Detección de Líneas de Carril

Javier Gómez Domínguez

Resumen— Los Sistemas Avanzados de Asistencia a la conducción (ADAS) cada día están ganando más peso en la industria automovilística como sistemas de seguridad activa para evitar accidentes o minimizar su impacto. En este proyecto hemos adaptado un Lane Departure Warning para autopistas a un entorno de ciudad, un entorno mucho más cambiante y dinámico que la autopista. Además, hemos incluido una reconstrucción 3D en base a las imágenes tomadas con un par estéreo de cámaras para poder tener una referencia métrica que permita a un vehículo autónomo tomar decisiones en base a esta información.

Palabras clave— LDW, ADAS, Visión por Computador, Vehículo Autónomo.

Abstract— The Advanced Driver Assistant System (ADAS) progressively are gaining interest in the Automotive Industry as active safety systems to prevent traffic accidents or minimize their impact. In this project we have developed an adaptation of a Lane Departure Warning for highways to use it in roads within cities, an environment much more changing and dynamic than a highway. Furthermore, we include a 3D reconstruction based on the images captured with a stereo camera to have a metric reference that allows to an autonomous vehicle take decisions based on this information.

Index Terms— LDW, ADAS, Computer Vision, Autonomous Vehicle.

1 Introducción

Los accidentes de tráfico representan la octava causa de muerte en el mundo. Según el estudio anual de la OMS (Organización Mundial de la Salud), entre los jóvenes con edades de 15 a 29 años, los accidentes de tráfico son la primera causa de mortalidad en el mundo y la segunda causa de enfermedad y discapacidad.

Una de las causas más comunes de accidentes son las invasiones de carril. En las autopistas, debido a la monotonía de la conducción, se produce falta de atención e incluso somnolencia. Estos factores pueden provocar que el vehículo se salga de su carril y colisione con otro vehículo o con la infraestructura (Biondas, pivotes, etc).

En las ciudades, debido a las distracciones del conductor, como el manejo del GPS, teléfonos móviles, etc., el vehículo también se puede salir de su carril colisionando con otros vehículos o incluso atropellando algún peatón.

Los ADAS (Sistemas Avanzados de Asistencia a la Conducción) tratan de reducir accidentes de tráfico. Existen diferentes tipos de ADAS basados en la detección de peatones, señales, líneas de carril o sistemas de visión nocturna mejorada.

Figura 1: Detección de Líneas de Carril dentro de Ciudad

Los LDW (Lane Departure Warning) tratan de reducir los accidentes de tráfico alertando al conductor en caso de salirse del carril. Tradicionalmente los LDW se utilizan en autopistas y se basan en cámaras 2D.

- *E-mail de contacto: Javier.ggd@gmail.com*
- *Mención realizada: Computación*
- *Trabajo tutorizado por: Antonio López Peña (CVC), ADAS Group*
- *Trabajo supervisado por: David Vázquez (CVC), ADAS Group*
- *Curso 2014/15*

En este trabajo abordaremos el uso del LDW, adaptando un modelo existente pensado para autopistas a un entorno de ciudad, ver Figura 1. Además, para captar las imágenes a procesar utilizaremos cámaras estéreo.

Este proyecto forma parte de uno de los módulos principales del coche autónomo en el que está trabajando el grupo ADAS del CVC. Este proyecto se podrá utilizar en futuras aplicaciones como la detección de peatones dentro del carril, o calcular la distancia con el coche que tenemos delante y ajustar la distancia de seguridad y adaptar la velocidad a éste.

Actualmente, ya existen muchos proyectos de detección de carril, pero se basan en tecnologías antiguas y utilizan una única cámara para tomar las imágenes a procesar.

En este proyecto trabajaremos con un software que fue desarrollado por el grupo ADAS del Centro de Visión por Computador (CVC) de la Universidad Autónoma de Barcelona (UAB) [1] implementado sobre el vehículo de la figura 2. Este software, pese a su robustez, está basado en librerías antiguas y de pago, como las MIL (*Matrox Image Library*) para el tratado de imágenes o las MFC (*Microsoft Founding Class*) para la interfaz de usuario que aparte de estar en desuso actualmente, solo funcionan para Windows.

Para solventar los problemas relacionados al tratado de imagen, en lugar de una nueva versión de las MIL, utilizaremos las librerías OpenCV, que es independiente de la plataforma y está adaptada para los nuevos procesadores. Éstas son librerías open source mucho más versátiles que las MIL que nos permitirán desde el tratado de la imagen, hasta el procesamiento de la misma. Además, nos facilitará mucho el trabajo en la fase futura de reconstrucción 3D.

Por último, queremos mostrar los resultados en una visualización 3D de la escena, que nos permitirá extraer información útil de ésta en métricas válidas para un vehículo autónomo.

2 OBJETIVOS

El objetivo principal de este proyecto es la detección de líneas de carril de imágenes obtenidas por un par estéreo de cámaras adaptando un software de reconocimiento de líneas de carril en autopista para trabajar dentro de ciudad y visualizar los resultados en una reconstrucción 3D.

Para alcanzar este objetivo, hemos planteado los siguientes subobjetivos:

- **Refactorizar del software existente:**
  Esta fase se basa en la adaptación del software proporcionado por el grupo ADAS a las librerías OpenCV.

Figura 2: Vehículo Inteligente del Grupo ADAS del CVC

- **Adaptar del software al nuevo entorno dentro de ciudad:**
  La aplicación suministrada por el grupo ADAS está desarrollada para la detección de líneas de carril en autopistas. Uno de nuestros objetivos es conseguir un sistema robusto capaz de detectar las líneas que limitan un carril dentro de la ciudad y que sepa discriminar las que pertenecen a otros carriles o simplemente sean señales horizontales marcadas sobre el asfalto.

