UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

MODELAMIENTO SEMÁNTICO DEL ENTORNO PARA LA CONDUCCIÓN AUTÓNOMA DE UN VEHÍCULO TERRESTRE

TESIS PARA OPTAR AL GRADO DE DOCTOR EN INGENIERÍA ELÉCTRICA

FERNANDO JAVIER BERNUY BAHAMÓNDEZ

PROFESOR GUÍA:
DR. JAVIER RUIZ DEL SOLAR SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
DR. MARTIN ADAMS
DR. MIGUEL TORRES TORRITI
DR. EDUARDO MORALES MANZANARES

SANTIAGO DE CHILE
2017

Resumen Ejecutivo

La conducción autónoma de vehículos requiere de un método de localización robusto a cambios en el ambiente debidos, entre otros, a la iluminación natural, los objetos dinámicos presentes y a las condiciones ambientales. Los mapas topológicos permiten una representación concisa del mundo al mantener solo información sobre los lugares relevantes, mientras que los mapas semánticos permiten una representación de alto nivel que incluye etiquetas asociadas con lugares u objetos. Es por esto que en este trabajo se propone el uso de mapas topológicos semánticos como una solución robusta y eficiente para el mapeo en lugares extensos para su uso en la conducción de vehículos autónomos.

El objetivo de esta tesis es modelar el entorno de un vehículo a través de un mapa topológico semántico, utilizando información semántica obtenida del procesamiento de imágenes de la cámara frontal del vehículo, lo que permite generar una descripción de alto nivel del entorno para su uso en la localización de un vehículo autónomo en condiciones de navegación urbanas y rurales.

La metodología para la construcción del Mapa Topológico Semántico utiliza un método de segmentación semántica basado en redes neuronales profundas para extraer la información semántica de las imágenes, y construye el mapa en una estructura de grafo que describe el entorno del vehículo. Para llevar a cabo la localización del vehículo en un mapa topológico semántico conocido se implementó un método basado en filtros de partículas que resuelve exitosamente el problema de localización.

Esta metodología fue probada en una base de datos de conducción urbana, logrando describir 9,57[km] de caminos de ciudad con un grafo que contiene 232 aristas. La metodología de localización fue probada utilizando un segundo recorrido de 8,45[km] obteniéndose una precisión promedio de 7,7[m] en la estimación de la pose del vehículo, mostrando mejores resultados que otros dos métodos del estado del arte. Además, los experimentos realizados demuestran la robustez de la metodología en el caso de no tener una estimación inicial de la pose del vehículo.

"Did I ever tell you you're my hero?
You're everything, everything I wish I could be.
Oh, and I, I could fly higher than an eagle,
For you are the wind beneath my wings,
'cause you are the wind beneath my wings."

Wind Beneath My Wings - Bette Midler

Esta tesis fue financiada parcialmente por el proyecto FONDECYT 1130153
y por la beca de doctorado nacional 21110858 CONICYT

Agradecimientos

Agradecimientos.

Fernando Javier Bernuy Bahamóndez

Tabla de contenido

Índice de tablas

Índice de figuras

Índice de Algoritmos

Capítulo 1

Aportes de la Tesis

El principal aporte de esta tesis es el desarrollo de una metodología para la construcción de mapas topológicos semánticos para la localización de un vehículo autónomo en caminos urbanos y caminos rurales, utilizando información semántica obtenida de las imágenes tomadas por una cámara frontal ubicada en el vehículo.

La estructura de mapa topológico semántico presentada permite representar grandes tramos de caminos a través de un vector de características de alto nivel, que describe el entorno del camino. Esto permite que regiones que son homogéneas desde un punto de vista semántico, sean agrupadas como un segmento único en el mapa, permitiendo cubrir grandes áreas de caminos en forma liviana.

Además, esta metodología no utiliza información de sistemas de posicionamiento global (GPS), por lo que se puede utilizar en ambientes complejos, como en ciudades con edificios altos, túneles, estacionamientos, entre otros.

Para demostrar su utilidad, se desarrolló un método de localización para el mapa topológico semántico que entrega resultados similares a otros métodos del estado del arte al ser probado en una extensa base de datos de conducción en ciudad, tanto en precisión como en tiempo de ejecución. Esto demuestra que la estructura de mapa topológico semántico se puede utilizar para la localización de vehículos autónomos en áreas amplias e independientemente de sistemas de GPS.

En el estado del arte existen otros métodos de construcción de mapas topológicos para la conducción de vehículos autónomos, sin embargo, estos dependen de la existencia de un mapa construido previamente que requiere de mediciones precisas para describir las calles por las que se va a transitar. En la metodología propuesta, el mapa se construye de manera automática, requiriendo intervención humana solo para marcar las intersecciones entre los segmentos del mapa conocidos.

Los aportes secundarios de esta tesis incluyen dos bases de datos etiquetadas para segmentación de caminos no pavimentados, de las cuales una cuenta además con las etiquetas del contenido semántico del entorno, que en conjunto suman más de 10.000 imágenes. Una base de datos de

conducción en ciudad compuesta por dos recorridos sobre un conjunto de calles, utilizadas para la prueba de mapeo y localización. La base de datos urbana consiste en más de 50.000 imágenes etiquetadas.

Para precisar las contribuciones de esta tesis, se destaca a continuación las partes de los distintos módulos que fueron formulados para esta tesis:

- En la percepción semántica descrita en la Sección 4.1, el vector de características, su calculo y sus operadores (Ecuaciones 4.1-3) fueron formuladas para esta tesis, mientras que para la segmentación se utilizó un método propuesto y entrenado por otro autor.

- El método de construcción de mapa, descrito en la Sección 4.2 fue íntegramente formulado para esta tesis, incluidas las Ecuaciones 4.4-6 y los Algoritmos 4.1-2.

- Los métodos de localización presentados en la Sección 4.3 son variantes de métodos ampliamente conocidos, que fueron adaptados para funcionar en mapas puramente topológicos, y con las características diseñadas previamente.

- En la Sección 6.1 se presenta un método de segmentación adaptivo bayesiano basado en histogramas formulado durante el trabajo de doctorado, sin embargo las características de color y texturas utilizadas fueron propuesta en trabajos previos, al igual que el método de segmentación en superpixeles.

- Las variantes de mapeo y localización en caminos no pavimentados presentados en las Secciones 6.2 y 6.3 fueron formuladas como parte de esta tesis.

Capítulo 2

Introducción

2.1. Fundamentación General

La robótica de campo es el área de la robótica especializada en ambientes exteriores, dinámicos y no controlados, y sus aplicaciones están orientadas a la construcción, agricultura, minería, industria forestal, desarrollo militar y aeroespacial, entre otros. Esta disciplina nace como respuesta a las necesidades de la industria de automatizar procesos complejos que no están inmersos en espacios controlados, donde existen componentes dinámicas que podrían alterar el comportamiento del robot, desde el viento o cambios de iluminación, hasta personas u otros vehículos, aumentando la complejidad del diseño al exigir que sea robusto a estos cambios.

Un tema de particular interés en la robótica de campo es la conducción autónoma de vehículos terrestres, cuyo objetivo es remplazar operadores humanos en tareas de conducción. Es sabido que los seres humanos son poco confiables al ejercer tareas repetitivas ejecutadas durante largos periodos de tiempo, debido a que con el tiempo se subestiman los riesgos de los procesos y disminuyen su concentración, por lo que es probable que cometan algún error evitable. En el caso de la conducción de vehículos terrestres la mayor parte de los accidentes son debidos a fallas humanas. Según el Informe Anual 2015 del Carabineros de Chile [25] (Instituto Nacional de Estadísticas (Chile) & Carabineros de Chile, 2015), de un total de 79.880 accidentes de tránsito registrados, 46.822 son atribuibles fallas humanas en la conducción: 2.227 accidentes fueron debidos a adelantamientos indebidos, 36.059 por errores de conducción, 6.793 por desobedecer señalización de tránsito, 1.120 por exceso de velocidad y 2.850 por no respetar el derecho preferente de paso. Por lo tanto, es deseable reemplazar a estos operadores por sistemas autónomos que aseguren una conducción segura, tanto para el vehículo como para su entorno.

Para poder lidiar con un ambiente no controlado y altamente dinámico, es fundamental ser capaz de medir o estimar con alta precisión todos los factores que puedan alterar el comportamiento del robot, lo que hace de los sistemas perceptuales un aspecto vital en la robótica de campo. En el caso

de un vehículo terrestre se suelen utilizar sensores de inercia para caracterizar el desplazamiento del robot, y principalmente sensores de rango para poder modelar el ambiente, encontrar zonas navegables, ubicar obstáculos y otros vehículos, o encontrar puntos de interés para la navegación. Los sensores de rango gozan de una alta precisión en su medición, además de permitir una alta resolución y un largo alcance, por lo que la información obtenida a partir de estos sensores es altamente confiable. Pese a lo anterior, la caracterización del suelo está limitada a una distancia máxima con respecto al vehículo, ya que al aumentar la distancia a la cual se está midiendo, disminuye la densidad de observaciones y por lo tanto empeora la calidad de la representación. Si a esto se le agregan las vibraciones del vehículo, la confidencia de las mediciones cae dramáticamente con la distancia. Esto limita la velocidad máxima de conducción segura en un vehículo autónomo.

En la literatura, la utilización de algoritmos de visión computacional en los sistemas perceptuales en vehículos autónomos se enfoca principalmente en la detección de señalética y las líneas de los caminos pavimentados (en el caso de la conducción autónoma en caminos pavimentados), pero para caminos no estructurados su utilización es experimental y en general el desempeño es menor que el de cualquier sistema basado en sensores de rango. El uso de visión estéreo permite transformar las cámaras en un sensor de rango de corto alcance, permitiendo incluso remplazar el uso de sensores de rango para vehículos pequeños y de baja velocidad, pero sujetos a que el entorno no sea uniforme.

2.2. Definición del Problema

En los últimos años ha habido un creciente interés por las empresas automotrices por lograr la conducción autónoma de sus vehículos. Los esfuerzos por lograr esta tarea han sido claramente liderados por Google, con su exitoso y mediático proyecto de conducción autónoma, y más recientemente por Tesla Motors. Si bien, con la tecnología actual es posible lograr una conducción autónoma segura, existen diversos problemas que requieren mejoras, como lo es la localización del vehículo.

Para que un vehículo sea completamente autónomo, su localización debe ser robusta a cambios en el ambiente, como objetos dinámicos, cambios de iluminación y condiciones atmosféricas. Estos son los principales problemas no resueltos por los métodos actuales de SLAM (Mapeo y Localización Simultanea), según Cadena et al. [16]. De acuerdo a dicho autor, “Para lograr una autonomía de largo alcance (temporal y espacial) aun es necesario una gran cantidad de investigación tanto en técnicas para construir y mantener mapas grandes y variables en el tiempo, como en políticas que definan cuando recordar, actualizar u olvidar información”, debido a los problemas sin resolver como el *aliacing* perceptual, que puede llevar a uniones erróneas de lugares en el mapa, o múltiples representaciones de un mismo lugar, que lleva a un aumento en el consumo de memoria y de tiempo

de procesamiento. De hecho, estos problemas requieren de una comprensión de alto nivel del entorno que hace que los mapas métricos no sean adecuados, por lo que es necesario mapas topológicos y/o semánticos.

Los mapas topológicos permiten una representación concisa del mundo al solo conservar información relacionada con lugares relevantes, y por lo tanto solo es posible localizarse en estos lugares. Numerosos métodos de localización para vehículos autónomos basados en mapas topológicos se han desarrollado [15, 18, 29, 35, 59, 65, 74]. En particular destaca el trabajo de Gadd y Newman [30] en el que se presenta un sistema para construir, manejar y compartir mapas topológicos en una flota de vehículos autónomos.

Un mapa semántico se define como "un mapa que contiene, además de información espacial del entorno, asociaciones entre elementos mapeados y entidades de clases conocidas" [71]. Es decir, un mapa semántico contiene etiquetas asociadas con objetos y lugares que permiten una comprensión de alto nivel del mundo. La construcción de mapas semánticos requiere de un sistema de percepción que pueda convertir observaciones en bruto en abstracciones de alto nivel [44], proceso conocido como percepción semántica.

La generación autónoma de mapas semánticos busca combinar la fortaleza de los métodos de SLAM y la detección de objetos para obtener información semántico-espacial del entorno, para ser utilizada para planificación de tareas o inferir información de lugares no conocidos [32]. Este tema se ha transformado en un tema de investigación muy activo por sus posibles aplicaciones en vehículos autónomos, robot de agricultura y cualquier aplicación de robótica que implique interacción humano robot [53], debido a la necesidad de una operación robusta en ambientes no controlados [78].

Mapeo y localización son tareas fundamentales para vehículos autónomos, y aunque la utilización de GPS es comúnmente aceptada para esta tarea, la calidad del servicio no está asegurada para todos los casos, ya que en túneles y ambientes urbanos no existe una buena cobertura, lo que se traduce en una disminución en la precisión de este sistema [65]. Por otro lado, sensores de rango, como LIDAR y RADAR, has sido ampliamente utilizados por años en proyectos de vehículos autónomos [19, 62, 69, 102], pero recientemente ha habido un significativo aumento en el número de proyectos basados en visión computacional [15, 18, 29, 35, 59, 74], debido al bajo costo de las cámaras, la información densa que entregan y los avances recientes en *Deep Learning* [57], que han mejorado significativamente el desempeño y tiempo de ejecución de los métodos de visión computacional.

De acuerdo a Kostavelis y Gasteros [53], "un desafío para los próximos esfuerzos constituye el mapeo semántico de escenarios abiertos en gran escala", ya que la mayoría de los métodos existentes se enfocan en resolver el problema de mapeo semántico en interiores o ambientes controlados, donde

las separaciones entre lugares están bien definidas, mientras que en exteriores no.

En esta tesis se presenta una solución al problema de la navegación autónoma basado en mapas topológicos semánticos. Gracias a la estructura de grafo del mapa topológico y a la información semántica del sistema de percepción, este mapa logra dividir el camino en segmentos de acuerdo a la interpretación de alto nivel de su apariencia, de manera automática y permitiendo una localización eficiente y precisa. Para esta tesis se utilizan métodos de visión computacional como percepción semántica, particularmente métodos de segmentación semántica, pero es posible utilizar información semántica obtenida a partir de otros sensores, como alguna estimación del ancho, la pendiente o la rugosidad de un camino no pavimentado, que se podrían obtener a partir de sensores de rango o inerciales.

2.3. Objetivos

2.3.1. Objetivo General

Modelar el entorno de un vehículo terrestre a través de un mapa topológico semántico para la localización de un vehículo autónomo, utilizando para dichos efectos información obtenida desde un sistema de visión computacional, e implementar un método de localización basado en este modelo del entorno, que sea robusto y eficiente para la navegación autónoma de vehículos en mapas extensos. Este sistema de visión debe identificar distintos elementos de interés en el entorno (camino, edificios, árboles, peatones u otros obstáculos) utilizando métodos de visión computacional e información de contexto espacial. El mapa generado debe dividir el mundo en segmentos que sean semánticamente similares, para facilitar las tareas de localización.

2.3.2. Objetivos Específicos

- Diseñar y construir bases de datos que permitan evaluar el funcionamiento de los métodos de visión computacional, construcción de mapas y localización.
- Diseñar e implementar métodos de segmentación de caminos basados en características de color y texturas, a nivel de píxel y segmentos.
- Implementar métodos de segmentación semántica que permitan identificar los distintos elementos de interés para la conducción autónoma.
- Generar una descripción semántica, que resuma la información semántica de las imágenes e información de contexto espacial, que permita la construcción del mapa topológico semántico con base a ella.
- Diseñar e implementar un método de construcción de mapas topológicos semánticos del entorno basado en la descripción semántica de las imágenes del vehículo.
- Implementar un método de localización robusto y eficiente que utilice la descripción semántica de las imágenes del vehículo como observación para encontrar la ubicación del vehículo dentro del mapa topológico semántico del lugar por el cual se está navegando.

2.4. Hipótesis

- La utilización de métodos de visión computacional sobre imágenes de una cámara monocular frontal de un vehículo es suficiente para generar una descripción semántica del camino y su entorno, que incluya objetos de interés conocidos para la conducción autónoma, además de su posición con respecto al vehículo.

- Un mapa topológico semántico que representa la navegabilidad de los caminos e intersecciones, divide caminos o calles en segmentos que comparten una descripción semántica similar e incluye información semántica relacionada con los elementos estáticos observados en el entorno; permite representar un conjunto de calles, caminos e intersecciones, de manera que permita la auto-localización en dicho mapa.

- La información semántica obtenida por métodos de visión computacional sobre imágenes de una cámara monocular frontal de un vehículo que navega por una ruta, previamente descrita por un mapa topológico semántico, puede permitir localizar al vehículo en este mapa de manera robusta y eficiente.

Capítulo 3

Revisión Bibliográfica

3.1. Vehículos Autónomos Terrestres

Un vehículo autónomo terrestre, o UGV (Unmanned Ground Vehicle), es un vehículo que está en contacto con el suelo y que se controla autónomamente. La principal tarea de un UGV es percibir su entorno para poder navegar de manera segura. Para esto, el vehículo debe estar equipado con sensores que le permitan medir su entorno a distancia y una estrategia de control que permita lograr llegar a su destino. El primer desarrollo destacado de un vehículo autónomo es Shakey [70], desarrollado al final de la década de 1960 por el Stanford Research Institute. Shakey era un una plataforma con ruedas, equipada con una cámara móvil, sensores de ultrasonido, sensores de tacto y una conexión RF a su computador principal (ver Figura 3.1). Aunque Shakey fue considerado un fracaso en su época, porque nunca consiguió la conducción autónoma, abrió las puertas para la investigación en temas como planificación de trayectorias y visión computacional para la conducción autónoma [31]. En la actualidad, el problema de los vehículos autónomos pequeños en ambientes domésticos se considera resuelto, incluso es posible adquirir aspiradoras robóticas que son capaces de reconocer su entorno y recorrerlo completamente para aspirar la suciedad del piso (ver aspiradora Roomba en la Figura 3.1). El desafío actual está en sacar los robots de los ambientes controlados, como una casa o un laboratorio, y usarlos en ambientes exteriores y no controlados. Un caso particular es la

Figura 3.1: Ejemplos de vehículos autónomos. Shakey a la izquierda, y Roomba a la derecha.

Figura 3.2: Stanley, el robot ganador del DARPA Grand Challenge.

automatización de automóviles.

Desde la creación del Grand Challenge [46] de la *Defense Advanced Research Projects Agency* (DARPA) en el año 2002, ha aumentado notoriamente el interés de la comunidad científica en el desarrollo de vehículos terrestres autónomos en ambientes no controlados. Si bien, a la fecha de la primera competencia en el año 2004 existían numerosas publicaciones enfocadas a resolver este problema, ninguno de los 15 equipos participantes logró superar un 5 % de la ruta propuesta, una ruta a través del desierto de Mojave, siendo Sandstorm [96] el que llegó más lejos. Esto mostraba que los algoritmos propuestos y probados en simuladores no se comportaban de acuerdo a lo deseado en el mundo real. Para la segunda versión de dicha competencia, en el año 2005, cinco equipos lograron llegar a la meta, y 22 de los 23 equipos participantes recorrieron una distancia mayor que la lograda por el mejor competidor del año pasado. Los cinco robots que llegaron a la meta son, en orden de llegada, Stanley [91] (ver Figura 3.2), Sandstorm [101], H1ghlander [101], Kat-5 [93] y TerraMax [13].

Un enfoque de particular interés en el desarrollo de este tipo de tecnologías es la conducción autónoma en ciudades y carreteras, donde se busca reemplazar la conducción humana por una conducción autónoma segura. El primer gran proyecto en esta área es Prometheus [103] de la organización EUREKA, y fue desarrollado entre los años 1987 y 1995. Con sus vehículos VaMP y VITA-2 (ver Figura 3.3), lograron manejar en autopistas alcanzando velocidades de 130 km/h en condiciones normales de tráfico y con una muy baja intervención humana. En 1995 hicieron un viaje desde Múnich hasta Copenhaguen, ida y vuelta, a través de autopistas, logrando velocidades de 175 km/h y distancias de hasta 158 km sin intervención humana. En la actualidad ha destacado el proyecto del vehículo sin conductor de Google [90] (ver Figura 3.4), del cual han producido una

Figura 3.3: Prometheus VaMP (izquierda) y VITA-2 (derecha).

flota que en conjunto ha conducido más de 480.000 kilómetros de manera completamente autónoma y sin accidentes. Otro proyecto que ha destacado es el vehículo autónomo RobotCar UK [64] de la Universidad de Oxford (ver Figura 3.4), el cual está diseñado para navegar de manera autónoma por lugares que haya recorrido con anterioridad. La principal fortaleza de este vehículo es que su navegación está basada principalmente en un sistema visual para la localización y la detección visual de obstáculos. Adicionalmente cuenta con un láser situado debajo del parachoques que le permite encontrar obstáculos de manera más robusta, y dos láseres de 270° ubicados adelante y atrás del vehículo, haciendo un barrido ortogonal a la trayectoria, para tareas de mapeo. Diversos fabricantes de autos han mostrado sus nuevas tecnologías en conducción autónoma para sus productos, como aparcamiento automático, sistemas que toman el control en caso de emergencia y sistemas autónomos capaces de seguir a otro vehículo en una autopista.

3.2. Sistemas Sensoriales en Vehículos Autónomos

Los sistemas actuales de conducción autónoma de vehículos terrestres requieren de un sistema perceptual preciso y seguro, que le permita tomar decisiones correctas y bien informadas a los mecanismos de toma de decisiones. Por ejemplo, el vehículo de Google (ver Figura 3.4) basa su sistema sensorial en el *Velodyne 64-beam laser*, un sensor de rango basado en láseres que permite un campo de visión de 360° con una resolución de 0.09° horizontalmente, y un campo de visión de 26,8° con una resolución de 0.4° verticalmente, y una distancia máxima de 50 metros para la medición de pavimento y 120 metros para elementos más reflectantes, como otros vehículos. Además de este sensor, utilizan cámaras y radares para detección de señalética y obstáculos. La medición del Velodyne es comparada con una base de datos previamente construida, y procesada a través de los servidores de Google, para lograr las tareas de localización y planificación de trayectorias [90].

Las principales desventajas del proyecto de Google son: El costo de los sensores, la necesidad

Figura 3.4: Un vehículo de Google (izquierda) y el Robotcar UK(derecha).

de una conexión al servidor y la necesidad de una base de datos previamente construida. El costo del Velodyne es muy elevado para una aplicación final destinada al publico general. La necesidad de una conexión a los servidores de Google impide que se pueda navegar por lugares sin una buena conectividad. Además, la utilización de una base de datos previamente construida impide que se utilice para explorar nuevos caminos, y en particular rutas no pavimentadas.

Robotcar UK basa su sistema sensorial en un sistema de cámaras estéreo y tres láseres, siendo una solución de bajo costo de implementación. El sistema utiliza un mapa a priori que incluye una representación 3D del entorno, un mapa semántico con señalética e información de tránsito, y una base de datos con las imágenes obtenidas en las distintas posiciones. El vehículo extrae elementos de interés de las imágenes y los compara con los elementos descritos en el mapa a priori y genera un mapa tridimensional del ambiente con las medidas de láseres ubicados ortogonales al sentido del vehículo, uno en el parachoques frontal y otro en el trasero. El tercer láser se ubica en el parachoques y está orientado a la parte frontal del vehículo y su objetivo es la detección de obstáculos. Al pasar el vehículo por una zona conocida previamente, compara los elementos de interés observados con los del mapa a priori y a partir de ellos estima su posición y actualiza el mapa en caso de ser necesario. La principal desventaja de este proyecto es la utilización de un mapa a priori, ya que esto obliga a que cualquier ruta o camino que se vaya a visitar deba ser analizada previamente y, por lo tanto, no permite exploración.

En el caso de Stanley, el ganador del DARPA Grand Challenge, el sistema perceptual está compuesto por un GPS, cinco láseres ubicados en el techo e inclinados a distintos ángulos para caracterizar el suelo y el entorno del vehículo, dos radares para la detección de obstáculos masivos y una cámara para la detección del camino [91]. En este vehículo no existe un mapa a priori del entorno y su objetivo está dado por puntos en el GPS, por lo que la navegación está orientada a evadir obstáculos, manteniendo una orientación general definida por el GPS. A partir de los láseres se genera un modelo del entorno y se detecta el camino a seguir, el cual se utiliza para entrenar un de-

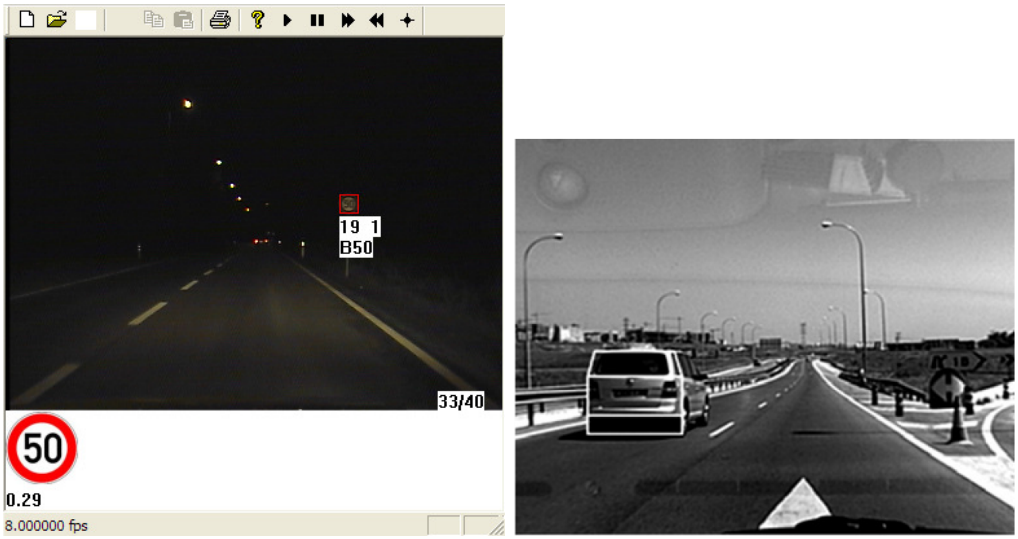


Figura 3.5: Deteccion de señalética [5](izquierda) y de otros vehículos [1](derecha).

tector visual del camino. Este detector permite proyectar la detección del camino de los sensores de rango a una distancia mayor, permitiendo así una conducción autónoma segura a mayor velocidad. Este enfoque permite una navegación segura en un entorno desconocido siguiendo una ruta definida por un camino no pavimentado. Sin embargo, la cantidad de sensores utilizados para modelar de manera segura el entorno implica un elevado costo económico y computacional. Además, al ser un sistema basado en GPS la localización del robot y de los *checkpoints* que marcan la ruta están sujetos a un error que podrían convertirse en comportamientos no deseados e incluso accidentes.

3.3. Visión Computacional para Vehículos Autónomos

Resulta intuitivo pensar que, dado que los humanos son capaces de conducir utilizando principalmente el sentido de la visión, es posible conseguir que un vehículo se maneje de manera autónoma utilizando visión computacional. Es por esto que existe una amplia gama de desarrollos orientados a esto.

La señalética de calles públicas representa información importante con respecto al comportamiento que debe seguir el vehículo para conducir por ellas, por lo que es fundamental que un vehículo sea capaz de reconocer estos elementos en su entorno. Una de las características principales de la señalética de tránsito es que deben ser llamativas a la vista y llamar la atención del conductor, es por esto que la mejor opción para la detección de la ésta es a través de visión computacional. Si bien el principal enfoque en el desarrollo de detección de señalética es en la asistencia al conductor [5,66], debido a que los grandes fabricantes de autos están interesados en aumentar la seguridad de sus modelos, la mayoría de estos desarrollos pueden ser utilizados de igual manera como información

Figura 3.6: Imagen de la competencia DARPA Urban Challenge.

para la conducción autónoma de un vehículo. Un ejemplo es el trabajo de Bahlmann et al. [5] (ver Figura 3.5), donde utilizan una etapa de detección y tracking basada en Adaboost, wavelets de Haar sensibles al color y propagación de información temporal, seguido de un clasificador bayesiano. La principal falencia de este trabajo es que está basado en características de color, por tanto llevan a que el método sea inestable a cambios de iluminación. En el caso del trabajo de Mogelmose et al. [66], utilizan un modelo de detección de figuras geométricas dentro de la imagen, para luego identificar en ellos la señalética representada. Este enfoque hace que el sistema sea robusto a cambios de iluminación, pero resulta sensible a oclusiones o a señalética en mal estado. Sin embargo, es posible suponer que la señalética está en buen estado y a la vista.

Otro elemento de interés dentro de la conducción es el tráfico en el entorno del vehículo. El tráfico representa un problema altamente dinámico, y es crítico para la seguridad vial. Es por esto que los métodos para detección de vehículos se basan principalmente en sensores de rango, como en el trabajo de Wender & Dietmayer [100] (2008) donde se utilizan lecturas de láser para detectar la presencia y orientación de otros vehículos, y basados en esa información se busca en una región de interés de la imagen con un detector visual para la orientación del vehículo. En dicho trabajo se propone disminuir la región de búsqueda para mejorar el tiempo de procesamiento del algoritmo utilizando un láser, mientras que utilizando información de contexto es posible efectuar el mismo trabajo, haciendo innecesaria la incorporación de sensores de rango. Otro ejemplo es el trabajo de Petrovskaya & Thrun [76] (2009) en el que se utiliza un Velodyne y un láser para medir el entorno y generar sensores virtuales que contienen la ubicación de las detecciones de los vehículos, obteniendo detecciones precisas y confiables, pero a un elevado costo computacional y económico.

La detección de calzada es otro aspecto fundamental para la conducción autónoma de un vehícu-

lo, ya que éstas indican la ruta que se debe seguir localmente de manera segura. En el caso de las calzadas en caminos pavimentados, las calzadas están delimitadas por líneas continuas o discontinuas de colores que contrastan con el color oscuro del pavimento. El principal enfoque para la detección de las líneas de las calzadas es a través de métodos de visión computacional [45, 89], aunque el uso de sensores de rango y sistemas de posicionamiento global son complementos importantes. El desarrollo de los métodos de detección de calzada está impulsado por su uso en sistemas de asistencia al conductor [63, 89], como avisos de salida de calzada, control de crucero adaptivo, entre otros. En el caso de la detección de calzada en la conducción autónoma en ciudad, los equipos participantes en el DARPA Urban Challenge (ver Figura 3.6) han demostrado en la practica el uso de visión computacional para esta tarea [3, 52, 67, 95].

La detección de calzada para caminos no pavimentos o no estructurados resulta una tarea muy diferente al caso anterior, ya que no existen delimitaciones claras y muchas veces corresponden a la huella dejada por los vehículos que han transitado anteriormente. Estas calzadas suelen tener características similares a las de su entorno, por lo que la diferenciación visual es más compleja. Existen trabajos que utilizan características de color para diferenciar el camino de su entorno basándose en el supuesto que el terreno frente al vehículo es parte del camino [10, 48, 107] o utilizando información de sensores de rango para discriminar qué zonas de la imagen corresponden al camino [91]. Otros trabajos utilizan características de textura para encontrar el punto de fuga en la imagen, para luego localizar los bordes del camino [50] o una región del camino que apunta a dicho punto [81]. Otra solución al problema de la detección de caminos es la utilización de sensores de rango [42, 75] (ver Figura 3.7) o cámaras estéreo [37, 38, 40, 80] para caracterizar el camino, que permiten una conducción segura pero que limita la distancia máxima a la cual las mediciones son confiables y así la velocidad máxima a la que se puede desplazar el vehículo. En el caso de las cámaras estéreo, un problema de particular interés es el caso de las regiones uniformes o sin textura, en las que los algoritmos de calce entre las imágenes cometen grandes errores, lo cual es resuelto en el trabajo de Guo et al. [37] (2012) a través de un análisis de las regiones sin textura en las imágenes antes de la reconstrucción 3D.

Los trabajos enfocados en modelar el camino utilizando exclusivamente el color en imágenes deben lidiar con numerosos problemas, como cambios de iluminación y la variabilidad de colores que presenta un camino, mientras que los trabajos que han modelado el camino utilizando texturas deben lidiar con la exigencia de tiempo de ejecución que esto presenta. Si bien ambos enfoques permiten buenos resultados, la utilización de información de bajo nivel no representa completamente el camino y por tanto los resultados serán limitados en casos de cambios ambientales, o en la estructura del

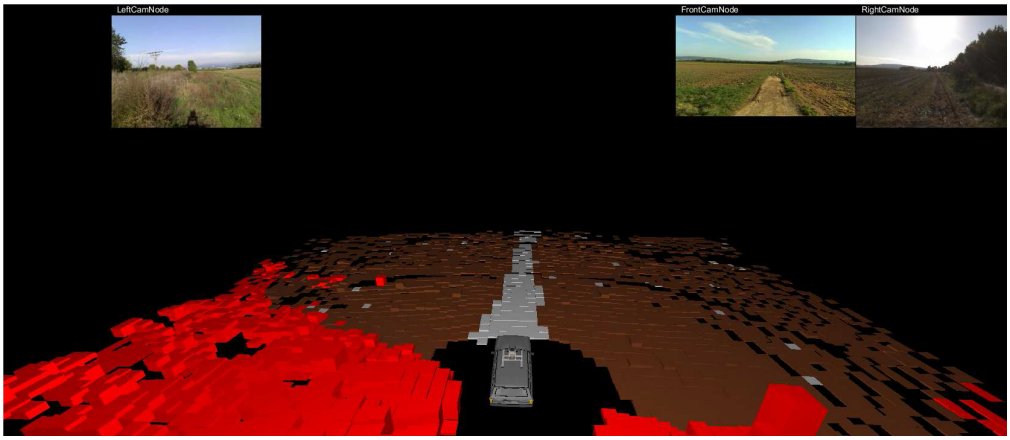


Figura 3.7: Segmentación de camino utilizando sensores de rango e información visual [42].

camino. Un ejemplo de utilización de información de alto nivel es la búsqueda del punto de fuga en la imagen [50, 51], que ha mostrado excelentes resultados en condiciones ambientales muy distintas. Sin embargo, estos trabajos están sujetos a que la estructura del camino esté claramente definida a través de sus bordes, y que la curvatura de estos sea baja, por lo que no representan una solución completa a la detección de caminos. Finalmente, los trabajos que utilizan información de distancia, ya sea a través de láseres o cámaras estéreo, tienen una alta confiabilidad a una distancia reducida, la cual decae rápidamente con la distancia.

3.4. Segmentación no Supervisada

En visión computacional, segmentación no supervisada de imágenes es el proceso de dividir una imagen digital en múltiples segmentos (grupos de píxeles o súper-píxeles) sin información previa del contenido de la imagen. El objetivo de la segmentación es simplificar y/o cambiar la representación de una imagen en algo más significativo y fácil de analizar [84]. Cada píxel de cada segmento es similar al resto del segmento de acuerdo a algún criterio, como color, intensidad o textura.

Diversos métodos de segmentación de imágenes se basan en métodos de clustering, dada que la naturaleza de ambos problemas es el mismo: agrupar datos similares. El trabajo de Comaniciu & Meer [20] (2002) presenta un algoritmo de clustering basado en la estimación de la función de densidad a partir de los datos y a partir del gradiente de la función de densidad estimada. En el mismo trabajo se presenta su aplicación a problemas de segmentación de imágenes. El trabajo de Pappas [72] (1992) presenta un método adaptivo de clustering basado en k-means que incluye las restricciones espaciales que representan las imágenes en los datos. En un trabajo posterior [17] incluye la utilización de información de color y textura.

En el trabajo de Guo, Yamabe, & Mita [38] (2012) se busca el borde que mejor delimita el camino

del resto del entorno a través de la solución de un problema general de optimización en un Modelo Oculto de Markov (HMM por sus siglas en inglés), asociado a un mapa semántico de las imágenes de un par de cámaras estéreo de un vehículo. Para esto construyen un mapa semántico utilizando una estructura de grafo, construida a partir de una segmentación no supervisada de una de las imágenes. Para segmentar la imagen utilizan el algoritmo descrito por Felzenszwalb & Huttenlocher [28] (2004), que mide la disimilitud entre dos segmentos contiguos a través de un análisis del borde entre ellos, y si son lo suficientemente parecidos entonces los une. Tras la segmentación se utiliza información de profundidad obtenida a partir de la homografía entre las imágenes estéreo. Esto permite que, para cada segmento de la imagen que representa un nodo en el grafo del mapa semántico, se tenga la información de color y una estimación de qué tan cercano es al plano en el que se encuentra el camino.

3.5. Mapas Semánticos

En el contexto de la robótica móvil, se define un mapa semántico como "un mapa que contiene, además de información espacial del entorno, asociaciones entre los elementos mapeados y clases conocidas anteriormente" [71], es decir, un mapa semántico tiene etiquetas asociadas a objetos y a lugares (mesa, pared, cocina, jardín, etc.). Mapeo semántico es el proceso de construir mapas semánticos y percepción semántica se define como el proceso de convertir observaciones sensoriales en información de la categoría de los objetos o lugares observados [44].

Mapeo y percepción semántica son áreas de investigación de alto interés para la robótica. Algunas razones por las cuales estos temas son de particular interés para la comunidad científica son: la necesidad de una operación más robusta en ambientes no controlados y la necesidad de una ejecución más eficiente de las tareas [78].

La generación de mapas semánticos busca combinar las fortalezas de las técnicas de SLAM (Simultaneous Localization and Mapping) y de las técnicas de categorización de objetos para crear conocimiento semántico-espacial del entorno, el cual puede ser usado para planificación de tareas o inferencias sobre regiones no exploradas [32]. Los métodos actuales están basados en la construcción de mapas y en la aplicación de categorización de objetos en tiempo real para agregar los objetos, sus categorías y las relaciones entre éstas en el mapa, el cual está representado a través de una estructura de jerárquica, generando así mapas con significado semántico.

En el trabajo de Galindo et al. [33] (2005) se utilizan dos mapas jerárquicos para representar la información semántica, un mapa jerárquico espacial y uno jerárquico conceptual, en los cuales la profundidad representa distintos niveles de abstracción del contenido semántico. Las relaciones

entre ambos mapas están definidas por anclajes, lo que son definidos por el usuario. El mapa semántico métrico-topológico es construido utilizando operaciones morfológicas sobre el mapa. En otros trabajos [55, 68, 106] se construye una representación multicapa para representar el entorno, basada en un mapa métrico, un mapa de navegación, un mapa topológico y un mapa conceptual. El mapa métrico está basado en un método de EKF SLAM basado en láser, la clasificación de lugares está basada en características geométricas simples obtenidas de las lecturas del láser, y la detección de objetos usa descriptores SIFT [61]. Para el mapa conceptual utiliza relaciones del tipo "es un" para describir elementos, o "tiene un" para describir relaciones. Estas relaciones son designadas manualmente.

En el trabajo de Douillard, Fox, Ramos & Durrand-Whyte [27] (2011) se construye un mapa semántico a partir de *Conditional Random Fields* (CRF). Este mapa es representado como un conjunto de nodos con un estado oculto que corresponde a una posición y una categoría. Para este trabajo se utilizan siete categorías (auto, tronco, follaje, persona, pared, césped y otros) y se utilizan características visuales y de un sensor de rango.

En el trabajo de Sengupta, Strurgess & Torr [83] (2012) se genera un mapa semántico topológico denso de un ambiente urbano utilizando un vehículo equipado con seis cámaras orientadas en diferentes posiciones. Cada imagen se segmenta de manera no supervisada y cada segmento generado se etiqueta en una de las trece categorías definidas utilizando CRF. Luego, se proyectan las imágenes al suelo, suponiendo un mundo plano y se define una etiqueta final a partir de las lecturas observadas utilizando CRF, generando un mapa etiquetado del suelo visto desde arriba. Su trabajo posterior [82] utiliza un par de cámaras estéreo para reconstruir la superficie del entorno y así reemplaza el supuesto de un mundo plano, llevando las etiquetas de las imágenes a la superficie para construir un mapa visto desde arriba con las etiquetas finales.

El modelo de mapa semántico denso de Sengupta et al. [82, 83](2012, 2013) contiene un nivel mínimo de información semántica, la cual es la etiqueta correspondiente a cada celda. Además, la estructura utilizada no permite una búsqueda rápida de los elementos, por lo que cualquier tarea de navegación sería igualmente costosa que el calce entre mapas topológicos construidos a partir de sensores 3D. Es por esto que este modelo no es de utilidad si lo que se desea es utilizar información de alto nivel del entorno. El caso del modelo propuesto por Douillard et al. [27](2011) es similar al anterior, desde el punto de vista que su contenido corresponde a las etiquetas y a la ubicación de los distintos elementos, por lo que no es recomendable su utilización para tareas enfocadas al alto nivel de información. Pero a diferencia del modelo anterior, este modelo está basado en una estructura de grafos, por lo que su utilización para tareas de localización y mapeo es más eficiente. Por último,

los modelos de mapa semánticos que incluyen un mapa semántico espacial y un mapa semántico conceptual, junto con distintos tipos de relaciones entre ambos mapas, permiten la inclusión de información de alto nivel en su construcción, además de una rápida búsqueda de los distintos elementos.

3.6. Deep Learning

Deep Learning es un campo del aprendizaje de máquinas que intenta aprender abstracciones de alto nivel de un conjunto de datos utilizando una estructura jerárquica [39]. *Deep Learning* aprende representaciones de datos con múltiples niveles de abstracción utilizando modelos computacionales compuestos de múltiples capas de procesamiento. Los métodos basados en *Deep Learning* han mejorado drásticamente el estado del arte en el reconocimiento voz, el reconocimiento de objetos, la detección de objetos y muchas otras aplicaciones. Las *Deep Convolutional Networks* (redes convolucionales profundas o DCN) han producido importantes avances en el procesamiento de imágenes, video, voz y audio [57].

