Universidad Zaragoza
1542

Trabajo Fin de Máster

Percepción y Análisis 3D de Escenas de Carretera para Vehículos Autónomos

3D Road Scene Perception and Analysis for Autonomous Vehicles

Autor

Jorge Barrio Arbex

Directores

José Javier Yebes Torres

Cristian Mahulea

Universidad de Zaragoza
Escuela de Ingeniería y Arquitectura
Máster en Ingeniería Industrial

2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Jorge Barrio Arbex,

con nº de DNI 73003396-Q en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Máster en Ingeniería Industrial, (Título del Trabajo) Percepción y Análisis 3D de Escenas de Carretera para Vehículos Autónomos

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 27 de Agosoto de 2018

Fdo: JORGE BARRIO ARBEX

Percepción y Análisis 3D de Escenas de Carretera para Vehículos Autónomos

Resumen

Este trabajo aborda dos dificultades presentes en los sistemas actuales de localización y creación de mapas de forma simultánea (SLAM, en inglés Simultaneous Localization And Mapping) para la conducción autónoma: la reconstrucción 3D del entorno recorrido por el vehículo (mapeado) y la estimación del movimiento y la posición del vehículo (odometría).

Existen varios métodos y tecnologías capaces de obtener soluciones, más o menos óptimas, para los problemas en cuestión. En este TFM se propone el uso de la tecnología LiDAR para la percepción 3D del entorno del vehículo y la posterior extracción de información semántica de interés. Los datos obtenidos del LiDAR se formatean como nubes dispersas de puntos 3D y se propone el uso de una variante del algoritmo ICP (Iterative Closest Point) para realizar la alineación de dichas nubes, obteniendo así la matriz de transformación necesaria para poder realizar la reconstrucción 3D y la odometría.

En primer lugar, se presenta un estudio de mercado de los dispositivos LiDAR actuales aplicados a los sistemas de asistencia a la conducción y vehículos autónomos. Para el desarrollo de este trabajo se han utilizado datos de varios LiDAR de la marca Velodyne: VLP-16, HDL-32E y HDL-64E.

En segundo lugar, se han estudiado y evaluado diferentes variantes del algoritmo ICP con el objetivo de realizar la reconstrucción 3D y la odometría. Para ello, destacan 5 funciones principales durante el desarrollo de este trabajo. 1) adquisición de datos del LiDAR y organización de éstos en nubes de puntos 3D; 2) construcción y representación de imágenes panorámicas de 360º con el valor de la intensidad y de la profundidad de cada punto; 3) estimación de la matriz de transformación entre dos nubes 3D consecutivas mediante distintas variantes de el algoritmo ICP, 4) cálculo de la odometría en el plano suelo y representación en una imagen del recorrido realizado por el vehículo; 5) reconstrucción 3D de la escena mediante la fusión de las nubes de puntos a lo largo del recorrido del vehículo.

Para la evaluación del método y algoritmos propuestos se han utilizado datos de dos sensores. Un LiDAR Velodyne HDL-32E instalado en un coche de la Fundación Vicomtech que realizó un recorrido por el Parque tecnológico Miramón en San Sebastián. Un LiDAR Velodyne HDL-64E instalado en un coche del Karlsruhe Institute of Technology cuyos datos están disponibles públicamente en la base de datos y herramienta de evaluación KITTI.[1] La precisión se ha medido mediante la comparación del recorrido estimado y un recorrido de referencia obtenido mediante sistemas de posicionamiento y sensores calibrados. Dichos datos de referencia se conocen comúnmente como "ground truth".

A pesar de que la tecnología LiDAR es muy empleada para la reconstrucción 3D y no tanto para la odometría, ya que existen dispositivos más rápidos y precisos, este TFM muestra resultados satisfactorios en la estimación del movimiento del vehículo. Este trabajo consigue una estimación precisa de la odometría del vehículo usando solamente datos procedentes de sensores LiDAR y también permite la reconstrucción de 3D del recorrido del vehículo. Los resultados obtenidos indican que el algoritmo Generalized-ICP es el más adecuado para la estimación de las matrices de transformación entre nubes de puntos 3D consecutivas.

[1] http://www.cvlibs.net/datasets/kitti/setup.php

Tabla de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En los últimos años se han producido grandes avances tecnológicos en los sistemas de transporte autónomos, pero aún quedan muchos aspectos por investigar y desarrollar para poder ver coches completamente autónomos y sin conductor en las carreteras. La investigación académica en este campo aumenta cada año. En el sector industrial se han creado grandes alianzas entre empresas de Internet, desarrollo software, hardware y las grandes marcas del sector de la automoción. En los dos ámbitos existe una fuerte inversión en I+D para diseñar, desarrollar y probar los vehículos autónomos del futuro.

Hasta hace unos años los sistemas de asistencia a la conducción se basaban principalmente en la percepción del entorno mediante cámaras y sensores de proximidad. Sus señales se procesaban de forma independiente para resolver tareas muy concretas. Sin embargo, el estado del arte actual y los retos futuros del coche sin conductor requieren sistemas complejos con múltiples sensores y potentes módulos hardware y software. En respuesta a dichas necesidades están surgiendo nuevos sistemas para adquisición, sincronización y procesamiento de múltiples flujos de datos y específicamente diseñados para instalarse en vehículos autónomos. Por un lado, dichos sistemas deben contar con módulos robustos, fiables y de rápida respuesta para las tareas más críticas. Por otro lado, deben ofrecer múltiples interfaces para los sensores e interconexión con otros sistemas y capacidades de procesamiento para grandes volúmenes de datos (imágenes 2D y nubes de puntos 3D) así como APIs de programación ágiles para la I+D de funcionalidades de alto nivel. Por ejemplo el entendimiento de escenas y la extracción de información semántica útil y usable.

Este trabajo aborda dos dificultades presentes en los sistemas actuales de localización y creación de mapas de forma simultánea (SLAM, en inglés Simultaneous Localization And Mapping) para la conducción autónoma: la reconstrucción 3D del entorno recorrido por el vehículo y la estimación del movimiento y la posición del vehículo (odometría).

En este capítulo introductorio se revisa el estado del arte, los objetivos, metodología y herramientas utilizadas durante el desarrollo de este trabajo.

1.1. Estado del arte

Los últimos años han sido testigos de un progreso sorprendente en los campos relacionados con la IA (Inteligencia Artificial), como la visión artificial, el aprendizaje automático y los vehículos autónomos. Desde las primeras demostraciones exitosas en la década de 1980 (Dickmanns & Mysliwetz [1], Dickmanns & Graefe [2], Thorpe et al. [3]), se ha avanzado mucho en el campo de los vehículos autónomos. Sin embargo, a pesar de estos avances, es seguro creer que la navegación totalmente autónoma en entornos complejos todavía está a décadas de distancia. La razón de esto es doble: en primer lugar, los sistemas autónomos requieren de inteligencia artificial que opere ante situaciones impredecibles. En segundo lugar, las decisiones requieren una percepción precisa, sin embargo, la mayoría de los sistemas de visión por computadora existentes producen errores inaceptables para la navegación autónoma [4].

Muchas instituciones gubernamentales de todo el mundo comenzaron varios proyectos para explorar sistemas inteligentes de transporte (ITS). El proyecto PROMETHEUS comenzó en 1986 en Europa e involucró a más de 13 fabricantes de vehículos, varias unidades de investigación de gobiernos y universidades de 19 países europeos.

Uno de los primeros proyectos en los Estados Unidos fue Navlab Thorpe et al. (1988) [3] de la Carnegie Mellon University, que logró un hito importante en 1995 al completar la primera conducción autónoma de Pittsburgh, PA y Sand Diego, CA. Después de que muchas universidades, centros de investigación y compañías automotrices lanzaran iniciativas, el gobierno de los EE. UU. estableció el Consorcio Nacional de Sistemas de Carreteras Automatizadas (NAHSC) en 1995. De forma similar lo hizo Japón en 1996 (Asociación Avanzada de Investigación de Sistemas de Autopistas Cruise). Bertozzi et al. (2000) [5] llegaron a la conclusión de que la potencia informática necesaria para desarrollar vehículos autónomos está cada vez cerca, pero las dificultades como los reflejos, las carreteras mojadas, la luz directa del sol, los túneles y las sombras todavía dificultan la interpretación de los datos. Por lo tanto, sugirieron la mejora de las capacidades del sensor.

El proyecto V-Charge Furgale et al. (2013) [6] presenta un automóvil eléctrico automatizado equipado con sensores del mercado. Se propone un sistema totalmente operativo que incluye la localización, el mapeo, la navegación y el control únicamente visuales. El proyecto apoyó muchos trabajos sobre diferentes problemas, como la calibración Heng et al. [7][8], estéreo Hane et al. [9], reconstrucción Haene et al. [10][11][12], SLAM Grimmett et al. [13] y detección de espacio libre Hane et al. [14]. Además de estos objetivos de investigación, el proyecto mantiene un fuerte enfoque en la implementación y valoración del sistema en entornos realistas.

Waymo (división de vehículos autónomos de Google) y Telsa son las compañía que actualmente se encuentran en la fase más avanzada del desarrollo de coches autónomos. Google comenzó su proyecto de automóvil autónomo en 2009 y completó más de 2.410.000 kilómetros de forma autónoma hasta marzo de 2016 en Mountain View, California, Austin, TX y Kirkland, WA. Diferentes sensores (por ejemplo, cámaras, radares, LiDAR, codificador de ruedas, GPS) permiten detectar peatones, ciclistas, vehículos, trabajos en carreteras y más, en todas las direcciones. De acuerdo con sus informes de accidentes, los coches autónomos de Google estuvieron involucrados solo en 14 colisiones, mientras que 13 veces fueron causados por otros. En 2016, el proyecto se dividió en Waymo [15], una empresa independiente de tecnología de conducción autónoma.

Tesla Autopilot [16] es un sistema avanzado de asistente de controladores desarrollado por Tesla que se lanzó por primera vez en 2015. El nivel de automatización del sistema permite la automatización total, pero requiere la plena atención del conductor para tomar el control si es necesario. A partir de octubre de 2016, todos los vehículos producidos por Tesla estaban equipados con ocho cámaras, doce sensores ultrasónicos y un radar orientado hacia adelante que otorgan una completa capacidad de autoconducción.

Existen numerosas pruebas y competiciones que fomentan la investigación y el desarrollo de nuevos sistemas software y hardware para la conducción autónoma. La Prueba Europea de Robots de Tierra (ELROB) [17] es una demostración y competencia de sistemas no tripulados en escenarios y terrenos realistas. A diferencia de los desafíos de conducción autónoma, los escenarios de ELROB suelen incluir navegación en terreno accidentado. La primera competencia de conducción autónoma enfocada en escenas de carreteras fue iniciada por la Agencia de Proyectos de Investigación Avanzada de Defensa Americana (DARPA) en 2004. El Grand Cooperative Driving Challenge (GCDC [18], véase también Geiger et al., (2012) [19]), una competencia centrada en el comportamiento de conducción cooperativa autónoma se celebró en Helmond, Países Bajos en 2011 por primera vez y en 2016 en una segunda edición. Ninguno de los vehículos ha conseguido completar aún todos los desafíos propuestos en las pruebas exitosamente.

En la conducción autónoma, es importante comprender tanto la información estructural como la semántica del entorno. Tradicionalmente, los métodos de segmentación de imágenes se centraban completamente en el dominio de imagen 2D. Se ha argumentado durante mucho tiempo que la semántica y la reconstrucción 3D se transmiten información valiosa entre sí. La reconstrucción en 3D eleva el razonamiento de 2D a 3D y actúa como un fuerte regularizador al imponer la consistencia geométrica sobre múltiples imágenes para la segmentación. La información de profundidad es crucial para aplicaciones en sistemas autónomos de conducción o asistencia al conductor. La estimación precisa de mapas de profundidad densa es un paso necesario para la reconstrucción en 3D, y muchos otros problemas, como la detección de obstáculos, el análisis del espacio libre y el seguimiento, se benefician de la disponibilidad de estimaciones de profundidad.

Desde la perspectiva de la aplicación, la representación de escenas es una forma común de clasificar los enfoques de reconstrucción 3D en mapas de profundidad, nubes de puntos, mallas y volumétricos. Las representaciones de superficies basadas en puntos o en nubes reconstruyen un único modelo de nube de puntos 3D utilizando todos los datos de entrada. Bajo supuestos de consistencia espacial, la nube de puntos en la superficie de la escena puede crecer o expandirse, lo que proporciona una manipulación fácil del modelo, como la fusión y la división.

La estimación del movimiento, la posición y la orientación del automóvil es otro problema fundamental para realizar la conducción autónoma. Este problema se puede enfocar desde dos puntos de vista, optical flow (es-

timación 2D) o scene flow (estimación 3D). Optical Flow, que se define como el movimiento bidimensional de los patrones de brillo entre dos imágenes, solo representa el movimiento de las intensidades en el plano de la imagen, pero no el movimiento 3D de los objetos en la escena. Scene Flow, tiene por objetivo estimar el campo de movimiento tridimensional, que es un vector de movimiento en 3D para cada punto en cada superficie visible en la escena. Tradicionalmente, estos problemas se abordaba en la odometría de las ruedas con codificadores de ruedas, que miden la rotación de la rueda, integrando las mediciones a lo largo del tiempo. Estos métodos sufren el deslizamiento de la rueda en terrenos desiguales o condiciones adversas y no pueden recuperarse de errores en las mediciones. Las técnicas de odometría visual o LiDAR basadas en odometría, que estiman el movimiento a partir de imágenes o mediciones de alcance láser, se popularizaron porque son menos afectadas por estas condiciones y puede corregir los errores de estimación mediante el reconocimiento de los lugares ya visitados que se denomina cierre de bucle [4].

La combinación de las tareas de reconstrucción 3D (mapeado), estimación del movimiento (ego-motion) y localización se denomina SLAM, un desafío particular en la conducción autónoma. El enfoque para resolver los problemas de SLAM es la coincidencia de escaneo, es decir, encontrar la superposición correcta de escaneos[1] a través de un proceso de traslación y rotación.

Existes 3 métodos de comparación de escaneos: Feature to Feature (F2F) Shaffer [20] Gonzalez et al. [21], Point to Feature (P2F) Cox [22] y Point to Point (P2P) Lu y Milios [23][24]. Los enfoques basados en F2F y P2F intentan utilizar menos información para representar los datos brutos a fin de acelerar los algoritmos, pudiendo reducir el rendimiento total. Por el contrario, los enfoques basados en P2P no tienen estas desventajas, en su lugar, usan todos los datos sin procesar [25]. Uno de los algoritmos P2P más exitosos y populares es el algoritmo ICP (abreviatura de Iterative Closest Point o Iterative Corresponding Point) [26]. La idea básica de ICP es usar una regla de punto más cercano para establecer correspondencias entre puntos en el escaneo actual y el escaneo previo, con una buena estimación inicial de su pose relativa, y luego resolver el punto a punto de mínimos cuadrados para calcular su posición relativa.

1.2. Objetivos y alcance

Los vehículos autónomos disponen de múltiples sensores para percibir el entorno a su alrededor. El objetivo de este TFM es la investigación y análisis de los datos 3D procedentes de un sensor LiDAR. Se abordarán tareas I+D relacionadas con los algoritmos de procesamiento que obtengan información útil de la escena con el objetivo final de habilitar funciones de asistencia a la conducción en vehículos potencialmente sin conductor.

En concreto se desarrollarán y evaluarán algoritmos que permitan la reconstrucción 3D del entorno recorrido por el vehículo y la estimación del movimiento y la posición del vehículo (odometría). Estos algoritmos pertenecen al campo de investigación de métodos SLAM (Simultaneous Localization and Mapping) para la localización relativa del vehículo y el mapeado del entorno 3D simultáneamente.

1.3. Metodología

En primer lugar se han realizado tareas de investigación, instalación y comprensión de las herramientas hardware y software disponibles y necesarias para el desarrollo del TFM.

El siguiente paso es la adquisición, análisis y procesamiento de datos procedentes del LiDAR para poder realizar I+D en algoritmos para la percepción de la escena 3D desde un vehículo. Para ello, se ha realizado un estudio de mercado de la tecnología LiDAR y se ha profundizado en el desarrollo de algoritmos para registrar nubes de puntos 3D y mapear el entorno recorrido por el vehículo.

Con el estudio de mercado, se procede a la elección del algoritmo más adecuado para la extracción de información semántica de interés. En concreto, en este TFM se investiga la adquisición y procesamiento de datos 3D del LiDAR para realizar reconstrucción 3D (mapeado de la escena) y odometría (representación del movimiento del vehículo en un plano 2D), como una primera aproximación a las técnicas de SLAM (Simultaneous Localization and Mapping), por sus siglas en inglés localización y mapeado simultáneos.

Una vez desarrollado el programa se pasa a una fase de experimentación y análisis de resultados, se van aplicando

[1] Datos recolectados por el dispositivo en cada instantánea.

métodos de optimización para mejorar el rendimiento del algoritmo y se evalúan los resultados obtenidos. Finalmente se concluye con los algoritmos y configuraciones que permiten conseguir los objetivos junto con una revisión de las bondades y limitaciones de los mismos.

1.4. Herramientas

1.4.1. Hardware

LiDAR: "Light Detection and Ranging" o "Laser Imaging Detection and Ranging", es un dispositivo que permite determinar la distancia desde un emisor láser a un objeto o superficie utilizando un haz láser pulsado. La distancia al objeto se determina midiendo el tiempo de retraso entre la emisión del pulso y su detección a través de la señal reflejada [27]. En este TFM se ha empleado un Velodyne HDL-32E [28].