- **Reconstruir la escena 3D y estimar el plano de la carretera:**
  En esta fase hemos utilizando las imágenes de la cámara estéreo para reconstruir la escena en tres dimensiones. Con esa información hemos extraído el plano y posteriormente sobre este, hemos proyectado las líneas de carril en un entorno 3D. Esta reconstrucción nos permitirá extraer la información en unidades métricas válidas para un vehículo autónomo.

3 ESTADO DEL ARTE

Actualmente, los trabajos existentes de detección de líneas de carril están basados en imágenes obtenidas por una única cámara.

Dentro de los LDW, Iteris desarrolló el primer sistema para camiones Mercedes Actros en 2000. Nissan y Toyota fueron los primeros fabricantes en introducir este sistema dentro de unos vehículos sedán en 2001.

Otros fabricantes, como Citroen, en 2004 optaron por situar dos sensores infrarrojos en el parachoques delantero que son capaces de reconocer y seguir las líneas de la carretera, manteniendo el vehículo dentro de ellas.

De hecho, empresas de automoción como SEAT, dentro del grupo Volskwagen, ya ofrecen en el mercado modelos de vehículos con un control de crucero avanzado.

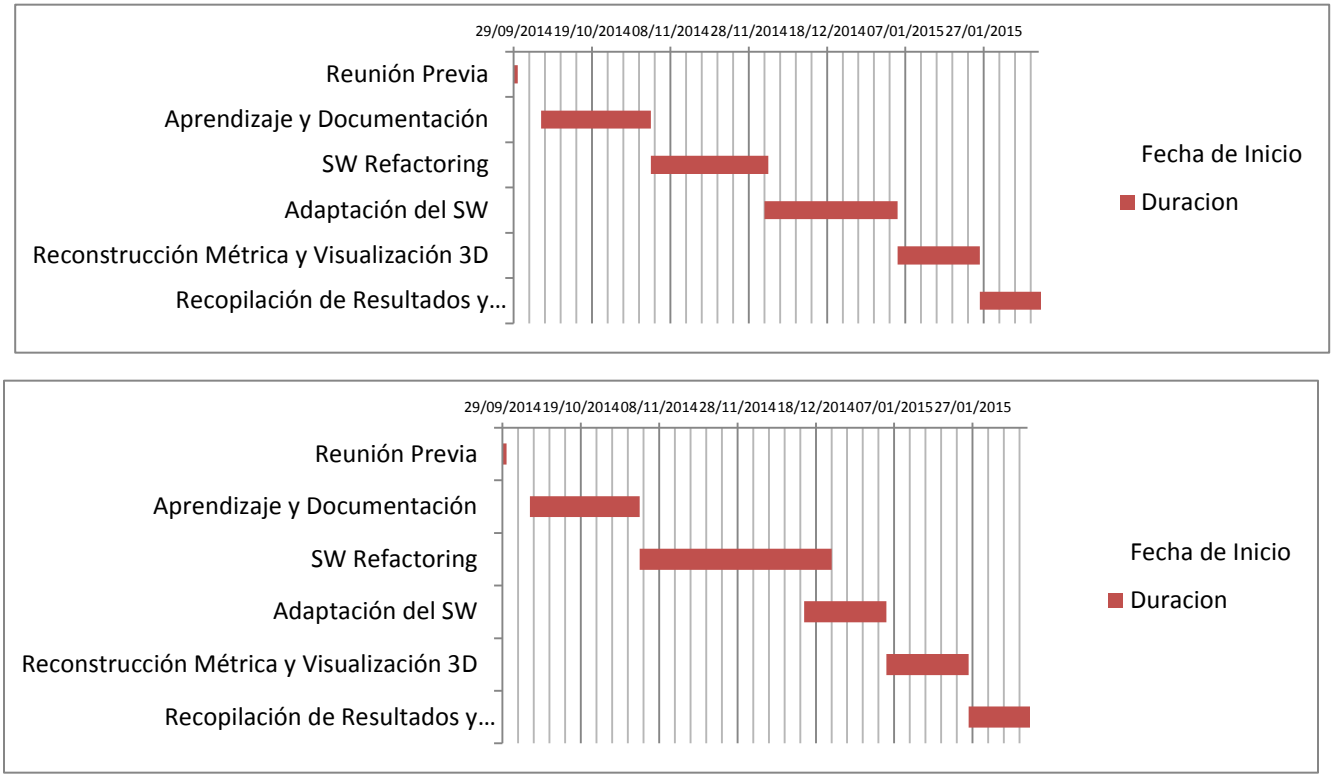


Figura 3: En la imagen superior, Diagrama de Gantt de la Planificación Inicial del Proyecto. En la imagen inferior, diagrama de la planificación final del proyecto.

Este control de crucero se basa en detectar las líneas del carril con una única cámara situada en la parte posterior del espejo retrovisor central del vehículo.

El código suministrado con el grupo ADAS del CVC se basa en la detección de líneas de carril tratando cada una como cadenas únicas [1]. Tal y como se explica en [2], dada una imagen, se filtra buscando las crestas. Posteriormente, se elimina el ruido mediante un análisis de blobs. A continuación se proyecta la imagen haciendo una homografía para obtener una vista cenital en la que se buscarán las líneas de carril basándose en la Transformada de Hough [3]. Estas líneas se seguirán utilizando un tracker y finalmente, un autómata de estados hará un razonamiento de alto nivel.

En cuanto a la reconstrucción 3D, tal y como se explica en [4], dicha reconstrucción se realizará mediante un *mapping* de las posiciones 2D de las imágenes rectificadas, para encontrar la posición de dichas coordenadas en un espacio 3D. Esto se consigue gracias al cálculo de la disparidad entre las dos imágenes tomadas, con el fin de determinar la profundidad de los objetos en la imagen rectificada 2D. Nosotros hemos aprovechado este trabajo realizado por el grupo ADAS, y hemos implementado un módulo para la proyección y visualización de las líneas de carril dentro de la reconstrucción 3D.