Los métodos convencionales de *machine-learning* están limitados por requerir un pre-procesamiento de los datos, ya que por décadas los métodos de reconocimiento de patrones han requerido un proceso cuidadoso de diseño de métodos de extracción de características (como color y textura) para la creación de vectores de características apropiados para su uso en un clasificador estadístico.

Los métodos basados en *Deep Learning* son capaces de descubrir automáticamente las representaciones necesarias para la clasificación utilizando representaciones con múltiples niveles, donde cada nivel es una composición de módulos no lineales simples que transforman la representación del nivel anterior a una representación con un nivel de abstracción más alto. Luego, con suficientes niveles, es posible aprender funciones muy complejas y de alto nivel de abstracción, que permiten amplificar los aspectos importantes a distinguir y suprimir la información irrelevante. Un ejemplo simplificado sería que en una imagen, que es representada como un arreglo de pixeles, las características aprendidas en el primer nivel representarían la presencia o ausencia de bordes en una orientación y ubicación determinada, mientras que un segundo nivel buscaría patrones formados por combinaciones específicas de bordes, un tercer nivel buscaría combinaciones de patrones de bordes que correspondan a partes de objetos conocidos y los niveles subsiguientes detectarían objetos como combinaciones de las partes del nivel anterior. Lo importante de los métodos de *Deep Learning* es que las características generadas por estos niveles no son diseñadas, sino que son aprendidas a partir de los datos utilizando un método de aprendizaje de uso general.

Existe un tipo particular de redes neuronales profundas que son muy fáciles de entrenar: las

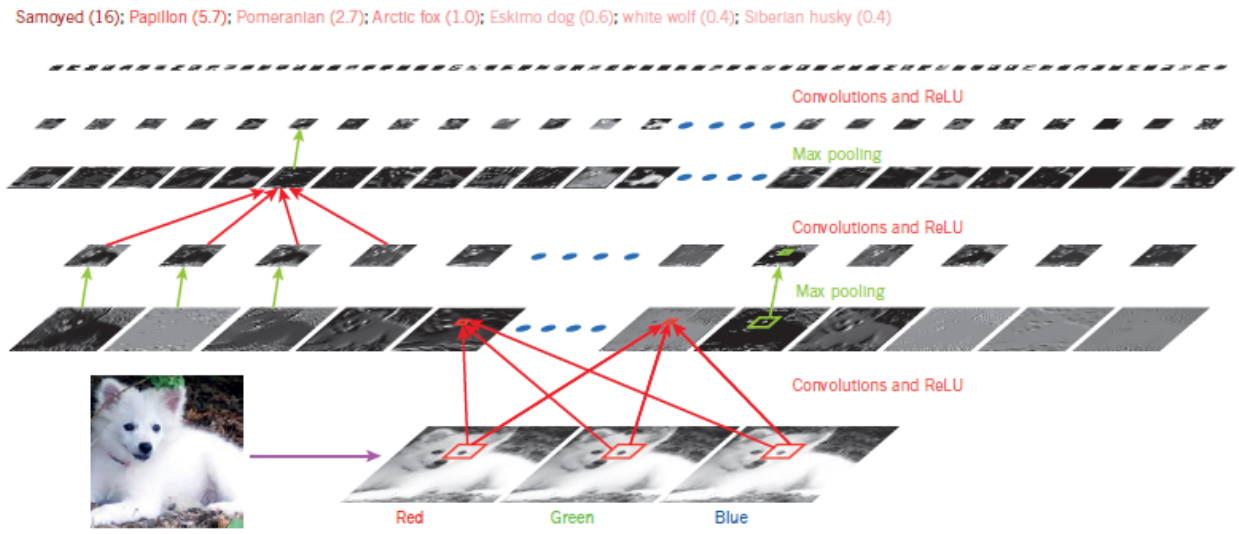


Figura 3.8: Estructura de la red ConvNet [57].

Convolutional Neural Networks (CNN). Este tipo de redes han tenido un gran éxito en aplicaciones prácticas durante el tiempo en el que las redes neuronales no parecían una solución efectiva, y recientemente han sido ampliamente adoptadas por la comunidad de visión computacional.

Las CNN están diseñadas para procesar datos que vienen en forma de tensores, como una imagen a color que se compone de tres arreglos bidimensionales que contienen la intensidad de pixel de cada canal de color. Algunos ejemplos de datos en forma de tensores son: arreglos unidimensionales para señales y secuencias, como audio o texto; arreglos bidimensionales para imágenes o espectrogramas de audio y arreglos tridimensionales para secuencias de video o imágenes volumétricas.

Los cuatro principales conceptos que utilizan las CNN para aprovechar las propiedades naturales de las imágenes son: (i) conectividad local, (ii) pesos compartidos, (iii) *pooling*, y (iv) el uso de muchas capas. La Figura 3.8 muestra una arquitectura típica de una CNN, que está estructurada como una serie de etapas consecutivas. Las primeras etapas están compuestas por capas convolucionales y capas de *pooling*, y suelen ser seguidas por una etapa de clasificación que contiene capas completamente conectadas.

Una capa convolucional contiene unidades de información que se organizan en *feature maps*. Una unidad de información está conectada con parches de los *feature maps* de la capa anterior a través de un conjunto de pesos llamado banco de filtros. La suma pondera del resultado del banco de filtros se pasa a través de una función no lineal, como una *ReLU*, para obtener el valor de la unidad de información. Todas las unidades de un *feature map* comparten el mismo banco de filtros, y distintos *feature maps* usan distintos bancos de filtros. La arquitectura de una capa convolucional se sustenta en dos razones: Primero, grupos locales en arreglos de datos suelen estar altamente correlacionados

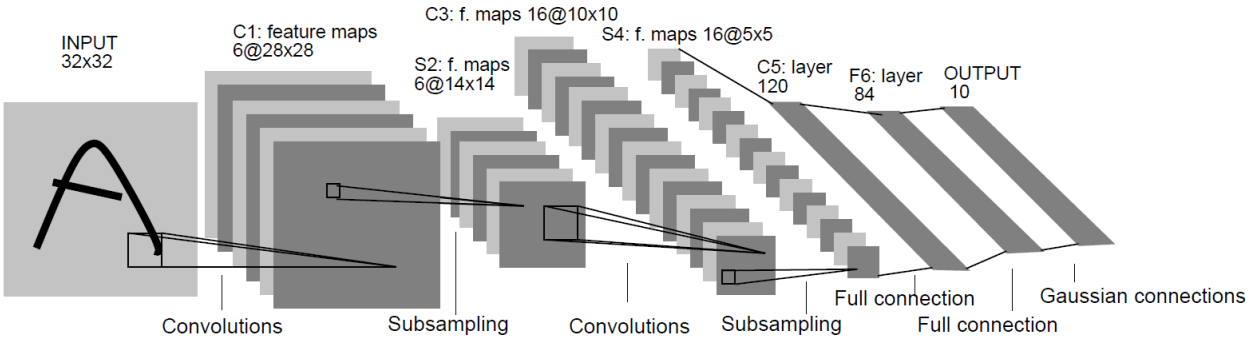


Figura 3.9: Estructura de la red LeNet [58].

formando patrones fácilmente reconocibles, y segundo, estos patrones locales suelen ser invariantes al a posición en la imagen, es decir que un patrón dado puede aparecer en cualquier parte de la imagen, por lo que la idea de que unidades de información que representan distintas regiones de la imagen compartan el mismo peso hace que un *feature map* represente un patrón determinado en la imagen.

El rol de una capa de *pooling* es unir características similares para permitir que los patrones de alto nivel sean robustos a pequeñas variaciones. Esto se puede lograr relajando la posición de cada característica, lo que típicamente se hace a través del cálculo del máximo en un vecindad de unidades de información en un *feature map*. Adicionalmente, estos parches se obtienen con un paso mayor a uno, lo que permite reducir el tamaño de la representación y darle invarianza a ruido y a pequeños cambios en la posición.

Esta estructura de red permite la utilización de *backpropagation* para el entrenamiento de los pesos de los bancos de filtros.

Las redes convolucionales aprovechan la composición jerárquica que comparten muchas señales naturales, donde los patrones de alto nivel son composiciones de patrones más simples. Por ejemplo, en imágenes las combinaciones de bordes forman patrones, combinaciones de patrones forman partes y combinaciones de partes forman objetos. De la misma manera se puede analizar el reconocimiento de voz, donde combinaciones de fonemas forman sílabas, las que forman palabras, que forman frases.

Reconocida como una de las primeras redes convolucionales *deep* en demostrar resultados exitosos, *LeNet-5* aparece como una estructura de red pionera en la clasificación de dígitos manuscritos. Esta estructura de red (ver Figura 3.9) fue creada por Yann LeCun [58], uno de los principales precursores de las redes convolucionales. *LeNet-5* está compuesta por 7 capas: (i) Convolucional+ReLU, (ii) *pooling*, (iii) Convolucional+ReLU, (iv) *pooling*, (v) completamente conectada, (vi) completamente conectada, y (vii) conexiones gaussianas que representan la salida de la red. Esta red usa imágenes de entrada de 32x32 pixeles y retorna uno de los 10 dígitos posibles. Esta red convolu-

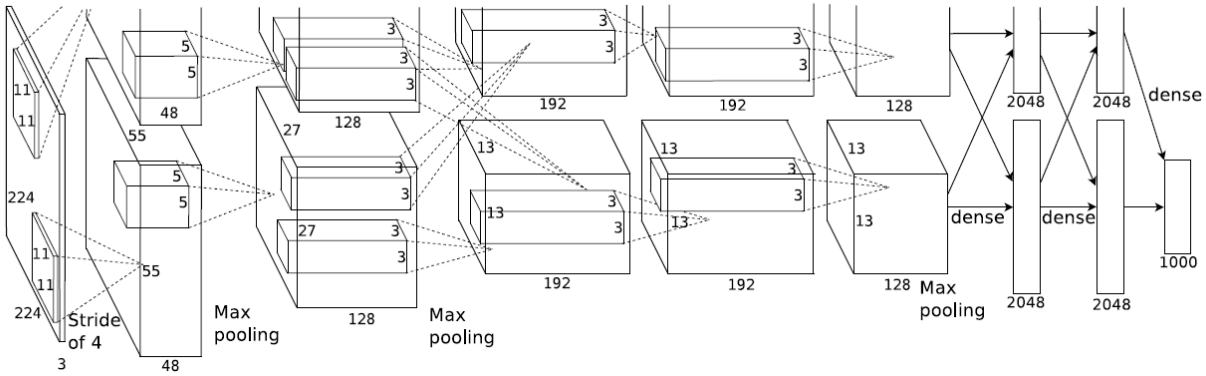


Figura 3.10: Estructura de la red AlexNet [54].

cional, además de obtener resultados destacados en la base de datos MNIST, fue comercialmente usada para el reconocimiento de letras manuscritas en cheques bancarios.

Alex Krizhevsky et al. [54] presentan en el 2012 la red convolucional pionera de *Deep Learning* conocida como *AlexNet*, ya que superó significativamente los resultados obtenidos por otros métodos de visión computacional en la ILSVRC 2012 (ImageNet Large-Scale Visual Recognition Challenge), una competencia anual de visión computacional en la que equipos de todo el mundo compiten en tareas de clasificación, localización, detección, entre otras. Los autores presentan resultados que superan por amplio margen a los mejores resultados obtenidos hasta ese momento: Según el error Top-5, AlexNet consiguió un 15,4 %, versus un 25,2 % de su competidor más cercano. Estos resultados impactaron a la comunidad científica de visión computacional, generando el estallido de las redes convolucionales en todas las competencias y tareas relacionadas con el procesamiento de imágenes. Cabe destacar que, desde entonces, los mejores resultados en estas competencias son obtenidos con redes convolucionales: Matthew D. Zeiler & Rob Fergus [105] obtuvieron el primer lugar de la ILSVRC 2013 con la red denominada ZF-Net, con una tasa de error de 11,2 %, reafirmando el desempeño de las redes convolucionales; el ganador del ILSVRC 2014, conocido como GoogLeNet [88], obtuvo una tasa de error de 6,7 % y en el ILSVRC 2015, el primer lugar fue para la arquitectura ResNet [43] de Microsoft Research Asia que obtuvo un 3,6 % con su variante de 152 capas de profundidad.

En 2014, Karen Simonyan y Andrew Zisserman de la Universidad de Oxford [86] presentaron una arquitectura de red compuesta exclusivamente por filtros de tamaño 3x3 y capas de *maxpooling* de tamaño 2x2, conocida como VGG. La configuración de esta arquitectura que consiguió los mejores resultados, VGG16 con un 7,3 % de tasa de error, está compuesta por 16 capas: 13 convolucionales y 3 completamente conectadas (ver Figura 3.11). Esta arquitectura se extiende en la profundidad de

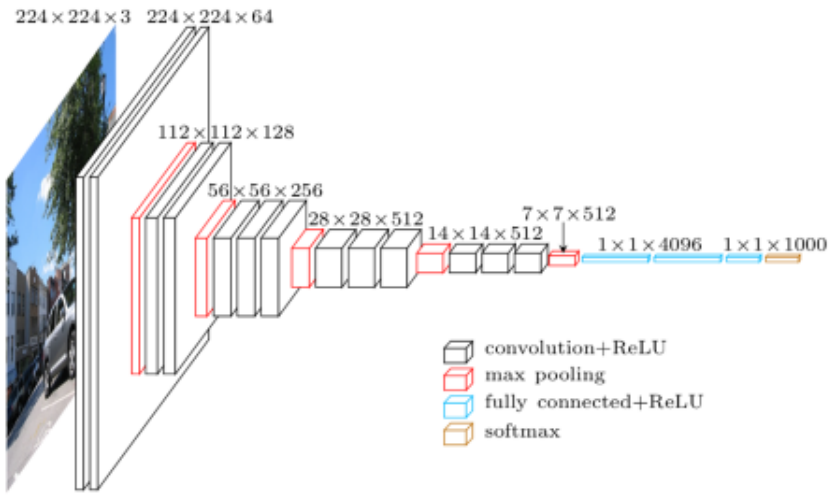


Figura 3.11: Estructura de la red VGG-16 [86].

la estructura de la red, y por lo tanto aumenta el nivel de abstracción utilizado para la clasificación, mientras mantiene unidades funcionales pequeñas (de 3x3). Una de las razones que explican los autores para justificar su estructura es que dos capas convolucionales con filtros de 3x3 tienen el mismo campo receptor efectivo que una capa convolucional de 5x5, pero aumentando la no-linealidad al incluir dos funciones *ReLU*. Este arquitectura ha sido ampliamente utilizada como base para el desarrollo de metodos de segmentacion semántica, como FCN [60], SegNet [4] y DilationNet [104]

Todos los trabajos de redes convolucionales presentados hasta ahora tienen como objetivo entregar una clase para una imagen de entrada. El paso siguiente a esta tarea es la de asignar una clase por cada pixel en la imagen, a lo que se le llama segmentación semántica (ver Figura 3.12). Los trabajos de segmentación semántica previos a AlexNet, y por tanto al *boom* del *Deep Learning*, no lograban cumplir satisfactoriamente con la tarea: trabajos como Textonboost [85] y JGMT [26] presentaron resultados prometedores, pero fueron rápidamente superados por trabajos basados en *Deep Learning*.

La primera red convolucional relevante para la segmentación semántica fue la *Fully Convolutional Network* (FCN) [60]. Los autores presentan una metodología que permite obtener una imagen de baja resolución con el resultado del clasificador para las ventanas correspondientes al tamaño del clasificador (ver Figura 3.13). Esto se logra reemplazando las capas completamente conectadas por capas convolucionales de 1x1, que resultan equivalentes para la tarea de clasificación, pero permiten a la red procesar imágenes de distintos tamaños al generar imágenes de salida de tamaño proporcional al de la imagen de entrada. Para resolver el problema de la baja resolución de la salida, los autores proponen entrenar capas de *upsampling* que reciben los *feature maps* de distintas capas de red (ver Figura 3.14), y alimentan una predicción densa del mismo tamaño que la imagen de entrada. Los

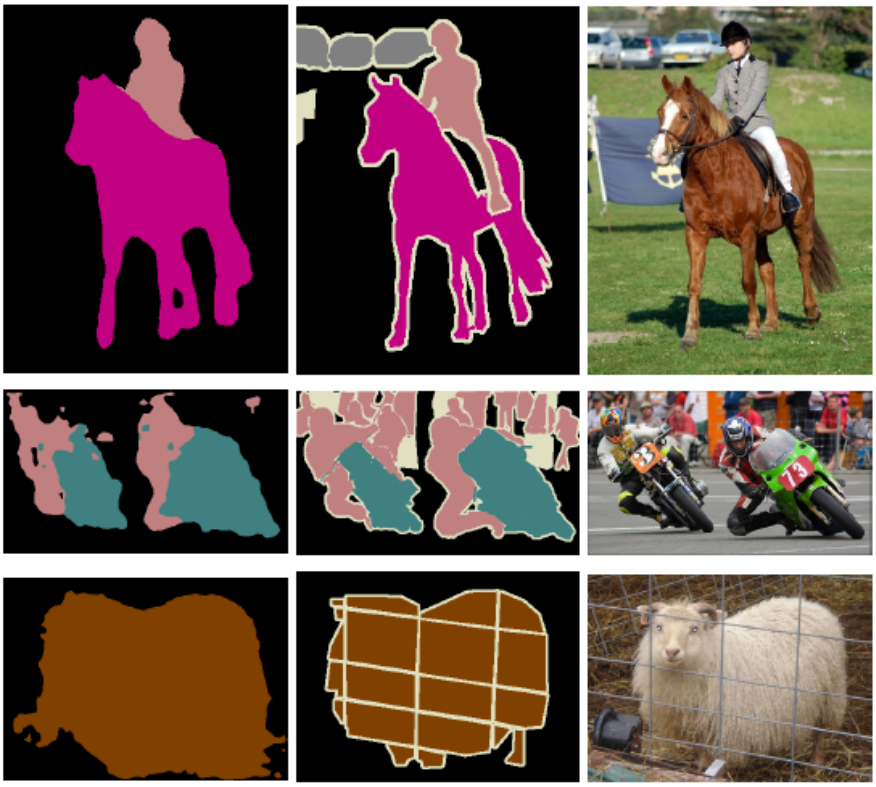


Figura 3.12: Ejemplo de segmentación semántica. A la izquierda el resultado del método de segmentación semántica FCN [60]. Al medio el etiquetado manual. Y a la derecha la imagen de entrada.

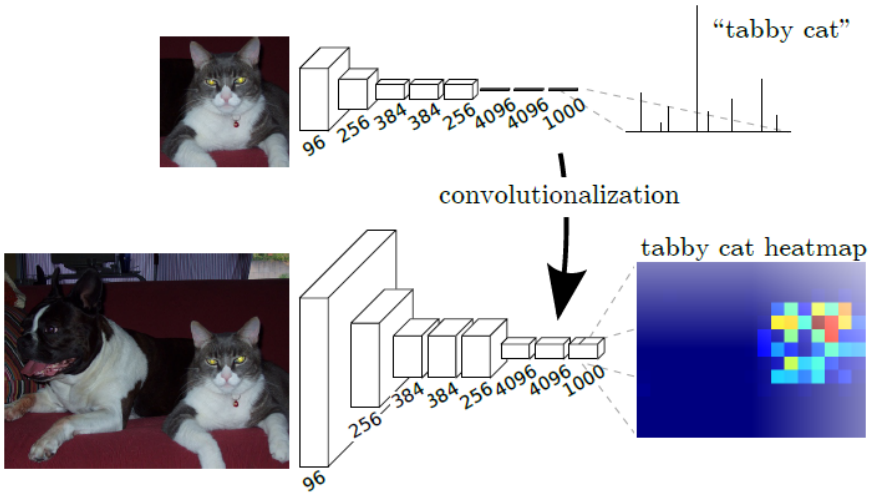


Figura 3.13: Reemplazo de capas completamente conectadas por capas convolucionales de 1x1 [60]. Arriba se muestra la red entrenada como un clasificador, que al ser modificada permite una segmentación semántica gruesa de la imagen, al recibir una imagen más grande.

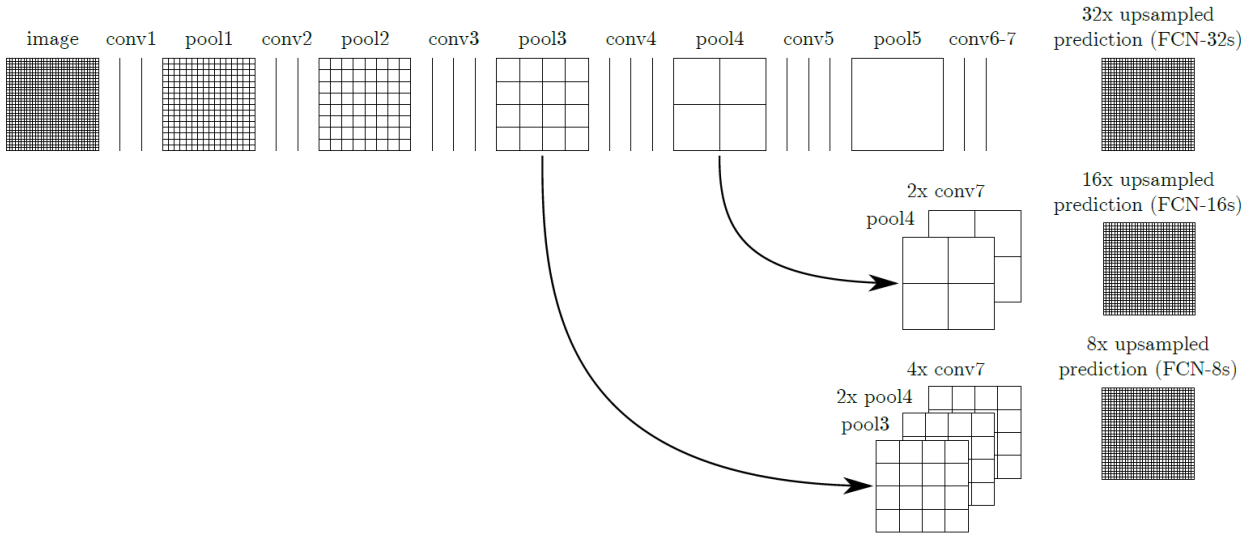


Figura 3.14: Estructura de un la red FCN [60]. Las flechas muestran la fuente de la que se obtienen características para la reconstrucción densa.

mejores resultados se obtienen utilizando VGG16 como arquitectura base.

Poco tiempo después se publica la red SegNet [4], que comparte diversas similitudes con FCN, pero su principal diferencia es la manera en la que hacen el *upsampling*: Aquí se utiliza la estructura VGG16 para obtener características dispuesta en forma de espejo, de manera de comprimir la información de la imagen, para luego descomprimirla en la predicción densa (ver Figura 3.15). Aquí se utiliza un ingenioso método de *upsampling*, en el que se imita el comportamiento del *pooling* equivalente, logrando reconstruir una predicción densa. Este trabajo logra una leve mejoría con respecto a FCN, pero ambos demuestran un gran avance en comparación con otros métodos de segmentación previos a las redes convolucionales, como Textonboost [85] y JGMT [26].

En la *International Conference on Learning Representations* del año 2016, Fisher Yu y Vladlen Koltun [104] presentaron un método de segmentación semántica basado en *Deep Learning* conocido como *DilatedNet*, que utiliza convoluciones dilatadas para aumentar el tamaño de los campos receptivos de manera exponencial mientras aumenta el número de parámetros de manera lineal, lo que permite mantener la resolución de las imágenes de entrada para generar una segmentación semántica densa de salida. En este trabajo se utiliza la estructura de red VGG16, reemplazando las últimas etapas de *pooling* por convoluciones dilatadas. Esta estructura supera los resultados obtenidos por FCN y SegNet. Adicionalmente, el autor presenta una etapa de contexto, que recibe los *feature maps* de salida de su estructura de red, agrega información de contexto a través de una secuencia de capas de convoluciones dilatadas, que aprenden las relaciones entre las ubicaciones de las distintas clases. En conjunto, la estructura VGG16 modificada junto con la etapa de contexto

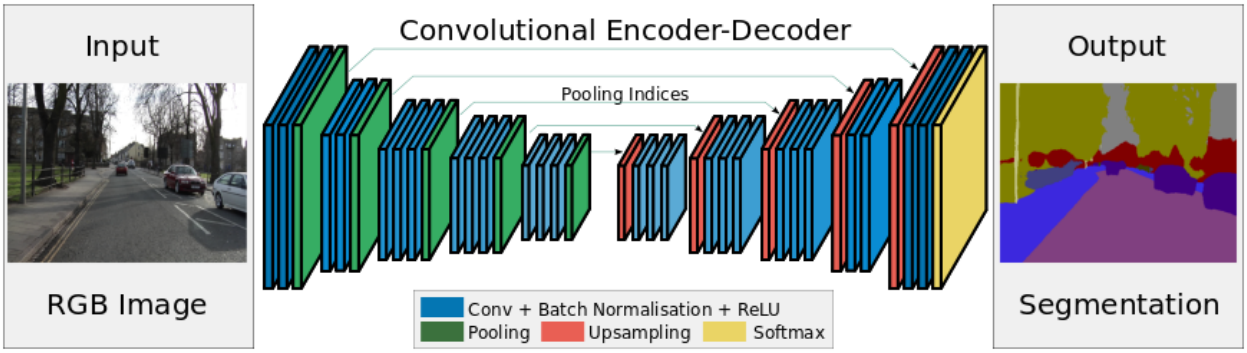


Figura 3.15: Estructura de un la red SegNet [4].

obtuvieron resultados significativamente mayores que los métodos presentados hasta ese entonces en las bases de datos CamVid, KITTI y CityScapes. Trabajos recientes se han inspirado en esta arquitectura para incorporar nuevas metodologías que han permitido mejorar el desempeño y el tiempo de procesamiento.

3.7. Localización para vehículos autónomos

Los mapas métricos para la localización de vehículos autónomos son costosos de utilizar y mantener ya que cualquier cambio en el ambiente puede afectar su contenido, desde el paso de vehículos y peatones hasta la construcción de un nuevo edificio. Es por esto que el desarrollo de métodos de localización para vehículos autónomos se ha enfocado principalmente en el uso de una representación topológica del entorno que permita manejar el mapa de áreas extensas en tiempo real.

Un ejemplo de un mapa topológico para localización a gran escala es el método de FAB-MAP [24], un método de SLAM visual basado en la apariencia de las imágenes. En este trabajo se utiliza el método de *bag-of-visual-words*, en el que se detecta un conjunto de características en la imagen, que luego son agrupadas con respecto a un vocabulario visual. Este vocabulario visual se entrena haciendo *clustering* sobre un conjunto de imágenes de entrenamiento, generando una división de Voronoi en el espacio de características, donde cada región corresponde a una palabra visual en particular. De esta manera el espacio de características, que se considera continuo, se transforma en un espacio discreto, lo que facilita el manejo y la comparación de la apariencia de imágenes. Luego, cada imagen es descrita a través de un vector binario que representa la presencia o ausencia de cada una de las palabras visuales en la imagen. Este vector binario es utilizado como observación en un modelo probabilístico que permite discernir si el lugar ya ha sido visitado anteriormente o no, además de estimar la pose dentro del mapa topológico en el primer caso.

Se debe destacar que el mapa topológico generado por el método de FAB-MAP no contiene

información métrica y cada nodo dentro del grafo contiene un vector binario que lo describe, lo que facilita la búsqueda de imágenes con apariencia similar para el cierre de bucles.

En el trabajo de Chris Linegar, Winston Churchill & Paul Newman [18] [59], se utiliza un sistema de odometría visual estéreo basada en el seguimiento de *landmarks* para definir *experiencias*. Una experiencia es una secuencia de poses del vehículo, que además contiene la pose de los *landmarks* observados. El mapa topológico se construye utilizando simultáneamente la pose estimada por la odometría visual y el método de localización: si la localización es coherente con las observaciones entonces se considera que el vehículo está localizado, pero cuando la pose estimada es distinta a las observaciones se crea una nueva experiencia conectada con la última pose en la que se consideraba localizado el vehículo. Mientras se están creando nuevas experiencias se busca en todo el mapa topológico por alguna experiencia que sea similar a las observaciones. Si se encuentra una experiencia similar, entonces la experiencia nueva se conecta con la experiencia encontrada y se considera el vehículo como localizado. El mapa topológico utiliza una estructura de grafo que contiene las experiencias generadas en los nodos y las interconexiones generadas entre ellas en las aristas. Cabe destacar que no se fuerza una coherencia métrica entre los nodos del mapa. De esta manera el mapa es robusto a errores de la odometría visual, además de permitir que varias experiencias representen un mismo lugar, pero con distintas condiciones de iluminación o meteorológicas.

En un trabajo reciente de Brubaker et al. [15] se presenta un método de localización basado en el mapa de calles de OpenStreetMap (OSM), un proyecto que tiene como objetivo crear mapas libres y editables de todo el mundo sin restricciones legales o técnicas para su uso de la misma manera en la que Wikipedia provee de una enciclopedia gratuita. La estructura del mapa de calles de OSM es un grafo dirigido, donde los nodos representan segmentos de calles y las aristas representan la conectividad entre los segmentos. Cada segmento de camino se representa como una línea recta o como un arco circular. De esta manera cualquier forma compleja que pueda tener un camino se puede representar por una secuencia de segmentos simples. En el trabajo de Brubaker se supone que el vehículo sólo puede estar en las calles que están en el mapa y que el vehículo respeta la dirección y orientación del camino. El método de localización recibe la odometría visual obtenida por unas cámaras estéreo montadas en el techo del vehículo y la utiliza para actualizar la estimación de la pose. Este método tiene la ventaja de no requerir un método de mapeo, ya que utiliza la información recopilada en OSM. Sin embargo, esto también representa una limitante, ya que este sistema es sensible a cualquier modificación en la ruta, además de estar imposibilitado de funcionar en ambientes en los que no se tiene un mapa construido con anterioridad.

Existen trabajos donde se ha propuesto el uso de mapas semánticos para la navegación autóno-

ma, como en el trabajo de Triebel et al. [94], donde se presenta el uso de active learning para la construcción offline de mapas semánticos para la navegación autónoma basado en la localización de semáforos. En este caso la única información semántica disponible es la ubicación de semáforos en un mapa métrico.

Por su parte, Gallart [34] construye un mapa topológico de interiores en el que se asignan etiquetas semánticas a cada habitación, identificando entre clases como: cocina, corredor, oficina o sala de reuniones. En este trabajo utilizan un método de SLAM métrico para construir el mapa, ademas de crear un mapa topológico con un nodo por cada metro que ha recorrido el robot, adicionalmente utilizan un método de detección de puertas para lograr segmentar las habitaciones en el mapa topológico, e información visual para clasificar cada habitación. Si bien este trabajo muestra excelentes resultados, no es posible aplicarlo en exteriores ni en casos de conducción autónoma, ya que no existe una clara separación entre lugares, como lo son las puertas para las habitaciones, y el problema de clasificar espacios es mucho mas complejo debido a la gran variedad de objetos presentes.

Finalmente, el trabajo de Grimmett et al. [36] presenta un método de construcción de mapas que permite mantener la información semántica junto con las actualizaciones métricas en el mapa, permitiendo así construir un mapa métrico que incorpora ubicación de estacionamientos y lugares donde se han observado peatones frecuentemente. Una desventaja de este método es que por cada vuelta que da al lugar se agregan mas puntos de interés, aumentando así el costo computacional, lo que lo hace inviable para que opere durante largos periodos de tiempo. Otra principal desventaja es que al estar construido sobre un mapa métrico, los requisitos computacionales para una operación en una zona extensa, como una ciudad, aumentan significativamente, ademas es susceptible a elementos móviles o cambios en el entorno.

Capítulo 4

Metodología de Mapeo y Localización en Mapas Topológicos Semánticos

En esta tesis se presenta una solución al problema de la navegación autónoma basado en mapas topológicos semánticos. Gracias a la estructura de grafo del mapa topológico y a la información semántica del sistema de percepción, este mapa logra dividir segmentos del camino de acuerdo a la interpretación de alto nivel de su apariencia, generando segmentos de manera automática a partir de las experiencias y permitiendo una localización eficiente y precisa. Esta metodología utiliza un método de percepción semántica como entrada, que transforma la información de los sensores en información de alto nivel. Para esta tesis se utilizan métodos de visión computacional, particularmente métodos de segmentación semántica, pero es posible utilizar información semántica obtenida a partir de otros sensores, como alguna clasificación semántica del ancho de un camino o una estimación de la rugosidad de un camino no pavimentado, que se podrían obtener a partir de sensores de rango o mediciones inerciales.

La metodología propuesta facilita la navegación autónoma, y tiene aplicación directa en navegación en autopistas, caminos rurales y rutas habituales en ciudad, donde el uso y la mantención de mapas métricos es difícil y costosa computacionalmente.

El objetivo es generar una descripción topológica del entorno a partir de información semántica, que permita llevar a cabo tareas de localización. Para esto se definen tres etapas como muestra el diagrama de bloques de la Figura 4.1: Percepción Semántica, Construcción de Mapas y Localización.

A continuación, se detalla la metodología para la construcción de mapas topológicos semánticos, detallando los distintos módulos que la componen. Esta metodología es luego evaluada en el capítulo 5. Finalmente, en el capítulo 6 se presenta una adaptación de esta metodología para su uso en caminos no pavimentados.

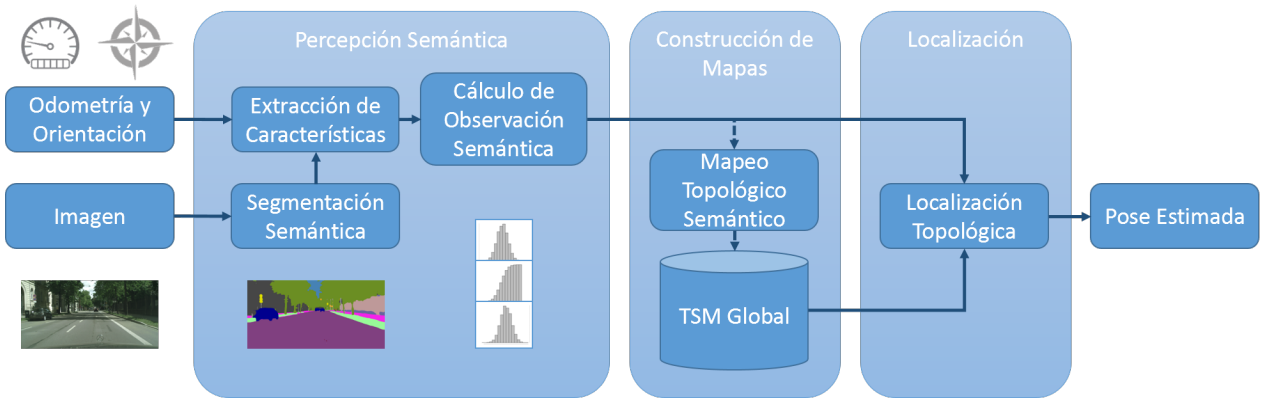


Figura 4.1: Diagrama de bloques.

4.1. Percepción Semántica

La etapa de *Percepción Semántica* es la encargada de transformar la información cruda de los sensores en abstracciones semánticas que permitan describir el entorno del camino. La entrada de esta etapa son los datos sensoriales que en este trabajo de tesis son principalmente las imágenes obtenidas desde una cámara frontal, pero es posible utilizar otros sensores que permitan obtener información semántica relevante para la descripción del entorno. La salida de esta etapa se denomina Descripción Semántica que, como su nombre lo indica, cumple con la tarea de resumir la información semántica obtenida en un vector de caracteristicas. Esta descripción semántica será luego utilizada tanto para la etapa de *Construcción de Mapas*, como por la etapa de *Localización*.

El objetivo de esta etapa es crear una descripción basada en la segmentación semántica de la imagen de entrada. Esta descripción incluye información de los elementos estáticos observados en el entorno (como edificios, postes, señalética, semáforos, entre otros) y utiliza información de contexto espacial de la distribución de estos elementos para agruparlos, creando una descripción semántica del entorno. Adicionalmente, cada descripción semántica incluye dos valores escalares que representan la odometría y la orientación del vehículo, lo que permite incorporar información de distancia y orientación a los segmentos del mapa topológico. Esta etapa incluye tres pasos: Segmentación Semántica, Extracción de Características y Cálculo de Observación Semántica.

Segmentación Semántica

El método de segmentación semántica propuesto por Yu & Koltun [104] (ver sección 3.6) es aplicado a cada imagen para obtener una etiqueta semántica por cada píxel. Este método utiliza convoluciones dilatadas, un operador de convolución modificado que aplica un filtro en una región extendida de la imagen, lo que permite a su red neuronal tenga campos receptivos exponencialmente

crecientes sin perder resolución. Esta arquitectura de red convolucional está dividida en dos módulos: *Front-End* y *Context*. El módulo *Front-End* esa una red VGG-16 [86] adaptada para una predicción densa a través de la remoción de las las dos últimas capas de *pooling* y *striding*, y el reemplazo de las capas de convolución subsiguientes por convoluciones dilatadas. Los mapas de características resultantes del módulo *Front-End* puede ser utilizado directamente como una predicción semántica densa de la imagen utilizando una capa *softmax*. El módulo *Context* mejora el rendimiento de la salida del módulo *Front-End* al agregar información de contexto espacial: recibe los C mapas de características de salida del módulo *Front-End* y produce C mapas de características de salida, donde C es el número de clases semánticas consideradas. Este módulo tiene un número fijo de capas de convolucionales con diferentes factores de dilatación, permitiendo al módulo utilizar información de contexto espacial de regiones amplias con pocas capas. Una capa final aplica una convolución de $1 \times 1 \times C$, que genera la salida de la red.

La red utilizada en este trabajo fue entrenada utilizando la base de datos Cityscapes Dataset [21] (ver imagen de ejemplo en la Figura 4.2), que define 35 etiquetas distintas. Para simplificar la decisión sobre que etiquetas utilizar para la construcción de mapas y localización, las etiquetas son clasificadas en las siguientes categorías:

Pisos: Todas las estructuras horizontales a nivel de suelo, incluyendo vegetación horizontal y suelo transitable por vehículos y peatones. Las etiquetas incluidas son: *tierra, pasto, camino, vereda, estacionamiento y vías de tren.*

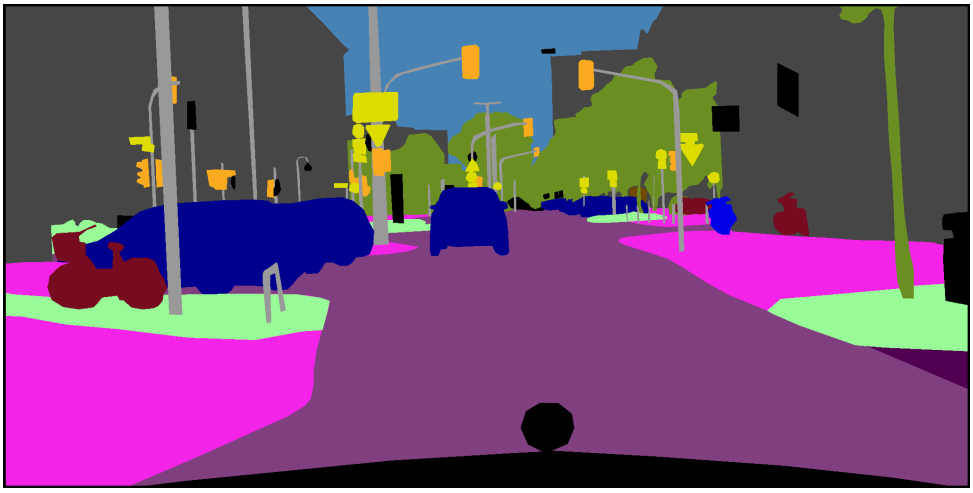
Fondos: Cualquier estructura vertical en torno al camino, además del cielo. Las etiquetas relacionadas con esta categoría son: *cielo, edificio, muralla, reja, barrera de contención, puente y túnel.*

Otros: Cualquier otro elemento estático de interés para el tráfico de vehículos y peatones. Las etiquetas relacionadas con esta categoría son: *poste, grupo de postes, semáforo, señalética y vegetación.*

Nulos: Cualquier etiqueta relacionada con elementos dinámicos, como otros vehículos y peatones, o elementos desconocidos o indefinidos. Las etiquetas incluidas son: *sin etiquetar, el capó del vehículo, borde de la imagen, fuera de la región de interés, estático, dinámico, peatón, ciclista, automóvil, camión, bus, furgón, carro de arrastre, tren, motocicleta, bicicleta y placa patente.*

(a) Imagen Original

(b) Segmentación Semántica



(c) Etiquetado Manual

Figura 4.2: Ejemplo de segmentación semántica: (a) muestra una imagen de la base de datos Cityscapes dataset [21], (b) muestra la segmentación resultante del método [104], y (c) es el *ground truth* de la base de datos.

