

Ingeniería FSAE División Driverless:

Detección con LiDAR y Control en el desarrollo del software para la competición del Upm Racing en FSUK 2020

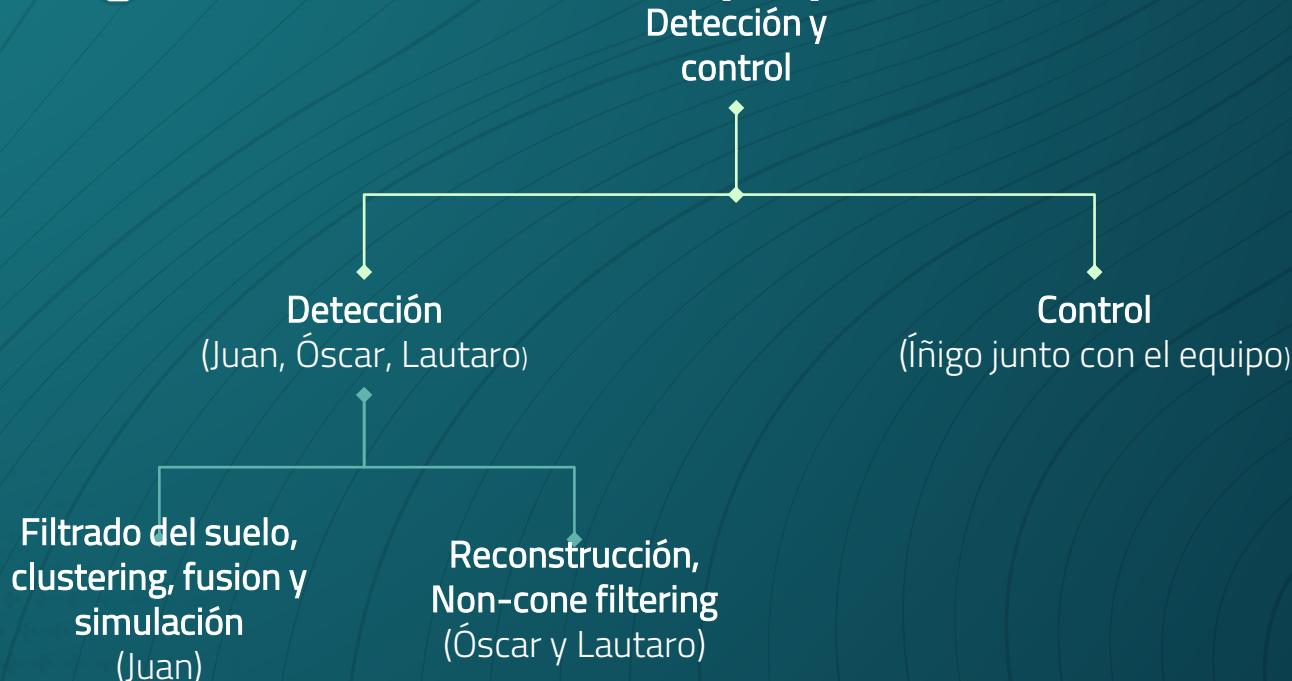


Nos presentamos



Juan Múgica
Óscar Alfageme
Íñigo del Río
Lautaro Di Giusto

Organización del equipo



→ Motivación



Coches

- Había varias opciones de INGENIA
- Desde la UMA por esta asignatura



Automatización

- Varias divisiones dentro del mismo INGENIA.
- Se eligió electrónica/automatización



Programación

- Se eligió la automatización.
- Interés del equipo por la programación.



→ Contexto ¿De qué se disponía?



• Objetivos

Objetivo general:

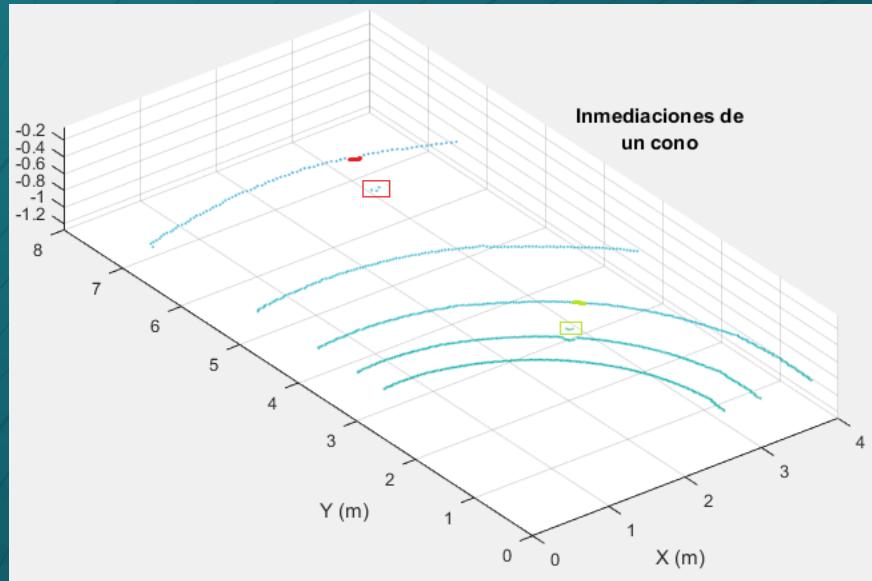
Crear un automóvil autónomo sin conductor con la finalidad de competir en verano de 2020.