Ordenador: El ordenador utilizado tiene un procesador Intel Core i5-3330 CPU @ 3.00GHz x 4, con una memoria RAM de 8.00 GB y un sistema operativo de 64-bit.

1.4.2. Software

Ubuntu: Es un sistema operativo de código abierto para computadores. Es una distribución de Linux basada en la arquitectura de Debian [29]. Para este TFM se ha utilizado la versión Ubuntu 16.04.4 LTS.

OpenCV: *Open Computer Vision* (por sus siglas en inglés) es una biblioteca de código abierto que ofrece funcionalidades de visión artificial y con licencia BSD, que permite ser usada libremente para propósitos comerciales y de investigación [30].

PCL: *Point Cloud Library* (por sus siglas en inglés) es una biblioteca de código abierto con funcionalidades de procesamiento de nubes de puntos 3D y procesamiento de geometría 3D. La biblioteca contiene algoritmos para la estimación de características, la reconstrucción de superficies, el registro en 3D, el ajuste del modelo y la segmentación. Está escrito en C++ y publicado bajo la licencia BSD [31].

CMake: Es una herramienta multiplataforma de generación o automatización de código. En particular, CMake es una familia de herramientas diseñada para construir, probar y empaquetar software. Se utiliza para controlar el proceso de compilación del software usando ficheros de configuración sencillos e independientes de la plataforma. CMake genera makefiles nativos y espacios de trabajo para usarse en el entorno de desarrollo deseado [32].

Qt Creator: Es un entorno de desarrollo integrado (IDE) para crear aplicaciones C++ y QML para múltiples plataformas [33]. Este entorno se ha usado junto con las librerías OpenCV y PCL para desarrollar el programa.

1.4.3. Bases de Datos

Vicomtech: Es un centro de investigación aplicada especializado en las tecnologías de Advanced Interaction, Computer Vision, Data Analytics, Computer Graphics y Language Technologies (Sección 1.5). El centro, donde he realizado mis prácticas y TFM, dispone de numerosos medios y sensores para extraer información del entorno. Estas herramientas e información se encuentran a disposición de los empleados para la investigación. Para este TFM, se han usado varias grabaciones realizadas con un vehículo equipado con un LiDAR Velodyne HDL-32E entorno a la localización de Vicomtech (Parque Miramón, San Sebastián).

KITTI: Es una plataforma de conducción autónoma que alberga numerosas bases de datos y puntos de referencia para la investigación. Esta plataforma cuenta con un coche equipado con un sensor LiDAR Velodyne, un sistema de navegación inercial GPS/IMU de alta precisión y un equipo de cámaras estéreo de alta resolución en color y escala de grises. El coche recorre ciudades, zonas rurales y autopistas obteniendo información del entorno con los sensores equipados. Los datos recolectados los ponen a disposición para que cualquier investigador pueda usarlos y obtener resultados y progresos en el mundo de la conducción autónoma, sin necesidad de disponer de los sensores [34]. Para la realización de este TFM, se han usado principalmente los datos de LiDAR, GPS, calibraciones y odometría.

1.5. Contexto

El desarrollo de este TFM se ha llevado a cabo en la empresa Fundación Vicomtech bajo la supervisión del investigador Dr. José Javier Yebes Torres.

Vicomtech es un centro de investigación aplicada especializado en las tecnologías de Advanced Interaction, Computer Vision, Data Analytics, Computer Graphics y Language Technologies [35].

Vicomtech tiene por objeto responder a las necesidades de innovación de las empresas e instituciones. Para ello:

- Realiza investigación aplicada y desarrolla tecnologías multimedia de interacción visual y comunicaciones.
- Colabora estrechamente con la industria, la universidad y otros centros tecnológicos, a quienes complementa.
- Fomenta la movilidad y formación de sus investigadores.

De este modo, los conocimientos y tecnologías que domina directamente o a través de la red aportan valor a sus clientes, ya que Vicomtech ofrece la respuesta adecuada a las necesidades de sus clientes, facilita a las empresas el aprovechamiento de las oportunidades que puedan surgir y propone mejoras o desarrollos para los productos basados en los últimos avances del conocimiento científico y tecnológico.

Concretamente, el TFM se ha realizado en el departamento de Sistemas de transporte inteligentes e Ingeniería. Este departamento se encarga de la aplicación de las tecnologías de Visión Artificial, Sistemas de Visualización Avanzada, Ingeniería de Conocimiento y Algorítmica Avanzada al sector industrial en general y al sector de transporte en particular, aportando soluciones tecnológicas (ITS) a las empresas del sector [36]. Las actividades principales del departamento son:

- I+D en Visión Artificial, Sistemas de Visualización Avanzada, Ingeniería de Conocimiento y Algorítmica Avanzada.
- Ofrecer soluciones tecnológicas a medida para empresas, particulares y administraciones.
- Mejorar la efectividad de las soluciones, productos o tareas a través de la aplicación real de los resultados del I+D como soluciones inteligentes.

1.6. Contenido de la memoria

Este trabajo esta dividido en 5 capítulos: introducción, tecnología LiDAR y datos 3D, procesamiento y análisis 3D de escenas de carretera, pruebas y análisis de resultados y conclusiones.

En la introducción se hace un breve repaso de la historia y el estado del arte de los vehículos autónomos y sus sensores. Se detallan los objetivos y alcance del proyecto, la metodología seguida para lograrlo y las herramienta software y hardware empleadas. Por último se hace una breve presentación de la Fundación Vicomtech, empresa en donde se ha realizado el proyecto.

El capítulo de tecnología LiDAR y datos 3D comienza con una breve introducción a la historia, las capacidades y los usos de esta tecnología, para seguir con un estudio de mercado de los dispositivos LiDAR más usados en los vehículos autónomos. Se detallan las características y forma de grabación del Velodyne LiDAR 32, dispositivo principal utilizado en este trabajo y se describen los diferentes software para obtener la información del sensor, así como la ubicación del LiDAR en el vehículo para extraer la información del entorno.

El siguiente capítulo, procesamiento y análisis 3D de escenas de carretera, se centra en las habilidades del programa desarrollado para realizar dicha tarea. Estas habilidades son la reconstrucción 3D de la escena, la reconstrucción del recorrido del vehículo en el plano suelo y la composición de imágenes panorámicas. En la primera sección se hace una introducción a los métodos y las utilidades del análisis de escenas de carretera y se describen las tareas realizadas para conseguir la reconstrucción 3D en este trabajo, así como los algoritmos disponibles y utilizados para reconstrucción 3D en los que se ha apoyado este proyecto. En las otras dos secciones se describen el procedimiento empleado para lograr realizar cada tarea.

En el penúltimo capítulo, pruebas y resultados, se exponen todos los métodos, pruebas y resultados obtenidos, los criterios de elección de los métodos/algoritmos más efectivos para abordar las tareas de este proyecto y se muestran cuantitativamente las capacidades de los algoritmos y configuraciones desarrolladas.

Por último se exponen las conclusiones obtenidas en este TFM y se plantean posibles líneas de investigación futuras.

Capítulo 2

Tecnología LiDAR y datos 3D

2.1. LiDAR

Por sus siglas en inglés, se puede definir como *"Light Detection and Ranging"* o *"Laser Imaging Detection and Ranging"*. El LiDAR es un sistema activo de detección remota que permite obtener las distancias desde un emisor a objetos, obstáculos y superficies en general de un determinado entorno. Mediante la calibración del sistema, dichas distancias pueden transformarse a coordenadas cartesianas 3D, de tal modo que se obtienen un conjunto de puntos tridimensionales de la escena o entorno observado por el LiDAR. Este sensor cuenta con uno o varios emisores de rayos láser y uno o varios receptores. Los láseres utilizados suelen emitir miles de pulsos por segundo. El tiempo de viaje de estos pulsos se mide y se graba para determinar la distancia, obteniendo una nube de puntos 3D del terreno [37].

En el contexto de los vehículos autónomos, este dispositivo a menudo se combina con otros equipos de medición, generalmente cámaras, sensores IMU (inertial measurement unit) y GNSS (Global Navigation Satelite System) para fusionar distintas fuentes de datos y realizar funciones de asistencia a la conducción.

Una de las principales características del LiDAR es que los datos se pueden recopilar, con bastante rapidez (de 5 a 20 Hz), ante cualquier tipo de condiciones climatológicas, si bien hay que añadir que con lluvia o nieve fuerte funciona peor porque el láser rebota y/o se difracta sin traspasar correctamente y no detecta bien los objetos, carretera, edificios. Este problema puede mitigarse usando LiDARs con una mayor número de rayos. Aún así, el uso de LIDAR en las primeras etapas de mapeo, antes de que sea necesaria una información de terreno altamente precisa, puede proporcionar ahorros de tiempo y costos tangibles. Diferentes estudios en proyectos corroboraron esta información, pero la conclusión extraída de todos estos proyectos fue que el LiDAR no puede reemplazar completamente al resto de métodos de mapeo [38].

La tecnología LIDAR fue utilizada por primera vez en la década de 1960 para instrumentos de teledetección. Desde su creación, se desarrolló lentamente durante 30 años, utilizado por la NASA para el estudio atmosférico y oceanográfico, la exploración espacial, el armamento y el mapeo topográfico. En la década de los 1990 se comenzó a utilizar para crear mapas tridimensionales precisos de terrenos, ya que los sistemas basados en fotografía y los sistemas de radar no eran ideales [39]. Pero no fue hasta 2005 cuando se introdujo en la conducción autónoma para competir en el "Grand Challenge"de la Agencia de Proyectos Avanzados de Investigación de Defensa (DARPA), un campeonato para estimular la innovación tecnológica en el campo de los vehículos terrestres no tripulados. Desde ese momento, el uso de esta tecnología fue incrementándose hasta hacerse indispensable cuando se habla de vehículos autónomos. El LiDAR no es un dispositivo cuya única utilidad es mapear el terreno (tema que se trata en este TFM), si no que también recopila suficientes datos del entorno para que las máquinas autónomas puedan identificar y reaccionar ante los obstáculos y las amenazas con tiempo suficiente [40].

En definitiva, las cualidades más significativas del LiDAR son: la capacidad que posee para recolectar gran cantidad de datos del entorno con información de profundidad (distancias) y con una precisión que varía de 2cm a 10cm (según marca y modelo) y el funcionamiento bajo cualquier condición climatológica (noche, lluvia, niebla y nieve).

Para el desarrollo de este trabajo solo se ha utilizado el sensor LiDAR, dejando la integración de otros sensores para trabajos futuros. El LiDAR proporciona nubes de puntos 3D del entorno que se utilizan para el mapeado

(reconstrucción de la escena) y la odometría (estimación y representación del movimiento de un vehículo).

2.2. Dispositivos LiDAR para vehículos

Existen numerosos dispositivos LiDAR en el mercado, en la Tabla 2.1 se detallan los modelos y las características de los LiDARs más utilizados para la conducción autónoma. Hay 4 compañías que destacan sobre el resto, ya sea por su posicionamiento en el sector o por el crecimiento que están teniendo, estas marcas son Velodyne, Ibeo, Robosense y Ouster.

La marca Velodyne [41] es la marca referencia mundial en cuanto a LiDARs se refiere si hablamos de conducción autónoma. Alcanzó la fama hace unos 10 años al ser la primera compañía en desarrollar un modelo de LiDAR capaz de disparar 64 rayos láser en la vertical, modelo que utilizan la mayoría de coches autónomos. Estos modelos tienen un amplio campo visual vertical y con mucha resolución. Además, Velodyne dispone de numerosos dispositivos para adaptarse mejor a las necesidades del cliente.

Ibeo [42] es un fabricante de LiDAR a nivel Europeo cuyo principal mercado son los sistemas de asistencia de la conducción y vehículos autónomos. Sus productos ofrecen menor resolución vertical (4-8 haces láser) comparado con Velodyne. Por el contrario, ofrecen sistemas completos con funcionalidades de detección automática de vehículos y peatones y la estimación de distancia a los mismos.

Robosense [43] es la competencia directa de Velodyne. Aunque esta marca lleva menos tiempo en el mercado las características de sus productos LiDAR son similares pero a un precio más accesible. Además, está realizando una fuerte inversión en las nuevas tecnologías LiDAR SSL (Solid State Lidar) y OPAL (Optical Phased Array Lidar). Esta compañía también ofrece, junto con el LiDAR, un software para la conducción autónoma, capaz de detectar carriles, detectar obstáculos, realizar seguimientos, etc.

El caso de Ouster [44] es parecido al de Robosense, lleva menos años en el mercado y está intentando hacerse un hueco. La característica que diferencia al producto de esta compañía es que el modelo de 64 rayos es más pequeño y compacto que el resto de modelos de 64 rayos del mercado. También ofrece dispositivos mas económicos que Velodyne.

Para el objetivo planteado en este TFM de odometría y mapeado de escenas, se necesita un LiDAR con alta resolución vertical y horizontal de tal modo que se obtengan múltiples puntos 3D del entorno por el que circula el vehículo. En este sentido Velodyne, Robosense y Ouster son adecuados. Ibeo[42] y Quanergy [45] no tienen muchos canales, por lo que la cantidad de información que pueden obtener del terreno es menor, aunque si puedan usarse para otras funciones de detección de obstáculos y/o objetos. En el futuro, la nueva tecnología LiDAR SSL y OPAL permitirá mayores resoluciones a un menor tamaño físico y a un menor coste (< $500) equiparando la resolución 3D del LiDAR a la resolución 2D de las cámaras.

LiDAR	**Rayos**	**FOV**[1] **vertical**	**FOV horizontal**	**Resolución angular vertical**	**Resolución angular horizontal**	**Precisión**	**Alcance**
Velodyne VLP-16 Puck	16	30º	360º	2.00º	0.1º a 0.4º	±3cm	100m
Velodyne VLP-16 Puck Hi-Res	16	20º	360º	1.33º	0.1º a 0.4º	±3cm	100m
Velodyne VLP-16 Puck LITE	16	30º	360º	2.00º	0.1º a 0.4º	±3cm	100m
Velodyne ULTRA Puck VLP-32C	32	40º	360º	0.33º	0.1º a 0.4º	±3cm	200m
Velodyne HDL-32E	32	40º	360º	1.25º	0.08º a 0.35º	±2cm	100m
Velodyne HDL-64E	64	26.8[a]	360º	0.4º	0.08º a 0.35º	±2cm	120m
ibeo Standard 8 Layer	8	6.4[a]	110º	0.8º	0.25º	±10cm	50m
ibeo Standard 4 Layer	4	3.2º	110º	0.8º	0.25º	±10cm	50m
ibeo Heavy Duty 4 Layer	4	3.2º	110º	0.8º	0.25º	±10cm	30m
ibeo Wide Angle Scanning	4	3.2º	145º	0.8º	0.25º	±10cm	150m
Robosense RS-LIDAR-16	16	30º	360º	2.00º	0.09º a 0.36º	±2cm	150m
Robosense RS-LIDAR-32A	32	30º	360º	0.9º	0.09º a 0.36º	±2cm	200m
Robosense RS-LIDAR-32B	32	40º	360º	0.33º	0.09º a 0.36º	±2cm	200m
Ouster OS-1-16	16	31.6º	360º	0.52º	0.09º	±3cm	120m
Ouster OS-1-64	64	31.6º	360º	0.52º	0.09º	±3cm	120m
Quanergy M8	8	20º	360º	2.50º	0.03º a 0.20º	±3cm	150m
Quanergy S3	-	120º	120º	-	-	±5cm	150m

Tabla 2.1: Modelos LiDAR más utilizados del mercado.

2.3. Velodyne LiDAR HDL-32E

El dispositivo empleado en este TFM es el Velodyne LiDAR HDL-32E (Figura 2.1). Este dispositivo crea imágenes 3D de 360º mediante el uso de 32 de láseres verticales. La carcasa superior del LiDAR gira para obtener la información del entorno circundante, proporcionando una nube de puntos 3D.

Figura 2.1: Velodyne LiDAR HDL-32E.

El procesamiento de la señal digital y el análisis de forma de onda proporcionan información sobre la distancia e intensidad de cada punto 3D. El HDL-32E utiliza un sistema de motor de transmisión directa, que no emplea correas o cadenas en el tren de transmisión [46].

Velodyne LiDAR HDL-32E tiene las siguientes propiedades [28]:

- 32 canales.
- Rango de alcance de 80-100 metros.
- Capaz de recolectar hasta 1.39 millones de puntos por segundo.
- Campo de visión horizontal de 360º y vertical de +10º a -30º (40º).
- Resolución vertical angular de 1.25º y horizontal de 0.08º a 0.35º (dependiendo de la frecuencia con la que gire 5-20 Hz).

Este dispositivo genera paquetes de datos UDP que son transmitidos a través de Ethernet (Figura 2.2). Cada paquete contiene un total de 1248 bytes, los 42 primeros corresponden al encabezado de la carga útil de la trama Ethernet, los 6 últimos bytes corresponden a la marca de tiempo GPS y el resto se dividen en 12 registros de 100 bytes (ver Figura 2.3). Cada registro de 100 bytes contiene primero un identificador de inicio, luego un valor de rotación de dos bytes, seguido de 32 combinaciones de 3 bytes que informan sobre cada láser disparado. Dos bytes informan de la distancia a los 0.2 cm más cercanos, y el byte restante informa la intensidad en una escala de 0 - 255 [46].