4 METODOLOGIA

Para este proyecto, la metodología de seguimiento ha sido basada en la metodología SCRUM.

En una primera reunión preliminar, nos hemos marcado pequeños objetivos a modos de sprint, y hemos fijado las reuniones al final de cada uno de ellos para evaluar los avances obtenidos y determinar si debíamos mantener el siguiente objetivo o rectificarlo en función de la dirección que ha ido tomando el proyecto.

Aparte de las reuniones al final de cada sprint, la comunicación ha sido constante y bidireccional vía email o pequeñas reuniones para resolver alguna duda concreta sobre el desarrollo del proyecto, tanto con mi tutor como con mi supervisor.

En una primera fase, dentro del primer objetivo de Refactoring del SW, realizamos una primera fase de aprendizaje y documentación sobre el proyecto, este sprint duró alrededor de un mes. Pese a ser una fase de documentación, tuvo una duración relativamente alta puesto que se trataba de un código totalmente desconocido, utilizando unas librerías obsoletas también desconocidas para mí. Además de lo existente en el código, nos documentamos sobre la nueva librería OpenCV que íbamos a introducir en la aplicación.

Al final del sprint organizamos una pequeña reunión para resolver cualquier duda sobre el código existente o los algoritmos que éste emplea antes de ponernos a realizar la propia adaptación del código.

Una vez entendida la lógica del código, en el siguiente período, estuvimos realizando la adaptación del código a la nueva librería OpenCV. Esta fase se alargó más de lo esperado, y tuvo una duración de un mes y medio en lugar de un mes tal y como habíamos previsto, tal y como se puede apreciar en la Figura 3. Pese a alargarse, esta fase nos ayudó a conocer el código y a proporcionarnos agilidad y dinamismo en el trabajo con éste, que nos facilitaría el avance de las fases posteriores de una manera mucho más rápida y eficiente.

Durante el siguiente período del proyecto, realizamos la adaptación del procesado de imágenes a un entorno dentro de ciudad. Esto nos ocupó alrededor de tres semanas y media. En esta fase entró tanto el calibrado de los parámetros del procesado de imagen del vídeo suministrado dentro de ciudad, como la implementación de una interfaz de usuario para visualizar los resultados de las detecciones.

En la cuarta fase del proyecto, procedimos a la reconstrucción de la escena en 3D, esta fase nos llevó dos semanas más, puesto que había que incrustar nuestro módulo de detector de líneas, dentro del código que se encargaba de realizar la reconstrucción 3D y la detección de peatones y vehículos.

Finalmente, la última fase del proyecto se basa en la fase de recopilación de resultados, evaluación y redacción del artículo final del proyecto.

5 SW REFACTORING

Esta fase corresponde al período de adaptación del código a una nueva plataforma funcional basada en OpenCV [6]. El código suministrado por el grupo ADAS del CVC se trata de un detector de líneas de carril basado en una única cámara y con librerías en desuso y de pago como ya hemos comentado anteriormente.

Previamente, tal y como hemos visto en el apartado anterior, hubo una fase inicial de aprendizaje. En esta fase hubo un periodo de documentación, basándonos en los artículos ya comentados anteriormente [1] [2] [5] con tal de entender el procesado de la imagen que realizaba el código [2], los manuales de ayuda de las librerías MIL, y la documentación de la librería OpenCV con la que íbamos a realizar la nueva implementación.

Las librerías MIL (Matrox Image Library) [7], son unas herramientas software ideadas para facilitar el trabajo de visión por computador, análisis de imágenes y aplicaciones médicas de *imaging software*. La versión de las MIL que nos encontramos fue la 7.0, una versión lanzada en 2001. Dicha versión actualmente no funciona en versiones recientes de Windows ni de Visual Studio, por lo tanto, no podíamos compilar y ejecutar nuestra aplicación para ver el flujo de ejecución.

En nuestro código, la funcionalidad de las MIL residía en el manejo de los buffers que contenían las imágenes y en el control de las salidas de imagen que iban a aparecer por pantalla. Con esta librería se puede desde superponer una imagen sobre otra, hasta combinar varias imágenes para mostrarlas a la vez en una sola visualización.

Gracias a los manuales de las MIL, tuvimos acceso a cada una de las funciones que utilizaba la aplicación, a sus variables de entrada y de salida, cosa que nos facilitó mucho el proceso posterior de adaptación del código.

Una vez analizadas las necesidades del código respecto a las MIL, procedimos a documentarnos sobre la librería OpenCV. Básicamente buscamos métodos y funciones equivalentes a los de las MIL, incluso en algunos casos mejorando la eficiencia de las funciones. Como ya hemos comentado, la mayoría de los métodos a sustituir eran del tratado de la imagen y la visualización de resultados. Además, OpenCV nos permite dibujar directamente sobre la imagen, mientras que con las MIL se tenía que trabajar con dos imágenes y finalmente superponer una a la otra. Esto nos permite trabajar con la mitad de buffers en OpenCV, suponiendo un ahorro considerable de los recursos a utilizar en la aplicación.

Una vez finalizada la fase de documentación, nos pusimos con el desarrollo de la adaptación del software. Primero de todo, con el análisis del diagrama UML pudimos identificar cuatro módulos principales: Adquisición, Visualización, Procesado de la Imagen e Interfaz de Usuario.

La fase de *refactoring* no ha sido fácil, como hemos comentado anteriormente se nos alargó más de lo estimado previamente. El proyecto tiene una cantidad muy elevada de clases. La mayor parte de ellas estaban diseñadas en base a las librerías MIL o MFC. La tabla 1 nos puede ayudar a imaginarnos la magnitud de este *refactoring*.

