Navegación Autónoma para vehículos sin conductor

David Trejo [1] Marco Negrete [1]

[1]Facultad de Ingeniería UNAM

Introducción

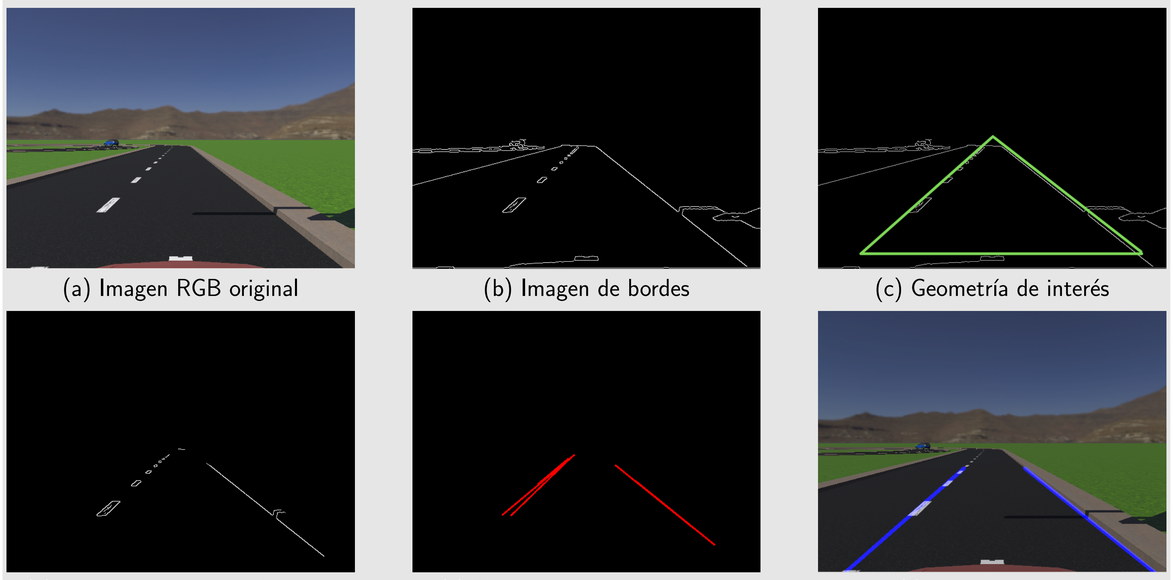
Este póster muestra las herramientas teórico matemáticas necesarias para la elaboración de un sistema de navegación autónoma para vehículos sin conductor usando el simulador **Webots** [Webots n.d.] y la plataforma **ROS**. Con el propósito de crear este sistema se pretenden desarrollar los siguientes módulos para tareas específicas.

- **Detección de Carril** con el uso de detector de bordes de Canny y transformada Hough para procesamiento de imágenes RGB con implementación en lenguaje de programación **Python** y uso de funciones de **OpenCV**.
- **Diseño de leyes de control** para el calculo de movimiento lateral y longitudinal del vehículo (velocidad y ángulo de dirección) implementadas en **Python**.
- **Seguimiento de Carril** en combinación con el resultado de detección de carriles.
- **Detección de Obstáculos** a partir de la implementación de un algoritmo de aprendizaje no supervisado para agrupación de datos desarrollado en lenguaje **C++**.
- **Evasión de Obstáculos** en base a comportamientos definidos en diferentes máquinas de estados implementadas en **Python**.

Detección de Carril

El objetivo de la detección de carriles es obtener una característica particular de la imagen que funcione como guía durante el movimiento del vehículo. Las líneas pertenecientes a los bordes cada línea del carril son el parámetro necesario para mantener al vehículo autónomo dentro del circuito. El proceso para completar esta tarea se enuncia a continuación.

1. Obtener una imagen **RGB** que ilustre la parte frontal del escenario de pruebas.
2. Utilizar el **Detector de bordes de Canny** [Gonzalez 2009] para obtener los bordes de la imagen.
3. Encontrar la geometría indicada para delimitar el área donde se encuentra el carril deseado.
4. Utilizar **Transformada de Hough** [Forsyth and Ponce 2011] para detección de líneas rectas y obtener las líneas que representan los bordes izquierdo y derecho del carril.
5. Obtener una sola línea recta en cada borde del carril expresada preferentemente en su forma polar .



(a) Imagen RGB original (b) Imagen de bordes (c) Geometría de interés

(d) Imagen de bordes delimitada (e) Bordes de carril detectados (f) Líneas promedio

Figure 1. Líneas del carril detectadas por medio del Detector de bordes de Canny y Transformada Hough

Seguimiento de Carril

El seguimiento de carril permite la navegación del vehículo autónomo dentro del escenario de pruebas tomando en cuenta el error que existe entre las líneas detectadas y originales del carril.

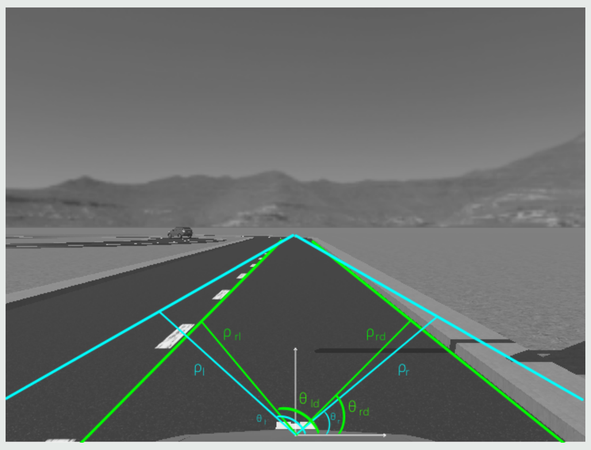


Figure 2. Leyes de Control

Movimiento lateral

Las leyes de control de control para el movimiento lateral del vehículo δ, es decir, el calculo del ángulo de dirección corresponden a las siguientes ecuaciones.