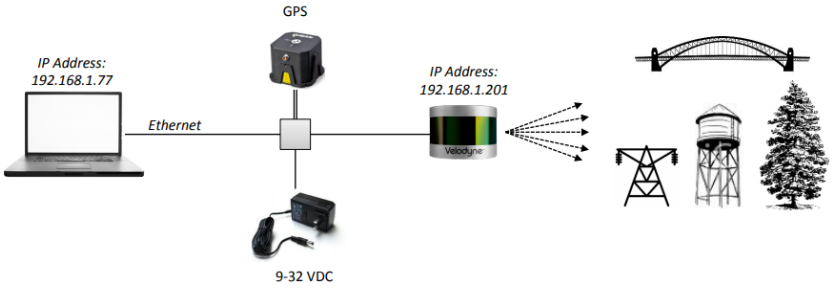


Figura 2.2: Transmisión de datos con Velodyne.[2]

[1]FOV: Campo de visión o, por su siglas en inglés, *Field of View*.
[2]Fuente: `https://velodynelidar.com`

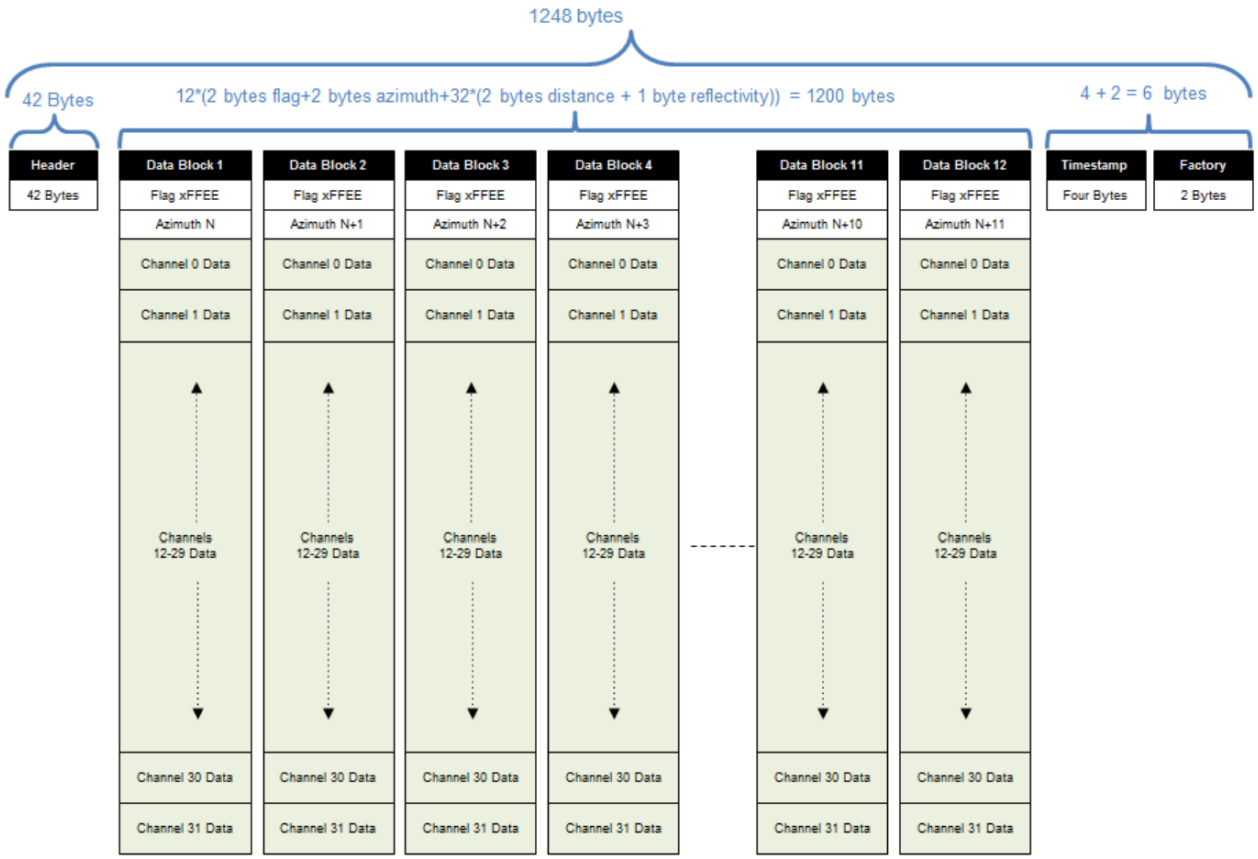


Figura 2.3: Estructura de datos Velodyne LiDAR HDL-32E.[3]

La secuencia de disparo láser se muestra en la Tabla 2.2, de los 32 láseres verticales que tiene este LiDAR, el láser que se encuentra con el menor ángulo, se dispara el primero, seguido del que se encuentra a mitad, este intercalado se repite durante todo el proceso.

Canal	**Ángulo vertical**
1	-30.67
2	-9.33
3	-29.33
4	-8.00
5	-28.00
6	-6.66
.	.
.	.
.	.
28	8.00
29	-12.00
30	9.33
31	-10.67
32	10.67

Tabla 2.2: Secuencia de disparo láser del Velodyne LiDAR HDL-32.

[3]Fuente: `https://velodynelidar.com`

2.4. Adquisición de datos

Este TFM ha explorado tres opciones para adquirir y formatear los datos recolectados por el LiDAR.

Opción A. La primera de ellas es utilizar Veloview, una aplicación de código abierto diseñada por Kitware and Velodyne [47] para visualización 3D en tiempo real y análisis de nubes de puntos 3D de los datos capturados de los sensores LiDAR HDL de Velodyne. VeloView incluye un motor de scripts en Python, mediante el cual, los usuarios pueden acceder a datos y atributos de las nubes de puntos y usar NumPy para realizar análisis de datos avanzados [48]. Veloview graba en formato binario como archivo PCAP y es capaz de exportar los datos en formato CSV (Comma Separated Values) o VTK (Visualization Tool Kit).

Opción B. Otra opción es utilizar la plataforma de computación abierta para automóviles con inteligencia artificial NVIDIA DRIVE PX2 [49]. Esta plataforma, junto con el kit de desarrollo de software para la conducción autónoma SDK Nvidia Driveworks [50], es capaz de obtener los datos recolectados por el LiDAR. Con este método se puede grabar directamente con el LiDAR y obtener los datos por defecto en formato binario en un archivo BIN, aunque también existe la posibilidad de guardar en otros formatos (CSV, TXT...).

Opción C. La tercera opción es emplear RTMaps (Real-Time Multisensor Applications), un software orientado a la grabación, sincronización y reproducción de datos obtenidos de diferentes sensores [51]. Los datos del LiDAR se almacenan en formato *raw stream 8*, esto es, se guardan como secuencia de datos binarios de 8bits tal y como llegan por el puerto UDP desde el LiDAR. Con este método también se pueden convertir los datos del LiDAR a formato CSV.

Para este trabajo se va a utilizar RTMaps. El motivo de su elección es que este software es usado habitualmente dentro del departamento de Vicomtech donde se está realizando el proyecto. Esto se debe a que es capaz de poner una marca de tiempo (timestamp) a los datos de entrada y posteriormente reproducirlos en modo "timestamp", facilitando así la sincronización con otros sensores.

Además, por conveniencia durante el desarrollo de este TFM y para la legibilidad de los datos, estos se decodificaron en archivos CSV. A parte de los archivos que RTMaps crea por defecto ante cualquier grabación (ficheros de sincronización, índice e información), los datos obtenidos por el LiDAR del entorno se decodificaron en 4 archivos. En el primero están los datos de las coordenadas 3D de cada punto capturado por el LiDAR, en otro la distancia euclídea a cada punto, en el tercero, el valor de la intensidad de cada punto y en el último el ángulo en el eje X e Y (plano suelo) de cada disparo (en la Figura 2.4 se muestra el eje de coordenadas del LiDAR). Cada fichero almacena en un vector de 1 fila los datos leídos por el LiDAR en cada vuelta (360º), en donde el primer valor corresponde a la marca de tiempo en milisegundos.

Como se introdujo en la Sección 1.4.3, se han usado dos fuentes de datos 3D. La primera han sido las grabaciones con los recursos de Vicomtech, que cuenta con un vehículo para la investigación en la conducción autónoma. Este vehículo es un coche equipado con diferentes sensores capaces de recopilar información sobre el terreno. Gran parte de las pruebas, resultados y conclusiones de este trabajo se obtuvieron de los datos recopilados por el LiDAR Velodyne HDL-32E integrado en dicho coche. La segunda fuente es KITTI [34]. Esta base de datos pública para investigación cuenta con numerosas secuencias disponibles con datos de múltiples sensores (LiDAR, cámaras, GPS, IMU), calibraciones de los mismos y anotaciones semánticas (objetos, odometría, etc.). Además, a diferencia de los datos recolectados con el coche de Vicomtech, las secuencias de KITTI cuentan con el ground truth para comprobar la fiabilidad del algoritmo creado. A parte de esto, KITTI utiliza el LiDAR Velodyne HDL-64E que, en vez de 32 rayos, lanza 64 rayos en la vertical, por lo que tiene un campo de visión vertical mayor y por ende una mejor resolución.

2.4.1. Ubicación en el vehículo

Para realizar el análisis de las nubes de puntos 3D, el mapeado/reconstrucción de la escenas y la estimación del movimiento del vehículo, es necesario definir claramente los sistemas de coordenadas utilizados. Esta sección describe la ubicación de los sensores en los vehículos de Vicomtech y KITTI.

Tanto en el coche de Vicomtech (Figura 2.4a) como en el coche de KITTI (Figura 2.4b), el LiDAR se encuentra ubicado en la parte exterior de techo del coche, centrado y unos centímetros por encima del techo, para aumentar el rango de visión. La orientación de su eje de coordenadas (X, Y, Z) es idéntica en ambos caso, tal como se observa en las imágenes. X en dirección longitudinal del movimiento del vehículo, Z perpendicular al plano suelo apuntando hacia arriba e Y perpendicular al plano XZ.

Aunque ambos LiDARs están ubicados de forma similar, el ground truth de KITTI esta realizado desde la cámara estéreo delantera izquierda con el eje de coordenadas de dicha cámara (ver Figura 2.5), por lo que para comparar los datos o bien se transforman los datos obtenidos desde el LiDAR a la cámara o el ground truth se lleva al LiDAR, para que ambos conjuntos tengan la misma referencia. En este trabajo se utiliza la primera opción, los datos obtenidos por el LiDAR se transforman para que su origen y eje de coordenadas sea el mismo que el de la cámara (ground truth).

(a) Coche de Vicomtech

(b) Coche de KITTI[4]

Figura 2.4: Ubicación del LiDAR en el vehículo.

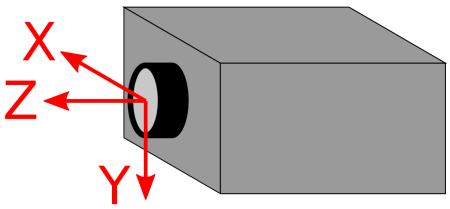


Figura 2.5: Ejes de coordenadas de las cámaras estéreo de KITTI.

[4]Fuente: `http://www.cvlibs.net/datasets/kitti/`

Capítulo 3

Procesamiento y análisis 3D de escenas de carretera

3.1. Reconstrucción 3D de la escena

Los datos 3D adquiridos mediante el LiDAR Velodyne permiten obtener una nube de puntos 3D en 360º en torno al vehículo que lleva incorporado dicho sensor. Esta nube podría asimilarse a una fotografía de baja resolución de los alrededores de vehículo donde los píxeles, en vez de ser colores, son valores de profundidad/distancia desde el origen de coordenadas del sensor hasta los obstáculos de la escena donde rebota la señal láser. Dicha instantánea[1]representa una pequeña parte del entorno recorrido por el vehículo. Mediante la reconstrucción 3D de la escena se pretende acumular una secuencia de instantáneas, es decir, un gran número de medidas 3D tomadas durante la conducción del vehículo por una carretera o zona determinada. La fusión, concatenación o unión de las nubes de puntos no es una cuestión trivial y se requieren algoritmos avanzados que permitan comparar dos nubes consecutivas, estimar su movimiento relativo en 3D y transformar las mismas para situar los puntos 3D respecto a un único origen de coordenadas.

A continuación se realizará una introducción a las técnicas de localización y mapeado. Se trata de un campo de investigación de elevado interés por su aplicación a los vehículos autónomos aunque llevan aplicándose al área de robótica en los últimos 20 años.

3.1.1. SLAM: Localización y mapeado

La localización simultánea y mapeado (SLAM - Simultaneous Localization and Mapping) es la capacidad que tiene cualquier máquina de crear simultáneamente un mapa del entorno y ser capaz de localizarse a si mismo en dicho entorno, lo cual, es un gran desafío en la conducción autónoma.

Un mapa detallado del entorno es un requisito necesario para la planificación de rutas y la navegación de un vehículo autónomo. Sin embargo, en lugares donde no se proporciona un mapa o este está incompleto, el vehículo autónomo necesita ubicarse mientras se genera el mapa. Además, el mapa debe actualizarse continuamente para reflejar los cambios ambientales a lo largo del tiempo [4].

La localización, desde una perspectiva de conducción autónoma, es la tarea de localizar con precisión el vehículo en un mapa. También se utiliza para detectar cierres de bucles y corregir la desviación cuando se mapea el entorno. Habitualmente la localización solo se basa en sistemas de posicionamiento por satélite GNSS (ej. GPS, GALILEO, GLONASS) y mapas 2D estáticos. Sin embargo, la localización se puede realizar combinando diferentes sensores (GNSS, sensores inerciales, cámaras, LiDARs) y un conjuntos técnicas de análisis y fusión de los datos. Aunque la precisión a nivel de centímetro es posible en espacios abiertos usando combinaciones de sensores, a menudo se vuelve inviable en entornos urbanos debido a efectos perturbadores como oclusiones por vegetación y edificios [4].

[1]Para esta descripción se asume un tiempo infinitesimal en la rotación 360º y captura de los datos 3D. En la realidad esto no es así y se requieren técnicas de compensación del movimiento del vehículo.

El mapeado del terreno es otra parte muy importante en la conducción autónoma, ya que esta habilidad del vehículo no solo es capaz de generar mapas 3D del entorno, si no que también ayuda a la adaptación a cambios dinámicos y a la localización del vehículo en el terreno.

La odometría, que consiste en la estimación del movimiento (ego-motion), la posición y la orientación del automóvil, es otro problema fundamental para la conducción autónoma, ya que sin esto es imposible realizar el SLAM. Los sensores GPS/INS o codificadores de ruedas son los más utilizados para realizar la odometría, pero estos acumulan errores, los primeros debido a la oclusión y los segundos por el deslizamiento de las ruedas. La odometría visual o las técnicas de odometría basadas en LiDAR, que estiman el movimiento a partir de imágenes o mediciones del láser, se ven menos afectadas por estas condiciones. Hasta el momento, la estimación de movimiento más precisa se logra utilizando información 3D [4].

Aunque los métodos basados en odometría visual muestran resultados competitivos, debido a que estos métodos usan cámaras (sensores actualmente más económicos que el LiDAR), los errores en el espacio 3D son mayores. Por ello, los escáneres láser y LiDAR proporcionan una fuente de información más rica y precisa para la estimación del movimiento. Los mejores métodos de odometría basada en LiDAR son el uso de nubes de puntos para estimación del movimiento [4].

Para obtener la transformación entre nubes de puntos, el algoritmo introducido por Besl y McKay en 1994, Iterative Closest Point (ICP) [26], se ha convertido en el método más utilizado para para la alineación de nubes de puntos en la conducción autónoma. Este método ha sido extensamente estudiado y se han propuesto muchas variantes para mejorar la precisión y reducir el tiempo de computo (Sección 3.1.3).

3.1.2. Reconstrucción

Uno de los objetivos de este TFM es realizar el mapeado del terreno que recorre el vehículo, para hacer dicha reconstrucción 3D se ha creado una nube de puntos global que almacena todas las nubes de puntos recolectadas por el LiDAR. Cada nube de puntos representa los datos recolectados por el LiDAR al dar una vuelta (360º) pero estas nubes tiene como referencia (origen de coordenadas) la ubicación del LiDAR en el instante que se recolectaron los datos. Como referencia global (referencia de la nube de puntos global con la que se realiza el mapeado) se ha elegido la ubicación del LiDAR en el primer instante de grabación, pero, como el vehículo se va moviendo, la referencia local (LiDAR) de cada nube de puntos no coincide con la referencia global (salvo la primera nube de puntos). Por este motivo es necesario transformar cada nube de puntos de la referencia local a la referencia global, para ello, el algoritmo ICP (Sección 3.1.3) es capaz de obtener la matriz de transformación necesaria para alinear dos nubes de puntos. Así, se obtiene la transformación entre la nube actual y la anterior y multiplicando esta matriz por las obtenidas en instantes anteriores, se obtiene una matriz capaz de llevar a la nube de puntos actual a la referencia global.

En la ecuación 3.1 se detalla el procedimiento para poder llevar la nube de puntos obtenida por el LiDAR a la referencia global.

$$P' = M_{\text{acumulada}} * P \qquad M_{acumulada} = M_{actual} * M_{acumulada} \tag{3.1}$$

Con el algoritmo ICP se obtiene la matriz de transformación (M_{actual}) entre dos nubes de puntos, esta matriz se multiplica por las matrices obtenidas con anterioridad para obtener la matriz ($M_{acumulada}$) capaz de llevar la nube de puntos (P) a la referencia global (P'). Así es posible que todas las nubes de puntos estén referenciadas al mismo sistema de coordenadas, cuyo origen es en este caso el inicio de la secuencia. Como se trabaja en un espacio euclídeo de 3 dimensiones, se utilizan coordenadas homogéneas.