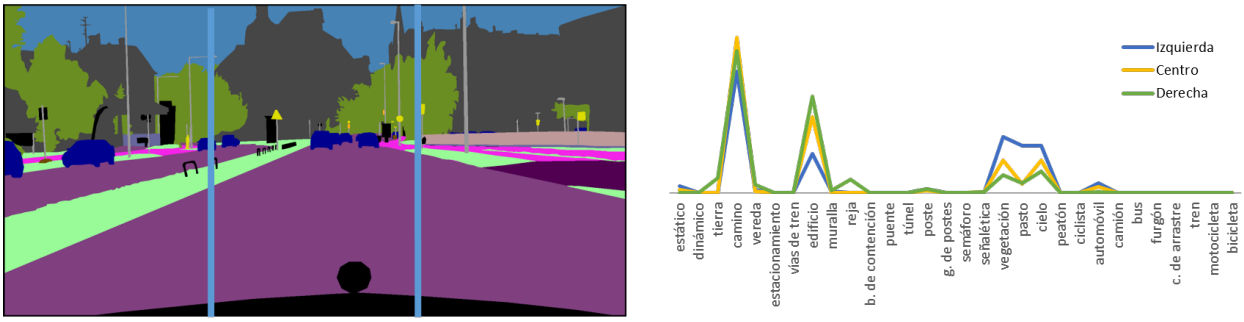


Figura 4.3: Ejemplo de las características obtenidas de una segmentación semántica. En la izquierda la imagen segmentada, y a la derecha los histogramas de izquierda, centro y derecha de la imagen.

Extracción de Características

La información de cada imagen segmentada es resumida en un vector de características que describe semánticamente el entono observado. El vector de características semánticas

$$F = \begin{bmatrix} d \\ \theta \\ h_l \\ h_c \\ h_r \end{bmatrix} \tag{4.1}$$

es un vector columna compuesto por cinco elementos: un valor escalar d que representa la odometría acumulada del vehículo relativa a la última imagen, una medida angular θ que representa la orientación del vehículo, y tres histogramas normalizados, h_l, h_c y h_r, de las regiones izquierda, central y derecha de la imagen semántica. Cada *bin* en los histogramas representa una de las 35 clases definidas por el método de segmentación semántica. Al construir el vector de características de esta manera, es posible asociar la información semántica con distintas regiones del camino, relativo a la orientación del vehículo.

La Figura 4.3 muestra una imagen segmentada junto a sus histogramas de las regiones izquierda, central y derecha. En este gráfico es posible reconocer diferencias semánticas importantes entre las regiones, por ejemplo la región izquierda contiene más cielo, tierra y árboles, y menos edificios que las otras dos regiones (ver etiquetas *cielo, pasto, vegetación* y *edificio*).

En los módulos de construcción de mapas y localización será necesario comparar y combinar la información de los vectores de características. Dado que cada elemento del vector de características tiene una significado diferente, es necesario definir operadores especiales: operador aditivo ($\oplus$), y distancia $d(\cdot, \cdot)$

El operador aditivo ($\oplus$) representa la suma de vectores de características y entrega como resultado un nuevo vector de características que representa la región cubierta por ambos vectores de características. La odometría del vector de características resultante es la suma de las odometrias de los vectores de características, y tanto la orientación como los histogramas son calculados como el promedio ponderado por sus respectivas odometrias:

$$F_a \oplus F_b = \begin{bmatrix} d^a \\ \theta^a \\ h_l^a \\ h_c^a \\ h_r^a \end{bmatrix} \oplus \begin{bmatrix} d^b \\ \theta^b \\ h_l^b \\ h_c^b \\ h_r^b \end{bmatrix} = \begin{bmatrix} d^a + d^b \\ \arctan\left(\frac{d^a \cdot \sin\theta^a + d^b \cdot \sin\theta^b}{d^a \cdot \cos\theta^a + d^b \cdot \cos\theta^b}\right) \\ \frac{d^a \cdot h_l^a + d^b \cdot h_l^b}{d^a + d^b} \\ \frac{d^a \cdot h_c^a + d^b \cdot h_c^b}{d^a + d^b} \\ \frac{d^a \cdot h_r^a + d^b \cdot h_r^b}{d^a + d^b} \end{bmatrix}. \tag{4.2}$$

El operador de distancia mide la diferencia entre la información semántica recopilada por los vectores de características como la diferencia entre los histogramas de las clases semánticas. El resultado es un escalar calculado como la máxima distancia coseno entre los histogramas correspondientes de ambos vectores de características:

$$d(F_a, F_b) = \max_{k \in \{l,c,r\}} \left(1 - \frac{h_k^a \cdot h_k^b}{\|h_k^a\| \cdot \|h_k^b\|}\right), \tag{4.3}$$

donde $h_k^a \cdot h_k^b$ es el producto punto entre histogramas.

Cálculo de Observación Semántica

Una observación semántica F_{SO} es un vector de característica construido al sumar los vectores de características de imágenes de entrada. Este vector acumula la información de vectores de características consecutivos similares, con un máximo de N_f vectores. Si la observación semántica actual acumula N_f vectores de características, o su diferencia con el último vector de características de entrada F_t es mayor que un umbral t_s, entonces se crea una nueva observación semántica con la información del último vector de características, como se muestra en el Algoritmo 4.1.

4.2. Construcción de Mapas

Un Mapa Topológico Semántico (TSM por sus siglas en inglés) es una estructura de grafo que describe el camino y el entorno utilizando la información de las observaciones semánticas. El TSM está organizado como una estructura de grafo dirigido $G = (N, E)$ donde cada arista $e_i \in E$ representa una región del camino, y cada nodo $n_i \in N$ representa la conexión entre dos o más aristas. Cada arista en el TSM corresponde a un segmento del camino donde las observaciones semánticas

Algoritmo 4.1 Cálculo de descripción semántica.

```
1: function SemanticObservationCalculation(F_SO^{t-1}, F_t)
2:     if d(F_t, F_SO^{t-1}) < t_s and n < N_f then
3:         F_SO^t = F_SO^{t-1} ⊕ F_t
4:         n = n + 1
5:     else
6:         F_SO^t = newSemanticObservation(F_t)
7:         n = 1
8:     end if
9: end function
```

obtenidas de imágenes consecutivas comparten una orientación e histogramas similares. Cada arista e_i es descrita por un vector de características con la información de las imágenes obtenidas dicha región, su orientación promedio y su largo estimado:

$$e_i = \begin{bmatrix} L \\ \theta \\ h_l \\ h_c \\ h_r \\ n_s \\ n_e \end{bmatrix}, \tag{4.4}$$

donde i es el identificador de la arista, L es el largo estimado de la arista, θ es la orientación promedio de la arista, $h_{l,c,r}$ son los histogramas de la segmentación semántica, n_s y n_e son el identificador del nodo inicial y final de la arista. Cada nodo n_i contiene los identificadores de los vértices que parten y de los que terminan en él:

$$n_i = \begin{bmatrix} S_s \\ S_e \end{bmatrix}, \tag{4.5}$$

donde i es el identificador del nodo, S_s es el conjunto de todas las aristas que empiezan en el nodo y S_e es el conjunto de todas las aristas que terminan en el nodo.

El TSM se construye de manera incremental agregando observaciones semánticas a la estructura de grafo. El algoritmo de mapeo es el siguiente: cada nueva observación semántica F_{SO}^t se compara con la última arista agregada en el TSM ($M_{t-1}.last$) utilizando el operador de distancia y la diferencia entre las orientaciones de la observación semántica ($F_{SO}^t.\theta$) y de la última arista ($M_{t-1}.last.\theta$). Si la distancia resultante es menor que un umbral de mapeo t_m y la diferencia de orientación es menor que un umbral de orientación t_o, entonces la observación semántica es incorporada a la última arista utilizando el operador de suma $\oplus$. En caso contrario, se considera terminado el vector de

caractarísticas de la última arista al agregar un nuevo nodo donde ésta termina, que cumple con ser el nodo donde parte una nueva arista con la información de la observación semántica:

$$M_t.last = \begin{cases} M_{t-1}.last \oplus F^t_{SO} & \textbf{si } d\left(F^t_{SO}, M_{t-1}.last\right) < t_m \\ & \textbf{y } |F_{SO}.\theta - M_{t-1}.last.\theta| < t_o \ , \\ M_t.NewEdge(F^t_{SO}) & \textbf{en caso contrario} \end{cases} \tag{4.6}$$

donde M_t es el TSM en tiempo t, y resume la información de las últimas t observaciones semánticas F^i_{SO}, with $i = 1, \ldots, t$. $M_t.last.F$ es el vector de características correspondiente a la última arista del TSM $M_t.last$. El Algoritmo 4.2 muestra el pseudo-código del algoritmo de mapeo.

Algoritmo 4.2 Algoritmo de mapeo topológico semántico.

```
1: function MAPPING(F^t_SO, M_{t-1})
2:     if d(M_t.last, F^t_SO) < t_m and |F_SO.θ − M_{t-1}.last.θ| < t_o then
3:         M_t.last = M_t.last ⊕ F^t_SO
4:     else
5:         N = M_t.NewNode()
6:         M_t.last.SetEndingNode(N)
7:         M_t.NewEdge(F^t_SO)
8:         M_t.last.SetStartingNode(N)
9:     end if
10:    return M_t
11: end function
```

Dado que las características obtenidas de la segmentación semántica representan una descripción de alto nivel del camino y su entorno, no es posible reconocer con ellas cuando el vehículo pasa por un lugar conocido, ya que dos lugares separados pueden compartir un vector de características similar. Por lo tanto, el trabajo de conectar nodos equivalentes en el mapa debe ser hecho de manera manual. Sin embargo, es posible automatizar este trabajo utilizando métodos adicionales de SLAM visual o de reconocimientos de lugares basados en imágenes.

4.3. Localización

Lista de Símbolos		
t	:	índice de tiempo
i, j	:	índices de aristas en el GTSM
e_i	:	i-ésima arista en el GTSM
k	:	índice de partículas
x_t	:	pose topológica del vehículo en tiempo t como una arista del GTSM
$\hat{x}_t$	:	pose topológica estimada del vehículo en tiempo t
z_t	:	observación semántica en tiempo t
x_k^t	:	pose topológica de la k-ésima partícula en tiempo t
w_k^t	:	peso de importancia de la k-ésima partícula en tiempo t
l_k^t	:	posición unidimencional de la k-ésima partícula en tiempo t
N_p	:	Numero de partículas
S_s^i	:	Conjunto de las aristas que parten en el nodo en el que termina la i-ésima arista
S_e^i	:	Conjunto de las aristas que terminan en el nodo en el que parte la i-ésima arista

Cuando se utiliza la estructura de TSM, el problema de localización se reduce a la búsqueda de la arista que mejor corresponda a las observaciones obtenidas por el vehículo, por lo tanto, una pose topológica estimada $\hat{x}_t$ es definida como una de las aristas del mapa global (*Global TSM* o GTSM), que es una TSM previamente construido, que representa el mundo e incluye información del largo, orientación y características semánticas en todas las aristas. Este mapa es construido de acuerdo al algoritmo descrito en Algoritmo 4.2, y resume la información de recorridos previos por un lugar.

Dado que el resultado de esta metodología de localización es un elemento dentro del mapa topológico y no una pose métrica, ésta puede funcionar de manera independiente de cualquier sistema de posicionamiento global o métrico, además de que no requiere de un mapa métrico previo para su funcionamiento. Además, tanto el sistema de construcción de mapas como el sistema de localización son alimentados exclusivamente con información semántica de alto nivel de las observaciones, lo que implica que la metodología es invariante a cambios en el entorno: el color de las hojas de un árbol, la presencia de vehículos, o la sombra en una muralla no afectan la descripción semántica y por tanto tampoco al método de localización.

El problema de localización es modelado como un modelo oculto de Markov (HMM por las siglas

en inglés), donde el conjunto de estados Q corresponde a las aristas del mapa global, $\pi = \{\pi_i\}$ es la distribución inicial de probabilidades, donde π_i es la probabilidad inicial del vehículo de estar en la arista i, y representa la estimación inicial de la pose del vehículo en el mapa global. $A = \{A_{i,j}\}$ es la matriz de transición de estados, donde $A_{i,j}$ es la probabilidad de pasar del estado i al estado j, y queda definida por los nodos del mapa global. El conjunto de las verosimilitudes para la observación semántica z_t en las aristas i del mapa global se define como $B = \{b_i(z_t)\}$. Finalmente, definiremos la estimación del estado en tiempo t como $\hat{x}_t$, una arista del mapa global.

El módulo de localización utiliza cada observación semántica $z_t = F_{SO}^t$ como observación para el método de localización, y su verosimilitud $p(z_t \mid e_i)$ es estimada a través de la distribución de máxima entropía [23] con soporte en el rango $[0, \infty]$ para la información semántica, y la distribución de Von Mises [11] para la orientación, como:

$$p(z_t \mid e_i) \propto e^{-d(z_t, F_i)} \cdot e^{\kappa \cdot cos(\theta_{z_t} - \theta_i)}, \tag{4.7}$$

donde F_i es el vector de características relacionado con la arista e_i, θ_{z_t} es la orientación de la observación semántica, θ_i es la orientación de la arista e_i y κ es el parámetro de concentración de la distribucion de Von Mises.

Dos métodos de localización son propuestos y comparados: uno basado en *Forward Algorithm*, y otro basado en *Filtro de Partículas*. A continuación se detalla cada una de estas alternativas.

4.3.1. Forward Algorithm

En este método se resuelve el problema de localización utilizando *Forward Algorithm* [79]. Se define $\alpha_t(e_i) = p(e_i, z_{1:t})$ como la probabilidad conjunta del estado $x_i = e_i$ y las observaciones $z_{1:t}$. El *Forward Algorithm* calcula esta probabilidad recursivamente basado en la regla de independencia condicional del modelo oculto de Markov a través de

$$\widehat{\alpha}_t(e_i) = p(z_t \mid e_i) \cdot \sum_{j \in GTSM} A_{i,j}^t \cdot \alpha_{t-1}(e_j), \tag{4.8}$$

donde $p(z_t|e_i)$ es la verosimilitud de una observación z_t en una arista e_i que se calcula según la ecuación (4.7), y $A_{i,j}^t = p(e_i|e_j)$ es la probabilidad de transición desde la arista i a la arista j, que se calcula como

$$A_{i,j}^t = \begin{cases} \frac{L_i - d_t}{L_i} & \text{si } i = j \\ \frac{1}{|S_s^j|} \cdot \frac{d_t}{L_j} & \text{si } i \in S_s^j \\ 0 & \text{en caso contrario} \end{cases}, \tag{4.9}$$

donde L_i y L_j son el largo de las aristas e_i y e_j, d^t es la odometría acumulada de la última observación, S_s^j es el conjunto de todos los vértices que parten desde el nodo donde termina el vértice e_j y $\left|S_s^j\right|$ es la cardinalidad de S_s^j.

Luego la probabilidad conjunta de los estados posibles, es decir los nodos en el mapa global, se calculan normalizando con la función *softmax*:

$$\alpha_t(e_i) = \frac{e^{\hat{\alpha_t}(x_i)}}{\sum_j e^{\hat{\alpha_t}(e_j)}}. \tag{4.10}$$

Finalmente, el problema de localización se resuelve encontrando el nodo $\hat{x}_t$ con máxima probabilidad conjunta:

$$\hat{x}_t = \underset{i \in GTSM}{\operatorname{argmax}}\, \alpha_t(e_i). \tag{4.11}$$

La complejidad de la localización utilizando Forward Algorithm es $O(N_e)$, donde N_e es el número de aristas en el GTSM, ya que el algoritmo debe comparar cada observación con cada arista.

4.3.2. Filtro de Partículas

La idea principal de este método es aproximar la probabilidad a posteriori de la pose topológica del vehículo x_t en el mapa global en un tiempo t a través de un conjunto N_p de muestras con pesos en la estructura de grafo del GTSM. Este trabajo está basado en el trabajo de Merriaux et al. [65], donde la k-ésima partícula tiene una pose $x_k^t = e_i$ definida como la i-ésima arista del GTSM, un peso w_k^t y una posición unidimensional l_k^t dentro de la arista. Dado que el vehículo se mueve en la estructura de grafo, la función de transición de posición se define como:

$$l_k^t = l_k^{t-1} + \nu(d^t), \tag{4.12}$$

donde $\nu(d^t) \sim \mathcal{N}\left(d^t, \left(d^t\right)^2\right)$ es la odometría acumulada de la observación (d^t) con ruido Gaussiano agregado.

Además, l_k^t puede ser mayor que L_k, el largo de la arista correspondiente a la pose x_k^t, o menor que cero. Esto significa que la partícula está cambiando de vértice, y por lo tanto se requiere una función de transición de vértices. Sea $R_f^i \sim \mathcal{U}(S_s^i)$ una muestra aleatoria de S_s^i, el conjunto de todos los vértices que parten en el nodo en el que termina el vértice e_i, y $R_b^i \sim \mathcal{U}(S_e^i)$ una muestra aleatoria de S_e^i, el conjunto de todos los vértices que terminan en el nodo en el que empieza el vértice e_i. Entonces, la función de transición de vértices se define como:

$$x_k^t = \begin{cases} R_b^i & \text{si } l_k^t < 0 \\ x_k^{t-1} & \text{si } l_k^t \in [0, L_k] \\ R_f^i & \text{si } l_k^t > L_k \end{cases}. \tag{4.13}$$

Además, si hay una transición de vértice, la posición de la partícula l_k^t debe ser corregida sumándole el largo del vértice actual o restándole el largo del vértice antiguo. Luego de aplicar las funciones de transición de posición (4.12) y de vértice (4.13) para cada partícula, el peso normalizado de cada partícula es calculado como:

$$w_k^t = K_t \cdot p\left(z_t \mid x_k^t\right) \cdot w_k^{t-1}, \tag{4.14}$$

donde K_t es un factor de normalización y $p\left(z_t \mid x_k^t\right)$ es la verosimilitud de la observación, obtenido de la ecuación (4.7).

Una vez que se obtienen los pesos para las partículas, se efectúa un proceso de *resampling* para disminuir la varianza en los pesos y evitar la degeneración de las partículas. El conjunto de partículas remuestreadas es generado tomando N_p muestras al azar entre las partículas anteriores, de acuerdo a sus pesos (Algoritmo SIR [49]). El *resampling* se lleva a cabo si el tamaño efectivo de la muestra, definido como $\hat{N}_{eff} = 1/\sum_k (w_k)^2$, es menor a la mitad del número de partículas $N_p/2$ [2]. Finalmente, el problema de localización se resuelva al estimar la pose del vehículo en el mapa global como la arista con la mayor suma de pesos entre partículas en esa arista:

$$\hat{x}_t = \underset{e_i \in \text{ GTSM}}{\text{argmax}} \sum_{k=1}^{N_p} w_k \cdot \delta_{e_i}\left(x_k^t\right), \tag{4.15}$$

donde $\delta_{e_i}\left(x_k^t\right)$ es el delta de Kronecker que entrega 1 si $x_k^t = e_i$ y 0 en caso contrario.

El tiempo de ejecución del Filtro de Partículas esta relacionado con el numero de partículas y el numero de aristas con al menos una partícula en ella. Por lo tanto, la complejidad del filtro de partículas es $O(|N_p| + |N_e^p|) \simeq O(|N_p|)$ cuando $|N_p| \gg |N_e^p|$.

Capítulo 5

Evaluación de Metodología en Caminos Urbanos

A continuación, se presenta la metodología utilizada para evaluar el desempeño de los métodos de mapeo y localización en ambientes urbanos. Primero se presenta la base de datos utilizada para llevar a cabo esta evaluación. Luego se presentan los experimentos realizados para el método de mapeo y finalmente los experimentos para evaluar los distintos métodos de localización.

5.1. Base de Datos en Caminos Urbanos

Para probar y validar los métodos de mapeo y localización en ambientes urbanos, fue necesario el diseño y la construcción de una base de datos de imágenes tomadas desde un vehículo, que recorriera diversas calles en un ambiente urbano, y que presentara suficiente dificultad para los métodos de localización. Se estableció que, para aumentar la dificultad de la prueba, sería necesario favorecer la inclusión de la mayor cantidad de intersecciones posibles, por sobre recorridos rectos en los que no habría muchas intersecciones.

La base de datos fue grabada en el centro de Santiago de Chile, utilizando una cámara montada al interior del parabrisas de un vehículo particular. La base de datos está grabada con una resolución de 1280x720 y contiene dos secuencias de dos recorridos distintos, la primera grabada el día 6 de diciembre del año 2016, y la segunda fue grabada al día siguiente. Los trayectos recorridos se muestran en la Figura 5.1.

La primera secuencia (en verde) tiene un largo total de $11,29$[km], contiene 31.144 imágenes. Su trayectoria recorre 14 calles y 33 intersecciones dentro del cuadrante delimitado al norte por la calle Grajales, al este por la avenida Ejercito Libertador, al sur por la avenida Blanco Encalada y al oeste por la calle Unión Latinoamericana. Esta secuencia fue diseñada para ser utilizada para la construcción del mapa.

La segunda secuencia (en rojo) tiene un largo de $8,35$[km] y contiene 20.718 imágenes. Su

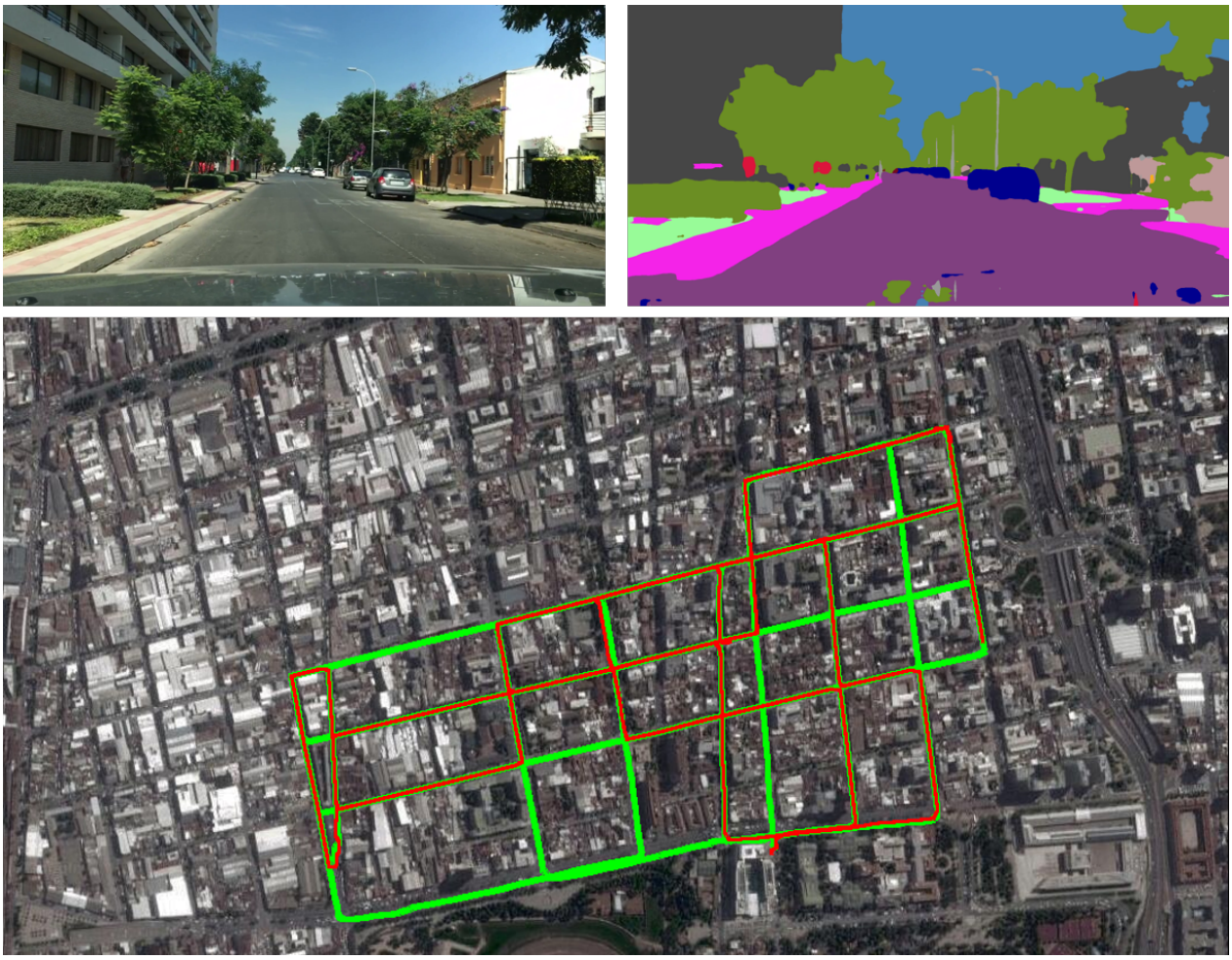


Figura 5.1: Arriba se muestra una imagen de la base de datos grabada con la imagen segmentada resultante obtenida con el método de segmentación semántica [104]. Abajo, un imagen satelital ilustrando las rutas recorridas en la base de datos.

trayectoria está contenida completamente en las calles recorridas por la primera secuencia, e incluye bucles y recorridos por las mismas intersecciones.

5.2. Resultados de Mapeo Semántico de Caminos Urbanos

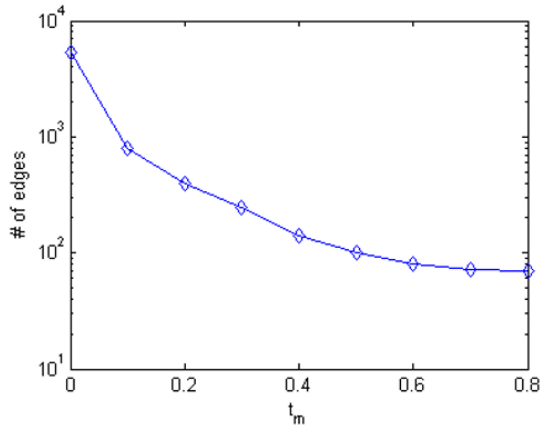
Para evaluar la metodología descrita en el Algoritmo 4.2, se utiliza sobre la base de datos de caminos urbanos presentada en la Sección 5.1, particularmente en la primera secuencia, ya que es la diseñada para ser utilizada como base para la construcción del mapa.

Para efectos de visualización y evaluación, cada imagen de la base de datos tiene asociada una posición GPS que servirá como *ground truth*, y cada arista en el mapa topológico incluye las posiciones GPS de cada imagen asociada con dicho vértice.

Los parámetros utilizados para este experimento fueron: Número máximo de imágenes por observación $N_f = 5$, umbral de orientación $t_o = \pi/6$ y umbral de observación $t_s = 0,3$. El umbral de mapeo se mueve desde 0,0 hasta 0,8 para observar los efectos de este parámetro sobre el mapa resultante. Cabe destacar que el mapa construido tiene un largo de $9,57$[km], ya que los segmentos de caminos que son recorridos más de una vez son descartados para la construcción del mapa. Estos parámetros fueron configurados utilizando una base de datos adicional, descrita en el Anexo B .

La Figura 5.2(b) muestra que el valor del umbral de mapeo está relacionado con el número de aristas presentes en el mapa. Con un valor de $t_m = 0$ el método de mapeo transforma cada observación en una nueva arista, ya que considera que cada observación es distinta a la siguiente. Un mapa como este podría permitir una localización sumamente precisa, pero ésta sería altamente sensible a perturbaciones en la observación. Además, un mapa tan fino sería muy costoso para cubrir lugares amplios. Por otro lado, el mapa construido con un umbral $t_m = 0,8$ (ver Figura 5.2(f)) presenta nodos únicamente en los lugares donde hay cambios de orientación importante y en las intersecciones. Un mapa como este resulta más conveniente para cubrir espacios amplios, ya que es capaz de representar grandes distancias con un vector de características, lo que permitiría una localización más eficiente desde el punto de vista computacional. Sin embargo, la ausencia de nodos dentro de una cuadra, donde se observan cambios importantes desde un punto de vista semántico, muestra que la información semántica no está siendo utilizada para la construcción del mapa, por lo tanto la información semántica contenida en cada arista puede no ser representativa del entorno.

Estos resultados muestran que es necesario encontrar un parámetro del umbral de mapeo que permita representar regiones homogéneas sistemáticamente en aristas en el mapa topológico semántico. El valor óptimo para este umbral, y por lo tanto cuán grueso o fino resulta el mapa, dependen de la aplicación y los métodos que se deseen utilizar. Por lo tanto, la definición sobre el valor óptimo



(a) Aristas en GTSM

(b) GTSM con $t_m = 0,0$

(c) GTSM con $t_m = 0,2$

(d) GTSM con $t_m = 0,4$

(e) GTSM con $t_m = 0,6$

(f) GTSM con $t_m = 0,8$

Figura 5.2: Efectos del umbral t_m en el mapa resultante. (a) muestra el número de vértices en el GTSM en escala logarítmica para distintos valores del umbral t_m. (b-f) muestran una imagen satelital con el GTSM resultante con distintos valores de t_m, donde los círculos rojos representan nodos, y las líneas azules representan aristas.

de este parámetro queda sujetos a la evaluación de los métodos de localización que utilizarán el método en la sección 5.3.

5.3. Resultados de Localización

Se valida el uso de la metodología de mapeo y localización basada en TSM para su uso en vehículos autónomos. Estos métodos son validados utilizando la base de datos presentada en la sección 5.1. En particular, se utiliza la primera secuencia para construir el mapa, y la segunda secuencia para probar la localización.

La red neuronal *Dilation10* propuesta por Yu y Koltun [104] es utilizada para la segmentación semántica. Esta red fue entrenada utilizando el conjunto de entrenamiento con etiquetado fino de la base de datos Cityscapes dataset [21], una extensa base de datos para segmentación semántica a nivel de píxel y de instancias. La red entrenada se encuentra disponible en la página web del autor. Las imágenes de las bases de datos a utilizar fueron redimensionadas para cumplir con el tamaño de entrada de la red, que corresponde a 2048x1024 píxeles.

Para esta metodología se define como estimación de pose correcta a una estimación de pose topológica del método de localización si la distancia entre la posición GPS de la imagen y la pose mas cercana almacenada a la arista es menor a 20[m], como en [14, 65].

Para hacer una evaluación cuantitativa del desempeño de la metodología de localización, es necesario definir métricas que describan el comportamiento de la localización en el mapa topológico. Cabe destacar que métricas conocidas como las propuestas por Kummerle et al. [56] y Sturm et al. [87] no pueden ser utilizadas debido a que la localización es estrictamente topológica. En [92] se define un *Falso Estimado* como una estimación incorrecta de la pose por el método de localización, basado en esto se define el *Verdadero Estimado* (TE por sus siglas en inglés) como 1 cuando la estimación de la localización es correcta y 0 en caso contrario. Por lo tanto, el TE para la pose estimada $\hat{x}_t$ en tiempo t es:

$$\mathrm{TE}_t = \begin{cases} 1 & \text{si } \hat{x}_t \text{ es correcto} \\ 0 & \text{en caso contrario} \end{cases}. \tag{5.1}$$

Entonces definiremos la *Razón de Verdaderos Estimados* (TER por las siglas en inglés) como la razón entre la cantidad de *Verdaderos Estimados* sobre el total de estimaciones de poses N_t, es decir el total de observaciones semánticas:

$$\mathrm{TER} = \frac{1}{N_t} \sum_t \mathrm{TE}_t. \tag{5.2}$$

Y adicionalmente definiremos la *Razón de Verdaderos Estimados según Distancia* (D-TER) como

la razón entre la distancia recorrida con una estimación correcta del método de localización, y la distancia total recorrida. Esta métrica basada en las distancias recorridas representa una evaluación justa de los métodos de localización, ya que ambos metodos de localización dependen explícitamente de la distancia recorrida para sus estimaciones y en caso de que el vehículo esté detenido, ninguno cambia su estimación. El D-TER se calcula como:

$$\text{D-TER} = \frac{\sum_t d_t \cdot \text{TE}_t}{\sum_t d_t}, \tag{5.3}$$

donde d_t es la odometría acumulada en la observación semántica obtenida en el tiempo t.

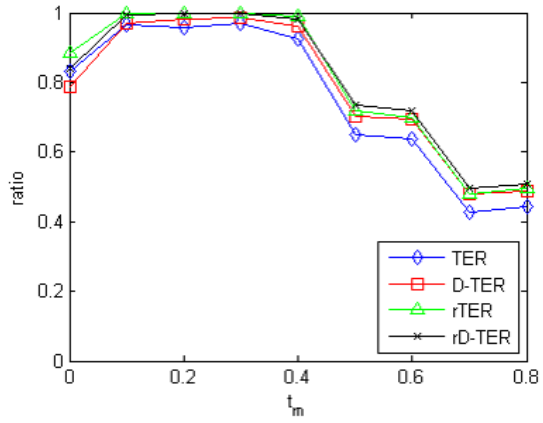
Finalmente, se definen versiones relajadas de ambas métricas: TER relajado (rTER) y D-TER relajado (rD-TER), donde la localización se considera correcta si la distancia al *ground truth* es menor que 50[m].

Tres experimentos se llevan a cabo para analizar el comportamiento y desempeño de los métodos de localización. El primer experimento compara y caracteriza los métodos de localización con respecto a su desempeño en el mapa topológico conociendo la pose inicial del vehículo, mientras que el segundo experimento analiza la convergencia de los métodos de localización cuando la posición inicial es desconocida. Finalmente, el tercer experimento evalúa el uso del Filtro de Partículas como un método de localización híbrido topológico-métrico y compara sus resultados con otro métodos existentes.

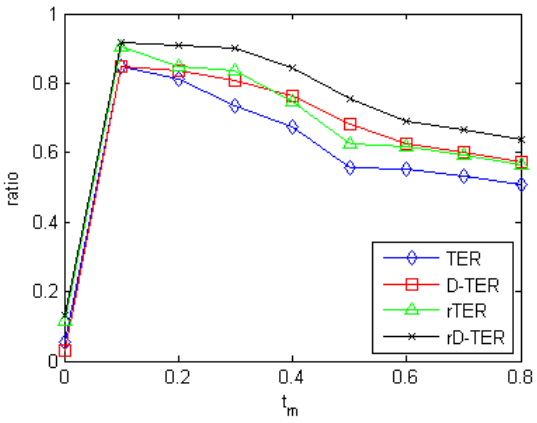
Para estos experimentos, el GTSM se construye con el método de mapeo descrito en el Algoritmo 4.2 sobre la primera secuencia de la base de datos prestada en la sección 5.2, utilizando las etiquetas de *Fondos* y *Otros* en el vector de características, ya que las etiquetas clasificadas como *Pisos* son ocluidas constantemente por los objetos dinámicos. Los parámetros utilizados para este experimento fueron: Número máximo de imágenes por observación $N_f = 5$, umbral de orientación $t_o = \pi/6$ y umbral de observación $t_s = 0,3$. El umbral de mapeo se elige según los resultados del primer experimento. El filtro de partículas utiliza N_p =1.000 partículas. Los detalles de cada experimento son explicados a continuación.

Experimento 1: compara y caracteriza el desempeño general de los métodos de localización, basado en las métricas de desempeño para distintas configuraciones. Aquí se evalúa el efecto que tiene el umbral de mapeo t_m sobre el tiempo de ejecución de los métodos, el número de aristas en el mapa y las cuatro métricas de desempeño. La pose inicial es conocida en este experimento.

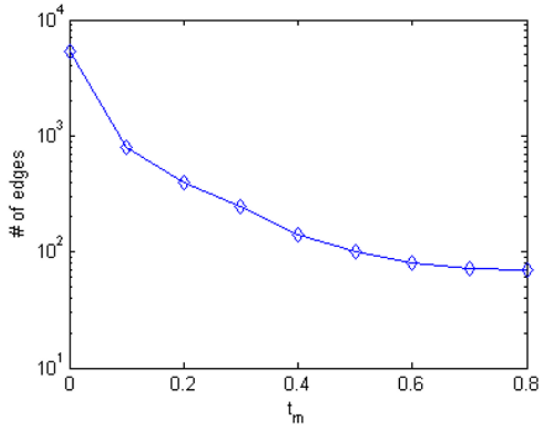
Los valores de t_m utilizados varían desde $0,0$ hasta $0,8$. Para $t_m = 0,0$ el método de mapeo crea una nueva arista por cada observación semántica, mientras que con valores de $t_m = 0,8$ o mayor, se puede apreciar que el método de mapeo deja de discriminar basado en la información visual y



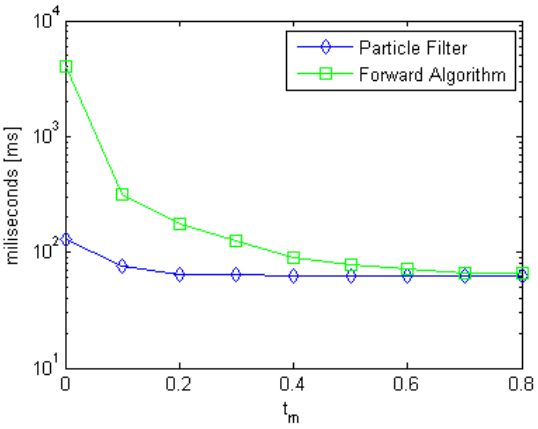
(a) Desempeño del Filtro de Partículas



(b) Desempeño del *Forward Algorithm*



(c) Aristas en GTSM



(d) Tiempo de Ejecución

Figura 5.3: Desempeño general de los métodos de localización según el umbral de mapeo t_m. (a) y (b) muestran las métricas de desempeño para el Filtro de Partículas y el *Forward Algorithm* respectivamente. (c) es el número de aristas en el GTSM en escala logarítmica, y (d) es el tiempo de ejecución en escala logarítmica para ambos métodos de localización.

solo genera nodos cuando hay cambios importantes de orientación o intersecciones. Los resultados de este experimento se presentan en la Figura 5.3.

De acuerdo a los resultados mostrados en las Figuras 5.3(a) y 5.3(b) la precisión de ambos métodos está relacionada con el umbral de mapeo. Los resultados de los métodos de localización están fuertemente relacionados con cuan representativo es la descripción acumulada de una arista. Ya que las aristas representan segmentos de camino con descripciones una descripción semántica común, basada en el umbral de mapeo t_m, un valor permisivo de este umbral hace que la descripción sea demasiado general y por lo tanto no sea suficientemente distintivo como para permitir la localización, mientras que un valor adecuado permite a los métodos de localización lograr una localización precisa del vehículo en el GTSM. El *Forward Algorithm* alcanza un máximo desempeño con $t_m = 0,1$, mientras que el Filtro de Partículas alcanza su máximo desempeño con un umbral de mapeo $t_m = 0,3$. Estos resultados muestran que el uso del mapa topológico, en el que el mundo está segmentado en regiones semánticamente uniformes, mejora la estimación de la posición obtenida por los métodos de localización cuando es comparada con los resultados obtenidos cuando el mapa está representado por observaciones individuales.

El Filtro de Partículas logra resultados significativamente mejores que los obtenidos por el *Forward Algorithm* (una diferencia de $\sim 10\,\%$), para todas las métricas de desempeño con el umbral de mapeo entre 0,0 y 0,4. Para valores más altos de este umbral, la información semántica contenida en las aristas del mapa no es suficientemente representativa para permitir que los métodos de localización propuestos sigan exitosamente la posición del vehículo.

Como se observa en la Figura 5.3(c), el número de aristas en el GTSM aumenta exponencialmente cuando el umbral de mapeo disminuye, lo que significa que la memoria utilizada por el mapa está relacionada con este parámetro. La Figura 5.3(d) confirma que el tiempo de ejecución del *Forward Algorithm* está fuertemente relacionado con el número de aristas en el mapa, mientras que el tiempo de ejecución del Filtro de Partículas está relacionado con el número de partículas y la dispersión de éstas, pero no en el tamaño del mapa. Ya que el sistema esta pensado para funcionar a 10 imágenes por segundo, el tiempo de procesamiento logrado por el Filtro de Partículas con umbral de mapeo $t?m \geq 0{,}2$ ($\simeq 60$[ms]) es adecuado para su uso en conducción autónoma. Sin embargo, este tiempo puede ser mejorado con implementación optimizada del código actual desarrollado en Matlab. De acuerdo a estos resultados, el Filtro de Partículas representa una solución más adecuada que el *Forward Algorithm* para localización en mapas a gran escala. El Filtro de Partículas representa la mejor solución cuando se consideran tanto el tiempo de ejecución como el desempeño.

Cabe destacar que el tiempo de procesamiento del método de segmentación semántica es de 4[s]

Tabla 5.1: Resultado del Experimento 1 de acuerdo a la Razón de Verdaderos Estimados (TER), la Razón de Verdaderos Estimados según Distancia (D-TER), la TER relajada (rTER), la D-TER relajada (rD-TER), el Error Medio (EM) y el EM de los Falsos Estimados, para cada método de localización.