$$\delta = K_\rho e_{\rho p} + K_\theta e_{\theta p}$$

Donde:

$$e_{\rho p} = ((\rho_{ld} - \rho_l + \rho_{rd} - \rho_r)/2) \qquad e_{\theta p} = ((\theta_{ld} - \theta_l + \theta_{rd} - \theta_r)/2)$$

Movimiento longitudinal

El control de velocidad v del vehículo autónomo es constante en todo momento. Es decir, no existe un calculo para este parámetro mientras se encuentre en navegación sin obstáculos, además es establecido al inicio de cada simulación.

$$v = C$$

Detección de obstáculos

Una de las aplicaciones del aprendizaje no supervisado son los algoritmos de clustering. La detección de obstáculos es alcanzada mediante el algoritmo **K-means** [Han et al. 2011], el cual utiliza una técnica de división basada en la distancia entre objetos y permite agrupar los objetos mientras más cercanos se encuentren. La nube de puntos otorgada por el sensor **LIDAR** del vehículo es el conjunto de datos por agrupar para la identificación de objetos.

Detección de obstáculos

- Obtener la nube de puntos del sensor **LIDAR**.
- Definir el número de grupos iniciales.
- Aplicar **K-means** a la nube de puntos.
- Obtener la posición de los obstáculos identificados mediante su centroide.

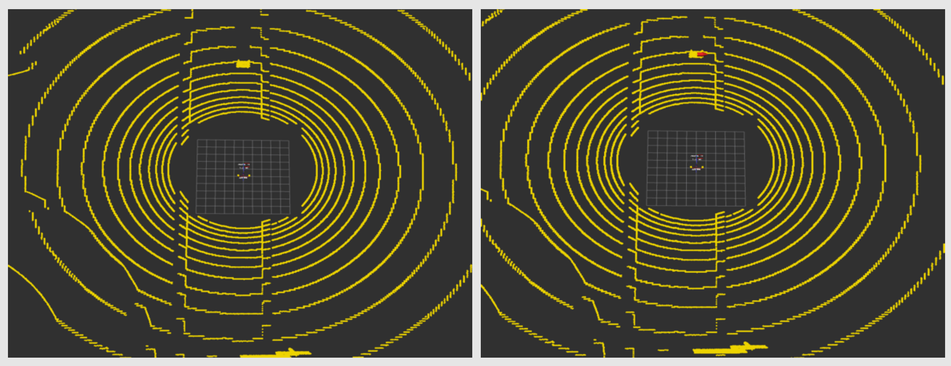


Figure 3. Detección de objetos por medio del algoritmo de clustering **k-means**.

Evasión de Obstáculos

La evasión de obstáculos se logra definiendo dos comportamientos: **Crucero** y **Rebase**. Mientras no se detecten obstáculos el vehículo se mantiene en el estado de **Crucero**, en el caso cuando existan obstáculos en el camino el vehículo pasa al estado de **Rebase**.

Crucero

El comportamiento de crucero se realiza a través del nodo de seguimiento de carril con las mismas leyes de control para movimiento lateral y longitudinal.

Rebase

La acción de rebase es completada por medio de una máquina de estados que calcula las leyes de control para movimiento lateral. Sin embargo, la velocidad del movimiento longitudinal también es constante pero menor que en el caso del comportamiento de Crucero.

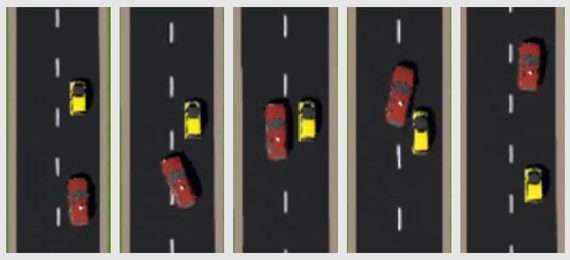


Figure 4. Fases para evasión de obstáculos.

Estacionamiento Autónomo

El estacionamiento autónomo se resolvió mediante el siguiente algoritmo.

1. Avanzar hasta detectar dos obstáculos no contiguos.
2. Frenar.
3. Retroceder un tiempo t con ángulo de dirección θ en sentido horario.
4. Retroceder un tiempo t con ángulo de dirección θ en sentido antihorario.
5. Frenar.
6. Avanzar un tiempo t con ángulo de dirección $\theta = 0$ para alinear.

La velocidad v siempre es constante en todo momento para la acción de estacionamiento autónomo.

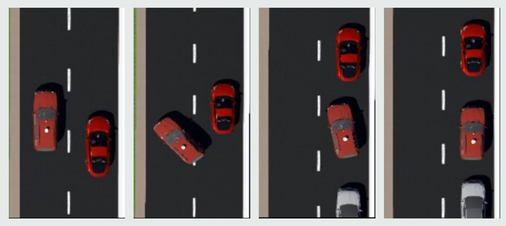


Figure 5. Estacionamiento Autónomo

References

Webots (n.d.). *http://www.cyberbotics.com*. Ed. by C. Ltd. Open-source Mobile Robot Simulation Software. URL: http://www.cyberbotics.com.

Forsyth, D. and J. Ponce (2011). *Computer vision: A modern approach*. Prentice hall.

Han, J., J. Pei, and M. Kamber (2011). *Data mining: concepts and techniques*. Elsevier.

Gonzalez, R. C. (2009). *Digital image processing*. Pearson education india.