Tipo	**Numero de Clases**	**Refactorizadas**
Procesado de Imágenes	25	14
Visualización	12	9
Adquisición	2	2
Interfaz de Usuario	16	4
Total	55	26

Tabla 1: Tabla con el número de clases

La primera decisión que tomamos fue eliminar la interfaz de usuario existente por completo, salvando exclusivamente el módulo de guardado de las imágenes. Esto se debe a que estaba implementada con MFC (Microsoft

Foundation Class), unas librerías que actualmente están en desuso y solo funcionan con Windows. Además, la aplicación está pensada para que trabaje en tiempo real dentro de un coche, así que no necesitamos ningún tipo de interfaz de usuario. Únicamente realizamos una sencilla interfaz dentro del módulo de Visualización para poder visualizar los resultados intermedios durante el desarrollo de la aplicación.

Para las siguientes adaptaciones, realizamos un paso intermedio. Para poder compilar y ejecutar el código implementamos una clase Wrapper con los métodos de las MIL que utiliza la aplicación con la finalidad de que el código no echara en falta las MIL. Una vez realizado el Wrapper, procedimos a sustituir las llamadas de cada una de las funciones del Wrapper por funciones de las librerías OpenCV.

Dentro del módulo de Adquisición y Visualización, hemos modificado todos los tipos de los buffers de las imágenes a tipos propios de las OpenCV, y hemos utilizado los propios métodos de la librería para dibujar las líneas detectadas directamente sobre la imagen, en lugar de almacenarlo en otro buffer y superponer una sobre la otra a la hora de visualizarlo como hacían las MIL. Este proceso nos permite ahorrar memoria y recursos de la aplicación, aumentando la eficiencia de este mismo.

La adaptación del módulo de procesado de la imagen no sufrió grandes modificaciones, simplemente adaptar las entradas y salidas de los métodos y funciones en base a los nuevos tipos. Además, eliminamos todas las funciones relacionadas con la combinación de los buffers de superposición que hemos explicado anteriormente.

6 Image Processing

Una vez finalizado el refactoring de la aplicación, y teniéndola completamente funcional, procedimos a implementar la detección de líneas de carril en un entorno dentro de ciudad.

Las imágenes a procesar provienen de la rectificación de las imágenes obtenidas por el par estéreo del vehículo.

El procesado de las imágenes para la detección de líneas de carril consta de cuatro fases principales: Detección de las Crestas, filtrado de las Crestas mediante Blob Análisis, Homografía y Transformada de Hough.

La primera fase, la detección de crestas, se basa en el análisis píxel a píxel de la imagen, teniendo en cuenta los píxeles vecinos. Este proceso realiza una reconstrucción tridimensional en base a la intensidad de la claridad de cada píxel y sus vecinos, tal y como podemos ver en la Figura 5.

La segunda fase del procesado corresponde a la suavización de las Crestas mediante Blob Análisis. Este análisis se dedica a descartar *outliers* en base a unas características

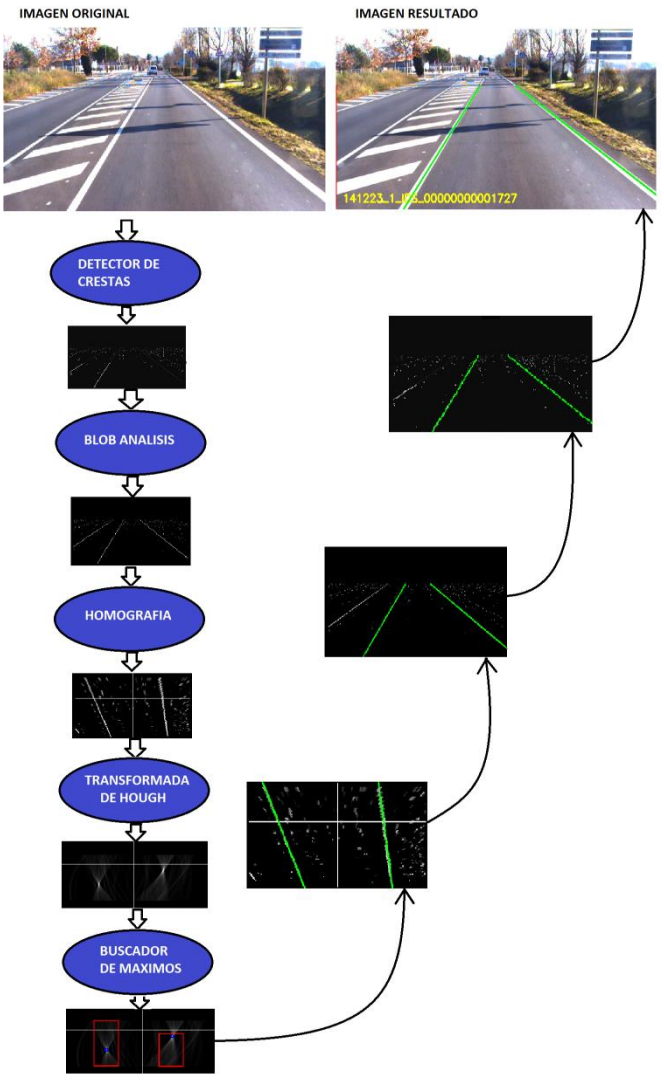


Figura 4: Diagrama de Flujo del procesado de la imagen. En el apéndice A podemos ver con más detalle el procesado de la imagen.

predefinidas de las líneas de carril. Por ejemplo el grado de inclinación de las líneas, jamás serán totalmente horizontales o verticales, por lo tanto, mediante este análisis descartaríamos los árboles o las líneas horizontales de las señales pintadas en el asfalto.

En la siguiente fase, se aplica una homografía sobre la imagen procesada con la finalidad de tener una vista cenital del plano de la carretera en la que las líneas se muestran paralelas. Además, este procesado limita el área de la imagen de búsqueda de las líneas de carril al plano de la carretera, obviando el resto del entorno, puesto que no nos interesa para nuestra finalidad, aumentando así la velocidad de procesado y eliminando posibles *outliers*.