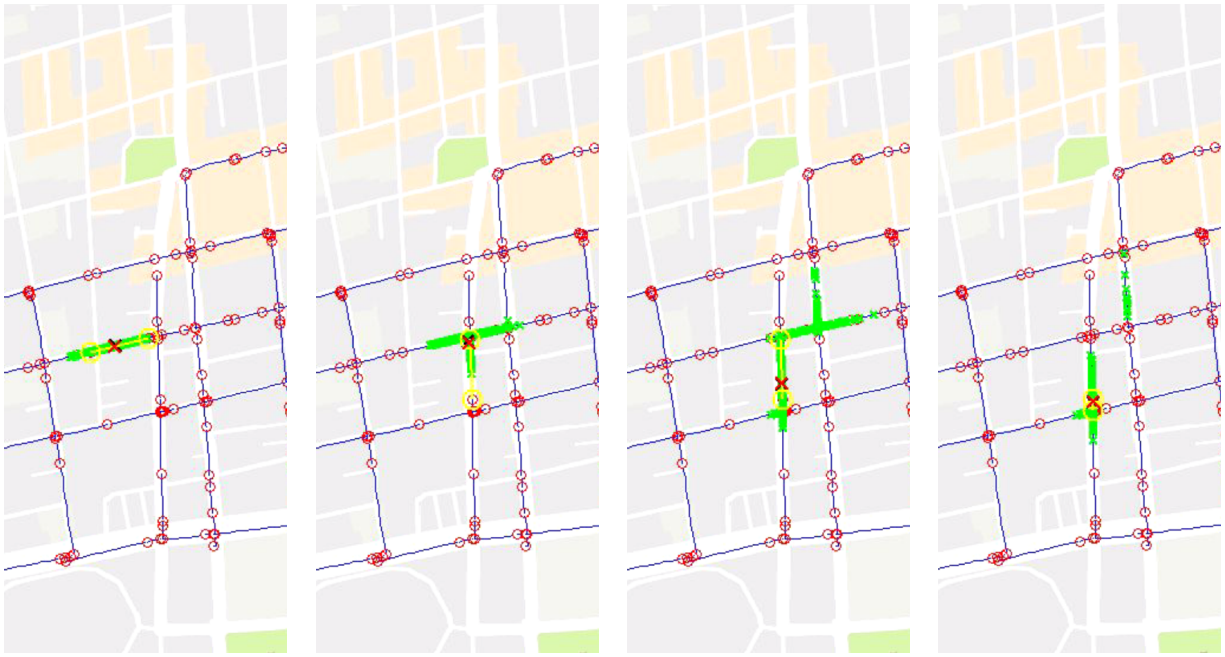
Método	Particle Filter $t_m = 0,3$	Forward Algorithm $t_m = 0,1$
Razón de Verdaderos Estimados (TER)	**0,9690**	0,8472
TER según Distancia (D-TER)	**0,9867**	0,8475
TER relajada (rTER)	**0,9998**	0,9030
D-TER relajada (rD-TER)	**0,9996**	0,9186
Error Medio (EM) [m]	**3,9**	14,6
EM de los Falsos Estimados [m]	**25,3**	72,7

por imagen, sin embargo métodos existen métodos que reportan hasta un tiempo de 20[ms] y mejor desempeño dentro del benchamark publicado en la pagina web de la base de datos Cityscapes, por lo que es posible implementar el método propuesto para su uso en tiempo real.

En la Tabla 5.1 se muestran las métricas de desempeño para la mejor configuración de cada método de localización. Además de las métricas definidas anteriormente, se incluye dos nuevas métricas: el Error Medio (EM) y el EM de los Falsos Estimados. El EM se define como la distancia promedio entre el *ground truth* y la pose estimada, sobre toda la secuencia. El EM de los Falsos Estimados se calcula de la misma manera, pero sólo se consideran las observaciones que generan estimaciones de localización incorrectas.

El Filtro de Partículas muestra mejores resultados que el *Forward Algorithm* para todas las métricas de desempeño utilizadas, con más de un 12 % de diferencia para las métricas TER y D-TER, y más de un 8 % de diferencia para las métricas relajadas rTER y rD-TER. El Error Medio (EM) del *Forward Algorithm* es 3 veces mayor que del Filtro de Partículas, mientras que el EM de los Falsos Estimados es 2,8 veces mayor. Además, el EM obtenido para el Filtro de Partículas es menor que los 5,8[m] obtenidos en [65] y similar a los presentados en [14]. La alta precisión obtenida por el Filtro de Partículas, de acuerdo con las métricas de desempeño, muestra que la metodología propuesta es adecuada para su uso en vehículos autónomos y ADAS para resolver el problema de localización de vehículos en ambientes urbanos.

Otro resultado de este experimento es que el D-TER es mayor que el TER, lo que se explica a través de que el vehículo suele detenerse cerca de las intersecciones con semáforos que separan aristas en el mapa. Esto lleva a numerosas observaciones que son consideradas como erróneas según la TER, ser consideradas con correctas en el D-TER. Los altos valores obtenidos para las métricas relajadas y el EM de los Falsos Estimados muestra que cuando la pose estimada es incorrecta, esta



(a) (b) (c) (d)

Figura 5.4: Secuencia de imágenes que muestran el comportamiento del filtro de partículas en las intersecciones. El GTSM se muestra con círculos rojos para los nodos y líneas azules para las aristas. Las cruces verdes representan partículas, la posición GPS asociada con el vehículo se muestra con una cruz roja y la arista estimada como pose se muestra en amarillo. (a) muestra a la mayoría de las partículas en la pose estimada con el vehículo moviéndose a la derecha. (b) muestra al vehículo después de haber girado a la derecha y a las partículas dividiéndose entre las posibles salidas de la intersección. (c) muestra como el filtro de partículas mantiene múltiples hipótesis de la posición del vehículo, eligiendo a la correcta gracias a las observaciones. Y (d) muestra como el proceso de resampling descarta la mayor parte de las partículas que no corresponden a la hipótesis correcta.

se encuentra cerca de la pose correcta en el mapa topológico. Esto permite al Filtro de Partículas reponerse rápidamente en caso de errores de localización. Un ejemplo del comportamiento de la localización basada en Filtro de Partículas se muestra en la Figura 5.4, con un umbral de mapeo de $t_m = 0,3$, que resulta en un GTSM con 232 aristas con largo promedio de 41,3[m]. Esta secuencia muestra la dispersión de las partículas cuando un vehículo atraviesa una intersección y como la etapa de resampling del filtro descarta la mayoría de las partículas erróneas. La secuencia completa se encuentra disponible en [7].

Experimento 2: evalúa la convergencia de los métodos de localización cuando la posición inicial del vehículo es desconocida. Aquí, el *Forward Algorithm* es inicializado con una distribución de probabilidades uniforme para la pose sobre las aristas del GTSM y el Filtro de Partículas es inicializado con una partícula cada 5[m] en todas las aristas de GTSM (∼2.000 partículas) para asegurar que todos los posibles estados iniciales estén cubiertos. La segunda secuencia de la base de datos

Tabla 5.2: Resultados del Experimento 2, con posición inicial desconocida.

Method	Tiempo Promedio de Localización [s]	Tasa de Éxito[%]
Filtro de Partículas	31,1	**100 %**
Forward Algorithm	**23,9**	**100 %**

presentada en la sección 5.2 es utilizada para la prueba, pero el experimento parte de posiciones aleatoria dentro de ésta. Un total de 1.000 repeticiones fueron realizadas para cada método de localización y dos valores son obtenidos de cada uno: el *Tiempo de Localización* y el *Éxito*. El *Tiempo de Localización* es el tiempo transcurrido desde que se inicia el experimento hasta que se alcanza una localización correcta por al menos 5[s], como en [14]. El *Éxito* es es 1 si el método alcanza una localización correcta durante la ejecución del experimento y 0 en caso contrario. La evaluación de este experimento incluye 2 métricas: la *Tasa de Éxito* y el *Tiempo Promedio de Localización*. La *Tasa de Éxito* es el *Éxito* promedio sobre las repeticiones del experimento y el *Tiempo Promedio de Localización* es el promedio del *Tiempo de Localización* sobre las repeticiones exitosas.

Los resultados de este experimento, resumidos en la Tabla 5.2, muestran que el Filtro de Partículas requiere un 30 % más de tiempo para converger a una estimación correcta que el *Forward Algorithm*, mientras que ambos consiguen una estimación correcta en el 100 % de las repeticiones. El *Forward Algorithm* actualiza la probabilidad conjunta para todas las aristas del mapa, por lo que es esperable un mejor comportamiento cuando la posición inicial es desconocida. Mientras que el Filtro de Partículas es propenso a alcanzar mínimos locales erróneos. Para resolver este problema se incorpora una etapa de re-inicialización, en el que se vuelven a distribuir las partículas cada 5[m] en todas las aristas del GTSM si la pose resultante tiene baja verosimilitud durante varias observaciones. Una consecuencia negativa del uso de esta etapa es un aumento significativo del tiempo de localización. Para este experimento, un 92,4 % de las repeticiones alcanzaron una localización exitosa sin la necesidad de la etapa de re-inicialización, lo que demuestra la importancia de este paso para una operación robusta del método de localización.

Experimento 3: estudia el desempeño de la metodología propuesta basada en el Filtro de Partículas como un método de localización métrico. Si bien, esta metodología fue diseñada para funcionar de manera topológica, el Filtro de Partículas permite estimar una pose métrica del vehículo basada en la posición GPS de los nodos del GTSM. Por otro lado, la metodología basada en Forward Algorithm no se incluye en este experimento porque no existe una manera directa de obtener una pose métrica a partir de su estimación. Para este experimento, cada nodo del GTSM incluye una pose GPS, y la pose métrica de cada partícula se estima a través de una interpolación lineal entre

Tabla 5.3: Resultados del Experimento 3 en localización métrica.

Metodo	Error Promedio de Posición [m]	Error GPS a Mapa [m]
Método Propuesto	**7.7**	2.50
Brubaker et al. [14]	34.6	6.97
Merriaux et al. [65]	51.9	6.97

las poses de los nodos de la arista en la cual está la partícula y la posición unidimensional de ésta. Finalmente, la pose métrica estimada del vehículo se obtiene como el promedio ponderado de las poses de las partículas.

Para llevar a cabo una comparación válida de los resultados obtenidos, se utilizan otros 2 métodos de localización basados en mapas topológicos. El primer método, propuesto por Brubaker et al. [14], usa un método de odometría visual para estimar el movimiento del vehículo y una mezcla de Gaussianas sobre la posición y la orientación para encontrar la posición del vehículo en un mapa de OpenStreetMaps (OSM). Por lo tanto, el desempeño de este método depende fuertemente de la geometría del lugar y de la trayectoria que siga el vehículo. En la implementación de este método se utilizó *libviso2* para obtener la odometría visual y el código del método de localización disponible en la pagina del autor. El segundo método, propuesto por Merriaux et al. [65], usa la odometría del vehículo y un filtro de partículas sobre un mapa de OSM para estimar la pose del vehículo, siendo igualmente dependiente de la geometría del lugar y la trayectoria del vehículo. Este método fue implementado para esta comparacion. Para realizar una comparación cuantitativa de los resultados de los métodos de localización se utiliza como métrica el Error Promedio de Posición, calculado como el promedio sobre la base de datos de la distancia entre la pose de la imagen y la pose estimada. Adicionalmente, el mismo valor es calculado sobre el punto mas cercano en el mapa, con el fin de ilustrar el error del GPS con respecto al mapa utilizado, GTSM u OSM.

La Tabla 5.3 muestra el Error Promedio de Posición para los métodos comparados. Se observa que el método propuesto basado en Filtro de Partículas tiene un mejor desempeño que los otros métodos. La Figura 5.5 muestra las trayectorias obtenidas con los distintos algoritmos de localización, junto con las poses GPS utilizadas como *ground truth*. La trayectoria generada por el método de Merriaux et al. [65] cambia bruscamente de estimación generando saltos entre diversas regiones del mapa debido a que su filtro de particulas mantiene diversas hipotesis en el mapa que no puede ser descartados con la odometría del vehículo debido a la simetría del mapa. Por otro lado, el método propuesto por Brubaker et al. [14] mantiene una estimación mas estable de la posición del vehículo, pero cuando el vehículo toma una curva después de un tramo recto, la alta varianza de la estimación de la pose hace que el método genere mas de una hipótesis en calles paralelas cercanas,

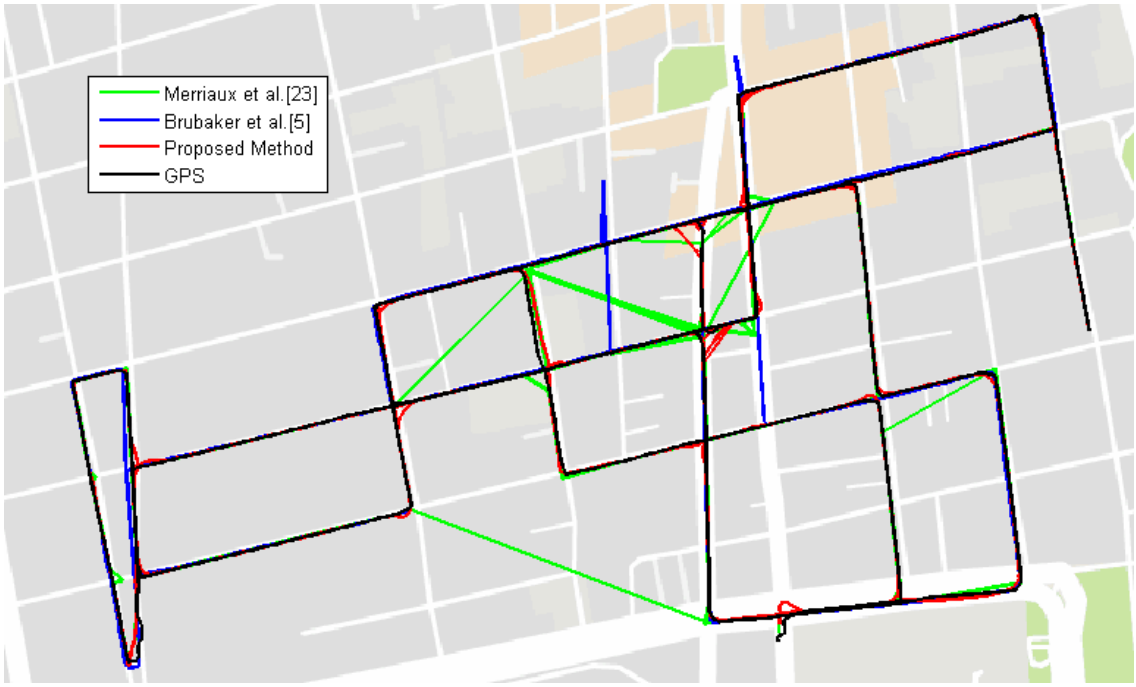


Figura 5.5: Estimated trajectory for topological map-based localization algorithms. On red the estimated trajectory from the Particle Filter proposed method, on green the method proposed by Merriaux et al. [65], on blue the trajectory from Brubaker et al. [14], and on black the GPS trajectory used as ground truth.

y llevando la estimación de la pose a la hipótesis incorrecta. La estimación es corregida cuando la trayectoria del vehículo hace improbable la estimación equivocada y retomando la hipótesis correcta. Es importante destacar que el método propuesto sigue la localización del vehículo durante toda la secuencia con solo algunos errores en intersecciones. Algunos ejemplos de esto se observan en la Figura 5.6.

El bajo rendimiento logrado por los metodos de Merriaux et al. [65] y Brubaker et al. [14] estan relacionados con la complejidad de la base de datos, grabada sobre calles que dibujan un grilla aproximadamente cuadrada. Ya que ambos métodos dependen de la odometría del vehículo para actualizar la estimación de la pose, estos solo lo pueden corregir sus estimación cuando el vehículo dobla, o cuando se espera que este lo haga. Esto lleva a estimaciones erróneas cuando el vehículo sigue una trayectoria recta muy larga. Sin embargo, el método propuesto basado en Filtro de Partículas utiliza continuamente la información semántica de las imágenes para corregir la estimación de la pose, sin importar la trayectoria del vehículo.

De acuerdo a los resultados obtenidos en los experimentos realizados, el Filtro de Partículas muestra el mejor desempeño cuando la posición inicial es conocida, mientras que el *Forward Algorithm* es el más rápido para converger cuando la posición inicial es desconocida. El método de

localización basado en filtros de partículas representa la solución más robusta para su utilización en vehículos autónomos y ADAS. Finalmente, el método propuesto basado en Filtro de Partículas muestra excelentes resultados cuando se compara como un método de localización métrica con otros métodos de localización basados en mapas topológicos.

(a) (b) (c) (d) (e) (f)

Figura 5.6: Ejemplos de errores en el método propuesto. (a) error asociado a la conducción en una pista distinta en la misma calle. (b) efecto de la dispersión de las partículas en torno a las intersecciones. (c) y (d) errores asociados con el GPS, ya que el ground truth yace fuera de la calle en la imagen satelital. (e) y (f) errores en la estimación después de una intersección, que son rápidamente corregidos.

Capítulo 6

Aplicación de Metodología de Mapeo y Localización en Caminos no Pavimentados

En este capítulo se presenta el trabajo realizado para implementar una variante de la metodología presentada en el capítulo 4 para su utilización en caminos no pavimentados. Cabe destacar que este trabajo fue realizado antes de la implementación en ciudad, y representa las bases sobre las cuales se llego al resultado del capítulo 4.

Debido a que los caminos no pavimentados presentan una gran variabilidad en su descripción visual, resulta necesario construir un método de permita identificarlos en una imagen. En la sección 6.1 se presenta una metodología para la segmentación de la región de camino utilizando un modelo adaptivo, que permite al sistema de mapeo reconocer una gran variedad de caminos no pavimentados, además de las bases de datos de caminos no pavimentados generadas para la evaluación de los métodos propuestos. Luego, en la sección 6.2, se presenta la implementación del método de construcción de mapas topológicos semánticos en caminos no pavimentados, junto con los resultados obtenidos en las bases de datos. Finalmente, en la sección 6.3, se presenta el método de localización en mapas topológicos semánticos en caminos no pavimentados y los resultados obtenidos en la base de datos.

6.1. Segmentación de Caminos no Pavimentados

Se utiliza un método de segmentación de camino basado en un clasificador bayesiano y se lleva a cabo una comparación de las distintas variantes de características de color, textura y de agrupación de píxeles [9]. En este método se define una región fija de camino para actualizar el modelo de *camino* y se utiliza la salida del clasificador para actualizar un modelo de *no-camino* (ver Figura 6.1). La comparación se lleva a cabo con dos secuencias de imágenes de conducción en caminos no

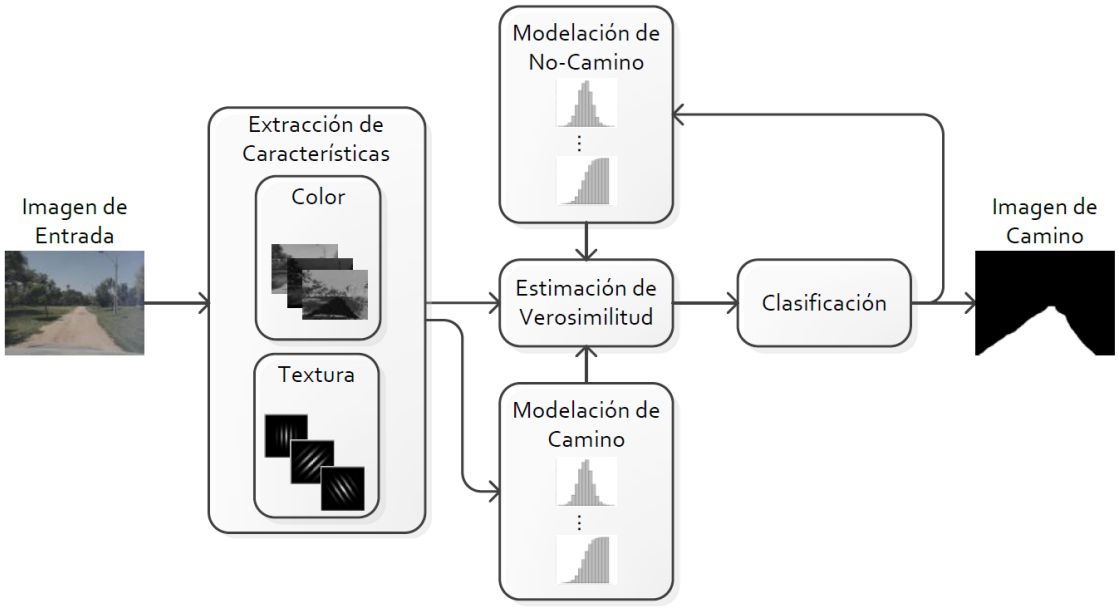


Figura 6.1: Diagrama de bloques del método de segmentación de caminos.

pavimentados.

Debido a que existen diversos tipos de suelo que pueden ser identificados como caminos, además del efecto que tiene la iluminación natural, la estrategia de segmentación explorada en esta comparación es adaptativa y no basada en un modelo a priori. Para esto se utiliza un clasificador ingenuo de Bayes adaptativo.

La metodología utilizada para la segmentación no supervisada de caminos, resumida en la Figura 6.1, corresponde al trabajo previo [6] en el que se utiliza exclusivamente características de color. Esta metodología consiste en cinco etapas: Extracción de Características, Modelación de Camino, Modelación de No-Camino, Estimación de Verosimilitud y Clasificación. La etapa de Extracción de Características recibe las imágenes a color de la cámara monocular frontal montada en el vehículo y extrae características de color y/o textura. La etapa de Modelación de Camino utiliza una región fija frente al vehículo para actualizar el modelo de camino y la etapa de Modelación de No-Camino utiliza la salida de la última imagen para actualizar el modelo de No-Camino. Estas dos etapas permiten al sistema ser adaptativo a cambios en el camino. La etapa de Estimación de Verosimilitud calcula una razón de las verosimilitudes de las clases de Camino y No-Camino para cada píxel en la imagen, creando una imagen de verosimilitudes representadas como un valor de punto flotante para cada píxel. La etapa de Clasificación utiliza la imagen de verosimilitudes para crear una imagen binaria que rescata la región de la imagen que representa el camino.

Sea la probabilidad de un píxel con un vector de características $\vec{x_F} = [x_1, x_2..., x_n]$ de pertenecer a la clase C es $P(C \mid \vec{x_F})$. Suponiendo independencia entre las características (el supuesto del

clasificador ingenuo de Bayes), es posible calcular esta probabilidad como

$$P(C \mid \vec{x}_F) = \frac{P(C)}{Z} \prod_{i=1}^{n} P(x_i \mid C), \tag{6.1}$$

donde $P(C \mid \vec{x_F})$ es la probabilidad de la clase C dado un píxel con características $\vec{x_F}$, $P(x_i \mid C)$ es la verosimilitud de la i-ésima característica, $P(C)$ es la probabilidad a priori de la clase C y $Z = P(\vec{x_F})$ es la evidencia, que en este caso es un factor de normalización, ya que los valores de $\vec{x}_F$ son conocidos.

Siguiendo el paradigma del clasificador ingenuo de Bayes, las probabilidades posteriores para las clases de camino y no-camino son comparadas como un cociente. Al utilizar este cociente, los factores de normalización se cancelan y el cociente entre las probabilidades a priori de las clases es una constante desconocida k_{prior}. Este cociente se calcula para cada píxel en la imagen, generando una imagen de cocientes de verosimilitudes I_L.

Las distribuciones de probabilidad de las verosimilitudes de las clases de camino y no-camino son aproximadas por los histogramas normalizados H_R y H_{NR} [22]. La imagen de cocientes de verosimilitudes se calcula como:

$$I_L(\vec{x}_F) = \frac{P(Road \mid \vec{x}_F)}{P(Non-Road \mid \vec{x}_F)} \approx k_{prior} \prod_{i=1}^{n} \frac{H_R^i(\vec{x}_F)}{H_{NR}^i(\vec{x}_F)}. \tag{6.2}$$

Para tomar la decisión sobre a cuál clase pertenece cada píxel, se compara el cociente de verosimilitudes de cada píxel con un umbral fijo. Si el cociente es mayor que el umbral, entonces el píxel pertenece a la clase de camino.

$$I_{CL}(i,j) = \begin{cases} 1 & \text{si } I_L(i,j) \geq T_{CL} \\ 0 & \text{en caso contrario} \end{cases}, \tag{6.3}$$

donde I_{CL} es la imagen clasificada, I_L es la imagen de cocientes de verosimilitudes, y T_{CL} es el umbral fijo, un parámetro del sistema.

La actualización del histograma de la verosimilitud de la clase C (camino o no-camino) se actualiza con un histograma normalizado H_T de los píxeles seleccionados de la imagen $k-1$, como

$$H_C^{i,k} = \alpha \cdot H_C^{i,k-1} + (1-\alpha) \cdot H_T^{i,k-1}, \tag{6.4}$$

donde $H_C^{i,k}$ es el histograma de la característica i de la imagen k para la clase C, y α es el peso de la actualización.

6.1.1. Variantes del Método de Segmentación

A continuación, se describen las distintas variantes del método de segmentación de caminos propuesto. Dentro de las variantes se incluye el uso de distintas características de color, características de textura, de características conjuntas de color y textura, además de dos métodos de agrupación de píxeles para llevar a cabo una clasificación sobre regiones en vez de píxeles.

Características de Color

Las variantes del método basadas en características de color buscan encontrar el mejor espacio de color para la segmentación de camino. Siete diferentes espacios de color son utilizados como características de color: RGB, HSV, HSL, CIE-XYZ, YCrCb, CIE-Lab, y CIE-Luv.

Cada espacio de color describe el color del píxel a través de tres valores o canales. Los canales de un espacio de color son utilizados para la construcción del vector de características, creando un vector de tres dimensiones para cada espacio de color. Para un espacio de color con canales A,B y C, el vector de características $\vec{x}_{ABC}$ del píxel ubicado en (x, y) se define como:

$$\vec{x}_F(x,y) = \begin{bmatrix} A(x,y) \\ B(x,y) \\ C(x,y) \end{bmatrix} \tag{6.5}$$

.

Características de Textura

La textura de una región de la imagen se refiere a la disposición espacial de las intensidades de los píxeles. Cuatro métodos de extracción de texturas son estudiados: (i) Filtros de Gabor, (ii) *Gray Level Coocurrence Matrix* (GLCM), (iii) *Local Binary Patterns* (LBP) y (iv) *Gaussian Markov Random Fields* (GMRF).

Filtros de Gabor: son filtros lineales que se asemejan a los campos receptivos visuales humanos por su representación de frecuencia y orientación. La respuesta al impulso de un filtro bi-dimensional de Gabor es una onda sinusoidal multiplicada por una función Gaussiana:

$$\begin{aligned} h\left(x, y; \lambda, \theta, \psi, \sigma, \gamma\right) &= exp\left(-\frac{x'^2+\gamma^2 \cdot y'^2}{2\sigma^2}\right) \cdot \cos\left(\frac{2\pi x'}{\lambda} + \psi\right) \\ x' &= x \cdot \cos(\theta) + y \cdot sin(\theta) \\ y' &= -x \cdot sin(\theta) + y \cdot \cos(\theta), \end{aligned} \tag{6.6}$$

donde λ es la longitud de onda de la función sinusoidal, θ es la orientación, ψ es el desfase, σ es la desviación estándar de la función Gaussiana, y γ es el factor elíptico para la forma del filtro.

Un total de 24 filtros son aplicados sobre la intensidad de la imagen, para obtener un vector de características de 12 dimensiones para cada píxel de la imagen. Los parámetros de los filtros son una combinación de: $\lambda = \{4, 8, 12\}$, $\theta = \{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\}$ y $\psi = \{0, \frac{\pi}{2}\}$, con valores fijos de $\sigma = \frac{\lambda}{2}$ y $\gamma = 0$. Las características utilizadas son la energía total de respuesta de los filtros para los distintos valores de λ y θ, que es la raíz cuadrada de la suma de los cuadrados de la convolución ($*$) sobre los valores de ψ:

$$E_{\lambda,\theta} = \sqrt{\left(h\left(\lambda, \theta, 0, \frac{\lambda}{2}, 0\right) * I\right)^2 + \left(h\left(\lambda, \theta, \frac{\pi}{2}, \frac{\lambda}{2}, 0\right) * I\right)^2}. \tag{6.7}$$

Por lo tanto, el vector de características $\vec{x}_{GF}$ para los filtros de Gabor es:

$$\vec{x}_{GF_{i,j}}(x, y) = E_{\lambda_i,\theta_i}(x, y), \tag{6.8}$$

con $i = 1, .., 3$ y $j = 1, .., 4$.

GLCM: El método de *Gray Level Coocurrence Matrix* (o Matriz de coocurrencia de niveles de grises) propuesto por Haralick [41] se basa en la construcción de histogramas de combinaciones de intensidades de píxeles para distintas distribuciones espaciales, para el cálculo posterior de algunas estadísticas sobre estos histogramas que son usados como características de texturas.

Sea $P_{(a,b)}$ el histograma o matriz de coocurrencia para una distribuciones espacial (a, b) en un segmento S de la imagen I. El elemento de la matriz de coocurrencia ubicado en las coordenadas (i, j) se calcula como:

$$P_{(a,b)}(i, j) = \#\left\{((k, l), (m, n)) \in S \mid \begin{array}{l} (k, l) - (m, n) = \pm(a, b) \\ \wedge I(k, l) = i \wedge I(m, n) = j \end{array}\right\}, \tag{6.9}$$

donde # es la cardinalidad de un conjunto. Para este trabajo esta basado en la metodología presentada en [98], donde se consideran matrices de coocurrencia simétricas, por lo que las matrices de coocurrencia para las distribuciones espaciales (a,b) y (b,a) son iguales.

Se utilizan tres estadísticas de las matrices de coocurrencia: segundo momento angular (ASM), contraste y correlación:

$$GLCM_{(a,b)} = \begin{bmatrix} ASM_{(a,b)} \\ Cont_{(a,b)} \\ Corr_{(a,b)} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \left(\frac{P_{(a,b)}(i,j)}{R}\right)^2 \\ \sum_{n=0}^{N_g-1} \left(n^2 \sum_{|i-j|=n} \frac{P_{(a,b)}(i,j)}{R}\right) \\ \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i \cdot j \cdot P_{(a,b)}(i,j)/R) - \mu_x \cdot \mu_y}{(\sigma_x \cdot \sigma_y)} \end{bmatrix}, \tag{6.10}$$

donde μ_x, μ_y, σ_x, y σ_y son las medias y las desviaciones estándar de las distribuciones marginales de $P(i, j)/R$, y R es una constante de normalización.

Las intensidades de color son discretizadas a 32 valores y la imagen es dividida en segmentos superpuestos de 32x32 píxeles generados cada 4 píxeles tanto horizontal como verticalmente. Para cada segmento se obtienen 4 matrices de coocurrencia y para cada una se obtienen las tres estadísticas anteriores. Con esto se genera un vector de características de tamaño 12 para cada segmento, y la imagen de características es completada con el vector de características calculado más cercano a cada píxel.

El vector de características $\vec{x}_{GLCM}$ se calcula como:

$$\vec{x}_{GLCM}(x,y) = \begin{bmatrix} GLCM_{(4,0)}(x,y) \\ GLCM_{(4,4)}(x,y) \\ GLCM_{(0,4)}(x,y) \\ GLCM_{(-4,4)}(x,y) \end{bmatrix}. \tag{6.11}$$

LBP: *Local Binary Pattern* (o Patrón Binario Local) es un descriptor de texturas efectivo basado en una descripción binaria de los valores en escala de grises de la vecindad local relativa a su centro. En este trabajo se utiliza una vecindad de 3x3 píxeles, y los 8 píxeles que rodean al centro son comparados con éste para generar la descripción binaria. La imagen LBP I_{LBP} se obtiene de la siguiente manera:

Sea S una vecindad local de 3x3 píxeles centrada en el píxel g_c ubicado en (x,y), y $g_0, ..., g_7$ los píxeles vecinos ordenados en sentido horario, partiendo por el píxel ubicado sobre el centro g_c. El valor del píxel (x,y) en la imagen LBP es:

$$I_{LBP}(x,y) = \sum_{i=0}^{7} s(g_i - g_c) \cdot 2^i, \text{with } s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{6.12}$$

Para cada píxel de la imagen de características se obtienen histogramas normalizados de la imagen LBP H_i^{LBP} utilizando 4 tamaños de ventanas (9x9, 17x17, 33x33 y 65x65). Estos histogramas son utilizados en el vector de características $\vec{x}_{LBP}$ como:

$$\vec{x}_{LBP}(x,y) = \begin{bmatrix} H_{9x9}^{LBP}(x,y) \\ H_{17x17}^{LBP}(x,y) \\ H_{33x33}^{LBP}(x,y) \\ H_{65x65}^{LBP}(x,y) \end{bmatrix}, \tag{6.13}$$

y el cociente de verosimilitudes para cada píxel se estima como:

$$I_L(x,y) = \prod_{i=(9x9)}^{(65x65)} \frac{D(H_i^{LBP}(x,y), H_R)}{D(H_i^{LBP}(x,y), H_{NR})}, \tag{6.14}$$

donde $D(H,G) = 1 - \sum_i |H_i - G_i|$ es la medida de distancia entre histogramas.

GMRF: Los parámetros del *Gaussian Markov Random Fields* (o Campos Aleatorios Gaussianos de Markov) son utilizados como descriptores de texturas, siguiendo la metodología propuesta en [89]. GMRF describe la dependencia de la intensidad $I(p)$ de un píxel p con la de sus vecinos como $I(p) = \sum_{r \in \Delta N} \theta_r \left(I(p+r) + I(p-r) \right) + e(p)$, donde $\Delta N = \{ r : p \pm r \in N(p) \}$ para una vecindad $N(p)$ de p, y $e(p)$ es ruido Gaussiano de media cero.

Los parámetros $\Theta = [\theta_1 \theta_2 \cdots \theta_{\#\Delta N}]$ del GMRF son estimados como:

$$\Theta^* = \left[\sum_{p \in \Omega_l} q(p) q(p)^T \right]^{-1} \left[\sum_{p \in \Omega_l} q(p) I(p) \right] \tag{6.15}$$

donde Ω_l es una region entorno al píxel p, y

$$q(p) = \begin{bmatrix} I(p+r_1) + I(p-r_1) \\ I(p+r_2) + I(p-r_2) \\ \vdots \\ I(p+r_n) + I(p-r_n) \end{bmatrix}, r_i \in \Delta N. \tag{6.16}$$

La varianza del ruido se calcula como $v^* = \frac{1}{\#\Omega_l} \sum_{p \in \Omega_l} \left(I(p) - \Theta^{*T} q(p) \right)^2$. Y el vector de características $\vec{x}_{GMRF}$ se define como:

$$\vec{x}_{GMRF}(x,y) = \left[\frac{\Theta^* v^*}{\rho^2} \right], \tag{6.17}$$

donde ρ es la varianza muestral del vector de la textura como vector de características $[\Theta^* v^*]^T$, ΔN es la región con forma de estrella [89] y Ω_l es una región de 11x11 píxeles.

Características Conjuntas de Color y Textura

Para esta variante del método se utilizan de manera conjunta las características de color y textura para crear el vector de características. Todas las combinaciones posibles entre las características de color y las de texturas presentadas anteriormente son estudiadas.

Para cada píxel de la imagen de entrada se calcula el vector de características conjuntas $\vec{x}_J$ como la concatenación del vector de características de color $\vec{x}_C$ y el vector de características de texturas $\vec{x}_T$:

$$\vec{x}_J = \begin{bmatrix} \vec{x}_C \\ \vec{x}_T \end{bmatrix}. \tag{6.18}$$

Segmentación Basada en Regiones

Existen trabajos publicados [97] [99] que han mostrado obtener mejoras significativas en la segmentación de camino al utilizar clasificación basada en regiones de la imagen. Para evaluar el

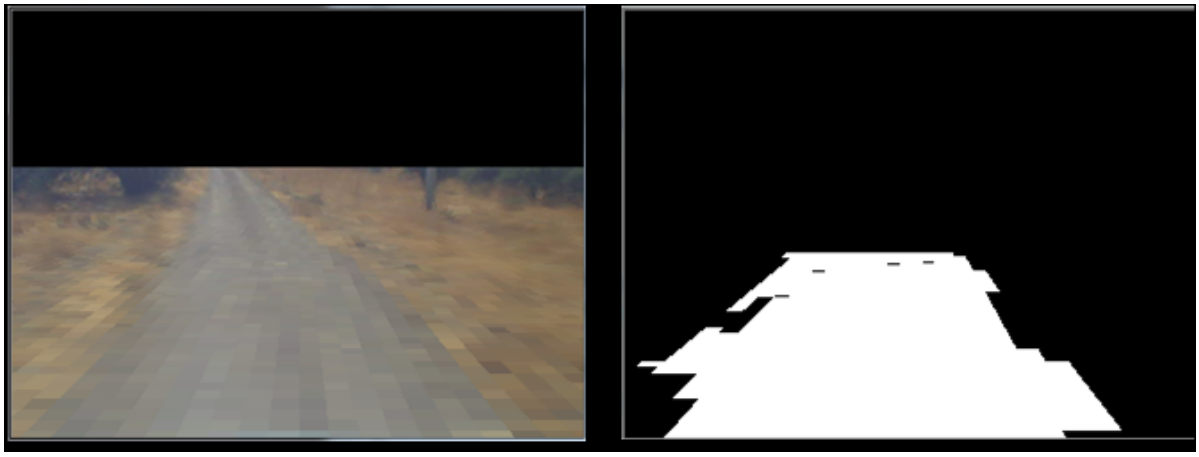


Figura 6.2: Ejemplo del clasificador basado en regiones para la segmentación de camino utilizando una grilla cuadrada sobre el suelo. A la izquierda el color promedio de cada región, demostrando el efecto de la utilización de la grilla sobre la información de la imagen, y a la derecha la imagen de salida del clasificador.

efecto de la utilización de información de contexto espacial en el clasificador, se comparan dos variantes basadas en regiones con los resultados obtenido con las variantes anteriores.

La primera variante basada en regiones llamada *Ground-Grid* utiliza una grilla cuadrada dibujada sobre el suelo, suponiendo un mundo plano y conociendo la pose de la cámara con respecto al vehículo para crear una grilla bidimensional cuadrada en el suelo (ver Figura 6.2). La segunda variante utiliza el método de segmentación no supervisada propuesto en [28] para dividir la imagen en regiones homogéneas llamadas superpíxeles. Las regiones definidas por estos métodos son utilizadas en la etapa de clasificación para clasificar cada región, en vez de cada píxel.

Segmentación Basada en Ground-Grid: La posición de cada píxel en la imagen se asocia con coordenadas en el suelo, utilizando la posición de la cámara y suponiendo un suelo plano. Los píxeles son agrupados de acuerdo a una grilla cuadrada en el suelo. El tamaño de la grilla es un parámetro del sistema. La imagen de la grilla se crea en base a la agrupación de píxeles por la grilla y la decisión del clasificador se lleva a cabo de acuerdo a cada grupo como:

$$C_{gg}(G) = \begin{cases} Road & \text{si } \frac{\#\{p \in G | I_{CL}(p)=1\}}{\#G} > T_r \\ Non-Road & \text{en caso contrario} \end{cases} . \tag{6.19}$$

La razón de píxeles de camino en cada grupo G de la imagen de la grilla se compara con un umbral T_r. Esta razón es el cociente entre el número de píxeles clasificado como camino por un clasificador basado en píxeles, y el total de píxeles en el grupo. Finalmente, un algoritmo de inundación [12] se utiliza para extraer el segmento de camino conectado con la región de entrenamiento (ver Figura 6.2).

Figura 6.3: Ejemplo del clasificador basado en superpíxeles para la segmentación de caminos. A la izquierda, la imagen de entrada utilizada para generar los superpíxeles. A la derecha la salida del clasificador con la división de los superpíxeles dibujados en gris.

Segmentación Basada en Superpíxeles: El método de segmentación no supervisada propuesto por Felzenszwalb [28] está basado en un grafo no dirigido donde los nodos del grafo son los píxeles de la imagen y las aristas conectan los píxeles vecinos. Cada arista tiene un peso asociado que mide la disimilitud entre dos píxeles, obtenida a través de un método de detección de bordes. En este método, la segmentación es una partición de los píxeles de una imagen en componentes conectados o regiones. La segmentación parte ordenando las aristas según su peso y definiendo cada píxel como un segmento, y evalúa una afirmación por cada arista, de menor a mayor, para saber si las regiones separadas por esta arista debiesen ser conectados o no. Esta afirmación compara la disimilitud interna de la región con la disimilitud entre las regiones. La disimilitud interna de una región C se define como el máximo peso en el Árbol de Expansión Mínimo (Minimum Spanning Tree, MST) de la región:

$$Int(C) = \max_{e \in MST(C)} w(e), \tag{6.20}$$

donde $w(e)$ es el peso de la arista e. La disimilitud entre las regiones C_1 y C_2 se define como el mínimo peso de las aristas que conectan ambas regiones:

$$Diff(C_1, C_2) = \min_{v_1 \in C_1, v_2 \in C_2} w(v_1, v_2). \tag{6.21}$$

La afirmación evalúa si la disimilitud entre las regiones es mayor que al menos una de las disimilitudes internas más una función de umbral:

$$D(C_1, C_2) = \begin{cases} true & \text{si } Diff(C_1, C_2) > \min\left(Int(C_1) + \frac{k}{\#C_1}, Int(C_2) + \frac{k}{\#C_2}\right) \\ false & \text{en caso contrario} \end{cases} \tag{6.22}$$

Donde k es un parámetro y en la práctica valores grandes de k produce regiones grandes. La

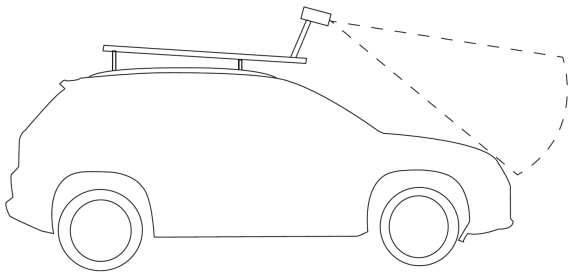


Figura 6.4: Posición de la cámara en el vehículo.

salida de este método de segmentación es un conjunto de regiones no superpuestas que dividen la imagen en superpíxeles homogéneos.

Para la segmentación basada en superpíxeles se utiliza un clasificador basado en píxeles: Por cada superpíxel S, se calcula la razón de píxeles de camino y se comparan con un umbral T_r. Si la razón es mayor que el umbral, entonces el superpíxel completo es etiquetado como camino, y si no, el superpíxel es etiquetado como no-camino (ver Figura 6.3).

$$C_{sp}(S) = \begin{cases} Road & \text{si } \frac{\#\{p \in S | I_{CL}(p)=1\}}{\#S} > T_r \\ Non-Road & \text{en caso contrario} \end{cases} \tag{6.23}$$

Finalmente, un algoritmo de inundación [12] se utiliza para extraer el segmento de camino conectado con la región e entrenamiento.

