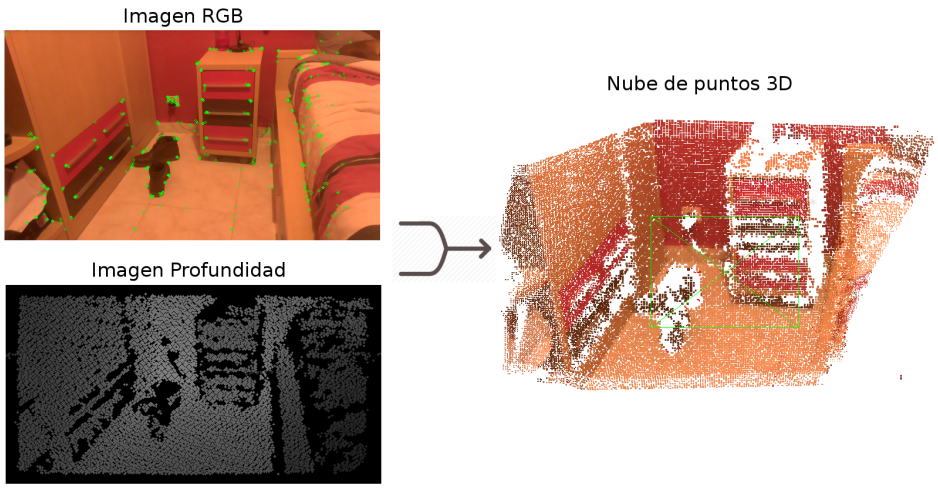

Figure 5.2: Generación nube de puntos 3D

Cada una de estas nubes de puntos corresponden a una pequeña parte del modelo que se quiere generar, por lo que para obtener el modelo completo del entorno es necesario combinar todas las nubes de puntos. Como estamos posicionados en el mapa en todo momento con la ayuda de ORB-SLAM2, se conoce la posición con la que se han capturado las imágenes y con esta información se pueden combinar todas las nubes de puntos generadas en una única reconstrucción total del entorno. A continuación se puede ver en la figura 5.3 un ejemplo de la creación progresiva de un modelo.