Para poder realizar la homografía en base al punto de fuga, se genera un trapezoide que se utiliza para estimar el plano de la carretera y la línea del horizonte desde la cual se va a proceder a analizar la detección de las líneas.

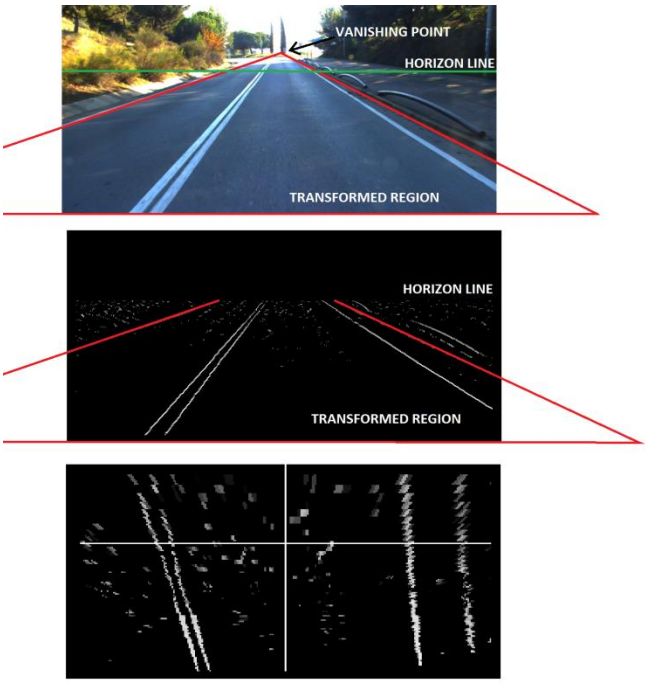


Figura 5: Estimación del punto de fuga en base a líneas paralelas (imagen superior), salida del detector de crestas filtrado mediante Blob Análisis (imagen central) y homografía del área del trapezoide sobre la cual se va a realizar la detección de líneas de carril (imagen inferior).

Como se puede apreciar en la Figura 5, el trapezoide generado procede de las rectas paralelas que se cruzan en el infinito, la línea horizontal que hemos fijado por debajo del punto de fuga a modo de horizonte y el margen inferior de la imagen. El valor del margen inferior de la imagen puede no limitarse al tamaño horizontal de ésta, debe ser un valor superior, puesto que puede darse el caso de que las líneas de carril desaparezcan por un lateral de la imagen y no por la base de esta.

En la última fase, se aplica una transformada de Hough [8] sobre la homografía generada. La transformada de Hough detecta las líneas encontradas dentro de las dos ROI (Una por cada lado del carril) y busca los máximos dentro de las dos ROI para determinar qué líneas son las correctas dentro de cada región. Tal y como podemos apreciar en la figura 6, la transformada de Hough explora dentro de la ROI (recuadro rojo) las líneas detectadas dentro de la homografía, y procede a buscar el máximo local dentro de la ROI (recuadro azul), asumiendo así que esa es la línea que corresponde a ese lado del carril.

Una vez seleccionados los máximos, se transfieren las líneas detectadas mediante una regresión sobre el procesado realizado capa a capa, obteniendo las líneas detectadas en la imagen original tal y como se aprecia en la figura 4.

Ahora que ya conocemos el funcionamiento para la detección de una línea de carril, debemos hablar de los parámetros que tuvimos que modificar para adaptar la aplicación a un entorno dentro de ciudad.

Esta aplicación estaba configurada para detectar las líneas de carril de autopistas y carreteras fuera de ciudad. Esto quiere decir que sus parámetros estaban configurados para carriles amplios y de ancho fijo, de líneas continuas o discontinuas muy largas y amplias.

Dentro de ciudad, las líneas discontinuas son más cortas y estrechas, los carriles más estrechos y variables y suelen estar más deteriorados por la afluencia continua del tráfico.

En primer lugar, cambiamos la configuración de los parámetros de detección al nivel más bajo, es decir, la detección de crestas. En primer lugar, la estimación del ancho mínimo y máximo de píxeles se tuvo que reducir. Para decidir los valores óptimos nos basamos en las medidas de las líneas de carril de las imágenes tomadas dentro de la ciudad. Esta modificación nos permite descartar muchos objetos de los laterales del carril o señales pintadas dentro del asfalto como stops o ceda el paso. El mismo proceso es el que seguimos para determinar la longitud mínima de una línea de carril, ésta debe ser bastante más baja que en autopista o carretera.

Una vez ajustados los parámetros de las crestas, tuvimos que modificar los de la Homografía. Modificamos el punto de fuga en base a la configuración de nuestras cámaras. Fijamos la fila inicial, fijada en el horizonte desde donde empezaremos a procesar la imagen, en base al punto de fuga. Este valor siempre estará situado por debajo del punto de fuga y se debe estimar un valor que coincida con el horizonte. Por último, fijamos el valor del margen inferior del trapezoide a un valor más amplio que el de la imagen, para poder controlar las salidas de las