6.1.2. Bases de datos de Caminos no Pavimentados

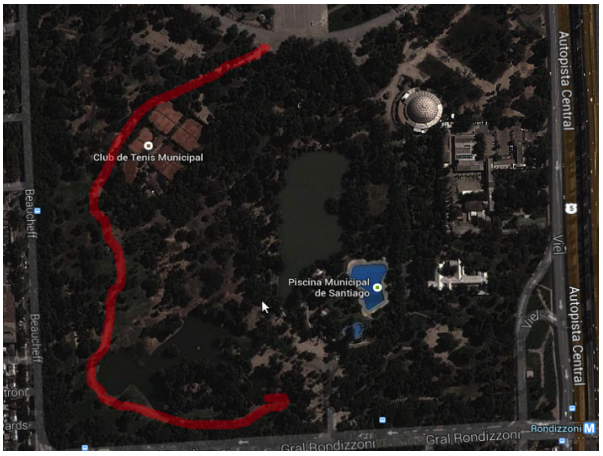
Con el fin de hacer una comparación válida entre los diferentes métodos que permita escoger la mejor combinación de características, se construyeron dos bases de datos etiquetadas manualmente. Estas bases de datos consisten en secuencias de imágenes adquiridas por el vehículo autónomo del Advanced Mining Technology Center (AMTC), un Volkswagen Tiguan 2010 [73] conducido en caminos no pavimentados, para lo cual se utilizó una cámara montada sobre el techo del vehículo (ver Figura 6.4).

La primera base de datos (DB1) se grabó en el predio de la Laguna Carén, un ambiente seco con árboles pequeños y arbustos, casi sin sombras o cambios de iluminación. El camino pasa por una región plana y termina al lado de un cerro. El material del camino es principalmente tierra con pequeñas zonas de pasto seco en el centro, y la mayor parte del terreno al lado del camino estaba cubierto de pasto seco, por lo que algunas partes el camino son difíciles de diferenciar de su entorno. Esta base de datos está compuesta por 2.600 imágenes, con una resolución de 320x240 píxeles grabados a 10 imágenes por segundo.

(a)

(b)

(c)



(d)

Figura 6.5: Bases de datos de caminos no pavimentados. (a) y (c) son ejemplos de las imagenes de ambas bases de datos. (b) y (d) son imagenes satelitales que muestrasn el recorrido de cada base de datos. (Fuente: Google Maps satellite image)

La segunda base de datos (DB2) es la base de datos utilizada por Parra-Tsunekawa et al. [73]. Esta base de datos fue grabada dentro del Parque O'Higgins, ubicado cerca del centro de Santiago de Chile, mientras el vehículo se conducía sobre un camino no pavimentado e irregular a baja ($< 10 \left[\frac{m}{s}\right]$) y media velocidad($< 20 \left[\frac{m}{s}\right]$). El camino tiene una longitud de 800$[m]$ aproximadamente con pendientes positivas y negativas. El camino está hecho de tierra rodeada de pasto y algunos árboles, por lo que existe una fuerte diferencia visual entre el camino y sus alrededores. Como algunos árboles están cerca del camino, algunas partes de éste están totalmente cubiertas de sombras, mostrando fuertes cambios de iluminación. Esta base de datos está compuesta por 7.532 imágenes, con una resolución de 640x480 píxeles.

Ambas bases de datos estarán disponibles en: http://vision.die.uchile.cl/databases.php

6.1.3. Resultados de Segmentación de Camino

Se evalúa el desempeño de las distintas variantes del método de segmentación de caminos no pavimentados presentado en esta sección. Las distintas variantes incluyen el uso de diferentes características, ya sea de color, textura o una mezcla de ambas, así como el uso de una segmentación basada en regiones en vez de una basada en píxeles.

Se realizaron experimentos para evaluar el rendimiento de los diferentes algoritmos y a continuación se presentan sus resultados. Cada experimento consiste en la ejecución del algoritmo de segmentación de caminos con diferentes características y regiones en cada una de las bases de datos. El resultado de la segmentación de caminos se utiliza para crear una curva de característica operativa del receptor (curva ROC, por sus siglas en inglés). Cada punto de la curva ROC es la tasa de verdaderos positivos (TPR, acrónimo de siglas en inglés) y la tasa de falsos positivos (FPR, acrónimo de siglas en inglés) de cada secuencia de imágenes para una configuración diferente. Por último, se definió una métrica de desempeño para cada experimento para realizar una comparación cuantitativa entre las curvas ROC obtenidas para los distintos métodos. Dado que el objetivo de esta comparación es encontrar un algoritmo seguro y robusto para la segmentación de caminos no pavimentados, la medición del desempeño debe estar relacionada con el comportamiento de las curvas ROC en valores FPR bajos, y no en toda la curva como el valor del área bajo la curva (AUC). Para este trabajo, se obtuvo el valor TPR10 para cada experimento como medida de rendimiento, donde el TPR10 es el valor TPR estimado que arroja un FPR del 10 %, que es un FPR aceptable dado que los bordes de los caminos no pavimentados suelen ser difusos y no estar bien definidos.

Todos los algoritmos fueron implementados en C++ usando la librería OpenCV. Las características de color se implementaron utilizando las transformaciones espacio de color de OpenCV. Las características de Gabor fueron implementadas basándose en los kernels de los filtros de Gabor

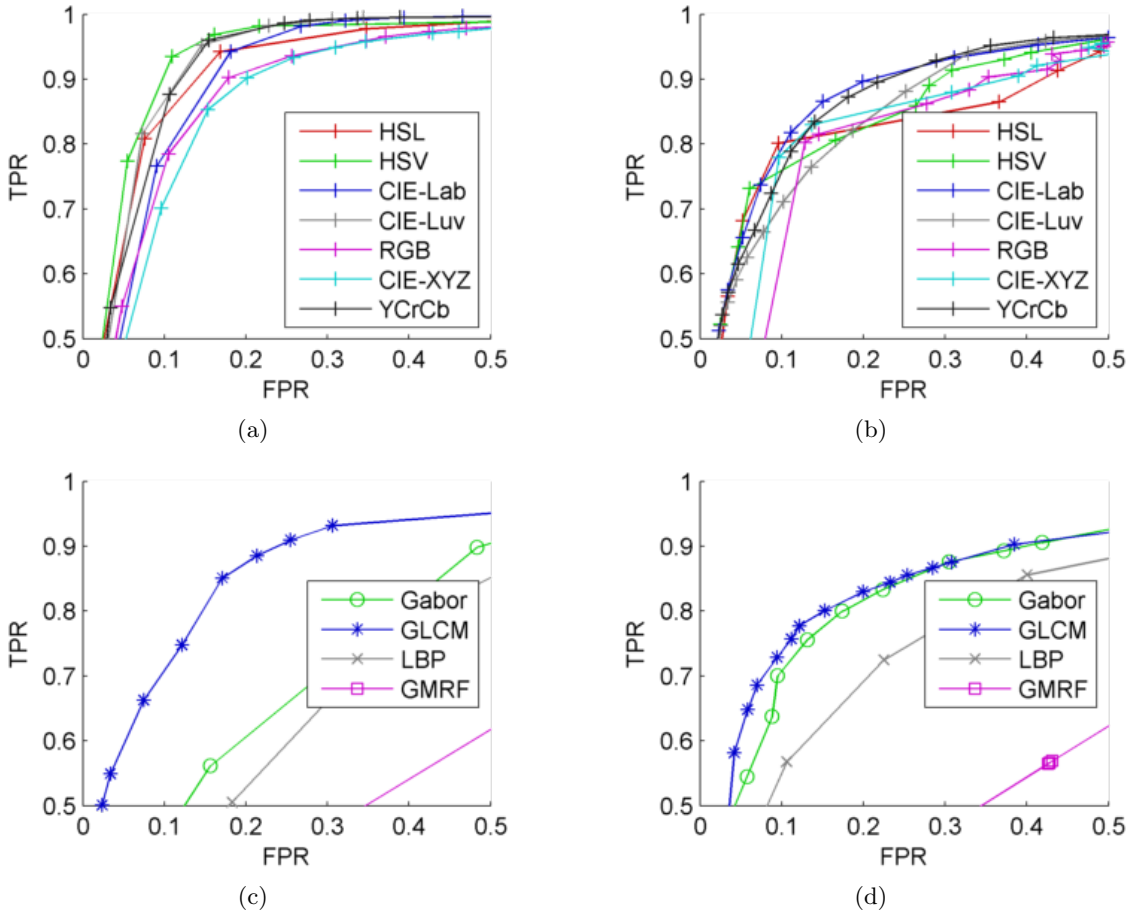


Figura 6.6: Curvas ROC con los resultados de los experimentos. (a) y (b) muestran las curvas ROC de los 7 algoritmos de características de color en DB1 y DB2 respectivamente. (c) y (d) muestran las curvas ROC de los 4 algoritmos de características de texturas en DB1 y DB2 respectivamente.

de OpenCV. El método de los superpíxeles fue implementado usando la segmentación C ++ de Pedro Felzenszwalb [28]. Las características de textura de GLCM, LBP y GMRF, el método de Ground-Grid, y el algoritmo de inundación son implementaciones propias realizadas para esta tesis.

Comparación de Características

En las Figuras 6.6 y 6.7, y en la Tabla 6.1 se muestran los resultados de los experimentos que comparan el uso de diferentes características de color, textura y las características conjuntas de color y textura en las DB1 y DB2. Al usar sólo características de color en la DB1, la característica de color HSV mostró los mejores resultados, seguido por CIE-Lab e YCrCb. En el caso de la DB2, los mejores resultados se obtuvieron al utilizar las características de HSL, seguidas de CIE-Lab y CIE-XYZ. Se puede observar que, debido a las diferentes características de las bases de datos utilizadas, el rendimiento de los algoritmos de segmentación que utilizan diferentes características de color cambian dependiendo de la base de datos que se utiliza y también que no existe una

característica de color que funcione como la mejor en ambas bases de datos. Por ejemplo, ninguna de las características de color puede manejar correctamente los cambios de iluminación y las sombras, por lo que los resultados de las características de color son mejores para la DB1 que para la DB2, ya que la primera tiene menos sombras que la segunda. Por lo tanto, en el caso de usar características de color, la selección del mejor espacio de color a utilizar dependerá de las condiciones ambientales.

En el caso de utilizar sólo características de textura, las características de GLCM obtuvieron los mejores resultados en ambas bases de datos, seguidas por las características de Gabor, y LBP. Se puede observar que al utilizar sólo características de textura se obtuvieron peores resultados en la segmentación que al hacerlo sólo con características de color. Por lo tanto, surge la siguiente pregunta: ¿hay algún beneficio en el uso de características conjuntas de color y textura, cuando las características de color se comportan siempre mejor? Como se puede ver en la Figura 6.7 y la Tabla 6.1, la respuesta es que el uso combinado de las características de color y textura permitió obtener resultados más estables, por ejemplo, se redujo la variabilidad en el rendimiento de los algoritmos y un algoritmo que funciona mejor en una base de datos puede obtener también un buen rendimiento en la otra. Como se puede observar en la Tabla 6.1, las características de Gabor + HSV mostraron los mejores resultados cuando las características de Gabor se usan en conjunto con diferentes características de color (ver la columna *+Gabor* en la Tabla 6.1). También es importante notar que casi todas las características de Gabor en conjunto con textura obtuvieron mejores resultados cuando se comparan con el caso de usar sólo las características de color correspondientes, siendo la única excepción el caso HSV en la DB1.

Cuando las características de LBP se usan en conjunto con diferentes características de color, los mejores resultados en ambas bases de datos son obtenidos por HSV+LBP (ver la columna *+LBP* en la Tabla 6.1). Es importante tener en cuenta que en este caso se mejora el rendimiento de todas las características de color en la DB2. De forma similar, cuando se usan características de GMRF junto con diferentes características de color, los mejores resultados en ambas bases de datos se obtienen mediante HSV+GMRF (véase la columna *+GMRF* en la Tabla 6.1). Sin embargo, en este caso la inclusión de características de GMRF no mejora los resultados obtenidos por el uso de sólo características de color, excepto para HSV+GMRF y RGB+GMRF en la DB2.

En el caso de utilizar las características de GLCM junto con características de color, HSV+GLCM obtiene los mejores resultados en DB1 seguido por RGB+GLCM, mientras que en DB2 CIE-XYZ+GLCM obtiene los mejores resultados, seguido también de RGB+GLCM. Se concluye entonces que RGB+GLCM es la mejor característica conjunta basada en GLCM, ya que logra resultados competitivos en ambas bases de datos (ver la columna *+GLCM* en la Tabla 6.1).

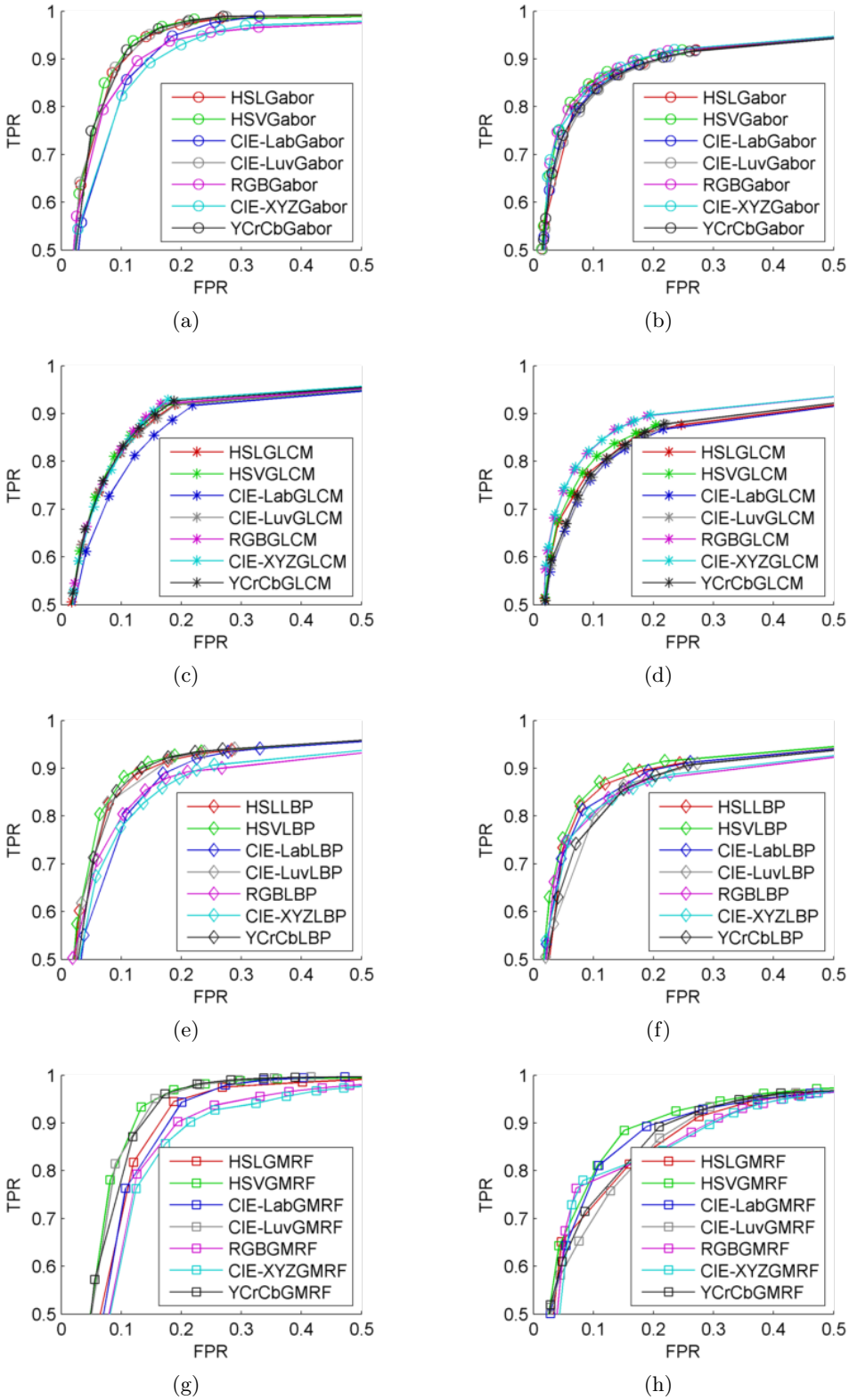


Figura 6.7: Curvas ROC con los resultados de los algoritmos de las características conjuntas de color y textura. (a) y (b) muestran los resultados de las características conjuntas con Gabor en DB1 y DB2 respectivamente. (c) y (d) muestran los resultados de las características conjuntas con GLCM en DB1 y DB2 respectivamente. (e) y (f) muestran los resultados de las características conjuntas de LBP en la DB1 y DB2 respectivamente. (g) y (h) muestran los resultados de las características conjuntas de GMRF en la DB1 y DB2 respectivamente..

Tabla 6.1: Resultados experimentales de los métodos basados en características de color y/o texturas para ambas bases de datos. Los valores en negrita indican los 3 mejores resultados para cada columna en términos del índice TPR10.

	Feature	Single	+Gabor	+GLCM	+LBP	+GMRF
DB1	HSL	0,8435	0,8918	0,8204	**0,8533**	0,7025
	HSV	**0,9078**	**0,9019**	**0,8310**	**0,8733**	**0,8355**
	CIE-Lab	0,7848	0,8224	0,7680	0,7712	0,7125
	CIE-Luv	**0,8669**	**0,8953**	0,8186	0,8473	**0,8354**
	RGB	0,7652	0,8490	**0,8284**	0,7982	0,6293
	CIE-XYZ	0,7128	0,8198	0,8185	0,7779	0,6126
	YCrCb	**0,8474**	**0,8936**	**0,8258**	**0,8611**	**0,7816**
	Gabor	0,4540	-	-	-	-
	GLCM	0,7089	-	-	-	-
	LBP	0,3825	-	-	-	-
	GMRF	0,2304	-	-	-	-
DB2	HSL	**0,8025**	0,8449	0,07844	**0,8453**	0,7279
	HSV	0,7599	**0,8549**	**0,8025**	**0,8588**	**0,7948**
	CIE-Lab	**0,7937**	0,8347	0,7683	**0,8264**	**0,7853**
	CIE-Luv	0,7073	0,8242	0,7689	0,7980	0,7015
	RGB	0,6273	**0,8495**	**0,8285**	0,8051	0,7798
	CIE-XYZ	**0,7842**	**0,8457**	**0,8299**	0,8108	**0,7883**
	YCrCb	0,7587	0,8296	0,7783	0,7839	0,7359
	Gabor	0,7080	-	-	-	-
	GLCM	0,7380	-	-	-	-
	LBP	0,5511	-	-	-	-
	GMRF	0,3065	-	-	-	-

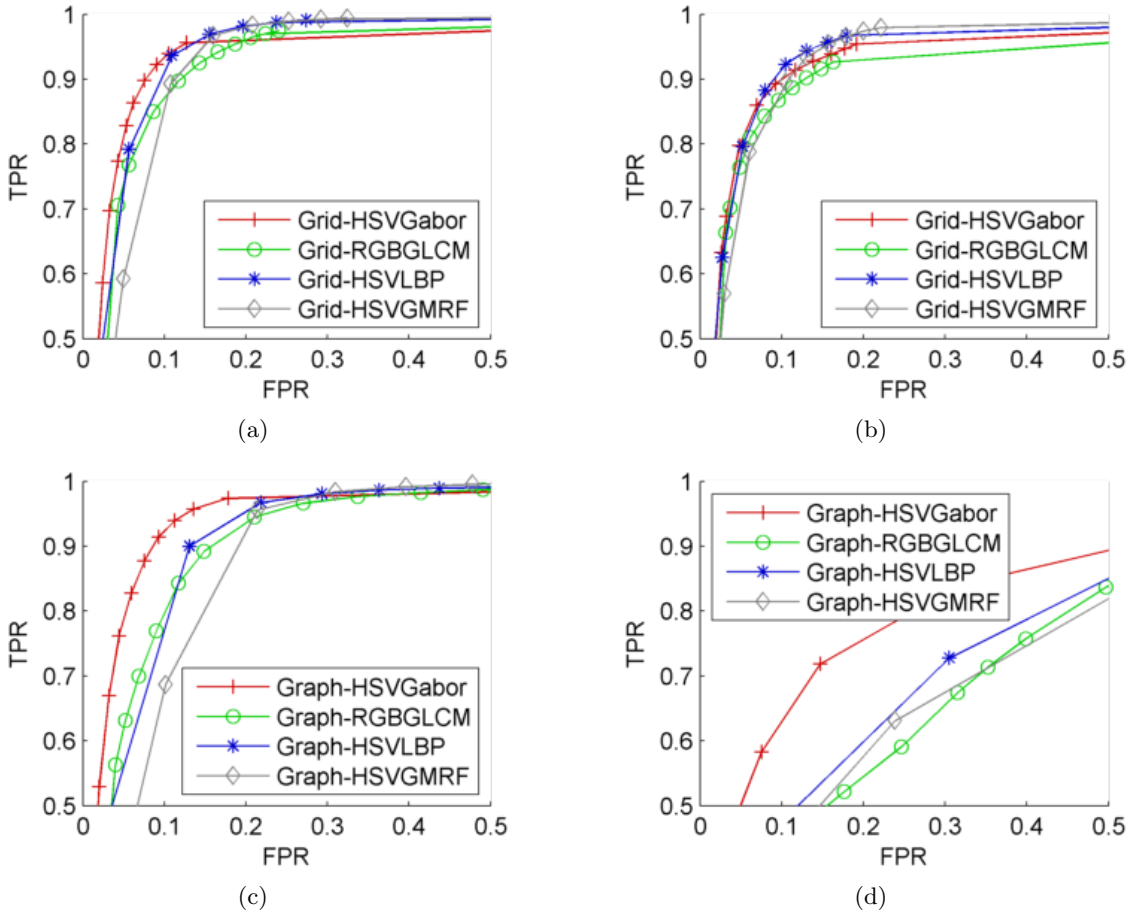


Figura 6.8: Detalle de las curvas ROC de los métodos basados en regiones para cada base de datos. (a) y (b) muestran los resultados del método de Ground-Grid en DB1 y DB2 respectivamente. (c) y (d) muestran los resultados del método basado en superpíxeles en DB1 y DB2 respectivamente.

En resumen, el uso de las características conjuntas de color y textura permite que el proceso de segmentación logre resultados competitivos en las diferentes bases de datos utilizando el mismo vector de características, independientemente de las características particulares de las bases de datos (alta similitud del camino y sus alrededores en la DB1 y cambios de iluminación y sombras en el caso de la DB2), ya que existe una combinación óptima de características conjuntas de color y textura para cada método de textura. Esto es debido a la invariancia a la iluminación propia de las características de textura y que las características de color permiten al sistema mejorar los resultados de la segmentación. Esto es observable principalmente en la DB2, ya que presentan más sombras en el camino que la DB1.

En los experimentos realizados, las mejores características conjuntas de color y textura son HSV+Gabor, seguido de HSV+LBP (segundo mejor), RGB+GLCM (tercero mejor) y HSV+GMRF (cuarto mejor). El uso de estas cuatro combinaciones de colores y texturas será analizado en la siguiente sección.

Tabla 6.2: Resultados experimentales de los métodos basados en regiones para ambas bases de datos. Los valores en negrita indican los 3 mejores resultados para cada método en términos del índice TPR10.

	Approach	HSV+Gabor	RGB+GLCM	HSV+LBP	HSV+GMRF
DB1	Pixel-wise	0,9019	0,8284	0,8733	0,8355
	Ground Grid	**0,9341**	**0,8714**	**0,9122**	**0,8545**
	Superpíxel	0,9237	0,7969	0,7720	0,6796
DB2	Pixel-wise	0,8549	0,8285	0,8588	0,7948
	Ground Grid	**0,8920**	**0,8725**	**0,9157**	**0,8825**
	Superpíxel	0,6303	0,3956	0,4763	0,4340

Segmentación Basada en Regiones

Se realizan experimentos para evaluar la conveniencia de utilizar la segmentación basada en regiones de las imágenes de caminos. Los resultados de los dos métodos de segmentación basada en regiones descritos en la Sección 6.1.1, Ground-Grid y superpíxeles, se compararon con la segmentación de píxeles estándar. Los experimentos se llevaron a cabo en la DB1 y la DB2 usando las mejores características conjuntas de color y textura obtenidas en la sección anterior (HSV+Gabor, HSV+LBP, RGB+GLCM y HSV+GMRF).

Los resultados obtenidos se presentan en la Figura 6.8 (curvas ROC) y en la Tabla 6.2 (Índice de rendimiento TPR10). Como se puede ver, el uso del método Ground-Grid mejoró los resultados de todos los algoritmos en ambas bases de datos. En la mayoría de los casos, excepto en HSV+GMRF, las tasas de TPR10 aumentan en 0,04 aproximadamente. En el caso de HSV+GMRF el incremento depende de la base de datos. Esta mejora se debe a que las regiones más próximas al vehículo están representadas por segmentos grandes, ayudando a manejar pequeñas regiones que están mal clasificadas en la segmentación basada en píxeles. Otra razón es el hecho de que cualquier píxel sobre el horizonte se descarta, ya que no pertenece a la grilla del piso. Por otro lado, el uso del método de superpíxeles no mejora los resultados del proceso de segmentación, excepto en el caso de utilizar HSV+Gabor en la DB1. Para ambas bases de datos, el método de superpíxeles falla en diferenciar correctamente los bordes de los caminos, especialmente en las regiones de la que están lejos del vehículo, ya que el borde de las regiones no está bien definido y el algoritmo de segmentación es incapaz de diferenciarlos.

En resumen, la segmentación basada en regiones de las imágenes utilizando el método de Ground-Grid mejora el rendimiento de todos los algoritmos de segmentación, independientemente de qué características conjuntas de color y textura se utiliza. Esto demuestra que el uso de la información del contexto espacial local, capturado en las regiones del suelo, ayuda al proceso de segmentación de caminos.

Tabla 6.3: Tiempo medio de ejecución de las diferentes características y métodos basados en regiones, medidos en milisegundos. Tiempo de procesamiento medido al procesar imágenes de entrada de 320x240.

[ms]	Approach	HSV+Gabor	RGB+GLCM	HSV+LBP	HSV+GMRF
	Pixel-wise	185,5	133,8	**33,1**	67,4
	Ground Grid	188,0	136,3	**35,6**	69,9
	Superpíxel	252,6	200,9	**100,2**	134,5

Tiempo de Ejecución

El tiempo de ejecución de los algoritmos es considerado dentro de la evaluación, ya que la ejecución en tiempo real es necesaria para una conducción segura y autónoma. La Tabla 6.3 muestra el tiempo de ejecución promedio de las mejores características conjuntas de color y textura, y métodos basados en regiones analizados en la sección 6.1.3. La implementación de los algoritmos se realizó en C++, ejecutándose en un computador con Intel Core I5 de 3.1Ghz y las imágenes de entrada se redimensionaron a 320x240.

En la Tabla 6.3 se muestra que HSV+LBP es la combinación más rápida seguida por HSV+GMRF y RGB+GLCM, dejando a HSV+Gabor como el algoritmo con mayor consumo de tiempo de procesamiento. Para los métodos basados en regiones, el método Ground-Grid añade una pequeña cantidad de procesamiento a la segmentación, haciéndolo adecuado para el funcionamiento en tiempo real, mientras que el método de superpíxeles requiere una cantidad importante de tiempo de procesamiento. Las alternativas que permiten que el sistema funcione a más de 10 fotogramas por segundo son HSV+LBP y HSV+GMRF en la segmentación basada en píxeles y el método de Ground-Grid, cuando las imágenes son redimensionadas a 320x240 antes del procesamiento. El algoritmo HSV+LBP+Ground-Grid es la mejor alternativa para la segmentación de caminos no pavimentados al considerar el tiempo de ejecución y el rendimiento.

6.2. Mapeo Semántico de Caminos no Pavimentados

El objetivo es construir TSM a partir de las imágenes tomadas por una cámara frontal montada en el vehículo (ver Figura 6.4) y la información de alto nivel obtenida de una descripción semántica de la imagen (ver Figura 6.9). Esta metodología tiene dos etapas: Descripción Semántica y Mapeo Semántico Topológico.

En la etapa de Descripción Semántica, que es equivalente a la Percepción Semántica de la sección 4.1, cada imagen es procesada para obtener una descripción semántica de la escena, incluyendo la forma del camino, vegetación, suelos alrededor del camino, obstáculos y objetos de interés. En la etapa de Mapeo Semántico Topológico, la descripción semántica de la imagen es utilizada para

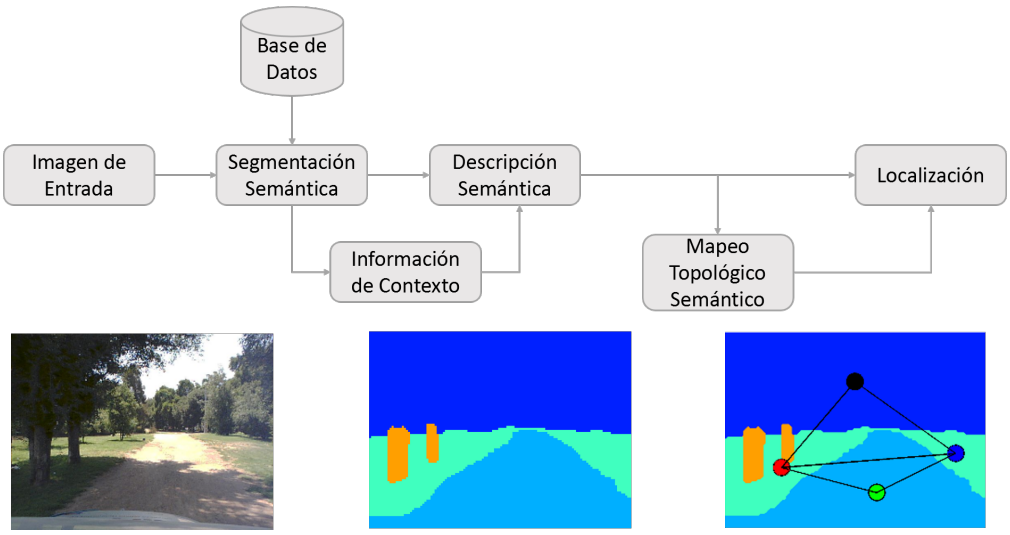


Figura 6.9: Diagrama de bloques de la metodología propuesta

generar un TSM.

6.2.1. Descripción Semántica

Esta etapa crea una representación semántica del contenido de la imagen utilizando una estructura de grafo basada en la segmentación semántica de cada imagen, así como información de contexto espacial. La Figura 6.10 muestra un ejemplo de la descripción semántica de tres diferentes imágenes, y la estructura de grafo resultante para cada una. Para ilustrar de mejor manera, se utilizan diagramas en vez de imágenes reales.

Cada imagen es segmentada semánticamente utilizando el método de Textoboost, propuesto por Shotton et al. [85], un método de segmentación multi-clase que incorpora forma, textura, color, ubicación y bordes en un único modelo de Conditional Random Fields (CRF). Para este modelo, la probabilidad condicional de la etiqueta de la clase c dada una imagen x se define como:

$$\begin{aligned} logP\left(c \mid x, \theta\right) = \sum_{i} \psi_i \left(c_i, x; \theta_\psi\right) + \pi \left(c_i, x_i; \theta_\pi\right) \\ +\lambda \left(c_i, i; \theta_\lambda\right) + \sum_{(i,j)\in\xi} \phi \left(c_i, c_j, g_{ij}\left(x\right); \theta_p hi\right) \\ -logZ\left(\theta, x\right) \end{aligned} \tag{6.24}$$

donde ξ es el conjunto de los bordes conectados de manera horizontal o vertical en la grilla, $Z\left(\theta, x\right)$ es una función de partición, $\theta = \{\theta_\psi, \theta_\pi, \theta_\lambda, \theta_\phi\}$ son los parámetros del modelo e i, j son coordenadas de la en la imagen.

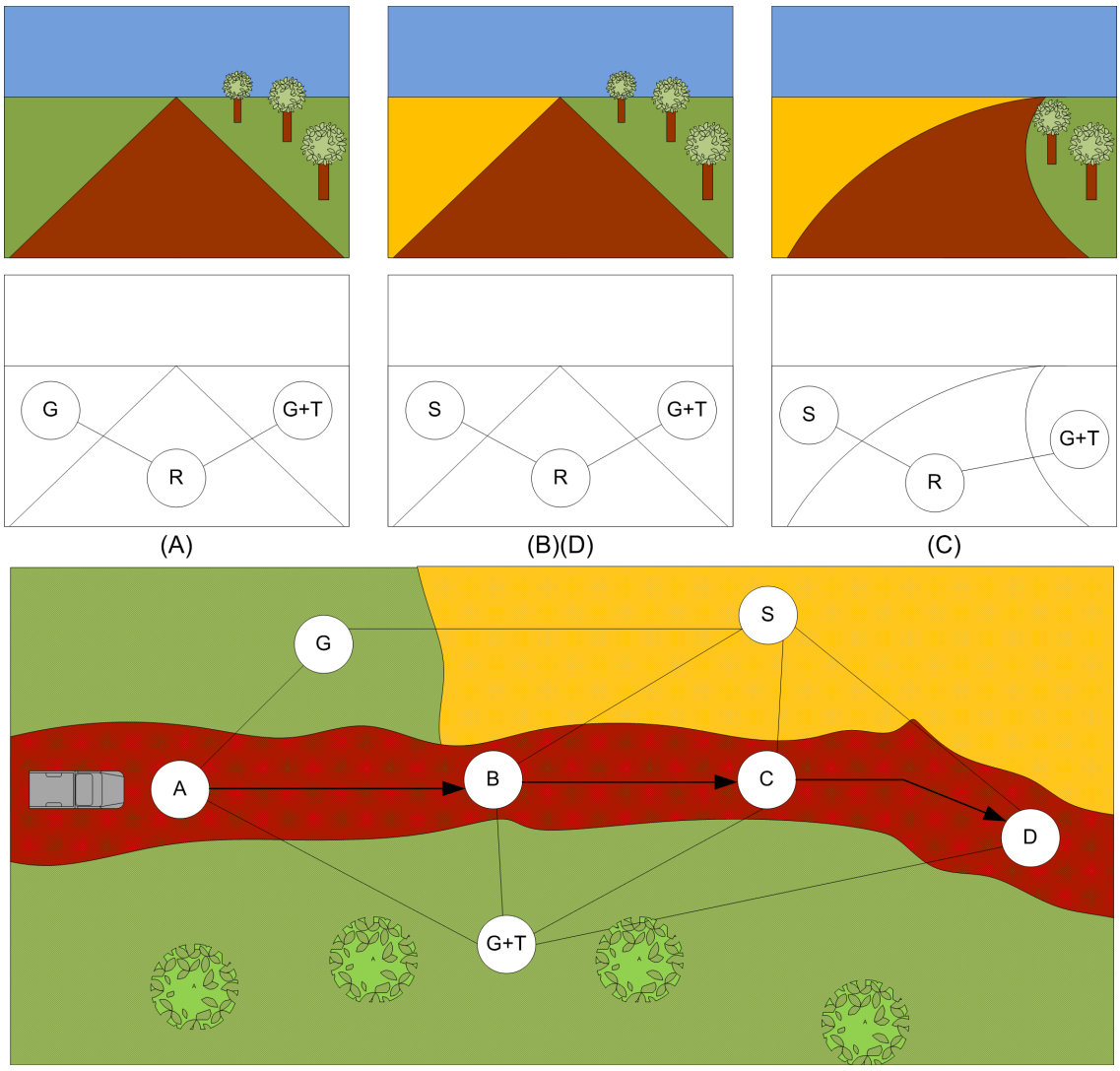


Figura 6.10: Ejemplo de la estructura del mapa topológico. La primera fila muestra tres posibles imágenes de entrada para el sistema. La segunda fila muestra la descripción semántica para cada una de las imágenes anteriores, y la tercera fila muestra el mapa topológico resultante. En esta figura G significa pasto y S es arena. El nodo G+T es de tipo pasto y contiene árboles. A, B, C y D son nodos de camino, donde B y D comparten la misma descripción semántica, pero representan distintas zonas del camino.

El termino ψ en la ecuación representa los potenciales de forma-textura, están basados en la combinación de características tipo texton [47]. El termino π representa el potencial de color y esta basado en un modelo de mezclas de Gaussianas (GMM). El término λ representa el potencial de ubicación en la imagen y funciona como una *look-up table*. Finalmente, el termino ϕ representa el potencial de borde, y está basado en el modelo de Potts [77], sensible al contraste.

Dado que el sistema está diseñado para funcionar en caminos no pavimentados, las siguientes etiquetas fueron consideradas en el entrenamiento: cielo, árbol, tierra, pasto, arbustos, obstáculos y agua.

Luego, cada segmento de tipo suelo (tierra, arbustos, agua y pasto) es representado por un nodo en el grafo de la descripción semántica, y las aristas representan la vecindad entre estos segmentos. Cada nodo contiene la siguiente información: (i) tipo, (ii) ubicación espacial, (iii) lista de objetos y (iv) manejabilidad, además los nodos del camino incluyen (v) curvatura y (vi) largo.

(I) Tipo: La etiqueta adquirida por la segmentación semántica.

(II) Ubicación Espacial: relación espacial con el camino. El primer segmento de tierra frente al vehículo se denomina "Camino", y el resto se identifica como "derecha" o "izquierda".

(III) Lista de Objetos: lista de segmentos de obstáculos (árbol y obstáculos) observados sobre cada uno de los segmentos de suelo.

(IV) Manejabilidad: representa qué tan seguro es para el vehículo para ser manejado por esa región y se calcula a partir de la cantidad de objetos presentes en la lista de objetos [8].

(V) Curvatura: una clasificación gruesa de la curvatura del camino, y puede tomar 5 valores: 'Cerrada Izquierda', 'Izquierda', 'Recta', 'Derecha' y 'Cerrada Derecha'. Esta curvatura se obtiene utilizando un modelo de camino con curvatura fija proyectado sobre un mundo plano a través de una transformada de Hugh.

(VI) Largo: Es la suma de la odometría del vehículo. Esto será utilizado después para la construcción del mapa.

Ademas, cada nodo incluye referencias a los principales nodos de Derecha e Izquierda del camino, donde por principales nos referimos a los segmentos más grandes de cada lado.

Definimos SD_k como la descripción semántica para la k-ésima imagen, y SD_k.Road como el nodo del camino de la misma imagen. SD_k.Left y SD_k.Right son la listas de nodos a la izquierda y derecha del camino, respectivamente.

6.2.2. Mapeo Topológico Semántico

Para esta metodología, un TSM es una estructura de grafo que describe el camino y su entorno, basado en la descripción semántica de las imágenes, descrita en la sección anterior. El TSM se organiza como una secuencia de nodos de camino con dos secuencias de nodos 'ambientales', que representan las regiones a la izquierda y derecha del camino. Un nodo de camino representa una región en la que el camino está definido por una descripción semántica. Cada nodo ambiental representa la principal región al costado del camino, para uno o mas nodos de camino.

Cada nodo de camino en el TSM representa una sección del camino donde las descripciones semánticas obtenidas de imágenes consecutivas son iguales en términos del nodo derecho, izquierdo y la curvatura del camino.

Los nodos ambientales se ubican correspondientemente en el TSM utilizando la etiqueta espacial (derecha o izquierda) y están conectados a sus nodos de camino vecinos, creando una descripción consistente de la vecindad del camino dentro del mapa topológico.

El TSM se construye de manera incremental a partir de las descripciones semánticas obtenidas de las imágenes del vehículo. Sea M_k el TSM en tiempo k. M_k resume la información de las descripciones semánticas pasadas SD_i, con $i = 1, \ldots, k$. Los nodos de M_k tienen la misma estructura básica que los nodos de las descripciones semánticas, pero cada nodo de camino en el TSM incluye el largo como la suma de los largos de las descripciones semánticas que construyeron dicho nodo.

Algoritmo 6.1 Criterio de actualización del mapa

```
1: function MappingUpdate(SD_k, M_{k-1})
2:     new_road_node ← false
3:     M_k ← M_{k-1}
4:     M_k.Road.last.length ← SD_k.Road.odometry − first_odom
5:     if SD_k.Road.curv ≠ M_{k-1}.Road.curv then
6:         new_road_node ← true
7:     end if
8:     if SD_k.Left.main ≠ M_{k-1}.Left.last then
9:         M_k.add_left(SD_k.Left.main)
10:        new_road_node ← true
11:    end if
12:    if SD_k.Right.main ≠ M_{k-1}.Right.last then
13:        M_k.add_right(SD_k.Right.main)
14:        new_road_node ← true
15:    end if
16:    if new_road_node then
17:        M_k.add_node(SD_k.Road)
18:        first_odom ← SD_k.Road.odometry
19:    end if
20: end function
```

El criterio de actualización del TSM se detalla en el Algoritmo 6.1. Cada vez que un nuevo nodo de mapa es agregado alguna de las tres listas de nodos (Camino, Izquierda o Derecha), la referencia al último elemento de cada lista se actualiza. Las funciones *add_left* y *add_right* agregan un nuevo nodo ambiental al TSM al final de la lista correspondiente y actualizando la referencia correspondiente. La función *add_node* agrega un nuevo nodo de camino al TSM, definiendo sus referencias a la izquierda y derecha como los últimos nodos ambientales en dichas listas.

Por cada nueva descripción semántica SD_k, la función *MappingUpdate* actualiza el largo del nodo actual y evalúa si la curvatura del camino o los nodos principales de derecha o izquierda son los mismos al último nodo de camino agregado al TSM. Si el nodo principal de la izquierda es distinto del último agregado al TSM, entonces se crea un nuevo nodo a la izquierda con esta información. Lo mismo ocurre con los nodos de la derecha del camino. Si un nodo ambiental es agregado, o la curvatura del camino es distinta, entonces se agrega un nuevo nodo de camino, utilizando la información de la descripción semántica.

La Figura 6.11 muestra un ejemplo del mapa topológico construido a partir de imágenes de camino. Las descripciones semánticas para cada una de las imágenes de ejemplo se muestran sobre el resultado de la segmentación semántica.

6.2.3. Resultado Mapeo Semántico de Caminos no Pavimentados

La metodología propuesta para construcción de mapas topológicos semánticos descrita en la sección 6.2 es evaluada utilizando la base de datos tomada en el Parque O'Higgins por Parra-Tsunekawa et al. [73], que corresponde a un recorrido de alrededor de 800[m] sobre un camino de tierra irregular con pendientes positivas y negativas. El camino es principalmente de tierra y está rodeado de pasto, arbustos y árboles, además de rondar una laguna. La manejabilidad de un nodo puede tener 3 valores: bajo, medio y alto. Tierra y pasto tienen manejabilidad alta, mientras que arbustos tiene manejabilidad media y agua tiene manejabilidad baja. Cualquier obstáculo detectado cerca del borde del camino, es decir a menos que el ancho del camino, se le asigna una manejabilidad baja. Si los obstáculos se encuentran a una distancia entre uno y dos anchos del camino, entonces se les asigna una manejabilidad media, y si se encuentran a más que esto, o no presenta obstáculos, entonces se le asigna una manejabilidad alta.

Un primer experimento constó en probar la metodología en un segmento reducido de la base de datos y utilizando imágenes segmentadas de manera manual. El objetivo fue evaluar si la información semántica de las imágenes permitía construir un mapa topológico que describiera el entorno de manera coherente. Se utilizó el segmento que parte en las coordenadas (-33.468614,-70.662464), y tiene un largo de 160[m]. Los resultados de esta prueba se muestran en la Figura 6.11. Aquí se

muestran cuatro imágenes de la base de datos en la primera fila, con las 4 segmentaciones semánticas manualmente anotadas en la segunda fila, y en la tercera fila se muestra el mapa topológico construido sobre una imagen satelital. Cada nodo esta etiquetado con una letra y un color, donde 'R' representa un nodo de camino, 'G' para un nodo de tipo pasto y 'B' para uno de tipo arbustos. El color representa la manejabilidad de cada nodo: verde para manejabilidad alta, naranjo para media y rojo para baja.

El mapa resultante es consistente con el entorno del camino para el segmento construido, lo que muestra que la metodología es capaz de conservar la información semántica más relevante del entorno de manera consistente. El mapa resultante contiene 12 nodos de camino y 22 nodos de entorno, lo que se traduce en que la metodología es capaz de resumir la información de alto nivel de 160[m] de camino en tan sólo 22 nodos, lo que demuestra que la metodología propuesta puede ser una solución eficiente para el mapeo en ambientes urbanos.

El segundo experimento consistió en utilizar la base de datos completa con el método de segmentación semántica para la generación de un mapa topológico que represente los 800[m] que cubre la base de datos y sus resultados se muestran la Figura 6.12. Pese a que la segmentación semántica es menos precisa que el etiquetado manual utilizado anteriormente, el resultado sigue siento coherente con el entorno del camino para toda la región. En este caso se logró representar los 823[m] de camino con 24 nodos de camino y 43 nodos ambientales, lo que corrobora esta metodología como una herramienta para el mapeo topológico semántico de caminos.

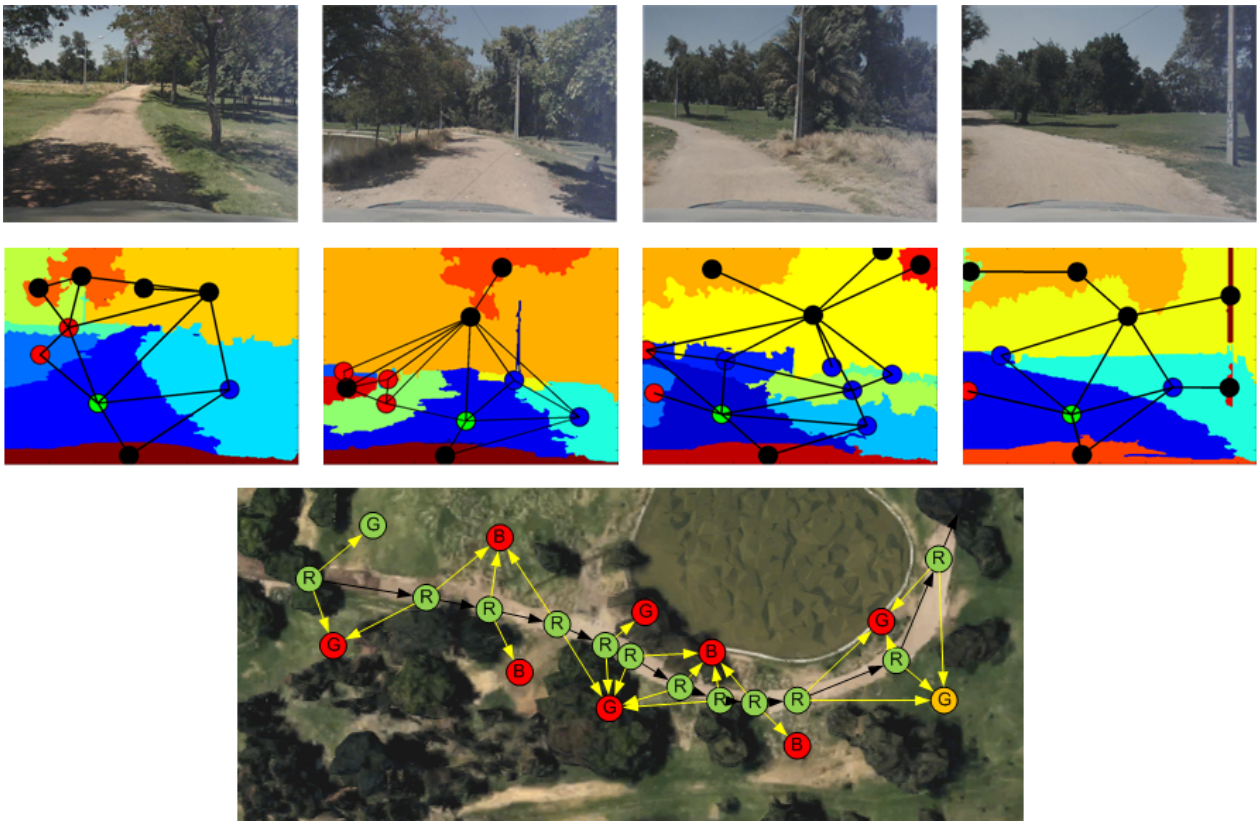


Figura 6.11: Mapa topológico semántico resultante de la primera prueba. En la primera fila se muestran cuatro imágenes de la base de datos, con las 4 segmentaciones semánticas manualmente anotadas en la segunda fila, y en la tercera fila se muestra el mapa topológico construido sobre una imagen satelital. (Fuente: Google Maps, satellite image).

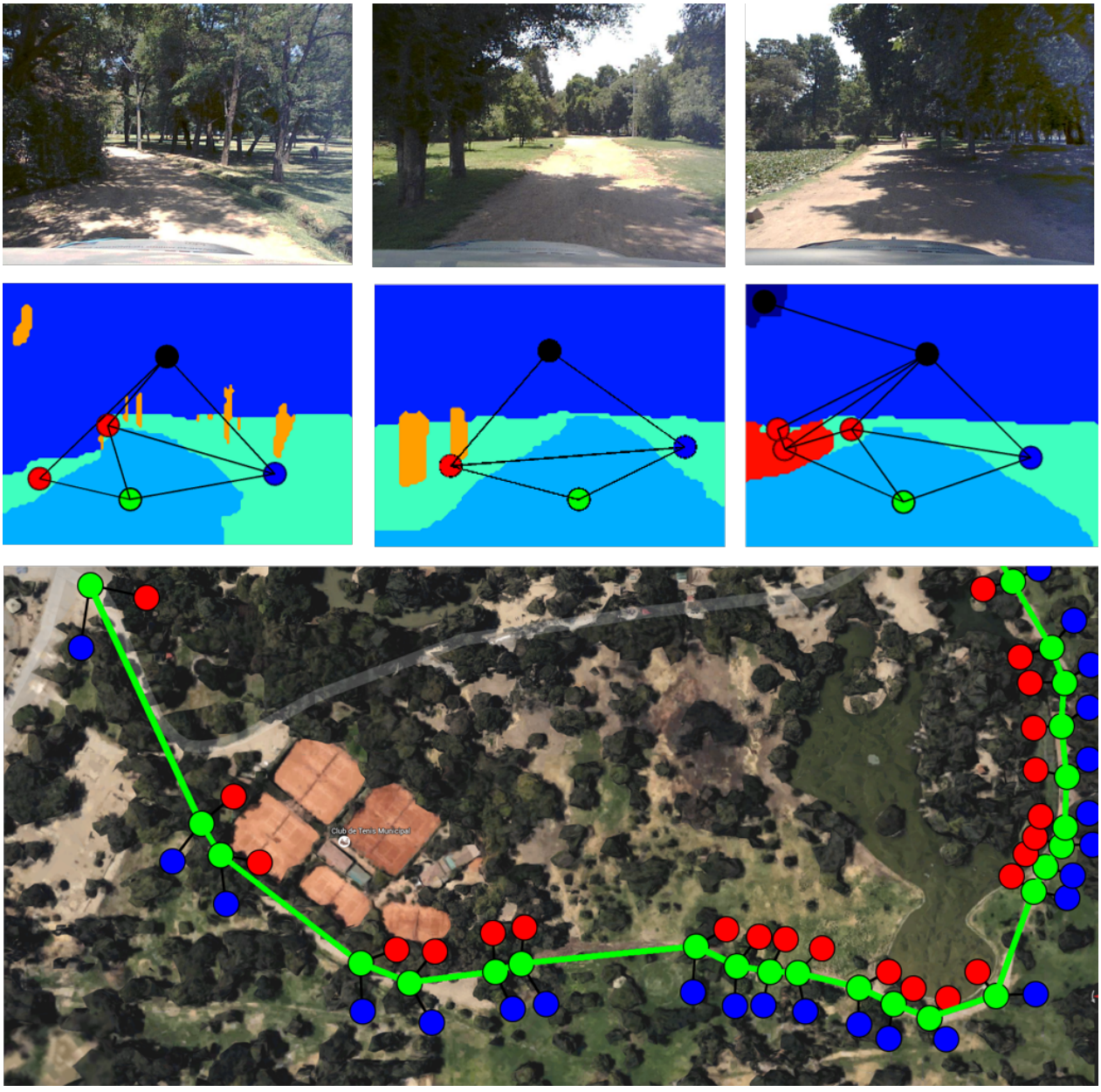


Figura 6.12: Mapa topológico semántico resultante de la segunda prueba. En la primera fila se muestran tres imágenes de la base de datos, con las 3 segmentaciones semánticas obtenidas del método de segmentación semántica en la segunda fila, y en la tercera fila se muestra el mapa topológico construido sobre una imagen satelital. (Fuente: Google Maps, satellite image).

6.3. Localización en Caminos no Pavimentados

Al utilizar la estructura de mapa topológico semántico, el problema de localización se reduce a encontrar el nodo de camino en el mapa que corresponde a las observaciones obtenidas del vehículo. Dado esto, la posición del vehículo se define como uno de los nodos de camino del mapa. El método de localización se muestra en el Algoritmo 6.2.

Algoritmo 6.2 Método de localización para caminos no pavimentados.

```
1: function LOCALIZATION(SD_k, globalTSM_k, localTSM_k)
2:     localTSM_k.MappingUpdate(SD_k)
3:     if is_lost then
4:         for i=1...length(hypothesisList) do
5:             if D(localTSM, globalTSM, hypothesisList[i]) ≥ T then
6:                 hypothesisList.remove(i)
7:             end if
8:             new_road_node ← true
9:         end for
10:        if length(hypothesisList) = 1 then
11:            is_lost ← false
12:            pose_estimation ← hypothesisList(1)
13:        end if
14:        if length(hypothesisList) = 0 then
15:            localTSM.reset()
16:            hypothesisList ← globalTSM.Road
17:        end if
18:    else
19:        if D(localTSM, globalTSM, hypothesisList[i]) ≥ T then
20:            localTSM.reset()
21:            hypothesisList ← globalTSM.Road
22:            is_lost ← true
23:        else
24:            pose_estimation ← localTSM.Road.last
25:        end if
26:    end if
27: end function
```

Con respecto a la localización, el vehículo puede estar en dos estados: (i) localizado o (ii) perdido. El primer estado se cumple cuando existe una pose estimada en el mapa que es consistente con las observaciones obtenidas, mientras que el segundo estado se cumple cuando la pose estimada no es consistente con las observaciones, o cuando no existe una pose estimada (cuando no hay una pose inicial, por ejemplo).

Para resolver el problema de localización se utiliza una lista de hipótesis de posiciones en el mapa y dos TSM: el TSM global, un mapa pre-construido del entorno, y un TSM local, construido con las últimas observaciones de acuerdo al método de mapeo descrito anteriormente.

Cuando la localización pasa a estar en estado perdido, el TSM local se vacía y se le agrega la última observación,y la lista de hipótesis se llena con todos los nodos de camino del mapa. En este estado se compara el TSM local con cada candidato de la lista de hipótesis usando la distancia definida en la ecuación 6.25, para cada nueva observación desde el vehículo. Si la distancia para un candidato es mayor que un cierto umbral T, este candidato es removido de la lista de hipótesis. Si después de una observación se eliminan todas las hipótesis, entonces se reinicia la lista de hipótesis con todos los nodos de camino del mapa, y si queda un único elemento en la lista, entonces se cambia a estado localizado.

En estado localizado, solo existe una hipótesis, que se asume correcta, y se actualiza de acuerdo a las observaciones. Para cada observación se compara la estimación de la pose con la distancia de la ecuación 6.25. Si se cumple la condición para eliminar la última hipótesis, entonces el vehículo se considera perdido.

$$D\left(M_a, M_b, k\right) = k_p \cdot P + k_q \cdot \sum_i Q_i. \tag{6.25}$$

La función de distancia de la ecuación 6.25 compara un TSM global M_a con un TSM local M_b en un nodo k como una suma ponderada de dos términos: P y Q. El término $P = max(0, M_b.Road.last.length - M_a.Road[k].length)$ es la diferencia entre los largos del nodo actual en ambos mapas, y los términos $Q_i \in [0, 1]$ se evalúan sobre cada nodo de camino del TSM local y miden el porcentaje de la información semántica del nodo de camino es igual a la información semántica del nodo de camino equivalente en el mapa global según k. Por ejemplo, si al comparar un nodo de camino del mapa local con un nodo del mapa global existe un término diferente de entre los 6 términos definidos en la sección 6.2.1, entonces $Q_i = 16{,}7\,\%$.

6.3.1. Resultados de Localización en Caminos no Pavimentados

Se valida el uso de la metodología de localización de vehículos autónomos en TSM para caminos no pavimentados utilizando la base de datos tomada en el Parque O'Higgins. Se efectúan dos experimentos que buscan evaluar el comportamiento del método de localización bajo dos escenarios: con y sin pose inicial conocida dentro del mapa. El primer caso, con pose inicial conocida, evalúa la capacidad del método de localización de seguir la pose del vehículo de manera precisa, mientras que el segundo experimento, sin pose inicial conocida, busca evaluar la robustez del sistema ante posibles fallas en la localización.

La base de datos utilizada está compuesta por un único recorrido de aproximadamente 800[m], por lo que para poder efectuar los experimentos de localización fue necesario dividir las imágenes

Tabla 6.4: Resultado del Experimento 1 de acuerdo a la Razón de Verdaderos Estimados (TER), la Razón de Verdaderos Estimados según Distancia (D-TER), la TER relajada (rTER), la D-TER relajada (rD-TER), el Error Medio (EM) y el EM de los Falsos Estimados, para cada método de localización.

Métrica	Resultado
Razón de Verdaderos Estimados (TER)	**0,9640**
TER según Distancia (D-TER)	**0,9619**
TER relajada (rTER)	**0,9960**
D-TER relajada (rD-TER)	**0,9958**
Error Medio (EM) [m]	**2,3**
EM de los Falsos Estimados [m]	**23,7**

en dos grupos: uno para la construcción del mapa y otro para la evaluación de la localización. Para construir estos grupos se tomó una imagen cada medio segundo de la base de datos y a partir de este subconjunto de imágenes se seleccionaron las imágenes impares para construir el mapa y las imágenes pares para evaluar la localización.

El mapa construido con el primer grupo está compuesto por 22 nodos de camino y 36 nodos ambientales y es utilizado como base para los experimentos siguientes.

Experimento 1: se evalúa el comportamiento del método de localización cuando la pose inicial es conocida. Para esto se utiliza el método de localización sobre el segundo grupo de la base de datos partiendo desde el inicio del recorrido. Para cada imagen se estima la pose del vehículo con el *ground truth* dentro del TSM construido. Las métricas utilizadas son las mismas que las presentadas en la sección 5.3.

Los resultados obtenidos para este experimento, resumidos en la Tabla 6.4, muestran que el sistema logra seguir exitosamente la pose del vehículo durante el recorrido, ya que estima con precisión la pose del vehículo en el mapa en más del 96 % del recorrido. Destaca que los valores obtenidos para TER y D-TER son prácticamente iguales. Esto se explica al considerar que en esta base de datos el vehículo viaja a una velocidad constante en la mayor parte del recorrido, lo que hace que estas métricas sean equivalentes.

Experimento 2: se evalúa la convergencia del método de localización cuando la posición inicial del vehículo es desconocida. El método de localización se inicia con todos los nodos de camino dentro de la lista de hipótesis y la pose del vehículo es elegida al azar dentro del 75 % inicial del grupo de imágenes utilizadas para localización. Luego se calcula el tiempo de convergencia del método junto con la tasa de éxito para esta convergencia, de la misma manera que en el experimento 2 de la sección 5.3.

Tabla 6.5: Resultados del Experimento 2, con posición inicial desconocida.

	Tiempo Promedio de Localización [s]	Tasa de Éxito[%]
Convergencia de Localización	21,6	**100 %**

Los resultados de este experimento están resumidos en la Tabla 6.5 y muestran que la metodología logra resolver satisfactoriamente el problema la localización sin posición inicial en todas las ejecuciones realizadas. Por otro lado, el tiempo promedio de 21,6[s] puede parecer alto, pero se explica porque existen varios nodos en el mapa que comparten la misma descripción semántica, por lo que es necesario más imágenes para poder descartar las hipótesis incorrectas.

Los resultados obtenidos muestran que la metodología resuelve exitosamente el problema de la localización en TSM para caminos no pavimentados en la base de datos utilizada, tanto en la precisión obtenida en el experimento 1, como en el tiempo de convergencia y la tasa de éxito del experimento 2. Sin embargo, los resultados pueden no ser representativos del funcionamiento en condiciones reales, ya que los datos utilizados para construir el mapa y probar la localización corresponden al mismo recorrido, aunque sean imágenes distintas. Además, la base de datos es muy simple como para evaluar la localización, ya que el recorrido es lineal y sin intersecciones. Es por esto que es necesario utilizar una base de datos más compleja para validar el funcionamiento de esta metodología, y para hacerlo se diseñó e implementó la metodología presentada en los capítulos 4 y 5, cuyos resultados confirman la validez de la metodología presentada.

Capítulo 7

Conclusiones

El trabajo presentado en esta tesis logró modelar el entorno de un vehículo terrestre a través de un mapa topológico semántico para la localización de un vehículo autónomo. El sistema hace uso de la información obtenida desde un sistema de visión computacional, que identifica distintos elementos de interés para la conducción autónoma, basado en un método de segmentación semántica de Deep Learning que incluye explícitamente contexto espacial. El mapa generado divide exitosamente el mundo en segmentos semánticamente similares que al ser utilizado en conjunto con un método de localización permite estimar de manera robusta y eficiente la posición del vehículo, siendo una herramienta útil para la navegación autónoma de vehículos.

El trabajo realizado incluyó el diseño y la construcción bases de datos que permitieron evaluar el rendimiento de los métodos de visión computacional y localización; el diseño y la implementación de métodos de segmentación de caminos basados en características de color y texturas, a nivel de píxel y regiones; la implementación de métodos de segmentación semántica que permitieron identificar los distintos elementos de interés para la conducción autónoma tanto en ambientes rurales como en ambientes urbanos; la generación de una descripción semántica que resume la información semántica de las imágenes e información de contexto espacial que permite la construcción del mapa topológico semántico y la localización; el diseño e implementación de un método de construcción de mapas topológicos semánticos del entorno basado en la descripción semántica; y la implementación de un método de localización robusto y eficiente que utiliza la descripción semántica de las imágenes del vehículo como observación para encontrar la ubicación del vehículo dentro del mapa topológico semántico del lugar por el cual se está navegando.

En el estudio comparativo de métodos de segmentación de caminos no pavimentados basados en visión se analizaron y compararon dentro de un esquema bayesiano adaptativo el rendimiento de 2 métodos de segmentación basado en regiones (Ground-Grid y superpíxeles), 7 características de color diferentes (RGB, HSV, HSL, CIE-XYZ, YCrCb, CIE-Lab y CIE-Luv), 4 características

de texturas (Gabor, GLCM, LBP y GMRF) y 28 combinaciones de características conjuntas de color y textura, para proporcionar guías para el desarrollo de sistemas autónomos de detección de caminos no pavimentados. Los resultados obtenidos muestran que, en el caso de utilizar sólo características de color, la selección del mejor espacio de color a utilizar depende de las condiciones ambientales y que el uso de características de texturas da peores resultados que el uso de características de color. Sin embargo, el uso de características combinadas de color y textura permite que el proceso de segmentación logre resultados competitivos en diferentes bases de datos usando la misma combinación de características, ya que existe una combinación de características de color y textura óptima para cada método de textura. En los experimentos realizados, las mejores combinaciones de características de color y textura son: HSV+Gabor, seguida por HSV+LBP (segunda mejor), RGB+GLCM (tercera mejor), y HSV+GMRF (cuarta mejor). Además, el uso del método de Ground-Grid para la segmentación de las imágenes basada en regiones mejora los resultados de todos los algoritmos de segmentación, independiente de la combinación de color y textura que se utilice. Y finalmente, pese a que HSV+Gabor+Ground-Grid muestra el mejor rendimiento, es recomendable utilizar HSV+LBP+Ground-Grid, ya que alcanza un rendimiento similar usando una quinta parte del tiempo de procesamiento.

El mapa topológico construido para la conducción en caminos de tierra logra describir consistentemente el recorrido de 800[m] en tan solo 24 nodos de camino y 43 nodos ambientales, lo que corrobora esta metodología como una herramienta para la construcción de mapas topológicos basados en información semántica del entorno y demuestra que la estructura de mapa topológica es una opción viable para describir grandes regiones de caminos de manera liviana computacionalmente. Sin embrago la base de datos utilizada no permitió una evaluación adecuada de algún método de localización, por lo que se requirió extender esta metodología a ambientes urbanos para analizar adecuadamente la localización.

Para evaluar las metodologías de mapeo y localización en ambientes urbanos se construyó una base de datos de conducción en ambientes de ciudad utilizando un vehículo comercial con una cámara monocular frontal. Esta base de datos fue grabada en el centro de Santiago de Chile bajo condiciones normales de tráfico y consiste en dos secuencias grabadas en días distintos sobre un mismo conjunto de calles, pero con recorridos distintos, diseñados de forma tal que la primera secuencia sirva para construir el TSM y la segunda sirva para probar la localización.

El método de mapeo topológico semántico fue capaz de describir un total de $9,57$[km] de calles utilizando 232 segmentos con una longitud promedio de 41,25 [m], manteniendo una descripción semántica de cada segmento que permitió la auto-localización exitosa del vehículo.

Se compararon dos métodos de localización, siendo el método de filtro de partículas el que obtuvo el mejor rendimiento cuando la posición inicial del vehículo es conocida, mientras que el *Foward Algorithm* mostró la convergencia más rápida cuando la localización inicial es desconocida. Sin embargo, el filtro de partículas fue capaz de mantener una estimación precisa de la pose en el 98,67 % del recorrido realizado, y pudo auto-localizarse correctamente después de una localización inicial desconocida en el 100 % de los casos. Los resultados muestran que la localización basada en filtros de partículas para el mapa semántico topológico representa una solución robusta para ADAS y navegación de vehículos autónomos, con aplicación directa en tareas de navegación en autopistas, caminos rurales o áreas urbanas recorridas regularmente.

El uso conjunto de información topológica y semántica para la conducción en ambientes urbanos, donde las categorías de objetos presentes son limitados, permite lograr una auto-localización eficiente. Esta metodología representa una solución para la navegación autónoma con aplicación directa en autopistas, caminos rurales y rutas en ciudad recorridas periódicamente, donde el uso y la mantención de mapas métricos es difícil y costoso desde un punto de vista computacional. Esta metodología puede ser también usada como un sistema de asistencia ya que puede entregar información de localización a un conductor humano.

La metodología propuesta no necesita ninguna información métrica para su ejecución. Sin embargo, es posible agregar información métrica, como una posición GPS, a los nodos del GTSM para futuras aplicaciones.

Apéndices

A . Experimentos Adicionales

A continuación se presentan los experimentos adicionales realizados al método de localización y mapeo, que buscan aclarar el desempeño del sistema ante diversas perturbaciones inducidas artificialmente en los datos.

En esta comparación su utiliza como punto de referencia el mejor resultado obtenido en el Experimento 3 de la Sección 5.3, que corresponde a una distancia promedio de 7,6$[m]$ entre la estimación métrica de la pose del vehículo y la pose GPS asociada al mapa, obtenido con el método de localización por Filtro de Partículas. Para esta comparación se utilizara la misma configuración utilizada para obtener esos resultados.

Para facilitar la comparación de los resultados, se utilizara la distancia promedio de la estimación métrica de localización para comparar los resultados de los distintos experimentos.

A continuación se detallan los dos experimentos adicionales: Primero por perturbaciones en la odometría en la etapa de construcción de mapas y luego la inclusión de perturbaciones en la observación semántica en la etapa de localización.

A .1. Perturbación en Odometría

El objetivo de este experimento es evaluar la sensibilidad del sistema a perturbaciones en la medida de la odometría del vehículo. Es sabido que los sensores inerciales son poco confiables debido a que el error en la estimación se acumula en el tiempo, por lo que el error después de una ejecución larga es demasiado grande para efectos de localización. En el caso del trabajo propuesto, la odometría del vehículo es sometida a una gran cantidad de ruido al momento de ser incorporada al filtro de partículas, por lo que es posible aseverar que la localización es robusta a errores en la odometría. Si embargo, el proceso de construcción de mapas utiliza la odometría directamente, por lo que el error en la odometría queda plasmado en el mapa.

Para evaluar el efecto del ruido en la odometría en la etapa de construcción del mapa, se incorpora un ruido proporcional al largo de cada arista y un error aditivo a su orientación, para emular el

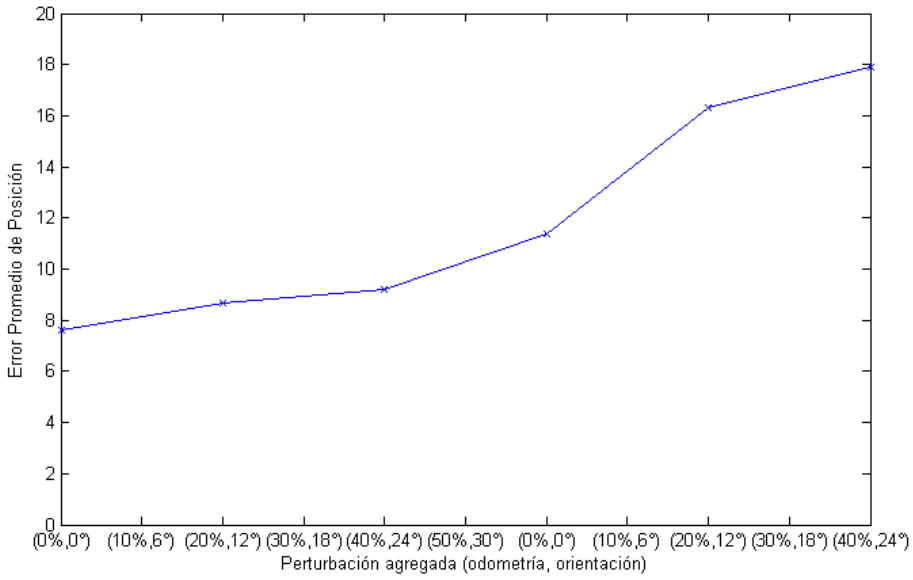


Figura 7.1: Resultados del experimento de perturbación en la odometría.

efecto en el mapa del error en la odometría. Se estudia el uso de un ruido de entre un 0 % y un 50 % en el largo, y un ruido entre 0° y 30° en la orientación.

En la figura 7.1 se observan los resultados del experimento. Se observa que los errores inducidos en el mapa afectan el rendimiento del método de localización, empeorando la distancia promedio de manera aproximadamente lineal con el error inducido.

El error inducido afecta al método de localización de dos maneras distintas: La primera manera es que el error en la estimación de la pose aumenta cuando el vehículo navega dentro de una arista larga, ya que las observaciones semánticas son similares a la descripción semántica de la arista en la cual se encuentra el vehículo y todas las partículas dentro de la misma arista mantienen un peso similar, y por tanto la estimación de la pose se desplaza de la misma manera que la odometría. Al ser distintos el largo del segmento en el mapa con el largo del segmento real, la estimación de la pose se adelanta o retrasa, aumentando el error. La segunda manera es que al variar la orientación de las aristas del mapa, estas aportan mas error en las correcciones relacionadas con las esquinas, por lo que el método se demora mas en corregir la pose después de las intersecciones en las que el vehículo dobló.

Pese a que el ruido inducido en el mapa afecta negativamente el desempeño del método, éste sigue siendo capaz de entregar una estimación precisa de la pose del vehículo y por sobre los resultados de los otros métodos implementados presentados en la tabla 5.3. Esto que demuestra que esta metodología es robusta a los errores en la odometría, tanto en la etapa de construcción de mapa, como en la etapa de localización.

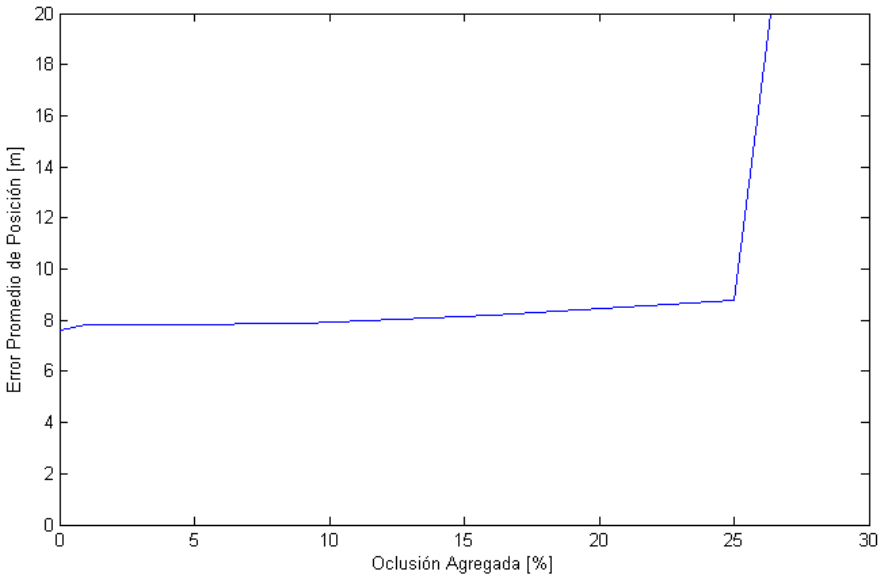


Figura 7.2: Resultados del experimento de perturbación en la odometría.

A .2. Perturbación Tipo Oclusión

El objetivo de este experimento es evaluar la sensibilidad del sistema a oclusiones en las imágenes. Uno de los principales problemas en los sistemas robóticos basados en visión computacional son las oclusiones de los elementos de interés. Si bien la metodología propuesta no esta basada en elementos de particulares del entorno, como beacons o puntos de interés, si no que en la apariencia de los segmentos de camino, las oclusiones también pueden afectar negativamente el desempeño.

Adicionalmente a los problemas de oclusión ya existentes en la base de datos, debidos principalmente al trafico vehicular y la diferencia de vehículos estacionados entre ambas secuencias, se efectúa un experimento en el cual se incluye ruido aleatorio a las observaciones semánticas para emular la presencia de oclusiones en el entorno. Este ruido es incluido en los vectores de características de las observaciones durante la etapa de localización, por lo que no afectan al mapa construido.

El ruido es incorporado como una oclusión aleatoria de un $k\,\%$ de la imagen, en cada imagen de la secuencia de localización, donde k es el parámetro utilizado en este estudio, y toma valores entre 0 y 30 %.

En la figura 7.2 se observan los resultados del experimento. Se observa que los errores inducidos en el las observaciones afectan levemente el rendimiento del método de localización, pero aumentando fuertemente por sobre el 25 %. Esto último es debido que con este nivel de ruido el método no es capaz de seguir la pose del vehículo, lo que se traduce en un importante aumento en el error.

El error inducido afecta negativamente al método de localización, ya que la oclusión incluida

aumentar la diferencia entre las observaciones y la descripción semántica del segmento en el que se encuentra el vehículo, los pesos de las partículas se distribuyen equivocadamente, favoreciendo las hipótesis erróneas.

Cuando el ruido inducido en las observaciones es moderado, éste afecta negativamente el desempeño del método pero sigue siendo capaz de entregar una estimación precisa de la pose del vehículo y por sobre los resultados de los otros métodos implementados presentados en la tabla 5.3. Esto que demuestra que esta metodología es robusta a los errores en la segmentación en la etapa de localización.

Figura 7.3: Mapa adicional utilizado para la calibración de parámetros.

B . Base de Datos Adicional

La base de datos adicional fue grabada en el centro de Santiago de Chile, utilizando una cámara montada sobre el vehículo Tiguan del proyecto del vehículo autónomo del AMTC. La base de datos está grabada con una resolución de 1280x960 y contiene dos secuencias sobre una misma trayectoria de 4[km], que rodean el Club Hípico de Santiago. Los recorridos de esta base de datos, una imagen de muestra y el resultado de la segmentación se muestran en la Figura 7.3.

C . Publicaciones Relacionadas

C .1. Presentado en CVRSUAD de ICCV-2015

Semantic Mapping of Large-Scale Outdoor Scenes for Autonomous Off-Road Driving

Fernando Bernuy[1]
[1]Dept. of Electrical Engineering
Universidad de Chile
Santiago, Chile
fbernuy@ing.uchile.cl

Javier Ruiz del Solar[1,2]
[2]Advanced Mining Technology Center
Universidad de Chile
Santiago, Chile
jruizd@ing.uchile.cl

Abstract

Semantic mapping is a very active and growing research area, with important applications in indoor and outdoor robotic applications. However, most of the research on semantic mapping has focused on indoor mapping and there is a need for developing semantic mapping methodologies for large-scale outdoor scenarios. In this work, a novel semantic mapping methodology for large-scale outdoor scenes in autonomous off-road driving applications is proposed. The semantic map representation consists of a large-scale topological map built using semantic image information. Thus, the proposed representation aims to solve the large-scale outdoors semantic mapping problem by using a graph based topological map, where relevant information for autonomous driving is added using semantic information from the image description. As a proof of concept, the proposed methodology is applied to the semantic map building of a real outdoor scenario.

1. Introduction

In robotics, a semantic map is defined as "a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes" [9]. Thus, a semantic map contains labels associated to objects (e.g., "tree", "traffic sign") and places (e.g., "road", "building"). Semantic mapping is the process of building semantic maps. Semantic perception is defined as the process of converting sensory observations to this kind of abstractions [4].

Currently, semantic mapping is a very active and growing research area. The semantic representation of the environment is considered essential in the forthcoming unmanned cars, agriculture robotics, and any robotic application requiring human-robot interaction [5]. Some of the reasons for the high interest of the community in those areas are the need for a more robust operation in unconstrained environments and more efficient task execution [12].

The autonomous generation of semantic maps aims to combine the strengths of SLAM techniques and object categorization to recover semantic-spatial knowledge about the environment, which can be used for task planning and inference about non-sensed areas [2]. Current methods are based on map building and application of real-time object categorization techniques over captured data for attaching object categories into the map, usually as a hierarchical structure, generating maps with semantic significance.

However, to the best of our knowledge, most of the research on semantic mapping has focused on indoor mapping or outdoor mapping of restricted outdoor areas. In [3], hierarchies related to spatial and conceptual information are represented using a spatial hierarchy map and a conceptual hierarchy map, in which depth represents different levels of abstraction in semantic content. Nodes between both trees are related via anchoring, which is provided through a user interface. In [6, 8, 18], a multi-layered representation based on a metric map, a navigation map, a topological map, and a conceptual map is used for representing the environment. The metric map is based on an EKF laser-based SLAM, while place classification is based on simple geometrical features extracted from laser scans, and object recognition capabilities are added through the use of SIFT descriptors [7]. Authors use an ontology-based system, and a commonsense OWL ontology of an indoor environment that describes taxonomies (is-a relations) of room types, and typical objects found therein (has-a relations). These conceptual taxonomies are handmade. In [1], Conditional Random Fields (CRF) are used for modeling the map, which is represented as a set of nodes with a hidden state that corresponds to a position plus a category. Categories are related to the kind of objects normally found in street views, mixing visual and laser information. Laser features are computed by generating local descriptors based in angles between laser

measurements, and image features are generated by composing different types of features, like local gradient directions, color information, and Haar features. In [16], a dense semantic map is built in an urban environment, using a vehicle equipped with six cameras with different orientations. Each image is segmented with an unsupervised method and labeled into one of the thirteen categories, using CRF. Then, the labeled images are projected assuming a flat world model, creating a labeled metric map, called dense semantic map. In a later work [14], stereo cameras are used to reconstruct the environment surface, replacing the flat world assumption. In a more recent work [15], an octree based 3D map is built using the stereo cameras, and a CRF method is used over the map to label it, creating a 3D occupancy semantic map. The semantic maps proposed by Sengupta et al.[14, 15, 16] archive a great description of the environment but the semantic information included is limited to the labels obtained. Also, due to its construction on a metric map the results are strongly dependent on the localization. The semantic map proposed by Douillard et al. [1] contains a similar amount of semantic knowledge, as it only contains the labels of the different objects detected, but its graph structure allows faster localization tasks.

According to Kostavelis & Gasteros [5], "a challenge for the upcoming endeavors constitutes the semantic mapping of large scale outdoors scenarios", as most of the existing methods aim to solve the problem for indoor topological mapping environments rather than outdoors, and an important aspect in indoor semantic mapping is place and object recognition, which is still underdeveloped for outdoors.

In order to address this challenge, a novel semantic mapping methodology for large-scale outdoor scenes in autonomous off-road driving applications is proposed in this work. The semantic map representation consists of a large-scale topological map built using semantic image information. Thus, the proposed representation aims to solve the large-scale outdoors semantic mapping problem by using a graph based topological map, where relevant information for autonomous driving is added using semantic information from the image description. The proposed mapping methodology is restricted to off-road driving applications where a limited number of object categories is found. Its extension to other driving applications, such as driving in highways or city driving, will depend on a proper definition of the object categories to be stored in the semantic map.

Thus, the main contribution of this work is a graph based topological semantic mapping method suitable for large scale off-road autonomous driving.

This paper is organized as follows. In Section 2, the proposed mapping methodology is described. In Section 3, experiment conducted to test the methodology is presented. Finally, in Section 4 main conclusions are drawn.

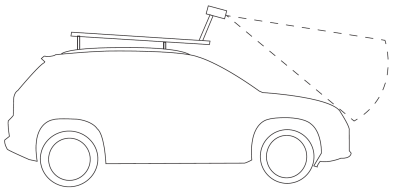


Figure 1. Camera location on the vehicle.

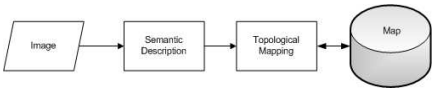


Figure 2. Block diagram of the proposed method

2. Proposed Methodology

The objective is to build a semantic map based on a consistent topological map constructed from images taken with a camera looking to the front of the vehicle (see Figure 1), and fed with high-level information obtained from an online built semantic description of the image (see Figure 2). The proposed method has two main stages: Semantic Description and Topological Semantic Mapping.

In the Semantic Description stage, each image is processed in order to obtain a semantic description of the scene, including the road shape, vegetation and soil around the road, as well as obstacles and objects of interest (e.g. trees, posts, pedestrians, etc.). In the Topological Semantic Mapping stage, the semantic description of the image is used to generate a topological map. This topological map is either added to the global topological map in case that the vehicle is driving for the first time in this area, or used for the vehicle self-localization.

2.1. Semantic Description

This stage aims to create a semantic representation of the images content using a graph structure based on the semantic segmentation of the image as well as context information. Figure 3 shows an example of the semantic description of three different images, and the resulting graph structure for each one. For demonstrative purposes, sketch versions of the images are used in this figure instead of the real images.

Each image is semantically segmented using the TextonBoost method proposed by Shotton et al. [17], a well known multi-class segmentation method that incorporates shape, texture, color, location, and edge cues in a single unified Conditional Random Fields (CRF) model. For this model, the conditional probability for class label c given the image x can be defined as

$$logP\left(c \mid x, \theta\right) = \sum_{i} \psi_i\left(c_i, x; \theta_\psi\right) + \pi\left(c_i, x_i; \theta_\pi\right)$$
$$+\lambda\left(c_i, i; \theta_\lambda\right) + \sum_{(i,j)\in\xi} \phi\left(c_i, c_j, g_{ij}\left(x\right); \theta_p hi\right) \quad (1)$$
$$-logZ\left(\theta, x\right)$$

ξ is the set of edges in the 4-connected grid, $Z\left(\theta, x\right)$ is a partition function, $\theta = \{\theta_\psi, \theta_\pi, \theta_\lambda, \theta_\phi\}$ are the model parameters, and i, j represent coordinates on the image.

The ψ term in the equation represents the shape-texture potentials, and is based on a boosted combination of texton features. The π term represents the color potentials, and is based on a Gaussian Mixture Model (GMM). The λ represents the Location potential and it works as a look-up table. Finally, the ϕ term represents the edge potential, based on a contrast sensitive Potts model.

As the system is intended to be used in off-road driving applications, the following kinds of objects/labels are used: dirt, grass, road bushes, foliage, sky, tree trunk, post, and pedestrian.

Then, each ground type segment/object (e.g. dirt, grass, and road) is represented by a node in the semantic description graph, and the edges of the graph represent the neighborhood of the segments. Every node contains the following information: (i) type, (ii) spatial position, (iii) objects list, and (iv) traversability index. Additionally, road nodes include (v) curvature index, and (vi) odometry.

- The type of a node is the label acquired from the semantic segmentation, and it defines the kind of ground being represented by the node.

- The spatial position represents the relative spatial position of the node to the road. The node of the first ground segment in front of the vehicle is labeled as 'Road', and each other node is labeled as either 'Left' or 'Right'.

- The objects list corresponds to a list of the objects detected inside the segment being represented by the node, which are considered as important for the vehicle (e.g. trees, posts, pedestrians, etc.)

- The traversability index (TI) estimates how dangerous can be for the vehicle to drive over that region being represented by the road [13]. Each label from the semantic segmentation has associated a TI. The nodes TI is calculated as the most dangerous value between the TI of the objects in the node's object list, and the TI associated to the node's type.

- The curvature index is added for the road node only, and it is a coarse classification of the observed curvature of the road. This index can take 5 possible values 'Closed Left', 'Left', 'Straight', 'Right', and 'Closed Right' [10]. The road curvature is obtained by using a Hugh Transform based detection with a set of road curvatures on a flat world model.

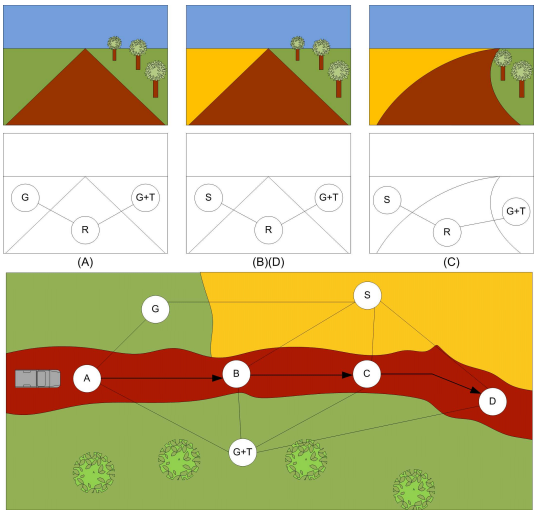


Figure 3. Example of a topological map. The first row shows three different images. The second row shows the semantic description. The third row shows the topological semantic map. For demonstrative purposes, sketch versions of the images are used instead of the real images. In this figure G stands for grass and S for sand. The node G+T has a grass type of ground with trees over it. A, B, C, and D are road nodes. B and D nodes have the same semantic description, the one second one, but they represent different regions of the road.

- Finally, the vehicle's odometry at the time of the image acquisition is included. This information will be later used to estimate the length of a section in the topological map.

In addition, each node includes pointers to the 'main' Left and Right nodes, where 'main' refers to the closest segment to the vehicle that is bigger than a given threshold in that side.

We define SD_k as the semantic description for the kth processed image, and SD_k.Road the road node of SD_k. Also, $SD_k.Left$ and $SD_k.Right$ are defined as the list of nodes in SD_k, tagged as *Left* and *Right*, respectively.

2.2. Topological Semantic Map

The Topological Semantic Map (TSM) is a graph structure that describes the road and its environment, based on the semantic description of the images described in the former section. The TSM is organized as a sequence of consecutive road nodes and two surrounding sequences of so called *environmental nodes*. A road node represents a region of the road defined by its semantic information. Each environmental node represents the main left or right neighboring node of a group of road nodes. Thus, environment nodes correspond to non-road nodes of the semantic description that are neighbors to the road.

Each road node in the TSM represents a section of the road where the semantic descriptions obtained from consecutive images are the same in terms of same main left node, main right node, and road curvature.

The environment nodes are located correspondently in the TSM using the spatial label of the semantic description (Left or Right) and are connected to its neighboring road nodes, creating a consistent description of the neighborhood of the road in the topological map.

The TSM is built incrementally from the semantic description for the observed image. Let us define M_k the TSM in time step k. M_k summarizes the information of k semantic descriptions SD_i, with $i = 1, , k$. The nodes of M_k have the same basic structure than the nodes of SD_i, but each road node in the TSM includes a *length* value, calculated as the difference between the first odometry observed in that node, and the first odometry from the next road node in the TSM.

Algorithm 1 Map update evaluation

function MAPPINGUPDATE(SD_k,M_{k-1})
  new_road_node $\leftarrow false$
  $M_k \leftarrow M_{k-1}$
  $M_k.Road.last.length \leftarrow SD_k.Road.odometry - first\_odom$
  if $SD_k.Road.curv \neq M_{k-1}.Road.curv$ **then**
    new_road_node $\leftarrow true$
  end if
  if $SD_k.Left.main \neq M_{k-1}.Left.last$ **then**
    $M_k.add_left(SD_k.Left.main)$
    new_road_node $\leftarrow true$
  end if
  if $SD_k.Right.main \neq M_{k-1}.Right.last$ **then**
    $M_k.add_right(SD_k.Right.main)$
    new_road_node $\leftarrow true$
  end if
  if new_road_node **then**
    $M_k.add_node(SD_k.Road.curv)$
    $first\_odom \leftarrow SD_k.Road.odometry$
  end if
end function

The TSM update criterion is detailed in Algorithm 1. Every new map node is added to one of the three lists of nodes: *Road, Left, or Right*, accordingly to its spatial label. The last pointer of each list points to the last added element to that list. The *add_left*, and *add_right* functions add a new environment node to the TSM, adding it to the *Left* or *Right* list, and updating their *last* pointer. The *add_node* function adds a new road node to the TSM, sets its Left and Right nodes as the last from the *Left* and *Right* lists, and updates the *Road* list *last* pointer.

For every new semantic description SD_k, the *MappingUpdate* function updates the current road node length and evaluates if the semantic description's road curvature, main left node, and main right node, are the same as the TSMs last road node curvature, the last left node, and the last right node. If the semantic description's main left node is different to the TSM last left node, then a new environment node is added to the TSM, and the same applies to the right nodes. If a new environmental node was added or if the road curvature is different, then a new road node is added to the map, using the semantic description's road node.

Figure 4 shows an example of a topological map built from the images of the road shown in the map below. The semantic descriptions for the images are presented in the second row, and the topological map is shown in the third row, over a virtual satellite image of the road.

When using this map structure, the vehicle's localization is reduced to the problem of finding the TSM road's node corresponding to the observations obtained by the vehicle. Given that, a vehicle position is defined as the one of the corresponding road's node. The proposed localization method is detailed in Algorithm 2.

Algorithm 2 Proposed localization method.

function LOCALIZATION(SD_k, $globalTSM_k$, $localTSM_k$)
  $localTSM_k$.MappingUpdate(SD_k)
  if is_lost **then**
    for i=1...length($hypothesisList$) **do**
      if $D(localTSM, globalTSM, hypothesisList[i]) \geq T$ **then**
        $hypothesisList$.remove(i)
      end if
      new_road_node $\leftarrow true$
    end for
    if length($hypothesisList$) $= 1$ **then**
      $is\_lost \leftarrow false$
      $pose\_estimation \leftarrow hypothesisList(1)$
    end if
    if length($hypothesisList$) $= 0$ **then**
      $localTSM$.reset()
      $hypothesisList \leftarrow globalTSM.Road$
    end if
  else
    if $D(localTSM, globalTSM, hypothesisList[i]) \geq T$ **then**
      $localTSM$.reset()
      $hypothesisList \leftarrow globalTSM.Road$
      $is\_lost \leftarrow true$
    else
      $pose\_estimation \leftarrow localTSM.Road$.last
    end if
  end if
end function

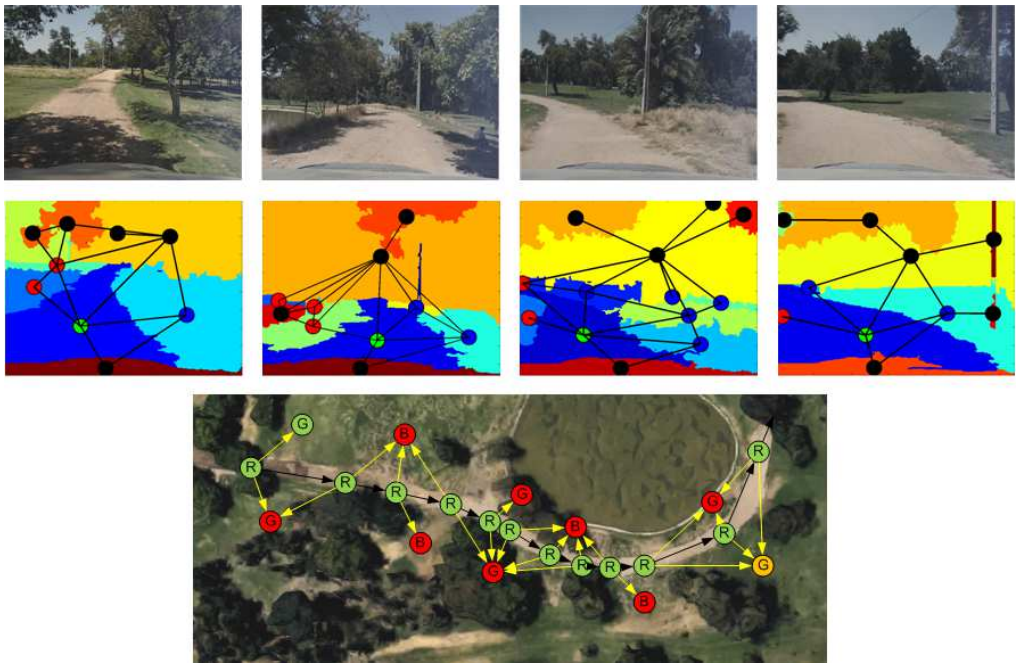


Figure 4. Example of a topological map generated with the proposed method. First row: Exemplar images from the used database. Second row: corresponding semantic description of these images. Third row: semantic map obtained displayed over the satellite image of the road (Source: Google Maps, satellite image).

The vehicle can be in two localization states: (i) located, or (ii) lost. While the vehicle has a TSM pose estimate consistent with the incoming observations, the vehicle is located. The vehicle is lost either if the pose estimate is inconsistent with the observations, or if there is no initial pose estimate.

Two TSM are used for localization purposes: a global TSM, which is a previously built map of the environment, and local TSM, built locally based on the last observations, and using the previously described mapping method.

When a lost condition is detected, the local semantic map is set as empty and the hypothesis list is filled with all the road nodes from the global TSM.

In lost state, the local TSM is compared with every candidate in the hypothesis list using the distance D (see equation 2), in each new observation (a.k.a. image) from the vehicle. If the distance is greater than a given threshold, that hypothesis is removed from the list. If after an observation, the list ends up with only one hypothesis in the list, then the vehicle's pose estimate is set as the hypothesis, and the vehicle is no longer lost. If the hypothesis list ends up empty, then the local TSM is reset, and the hypothesis list is refilled.

In located state, the local TSM only keeps track of the current road node. Here, the local TSM is compared with the estimated pose in the global TSM with the distance D. If the distance is greater than a second threshold, then the vehicle is set as lost, otherwise the pose estimation is updated.

$$D\left(M_a, M_b, k\right) = k_p \cdot P + k_q \cdot \sum_i Q_i. \qquad (2)$$

The distance function D (see equation 2) compares two TSM using the weighted sum of two terms: a P and Q. The P term depends on the length of the current road node of both TSM, and the Q term is proportional to the differences in information for each equivalent node between the TSMs.

3. Experiment

The proposed method is tested using the same database used by Parra-Tsunekawa et al. [11]. The database was recorded with the Advanced Mining Technology Center's (AMTC) autonomous vehicle (Volkswagen Tiguan 2010) inside O'Higgins Public Park, located near the downtown area of Santiago, Chile. The database was captured while the vehicle was driven on unpaved, rough terrain at low

(<10 [m/s]) and medium speeds (<20[m/s]). The unpaved area is formed by a very irregular road having a length of about 800 [m] with positive and negative slopes. The road is a track made of dirt surrounded of grass and some trees. The height difference between the lowest and the highest point of the track is about 4 meters.

For this work, only one segment of the database was used (starting at coordinates (-33.468614,-70.662464)). The image labeling and the object detection were manually annotated. The traversability index considers three levels: low, medium, and high. Grass and dirt regions have high traversability level (TL), while bushes have medium TL. Any obstacle detected close to the road border, at a distance smaller than the road width, has a low TL. Obstacles located at distances between 1 and 2 road widths of the road border have medium TL. Other obstacles have a high TL.

As a proof of concept, Figure 4 shows the results of the application of the proposed method over the used database. Four images of the dataset are shown in the first row. In the second row the semantic descriptions of the images are shown. The third row shows the topological map obtained, over a satellite image of the road. Each node is marked with a letter and color: 'R' stands for Road node, 'G for Grass node, and 'B' for Bushes, while the color represents the associated Traversability Index: green for high, orange for medium, and red for low.

The semantic map obtained is shown over the satellite image of the road to make a qualitative comparison of the topological map. The location of the environment nodes is just a reference. Naturally, only the topologic representation of the environment is the one that needs to be stored.

4. Conclusions

A novel outdoor semantic mapping method based on visual information and topological maps is presented and applied on a complex database.

Figure 4 shows that the resulting topological map is consistent to the satellite image of the road. The resulting map is composed by 12 road nodes and 10 environment nodes, which means that a road segment of about 160 meters is represented by a lightweight structure of only 22 nodes containing high level information of the road and its surroundings.

The results of the proposed method are consistent to the road environment, while keeping a lightweight and computationally efficient structure, making the proposed method suitable for large scale outdoor semantic mapping.

The high level interpretation of the road allows easy and fast ways to include high level deductions to the semantic map, as shown in the results of the experiment in Figure 4. By adding simple rules and expert information to the conceptual map, it is possible to give new kinds of information to an autonomous driving scheme, such as how dangerous can be to take the vehicle to the sides of the road in case of an emergency or an obstruction on the road.

Future work for this method includes automatic image labeling and object detection, the inclusion of range sensors for object detection and scene understanding. Another important challenge to solve for the proposed method is the closed loop case, and how can it be implemented keeping the method free from any global localization systems.

References

[1] B. Douillard, D. Fox, F. Ramos, and H. Durrant-Whyte. Classification and semantic mapping of urban environments. *The international journal of robotics research*, 30(1):5–32, 2011.

[2] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955–966, 2008.

[3] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J. Fernández-Madrigal, J. Gonzalez, et al. Multi-hierarchical semantic maps for mobile robotics. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 2278–2283. IEEE, 2005.

[4] C. Henson, A. Sheth, and K. Thirunarayan. Semantic perception: Converting sensory observations to abstractions. *Internet Computing, IEEE*, 16(2):26–34, 2012.

[5] I. Kostavelis and A. Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015.

[6] G.-J. M. Kruijff, H. Zender, P. Jensfelt, and H. I. Christensen. Situated dialogue and spatial organization: What, where... and why. *International Journal of Advanced Robotic Systems*, 4(2):125–138, 2007.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[8] O. M. Mozos, P. Jensfelt, H. Zender, G.-J. M. Kruijff, and W. Burgard. From labels to semantics: An integrated system for conceptual spatial representations of indoor environments for mobile robots. In *ICRA Workshop: Semantic Information in Robotics*, 2007.

[9] A. Nüchter and J. Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

[10] J. W. Park, J. W. Lee, and K. Y. Jhang. A lane-curve detection based on an lcf. *Pattern Recognition Letters*, 24(14):2301–2313, 2003.

[11] I. Parra-Tsunekawa, J. Ruiz-del Solar, and P. Vallejos. A kalman-filtering-based approach for improving terrain mapping in off-road autonomous vehicles. *Journal of Intelligent & Robotic Systems*, 78(3-4):577–591, 2015.

[12] A. Pronobis, A. Aydemir, K. Sjöö, and P. Jensfelt. Exploiting semantics in mobile robotics. In *ICRA 2012 Workshop on Semantic Perception, Mapping and Exploration*, 2012.

[13] A. L. Rankin, A. Huertas, and L. H. Matthies. Stereo-vision-based terrain mapping for off-road autonomous navigation. In *SPIE Defense, Security, and Sensing*, pages

733210–733210. International Society for Optics and Photonics, 2009.

[14] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. Torr. Urban 3d semantic modelling using stereo vision. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 580–585. IEEE, 2013.

[15] S. Sengupta and P. Sturgess. Semantic octree: Unifying recognition, reconstruction and representation via an octree constrained higher order mrf. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1874–1879. IEEE, 2015.

[16] S. Sengupta, P. Sturgess, P. H. Torr, et al. Automatic dense visual semantic mapping from street-level imagery. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 857–862. IEEE, 2012.

[17] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Computer Vision–ECCV 2006*, pages 1–15. Springer, 2006.

[18] H. Zender, P. Jensfelt, Ó. M. Mozos, G.-J. M. Kruijff, and W. Burgard. An integrated robotic system for spatial understanding and situated interaction in indoor environments. In *AAAI*, volume 7, pages 1584–1589, 2007.

C .2. Aceptado en Journal of Intelligent and Robotics Systems

Journal of Intelligent and Robotic Systems manuscript No.
(will be inserted by the editor)

Topological Semantic Mapping and Localization in Urban Road Scenarios

Fernando Bernuy · Javier Ruiz del Solar

Received: date / Accepted: date

Abstract Autonomous vehicle self-localization must be robust to environment changes, such as dynamic objects, variable illumination, and atmospheric conditions. Topological maps provide a concise representation of the world by only keeping information about relevant places, being robust to environment changes. On the other hand, semantic maps correspond to a high level representation of the environment that includes labels associated with relevant objects and places. Hence, the use of a topological map based on semantic information represents a robust and efficient solution for large-scale outdoor scenes for autonomous vehicles and Advanced Driver Assistance Systems (ADAS).

In this work, a novel topological semantic mapping and localization methodology for large-scale outdoor scenarios for autonomous driving and ADAS applications is presented. The methodology uses: (i) a deep neural network for obtaining semantic observations of the environment, (ii) a Topological Semantic Map (TSM) for storing selected semantic observations, and (iii) a topological localization algorithm which uses a Particle Filter for obtaining the vehicle's pose in the TSM.

The proposed methodology was tested on a real driving scenario, where a True Estimate Rate of the vehicle's pose of 96.9% and a Mean Position Accuracy of 7.7[m] were obtained. These results are much better than the ones obtained by other two methods used for comparative purposes. Experiments also show that the method is able to obtain the pose of the vehicle when its initial pose is unknown.

Keywords Autonomous Driving · Semantic Map · Topological Map · Semantic Segmentation · Semantic Localization · Topological Semantic Map

Universidad de Chile
Beauchef 850, Santiago. Chile.
E-mail: fbernuy@ing.uchile.cl

Advanced Minning Technology Center
Av. Tupper 2007, Santiago. Chile.

1 Introduction

Autonomous vehicle self-localization must be robust to environment changes, such as dynamic objects, variable illumination, and atmospheric conditions, to achieve long-term autonomy; these are some of the main open problems in modern Simultaneous Localization and Mapping (SLAM) systems according to Cadena et al. [7]. According to the author, "For long-term autonomy, techniques to construct and maintain large-scale time-varying maps, as well as policies that define when to remember, update, or forget information, still need a large amount of fundamental research", as there still unsolved issues as perceptual aliasing, that can lead to wrong loop closing that can severely corrupt the resulting map, and the multiple representation of the same place, that leads to an increasing memory consumption and processing time. In fact, these issues require a high level understanding and representation of the environment that makes metric maps inadequate, requiring alternative approaches such as topological and semantic maps.

Topological maps provide a concise representation of the world by only keeping information about relevant places; then, it is possible to self-localize by only recognizing key places. Several self-localization methods have been developed based on topological maps [6, 8, 12, 16, 21, 23, 26]. Particularly, Gadd and Newman [13] present a framework for building, managing, and sharing topological maps in a fleet of vehicles, using this approach.

A semantic map is defined as "a map that contains, in addition to spatial information about the environment, assignments of mapped features to entities of known classes" [25]. Thus, a semantic map contains labels associated to objects (e.g., "tree", "traffic sign") and places (e.g., "road", "building"), which provides a high level understanding of the world. Building a semantic map, or semantic mapping, requires perception systems able to convert raw observations into its corresponding abstractions [17], process known as semantic perception. The autonomous generation of semantic maps aims to combine the strengths of SLAM techniques and object categorization to recover semantic-spatial knowledge about the environment, which can be used for task planning and inference about nonsensed areas [14]. This has become a very active and growing research area due to its applications in unmanned cars, agriculture robotics, and any robotic application requiring human-robot interaction [19], due to the need for a robust operation in unconstrained environments and a more efficient task execution [27].

Mapping and self-localization are fundamental tasks for autonomous vehicles and Advanced Driver Assistance Systems (ADAS). GPS-based localization is often used, but GPS quality of service (QoS) is not ensured in general, due to tunnels or urban environments [23]. On the other hand, range sensors, such as LIDAR and RADAR, have been used for years and still are the main sensors for autonomous vehicles [9, 22, 24, 31]. However, recently there has been a significant increase in the number of vision-based solutions [6, 8, 12, 16, 21, 26]. This is related with the low cost of cameras [15], the dense information they provide, and the recent advances of Deep Neural Networks (DNN) [20] that have improved vision system's performance and execution time.

According to Kostavelis and Gasteros [19], "a challenge for the upcoming endeavors constitutes the semantic mapping of large scale outdoors scenarios," as most of the existing methods aim to solve the problem for indoor topological mapping environments rather than outdoors, and an important topic for indoor

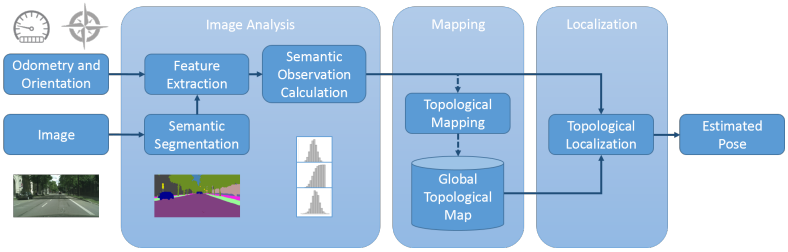


Fig. 1 Block diagram of the proposed method.

semantic mapping is place and object recognition, which is still underdeveloped for outdoors.

In order to address this challenge, a novel topological semantic mapping and localization methodology for large-scale outdoor scenarios is proposed. This approach relies on the vehicle's odometry and orientation, and a state-of-the-art semantic segmentation method to build a so-called Topological Semantic Map (TSM), which corresponds to a large scale topological map built to describe the environment and to allow self-localization. The results show that by using both the topological and semantic information in urban driving scenarios, where a limited number of object categories is found, the proposed method is able to self-localize efficiently. This methodology is a solution for large-scale autonomous navigation, with direct application in navigation tasks in highways, rural roads, and city areas traveled in regular basis, where the use and maintenance of metrics maps is challenging and computationally expensive. The methodology can also be used for building ADAS systems that can provide localization information to the driver.

The main contributions of this work are a graph based topological semantic mapping method suitable for large scale autonomous driving, and an efficient self-localization method based on the use of a topological semantic map and a particle filter.

This paper is organized as follows. First, in Section 2 the proposed mapping and localization methodologies are presented. The methodology is characterized and validated in Section 3 using urban driving datasets. Finally, in Section 4 conclusions of this work are drawn.

2 Proposed Methodology

The objective is to build a Topological Semantic Map (TSM) for efficient large-scale mapping and self-localization of autonomous vehicles using images from a frontward looking monocular camera located in the vehicle's roof. This method uses a high precision semantic segmentation algorithm computed by a state-of-the-art DNN-based method [32], allowing topological mapping for a wide range of urban road scenarios, and achieving simple and fast self-localization for ADAS or autonomous vehicles.

The proposed method, whose block diagram is shown in Fig.1, has three main stages: Image Analysis, Mapping, and Localization. In the Image Analysis stage,

(a) Original Image (b) Semantic Segmentation (c) Ground Truth

Fig. 2 Semantic Segmentation Example: (a) shows an image from the Cityscapes dataset[10], (b) shows the results of [32], and (c) is the ground truth of the dataset.

each image is processed using a state-of-the-art semantic segmentation algorithm to recognize the main elements of the road scene in order to obtain semantic features that describe the permanent elements observed, (e.g. building, poles, signs, traffic lights, etc.). The vehicle's odometry and orientation are added to every image as part of the feature vector. Later, the feature vectors are accumulated to create a *Semantic Observation*. In the Mapping stage, consecutive Semantic Observations are used to create a TSM, that is a high level representation of the environment. Finally, the Localization stage uses the Semantic Observation to estimate the pose of the vehicle in a previously built Global TSM. Each stage of the proposed method is explained below.

2.1 Image Analysis

The objective of this stage is to create a description based on the semantic segmentation obtained from the input image. This description includes information of the permanent elements observed (such as buildings, poles, signs, traffic lights, etc.) and uses spatial context from the distribution of those elements to group them together, creating a semantic description of the environment. Additionally, each semantic description includes a scalar odometry and orientation measurement from the vehicle, that allows the mapping procedure to include the estimated distance and orientation in the topological map. This stage includes three steps: Semantic Segmentation, Feature Extraction, and Semantic Observation Calculation.

2.1.1 Semantic Segmentation

The state-of-the-art semantic segmentation method proposed by Yu and Koltun [32] is applied to each input image in order to obtain a semantic label for each single pixel. This method uses dilated convolutions, a modified convolution operator that applies a filter in a bigger image area, allowing the neural network to support exponentially expanding receptive fields without losing resolution. This convolutional network architecture is divided in two modules: *Front-End* and *Context*. The *Front-End* module is a VGG-16 network [29] adapted for dense prediction by removing the last two pooling and striding layers and replacing the subsequent convolution for dilated convolutions. The outcome of the *Front-End* is a set of feature maps that can be directly used for dense semantic prediction of the image. The *Context* module increases the performance of the *Front-End* prediction by adding context information. It takes C feature maps as input and produces C feature maps as output, where C is the number of possible predicted classes. This

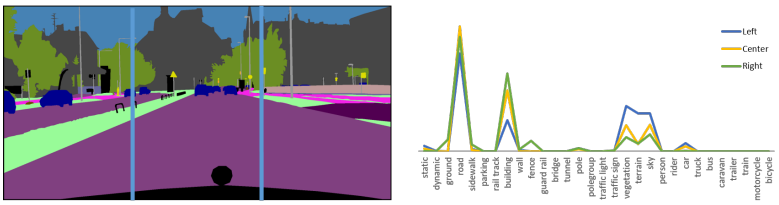


Fig. 3 Example of the features obtained from a segmented image. On the left the segmented image, and on the right the histograms of left, center, and right regions.

module has a fixed number of layers of dilated convolutions with different dilation factors, allowing the module to use spatial information of wider areas. A final layer performs a $1 \times 1 \times C$ convolution and produces the output of the whole network.

The network used in this work was trained using the Cityscapes Dataset [10] (see Fig.2), that defines 35 different labels. The labels are classified into four categories, in order to simplify the decision of which labels will be used for mapping and localization. The categories are:

Flat All forms of horizontal ground-level structures, including horizontal vegetation and ground parts for vehicles and pedestrians. The labels related with this category are: *ground, terrain, road, sidewalk, parking, and rail track.*

Background Any structure surrounding the roads and the sky. The labels of this category are: *sky, building, wall, fence, guard rail, bridge, and tunnel.*

Others Any other static element of interest for vehicles and pedestrian traffic. The labels related with this category are: *pole, pole-group, traffic light, traffic sign, and vegetation.*

Void Any label related with dynamic elements, such as vehicles and pedestrians, or with unknown or undefined elements. This labels are: *unlabeled, ego vehicle, rectification border, out of ROI, static, dynamic, person, rider, car, truck, bus, caravan, trailer, train, motorcycle, bicycle, and license plate.*

2.1.2 Feature Extraction

Each segmented image's information is summarized in a feature vector describing the observed environment semantically. The semantic feature vector

$$F = \begin{bmatrix} d \\ \theta \\ h_l \\ h_c \\ h_r \end{bmatrix} \tag{1}$$

is a column vector composed by five elements: a scalar value (d) that represents the accumulated vehicle odometry relative to the last frame, an angular measurement (θ) that represents the orientation of the vehicle, and three normalized histograms (h_l, h_c, and h_r) of the left, center and right regions of the semantic image. Each bin of the histograms represents one of the 35 classes of the semantic segmentation

algorithm. By building the feature vector this way, it is possible to associate the semantic information with different regions in the road, relative to the orientation of the vehicle.

Figure 3 shows a segmented image with its left, center, and right histograms. In this graph it is possible to identify important semantic differences between the regions; for example, the left region has more sky, terrain and vegetation pixels, and less buildings pixels than the other two regions.

In the Mapping and Localization modules it will be necessary to manipulate feature vectors to combine and compare their information. Since each element in a feature vector has different meaning, special binary operations must be defined: addition ($\oplus$) and distance $d(\cdot,\cdot)$.

The addition operator ($\oplus$) is a sum of feature vectors and results in a feature vector that represents the regions covered by the input features. The resulting feature's odometry is the sum of the input feature's odometries, and its histograms are a weighted average of its corresponding histograms according to each feature odometry:

$$F_a \oplus F_b = \begin{bmatrix} d^a \\ \theta^a \\ h_l^a \\ h_c^a \\ h_r^a \end{bmatrix} \oplus \begin{bmatrix} d^b \\ \theta^b \\ h_l^b \\ h_c^b \\ h_r^b \end{bmatrix} = \begin{bmatrix} d^a + d^b \\ \arctan\left(\frac{d^a \cdot \sin\theta^a + d^b \cdot \sin\theta^b}{d^a \cdot \cos\theta^a + d^b \cdot \cos\theta^b}\right) \\ \frac{d^a \cdot h_l^a + d^b \cdot h_l^b}{d^a + d^b} \\ \frac{d^a \cdot h_c^a + d^b \cdot h_c^b}{d^a + d^b} \\ \frac{d^a \cdot h_r^a + d^b \cdot h_r^b}{d^a + d^b} \end{bmatrix}. \tag{2}$$

The distance operator $d(\cdot,\cdot)$ measures the difference between the histograms of semantic features, and results in a scalar value obtained as the max cosine distance between the corresponding histograms of the features:

$$d(F_a, F_b) = \max_{k \in \{l,c,r\}} \left(1 - \frac{h_k^a \cdot h_k^b}{\|h_k^a\| \cdot \|h_k^b\|}\right), \tag{3}$$

where $h_k^a \cdot h_k^b$ is the dot product between histograms.

2.1.3 Semantic Observation Calculation

A Semantic Observation F_{SO} is a feature vector built by fusing the semantic feature vectors obtained from the input images. This vector accumulates similar feature vectors with a maximum of N_f features. If the current semantic observation accumulates N_f features or if its difference with the last feature vector F_t is larger than a given threshold t_s, then a new semantic observation is initialized with the last feature vector, as shown in Algorithm 1.

2.2 Topological Semantic Mapping

The TSM is a graph structure that describes the road and its environment, using information obtained from semantic observations, based on the proposed map by Bernuy and Ruiz-del-Solar [3]. The TSM is organized as a directed graph

Algorithm 1 Semantic Observation calculation.

```
function SemanticObservationCalculation(F_SO^{t-1}, F_t)
    if d(F_t, F_SO^{t-1}) < t_s and n < N_f then
        F_SO^t = F_SO^{t-1} ⊕ F_t
        n = n + 1
    else
        F_SO^t = newSemanticObservation(F_t)
        n = 1
    end if
end function
```

$G = (N, E)$ where each edge $e_i \in E$ represents a region of the road, and a node $n_i \in N$ represent intersections and connections between edges. An edge in the TSM represents a region where the observations obtained from consecutive images share similar orientation, and Left, Center, and Right histograms. Each edge e_i is described by a feature vector with the information of the images obtained in that region, its orientation, and its estimated length:

$$e_i = \begin{bmatrix} L \\ \theta \\ h_l \\ h_c \\ h_r \\ n_s \\ n_e \end{bmatrix}, \tag{4}$$

where i is the edge id, L is the length of the edge, θ is the mean orientation of the edge, $h_{l,c,r}$ are the histograms of the semantic segmentation, n_s is the starting node id, and n_e is the ending node id. Each node n_i contains the edge id of all the edges starting from this node, and all the edges ending in this node:

$$n_i = \begin{bmatrix} S_s \\ S_e \end{bmatrix}, \tag{5}$$

where i is the node id, S_s is the set of all the edges starting from the node, and S_e is the set of all the edges ending in the node.

The TSM is built incrementally by adding the semantic observations to the graph structure. The mapping algorithm works as follows: each new semantic observation F_{SO}^t is compared with the last added edge in the TSM ($M_{t-1}.last$), using the distance operator and the difference between the semantic observation's orientation ($F_{SO}^t.\theta$) and the edge orientation ($M_{t-1}.last.\theta$). If the resulting distance is less than a mapping threshold t_m, and the orientation differences is smaller than a orientation threshold t_o, then the semantic observation is assigned to the current edge by adding it with the additive operator $\oplus$. Otherwise, a new node is created and added to the TSM, and a new edge is added with the semantic observation:

$$M_t.last = \begin{cases} M_{t-1}.last \oplus F_{SO}^t & \textbf{if } d\left(F_{SO}^t, M_{t-1}.last\right) < t_m \\ & \textbf{and } |F_{SO}.\theta - M_{t-1}.last.\theta| < t_o\ , \\ M_t.NewEdge(F_{SO}^t) & \text{otherwise} \end{cases} \tag{6}$$

where M_t is a TSM in time step t. M_t summarizes the information of t Semantic Observation F^i_{SO}, with $i = 1, \ldots, t$. $M_t.last.F$ is a feature vector that summarizes the information of the images associated with the edge $M_t.last$. Algorithm 2 shows the pseudo-code of the mapping algorithm.

Algorithm 2 Topological Semantic Mapping Algorithm.

```
1: function MAPPING(F^t_SO, M_{t-1})
2:     if d(M_t.last, F^t_SO) < t_m and |F_SO.θ − M_{t-1}.last.θ| < t_o then
3:         M_t.last = M_t.last ⊕ F^t_SO
4:     else
5:         N = M_t.NewNode()
6:         M_t.last.SetEndingNode(N)
7:         M_t.NewEdge(F^t_SO)
8:         M_t.last.SetStartingNode(N)
9:     end if
10:    return M_k
11: end function
```

For the localization method, the *Global TSM* (GTSM) is defined as a previously built TSM that represents the world, and includes the information of length, orientation, and features of all the edges. This map is automatically built using the mapping algorithm described in Algorithm 2, and summarizes the information obtained from previous routes. As the features obtained from the semantic segmentation are a high-level description of the environment, it is not possible to recognize when the vehicle passes through a known place (i.e., two separated places can share a similar semantic feature), so the loop closing task is manually done. Nevertheless, it is possible to automate the job by using an additional SLAM algorithm, or an image-based place recognition algorithm.

2.3 Localization

List of Symbols

t	:	time index
i, j	:	index of edges in the GTSM
e_i	:	ith edge in the GTSM
k	:	index of particles
x_t	:	topological pose of the vehicle in time t as an edge of the GTSM
$\hat{x}_t$	:	estimated topological pose of the vehicle of the vehicle in time t
z_t	:	semantic observation at time t
x^t_k	:	topological pose of the kth particle in time t
w^t_k	:	importance weight of the kth particle in time t
l^t_k	:	unidimensional position in its edge of the kth particle in time t
N_p	:	Number of particles
S^i_s	:	Set of the edges that start in the ending node of the ith edge
S^i_e	:	Set of the edges that end in the starting node of the ith edge

When using the proposed map structure, the vehicle's self-localization problem is reduced to finding the edge corresponding to the semantic observations obtained by the vehicle. Hence, a vehicle estimated topological pose $\hat{x}_t$ is defined as one of the GTSM edges. The self-localization problem is modeled as a Hidden Markov Model (HMM) where the position of the vehicle in time t is the ith edge of the GTSM $\hat{x}_t = e_i$, and z_t is the observation obtained in time t.

The localization module uses each semantic observation $z_t = F^t_{SO}$ as an observation for the localization method, and its likelihood $p\left(z_t \mid e_i\right)$ is estimated using the maximum entropy distribution [11] supported in $[0, \infty]$ for the semantic information, and the Von Mises distribution [4] for the orientation, as:

$$p\left(z_t \mid e_i\right) \propto e^{-d(z_t, F_i)} \cdot e^{\kappa \cdot cos(\theta_{z_t} - \theta_i)}, \tag{7}$$

where F_i is the feature vector related for the edge e_i, θ_{z_t} is the orientation of the semantic observation, and θ_i is the orientation of the edge e_i.

Two localization methods are proposed and compared: a Forward Algorithm-based localization, and a Particle Filter-based localization.

Forward Localization In this method, the self-localization problem is solved using the Forward Algorithm [28]. The joint probability of the state $x_i = e_i$ and observations $z_{1:t}$ is defined as $\alpha_t(e_i) = p(e_i, z_{1:t})$. The Forward Algorithm calculates this probability recursively based on the conditional independence rules of the hidden Markov model as

$$\widehat{\alpha_t}(e_i) = p(z_t \mid e_i) \cdot \sum_{j \in GTSM} A^t_{i,j} \cdot \alpha_{t-1}(e_j), \tag{8}$$

where $p(z_t|e_i)$ is the likelihood for the observation z_t given the edge e_i obtained from equation (7), and $A^t_{i,j} = p(e_i|e_j)$ is the state transition probability from the jth edge to the ith, which is calculated as

$$A^t_{i,j} = \begin{cases} \frac{L_i - d_t}{L_i} & \text{if } i = j \\ \frac{1}{\left|S^j_s\right|} \cdot \frac{d_t}{L_j} & \text{if } i \in S^j_s \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where L_i and L_j are the length of the edges e_i and e_j, d_t is the accumulated odometry of the observation z_t, S^j_s is the set of all the edges that start in the node where the jth edge ends, and $\left|S^j_s\right|$is the cardinality of S^j_s.

Then, the joint probability of the states of the GTSM is normalized using the softmax function:

$$\alpha_t(e_i) = \frac{e^{\widehat{\alpha_t}(e_i)}}{\sum_j e^{\widehat{\alpha_t}(e_j)}}. \tag{10}$$

Finally, the localization problem is solved by finding the most probable edge $\hat{x}_t$ as

$$\hat{x}_t = \underset{i \in GTSM}{\operatorname{argmax}}\ \alpha_t(e_i). \tag{11}$$

The Forward Algorithm localization's complexity is $O(N_e)$, where N_e is the number of edges in the GTSM, as the algorithm needs to compare each observation with each edge.

Particle Filter Localization The main idea of this particle filter is to represent the posterior over the vehicle's topological pose x_t in the time t by a set of N_p weighted samples on the graph structure of the GTSM. This particle filter is based on the work of Merriaux et al. [23], where the kth particle has a pose $x_k^t = e_i$ in time t defined as the ith edge of the GTSM, a weight w_k^t, and an unidimensional position l_k^t along that edge. As the vehicle moves on the graph, the position transition function is defined as follows:

$$l_k^t = l_k^{t-1} + \nu(d^t), \tag{12}$$

where $\nu(d^t) \sim \mathcal{N}\left(d^t, \left(d^t\right)^2\right)$ is the accumulated odometry of the observation d^t with added Gaussian noise.

However, l_k^t may be greater than L_k, the length of the edge corresponding to the pose x_k^t, or less than zero. It means that the particle is leaving the current edge. As consequence, an edge transition function is required. Let $R_f^i \sim \mathcal{U}(S_s^i)$ be a random sample from S_s^i, the set of all the edges that start in the ending node of e_i, and $R_b^i \sim \mathcal{U}(S_e^i)$ be a random sample from S_e^i, the set of all the edges that end in the starting node of e_i. The edge transition function is defined as:

$$x_k^t = \begin{cases} R_b^i & \text{if } l_k^t < 0 \\ x_k^{t-1} & \text{if } l_k^t \in [0, L_k] \\ R_f^i & \text{if } l_k^t > L_k \end{cases}. \tag{13}$$

Also, if there is an edge transition, then the particle position l_k^t must be corrected by adding the new position length, or subtracting the old edge length L_k. After applying the position transition (12) and the edge transition functions (13) for each particle, the normalized weights w_t^i are calculated as

$$w_k^t = K_t \cdot p\left(z_t \mid x_k^t\right) \cdot w_k^{t-1}, \tag{14}$$

where K_t is a normalization factor, and $p\left(z_t \mid x_k^t\right)$ is the likelihood of the observation, obtained from (7).

A resampling procedure is performed to decrease the variance in the weights and avoid particle degeneracy. The resampled particle set is generated by sampling N_p new particles according to the distribution of the old particle weights (SIR algorithm [18]). The resampling is done if the effective sample size $\hat{N}_{eff} = 1/\sum_k (w_k)^2$is less than $N_p/2$ [1]. Finally, the localization problem is solved by finding the estimated pose $\hat{x}_t$ as the edge with the highest particle weight's sum:

$$\hat{x}_t = \underset{e_i \in \text{ GTSM}}{\operatorname{argmax}} \sum_{k=1}^{N_p} w_k \cdot \delta_{e_i}\left(x_k^t\right), \tag{15}$$

where $\delta_{e_i}\left(x_k^t\right)$ is the Kronecker delta, that returns 1 if $x_k^t = e_i$ and 0 otherwise.

The Particle Filter's execution time is related with the number of particles and the number of edges with a particle in it N_e^p, as the likelihood is calculated once for each edge with a particle in it. Therefore, the Particle Filter complexity is $O(|N_p| + |N_e^p|) \simeq O(|N_p|)$ when $|N_p| \gg |N_e^p|$.

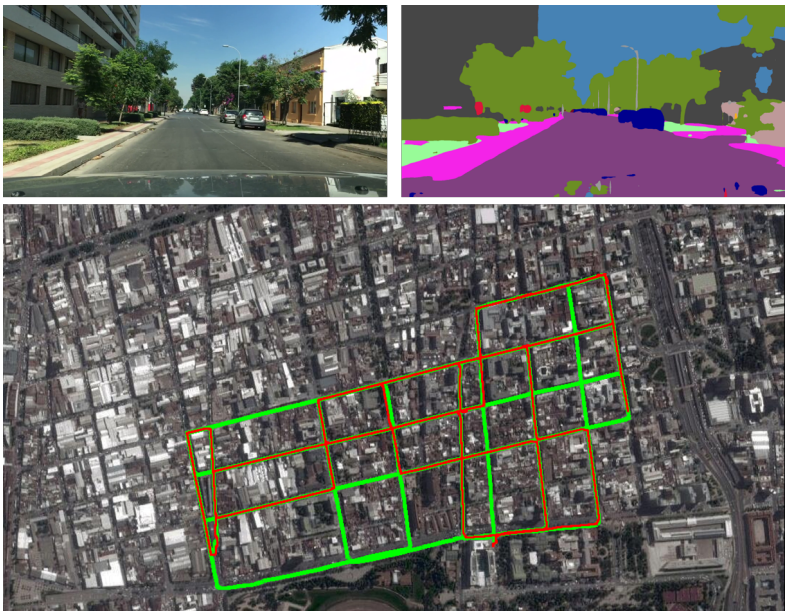


Fig. 4 Top: An example of the recorded database, the input image and the outcome of the segmentation algorithm [32]. Bottom: a satellite image showing the routes traveled in the dataset.

3 Experimental Results

The objective of this section is to validate the use of the proposed TSM in autonomous vehicles, using the proposed localization methodologies. These methods are validated using a dataset recorded in downtown Santiago, Chile. The dataset was recorded at 10 frames per second with a resolution of 1280x720, and contains two sequences recorded in two different days, following the trajectories described in Figure 4. The first sequence (in green) is 11.29[km] long, contains 31,144 images, and is used for mapping: the resulting GTSM includes 32 intersections and 14 streets with a total 9.57[km], as repeated runs over the same streets are discarded. The second sequence (in red) is 8.35[km] long, contains 20,718 images, and is used for testing the proposed localization methodologies.

The Dilation10 network proposed by Yu and Koltun [32] is used for the semantic segmentation. This network was trained using the *fine annotations* from the training set of the Cityscapes dataset [10], a large-scale dataset for pixel-level and instance-level semantic labeling. The trained network is available on-line in the author's website. The images were resized before the segmentation, as the network's input size is 2048x1024 pixels.

For evaluation and visualization purposes, each image in the dataset is associated with a GPS pose. Then, each edge in the GTSM includes the GPS poses

obtained from the corresponding images. Thus, the topological pose obtained by a given localization method is defined as correct, if distance between the current GPS pose and the closest GPS pose in the edge is less than 20[m], as in [5,23].

In order to make a quantitative evaluation of the performance of the proposed localization methods, it is necessary to define metrics that describe the behavior of the localization in a topological map. In [30] is defined a False Estimate as an incorrect localization output. Based on this, the True Estimate (TE) is defined for this work as 1 when the localization output is correct. Hence, the TE for the estimated pose $\hat{x}_t$ on time t is:

$$\mathrm{TE}_t = \begin{cases} 1 & \text{if } \hat{x}_t \text{ is correct} \\ 0 & \text{otherwise} \end{cases}. \tag{16}$$

For this work, the True Estimate Ratio (TER) is defined as the ratio between the amount of True Estimates and the total amount of pose estimations N_t (e.g. outputs of the localization method):

$$\mathrm{TER} = \frac{1}{N_t} \sum_t \mathrm{TE}_t. \tag{17}$$

Additionally, the Distance-based True Estimate Ratio (D-TER) is defined as the ratio between the distance traveled with a True Estimate, and the total distance traveled. This is a distance-oriented metric, and a fair evaluation of the two localization methods proposed, as both depend on the distance traveled for each update. The D-TER is calculated as shown in

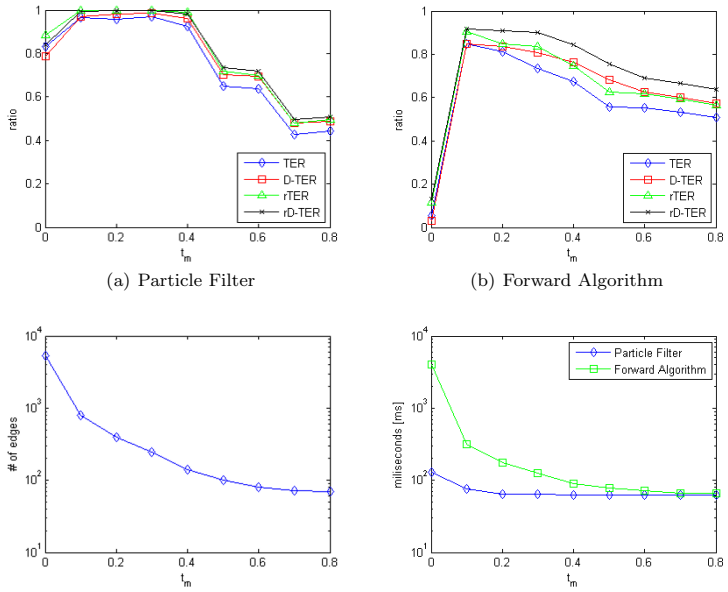
$$\text{D-TER} = \frac{\sum_t d_t \cdot \mathrm{TE}_t}{\sum_t d_t}, \tag{18}$$

where d_t is the accumulated odometry of the semantic observation in time t.

Additionally, a relaxed version of each metric is defined, relaxed TER (rTER) and relaxed D-TER (rD-TER), where the localization is considered correct if the distance to the estimated pose is less than 50[m].

Three experiments were carried out in order to analyze the behavior and performance of the proposed methods. The first experiment compares and characterizes the localization methods according to their performance in a topological map with known initial pose. The second experiment analyzes the convergence of the localization methods when initial pose is unknown. Finally, the third experiment evaluates the Particle Filter as an hybrid topological-metric localization system and compares with other existing methods.

For the following experiments, the GTSM was built using the mapping algorithm over the sequence 1, using only the *background* and *others* labels in the feature vector, as *flat* labels are commonly occluded by dynamic objects. The parameters values used for the experiments are: maximum number of frames per observation $N_f = 5$, orientation threshold $t_o = \pi/6$, observation threshold $t_s = 0.3$, and the mapping threshold t_m selected according to the results of the first experiment. The particle filter uses $N_p = 1,000$ particles. The details of each experiment are explained next.



(a) Particle Filter (b) Forward Algorithm

(c) Edges in GTSM (d) Execution Time

Fig. 5 Overall performance of the localization algorithms against the mapping threshold t_m. (a) and (b) display the performance metrics for the Particle Filter and the Forward Algorithm respectively. (c) is the number of edges in the GTSM in logarithmic scale, and (d) is the execution time for both algorithms in logarithmic scale.

Experiment 1 compares and characterizes the overall performance of the localization algorithms based on the performance metrics for different configurations. To do so, effect of the mapping threshold t_m is measured in the execution time of the algorithms, the number of edges in the Global TSM, and the four performance metrics. For this experiment initial position is know.

Here t_m takes values from 0.0 to 0.8. For $t_m = 0.0$, the mapping algorithm makes a new edge with each semantic observation. But with $t_m = 0.8$ or higher, it is unable to differentiate regions, hence the GTSM only has nodes where the orientation changes, or in street intersections. The results of this experiment are summarized in Figure 5.

According to the results of Figures 5(a) and 5(b), the precision of both methods is highly related to the mapping threshold. The results of the localization method are strongly related with how representative the accumulated description of an edge is. As the edges represent road segments with similar semantic description based on the mapping threshold, a permissive threshold makes accumulated description to be too general and not distinctive enough for localization, while a correct threshold makes the localization to achieve accurate localization. The Forward Algorithm reaches his maximum performance with $t_m = 0.1$, while the

Table 1 Results of Experiment 1 according to the True Estimate Rate (TER), Distance based True Estimate Rate (D-TER), the relaxed TER (rTER), the relaxed D-TER (rD-TER), the Mean Error and the Mean False Estimate Error. Performance metrics of the best result obtained for each localization algorithm.

Method	Particle Filter $t_m = 0.3$	Forward Algorithm $t_m = 0.1$
True Estimate Rate (TER)	**0.9690**	0.8472
Distance-based TER (D-TER)	**0.9867**	0.8475
Relaxed TER (rTER)	**0.9998**	0.9030
Relaxed D-TER(rD-TER)	**0.9996**	0.9186
Mean Error [m]	**3.9**	14.6
Mean False Estimate Error [m]	**25.3**	72.7

Particle Filter reaches its maximum with $t_m = 0.3$. These results show that the use of a topological map, where the world is segmented into semantically similar regions, allows a better self-localization than when using the semantic information of a single image to describe the environment.

The Particle Filter achieves significantly better results than the Forward Algorithm ($\sim 10\%$ difference) for all the performance metrics with t_m between 0.0 and 0.4. For higher values of t_m, the semantic information of the edges is not representative enough to allow the proposed localization methods to keep track of the vehicle localization.

As it can be observed in Figure 5(c), the number of edges in the GTSM increases exponentially as the mapping threshold decreases, which means that the memory required for the map is strongly related with this parameter. Figure 5(d) confirms that the execution time of the Forward Algorithm is strongly related with the map size, while the Particle Filter's execution time depends on the number of particles and their dispersion on the GTSM, but not on the total number of edges in the map. As the system is meant to work at 10 frames per second, the processing time achieved for the Particle Filter with $t_m \geq 0.2$ ($\simeq 60$[ms]) is adequate for autonomous driving. Nevertheless, it can be improved with an optimized implementation of the current Matlab code. According to this results, the Particle Filter approach is a suitable solution for large-scale maps. The Particle Filter is the best self-localization solution for TSM, when considering the number of edges, execution time and the performance.

Table 1 shows the performance metrics for the best configuration of each localization method. In addition to the previous defined metrics, it includes two new metrics: the Mean Error, and the Mean False Estimate Error. The Mean Error is defined as the average distance from the ground truth to the estimated pose over the test sequence. The Mean False Estimate Error is defined as the mean distance from the estimated pose to the ground truth pose over the incorrectly localized estimates.

The Particle Filter shows better results than the Forward Algorithm for all the performance metrics used, with over a 12% difference for the TER and D-TER metrics, and over a 8% for the relaxed metrics. The Forward Algorithm's Mean Error is more than 3 times larger than the Particle Filter's, while the Mean False Estimate Error is 2.8 times larger. Also, the Mean Error obtained for the particle filter is smaller than the 5.8[m] obtained in [23], and similar to the ones

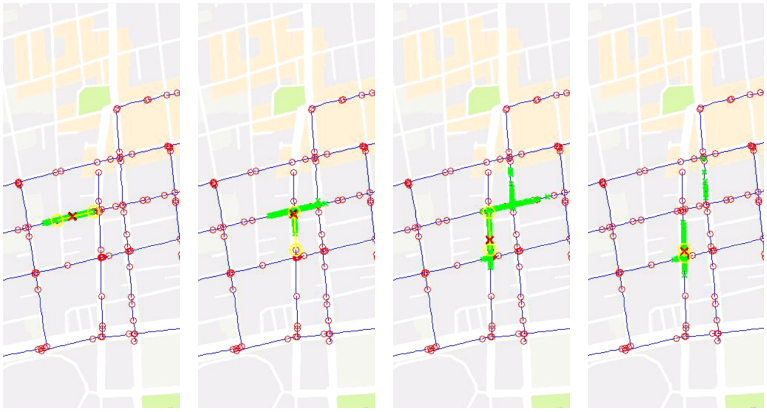


Fig. 6 Sequence of images from the particle filter localization behavior in intersections. The GTSM is shown with red circles for the nodes and blue lines for the edges. The green crosses are particles, the red cross is the GPS position of the vehicle, and the yellow segment is the estimated topological pose. (a) shows most of the particles in the estimated pose with the vehicle moving right. (b) shows a right turn, with the particles splitting in the intersection. (c) shows how the particle filter keeps multiple hypothesis for the vehicle estimation, choosing the correct one. And (d) shows how the resampling stage discards most of the wrong particles.

presented in [5]. The high precision obtained by the Particle Filter, according to the performance metrics and the Mean Error, shows that the proposed methodology is suitable for ADAS and autonomous vehicles to solve the self-localization problem on large-scale urban scenarios.

Another result from this experiment is that D-TER was higher than TER, which can be explained by noticing that the vehicle stops near some intersections with traffic lights that separate two edges in the GTSM. This can lead to numerous frames taken in the same position to be considered as incorrectly localized that have a minor influence in the D-TER. The high values obtained for the relaxed metrics and the Mean False Estimate Error, shows that when the estimated pose is incorrect, it is close to the correct one in the topological map. This allows Particle Filter localization to recover quickly from localization errors. An example of the Particle Filter-based localization method is shown in Figure 6, using $t_m = 0.3$, resulting an a GTSM with 232 edges, with an average length of 41.3[m]. This sequence shows the particles dispersion when the vehicle takes a turn on an intersection, and how the resampling method discard the less likely particles. The complete sequence is available in [2].

Experiment 2 evaluates the convergence of the localization methods when the vehicle has an unknown initial pose. Here, the Forward Algorithm is initialized using a uniform probability distribution for the pose over the GTSM, and the Particle Filter is initialized with a particle every 5[m] in the GTSM (∼2,000 particles), to ensure that all possible initial states are covered. The second sequence is used as input but starting from random positions. A total of 1,000 runs of the

Table 2 Results of Experiment 2, using Sequence 2 and unknown initial self-localization.

Method	Localization Time [s]	Success Rate [%]
Particle Filter	31.1	**100%**
Forward Algorithm	**23.9**	**100%**

experiment were made for each localization algorithm, and two values are obtained for each one: the Localization Time, and the Success Value. The Localization Time is the elapsed time until a correct localization is hold for at least 5[s], as in [5]. The Success Value is 0 if the localization method is unable to find the right localization, and 1 otherwise. Thus, the evaluation of this experiment is based on two metrics: The Success Ratio and the Mean Localization Time. The Success Ratio is the average Success Value among the experiment runs, and the Mean Localization Time is the mean localization time among the successful executions.

The results summarized in Table 2 show that the Particle Filter requires 30% more time to converge than the Forward Algorithm, while both localization methods reach a perfect Success Rate over the experiments run. The Forward Algorithm updates the joint probability distribution for the whole map, hence it is expected to reach a high success rate. On the other hand, the particle filters are prone to reach local minimum with a wrong estimation. To solve this issue, a re-initialization step is included that distributes particles every 5[m] in the GTSM whenever the resulting pose estimate has a low likelihood for several observations. A negative consequence of the use of this step, is a significant increase in the localization time. For this experiment, 92.4% of the runs reached a successful localization without the need of a reinitialization step, which demonstrates the importance of this step for a robust operation of the localization method.

Experiment 3 studies the performance of the proposed particle-based methodology for metric localization. Although this methodology is meant to work in a topological manner, the Particle Filter approach allows the system to estimate a metric position of the vehicle based on the TSM nodes position. On the other hand, the Forward Algorithm is not included in this experiment as there is no straightforward way to get a metric pose from its pose estimation. For this experiment, each TSM node includes a GPS pose, and each particle's metric pose is estimated by a linear interpolation between the starting and ending nodes GPS pose of the particle's edge. Later, the metric pose estimation is obtained as the weighted mean of the particles poses.

To make a valid performance comparison, two other topological map-based methods are used. The first method, proposed by Brubaker et al. [5], uses visual odometry to estimate the movement of the vehicle, and a Mixture of Gaussians over position and orientation to find a position in a map from OpenStreetMaps (OSM). Hence, the performance of this method depends on the map geometry and the vehicle trajectory. The method was run using libviso2 for visual odometry and the code available in the author website for the localization algorithm. The second method, proposed by Merriaux et al. [23], uses the vehicle's odometry and a particle filter over an OSM map to estimate the vehicle's pose, making its performance also dependent on the map geometry and the vehicle's trajectory. This method was implemented for this comparison. For a quantitative comparison,

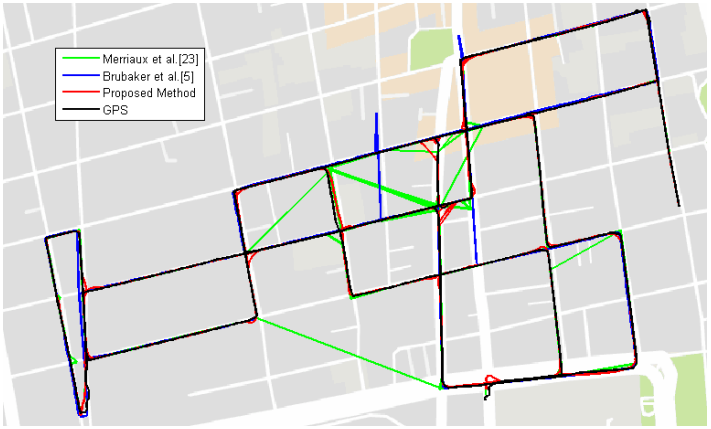


Fig. 7 Estimated trajectory for topological map-based localization algorithms. On red the estimated trajectory from the Particle Filter proposed method, on green the method proposed by Merriaux et al. [23], on blue the trajectory from Brubaker et al. [5], and on black the GPS trajectory used as ground truth.

Table 3 Results of Experiment 3 on metric localization.

Method	Mean Position Error [m]	GPS to Map Error [m]
Proposed Method	**7.7**	2.50
Brubaker et al. [5]	34.6	6.97
Merriaux et al. [23]	51.9	6.97

the Mean Position Error is obtained as the mean minimum distance between the pose estimation and the map (GTSM or OSM) across the sequence. Additionally, the same value is obtained for the GPS against the map used, GTSM or OSM.

Table 3 shows the resulting Mean Position Error for the compared methods. It can be observed that how the proposed Particle Filter-based method performs better than the other topological map-based localization algorithm. Figure 7 shows the trajectories obtained by the different localization algorithms, and the GPS poses used as ground truth. The trajectory generated by the method of Merriaux et al. [23] jumps through different regions of the map due to its particle filter keep several hypotheses of the vehicle position that can not be discarded with the odometry. On the other hand, the method proposed by Brubaker et al. [5] keeps a stable estimation of the vehicle pose. But, when the vehicle turns after a long straight road, the high variance of the pose estimation lead the method to generate more than one hypothesis on parallel roads, and the pose estimation is set on the wrong hypothesis. The estimation is later corrected when the vehicle's trajectory turns and increases the right hypothesis. It is important to remark that both methods are able to recover from miss-localization events due to the ability to handle multiple hypotheses. Finally, the proposed method follows the vehicle

localization during all the sequence with only some minor errors in intersections. Some example errors are detailed in Figure 8.

The low performance achieved by Merriaux et al. [23] and Brubaker et al. [5] methods is related to the complexity of the dataset, recorded on a grid street plan. As both methods rely on the vehicle's odometry to update the pose estimation, they can only correct the estimation when the vehicle turns, or when the pose estimation is expected to turn and the vehicle does not. This leads to wrong pose estimations after long straight roads. On the other hand, the proposed Particle Filter method continuously uses semantic information to correct the pose estimation regardless the vehicle's trajectory.

As a general overview of the results obtained from the presented experiments, it is possible to notice that the Particle Filter shows the best performance with a proper initial probability distribution for the pose, while the Forward Algorithm has the fastest results for uncertain initial self-localization. The particle filter-based localization with the TSM represents a robust solution for autonomous vehicles and ADAS for large-scale autonomous navigation. Finally, when evaluated as a metric localization system, the proposed Particle Filter method shows good results when compared with other topological map-based localization algorithms.

4 Conclusions

A novel topological semantic mapping and localization methodology for large-scale outdoor scenes in autonomous driving applications was presented and tested under realistic driving conditions. The tests were carried out using a dataset consisting on video sequences recorded in downtown Santiago, using a monocular camera mounted on a commercial vehicle under normal traffic conditions. The dataset consists of two sequences of 31,144 and 20,718 images each one, recorded at 10 frames per second, with a resolution of 1,280x720. This dataset will be available at: http://vision.die.uchile.cl/databases.php.

The proposed mapping method was able to describe a traveled path using the proposed TSM structure, allowing a successful self-localization under a complex urban driving scenario. For the best configuration of the method's parameters, the topological map divides the map using 232 edges with an average length of 41.3[m].

Two localization methods were compared under the proposed methodology. The Particle Filter method obtained the best performance when the initial position of the vehicle is known, while Forward Algorithm showed the fastest convergence when the initial localization is unknown. Nevertheless, the particle filter was able to keep precise pose estimation on 98.67% of the traveled route, and was able to self-localize correctly after an unknown initial localization in 100% of the cases. The results show that the particle filter-based localization for the topological semantic map represents a robust solution for ADAS and large-scale autonomous navigation, with direct application in navigation tasks in highways, rural roads, or city areas traveled in regular basis.

The proposed methodology does not need any metric information for its execution. Nevertheless, it is possible to add metric information to the nodes of the GTSM in order to obtain a metric pose estimation, with better accuracy than other topological map-based localization methods.

Fig. 8 Error examples of the proposed method. (a) displays an error related with driving on a different lane in the same road. (b) shows the effect of the particle dispersion in the pose estimation near an intersection. (c) and (d) show GPS related errors, as the ground truth poses are outside of the road in the map. And (e) and (f) show intersections where pose estimation goes wrong but is soon corrected.

Acknowledgements This work was partially funded by FONDECYT Project 1161500.

References

1. Arulampalam, M.S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on signal processing **50**(2), 174–188 (2002)
2. Bernuy, F., Ruiz del Solar, J.: Topological semantic mapping and localization demo. URL https://youtu.be/H9TPohkC4yE
3. Bernuy, F., Ruiz del Solar, J.: Semantic mapping of large-scale outdoor scenes for autonomous off-road driving. In: 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), pp. 124–130. IEEE (2015)
4. Best, D., Fisher, N.I.: Efficient simulation of the von mises distribution. Applied Statistics pp. 152–157 (1979)
5. Brubaker, M.A., Geiger, A., Urtasun, R.: Lost! leveraging the crowd for probabilistic visual self-localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3057–3064 (2013)
6. Brubaker, M.A., Geiger, A., Urtasun, R.: Map-based probabilistic visual self-localization. IEEE transactions on pattern analysis and machine intelligence **38**(4), 652–665 (2016)
7. Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.J.: Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics **32**(6), 1309–1332 (2016)
8. Churchill, W., Newman, P.: Experience-based navigation for long-term localisation. The International Journal of Robotics Research **32**(14), 1645–1661 (2013)
9. Cole, D., Newman, P.: Using laser range data for 3d SLAM in outdoor environments. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Orlando Florida USA (2006)
10. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
11. Cover, T.M., Thomas, J.A.: Elements of information theory. John Wiley & Sons (2012)
12. Floros, G., van der Zander, B., Leibe, B.: Openstreetslam: Global vehicle localization using openstreetmaps. In: Robotics and Automation (ICRA), 2013 IEEE International Conference on, pp. 1054–1059. IEEE (2013)
13. Gadd, M., Newman, P.: Checkout my map: Version control for fleetwide visual localisation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2016)
14. Galindo, C., Fernández-Madrigal, J.A., González, J., Saffiotti, A.: Robot task planning using semantic maps. Robotics and Autonomous Systems **56**(11), 955–966 (2008)
15. Garcia-Fidalgo, E., Ortiz, A.: Vision-based topological mapping and localization methods: A survey. Robotics and Autonomous Systems **64**, 1–20 (2015)
16. Glover, A.J., Maddern, W.P., Milford, M.J., Wyeth, G.F.: Fab-map+ ratslam: Appearance-based slam for multiple times of day. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 3507–3512. IEEE (2010)
17. Henson, C., Sheth, A., Thirunarayan, K.: Semantic perception: Converting sensory observations to abstractions. Internet Computing, IEEE **16**(2), 26–34 (2012)
18. Kitagawa, G.: Monte carlo filter and smoother for non-gaussian nonlinear state space models. Journal of computational and graphical statistics **5**(1), 1–25 (1996)
19. Kostavelis, I., Gasteratos, A.: Semantic mapping for mobile robotics tasks: A survey. Robotics and Autonomous Systems **66**, 86–103 (2015)
20. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
21. Linegar, C., Churchill, W., Newman, P.: Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 90–97. IEEE (2015)
22. Maddern, W., Pascoe, G., Newman, P.: Leveraging Experience for Large-Scale LIDAR Localisation in Changing Cities. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). Seattle, WA, USA (2015)
23. Merriaux, P., Dupuis, Y., Vasseur, P., Savatier, X.: Fast and robust vehicle positioning on graph-based representation of drivable maps. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 2787–2793. IEEE (2015)

24. Newman, P., Sibley, G., Smith, M., Cummins, M., Harrison, A., Mei, C., Posner, I., Shade, R., Schroeter, D., Murphy, L., Churchill, W., Cole, D., Reid, I.: Navigating, recognising and describing urban spaces with vision and laser. The International Journal of Robotics Research **28** (2009). DOI 10.1177/0278364909341483
25. Nüchter, A., Hertzberg, J.: Towards semantic maps for mobile robots. Robotics and Autonomous Systems **56**(11), 915–926 (2008)
26. Paul, R., Newman, P.: Fab-map 3d: Topological mapping with spatial and visual appearance. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 2649–2656. IEEE (2010)
27. Pronobis, A., Aydemir, A., Sjöö, K., Jensfelt, P.: Exploiting semantics in mobile robotics. In: ICRA 2012 Workshop on Semantic Perception, Mapping and Exploration (2012)
28. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE **77**(2), 257–286 (1989)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
30. Tomatis, N., Nourbakhsh, I., Siegwart, R.: Hybrid simultaneous localization and map building: a natural integration of topological and metric. Robotics and Autonomous systems **44**(1), 3–14 (2003)
31. Wolcott, R.W., Eustice, R.M.: Visual localization within lidar maps for automated urban driving. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 176–183. IEEE (2014)
32. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. In: ICLR (2016)

Bibliografía

[1] J. M. Armingol, A. de la Escalera, C. Hilario, J. M. Collado, J. P. Carrasco, M. J. Flores, J. M. Pastor, and F. J. Rodríguez. Ivvi: Intelligent vehicle based on visual information. *Robotics and Autonomous Systems*, 55(12):904–916, 2007.

[2] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, 2002.

[3] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, et al. Odin: Team victortango's entry in the darpa urban challenge. *Journal of Field Robotics*, 25(8):467–492, 2008.

[4] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[5] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 255–260. IEEE, 2005.

[6] F. Bernuy. Detección de calzada para un vehículo autónomo. 2011.

[7] F. Bernuy and J. Ruiz del Solar. Topological semantic mapping and localization demo. URL `https://youtu.be/H9TPohkC4yE`.

[8] F. Bernuy and J. Ruiz-del Solar. Semantic mapping of large-scale outdoor scenes for autonomous off-road driving. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 35–41, 2015.

[9] F. Bernuy and J. Ruiz-del Solar. Unstructured road segmentation: A comparative study. *Submitted to Image and Vision Computing*, 2016.

[10] F. Bernuy, J. Ruiz del Solar, I. Parra, and P. Vallejos. Adaptive and real-time unpaved road segmentation using color histograms and ransac. In *2011 9th IEEE International Conference on Control and Automation (ICCA)*, pages 136–141, Dec 2011.

[11] D. Best and N. I. Fisher. Efficient simulation of the von mises distribution. *Applied Statistics*, pages 152–157, 1979.

[12] A. Bevilacqua, A. Lanza, G. Baccarani, and R. Rovatti. A single-scan algorithm for connected components labelling in a traffic monitoring application. In *Image Analysis*, pages 677–684. Springer, 2003.

[13] D. Braid, A. Broggi, and G. Schmiedel. The terramax autonomous vehicle concludes the 2005 darpa grand challenge. In *2006 IEEE Intelligent Vehicles Symposium*, pages 534–539. IEEE, 2006.

[14] M. A. Brubaker, A. Geiger, and R. Urtasun. Lost! leveraging the crowd for probabilistic visual self-localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3057–3064, 2013.

[15] M. A. Brubaker, A. Geiger, and R. Urtasun. Map-based probabilistic visual self-localization. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):652–665, 2016.

[16] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *arXiv preprint arXiv:1606.05830*, 2016.

[17] J. Chen, T. N. Pappas, A. Mojsilovic, and B. E. Rogowitz. Adaptive perceptual color-texture image segmentation. *IEEE Transactions on Image Processing*, 14(10):1524–1536, 2005.

[18] W. Churchill and P. Newman. Experience-based navigation for long-term localisation. *The International Journal of Robotics Research*, 32(14):1645–1661, 2013.

[19] D. Cole and P. Newman. Using laser range data for 3d SLAM in outdoor environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Orlando Florida USA, May 2006.

[20] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[21] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] M. Correa, J. Ruiz-del Solar, R. Verschae, J. Lee-Ferng, and N. Castillo. Real-time hand gesture recognition for human robot interaction. In *RoboCup 2009: Robot Soccer World Cup XIII*, pages 46–57. Springer, 2010.

[23] T. M. Cover and J. A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[24] M. Cummins and P. Newman. Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9):1100–1123, 2011.

[25] I. N. de Estadisticas and C. de Chile. Carabineros de Chile, informe anual, Aug. 2016. URL `http://www.ine.cl/canales/menu/publicaciones/calendario_de_publicaciones/pdf/carabineros_2015.pdf`.

[26] J. Domke. Learning graphical model parameters with approximate marginal inference. *IEEE transactions on pattern analysis and machine intelligence*, 35(10):2454–2467, 2013.

[27] B. Douillard, D. Fox, F. Ramos, and H. Durrant-Whyte. Classification and semantic mapping of urban environments. *The international journal of robotics research*, 30(1):5–32, 2011.

[28] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[29] G. Floros, B. van der Zander, and B. Leibe. Openstreetslam: Global vehicle localization using openstreetmaps. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1054–1059. IEEE, 2013.

[30] M. Gadd and P. Newman. Checkout my map: Version control for fleetwide visual localisation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[31] D. W. Gage. Ugv history 101: A brief history of unmanned ground vehicle (ugv) development efforts. Technical report, DTIC Document, 1995.

[32] C. Galindo, J.-A. Fernández-Madrigal, J. González, and A. Saffiotti. Robot task planning using semantic maps. *Robotics and autonomous systems*, 56(11):955–966, 2008.

[33] C. Galindo, A. Saffiotti, S. Coradeschi, P. Buschka, J.-A. Fernandez-Madrigal, and J. Gonzalez. Multi-hierarchical semantic maps for mobile robotics. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2278–2283. IEEE, 2005.

[34] X. Gallart Del Burgo. Semantic mapping in ros. 2013.

[35] A. J. Glover, W. P. Maddern, M. J. Milford, and G. F. Wyeth. Fab-map+ ratslam: Appearance-based slam for multiple times of day. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3507–3512. IEEE, 2010.

[36] H. Grimmett, M. Buerki, L. Paz, P. Pinies, P. Furgale, I. Posner, and P. 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.

[37] C. Guo, S. Mita, and D. McAllester. Robust road detection and tracking in challenging scenarios based on markov random fields with unsupervised learning. *IEEE Transactions on intelligent transportation systems*, 13(3):1338–1354, 2012.

[38] C. Guo, T. Yamabe, and S. Mita. Robust road boundary estimation for intelligent vehicles in challenging scenarios based on a semantic graph. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 37–44. IEEE, 2012.

[39] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[40] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, 26(2):120–144, 2009.

[41] R. M. Haralick, K. Shanmugam, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973.

[42] M. Häselich, M. Arends, D. Lang, and D. Paulus. Terrain classification with markov random fields on fused camera and 3d laser range data. In *ECMR*, pages 153–158, 2011.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[44] C. Henson, A. Sheth, and K. Thirunarayan. Semantic perception: Converting sensory observations to abstractions. *IEEE Internet Computing*, 16(2):26–34, 2012.

[45] A. B. Hillel, R. Lerner, D. Levi, and G. Raz. Recent progress in road and lane detection: a survey. *Machine vision and applications*, 25(3):727–745, 2014.

[46] K. Iagnemma and M. Buehler. Editorial for journal of field robotics-special issue on the darpa grand challenge. *Journal of Field Robotics*, 23(9):655–656, 2006.

[47] B. Julesz et al. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91–97, 1981.

[48] I. Katramados, S. Crumpler, and T. P. Breckon. Real-time traversable surface detection by colour space fusion and temporal analysis. In *International Conference on Computer Vision Systems*, pages 265–274. Springer, 2009.

[49] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

[50] H. Kong, J.-Y. Audibert, and J. Ponce. Vanishing point detection for road detection. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 96–103. IEEE, 2009.

[51] H. Kong, J.-Y. Audibert, and J. Ponce. General road detection from a single image. *IEEE Transactions on Image Processing*, 19(8):2211–2220, 2010.

[52] A. Kornhauser, A. Atreya, B. Cattle, S. Momen, B. Collins, A. Downey, G. Franken, J. Glass, Z. Glass, J. Herbach, et al. Darpa urban challenge princeton university technical paper. *DARPA Urban Challange*, 2007.

[53] I. Kostavelis and A. Gasteratos. Semantic mapping for mobile robotics tasks: A survey. *Robotics and Autonomous Systems*, 66:86–103, 2015.

[54] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[55] G.-J. M. Kruijff, H. Zender, P. Jensfelt, and H. I. Christensen. Situated dialogue and spatial organization: What, where... and why. *International Journal of Advanced Robotic Systems*, 4(2):125–138, 2007.

[56] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. On measuring the accuracy of slam algorithms. *Autonomous Robots*, 27(4):387–407, 2009.

[57] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[58] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[59] C. Linegar, W. Churchill, and P. Newman. Work smart, not hard: Recalling relevant experiences for vast-scale but time-constrained localisation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 90–97. IEEE, 2015.

[60] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[61] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[62] W. Maddern, G. Pascoe, and P. Newman. Leveraging Experience for Large-Scale LIDAR Localisation in Changing Cities. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, USA, May 2015.

[63] J. C. McCall and M. M. Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *IEEE transactions on intelligent transportation systems*, 7(1):20–37, 2006.

[64] C. McManus, W. Churchill, A. Napier, B. Davis, and P. Newman. Distraction suppression for vision-based pose estimation at city scales. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3762–3769. IEEE, 2013.

[65] P. Merriaux, Y. Dupuis, P. Vasseur, and X. Savatier. Fast and robust vehicle positioning on graph-based representation of drivable maps. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2787–2793. IEEE, 2015.

[66] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497, 2012.

[67] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[68] O. M. Mozos, P. Jensfelt, H. Zender, G.-J. M. Kruijff, and W. Burgard. From labels to semantics: An integrated system for conceptual spatial representations of indoor environments for mobile robots. In *ICRA Workshop: Semantic Information in Robotics*, 2007.

[69] P. Newman, G. Sibley, M. Smith, M. Cummins, A. Harrison, C. Mei, I. Posner, R. Shade, D. Schroeter, L. Murphy, W. Churchill, D. Cole, and I. Reid. Navigating, recognising and describing urban spaces with vision and laser. *The International Journal of Robotics Research*, 28, October 2009.

[70] N. J. Nilsson. A mobile automaton: An application of artificial intelligence techniques. Technical report, DTIC Document, 1969.

[71] A. Nüchter and J. Hertzberg. Towards semantic maps for mobile robots. *Robotics and Autonomous Systems*, 56(11):915–926, 2008.

[72] T. N. Pappas. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on signal processing*, 40(4):901–914, 1992.

[73] I. Parra-Tsunekawa, J. Ruiz-del Solar, and P. Vallejos. A kalman-filtering-based approach for improving terrain mapping in off-road autonomous vehicles. *Journal of Intelligent & Robotic Systems*, 78(3-4):577–591, 2015.

[74] R. Paul and P. Newman. Fab-map 3d: Topological mapping with spatial and visual appearance. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2649–2656. IEEE, 2010.

[75] K. Peterson, J. Ziglar, and P. E. Rybski. Fast feature detection and stochastic parameter estimation of road shape using multiple lidar. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 612–619. IEEE, 2008.

[76] A. Petrovskaya and S. Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009.

[77] R. B. Potts. Some generalized order-disorder transformations. In *Mathematical proceedings of the cambridge philosophical society*, volume 48, pages 106–109. Cambridge University Press, 1952.

[78] A. Pronobis, A. Aydemir, K. Sjöö, and P. Jensfelt. Exploiting semantics in mobile robotics. In *ICRA 2012 Workshop on Semantic Perception, Mapping and Exploration*, 2012.

[79] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[80] A. L. Rankin, A. Huertas, and L. H. Matthies. Stereo-vision-based terrain mapping for off-road autonomous navigation. In *SPIE Defense, Security, and Sensing*, pages 733210–733210. International Society for Optics and Photonics, 2009.

[81] C. Rasmussen and T. Korah. On-vehicle and aerial texture analysis for vision-based desert road following. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 66–66. IEEE, 2005.

[82] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. Torr. Urban 3d semantic modelling using stereo vision. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 580–585. IEEE, 2013.

[83] S. Sengupta, P. Sturgess, P. H. Torr, et al. Automatic dense visual semantic mapping from street-level imagery. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 857–862. IEEE, 2012.

[84] L. Shapiro and G. C. Stockman. Computer vision. 2001. *ed: Prentice Hall*, 2001.

[85] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Computer Vision–ECCV 2006*, pages 1–15. Springer, 2006.

[86] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[87] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 573–580. IEEE, 2012.

[88] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[89] R. Tapia-Espinoza and M. Torres-Torriti. Robust lane sensing and departure warning under shadows and occlusions. *Sensors*, 13(3):3270–3298, 2013.

[90] S. Thrun. Google's driverless car, 2012. URL `http://www.ted.com/talks/sebastian_thrun_google_s_driverless_car.html`.

[91] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006.

[92] N. Tomatis, I. Nourbakhsh, and R. Siegwart. Hybrid simultaneous localization and map building: a natural integration of topological and metric. *Robotics and Autonomous systems*, 44(1):3–14, 2003.

[93] P. G. Trepagnier, J. Nagel, P. M. Kinney, C. Koutsougeras, and M. Dooner. Kat-5: Robust systems for autonomous vehicle navigation in challenging and unknown terrain. *Journal of Field Robotics*, 23(8):509–526, 2006.

[94] R. Triebel, H. Grimmett, R. Paul, and I. Posner. Driven learning for driving: How introspection improves semantic mapping. In *Robotics Research*, pages 449–465. Springer, 2016.

[95] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[96] C. Urmson, J. Anhalt, M. Clark, T. Galatali, J. P. Gonzalez, J. Gowdy, A. Gutierrez, S. Harbaugh, M. Johnson-Roberson, H. Kato, et al. High speed navigation of unrehearsed terrain: Red team technology for grand challenge 2004. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-04-37*, 2004.

[97] H. Wang, Y. Gong, Y. Liu, and M. Ren. Road detection via superpixels and interactive image segmentation. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on*, pages 152–155, June 2014.

[98] J. Wang, Z. Ji, and Y.-T. Su. Unstructured road detection using hybrid features. In *Machine Learning and Cybernetics, 2009 International Conference on*, volume 1, pages 482–486. IEEE, 2009.

[99] Q. Wang, J. Fang, and Y. Yuan. Adaptive road detection via context-aware label transfer. *Neurocomputing*, 158:174–183, 2015.

[100] S. Wender and K. Dietmayer. 3d vehicle detection using a laser scanner and a video camera. *IET Intelligent Transport Systems*, 2(2):105–112, 2008.

[101] W. Whittaker and L. Nastro. Utilization of position and orientation data for preplanning and real time autonomous vehicle navigation. In *2006 IEEE/ION Position, Location, And Navigation Symposium*, pages 372–377. IEEE, 2006.

[102] R. W. Wolcott and R. M. Eustice. Visual localization within lidar maps for automated urban driving. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183. IEEE, 2014.

[103] M. Xie, L. Trassoudaine, J. Alizon, M. Thonnat, and J. Gallice. Active and intelligent sensing of road obstacles: Application to the european eureka-prometheus project. In *Computer Vision, 1993. Proceedings., Fourth International Conference on*, pages 616–623. IEEE, 1993.

[104] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[105] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[106] H. Zender, P. Jensfelt, O. M. Mozos, G.-J. M. Kruijff, and W. Burgard. An integrated robotic system for spatial understanding and situated interaction in indoor environments. In *AAAI*, volume 7, pages 1584–1589, 2007.

[107] S. Zhou and K. Iagnemma. Self-supervised learning method for unstructured road detection using fuzzy support vector machines. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1183–1189. IEEE, 2010.