Objetivos específicos:

1. Aprender el funcionamiento del LiDAR y el tratamiento de sus datos
2. Aprender el lenguaje de programación Python
3. Representar los datos de entrada
4. Filtrar todos los puntos que no sean objetos
5. Detectar los objetos que haya en la nube de puntos
6. Detectar los que sean parte de los conos
7. Establecer un control longitudinal a velocidad constante y de forma suave.
8. Lograr un control lateral del vehículo que sea estable.
9. Crear una trayectoria virtual a partir de la posición de los conos detectados
10. Leer datos del vehículo proporcionados por el GPS y la IMU, así como la posición de los conos otorgada por la cámara o el Lidar.
11. Ser capaz de enviar las órdenes de giro, freno y acelerador con una frecuencia suficiente para el correcto funcionamiento.



→ Estado del arte – Punto de partida



Fast Segmentation of 3D Point Clouds: A Paradigm on LiDAR Data for Autonomous Vehicle Applications

AMZ Driverless: The Full Autonomous Racing System



→ Estado del arte - Detección



- Menor coste
- Independencia del equipo
- Mejor diferenciación por análisis de color y forma.



Detección con LIDAR

- Mayor coste
- Dependencia del INSIA
- Mayor precisión en la localización de los conos.



→ LiDAR Velodyne vlp-16



- 16 haces de luz pulsada
- $+15^\circ | -15^\circ$
- 600 rpm
- $R_h = 0.2^\circ$
- $R_v = 1.875^\circ$



→ Fases de la detección



Filtrado por distancia

Filtrado por altura

Filtrado de suelo

Ground Filtering - SVD

Distancia = 10 metros

Altura = 1 metro

SVD: 2 iteraciones

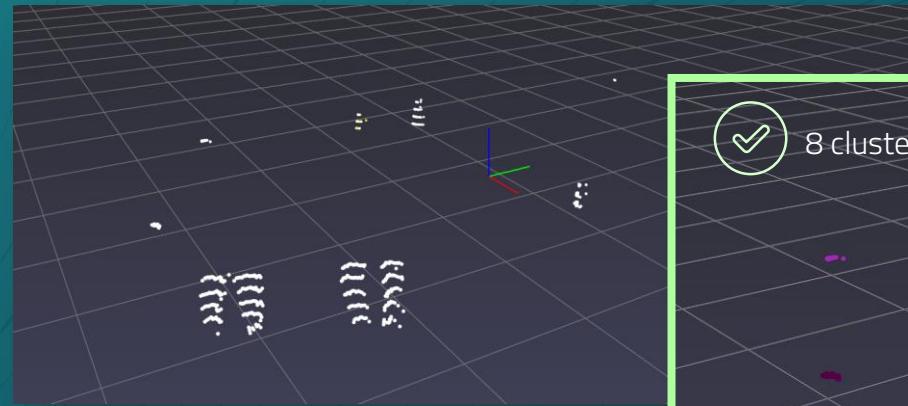


Clustering- DBSCAN

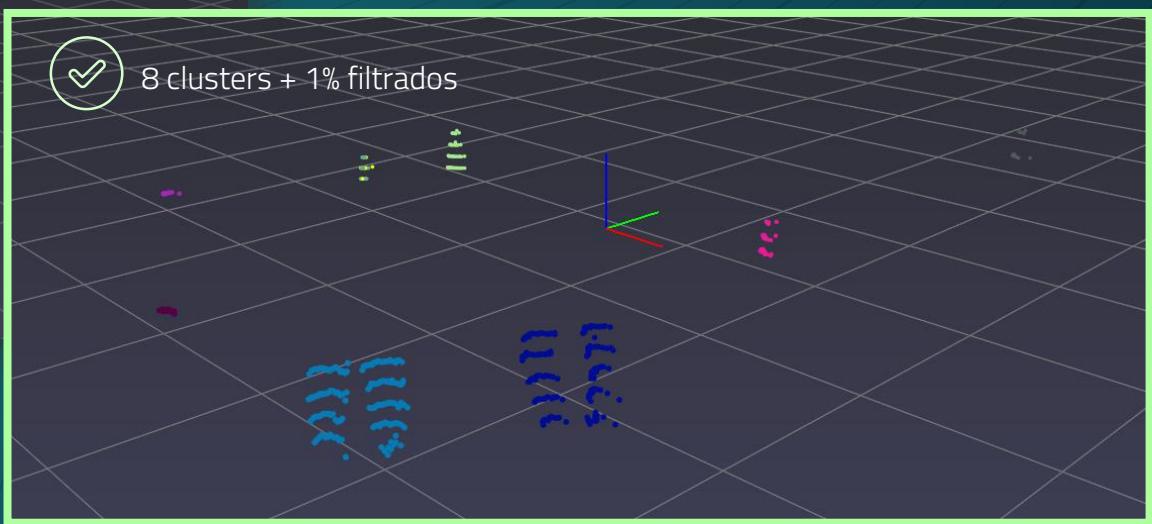
Parámetros:



- Epsilum
- MinPts



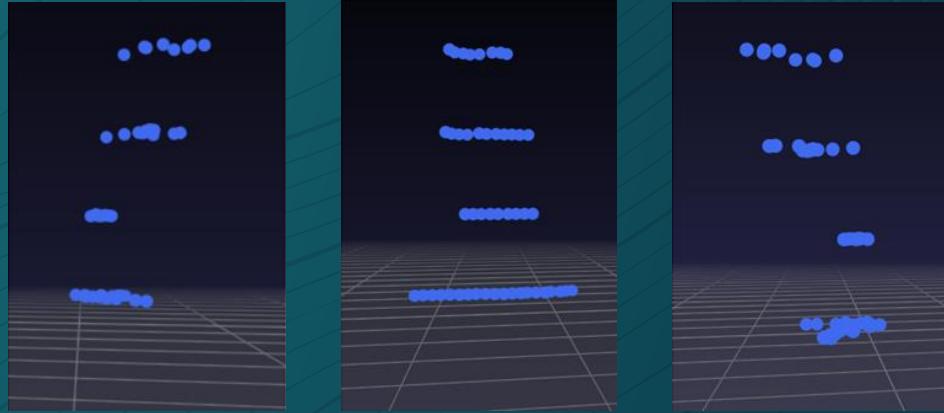
🚩 Salida de Ground Filtering



🚩 Clustering



→ Reconstruction



Librería:



- 🔗 SmallestEnclosingCircle.py
- 🧩 make_circle

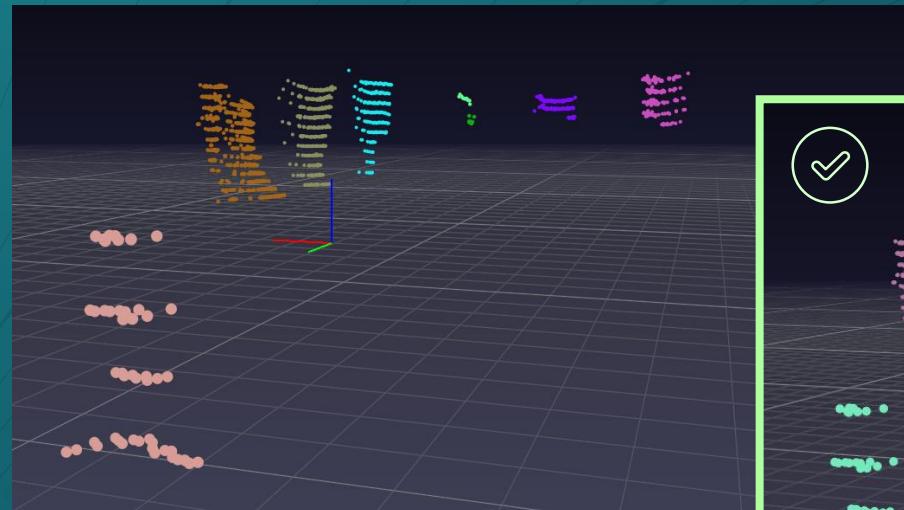


Parámetros:



DistMax

→ Reconstruction



🚩 Salida de Clustering

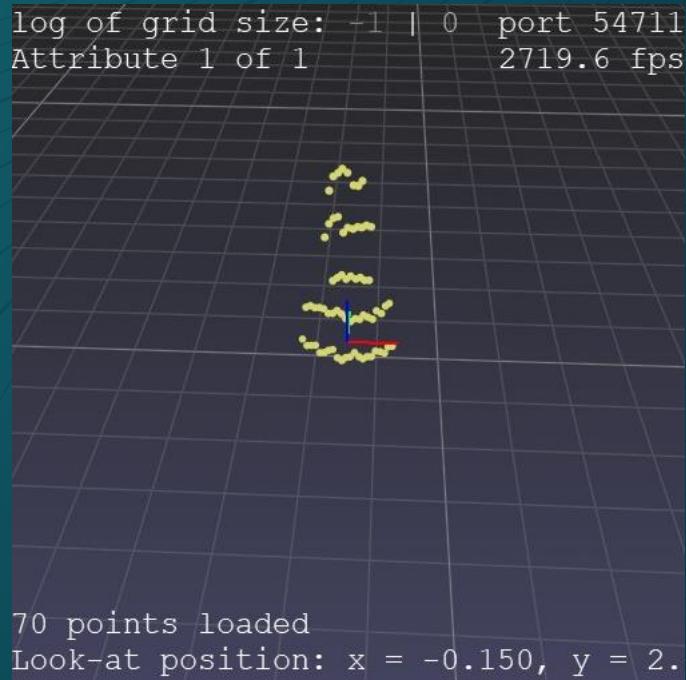
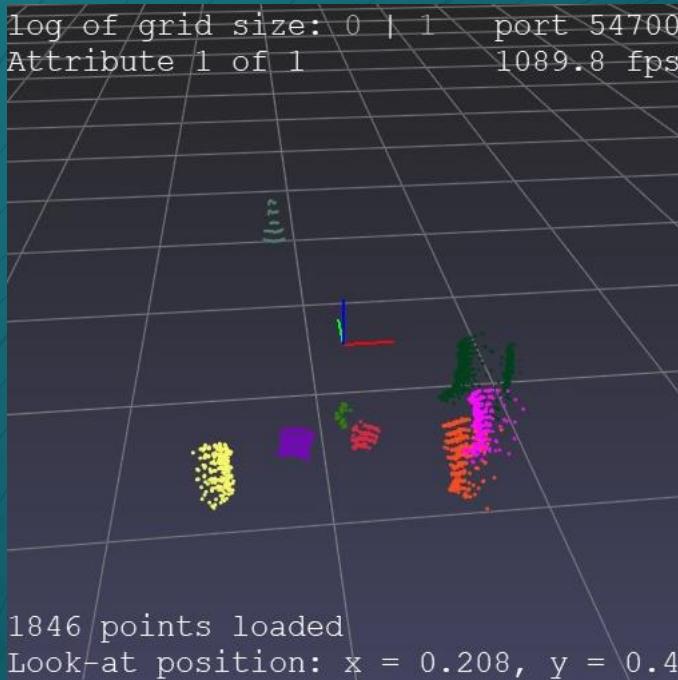


🚩 Reconstruction



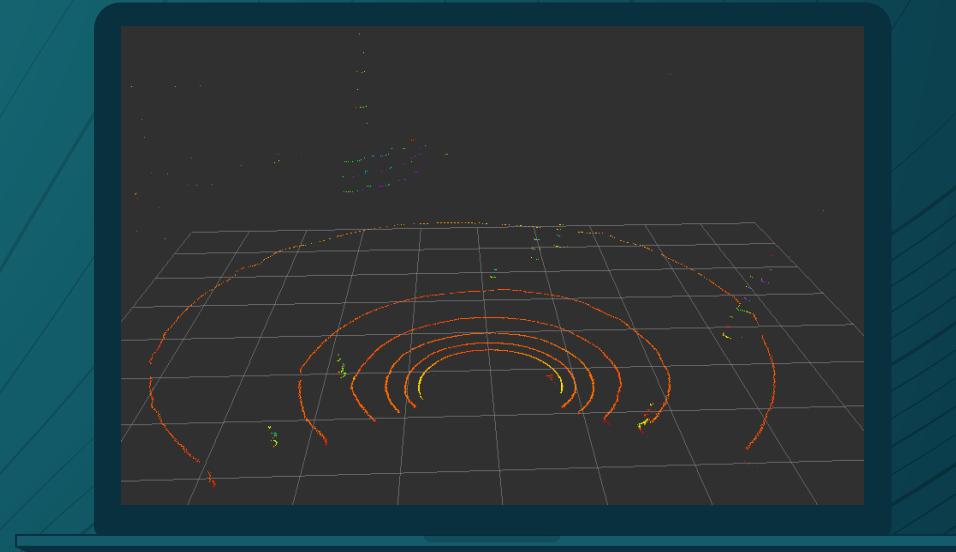
$$expectedpoints = \frac{1}{2} * \left(\frac{h_c}{2 * d * \tan\left(\frac{r_v}{2}\right)} \right) * \left(\frac{w_c}{2 * d * \tan\left(\frac{r_h}{2}\right)} \right)$$

• Non-Cone Filtering



Implementación en ROS2

Detección a tiempo real



ROS vs ROS2

INNOVACIÓN

ROS2 en Windows

ROS2 en Ubuntu

Puente ROS|ROS2

Lenguajes

ROS: Python 2.7 | C++11

ROS2: Python 3.5 | C++13-14

Centralizado vs Distribuido

ROS: centralizado

ROS2: distribuido. Menor dependencia.

Sistemas operativos

ROS: Ubuntu

ROS2: Windows|MacOS|Ubuntu

TCPROS vs DDS

ROS2 más recomendado para aplicaciones en tiempo real.



Datasets

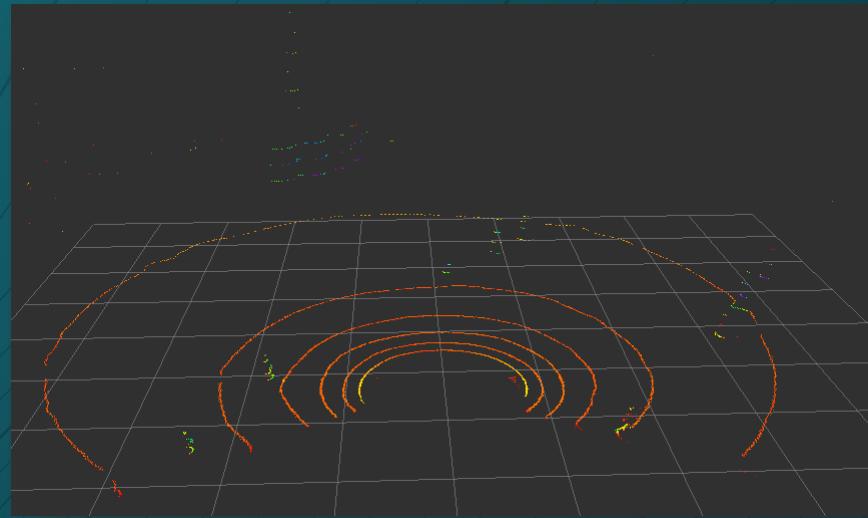
10 Hz
Vuelta completa

INSIA

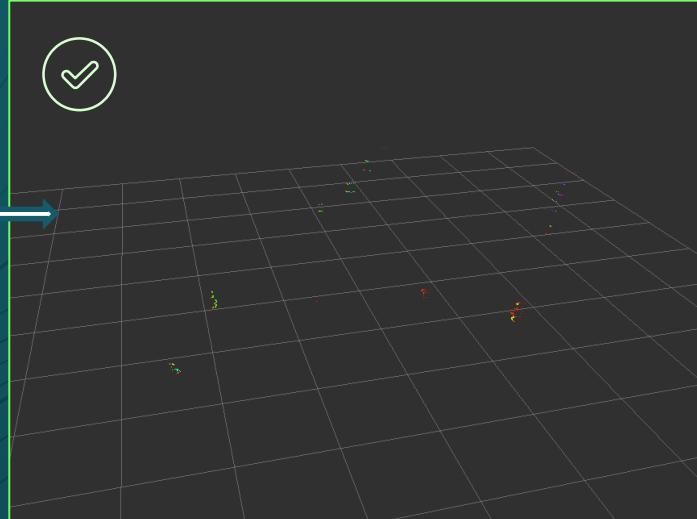
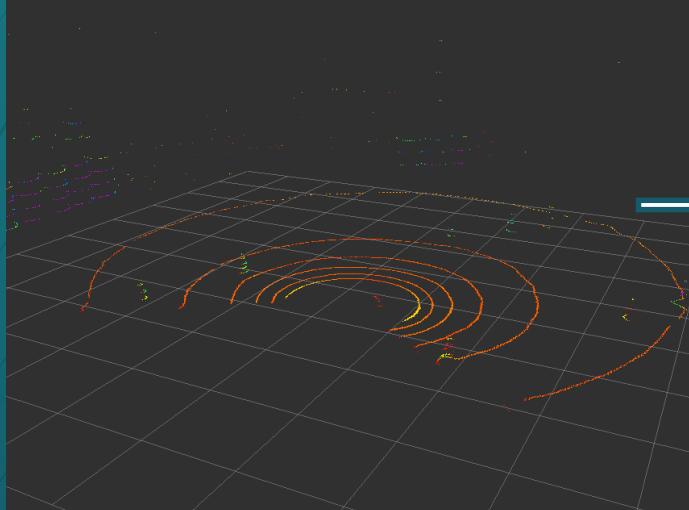
A través del paquete VELODYNE.
Archivo pcap

FSUK

Dataset equivalente al de la competición.
Archivo bag -> ROS|ROS2 bridge -> bag2



→ Ground Filtering



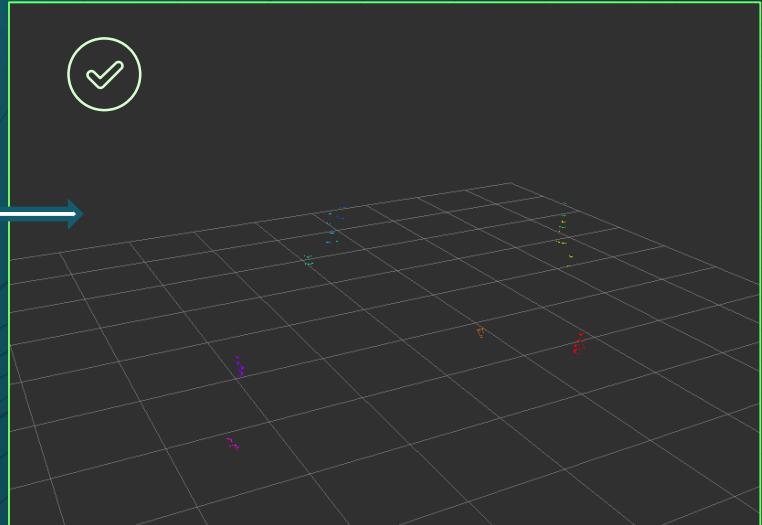
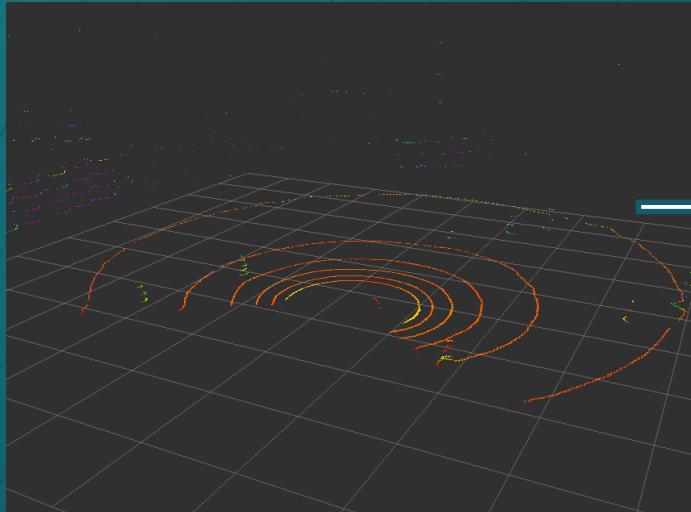
Eliminación total del suelo en gran parte de los frames



Ligera aparición de fragmentos de suelo en 5% de los frames



→ Ground Filtering + Clustering



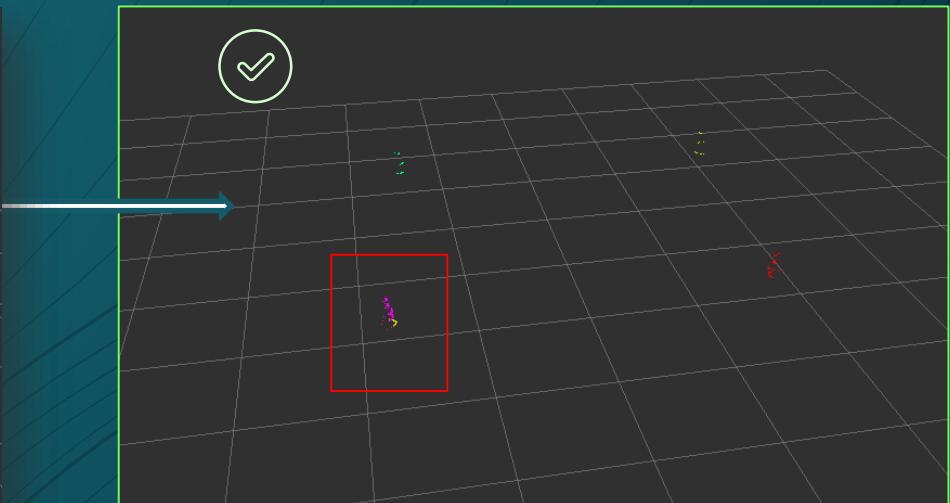
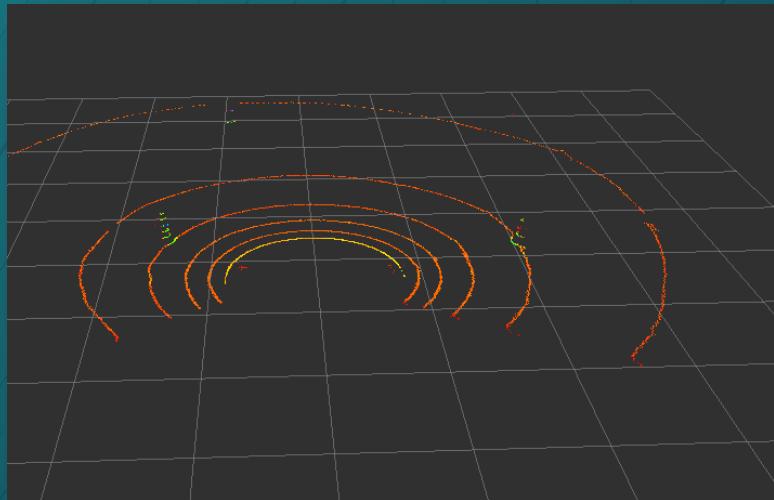
Separación correcta en clusters



Ligera aparición de clusters de suelo en 5% de los frames



→ Ground Filtering + Clustering + Reconstruction



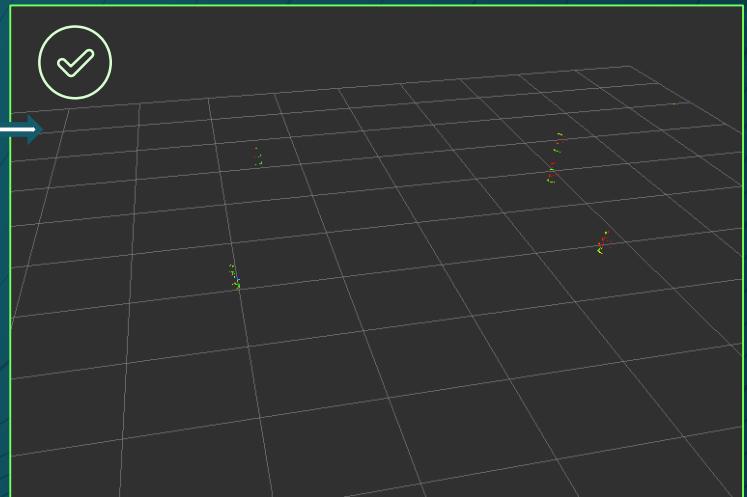
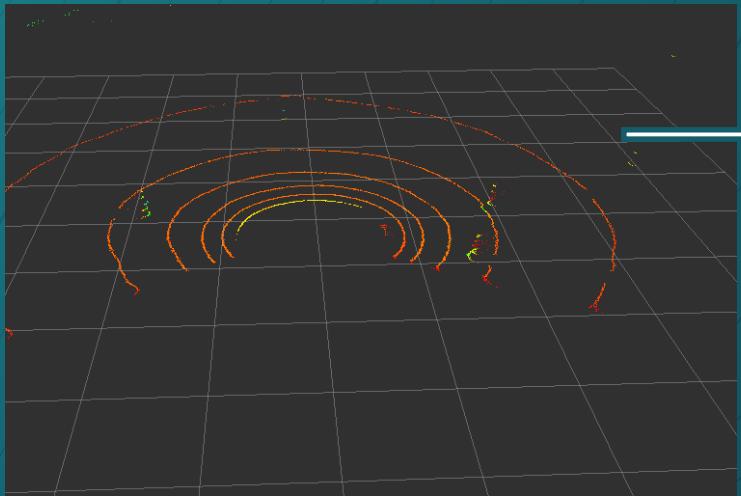
Reconstrucción correcta de los clusters



Continua la ligera aparición de suelo en <5% de los frames



Ground Filtering + Clustering + Reconstruction + Non-cone filtering



Desaparecen por completo los fragmentos de suelo



Detección de conos 0 -> 10m de distancia del Lidar



Desaparición casual de conos entre frames



Funcionamiento del algoritmo completo a 5Hz



Necesidad de testear fuera de la máquina virtual



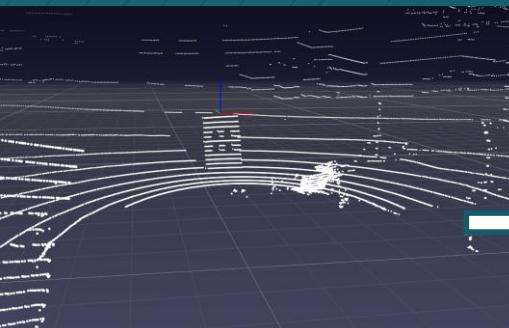
Optimización de parámetros



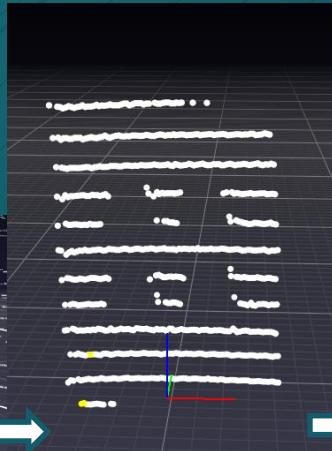
Fusion cámara stereo – Lidar

Calibración extrínseca 3D – 3D

Búsqueda del plano

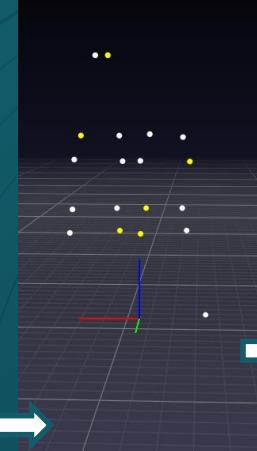


Búsqueda de discontinuidades



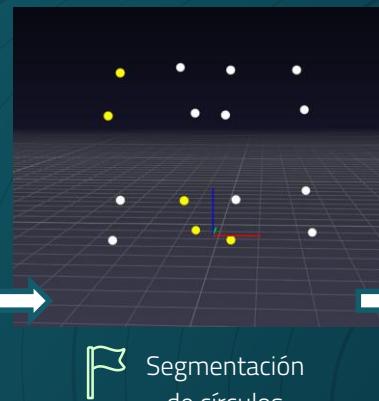
🚩 Búsqueda del
plano

Segmentación de
círculos

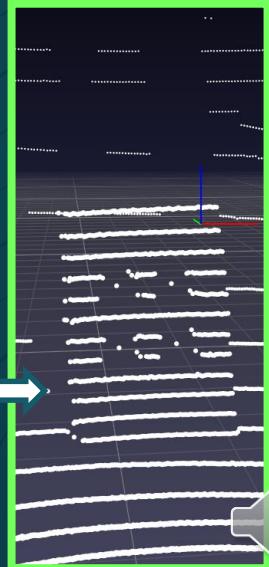


🚩 Búsqueda de
discontinuidades

Obtención de centros



🚩 Segmentación
de círculos





→ Fases del control del vehículo

Establecer un control longitudinal

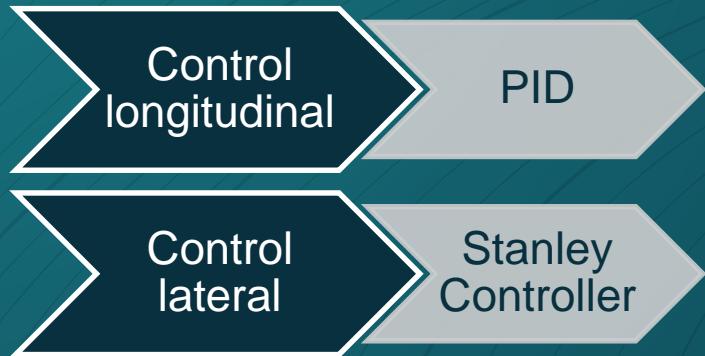
Establecer un control lateral

Crear una trayectoria dinámicamente

Leer datos del vehículo (GPS, IMU)

Sincronizar tanto las lecturas como las órdenes a una frecuencia

Explicación del algoritmo



Explicación del algoritmo [Stanley Controller]

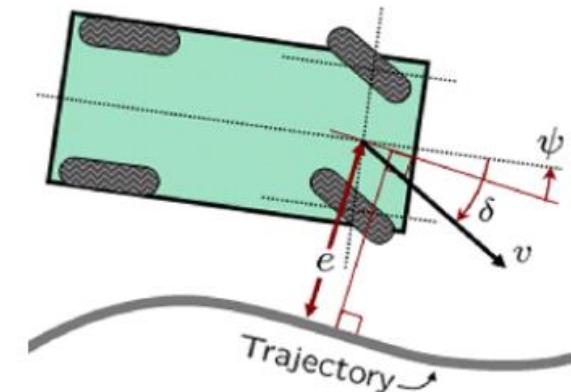
Creación de la trayectoria

Minimización de dos errores:

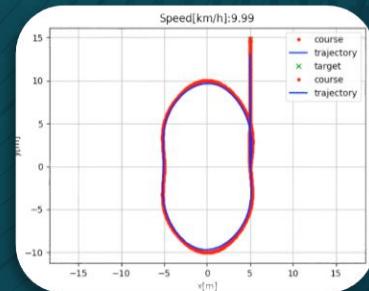
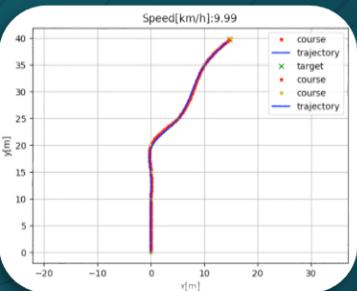
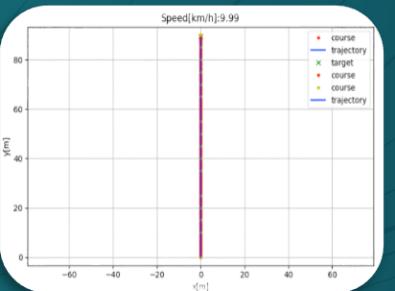
Obtención
siguiente de
guiado

*Heading
error*

*Crosstrack
error*



Simulación



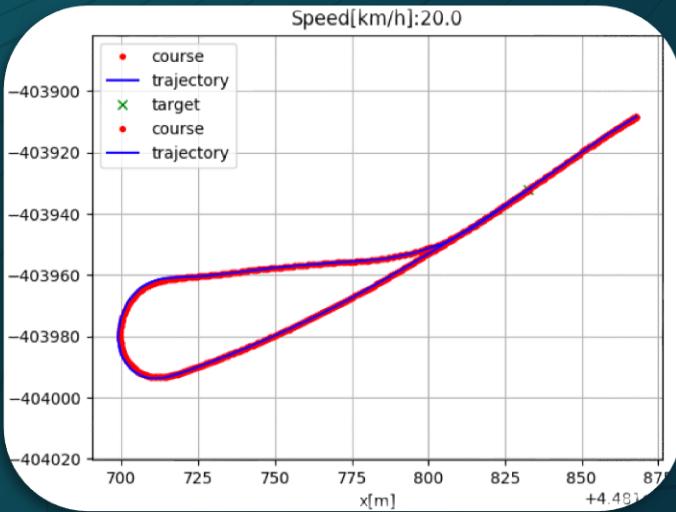
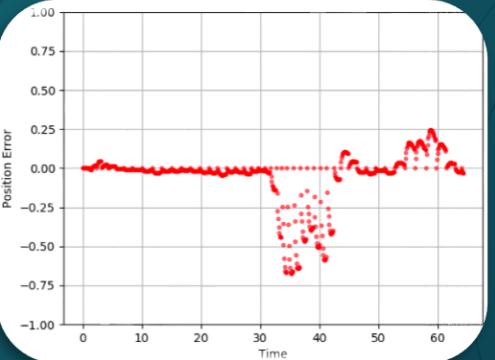
Permite comprobar el funcionamiento con pocos medios

Ensayos de dificultad progresiva

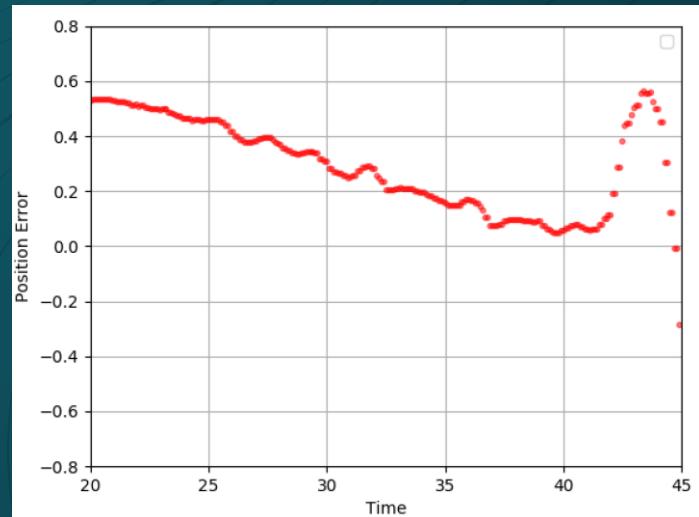
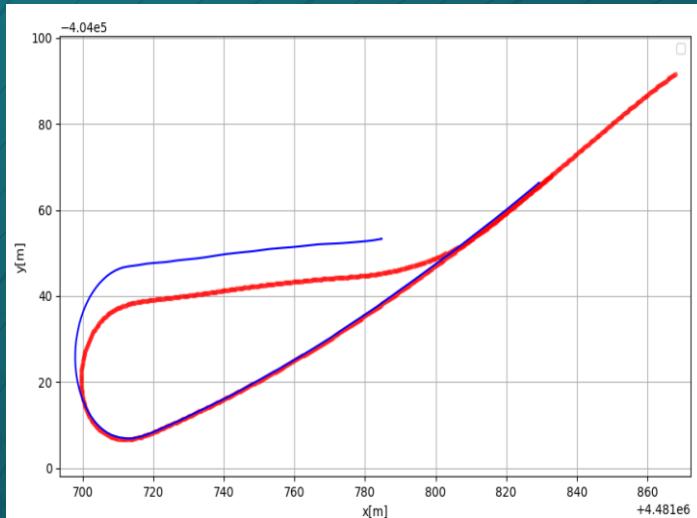
Último ensayo con circuito real



$$MSE=0,05m$$



Prueba real



→ Conclusiones y dificultades encontradas

Detección sólida con Lidar

Se ha logrado una primera versión con gran solidez a falta de la última etapa. Fue complicado definir una estrategia adecuada inicialmente.

Aprendizaje Python

Nivel básico | Nulo al comienzo.
Gran aprendizaje.

Definición del alcance

Supuso una dificultad al principio de organización al partir de cero junto con el UpmRacing

Aprendizaje de ROS|ROS2

Gran aprendizaje para implementar e innovar sobre pseudocódigos de artículos de prestigio a tiempo real

Trabajo con máquina virtual

Rendimiento del algoritmo irreal. Necesidad de un sistema con mejor procesador.

Confinamiento

Ausencia de Lidar y vehículo para tests. En el caso de Lidar se pudo solventar de mejor manera.



→ Líneas futuras

Simulación mejorada

- Pasar de Máquina Virtual a PC con mejor hardware.

Detección de posición de los conos

- ¿Cono a izquierda o derecha?

Optimizar códigos

- Menos recursos.
- Más precisos.