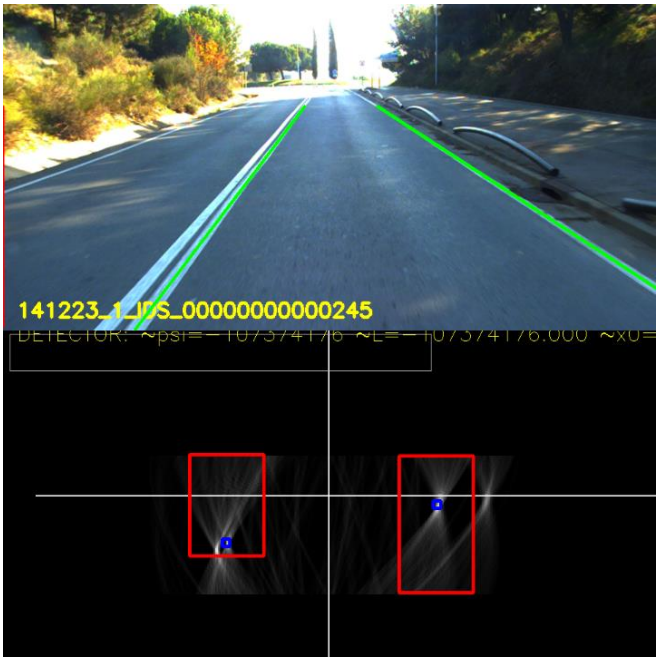


Figura 6: Visualización de Transformada de Hough en base al área de la homografía (imagen inferior) y Líneas detectadas como máximos dentro de cada ROI (imagen superior)

líneas de carril por el lateral de la imagen en lugar del inferior de ésta. El punto de fuga se sitúa siempre en el centro del margen inferior del trapezoide.

En la cuanto a los parámetros de la transformada de Hough, se ha limitado el ángulo de las líneas para descartar todas las detecciones completamente verticales y evitar así la detección de cualquier objeto vertical o árbol. Las líneas detectadas, como podemos apreciar en la Figura 7, se encuentran dentro de la ROI y nos encuentra el máximo dentro de esta.

Básicamente, estos son los cambios más significativos realizados para adaptar el código a un entorno dentro de ciudad. El resto de ajustes se han realizado a base de prueba y error para conseguir los mejores resultados posibles.

7 Reconstrucción Métrica y visualización 3D

Tal y como se explica en [4], un entorno 3D beneficia al detector por el conocimiento que se puede extraer de éste. La distancia es una de las informaciones más importantes que nos aporta el 3D. Podemos predecir cuándo se nos va a acabar un carril o que distancia nos encontraremos con una curva o una rotonda. En nuestro caso, el 3D nos permitirá extraer el plano de la carretera para poder realizar la proyección de las líneas sobre éste.

En el trabajo realizado por el grupo ADAS, para realizar la reconstrucción 3D se utilizan las imágenes obtenidas por ambas cámaras, se rectifican y se busca la disparidad entre estas. La rectificación de la imagen es necesaria para facilitar el proceso de relacionar donde se sitúan los puntos de una imagen respecto a la otra. Para buscar la disparidad, se recorre la imagen 2D buscando los desplazamientos horizontales entre ellas. Una vez encontrados los desplazamientos de cada píxel respecto a su homónimo de la otra imagen, se realiza la imagen de disparidad.

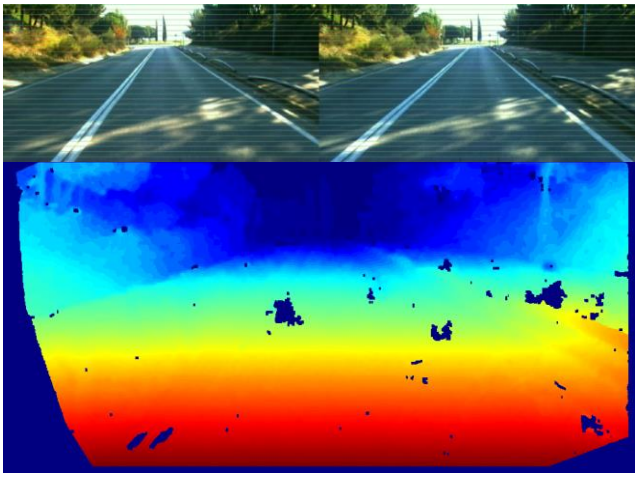


Figura 8: Imágenes de rectificación para calcular la disparidad (imagen superior) e imagen de disparidad a color (imagen inferior).

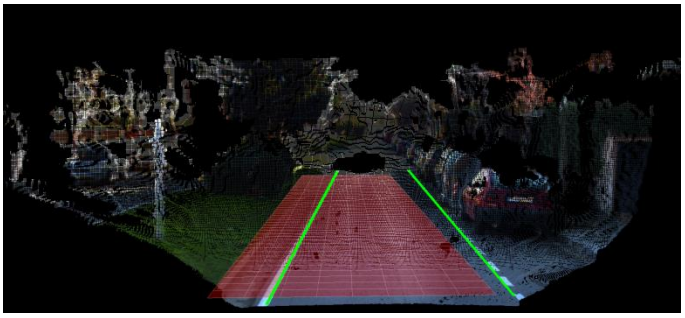


Figura 9: Reconstrucción de la escena en 3D.

Una vez calculada la imagen de disparidad y con las fórmulas explicadas en [4], se realiza la reconstrucción a 3D en base a la imagen de disparidad. Como se puede apreciar en la figura 8, si algún píxel no encuentra su disparidad, éste no está representado en la reconstrucción 3D.

Una vez tenemos realizada la reconstrucción 3D, se procede a calcular la estimación del plano de la carretera sobre el cual recaerán las líneas de carril. Para encontrar el plano, también se basa en la imagen de disparidad. En [5] se explica cómo se estima el plano en base al cálculo de la profundidad y la altura de los objetos que se sitúan en la imagen, se aplica una transformada de Hough sobre la *v-disaprity* y se estima el plano de la carretera.

Una vez tenemos estimado el plano, hemos añadido un módulo llamado *Lane Marking Manager*, que se encarga de proyectar sobre el plano las líneas detectadas.

En este módulo que hemos incrustado se utilizan las coordenadas de las líneas obtenidas en 2D durante el procesado de detección de la imagen, y se transfieren a coordenadas 3D en base a la estimación del plano obtenida en el módulo anterior. Las coordenadas se pasan de coordenadas cámara a coordenadas mundo para poder proyectar las imágenes sobre el plano de la reconstrucción 3D.