La matriz de transformación (M_{actual}) está compuesta por las matrices de rotación (R_{zyx}) y traslación (T_{xyz}) (Ecuación 3.2). A su vez la matriz de rotación está formada por las rotaciones de los ángulos RPY (Figura 3.1 y Ecuación 3.3) y la matriz de traslación representa el movimiento lineal en cada eje de coordenadas (Ecuación 3.4).

$$M_{actual} = R_{zyx} * T_{xyz} = R_z(\phi) * R_y(\theta) * R_x(\psi) * T_{xyz} \tag{3.2}$$

$$R_z(\phi) = \begin{bmatrix} \cos\phi & -\operatorname{sen}\phi & 0 & 0 \\ \operatorname{sen}\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \operatorname{sen}\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\operatorname{sen}\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_x(\psi) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\psi & -\operatorname{sen}\psi & 0 \\ 0 & \operatorname{sen}\psi & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$T_{xyz} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

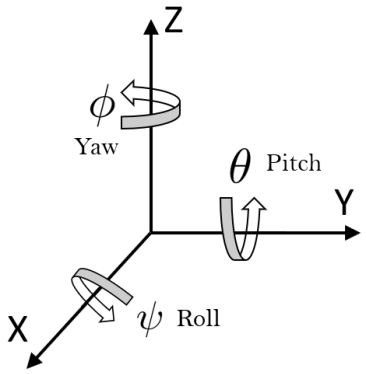


Figura 3.1: Ángulos RPY de rotación.

En la Figura 3.2 se muestran una serie de imágenes resultado de hacer la reconstrucción 3D. Los detalles y características de la representación del mapeado se describen en la Sección 4.4.

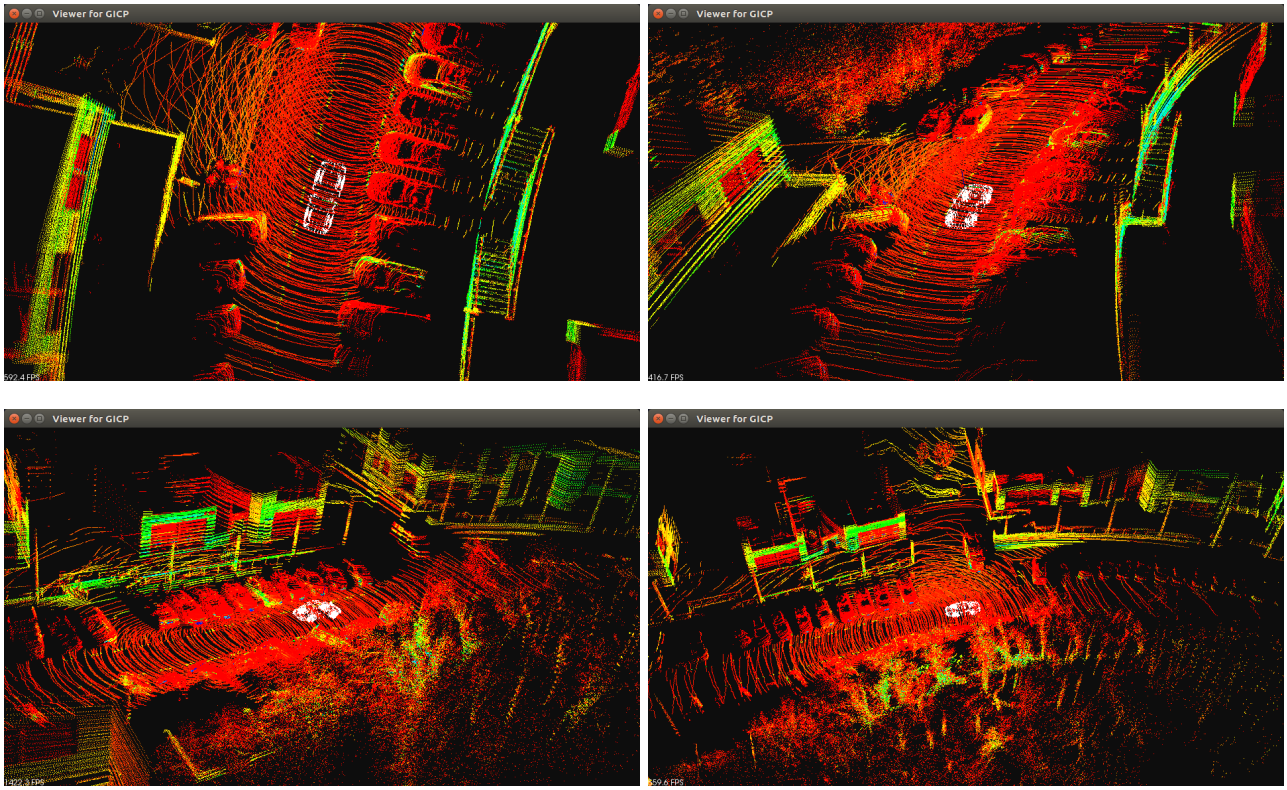


Figura 3.2: Ilustraciones en color[2] de la reconstrucción 3D del entorno.

[2]Mapa de color (Figura 4.9) con los valores de las intensidades de cada punto.

3.1.3. Algoritmo ICP y sus variantes

Como se ha detallado en el capítulo anterior, uno de los objetivos de este trabajo es realizar una reconstrucción 3D del entorno lo más precisa y en el menor tiempo posible, para ello es necesario obtener las matrices de transformación que alineen las nubes de puntos.

Para obtener dichas matrices, se han evaluado el algoritmo ICP y dos variantes que están implementadas en la librería PCL: Iterative Closest Point, Joint Iterative Closest Point y Generalized Iterative Closest Point. Se ha evaluado el rendimiento e idoneidad de cada algoritmo para la reconstrucción 3D.

El primer algoritmo, Iterative Closest Point (ICP), fue introducido por Besl and McKay [52] y posteriormente adaptado por Zhang [26] para la navegación de vehículos autónomos. Es uno de los primeros métodos desarrollados para encontrar la correspondencia entre dos escaneos. Este algoritmo minimiza la distancia euclídea entre los puntos vecinos más cercanos de dos escaneos para encontrar la transformación entre ellos [53]. Se necesita de un buen punto de partida para obtener la matriz de transformación o de lo contrario el algoritmo quedará atrapado en el primer mínimo local y la solución no será válida.

El segundo algoritmo, Joint Iterative Closest Point (JICP), es una extensión de ICP. La utilidad de este algoritmo radica en el uso de múltiples observaciones, es decir, varias nubes de puntos. Si por el contrario solo se utilizan 2 nubes de puntos, como es el caso estudiado, el algoritmo JICP se comporta igual que el ICP [54].

El tercer algoritmo propuesto es el Generalized Iterative Closest Point (GICP), propuesto por Segal et al. [55]. La diferencia de este algoritmo con el ICP es que en vez de minimizar la distancia euclídea entre los puntos, utiliza una métrica del error que generaliza sobre punto a punto y punto a plano. Esto se realiza utilizando covarianzas de los puntos locales vecinos para alinear las superficies, en lugar de los puntos en sí mismos [56]. Se emplea el método MLE (Maximum likelihood estimation) para estimar iterativamente la transformación y alinear las nubes de puntos. La finalidad de este algoritmo es mejorar la precisión intentando mantener la simplicidad y velocidad del ICP.

En la Sección 4.1 se hacen pruebas y análisis y se elige el algoritmo más idóneo para abordar las tareas de reconstrucción 3D.

3.2. Reconstrucción del recorrido del vehículo en el plano suelo

Otro de los objetivos de este trabajo es realizar la odometría (Sección 3.1.1) únicamente con los datos 3D obtenidos por el LiDAR. El propósito es representar en una imagen 2D la trayectoria del vehículo sobre el plano suelo, para está representación se han estudiado dos opciones.

Opción A: La primera opción es crear un vector 2D de dimensiones $1 \times N$ que guarda la posición 2D del coche en los N instantes anteriores. Cuando el LiDAR captura una nueva instantánea, los puntos almacenados en el vector se trasladan y se rotan, dependiendo de la orientación del coche, es decir, en el centro de la imagen aparece una representación del vehículo, y el recorrido realizado hasta ese instante va rotando dependiendo de la orientación actual del vehículo, como si fuera un GPS (ver Figura 3.3a). Para poder trasladar y rotar los puntos del vector, se aplica la Ecuación 3.5 a cada punto (P), en donde M es la matriz de transformación actual (Ecuación 3.2), obtenida al alinear las últimas dos nubes de puntos. En cada frame, al punto P se le va aplicando la nueva matriz de transformación obtenida y también, en cada frame, se añade un nuevo punto (0,0) y el punto más antiguo se elimina (limitando así el número de puntos representados en la imagen). De esta forma el origen de coordenadas se ubica en el último instante de grabación que se tiene registrado, y no en el primero (como se hace en el mapeado).

Opción B: La segunda opción es representar la trayectoria completa del vehículo, añadiendo cada punto nuevo calculado (Ecuación 3.1) a la imagen, partiendo de una posición fija (ver Figura 3.3b). Este método es parecido al del mapeado, se realiza multiplicando la matriz de transformación acumulada (Ecuación 3.1) por un punto homogéneo con coordenadas en el origen (Ecuación 3.6). De esta forma en cada frame se obtiene un nuevo punto 2D que se añade a la imagen, consiguiendo representar la posición actual del vehículo respecto al instante inicial.

Aunque al aplicar las ecuaciones, para ambas situaciones se obtienen puntos en 3D homogéneos, como se va a

representar la odometría en 2D, se omiten los datos obtenidos correspondientes al eje z.

$$P' = M_{\text{actual}}^{-1} * P \tag{3.5}$$

$$P' = M_{\text{acumulada}} * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.6}$$

(a) Opción a (b) Opción b

Figura 3.3: Representación del recorrido del vehículo: a) Opción A: el vehículo está fijo y el recorrido realizado por este es el que se mueve, b) Opción B: el vehículo va avanzando en el mapa.

El primer método presenta la ventaja de que el vehículo aparece representado en todo momento pero la visualización del recorrido realizado es limitada. El segundo cubre esa carencia, además de mostrar una imagen más visual del recorrido, pero si no se elige bien la escala ni el punto de inicio, o el recorrido es muy largo, la trayectoria del coche superaría los límites de la imagen y esta se dejaría de representar. Por lo tanto se puede emplear una representación u otra dependiendo del objetivo. En la sección de resultados se evaluará el error de la odometría.

3.3. Composición de imágenes panorámicas a partir de datos del LiDAR

En este trabajo no se utilizan cámaras que obtengan imágenes del entorno del vehículo, pero es posible crear dichas imágenes con el LiDAR. Como el LiDAR utilizado tiene 32 rayos en la vertical y un campo de visión horizontal de 360º, se pueden crear imágenes panorámicas de 360º con los datos de las intensidades y las distancias euclídeas de cada punto. Los datos necesarios para hacer esta reconstrucción provienen de la fase de adquisición de datos, de tal modo que de cada punto 3D se usan las coordenadas 3D, la distancia/profundidad e intensidad de la señal láser recibida. Esta idea se tomó del DRIVE PX2, en donde el kit de desarrollo de software (SDK) NVIDIA DriveWorks tiene un programa capaz de crear estas dos imágenes, aunque no es posible acceder a su código fuente.

Tal y como se ha descrito en la Sección 2.4, los datos recolectados por el LiDAR están desordenados. RTMaps es capaz de separar los paquetes de datos cada 360º, es decir, identifica cuando el LiDAR ha realizado una vuelta y guarda esos datos como un paquete. Esta es la única tarea de organización que realiza, pero es necesario realizar más tareas para componer correctamente la imagen, ya que el LiDAR tiene una secuencia de disparo alternante (Tabla 2.2).

Para solucionar este problema se ha creado una función que es capaz de leer los datos obtenidos del LiDAR con RTMaps, organizarlos y guardarlos. Para guardar estos datos se ha creado una estructura capaz de almacenar,

de cada punto recolectado, las coordenadas 3D (x, y, z), la distancia euclídea al punto y la intensidad del haz de vuelta una vez rebotado en el punto de la escena. En esta variable se almacenan los datos que recolecta en LiDAR en una vuelta completa (360º), cuando el LiDAR realiza la segunda vuelta los datos se sobrescriben ya que si no la memoria que ocuparía la variable y la potencia de la CPU/GPU necesaria para su manejo sería inadmisible.

Una vez que se tienen los datos ordenados, se pueden reconstruir las dos imágenes (intensidad y distancia). Para este caso, se utiliza la librería de OpenCV, con la que se crea un imagen en escala de grises con el valor de la intensidad de cada punto recolectado por el LiDAR en una vuelta (coloreando en negro los puntos cuyas intensidades son 0 o valores cercanos y en blanco los puntos cuyas intensidades se aproximan a 255). Lo mismo ocurre con la imagen de profundidad, solo que para esta hay que hacer una interpolación entre el rango en escala de grises de la imagen (0-255) y los valores de la distancia de cada punto, para que la distancia máxima sea 255, valor que corresponde al blanco en la escala de grises, y la mínima 0, valor correspondiente al negro.

A la imagen de intensidad se le aplica un mapa de color PINK mientras que a la imagen de profundidad se le aplica un mapa de color HSV (Figura 3.4), con el objetivo de obtener una mejor visualización desde un punto de vista humano (Figura 3.5 y 3.6).

Figura 3.4: Espectro del mapa de color, a la izquierda el mapa aplicado a la imagen de intensidad y a la derecha el aplicado a la imagen de profundidad.[3]

El motivo por el que la imagen de profundidad es menos nítida se debe a que el LiDAR es capaz de detectar puntos hasta 100m, pero la mayoría de puntos en la imagen se encuentran a menos de 20m, por lo que casi toda la imagen es coloreada con los primeros colores del espectro del mapa de color (rojo-naranja-amarillo) y presenta un menor contraste para el ojo humano. Esto no ocurre en la imagen de intensidad ya que el valor de la intensidad depende del material y por lo tanto está repartido más equitativamente por todo el espectro del mapa de color.

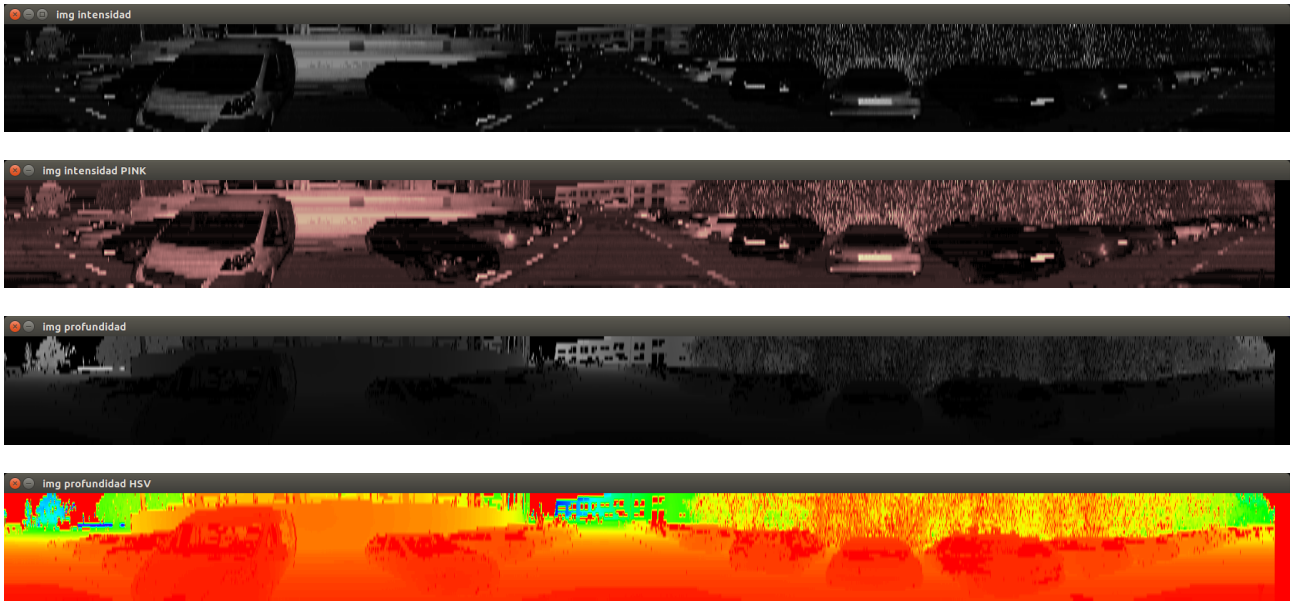


Figura 3.5: Composición de imágenes panorámicas a partir del LiDAR (1): a) Imagen de intensidad, b) Imagen de intensidad con un mapa de color PINK, c) Imagen de profundidad, d) Imagen de profundidad con un mapa de color HSV.