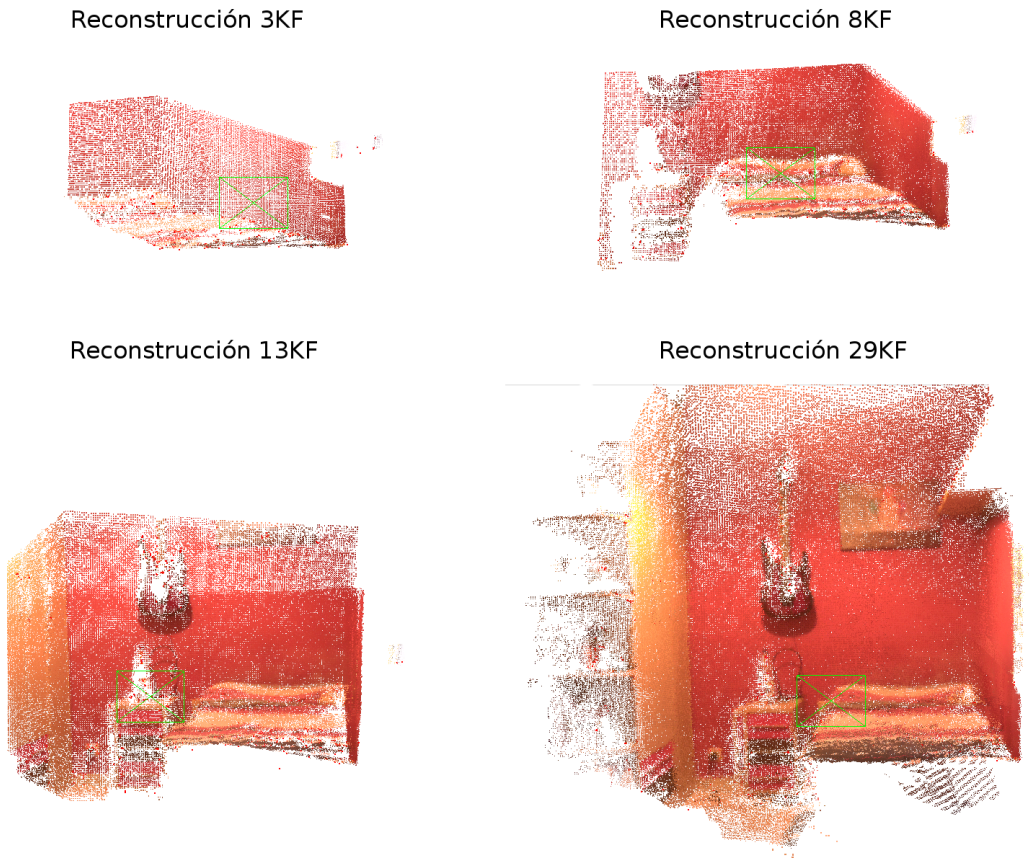

Figure 5.3: Generación progresiva del modelo

La generación de las nubes de puntos se ha integrado dentro del código de ORB-SLAM2 para que se ejecute al mismo tiempo que se realiza SLAM. La frecuencia con la que se genera una nueva nube de puntos es a nivel de *keyframe*, por lo que ahora cada vez que ORB-SLAM2 decida la inserción de

un nuevo *keyframe*, se almacenará con el resto de su información sus imágenes RGB y de profundidad. De esta forma el modelo final se pueda reconstruir accediendo a todos los *keyframes*. La decisión de crear las nubes de puntos al seleccionar un *keyframe* se debe a que de esta forma se garantiza que el modelo final contenga toda la información relevante del entorno ya que ORB-SLAM2 inserta un nuevo *keyframe* cuando detecta un cambio visual mínimo. Además ORB-SLAM2 también procesa los *keyframes* para realizar optimizaciones en sus poses, por lo que el modelo final es lo más preciso posible.

5.3 Visualización de los modelos

Para poder explorar los modelos generados, el visualizador implementado permite la navegación por el modelo. Se ha decidido que la cámara más adecuada para la observación de este tipo de modelos es una del tipo "arcball". Este tipo de cámara se caracteriza porque los movimientos que se realizan con la cámara siguen una trayectoria circular respecto a un punto. Se permite realizar rotaciones sobre el eje horizontal y el vertical y el punto sobre el que se realiza la rotación se puede mover para permitir el desplazamiento por el modelo. Además también se permite regular el radio de las trayectorias seguidas en las rotaciones para poder controlar el nivel de zoom en la visualización.

A la hora de visualizar el modelo se tienen todos los *keyframes* que lo forman, pero para obtener una visualización correcta es necesario realizar un control sobre los que se deben mostrar y los que no, ya que dependiendo de la posición desde la que se observe el modelo algunas partes no deberían ser visibles. Para evitar este efecto se comprueba posición desde la que fue capturado cada uno de los *keyframes* y se compara con la posición desde la que se observa el modelo en el visualizador, de forma que si desde ambas posiciones se mira en la misma dirección con un ángulo máximo de 90º, se dibujará la nube de puntos de dicho *keyframe* y en caso contrario se ignorará. En la figura 5.4 se puede ver la comparación cuando no se controla la dirección de visualización y cuando si se hace.

Figure 5.4: Comparación aplicando el control en la dirección de visualización

6 Optimizaciones

La librería ORB-SLAM2 requiere una gran capacidad de cálculo para funcionar correctamente ya que requiere el procesado de imágenes en tiempo real, por lo que si estas imágenes no pueden ser procesadas a una frecuencia adecuada la experiencia de uso no es satisfactoria.

6.1 Rendimiento inicial

En la primera versión de la aplicación se utilizaban las imágenes originales obtenidas de la cámara, con una resolución de 1280x720. Estas imágenes de resolución tan alta provocan que el procesado de las mismas requiera demasiado tiempo y en un dispositivo portátil con una capacidad de cálculo limitada no es viable. En las pruebas realizadas con esta resolución se consiguió procesar una imagen por segundo. Esto produce una mala experiencia de uso limitando demasiado los movimientos del usuario, ya que procesando las imágenes a una frecuencia tan baja la librería pierde la posición del dispositivo si se realizan movimientos rápidos y bruscos.

Una vez obtenidos estos primeros resultados se decidió reducir el tamaño de las imágenes a la mitad consiguiendo unas dimensiones de 640x360, que se trata de unas dimensiones bastante más adecuadas para la ejecución en tiempo real. Con esta nueva resolución la frecuencia a la que se podían procesar las imágenes aumentó hasta los 3-4 Hz, lo que permitía una mayor libertad de movimiento.

6.2 Medición de tiempos

Tras reducir la resolución de las imágenes se realizó un estudio sobre el tiempo empleado en cada una de las funciones principales de ORB-SLAM2. De esta forma se puede obtener información sobre el reparto del tiempo utilizado y comprobar si existe un cuello de botella en el cual se puedan realizar optimizaciones para mejorar la frecuencia de procesado de imágenes. Las pruebas se ha realizado escaneando una habitación al completo asegurándose que se ejecutan varias cerrados de bucles, utilizando una resolución de 640x360, extrayendo 800 puntos ORB y de 320x180, extrayendo 600 puntos.

Tablas de tiempos

En primer lugar tenemos los tiempos obtenidos de las funciones de más alto nivel en la tabla 6.1. Se puede observar como el mayor tiempo se encuentra en la realización del Local Bundle Adjustment, este tiempo es muy alto pero esta operación es realizada a frecuencia de *keyframe* por lo que el tiempo requerido en esta función no es el más crítico. El siguiente mayor tiempo con diferencia lo obtenemos en la extracción ORB. Esta función si que se realiza en cada frame por lo que es importante optimizarla lo mejor posible, ya que se pueden conseguir grandes mejoras en el rendimiento de la aplicación.

	640x360 (800 puntos)		320x180 (600 puntos)	
	Media (ms)	DS (ms)	Media (ms)	DS(ms)
ORB Extraction	146.31	62.64	74.25	34.56
Initial Pose Estimation	34.64	24.37	22.51	13.25
Track Local Map	36.60	23.10	19.91	11.11
New KeyFrame Decision	4.38	6.55	1.71	2.75
KeyFrame Insertion	60.48	23.42	37.55	15.44
Map Point Culling	2.04	1.32	1.29	1.28
Map Point Creation	4.20	7.17	0.53	1.39
Local BA	493.47	260.63	257.19	176.96
KeyFrame Culling	12.77	13.63	6.14	6.58

Table 6.1: Tiempos de las funciones de alto nivel

Se ha estudiado en profundidad la función de ORB extraction comentada en la tabla anterior, tras medir los tiempos se puede observar como casi todo el tiempo utilizado dentro de la función es gastado en la función ExtractORB que se detallará a continuación.

	640x360 (800 puntos)		320x180 (600 puntos)	
	Media (ms)	DS (ms)	Media (ms)	DS(ms)
Scale Level Info	0.01	0.05	0.01	0.09
ExtractORB	145.20	62.16	73.56	34.35
UndistorKeyPoints	0.02	0.02	0.02	0.04
ComputeStereoFromRGB	0.11	0.14	0.05	0.06
AssignFeaturesToGrid	0.80	0.83	0.53	0.67

Table 6.2: Tiempos ORB Extraction

Tras medir la función en profundidad se observa que se utiliza parte del tiempo en la función KeyPoints OctTree, que se encarga de la extracción de puntos FAST, pero la mayoría del tiempo es invertido en la función que se encarga de calcular los descriptores de los puntos ORB, con un tiempo medio de algo más de 100 ms para la resolución de 640x360. Es una de las funciones que más penalizan la frecuencia a la que trabaja la aplicación ya que esta función es ejecutada en cada una de las imágenes capturadas, por lo que es la mejor candidata a ser optimizada.

	640x360 (800 puntos)		320x180 (600 puntos)	
	Media (ms)	DS (ms)	Media (ms)	DS(ms)
Scale Pyramid	4.38	2.00	0.99	0.73
KeyPoints OctTree	19.23	6.72	6.28	3.03
Gaussian Blur	16.30	6.29	4.02	1.52
Compute Descriptors	105.02	52.70	62.14	31.63

Table 6.3: Tiempos ExtractORB

6.3 Optimización del cálculo de descriptores

La función a optimizar consiste en el cálculo de los descriptores ORB. El descriptor a calcular consiste en 32 bytes donde cada uno de los bits es independiente del resto, por lo que su cálculo se puede realizar de forma paralela.

Dado que el cálculo del descriptor consiste en realizar las mismas operaciones sobre distintos datos, se ha decidido utilizar la tecnología NEON[5] de ARM, que es la implementación del set de instrucciones *Advanced Single Instruction Multiple Data*(SIMD) para la arquitectura ARMv7-A y permite la ejecución de una operación sobre varios datos en una misma instrucción del procesador. En concreto se han utilizado las "ARM assembler instruction intrinsics", estas instrucciones son un conjunto de funciones para C/C++ que son sustituidas por código ensamblador que ejecuta instrucciones NEON al realizar la compilación. Como se ha comentado en el apartado de las características técnicas, la tableta Project Tango tiene con un procesador NVIDIA Tegra K1 que cuenta con un procesador ARM Cortex-A15. El banco de registros reservado para las instrucciones NEON está formado por 32 registros de 64 bits. Estos registros pueden interpretarse como registros de mayor tamaño si se agrupan en grupos de dos entre ellos, lo que forma un total de 16 registros de 128 bits y permite aumentar la cantidad de datos procesada por instrucción.

El código a optimizar está formado en su mayor parte por la realización de multiplicaciones, sumas y restas de números de tipo flotante, además de la realización de comparaciones. Utilizando las instrucciones NEON se puede optimizar el código para que las operaciones aritméticas se puedan realizar en grupos de 4 datos, ya que los datos de tipo flotante tienen un tamaño de 32 bits en nuestra arquitectura y el máximo tamaño de registro permitido es de 128 bits. Los detalles de las optimizaciones realizadas se encuentran en el anexo A.

Tras realizar la optimización, el tiempo utilizado en el cálculo de los descriptores ha mejorado notablemente, pasando de los 100 ms de media que tardaba sin las optimizaciones para la arquitectura NEON, a tan solo 5 ms de media. Como ya se ha comentado, esta función es ejecutada por cada imagen procesada, por lo que se ha mejorado la frecuencia hasta los 6 o 7 frames por segundo con la resolución de 640x360.

6.4 Carga del vocabulario

La librería ORB-SLAM2 necesita realizar la carga de un vocabulario para comenzar a funcionar. El vocabulario tiene un tamaño de 145MB y se encuentra almacenado en un fichero de texto. El formato en el que se almacenan los datos puede no tener mucha importancia en un ordenador convencional, pero en un dispositivo móvil donde la capacidad de cálculo y la memoria son mucho más limitadas si que juegan un papel muy importante. Con el fichero en formato txt la carga del vocabulario tarda 50 segundos hasta completarse, esto implica que el usuario tiene que esperar casi un minuto hasta que puede comenzar a usar la aplicación lo que ofrece una experiencia de uso muy mala.

Para mejorar el tiempo de carga se decidió usar como formato de almacenamiento un fichero binario que permite una lectura mucho más rápida que en un fichero de texto además de reducir su tamaño en disco. Tras convertir el vocabulario a formato binario y adaptar la función de lectura del fichero se consiguió reducir el tiempo de carga a tan solo 5 segundos, una reducción del 90%, además el tamaño del fichero también queda reducido en un 70%, ocupando finalmente solo 44MB.

	Tiempo	Tamaño
Texto	50 s	145 MB
Binario	5 s	44 MB

6.5 Almacenamiento en disco de los modelos

En primer lugar se decidió utilizar el formato de texto como medio de almacenamiento. Tras realizar varias pruebas se encontró el problema de que los modelos necesitaban bastante tiempo para cargarse, por lo que finalmente se decidió usar ficheros binarios. Con el uso de ficheros binarios los tiempos de carga se ven reducidos drásticamente y para comprobarlo se realizó una prueba en la que se generó un nuevo modelo con 110 *keyframes*, el modelo fue guardado tanto en fichero de texto como en binario para poder comparar los resultados. Los resultados indican que se disminuye el tiempo de carga de 11000ms que cuesta en fichero de texto a solo 600 ms en binario y en cuanto a tamaño se ha conseguido reducir el espacio utilizado a la mitad, ocupando en txt 116 MB y 50MB en binario.

7 Conclusiones

Tras haber finalizado el Trabajo de Fin de Grado se ha conseguido una aplicación que se ejecuta en una tableta Project Tango con la que se obtiene una reconstrucción 3D del entorno y que permite guardar dichos modelos para poder visualizarlos con posterioridad. Además la aplicación ofrece un rendimiento adecuado teniendo en cuenta las exigencias computaciones de la aplicación y las características del dispositivo sobre el que se ejecuta. Viendo estos resultados se puede decir que se han cumplido los objetivos propuesto al inicio del proyecto.

Problemas encontrados

Los principales problemas encontrados han estado relacionados con la utilización de tecnologías con las que se tenía poca o ninguna experiencia, especialmente en el caso de las que no son muy utilizadas como el NDK de Android o la tableta Project Tango, ya que la búsqueda de información es más complicada.

Posibles ampliaciones

Entre las posibles ampliaciones que se podrían hacer al trabajo, estaría la posibilidad de realizar más optimizaciones de código para aumentar la frecuencia a la que funciona la aplicación.

Otra posibilidad sería la de que una vez que obtenido un modelo se podría procesar para realizar simplificaciones al modelo si se detecta, por ejemplo, que una gran cantidad de puntos forman parte de una superficie plana, también se podrían realizar correcciones de colores que pueden producirse al capturar las imágenes con distintos niveles de iluminación.

Tiempo

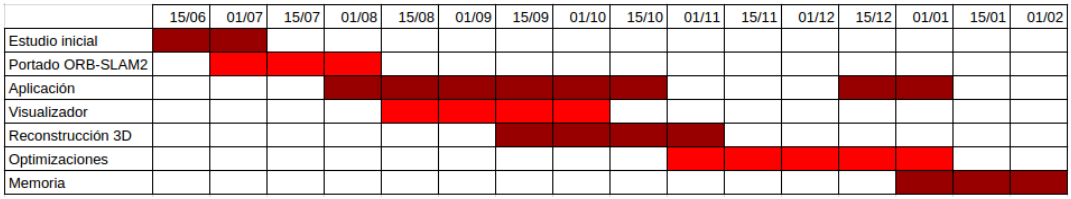

Figure 7.1: Diagrama de Gantt

Opinión personal

Una vez terminado el trabajo y tras haberlo valorado puedo decir que ha sido una experiencia satisfactoria. Es verdad que ha habido momentos difíciles, especialmente al tratar con algunos de los problemas encontrados, pero esto también me ha servido para aprender a afrontar los problemas de forma distinta.

Destacar también todos los conocimientos aprendidos a lo largo del proyecto sobre las herramientas utilizadas y conocer más a fondo como funciona un sistema de SLAM, además de haber tenido la oportunidad de trabajar con tecnología que tiene un futuro prometedor.

Bibliografía

[1] Raúl Mur-Artal, and Juan D. Tardós. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras., https://128.84.21.199/abs/1610.06475, 2016.

[2] Android SDK, https://developer.android.com/guide/index.html

[3] Android NDK, https://developer.android.com/ndk/index.html

[4] Project Tango API, https://developers.google.com/tango/apis/overview

[5] NEON, https://www.arm.com/products/processors/technologies/neon.php