Cómo podemos apreciar en la Figura 9, se han proyectado las líneas en 3D sobre el plano detectado de la imagen. También podemos apreciar que hay trozos de la reconstrucción vacíos, este fenómeno se debe a que no se ha encontrado la disparidad entre las dos imágenes iniciales.

8 Resultados obtenidos

En primer lugar, hemos conseguido adaptar el código a las nuevas librerías, obteniendo los resultados que se obtenían con las librerías anteriores en autopistas pero en un sistema moderno y en tiempo real. En la figura 10, podemos observar cómo se detectan las líneas de carril con una cámara monocromática, con el sistema actual.

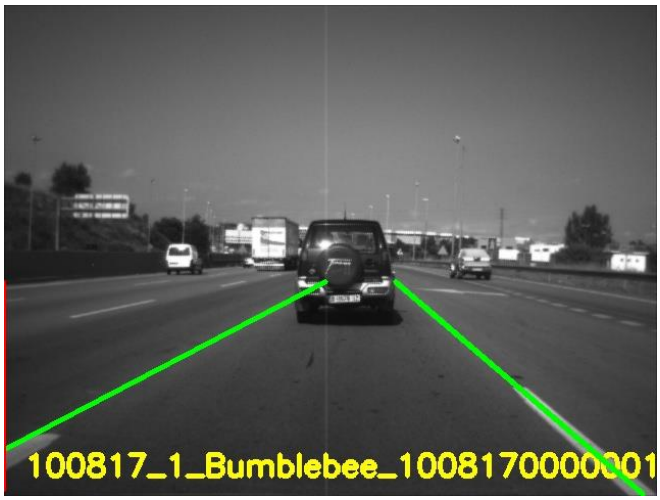


Figura 10: Detección de líneas de carril con una cámara.

Después de conseguir que el código funcionase en 2D con una cámara monocromática con las nuevas librerías, realizamos el procesado de la imagen y la detección de líneas de carril con imágenes a color obtenidas de cámaras 3D. Tal y como podemos apreciar en la figura 11, el modelo se adapta a carreteras más amplias de fuera de ciudad y a carriles más estrechos dentro de la ciudad. Adaptando los parámetros explicados anteriormente, podemos apreciar como la aplicación es capaz de detectar líneas de carriles con los distintos anchos que nos podemos encontrar dentro de una ciudad.

Hemos conseguido la visualización de la escena 3D, tal y como se puede apreciar en la Figura 9, la reconstrucción se realiza en base a la imagen de disparidad, se calcula el plano del a imagen y finalmente se proyectan las líneas detectadas sobre la reconstrucción. Además se nos permi-

Figura 11: En la imagen superior, detección de líneas de carril en carreteras fuera de ciudad. En la imagen inferior, detección de líneas carril dentro de ciudad.

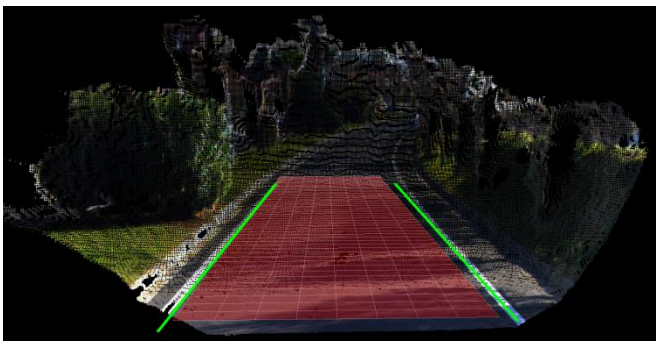


Figura 12: Reconstrucción 3D con la estimación del plano y las líneas proyectadas sobre éste. Escena visualizada desde otro punto de vista.

te navegar por la reconstrucción cambiando el punto de vista alejándonos o acercándonos de la escena, tal y como podemos ver en la Figura 12.

9 Conclusión

En este proyecto se ha realizado la actualización de un *framework* detector de líneas de carril del estado del arte en una plataforma moderna y abierta, adaptando el mismo a distintos entornos urbanos.

Además, se ha incorporado a este *framework* un sistema de reconstrucción 3D con una estimación del plano dominante. Esto posibilitará la derivación de información métrica útil para extender y robustificar otros procesos ADAS tales como la detección de peatones, mantener la distancia de seguridad en la carretera, etc.

Nuestro análisis cualitativo sobre secuencias urbanas capturadas desde nuestro vehículo inteligente prueba la precisión y la robustez de nuestro *framework* en distintas condiciones.

Esta motivación nos ha llevado a seguir trabajando en el proyecto en prácticas externas con el grupo ADAS del CVC. Se robustificará la detección de líneas de carril mediante la inclusión de la información 3D. Para ello, se planea usar la estimación del plano para eliminar los *outliers* verticales simplificando así considerablemente el proceso.

Los pasos dados en este proyecto nos acercan a la futura realidad de los vehículos autónomos confiables.

Agradecimientos

Gracias a Antonio López y a David Vázquez del grupo ADAS del CVC, por toda la confianza depositada, el asesoramiento y la ayuda prestada durante toda la duración de este proyecto. Sin ellos, este proyecto no hubiese sido posible.