[3]Fuente: `https://docs.opencv.org/3.1.0/d3/d50/group__imgproc__colormap.html`

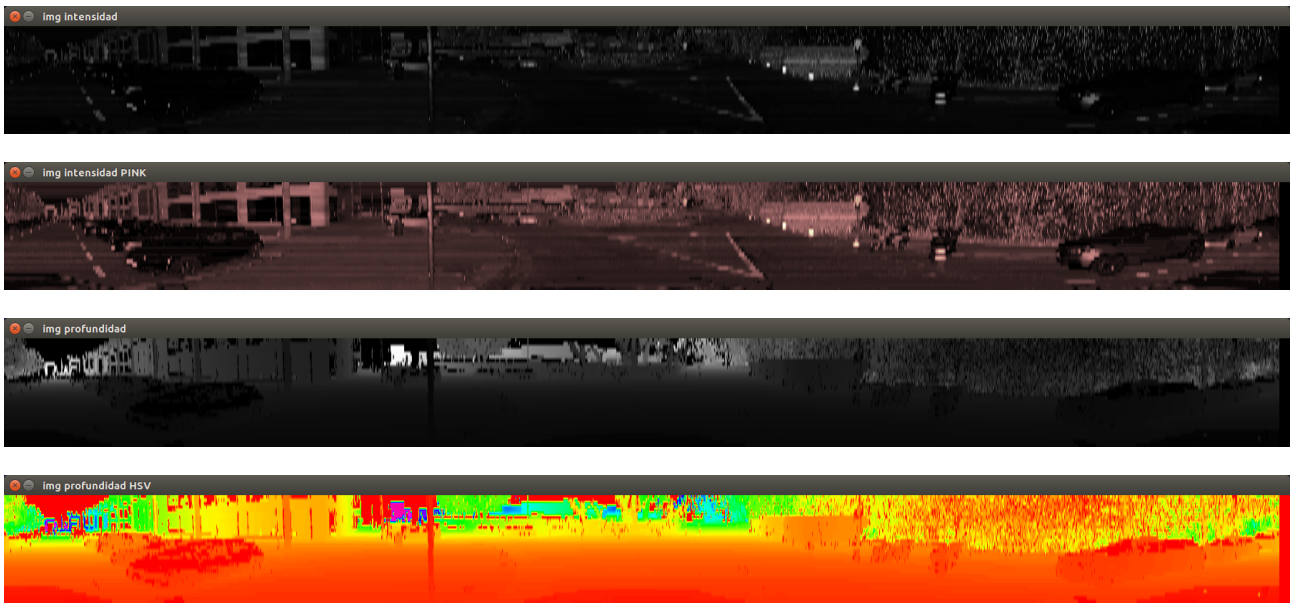


Figura 3.6: Composición de imágenes panorámicas a partir del LiDAR (2): a) Imagen de intensidad, b) Imagen de intensidad con un mapa de color PINK, c) Imagen de profundidad, d) Imagen de profundidad con un mapa de color HSV.

Capítulo 4

Pruebas y análisis de resultados

4.1. Comparativa de algoritmos ICP

Para este trabajo, se necesita un algoritmo que haga una buena alineación entre dos nubes de puntos 3D, tal cual se obtienen del sensor LiDAR y que sea capaz de hacerlo lo más rápido posible.

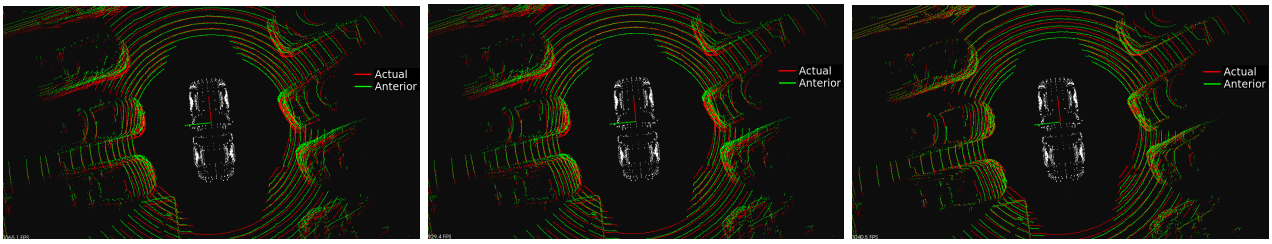
Para encontrar el algoritmo que mejor se adapta a nuestro caso, se estudió cada uno de los mencionados en la Sección 3.1.3 (Iterative Closest Point, Joint Iterative Closest Point y Generalized Iterative Closest Point). A cada uno de ellos se le pasa la información recolectada por el LiDAR y se analizan los resultados obtenidos.

Respecto al mapeado, no se dispone de ningún ground truth (datos de referencia) para conocer que algoritmo alinea mejor, por lo que se ha realizado una supervisión visual, pintando cada nube de puntos de distinto color y comprobando desde distintos puntos de vista 3D la calidad de su alineación (Figura 4.1).

Para analizar la odometría se utilizan datos GPS a modo de ground truth. Cuando se grabaron los datos con el coche de Vicomtech, a parte del LiDAR, el coche llevaba incorporado un GPS estándar. Sin embargo, los datos eran muy ruidosos e imprecisos por las limitaciones propias del sistema GPS. Por este motivo, se decidió crear manualmente un ground truth con los datos GPS de Google Maps. Estos datos, a parte de tener el error cometido durante la grabación con el coche de Google (en la Figura 4.3 se aprecia como la trayectoria representada no coincide exactamente con la carretera de Google Earth), tiene más errores derivados del desconocimiento del punto exacto donde empieza y acaba el recorrido realizado por el vehículo de Vicomtech.

Al igual que el mapeado, no es posible afirmar si el algoritmo funciona correctamente o no utilizando estos ground truth, pero sirve a modo de filtro para descartar aquellos algoritmos que produzcan grandes errores.

Tanto el mapeado como la odometría son tareas que el algoritmo tiene que realizar con un alto grado de precisión, aunque esto aumente el tiempo de cálculo del algoritmo. Este último, al no ser un factor determinante, solo se analiza para intentar optimizarlo, pero en ningún momento en el transcurso de este trabajo es un factor decisivo a la hora de elegir el algoritmo/método.



(a) ICP (b) JICP (c) GICP

Figura 4.1: Alineación de dos nubes de puntos utilizando cada uno de los algoritmos propuestos.

Algoritmo	ICP	JICP	GICP
Tiempo de cómputo (ms)	35107	36811	59046

Tabla 4.1: Tiempos medios de cálculo de la matriz de transformación de cada algoritmo.

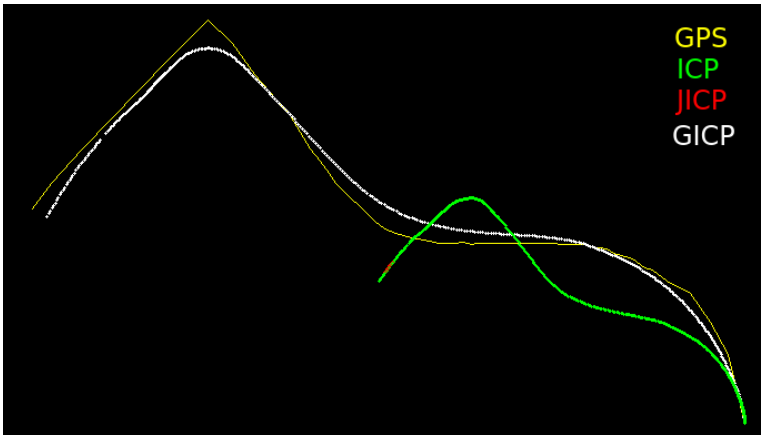


Figura 4.2: Representación del recorrido del vehículo utilizando las matrices de transformación de cada algoritmo.

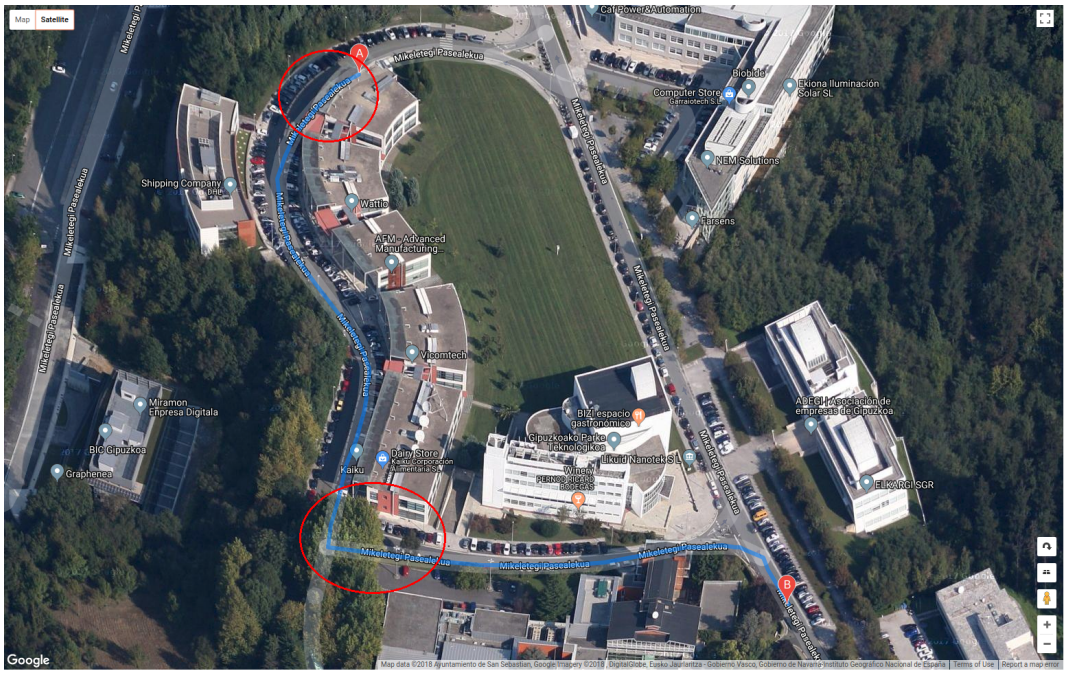


Figura 4.3: Error del ground truth obtenido de Google Maps.

Tal y como se detalla en la Sección 3.1.3, el algoritmo ICP y el JICP se comportan de forma parecida si solo se utilizan dos nubes de puntos para hacer la transformación, como es el caso estudiado. En la Figura 4.2 el recorrido realizado por el vehículo utilizando dichos algoritmos es prácticamente el mismo, ya que las trayectorias se solapan y los tiempos de cálculo (Tabla 4.1) son muy parecidos, pero los resultados obtenidos no son válidos (las nubes de puntos no se alinean) (Figura 4.1), el algoritmo se atasca en un mínimo local erróneo y no es capaz de encontrar la solución óptima. En cambio el algoritmo GICP es capaz de hacer una buena alineación de las nubes de puntos (Figura 4.1) y realizar una trayectoria similar al ground truth (Figura 4.2), pero esa precisión disminuye la velocidad, tardando una media de casi 1 minuto en calcular cada matriz de transformación (Figura 4.1).

A la vista de los resultados obtenidos, se ha decidido utilizar el algoritmo GICP (Generalized Iterative Closest Point), ya que es con el que mejores resultados se obtienen para el caso planteado. Este algoritmo se pondrá a prueba más exhaustivamente en la Sección 4.3 con el dataset de KITTI.

4.2. Optimización del algoritmo GICP

Uno de los grandes retos de todo algoritmo es minimizar su tiempo de ejecución. En el trabajo desarrollado, el cuello de botella ocurre al calcular las matrices de transformación (Tabla 4.2). No obstante, el foco de este TFM se centra en la búsqueda del algoritmo más adecuado con un menor error de alineación entre nubes 3D. A continuación se describen varias propuestas de filtrado y optimización que influyen en el cálculo de la matriz de transformación. Estos métodos no solo reducirán el tiempo de cálculo, si no que también optimizarán el algoritmo produciendo una mejor alineación entre nubes 3D consecutivas.

4.2.1. Métodos de optimización

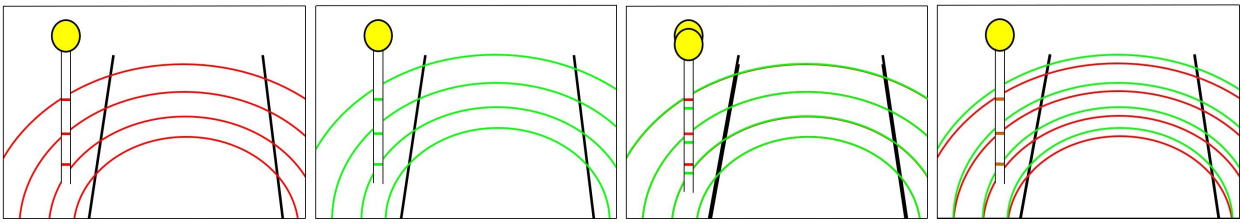
4.2.1.1. Matriz de estimación

Para minimizar el tiempo de cálculo de las matrices de transformación y evitar el que algoritmo se quede atrapado en un mínimo local, una de las cosas que se puede hacer es inicializar el algoritmo con una matriz de transformación estimada [57]. Esta matriz se aplica a la última nube de puntos obtenida por el LiDAR y con la nube de puntos cerca de su ubicación correcta, se aplica el algoritmo GICP para obtener la matriz de transformación corregida.

La obtención de la matriz de transformación estimada se puede realizar de diferentes formas. El método más utilizado y preciso se consigue utilizando el filtro de Kalman combinado con datos de los sensores GPS e IMU (inertial measurement unit) [57]. En este caso, como el único sensor que se está utilizando es el LiDAR, se utiliza como matriz de transformación estimada la matriz de transformación obtenida en el instante anterior. Este método se puede utilizar debido a que el LiDAR captura una instantánea del entorno cada 0.2-0.05 segundos (graba con una frecuencia de entre 5 y 20 Hz). Se asume por tanto que es suficientemente pequeño para que la trayectoria del vehículo sea estable y no aparezcan cambios bruscos.

4.2.1.2. Reducción del tamaño de la nube de puntos

El algoritmo GICP compara las dos nubes de puntos dadas buscando la correlación entre sus puntos. Como únicamente utiliza la localización 3D del punto y no usa datos de color, intensidad... para buscar sus correspondencias, es importante que en las nubes de puntos haya objetos estáticos y sin superficies lisas, ya que objetos como vehículos en movimiento y la carretera podría incitar al error y el algoritmo haría una mala alineación. En la Figura 4.4 se puede apreciar el caso descrito, las líneas rojas corresponden a los puntos capturados por el LiDAR en el instante anterior mientras que los puntos verdes corresponden a los capturados en el instante actual. Si el algoritmo da mas peso a las superficies planas (carretera) obtendrá una alineación incorrecta (Figura 4.4c), ya que estos puntos tienen características similares pero no son los mismos, por lo que no se deben alinear. Si por el contrario se da más pesos a los objetos estáticos, el algoritmo hará una buena alineación (Figura 4.4d). Por este motivo cuantos más objetos estáticos y con superficies discontinuas haya en la imagen, mejor alineación hará el algoritmo. Este proceso se asemejaría a una alineación de bordes entre imágenes 2D. Dada la limitación del LiDAR en su resolución vertical, las discontinuidades o bordes en dicha dirección proporcionan una información clave para la correcta alineación de las nubes de puntos.



(a) Nube de puntos anterior (b) Nube de puntos actual (c) Alineación errónea (d) Alineación correcta

Figura 4.4: Alineación errónea y correcta de las nubes nubes de puntos.

Para resolver esta situación, se plantean dos métodos que no utilizan toda la nube de puntos para hallar la matriz de transformación si no una versión reducida: filtrar la nubes de puntos por los rayos (reducción del

número de rayos emitidos por el LiDAR en la vertical) o filtrar la nube de puntos por el valor de la intensidad.

Filtrado por rayos: Este filtro simula una disminución del número de rayos emitidos por el LiDAR, eliminando los rayos que están por debajo de X ángulo, eliminando así la parte inferior de la imagen, ya que en esta parte, en su mayoría se ubicarían los objetos móviles y las superficies planas (carretera), mientras que en el parte superior quedarían los objetos estáticos (edificios, farolas, árboles...). Esto a su vez reduce también el tiempo de cálculo de la matriz de transformación.

Filtrado por intensidad: El objetivo de este filtro es eliminar las superficies planas correspondientes a la carretera. Como se sabe la ubicación del LiDAR en el vehículo y las medidas de este, se conoce la ubicación 3D de la carretera, de esta forma se extrae el valor de intensidad de los puntos correspondientes a la carretera en un radio de unos 5 metros. De estos puntos se extrae el valor medio y la desviación típica de la intensidad y con estos datos, se filtra la nube de puntos para eliminar los puntos que corresponden a la carretera. Al igual que el filtro anterior, este también reduce el tiempo de cálculo de la matriz de transformación

Para evaluar los resultados se han tomado imágenes de distinto tamaño y se ha realizado la alineación, midiendo los tiempos de cálculo de la matriz de trasformación y calculando la posición (x, y, z), la orientación (roll, pitch, yaw) y la media cuadrática (RMS) en 2D y en 3D (Ecuación 4.1 y 4.2).

$$D_{\mathrm{RMS}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} d_i^2} = \sqrt{\frac{d_1^2 + d_2^2 + \cdots + d_N^2}{N}} \tag{4.1}$$

En donde d es la distancia euclídea entre dos puntos (Ecuación 4.2)

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \tag{4.2}$$

4.2.2. Resultados y conclusiones de las propuestas de optimización

Al analizar el tiempo de ejecución de las funciones principales que intervienen en el programa (Tabla 4.2), se observa que el proceso más lento con diferencia es el del cálculo de la matriz de transformación. Esta función es la más importante del código, por lo que será la función a la que se le aplicarán los métodos de optimización propuestos anteriormente.

Funciones	**Tiempos (ms)**
Leer, organizar y guardar los datos	106.89
Crear nube de puntos	6.11
Obtener matriz de transformación (GICP)	59046.28
Mapeado	66.70
Odometría	2.15
Componer imágenes panorámicas	13.55
Visualizar la reconstrucción 3D	82.08

Tabla 4.2: Tiempos de procesamiento de cada función del programa.

En la Tabla 4.3 se muestran el tiempo que tarda el algoritmo GICP en obtener la matriz de transformación utilizando matriz de estimación (Sección 4.2.1.1) y sin utilizarla. El resultado es un ahorro medio del 58 % usando la matriz de estimación. Este método se utilizará para el resto de pruebas ya que su única influencia es en el tiempo.

	Tiempo de cómputo
Sin matriz de estimación	59046 ms
Con matriz de estimación	24752 ms

Tabla 4.3: Tiempos medios de cálculo de la matriz de transformación con y sin matriz de estimación.

Para comprobar la efectividad del método de reducción del tamaño de la nube de puntos, al igual que en la Sección 4.1, se utilizan a modo de ground truth los datos GPS obtenidos de Google Maps y se van a estudiar 7 casos diferentes. 5 de ellos utilizando el filtrado por rayos, reduciendo el tamaño de la nube de puntos como si esta hubiera sido obtenida por un LiDAR de 22, 18, 14, 10 o 6 rayos (siempre eliminando los rayos de la parte inferior de la imagen). Para lo otros 2 casos se ha utilizado el filtrado por intensidad (eliminación de los puntos 3D que corresponden a la carretera), en uno de ellos se calcula la media ($\bar{x}$) y la desviación típica (σ) de los valores de intensidad de los puntos que corresponden a la carretera y se elimina de la nube los puntos cuyo valor de intensidad coincide con $\bar{x} \pm \sigma$, el otro caso es igual que este, pero aumentando el umbral a $\bar{x} \pm 1{,}5 * \sigma$.

En la Figura 4.5 se representa la odometría después de haber aplicado el método de reducción de puntos. En la imagen de la izquierda (filtrado por rayos) se ve como las líneas azul y rosa se aproximan mejor al ground truth, mientras que en la imagen de la derecha (filtrado por intensidad) ambos casos tienen una odometría semejante. Como los datos obtenidos de Google Maps tienen poca densidad, no se pude hacer un análisis más en detalle utilizando ese ground truth, por lo que se ha escogido la línea verde oscura ($\bar{x} \pm 1{,}5 * \sigma$) de filtrado por intensidad como referencia para seguir extrayendo conclusiones, ya que en principio parece una buena aproximación. Así, en la Tabla 4.4 se observa como el error medio cuadrático (RMS) en 3D es mucho mayor que el error en 2D (Ecuación 4.1 y 4.2), esto quiere decir que la odometría (alineación en 2D) de todos los casos es bastante similar (RMS 2D son valores bajos), pero cuando el algoritmo tiene que alinear las nubes de puntos verticalmente es cuando surgen los problemas.

Para analizar más en profundidad esta cuestión, se ha representado gráficamente el valor de las coordenadas (x,y,z) y de la orientación (roll,pitch,yaw) de cada uno de los 7 casos descritos anteriormente en la Figura 4.6.

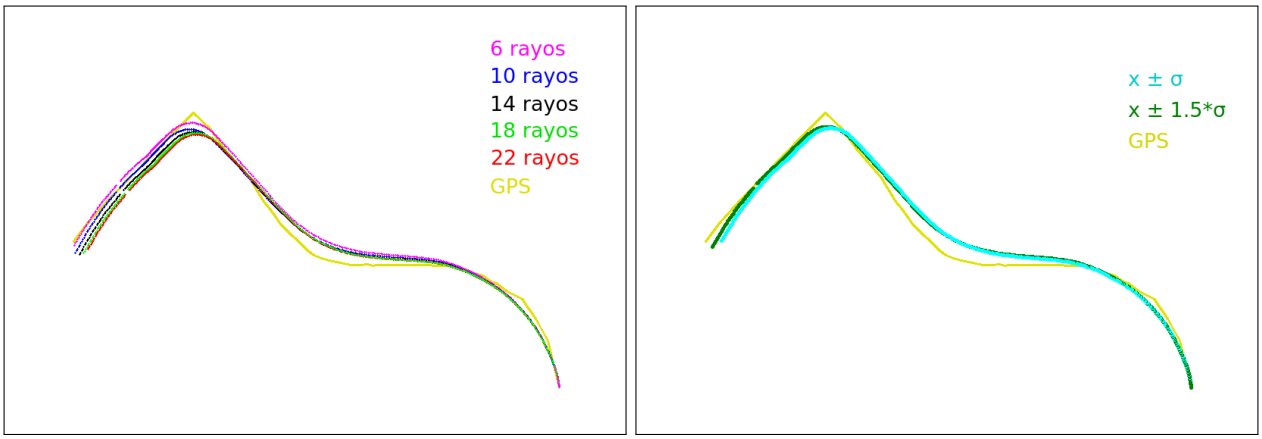
Considerando el error en la odometría en 2D (x,y,yaw), no se observa un claro ganador debido a que todos obtienen resultados parecidos y el ground truth usado también tiene un error no despreciable. Sin embargo, el análisis en 3D proporciona información útil. El roll y el pitch tienen un offset desconocido, ya que el vehículo estudiado no parte de la posición de reposo, a groso modo, el roll tendría que aumentar, disminuir y volver a aumentar (debido a las curvas del recorrido), mientras que el pitch tendría que disminuir hasta el frame 280, donde comenzaría a aumentar hasta estabilizarse. Si nos fijamos en la coordenada z, la gráfica tendría que aumentar hasta el frame 280, a partir de ahí disminuir un poco y estabilizarse (el coche se para) en los 10-15m para volver a disminuir un poco más.

Con todo esto se puede concluir que el filtrado por intensidad (líneas azul clara y verde oscura) obtiene mejores resultados que el filtrado por rayos, esto se puede observar claramente en la coordenada z, aunque ningún caso se acerca a la solución exacta (10-15m), las líneas azul clara, azul oscura y verde oscura son las que más se aproximan, pero cuando el valor de la coordenada z tiene que disminuir (frame 400) solo lo hace en los casos que se ha aplicado el filtro por intensidad, mientras que en el otro caso sigue aumentando.

De todo esto se concluye que la matriz de estimación reduce el tiempo de cálculo de la matriz de transformación, el método de filtrado por rayos no es válido y el método de filtrado por intensidad (eliminación de los puntos 3D que corresponden a la carretera) obtiene unos resultados más precisos pero con cierto error. El motivo de este error es debido a la densidad del LiDAR, se está utilizando un LiDAR que recolecta 32 puntos en la vertical y unos 2100 puntos en la horizontal, por lo que hay una diferencia de densidad muy significativa que influye a la hora de obtener una buena alineación en el eje z. Para seguir comprobando la efectividad del algoritmo se empleará el dataset de KITTI, en donde los datos se han obtenido con un LiDAR el doble de denso, y el método de optimización elegido para realizar dicha comprobación es el filtrado por intensidad ($\bar{x} \pm 1{,}25 * \sigma$), resultado de la combinación de los dos mejores casos estudiados.

Filtro aplicado	**Filtro por rayos**					**Filtro por intensidad**	
	22 rayos	**18 rayos**	**14 rayos**	**10 rayos**	**6 rayos**	$\bar{x} \pm \sigma$	$\bar{x} \pm 1{,}5 * \sigma$
Tiempo medio (ms)	38608.96	32032.61	24825.81	18135.29	11192.09	18950.43	12402.23
RMS 2D (m)	4.46862	5.37353	2.01295	2.20572	3.54622	1.23975	-
RMS 3D (m)	39.0856	43.7688	29.9384	4.07127	23.2235	19.2659	-

Tabla 4.4: Tiempos y RMS aplicando el filtrado por rayos y por intensidad al algoritmo GICP.



(a) Filtrado por rayos (b) Filtrado por intensidad

Figura 4.5: Representación del recorrido del vehículo utilizando el algoritmo GICP con filtrado de la nube de puntos por rayos a la izquierda y con filtrado de la nube de puntos por intensidad a la derecha.

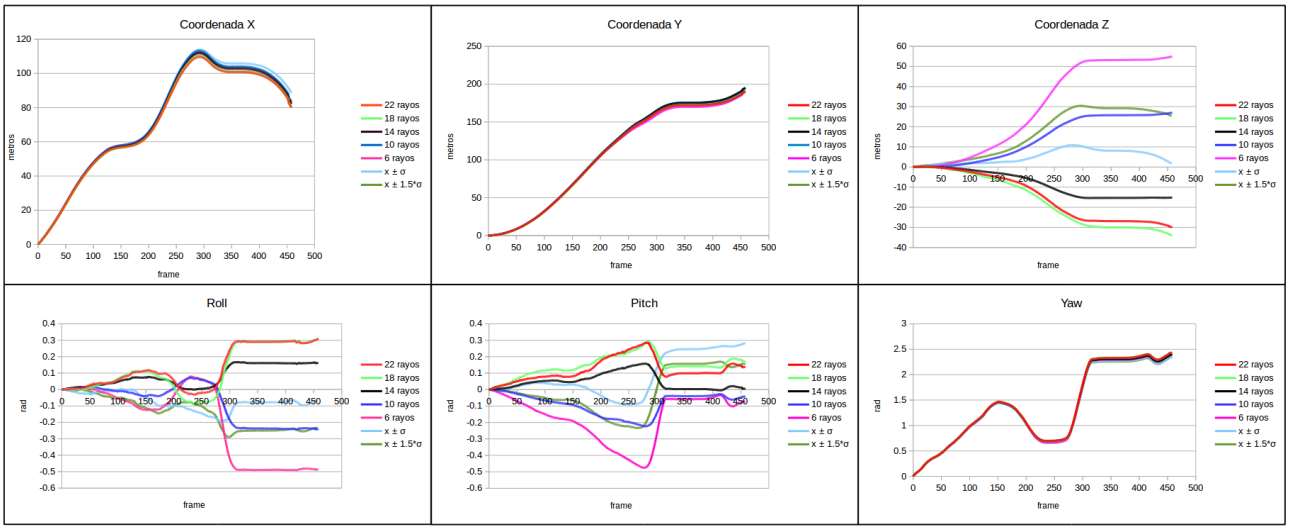


Figura 4.6: Coordenadas (x,y,z) y ángulos (roll,pitch,yaw) obtenidos al aplicar el algoritmo GICP utilizando las propuestas de reducción de la nube de puntos.

4.3. Test con datos de KITTI

Una vez elegido y optimizado el algoritmo, se comprueba su rendimiento con la base de datos de KITTI [58]. KITTI dispone de un dataset para odometría con 22 secuencias (22 recorridos distintos realizados por el coche de KITTI recopilando información del entorno con los sensores instalados), de las cuales, las 11 primeras disponen de ground truth, por lo que se puede comprobar la fiabilidad del algoritmo desarrollado. Para ello se ha modificado el código para poder leer el dataset, ya que los datos recolectados por el LiDAR de KITTI son proporcionados en formato BIN, en donde cada frame de nube de puntos es un archivo BIN distinto.

Para la comprobación se ha elegido la secuencia 07 del dataset, ya que se trata de un lazo cerrado, hay cuestas en el terreno y es un entorno urbano. Al ejecutar el algoritmo GICP con filtro por intensidad sobre este dataset, se obtienen mejores resultados que los mostrados con el coche de Vicomtech. La odometría en 2D (Figura 4.7) es bastante similar al ground truth, si cuantificamos estos resultados (Tabla 4.5) el error medio cuadrático 2D y 3D es bastante similar, no existe esa diferencia tan grande entre el error en 2D y el error en 3D que había al utilizar el Velodyne HDL-32E. Esto se puede comprobar en la Figura 4.8, en donde las líneas se solapan con bastante precisión.[1]

[1]Cada gráfica tiene su propia escala para una mejor visualización. En el caso de la Coordenada Z, al tener una escala mucho menor comparada con las otras dos, se aprecia más el error, pero esto no quiere decir que exista más error que en las otras coordenadas.

En consecuencia, se puede afirmar que el error de alineación en el eje z que se observó en la Sección 4.2.2 se debe a la densidad del LiDAR. Esto es, con un LiDAR más denso, se soluciona dicho error. Para el recorrido estudiado, de unos 2290m divididos en 1100 frames, se ha obtenido un error total del 0.91 % y un error medio cuadrático de 1.94m.

	Tiempo medio (ms)	**RMS 2D (m)**	**RMS 3D (m)**
Odometría	35426	1.756	1.940

Tabla 4.5: Tiempo medio de cómputo y RMS al utilizar el algoritmo GICP con filtrado por intensidad con el dataset de KITTI.

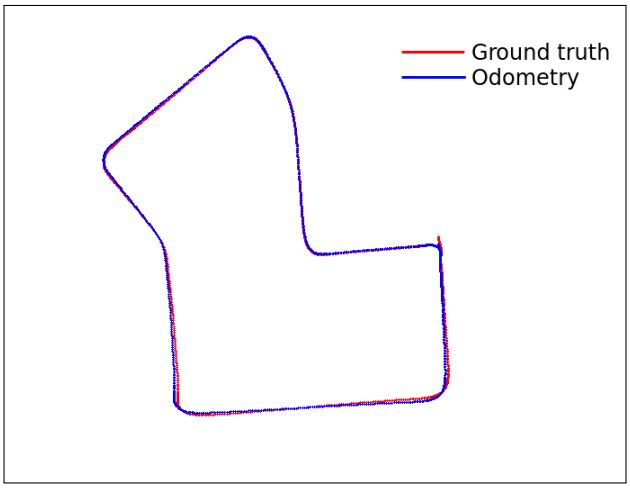


Figura 4.7: Odometría utilizando el algoritmo GICP con filtrado por intensidad al dataset de KITTI.

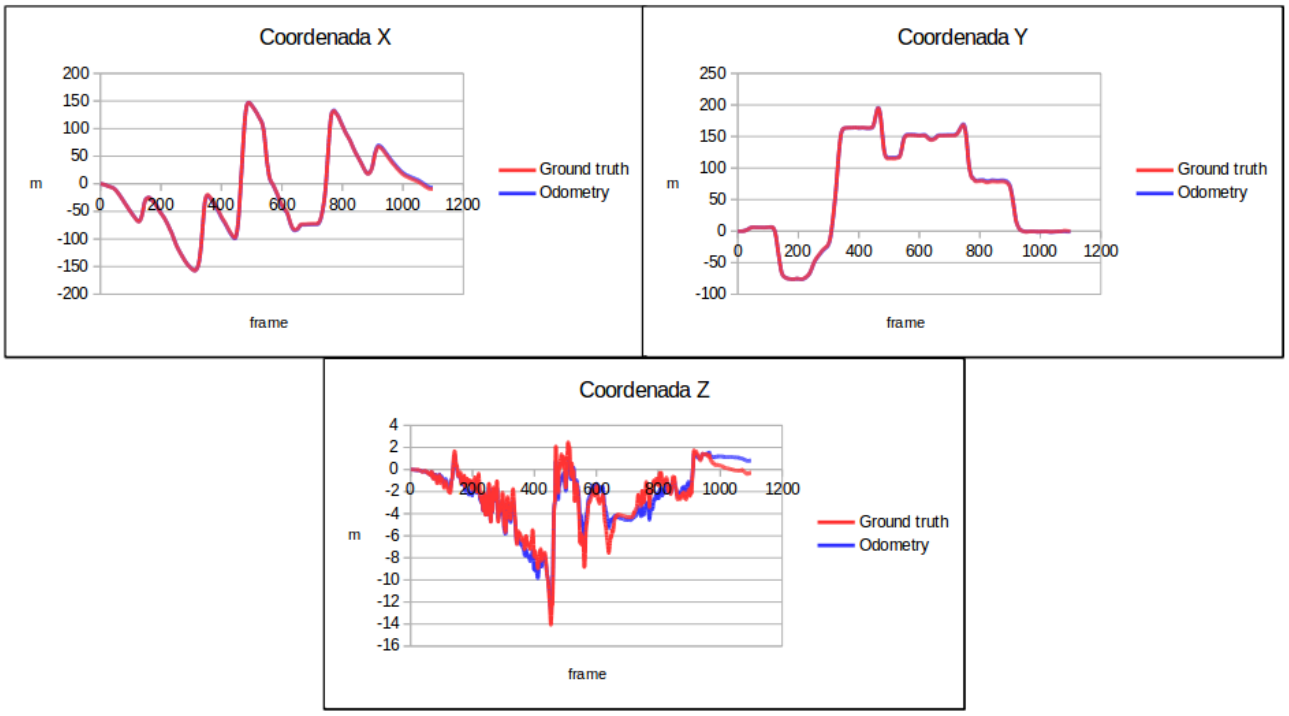


Figura 4.8: Coordenadas (x,y,z) de cada frame resultados de aplicar el algoritmo GICP con filtrado por intensidad al dataset de KITTI.

Haciendo una comparativa entre la reconstrucción 3D utilizando el algoritmo GICP con el filtrado por intensidad y sin aplicar dicho método de optimización, ambos utilizando matriz de estimación, se pueden apreciar las mejoras introducidas en el algoritmo (ver Tabla 4.6). El algoritmo creado reduce el tiempo de cálculo medio en un 70 % y mejora la precisión en un 1.84 %, llegando a operar con un 99.09 % de precisión.

	Tiempo medio (ms)	RMS 3D (m)	Precisión (%)
GICP sin filtro por intensidad	117796	4.3059	97.25
GICP con filtro por intensidad	35426	1.9448	99.09

Tabla 4.6: Comparativa del algoritmo GICP usando y sin usar el filtro por intensidad.

4.4. Representación de la odometría y la reconstrucción 3D

Una buena representación del las capacidades del algoritmo es algo fundamental de todo programa. En este caso se está trabando con un dispositivo capaz de recolectar en torno a un millón de puntos por segundo si se usa el Velodyne LiDAR HDL-32E y más de 2 millones de puntos por segundo si se usa el Velodyne LiDAR HDL-64E, lo que supone el manejo de varios millones de datos por segundo, ya que cada punto tiene información de su posición 3D y el valor de intensidad.

Los dispositivos utilizados (CPU y software) no son capaces de procesar, retener y mostrar toda la información que se maneja, por eso hay que ir liberando espacio. Cada vez el programa lee los datos obtenidos por el LiDAR en un frame, estos datos sobreescribe los anteriores, la representación 3D de nubes de puntos está limitada a 500.000 puntos y la representación de la odometría también está limitada.[2]

Al igual que a las imágenes panorámicas compuestas con los valores de intensidad y profundidad se les han aplicado filtros de color para una mejor visualización, las nubes de puntos resultantes del mapeado se han coloreado con el mapa de color de la Figura 4.9 en función de los valores de intensidad (0-255) de cada punto. También se se ha insertado una nube de puntos blanca con forma de coche (obtenido de un dataset de archivos PLY [59]) para mostrar de forma más visual la ubicación y el movimiento del vehículo.

Otro de los motivos por los que no se puede representar el mapeado completo que va realizando el algoritmo, es debido a que librería PCL está más enfocada al procesamiento y no tanto a la visualización de nubes de puntos. Aun así, al limitar el tamaño de la nube de puntos renderizada es posible manejarse por el interior de la reconstrucción 3D para visualizar el mapeado desde distintos puntos de vista. Es decir, es posible crear una reconstrucción completa del recorrido pero no representarla y visualizarla de manera incremental debido a las limitaciones comentadas.[3] Se necesitaría del uso de otros recursos software y hardware más enfocados a la representación de nubes de puntos.

En las Figuras 4.10, 4.11, 4.12, 4.13 y 4.14 se muestran varias secuencias resultado de aplicar el algoritmo desarrollado a los datos obtenidos por el LiDAR durante el recorrido realizado por el coche de Vicomtech. En la parte superior se encuentran las imágenes panorámicas de intensidad y profundidad con sus respectivos mapas de color (Sección 3.3), a la izquierda la representación de la odometría 2D de vehículo (Sección 3.2) y a la derecha la reconstrucción 3D del entorno (Sección 3.1).

Figura 4.9: Mapa de color con los valores de la intensidad para la reconstrucción 3D.

[2]Esta limitación es más por una cuestión de visualización que por sobrecarga de memoria, y el tamaño de puntos que se pueden representar es muy flexible, en función de las necesidades del usuario.

[3]Aplicando filtros para reducir la densidad de la nube de puntos es posible representarla, pero la resolución obtenida es tan baja que no se distinguen los objetos.

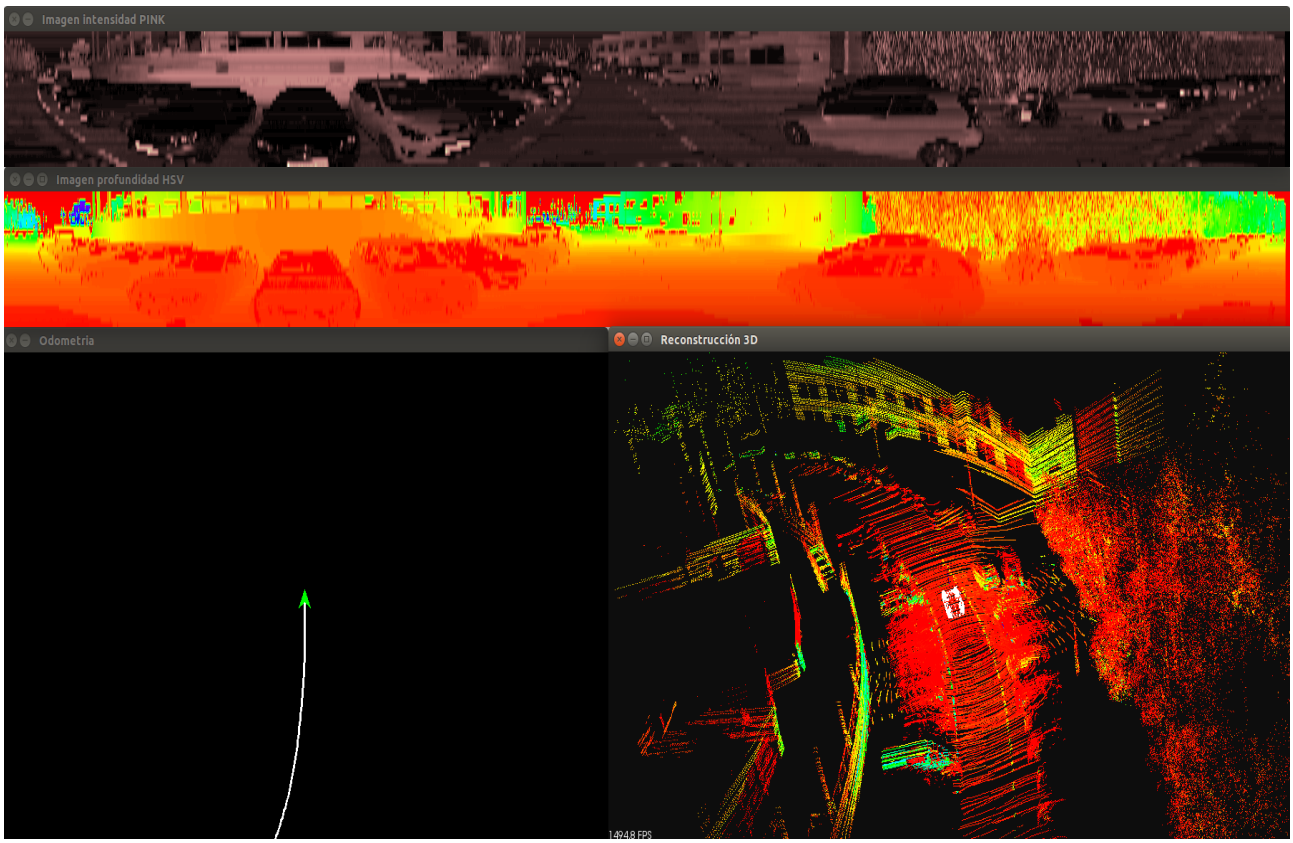


Figura 4.10: Reconstrucción 3D, odometría y composición de imágenes panorámicas de la escena (1).

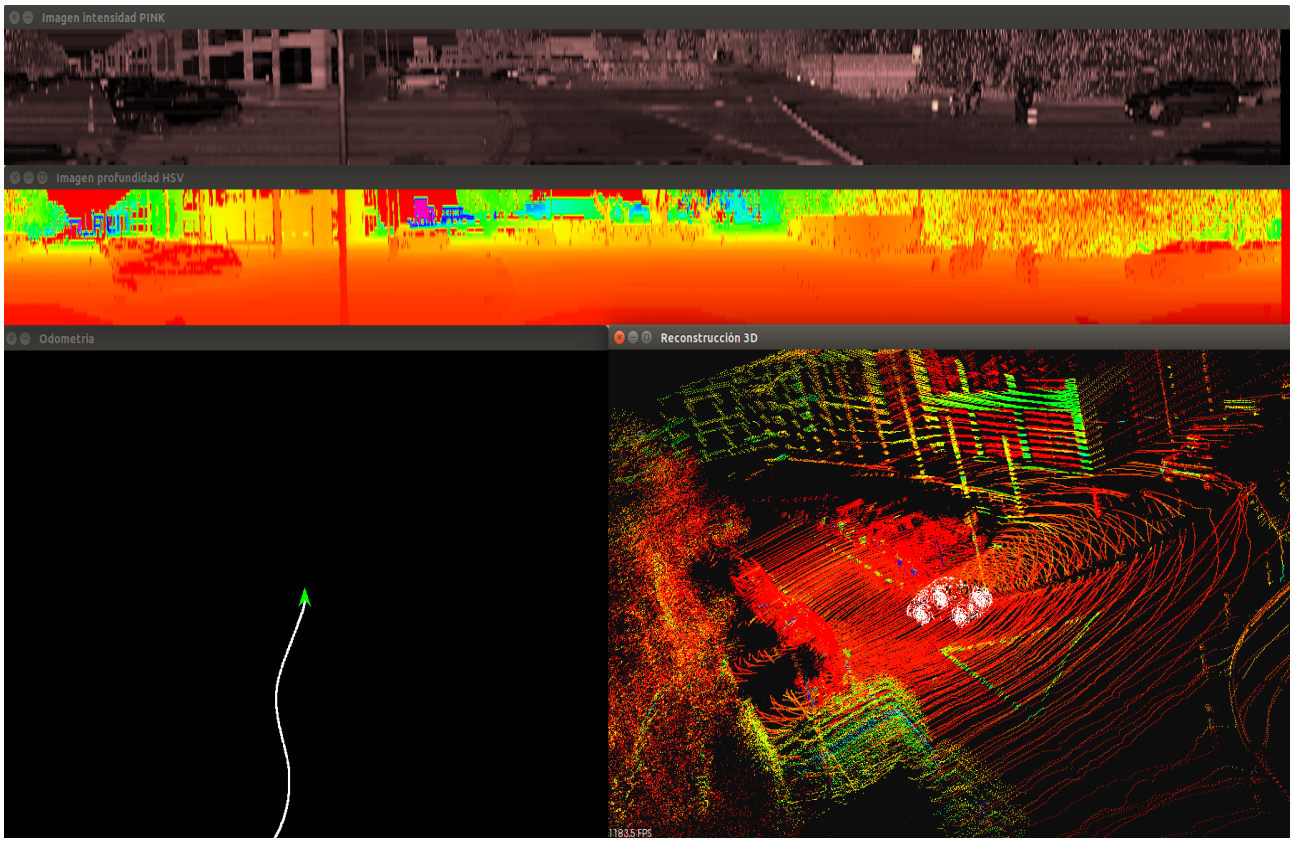


Figura 4.11: Reconstrucción 3D, odometría y composición de imágenes panorámicas de la escena (2).

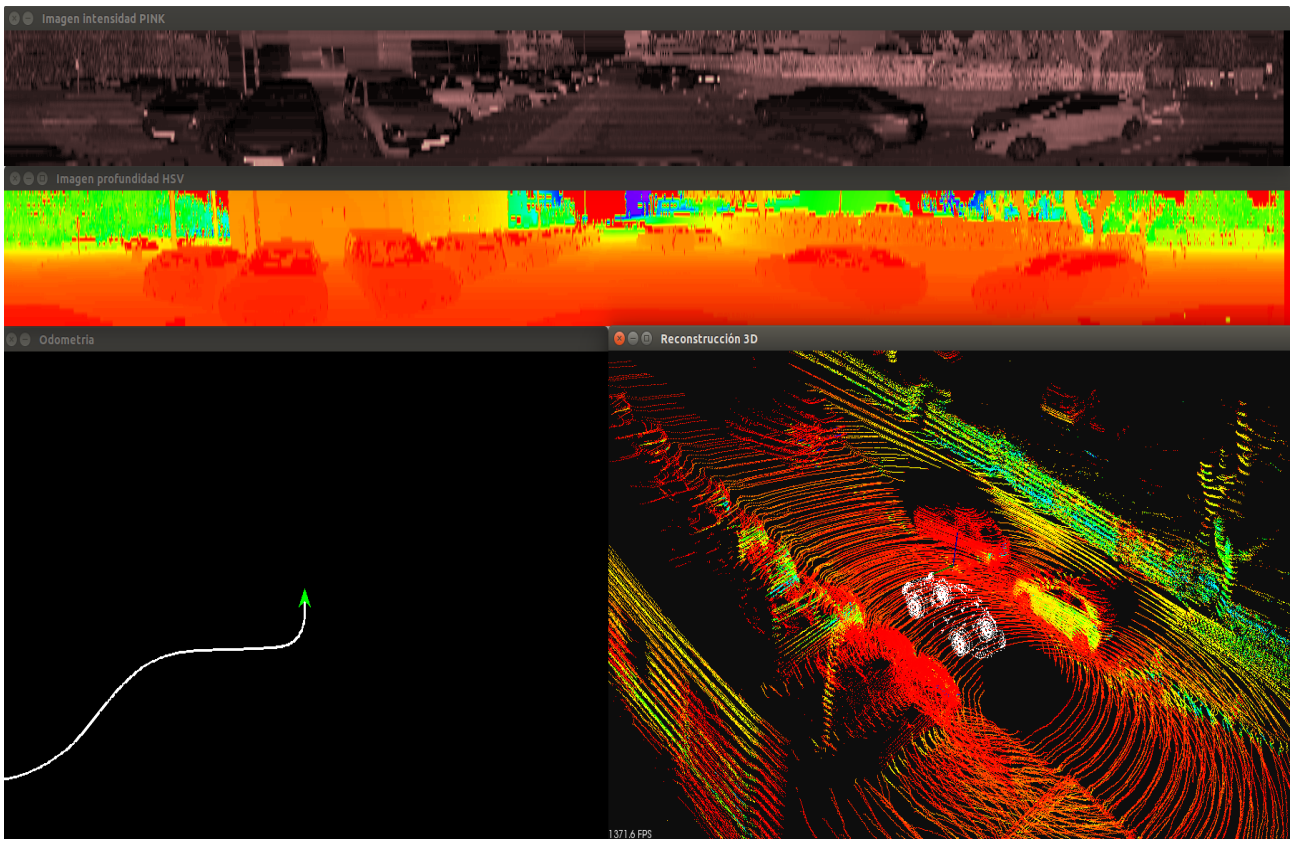


Figura 4.12: Reconstrucción 3D, odometría y composición de imágenes panorámicas de la escena (3).

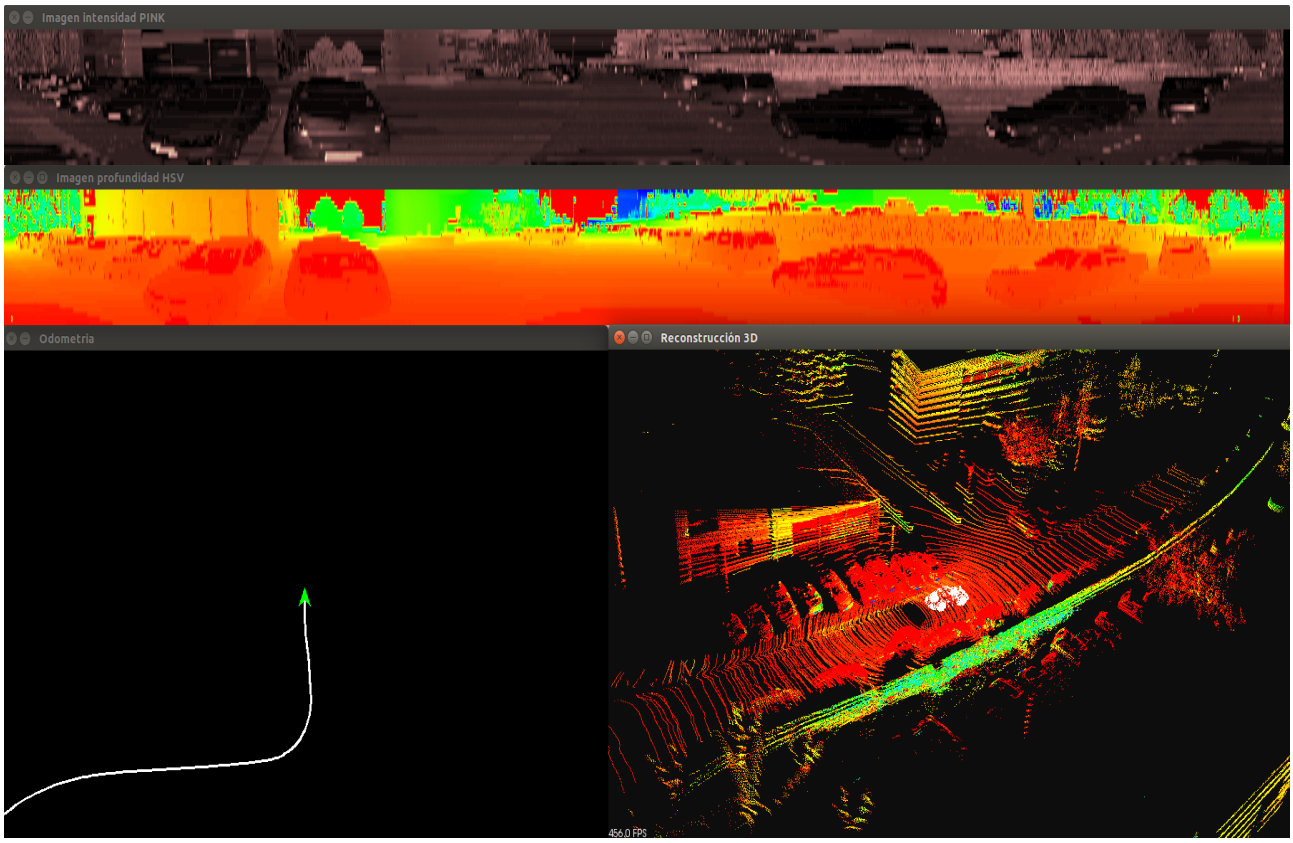


Figura 4.13: Reconstrucción 3D, odometría y composición de imágenes panorámicas de la escena (4).

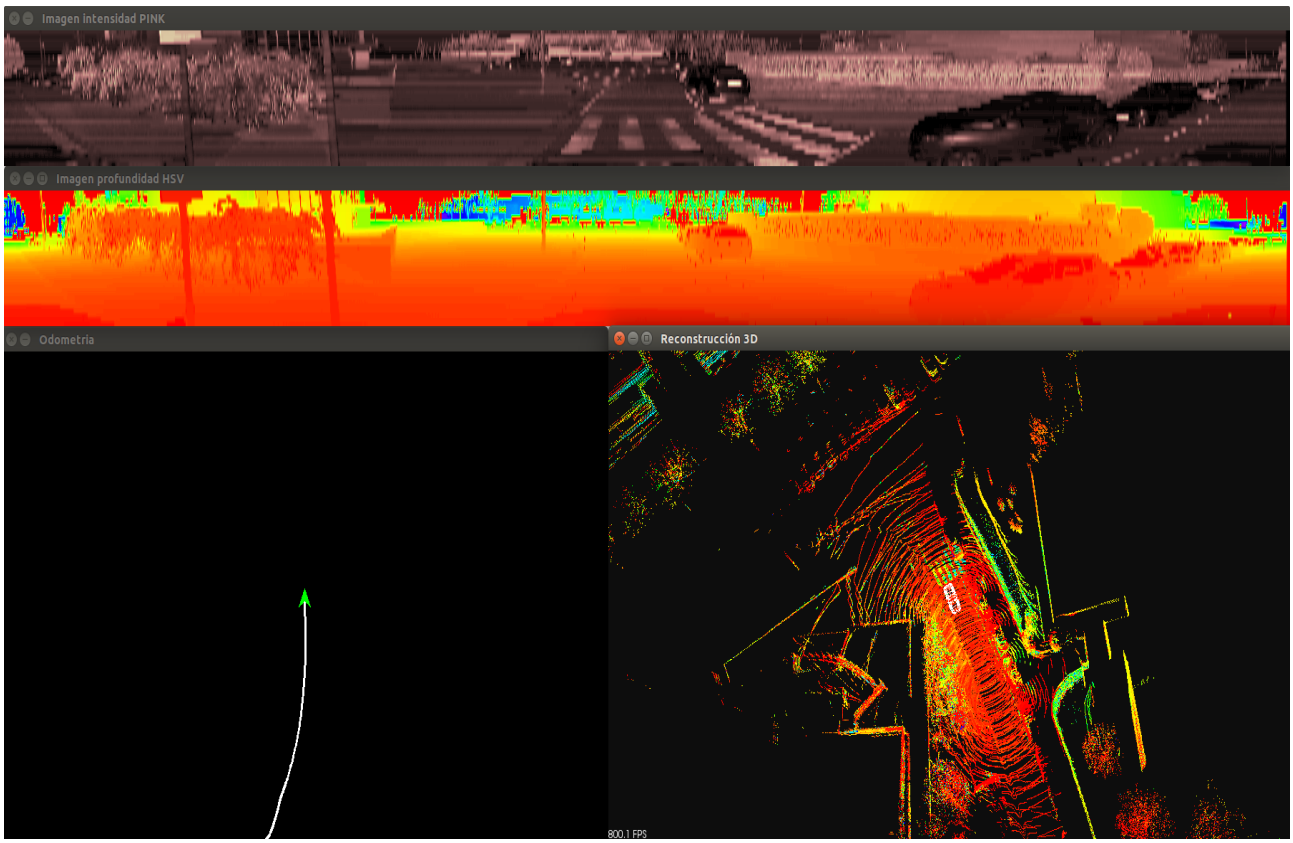


Figura 4.14: Reconstrucción 3D, odometría y composición de imágenes panorámicas de la escena (5).

Capítulo 5

Conclusiones

En este TFM se han abordado las tareas de reconstrucción 3D del recorrido del vehículo y estimación del movimiento y la posición del mismo, tareas esenciales para la realización del SLAM. Se ha empleado la tecnología LiDAR para la extracción de información semántica de interés, obteniendo nubes de puntos 3D del entorno y se han estudiado y evaluado diferentes variantes del algoritmo ICP (Iterative Closest Point), proponiendo el uso del algoritmo GICP (Generalized Iterative Closest Point) como método para obtener la matriz de transformación capaz de alinear las nubes de puntos. También se ha hecho un estudio de mercado de los dispositivos LiDARs más utilizados en vehículos autónomos que demuestra que los LiDARs Velodyne HDL-32E y Velodyne HDL-64E, utilizados en este trabajo, son los más adecuados para los temas tratados. Se ha utilizado la ayuda de las librerías de software libre OpenCV y PCL (Point Cloud Library) para el manejo y la representación de datos.

Con el objetivo de realizar la reconstrucción 3D y la odometría, se ha desarrollado un algoritmo del que destacan 5 funciones principales. 1) adquisición de datos del LiDAR y organización de éstos en nubes de puntos 3D; 2) construcción y representación de imágenes panorámicas de 360º con el valor de la intensidad y de la profundidad de cada punto; 3) estimación de la matriz de transformación entre dos nubes 3D consecutivas mediante el algoritmo GICP, 4) cálculo de la odometría en el plano suelo y representación en una imagen del recorrido realizado por el vehículo; 5) reconstrucción 3D de la escena mediante la fusión de las nubes de puntos a lo largo del recorrido del vehículo.

Para la evaluación del algoritmo se han utilizado datos procedentes del LiDAR Velodyne HDL-32E, instalado en un coche de la Fundación Vicomtech, y de la base de datos de KITTI, que obtiene los datos con LiDAR Velodyne HDL-64E. De este análisis se obtiene que el algoritmo GICP es muy lento ($\sim$1 minuto por frame con el HDL-32E) y que para nubes de puntos con poca densidad, las superficies planas tiene un impacto negativo en la alineación, por lo que se han propuesto diferentes métodos para solventar estos problemas.

Para reducir el tiempo de cómputo del algoritmo GICP, se propone el uso de una matriz de estimación inicial de la cual parte el algoritmo para hacer la alineación. Al no disponer de dispositivos capaces de obtener esta matriz (GPS, IMU, INS...) se usa la matriz de transformación anterior a modo de matriz de estimación. Este método reduce en un 58 % el tiempo de calculo del algoritmo.

La solución propuesta para la eliminación de la superficie plana predominante en el entorno (carretera) es el filtrado por intensidad, extrayendo de las nubes de puntos el valor de las intensidades de los puntos que corresponden a la carretera y eliminándolos durante la alineación. Con este método el algoritmo es capaz de realizar la odometría con un error medio cuadrático de 1.756m y un precisión del 99.64 % y la reconstrucción 3D con un error medio cuadrático de 1.94m y una precisión del 99.09 %, mejorando en un 1.84 % la precisión del algoritmo GICP. Ambas tareas conjuntas las realiza con una duración media de 35.426 segundos por frame, frente a los 117.796 segundos que tarda el algoritmo GICP. Estos resultados se han obtenido con el dataset de KITTI que usa un LiDAR de 64 rayos. Si se utiliza un LiDAR con menos rayos, esto repercute principalmente en la reconstrucción 3D, empeorando la alineación vertical, sin embargo, su efecto sobre la odometría 2D es mínimo, siempre y cuando el LiDAR tenga un campo de visión horizontal de 360º.

Para resumir, en este trabajo se ha demostrado el que algoritmo Generalized Iterative Closest Point (GICP) es el más adecuado para la estimación de las matrices de transformación entre nubes de puntos 3D consecutivas. Partiendo de esta base, se ha desarrollado un algoritmo que estima de forma precisa (99.64 %) la odometría del vehículo, realiza una reconstrucción 3D del recorrido del vehículo con una precisión del 99.09 % y construye

imágenes panorámicas de 360º del entorno con datos de profundidad e intensidad, todo ello usando solamente datos procedentes de sensores LiDAR. Este algoritmo no solo mejora la precisión si no que reduce el tiempo de procesamiento del algoritmo GICP en un 87.4 %.

Como líneas futuras se propone la integración de más sensores (p.e. IMU, GPS...) para obtener la matriz de transformación, esto reduciría considerablemente los tiempos de cómputo. También se propone la posibilidad de realizar detección de espacio libre con el mapeado, tarea muy importante en la conducción autónoma. Por último se plantea la opción de comprobar el rendimiento del algoritmo en escenarios abiertos y usar GPUs para el procesamiento, reduciendo así los tiempo de cómputo.

Bibliografía

[1] Ernst D. Dickmanns and Birger D. Mysliwetz. Recursive 3-d road and relative ego-state recognition. IEEE Transactions on Pattern Analysis & Machine Intelligence, (2):199–213, 1992.

[2] Ernst Dieter Dickmanns and Volker Graefe. Dynamic monocular machine vision. Machine vision and applications, 1(4):223–240, 1988.

[3] Charles Thorpe, Martial H Hebert, Takeo Kanade, and Steven A Shafer. Vision and navigation for the carnegie-mellon navlab. IEEE Transactions on Pattern Analysis and Machine Intelligence, 10(3):362–373, 1988.

[4] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. arXiv preprint arXiv:1704.05519, 2017.

[5] Massimo Bertozzi, Alberto Broggi, and Alessandra Fascioli. Vision-based intelligent vehicles: State of the art and perspectives. Robotics and Autonomous systems, 32(1):1–16, 2000.

[6] Paul Furgale, Ulrich Schwesinger, Martin Rufli, Wojciech Derendarz, Hugo Grimmett, Peter Mühlfellner, Stefan Wonneberger, Julian Timpner, Stephan Rottmann, Bo Li, et al. Toward automated driving in cities using close-to-market sensors: An overview of the v-charge project. In Intelligent Vehicles Symposium (IV), 2013 IEEE, pages 809–816. IEEE, 2013.

[7] Lionel Heng, Bo Li, and Marc Pollefeys. Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pages 1793–1800. IEEE, 2013.

[8] Lionel Heng, Paul Furgale, and Marc Pollefeys. Leveraging image-based localization for infrastructure-based calibration of a multi-camera rig. Journal of Field Robotics, 32(5):775–802, 2015.

[9] Christian Häne, Lionel Heng, Gim Hee Lee, Alexey Sizov, and Marc Pollefeys. Real-time direct dense matching on fisheye images using plane-sweeping stereo. In 3D Vision (3DV), 2014 2nd International Conference on, volume 1, pages 57–64. IEEE, 2014.

[10] Christian Häne, Christopher Zach, Bernhard Zeisl, and Marc Pollefeys. A patch prior for dense 3d reconstruction in man-made environments. In 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on, pages 563–570. IEEE, 2012.

[11] Christian Hane, Christopher Zach, Andrea Cohen, Roland Angst, and Marc Pollefeys. Joint 3d scene reconstruction and class segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 97–104, 2013.

[12] Christian Hane, Nikolay Savinov, and Marc Pollefeys. Class specific 3d object shape priors using surface normals. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 652–659, 2014.

[13] Hugo Grimmett, Mathias Buerki, Lina Paz, Pedro Pinies, Paul Furgale, Ingmar Posner, and Paul Newman. Integrating metric and semantic maps for vision-only automated parking. In Robotics and Automation (ICRA), 2015 IEEE International Conference on, pages 2159–2166. IEEE, 2015.

[14] Christian Häne, Torsten Sattler, and Marc Pollefeys. Obstacle detection for self-driving cars using only monocular cameras and wheel odometry. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pages 5101–5108. IEEE, 2015.

[15] Waymo. https://www.waymo.com/. [Web; accedido el 23-08-2018].

[16] Autopilot. https://www.tesla.com/autopilot. [Web; accedido el 23-08-2018].

[17] ELROB − The European Land−Robot Trial | ELROB.org. http://www.elrob.org/. [Web; accedido el 23-08-2018].

[18] I−GAME. Grand Cooperative Driving Challenge 2016 − gcdc.net. http://www.gcdc.net/en/. [Web; accedido el 23-08-2018].

[19] Andreas Geiger, Martin Lauer, Frank Moosmann, Benjamin Ranft, Holger Rapp, Christoph Stiller, and Julius Ziegler. Team annieway's entry to the 2011 grand cooperative driving challenge. IEEE Transactions on Intelligent Transportation Systems, 13(3):1008–1017, 2012.

[20] Gary K Shaffer. Two-dimensional mapping of expansive unknown areas. PhD thesis, Citeseer, 1995.

[21] Javier Gonzalez, Anthony Stentz, and Anibal Ollero. A mobile robot iconic position estimator using a radial laser scanner. Journal of Intelligent and Robotic Systems, 13(2):161–179, 1995.

[22] Ingemar J Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. IEEE Transactions on robotics and automation, 7(2):193–204, 1991.

[23] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. Journal of Intelligent and Robotic systems, 18(3):249–275, 1997.

[24] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. Autonomous robots, 4(4):333–349, 1997.

[25] Chieh-Chih Wang and Chuck Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on, volume 3, pages 2918–2924. IEEE, 2002.

[26] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. International journal of computer vision, 13(2):119–152, 1994.

[27] LiDAR. https://www.fundeu.es/consulta/lidar/. [Web; accedido el 29-05-2018].

[28] HDL-32E. http://velodynelidar.com/hdl-32e.html. [Web; accedido el 29-05-2018].

[29] Keir Thomas. Beginning ubuntu linux: From novice to professional. Apress, 2006.

[30] OpenCV. https://es.wikipedia.org/wiki/OpenCV. [Web; accedido el 30-05-2018].

[31] Point Cloud Library. https://en.wikipedia.org/wiki/Point_Cloud_Library. [Web; accedido el 26-07-2018].

[32] CMake. https://es.wikipedia.org/wiki/CMake. [Web; accedido el 30-05-2018].

[33] The Company. Libraries & APIs, Tools and IDE Qt. https://www.qt.io/qt-features-libraries-apis-tools-and-ide. [Web; accedido el 30-05-2018].

[34] The KITTI Vision Benchmark Suite. http://www.cvlibs.net/datasets/kitti/. [Web; accedido el 02-08-2018].

[35] vicomtech quienes somos. http://www.vicomtech.org/vicomtech-quienes-somos. [Web; accedido el 31-07-2018].

[36] sistemas de transporte inteligentes e ingenieria. http://www.vicomtech.org/d1/sistemas-de-transporte-inteligentes-e-ingenieria. [Web; accedido el 31-07-2018].

[37] David Veneziano, Reginald Souleyrette, and Shauna Hallmark. Integration of light detection and ranging technology with photogrammetry in highway location and design. Transportation Research Record: Journal of the Transportation Research Board, (1836):1–6, 2003.

[38] James W Langston and Tina L Walker. Highway study benefits from helicopter laser survey. Public Works, 131(3), 2001.

[39] LiDAR: Driving the Future of Autonomous Navigation. FROST & SULLIVAN, 2016.

[40] Brent Schwarz. Lidar: Mapping the world in 3d. Nature Photonics, 4(7):429, 2010.

[41] Velodyne LiDAR. http://www.velodynelidar.com/. [Web; accedido el 02-08-2018].

[42] LiDAR Laser Scanners | Product Category | Available now. https://autonomoustuff.com/product-category/lidar/. [Web; accedido el 02-08-2018].

[43] Robosense LiDAR. http://www.robosense.ai/. [Web; accedido el 02-08-2018].

[44] Overview. https://www.ouster.io/product-overview/. [Web; accedido el 02-08-2018].

[45] Quanergy – LiDAR sensors and smart sensing solutions. https://quanergy.com/. [Web; accedido el 02-08-2018].

[46] USER'S MANUAL AND PROGRAMMING GUIDE | HDL-32E High Definition LiDAR™ Sensor. Velodyne LiDAR, Inc, 2012.

[47] VeloView | ParaView. https://www.paraview.org/veloview/. [Web; accedido el 31-07-2018].

[48] Aashish Chaudhary, Matthias Baitsch, and Duy Thanh Truong. Sketchalyze: Structural analysis goes mobile.

[49] Supercomputadora NVIDIA AI para vehículos de conducción autónoma | NVIDIA DRIVE PX 2 | NVIDIA. http://la.nvidia.com/object/drive-px-la.html. [Web; accedido el 29-05-2018].

[50] Kit de desarrollo de software (SDK) para desarrollo automotriz | NVIDIA DriveWorks | NVIDIA. http://la.nvidia.com/object/driveworks-la.html. [Web; accedido el 29-05-2018].

[51] Intempora – RTMaps. https://intempora.com/products/rtmaps#about-rtmaps. [Web; accedido el 31-07-2018].

[52] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In Sensor Fusion IV: Control Paradigms and Data Structures, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[53] James Servos and Steven L Waslander. Multi channel generalized-icp. In Robotics and Automation (ICRA), 2014 IEEE International Conference on, pages 3644–3649. IEEE, 2014.

[54] Point Cloud Library (PCL) JointIterativeClosestPoint function. [Web; accedido el 03-07-2018].

[55] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In Robotics: science and systems, volume 2, page 435, 2009.

[56] Dirk Holz, Alexandru E Ichim, Federico Tombari, Radu B Rusu, and Sven Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. IEEE Robotics & Automation Magazine, 22(4):110–124, 2015.

[57] Erik Zhang. Integration of imu and velodyne lidar sensor in an icp-slam framework, 2016.

[58] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In Conference on Computer Vision and Pattern Recognition (CVPR), 2012.

[59] Greg Turk. The PLY Polygon File Format. https://people.sc.fsu.edu/ jburkardt/data/ply/ply.html. [Web; accedido el 27-08-2018].