BIBLIOGRAFÍA

[1] López, A., Serrat, J., Saludes, J., Cañero C., Lumbreras F., Graf T. *Ridgeness for Detecting Lane Markings*

[2] López, A., Serrat, J., Saludes, J., Cañero C., Lumbreras F. *Robus Lane Lines detection and quantitative assessment.*

[3] Richard O. Duda, Peter E. Hart. *Use of the Hough Transformation To Detect Lines and Curves in Picture.*

[4] Villalonga, G., López, A. *Estimación de la carretera y detección de peatones en 3D.*

[5] López, A., Serrat, J., Saludes, J., Cañero C., Lumbreras F., Graf T. *Robust Lane Markings Detection And Road Geometry Computation*

[6] OpenCV Documentation
http://docs.opencv.org/

[7] Matrox Image Library Documentation
http://www.matrox.com/imaging/media/pdf/products/mil/b_mil.pdf

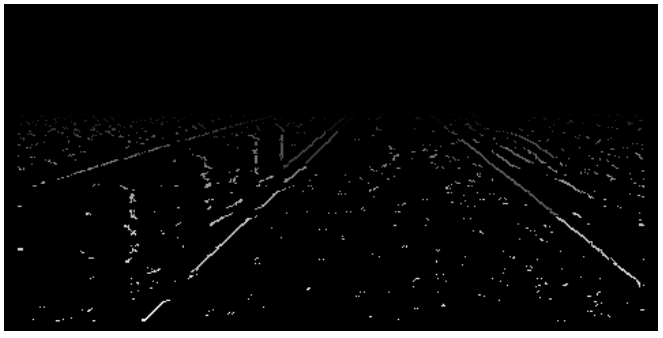
[8] Hough, P.V.C. *Method and means for recognizing complex patterns. U.S. Patent 3,069,654, Dec. 18, 1962.*

APENDICE

A1. APÉNDICE A

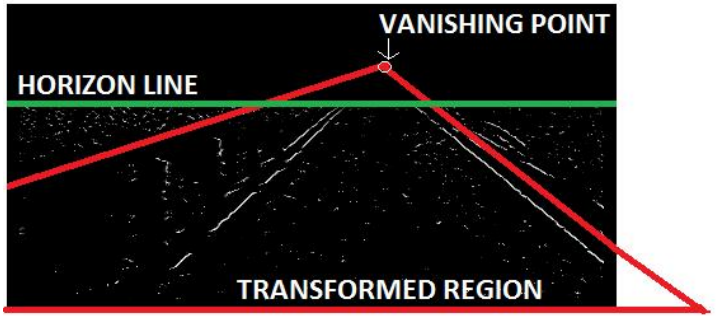
En este apartado queremos mostrar en detalle el proceso paso a paso del procesado de una imagen. La imagen que se muestra a continuación es la imagen desde la cual se inicia el procesado de la imagen. Esta imagen se obtiene de rectificar la imagen en base a las dos imágenes capturadas por el par estéreo de cámaras.

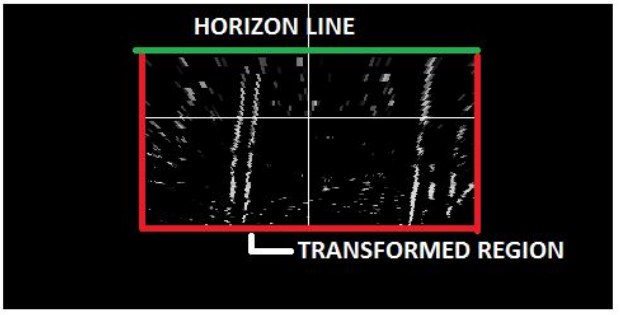
Esta imagen, pasa al detector de crestas:



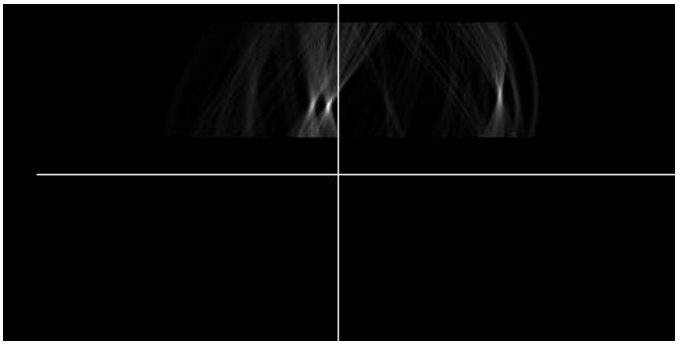
Y se filtra mediante Blob análisis, eliminando gran parte del ruido de líneas verticales:

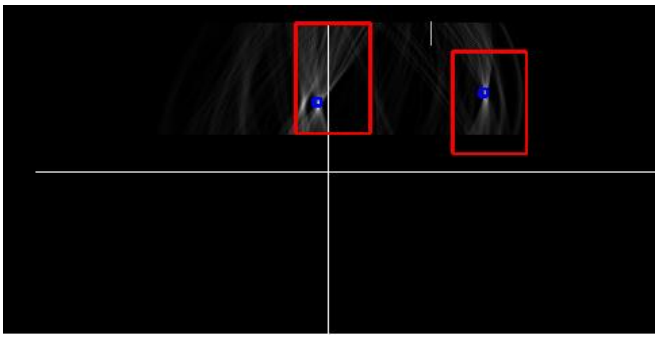
A partir la imagen filtrada por blob análisis y con la información que tenemos sobre el punto fuga, limitamos la región a transformar para realizar la homografía. En las siguientes imágenes se muestra la lógica seguida para realizar la homografía paso a paso.





A través de la homografía, se realiza una transformada de Hough tal y como se muestra en la imagen a continuación. La transformada de Hough se procesa con el buscador de máximos. Este buscador de máximos encuentra la línea con el valor máximo detectado dentro de cada ROI del espacio de la transformada de Hough. En las dos imágenes mostradas a continuación se puede observar la transformada de Hough generada a partir de la homografía y los máximos detectados dentro de cada ROI dentro de la transformada de Hough.





Una vez detectadas las líneas, se realiza una transformación inversa al flujo del procesado de la imagen, transfiriendo las coordenadas de las líneas detectadas capa a capa hasta encontrar las coordenadas en la imagen original sobre la cual mostraremos los resultados: