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

DETECCIÓN DE CALZADA PARA UN VEHÍCULO AUTÓNOMO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRICISTA

FERNANDO JAVIER BERNUY BAHAMONDEZ

PROFESOR GUÍA:
JAVIER RUIZ DEL SOLAR SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
HÉCTOR MILER AGUSTO ALEGRIA
SEBASTIÁN ISAO PARRA TSUNEKAWA

SANTIAGO DE CHILE
JUNIO 2011

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRICISTA
POR: FERNANDO BERNUY B.
FECHA: 05/08/2011
PROF. GUIA: JAVIER RUIZ DEL SOLAR

"DETECCIÓN DE CALZADA PARA UN VEHÍCULO AUTÓNOMO"

El objetivo de este trabajo de memoria es diseñar e implementar un sistema de visión para el reconocimiento de caminos no pavimentados para la conducción autónoma de un automóvil.

Este trabajo está inmerso en el proyecto *Vehículo Autónomo*, del *Advanced Mining Technology Center*, que tiene como objetivo la construcción de un vehículo capaz de ser conducido por caminos no pavimentados de manera autónoma.

Tras la revisión del estado del arte, se diseña una solución para el problema que contempla una segmentación adaptativa de calzada basada en color y una detección geométrica del camino, basada en el algoritmo RANSAC. Para representar el color del camino se utiliza un histograma en el espacio RGB, el cual es actualizado en cada imagen a partir de una región que se establece previamente como camino, y que es utilizado por un clasificador bayesiano para determinar si un pixel dado corresponde o no a camino. El modelo geométrico se define a partir de los bordes del camino y los aproxima por líneas rectas. Para verificar que las líneas corresponden a un modelo válido de camino se exige que el modelo final del camino cumpla con un conjunto de reglas.

Para obtener una medición cuantitativa del desempeño del sistema, se genera una base de datos de imágenes de caminos, con respecto a la cual se compara el sistema implementado. Los resultados de estos experimentos muestran que el sistema tiene una tasa de detección de un 90%, que la retroalimentación obtenida de la detección geométrica de camino mejora el rendimiento y que el sistema puede ser utilizado para toma de decisiones, ya que los resultados de esta toma de decisiones son similares a los obtenidos considerando una segmentación perfecta.

Se concluye que el diseño generado soluciona efectivamente los requerimientos exigidos para esta aplicación, entregando al sistema de conducción autónomo una medida confiable de la ubicación y forma del camino frente al vehículo.

Índice

Capítulo 1

Introducción

1.1. Motivación

El tema de esta memoria es el diseño y la implementación de un sistema de detección de calzada en caminos no pavimentados para su uso en aplicaciones de robótica de campo, y se desarrolla en el contexto del proyecto *Vehículo Autónomo*, del *Advanced Mining Tecnology Center* (AMTC), que busca construir un robot capaz de manejarse de manera autónoma en ambientes principalmente desérticos y con caminos no pavimentados.

Es sabido que los seres humanos no son buenos en tareas repetitivas que pueden ser peligrosas, debido a que con el tiempo toman excesiva confianza y se pasan por alto los protocolos de seguridad. Es por esto que pese a las exigentes medidas de seguridad en la industria minera, siguen habiendo accidentes vehiculares evitables. Este proyecto busca reemplazar a largo plazo la conducción humana de vehículos de gran envergadura asociados a la minería, y así reducir los accidentes y las pérdidas humanas relacionadas.

El vehículo elegido para ser automatizado es un Volkswagen Tiguan, sobre el cual se montarán una amplia gama de sensores que le permitan medir su entorno. Entre ellos está considerada una cámara frontal a color para la detección de camino.

Los sensores instalados permiten observar el entorno a distancias cortas, por lo que el vehículo está obligado a no superar los 30 Km/h. Para solucionar esto se considera la utilización del detector visual de calzada, y así lograr obtener una mayor extensión del mapa local

que permita asegurar que el vehículo tendrá un camino seguro para avanzar en caso de ir a velocidades superiores a 30 Km/h. Es de vital importancia que el sistema sea robusto y confiable, puesto que basado en éste se sobrepasa la velocidad en la que el tiempo de frenado es suficiente para evitar accidentes, y que funcione en tiempo real, para asegurar un tiempo de reacción adecuado. Este trabajo presenta una solución diseñada para resolver este problema.

1.1.1. Proyecto vehículo autónomo

La industria asociada a la extracción de recursos naturales es uno de los pilares fundamentales del desarrollo de la economía de Chile, situación que no se espera que cambie de manera significativa al menos en un horizonte a corto o mediano plazo. En este sentido, es especialmente importante para buscar formas de incorporar nuevas tecnologías que ayudan no sólo a mejorar la productividad y la eficiencia, sino también la seguridad de estas actividades, y hacerlas más amigables con el medio ambiente. El proyecto de Robotización de Maquinaria Minera Móvil tiene un impacto directo al garantizar la viabilidad económica de nuevas iniciativas en las áreas antes mencionadas, enfocándose y buscando soluciones tecnológicas encaminadas a la reducción de costos y a asegurar la continuidad del negocio mediante la aplicación de procedimientos de mantenimiento preventivo, la aplicación de las políticas de seguridad, y apoyo a la gestión operativa, mejorando así la eficiencia del sector de recursos naturales. Todo lo anterior es de especial importancia en áreas tales como la minería, donde el estado del arte ofrece un importante margen para la mejora rápida de las tecnologías existentes.

El objetivo general de este proyecto es desarrollar una iniciativa de investigación y desarrollo enfocada a la automatización de un vehículo, que considera las siguientes etapas: (i) Herramientas de asistencia al operador (independientes del fabricante) (1er-3er año) (ii) Control remoto y tele-operación (4to-6to año), y (iii) Operación autónoma (7-10 año). El plan de desarrollo incluye la integración de estas herramientas con los sistemas de información y redes de datos, y la operación simultánea y coordinada de múltiples vehículos operados por el hombre, semi-autónomos y autónomos. Algunas de las tecnologías a ser consideradas comprenden cámaras 2D y 3D, radar, sonar y sensores láser, RFID y redes de sensores, dispositivos fotónicos, redes de alta velocidad de datos, y realidad aumentada. También se espera que los esfuerzos de investigación asociados a las etapas antes mencio-

nadas contribuyan a desarrollar tecnologías de redes de sensores e información tales como análisis de confiabilidad y supervivencia, biometría, e interfaces hombre-robot (HRI), vigilancia y análisis de comportamiento.

1.2. Objetivos

1.2.1. Objetivos generales

El objetivo del trabajo a realizar es diseñar e implementar un sistema de visión que detecte la calzada de un camino no pavimentado en imágenes tomadas por una cámara con orientación frontal. La información entregada por este sistema será utilizada para la conducción autónoma del vehículo, por lo que es necesario que cumpla con los siguientes requisitos:

- Ser adaptativo a distintas condiciones de la calzada y ambientales.
- Funcionar en tiempo real.
- Ser confiable para los requisitos de una conducción autónoma veloz y segura.

1.2.2. Objetivos específicos

Para el trabajo de memoria se espera conseguir un sistema capaz de detectar efectivamente la calzada sobre la cual está viajando un vehículo autónomo a partir de las imágenes de la cámara con orientación frontal. Para ello es necesario cumplir con una serie de objetivos e hitos previos:

1. Investigación del estado del arte: Como paso previo al inicio de cualquier diseño es necesario recolectar conocimientos sobre el estado del arte. El objetivo es adquirir conocimientos suficientes para diseñar y planificar la implementación del sistema.
2. Diseño e implementación de sistemas de detección: A partir de la investigación hecha, se diseñarán e implementarán tres métodos distintos de detección de calzada.

3. Comparación de sistemas implementados: Se diseñará una batería de experimentos repetibles que permitan evaluar el rendimiento de los distintos sistemas implementados, y así tomar una decisión justificada sobre cuál utilizar.

4. Integración al vehículo: Se integrará el trabajo realizado en el sistema de medición del controlador del vehículo permitiendo su utilización en la conducción autónoma.

5. Generar documentación: Con el fin de permitir trabajos posteriores relacionados con el tema, se entregará una documentación con la información necesaria sobre el trabajo realizado.

1.3. Estructura de la memoria

En el capítulo 2 se introducirán los conceptos necesarios para comprender los aspectos técnicos referidos al trabajo realizado. En el capítulo 3 se presenta la implementación lograda y se explica en detalle las distintas etapas del sistema. Luego, en el capítulo 4 se presentan los experimentos realizados para medir el desempeño de la solución y sus resultados. Y finalmente en el capítulo 5 se presentan el análisis y las conclusiones del trabajo realizado.

Capítulo 2

Contexto

2.1. Definición del problema

Se requiere obtener una detección confiable del camino a distancias mayores que 20 metros para que una etapa posterior de procesamiento pueda definir una ruta manejable velocidades mayores que $35[Km/h]$ para un vehículo autónomo.

Es posible asumir una región fija de la imagen como parte del camino para su utilización en el entrenamiento en línea del sistema. Esta región será reemplazada para la detección del camino generada por otros sensores una vez que estos sistemas sean implementados.

El sistema debe funcionar en caminos no pavimentados, y debe ser capaz de adaptarse a nuevas definiciones de terreno y condiciones ambientales. No existe una definición previa de la ruta ni el tipo de suelo de las rutas que se seguirán, por lo que no es posible considerar una etapa de entrenamiento previo para el tipo de terreno.

La cámara estará situada en la parte superior del vehículo, orientada hacia el frente y sujetada a una parrilla especialmente diseñada para los sensores (ver figura 2.1).

Para que el sistema sea considerado seguro también es necesario que funcione en tiempo real, es decir que procese al menos 15 imágenes por segundo.

Este sistema será implementado en un computador de escritorio estándar, con sistema

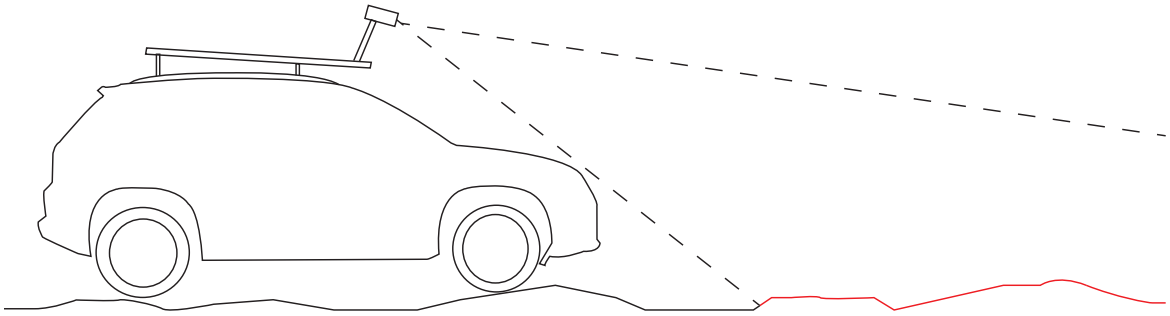


Figura 2.1: Diagrama de la posición de la cámara en el vehículo.

operativo Linux, por lo que la implementación debe ser compatible con estas condiciones.

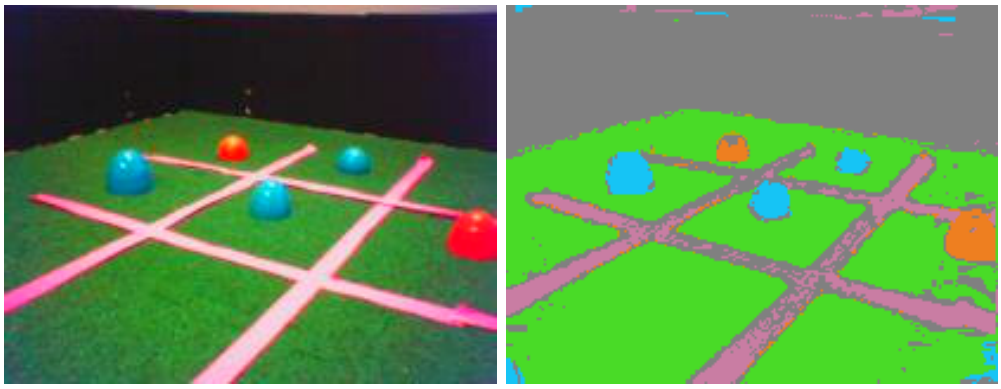
Muchos estudios han apuntado al tema de la segmentación de caminos no pavimentados [3, 10–14]. Algunos de ellos usan definiciones de camino pre-entrenadas, obteniendo buenos resultados en caminos entrenados o controlados [12], pero no son capaces de responder ante condiciones ambientales o nuevos escenarios. Otros metodos usan segmentacion adaptativa, permitiendo al sistema responder adecuadamente ante cambios de iluminacion y tipos de camino [3, 14]. Por otro lado, el enfoque geométrico permite un mejor desempeño en la segmentación cuando se incluye información como los bordes del camino o el punto de fuga, pero estos trabajos han sido diseñados para ser usados en caminos pavimentados [11, 14], y no funcionan adecuadamente fuera de estos. Otra solución al uso de información geométrica sugiere el uso de filtros de Gabor para determinar tanto los bordes del camino como el punto de fuga [10], pero el tiempo de procesamiento asociado requerido no cumple con los requisitos de funcionamiento en tiempo real de una aplicación de conducción autónoma.

2.2. Segmentación de color

El proceso de segmentación de colores corresponde a la habilidad innata de los seres humanos de reconocer objetos a partir de sus colores.

En un computador, una imagen es representada como un conjunto ordenado de unidades discretas elementales llamadas *pixeles*. Cada *pixel* contiene la información de su color el cual se representa a través de tres valores, por ejemplo la intensidad de rojo, azul y verde.

El objetivo de la segmentación de colores, desde un punto de vista computacional, es clasificar cada *pixel* de la imagen en un número discreto de clases de color. La cantidad y definición de estas clases y el clasificador a usar dependen de la aplicación. Un ejemplo es el fútbol robótico [17], donde se desea detectar objetos de colores predefinidos, es decir que se conoce la cantidad de clases y su definición a priori. En el caso de esta aplicación se utiliza como clases al camino y al resto (o no-camino). En ambas clases se desconoce su definición y esta cambia en el tiempo. En la figura 2.2 se observa el resultado de la segmentación de colores usando cuatro clases: rojo, azul, verde y rosado.



(a) Imagen original. (b) Imagen segmentada.

Figura 2.2: Segmentación de colores verde, naranjo, azul y rosado.

Normalmente se desconose la definicion de las clases de color, por lo que es necesario construir una representacion de ellas utilizando un conjunto de entrenamiento. Para ello existen diversos tipos de representaciones estadísticas. Las utilizadas en este trabajo son a través de gausianas y a través de histogramas. El primero busca aproximar la probabilidad de que un color pertenezca a una clase, mientras que el segundo aproxima la verosimilitud de la clase.

2.2.1. Representación de clases a través de gaussianas

Este método busca representar la definición de una clase de color a través de una función gaussiana en el espacio de colores, utilizando como referencia los valores de la media y la varianza de los *pixeles* de un conjunto de entrenamiento de la clase.

Se requiere tener una alta cantidad de *pixeles* que representen la clase que se desea segmentar de manera que los estimadores sean representativos del conjunto. Sea el conjunto P_t un conjunto con N pixeles de la clase deseada, luego el estimador de la media se calcula:

$$\hat{\mu}_t = \frac{1}{N} \sum_{i \in N} p_i \tag{2.1}$$

Y la matriz de covarianza:

$$\sum = \frac{1}{N} \sum_{i \in N} (p_i - \hat{\mu}_t)(p_i - \hat{\mu}_t)^T \tag{2.2}$$

A partir de estos estimadores se obtiene la función gaussiana:

$$f(p) = \frac{1}{((2\pi^{3/2} det(\sum^{1/2}))} \exp\left(-\frac{1}{2}(p - \hat{\mu}_t)^T \sum^{-1} (p - \hat{\mu}_t)\right) \tag{2.3}$$

Luego para medir el grado de pertenencia de un pixel a la clase se evalúa en la función gaussiana.

El método descrito supone que lo que la clase puede ser representada en el espacio de colores por una gausiana, por lo tanto este sistema entregará buenos resultados cuando la clase sea de un color uniforme.

Una aplicación adecuada de este método es la de detección de objetos coloreados especialmente para ser ubicados, como los faros instalados en las canchas de fútbol robótico que permiten a los jugadores conocer su ubicación. Sin embargo si se desea reconocer una textura dada o un conjunto de colores, es necesario implementar un sistema con varias funciones gaussianas, las que se deben entrenar según un criterio específico del problema. Un ejemplo de esto es el trabajo de Jones y Rehg en 2002 [1] (ver figura 2.3), que utiliza los parámetros de 16 funciones gaussianas para representar piel, y 16 para representar no-piel, de manera de considerar el cociente entre la suma ponderada del *pixel* evaluado en las funciones de piel y las de no-piel como el grado de pertenencia.

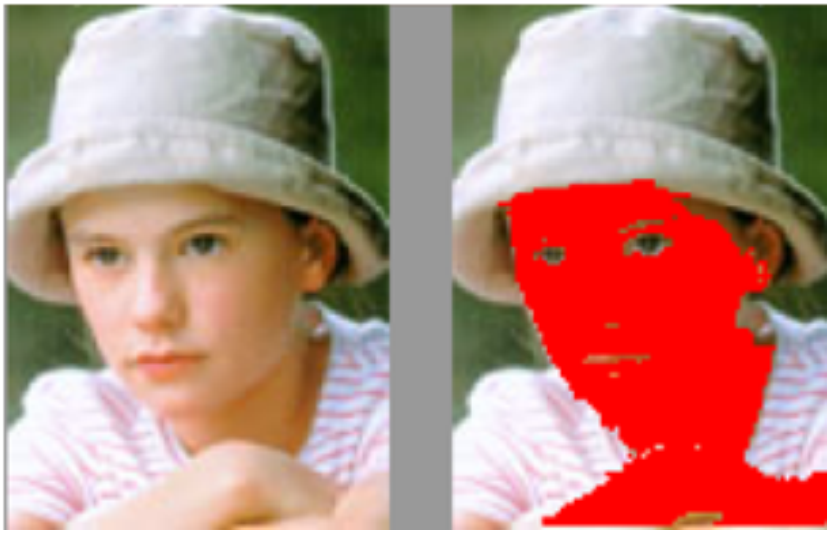


Figura 2.3: Segmentación de piel, ejemplo de una aplicación de segmentación de color [1].

2.2.2. Representación de clases a través de histogramas

Otro método utilizado para representar una clase es a través de histogramas [2] que consiste en calcular la frecuencia con la que se observan todos los colores en un determinado conjunto de entrenamiento a través de un histograma. Al normalizar este histograma se obtiene una aproximación de la verosimilitud de la clase, es decir, la probabilidad de observar un color al observar un elemento de la clase.

El método consiste en discretizar el espacio de colores y generar un histograma norma-

lizado a partir de los pixeles de entrenamiento de la clase. Luego se obtiene el grado de pertenencia de cada pixel como su valor correspondiente en el histograma.

A diferencia del método con gaussianas este método no hace el supuesto de unimodalidad, por lo que puede representar texturas de mejor manera. Tiene como inconveniente que solo es capaz de reconocer pixeles que estén previamente entrenados, ya que no interpola como la aproximación de las guassianas. Esta falencia puede ser suplida si se disminuye la resolución en el espacio de colores, lo que lleva consigo un error de aproximación que debe ser evaluado.

2.2.3. Segmentación adaptiva

En algunas aplicaciones es necesario que las definiciones de clase varíen en el tiempo y es deseable que la segmentación se adapte. Entregarle la capacidad de adaptación al sistema implica que se debe entregar un conjunto de entrenamiento durante la ejecución y que el conjunto que describe al color objetivo debe ser modificable en tiempo de ejecución. Adicionalmente se suele requerir que el sistema funcione en tiempo real, por lo que se va a necesitar que las etapas sean computacionalmente livianas. A continuación se muestran dos métodos que cumplen con estas condiciones: Uno aplicado al método de mezcla de gaussianas y otro a histogramas.

Segmentación adaptiva con gausianas

El método adaptativo basado en gausianas [3] utiliza un conjunto de k gausianas locales por cada imagen que recibe para describir una sección de ella que representa el entrenamiento, y un conjunto de n gaussianas para describir la clase a segmentar.

Las gausianas del conjunto de entrenamiento se pueden adaptar al sistema de dos maneras: Corrigiendo y reforzando una gausiana existente, o reemplazando una gausiana anterior. Estas formas de adaptación ayudan a darle invarianza a cambios lentos de iluminación y a cambios bruscos de color.

El proceso consiste en que para cada gausiana local j se calcula la distancia Mahalanobis con cada gausiana descriptora:

$$d(i,j) = (\mu_i - \mu_j)^T \left(\sum\nolimits_i + \sum\nolimits_j \right)^{-1} (\mu_i - \mu_j) \tag{2.4}$$

Sea i la gausiana que minimiza la distancia Mahalanobis para la gausiana de entrenamiento j. A partir de la distancia calculada se decide que hacer:

- Si la distancia es menor que cierto umbral se entiende que ambas gausianas i y j son similares, luego se actualiza la gausiana descriptora de la siguiente manera:

$$\mu_i \leftarrow \frac{m_i \mu_i}{m_i + m_j} + \frac{m_j \mu_j}{m_i + m_j} \tag{2.5}$$

$$\sum\nolimits_i \leftarrow \frac{m_i \sum_i}{m_i + m_j} + \frac{m_j \sum_j}{m_i + m_j} \tag{2.6}$$

$$m_i \leftarrow m_i + m_j \tag{2.7}$$

  Donde m_i es la cantidad de pixeles de la gausiana i.

- Si la distancia es mayor que dicho umbral entonces se asume que la gausiana j representa nueva información, por lo tanto se reemplaza a la gausiana con menor cantidad de pixeles m_i. En caso de haber gausianas no definidas, entonces se define con los valores de la gausiana.

Finalmente cada m_i de las gaussianas descriptoras se multiplica por un factor menor que uno, de manera de asegurar que las gaussianas en la memoria se pueden mover conforme varía el objetivo. El grado de pertenencia se calcula de igual manera que en en la descripción

de clases con múltiples gaussianas

Segmentación adaptiva con histogramas

También existe un método adaptativo para la descripción a través de histogramas. Para esto se genera un histograma a partir de los datos de entrenamiento H_t, representado como un arreglo de tres dimensiones que cubre todo el espacio de colores. Como descriptor de la clase a adaptar se tiene un histograma H_m.

Para llevar a cabo la adaptación, se promedian ponderadamente los histogramas:

$$H_m(i,j,k) \leftarrow \alpha H_m(i,j,k) + (1-\alpha) H_t(i,j,k) \mid \forall i,j,k \tag{2.8}$$

Donde α es un parámetro que controla la memoria del histograma, y por lo tanto la velocidad de adaptación del método.

Este método es presentado para adaptar un sistema de reconocimiento de piel, destacando su gran velocidad y fácil implementación [2].

2.2.4. Clasificadores estadísticos

Dentro del proceso de segmentación es necesario incluir un clasificador, es decir, una etapa que tome la decisión de qué clase representa una observación dada. Hasta el momento se ha utilizado como clasificador un simple umbral sobre los histogramas o las gaussianas, pero existen numerosos clasificadores cuya utilidad depende de la aplicación en particular. A continuación se presenta el clasificador utilizado en este trabajo.

Clasificador bayesiano

El clasificador bayesiano [4] es un clasificador probabilístico simple basado en el teorema de Bayes. En particular se utilizará un clasificador binario, donde las clases son *camino* o

no-camino, por lo que el clasificador se utiliza en su expresión más simple.

Primero es necesario destacar que la información recibida como entrenamiento debe ser interpretada estadísticamente como la probabilidad de que dada una clase, se obtenga una determinada observación, o en este caso, cúal es la probabilidad de observar un determinado color en el camino. Es decir, tanto los histogramas como las gaussianas que tratan de representar el color del camino, deben ser interpretados como la distribución de probabilidades antes definida.

$$P(x \mid x \in C) \tag{2.9}$$

Lo segundo es definir que en realidad se desea averiguar cuál es la probabilidad de que una observación sea de cada clase. Podemos ver entonces que no es coherente utilizar los valores de entrenamiento directamente para tomar una decisión, ya que no necesariamente son equivalentes.

$$P(x \mid x \in C) \neq P(x \in C \mid x) \tag{2.10}$$

Para encontrar una manera coherente de utilizar la información de entrenamiento se utiliza teorema de Bayes:

$$P(x \in C \mid x) = P(x \mid x \in C)\frac{P(x \in C)}{P(x)} \tag{2.11}$$

Sin embargo para utilizar esta expresión sería necesario tener una distribución de probabilidades para las observaciones independientemente de su clase y de la clase independientemente de las observaciones.

Si se toma en consideración que solo existen dos clases, entonces se puede reducir el problema a averiguar cuál clase la observación tiene un mayor grado de pertenencia, o en

este caso una mayor probabilidad. Para obtener la probabilidad de la observación de no pertenecer a la clase se utiliza la siguiente expresión:

$$P(x \notin C \mid x) = \frac{P(x \mid x \notin C)P(x \notin C)}{P(x)} \tag{2.12}$$

Luego, si la observación pertenece a la clase, se cumple que:

$$1 < \frac{P(x \in C \mid x)}{P(x \notin C \mid x)} = \frac{P(x \mid x \in C)P(x \in C)}{P(x \mid x \notin C)P(x \notin C)} \tag{2.13}$$

Se asume que la la probabilidad de una observación de ser o no de la clase es uniforme, y por lo tanto se puede asumir que $\frac{1}{k} = \frac{P(x \in C)}{P(x \notin C)}$ es una constante desconocida, luego una observación será parte de la clase C si:

$$k < \frac{P(x \mid x \in C)}{P(x \mid x \notin C)} \tag{2.14}$$

Donde k es un parámetro de diseño.

2.3. Filtrado de mediana

Es un filtro estadístico, que utiliza el operador mediana sobre un conjunto de datos determinados. En este caso se utiliza sobre la imagen de las probabilidades, y recibe como conjunto todos los valores de una vecindad, y reemplaza el centro por la mediana de estos.

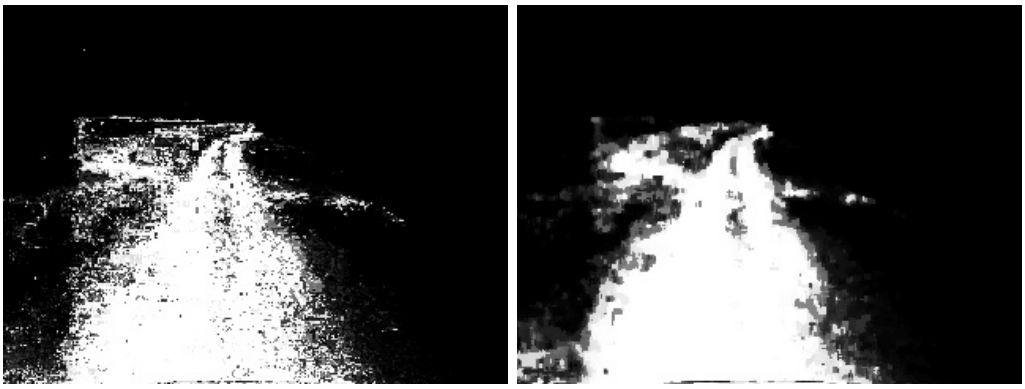
Este filtro se usa típicamente para casos de ruido pulsante o *salt and pepper*. Este filtro modifica una imagen de manera similar a un filtro pasa bajos (ver figura 2.4).

En este caso en particular utiliza este filtro ya que es posible encontrar valores muy altos o muy bajos debido al cociente en el cálculo de las verosimilitud del clasificador bayessiano.

Además es esperable la similitud con respecto a la vecindad.

Para el pixel (i, j), se calcula la mediana del conjunto de pixels de tamaño $(2k+1)$x$(2k+1)$ centrado en el pixel (i, j), como se observa en la ecuación (2.15), donde k es un parámetro del filtro.

$$M_{i;j} = Mediana\{I_{i+m;j+n} \mid \forall m, n \in [-k; k]\} \tag{2.15}$$



(a) Imagen original. (b) Imagen filtrada.

Figura 2.4: Efecto del filtro de mediana.

2.4. Filtrado mediante operadores morfológicos

Es una técnica de procesamiento de imágenes utilizada para obtener información sobre la morfología presente en una imagen binaria [5].

Un operador morfológico está compuesto de un elemento estructural y una operación. El elemento estructural es una imagen binaria con una forma y tamaño predefinidos dependiendo de la morfología que se busca. Los elementos estructurales más comunes son un cuadrado, una cruz o un círculo.

Las operaciones más comunes son erosión y dilatación. La erosión se define de la siguiente manera:

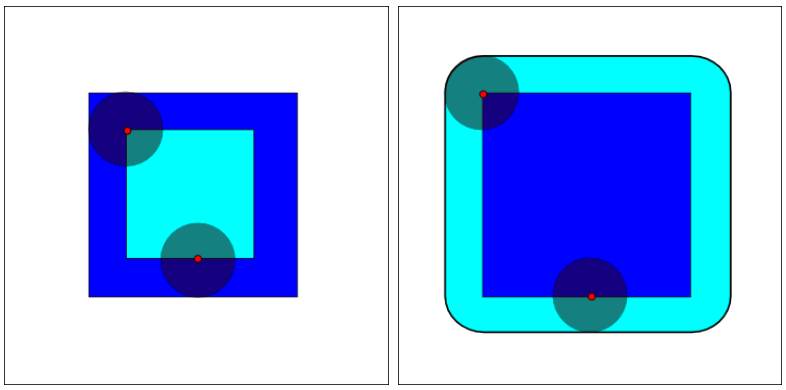
$$Im \ominus El = \{p \in Im \mid El_p \subset Im\} \tag{2.16}$$

La erosión es el conjunto de los pixeles de la imagen original en los que si se superpone el elemento estructural, este está contenido en la imagen. En la figura 2.5(a) se observa el efecto de la erosión con un elemento circular sobre el rectángulo azul.

La dilatación se define como:

$$Im \oplus El = \bigcup_{p \in Im} El_p \tag{2.17}$$

La dilatación se produce de la siguiente manera: Para cada pixel de la imagen, se agrega el elemento estructural superpuesto sobre el pixel. En la figura 2.5(b) se observa el efecto de la dilatación con un elemento circular sobre un rectángulo.



(a) Erosión. (b) Dilatación.

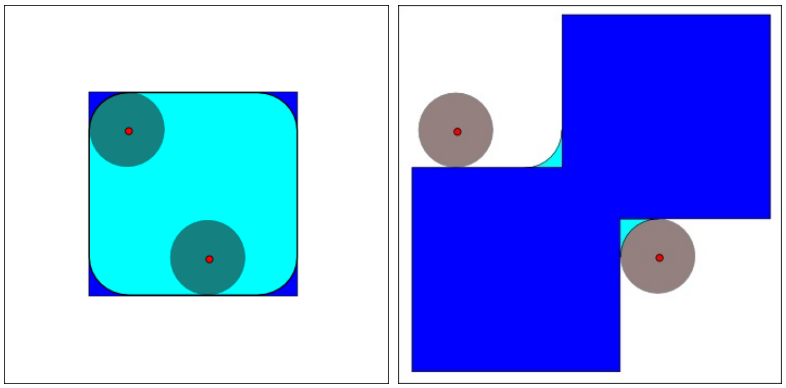
Figura 2.5: Ejemplo de operadores morfológicos.

Además es posible utilizar combinaciones consecutivas de operadores morfológicos para obtener distintos resultados. Las combinaciones más conocidas son la apertura y la clausura. La apertura es la utilización de una erosión seguida de una dilatación, y se utiliza para suavizar los bordes de la imagen 2.6(a).

$$Im \circ El = (Im \ominus El) \oplus El \tag{2.18}$$

La clausura es la utilización de una dilatación seguida de una erosión, y se utiliza para eliminar ruidos, especialmente para eliminar burbujas 2.6(b).

$$Im \bullet El = (Im \oplus El) \ominus El \tag{2.19}$$



(a) Apertura. (b) Clausura.

Figura 2.6: Ejemplos de combinaciones de operadores morfológicos.

2.5. Segmentación mediante inundación

Para el problema a abordar es necesario identificar una única región conexa: la que representa el camino. Debido a esto, y con el fin de cumplir con el funcionamiento en tiempo real, se utiliza un algoritmo de inundación adaptado para un único segmento [7]. La función de este algoritmo es de reconocer una región conexa de interés de la imagen segmentada.

El algoritmo utiliza una copia de la imagen de entrada, una imagen en blanco del mismo tamaño que será la salida y una lista de pixeles por revisar. Se parte de un pixel de la imagen previamente definido como parte del segmento conexo que se quiere reconocer, y éste se agrega a la lista, se borra de la imagen original y se marca en la imagen de salida. A continuación se busca el próximo elemento de la lista, se revisa si cada pixel vecino a él esta

marcado en la imagen original, de ser así se agrega a la lista, se borra de la imagen original y se marca en la imagen de salida. Este proceso se repite con cada elemento de la lista, y se termina cuando ya no se agregan nuevos elementos a la lista.

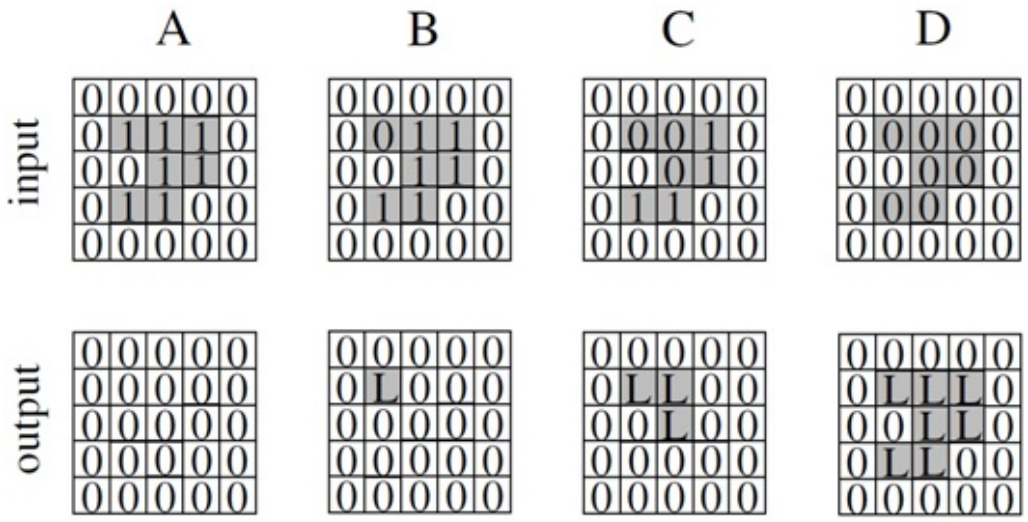


Figura 2.7: Evolución del algoritmo de inundación. En la parte superior la imagen de entrada, y en la parte inferior la imagen de salida.

2.6. Proyección de imágenes al espacio 3D

Una imagen es la proyección de los puntos visibles de los objetos en una escena sobre un plano. Una estimación común sobre esta proyección es la utilización de coordenadas esféricas para representar el espacio, con centro en el foco de la cámara. Para esto se considera que cada pixel de la imagen representa un segmento del casquete esférico, y que mide el color del punto visible más cercano al foco. Es decir que cuando se obtiene una imagen de una escena se pierde información sobre la profundidad de los distintos pixeles, y en consecuencia de los objetos que se representan en la imagen. Debido a esto no es posible reconstruir una escena utilizando únicamente la imagen, sino que es necesario utilizar otras fuentes de información.

2.6.1. Proyección de imágenes a un plano

En el caso de las imágenes de camino es posible asumir que el suelo es un plano a la altura de las ruedas del vehículo, por lo tanto es posible hacer una transformación de la imagen al espacio del vehículo. Para esto es necesario conocer las características constructivas de la cámara, su orientación y su posición relativa al suelo.

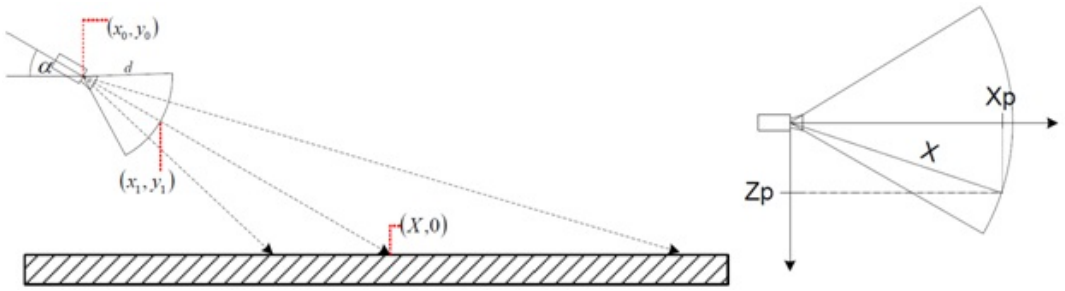


Figura 2.8: Esquema de proyección lateral (izq) y superior (der).

La proyección de una imagen sobre un plano de altura nula se calcula mediante la siguiente ecuación:

$$X(i,j) = x_0 - y_0 \left(\tan \left(\frac{-FoV_y}{nPix_y} j + \alpha \right) \right)^{-1} \tag{2.20}$$

Donde $X(i,j)$ es la distancia entre la proyección de la cámara en el suelo y la proyección del pixel de coordenadas (i,j), $nPix_y$ es la cantidad vertical de pixeles de la imagen, α es el ángulo de la cámara con respecto al piso y FoV_y es el campo de visión vertical. El valor de j es la coordenada vertical de un pixel determinado de la imagen. Destaca que esta fórmula se indefine en el horizonte, y proyecta hacia atrás de la cámara con los pixeles sobre el horizonte, por lo que sólo tiene sentido proyectar los pixeles por debajo del horizonte.

Finalmente se obtienen las coordenadas cartesianas en el plano para cada pixel a través de las siguientes ecuaciones:

$$Z_p(i,j) = Xsin\left(-\frac{FoV_x}{nPix_x}i + \frac{FoV_x}{2}\right) \tag{2.21}$$

$$X_p(i,j) = Xcos\left(-\frac{FoV_x}{nPix_x}i + \frac{FoV_x}{2}\right) \tag{2.22}$$

Donde FoV_x es el campo de visión horizontal y $nPix_x$ es la cantidad horizontal de pixeles de la imagen.

2.7. Codificación por largo de corridas de una imagen binaria

Las imágenes binarias, es decir representadas por solo dos posibles colores en cada pixel, pueden ser representadas de maneras más eficientes que a través de una imagen completa. Una posible representación es a través de largo de corridas o *run-length* [6] .

Esta representación transforma una imagen binaria en una lista de segmentos horizontales, definidos por las coordenadas del inicio y el largo del segmento medido en pixeles. Dado que existen dos colores posibles, se debe elegir cuál de ellos es el que será considerado como segmentos, y cuál será parte del fondo, en el caso particular de este sistema, la información relevante corresponde al color blanco.

El método para transformar una imagen binaria en una representación run-lenght es el siguiente: Se define un estado que indica si se está dentro de un segmento o no, por cada línea de la imagen se revisa cada pixel. Si el pixel es blanco y no se está dentro de un segmento, entonces se inicia uno, si se está dentro de un segmento, entonces se continúa. Si el pixel no es blanco y se está en un segmento, entonces se termina un segmento y se almacena en la lista, si no se está dentro de un segmento, entonces se continúa. Al finalizar una línea se cierra un segmento en caso de estar abierto, y al finalizar la imagen se considera terminada la transformación (ver figura 2.9).

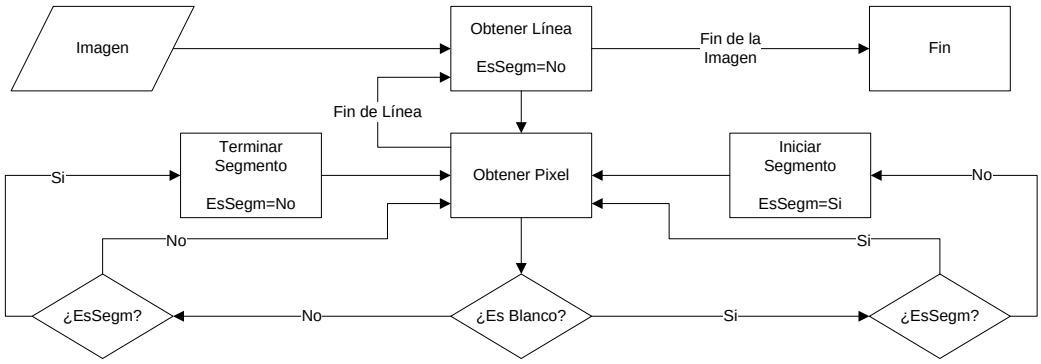


Figura 2.9: Diagrama de flujo de transformación run-lenght.

2.8. Detección de modelos matemáticos mediante RANSAC

RANSAC [8] es una abreviación de *Random Sample Consensus* (consenso de muestras aleatorias), y es un método iterativo para encontrar los parámetros de un modelo matemático en un conjunto con datos atípicos (ver figura 2.10). Es un método no determinístico, y por tanto existe una probabilidad de no encontrar el mejor modelo.

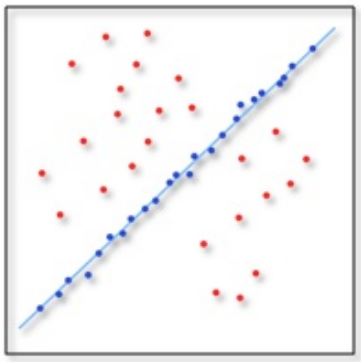


Figura 2.10: Línea encontrada con RANSAC.

El método recibe un conjunto de datos y un modelo matemático que se desea buscar dentro de este. El método sigue los siguientes pasos:

1. Se elige aleatoriamente un número de datos y construye un modelo a partir de ellos.
2. Se compara cada dato del conjunto con el modelo, y si un dato está bien representado por el modelo, se considera parte de este.
3. A partir de los datos que se consideran parte del modelo anterior, se calcula un nuevo modelo.
4. Si la cantidad de datos considerados parte del modelo son suficientes y su error es suficientemente bajo, entonces se considera el último modelo como la mejor representación. Si no, entonces se repite desde el paso 2 hasta un máximo de iteraciónes.

Dado que este método es altamente dependiente de los valores iniciales, se puede considerar que la iteración parte desde el paso 1 en vez del paso 2.

2.9. Detección de punto de fuga

Un punto de fuga en una imagen es el lugar donde convergen todas las rectas paralelas proyectadas en una dirección [18], excepto por las rectas paralelas al plano del observador. Este punto se encuentra a una distancia infinita del observador, existen tantos puntos de fuga como direcciones posibles y todos los puntos de fuga están situados en el horizonte de la imagen.

El punto de fuga fue un concepto creado en el renacimiento para representar la perspectiva en el dibujo, y fue utilizado inicialmente por artistas como Donatello, Masaccio y Leonardo Da Vinci.

En el caso de imágenes de caminos rectos, es posible seguir los bordes de la ruta hasta el horizonte para así poder encontrar el punto de fuga (ver figura 2.11(a)). La posición del punto de fuga en la imagen permite obtener información sobre la curvatura del camino: Si está en el centro de la imagen entonces la ruta es completamente recta, si está descentrada entonces indica una leve curvatura en ese sentido (ver figura 2.11(b)), y si está fuera de la

(a) Camino recto. (b) Curva leve. (c) Curva cerrada.

Figura 2.11: Ejemplos de imágenes de camino según curvatura.

imagen entonces se puede suponer una fuerte curvatura (ver figura 2.11(c)). Si el camino está ocluido, tiene una forma poco uniforme o presenta muchas curvas el punto de fuga no quedará bien definido.

Capítulo 3

Sistema desarrollado

3.1. Metodología

La solución implementada consta principalmente de dos etapas: la etapa de segmentación y la etapa de detección de camino (ver figura 3.1).

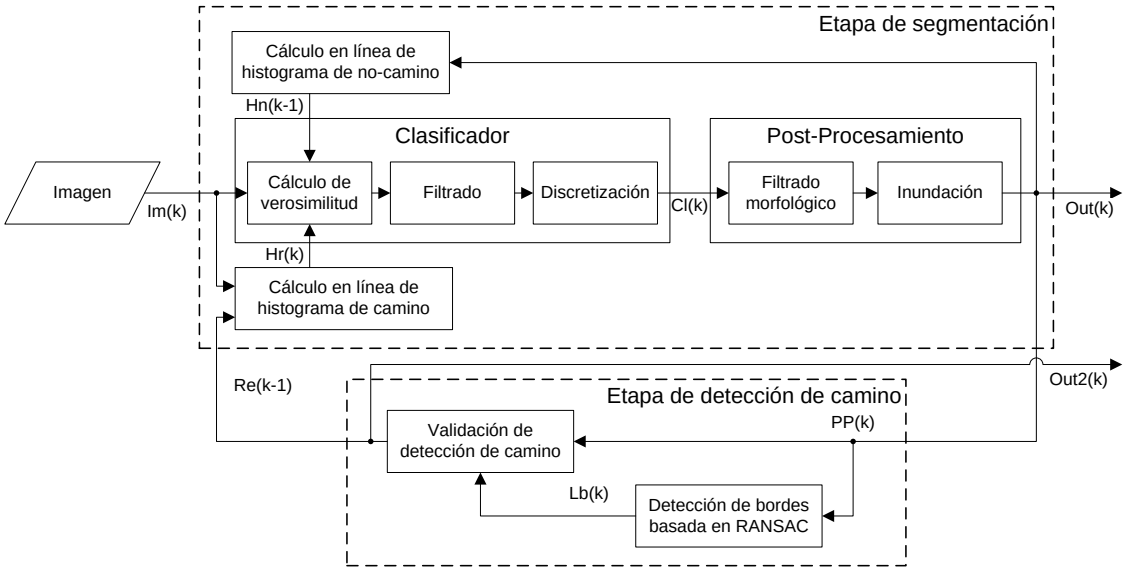


Figura 3.1: Diagrama del sistema implementado.

En la primera etapa se une la información de una región fija de la imagen con una retroalimentación para obtener un conjunto de entrenamiento. Este conjunto de entrenamiento se utiliza para actualizar la definición de color de camino. Utilizando la definición de camino actualizada se clasifica cada pixel utilizando un clasificador bayesiano, obteniéndose la verosimilitud para cada pixel representada con una imagen en escala de grises. Sobre esta imagen se aplica un procesamiento a fin de evitar ruido, luego se binariza aplicando un umbral. A esta última imagen binarizada se le aplicarán filtros que eliminaran ruidos pequeños y se elegirá solamente el segmento que esté conectado con la región de entrenamiento. El resultado es entonces una imagen binaria que define que segmento de la imagen representa el camino según el color. La zona que no se define como camino se utiliza para actualizar el modelo de no-camino.

La segunda etapa busca detectar la morfología del camino. Recibe la imagen binarizada e identifica en ella los bordes de la región segmentada. A partir de estos bordes se analiza si son consistentes con la definición de camino basada en la completitud de la región definida como camino, en la coherencia del punto de fuga con la posición de la cámara respecto al suelo y en las últimas mediciones de camino.

Si la morfología del camino es correcta, entonces se utiliza para reforzar la segmentación de la próxima imagen.

3.2. Etapa de segmentación

Esta etapa es la encargada de llevar a cabo la segmentación basada en colores del camino. Utiliza histogramas adaptativos para representar la clase camino y no camino, y un clasificador bayesiano. Recibe como entrada cada imagen, y entrega como resultado una imagen binaria con la región conexa que representa al camino de acuerdo con la segmentación de color. (ver figura 3.1).

El histograma de camino y el de no-camino representan la probabilidad de observar un color en el camino o no-camino para cada color del espacio de colores, respectivamente.

$$H_{Camino} \approx \{P(Color \mid Color \in \text{Camino})\}$$

$$H_{No-Camino} \approx \{P(Color \mid Color \notin \text{No-Camino})\}$$

Estos histogramas son utilizados como las probabilidades condicionales inversas en el clasificador bayesiano (ver seccion 2.2.4), y la adaptabilidad del sistema depende de cómo éstos se actualizan en el tiempo.

Al recibirse la imagen y su región de entrenamiento, se entregan al bloque de cálculo en línea de histograma de camino. Luego se aplica el bloque clasificador sobre la imagen, entregando una imagen binaria que define cuales pixeles son camino según su color. Esta imagen binaria se entrega a una etapa de post-procesamiento que disminuye el ruido y elimina las regiones que no están conectadas al la región de entrenamiento, entregando una última imagen binaria con la mejor aproximación del camino posible. El negativo de esta última imagen se entrega al cálculo en línea de histograma de no-camino. A continuación se da una explicación más detallada del funcionamiento de cada bloque.

3.2.1. Cálculo en línea de histograma de camino

Este bloque recibe la imagen original y su región de camino, y su fin es actualizar el histograma de camino. Para esto utiliza un histograma temporal y un contador.

Primero se vacía el contador y el histograma, luego se recorre la imagen, buscando los pixeles que están dentro de la región de entrenamiento. Para cada pixel que cumple con esta condición, se agrega en el histograma, y se aumenta el contador.

Al final del recorrido de la imagen, se normaliza el histograma, dividiendo cada valor por el contador y se actualiza el histograma de camino a través de un promedio ponderado, como se explica en la sección 2.2.3.

3.2.2. Clasificador

Recibe la imagen original y la retroalimentación entregada por la salida de la etapa de detección de camino, y su fin es entregar una imagen binaria indicando cuales pixeles pueden ser considerados como parte del camino (ver figura 3.2). Se divide en tres etapas consecutivas:

Cálculo de verosimilitudes: Primero se recorre la imagen calculando la probabilidad de cada pixel de ser camino según el clasificador bayesiano (ver sección 2.2.4) a través del cociente entre ambos histogramas evaluados en dicho pixel. Debido a que este valor calculado pertenece al intervalo entre cero e infinito positivo, se fija un valor máximo permitido.

Filtrado de Mediana: Sobre la imagen de probabilidades se aplica un filtro de medianas (ver sección 2.3) para eliminar el ruido e incluir efectos de vecindad sobre la probabilidad de cada pixel.

Binarización: A la imagen de probabilidades filtrada se aplica un umbral variable para entregar un resultado final discreto indicando cual pixel puede ser considerado camino y cual no.

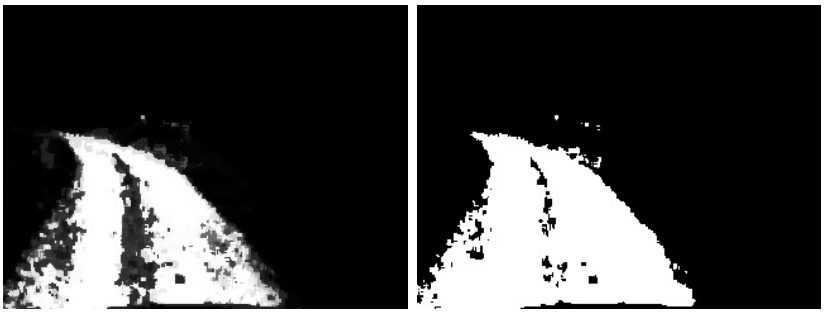
$$B[i] = \begin{cases} 1 & Im_f[i] \geq T \\ 0 & Im_f[i] < T \end{cases} \tag{3.1}$$

Donde Im_f es la imagen de probabilidades filtrada y T es un umbral constante predefinido.

3.2.3. Post-Procesamiento

El prost-procesamiento trabaja sobre la imagen binaria que representa los pixeles que pueden ser considerados como camino, y entrega una imagen binaria con la mejor aproxi-

(a) Imagen original. (b) Imagen de probabilidades.



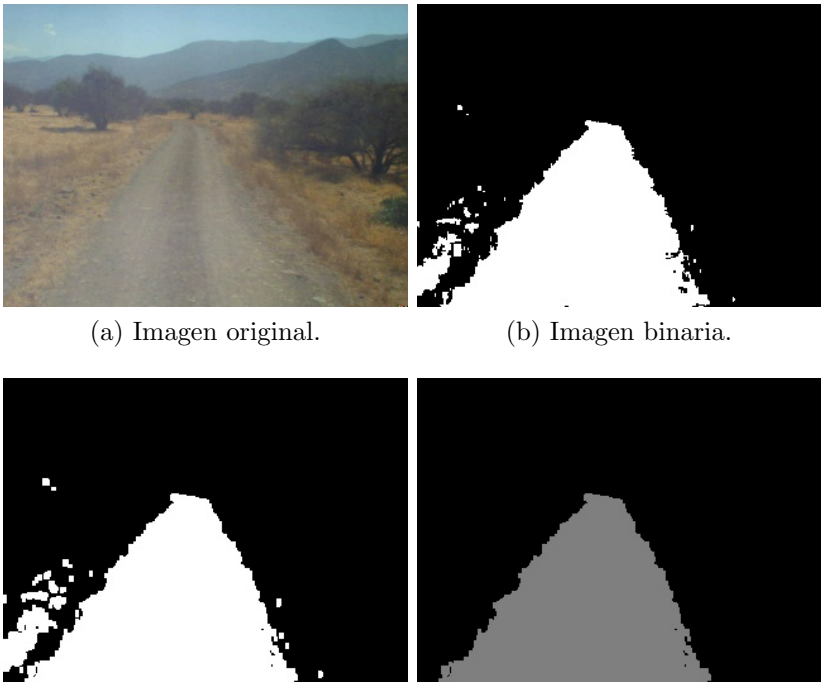
(c) Imagen filtrada. (d) Imagen binaria

Figura 3.2: Ejemplo del bloque clasificador.

mación al segmento de la imagen que puede ser interpretado como camino (ver figura 3.3). Este bloque consta de dos etapas:

Filtrado mediante Operadores Morfológicos: En esta etapa se aplican operadores morfológicos (ver seccion 2.4), a fin de eliminar las pequeñas burbujas no segmentadas al interior del camino, y disminuir los ruidos en los bordes. Los operadores utilizados son una dilatación seguida de dos erosiones [3]. El resultado de esta etapa es una versión suavizada de la imagen binaria original.

Inundación: Aquí se selecciona (ver sección 2.5) exclusivamente el segmento de la imagen binarizada conexo a la región de entrenamiento, y se omiten el resto de los pixeles. De esta manera se exige al resultado que sea coherente con el entrenamiento instantáneo, y que el resultado sea un camino conexo.



(a) Imagen original. (b) Imagen binaria.

(c) Operadores morfologicos. (d) Algoritmo de inundacion

Figura 3.3: Ejemplo del bloque de post-procesamiento.

3.2.4. Cálculo en línea de histogramas de no-camino

Se recibe la imagen original y la última imagen binarizada, y su fin es actualizar el histograma de no-camino. Para esto utiliza un histograma temporal y un contador.

Este bloque funciona de igual manera que el bloque de entrenamiento de histogramas de camino, pero se diferencia en el conjunto que utiliza como entrenamiento. Primero se vacía el contador y el histograma, luego se recorre la imagen buscando los pixeles que no son camino en la imagen segmentada. Para cada pixel que cumple con esta condición, se agrega en el histograma, y se aumenta el contador.

Finalmente, se normaliza el histograma, dividiendo cada valor por el del contador y se actualiza el histograma de camino a través de un promedio ponderado, como se explica en la sección 2.2.3.

3.3. Etapa de detección de camino

Se busca reconocer la forma del camino en la imagen segmentada, para tener una medida de que tan confiable es el resultado de la segmentación(ver figura 3.1).
Esta etapa recibe la imagen segmentada del camino, representada como una imagen binaria y entrega como resultado una imagen con una representación del camino.

La imagen segmentada es proyectada en el plano del suelo, y sobre el camino proyectado se buscan los bordes derecho e izquierdo del camino a través de un algoritmo basado en RANSAC (ver sección 2.8).

Una vez encontrados los bordes, se someten a un conjunto de reglas para validar que el camino que describen es coherente con una definición previa de camino, que incluye posición del punto de fuga, coherencia con medidas anteriores, y el porcentaje del camino segmentado.

La salida de esta etapa es una imagen que representa la mejor aproximación de la forma del camino a través de un filtro temporal.

3.3.1. Detección de bordes basada en RANSAC

En esta etapa se busca detectar y representar los bordes del camino a través de líneas recta. Si bien no todos los caminos pueden ser representados como con este modelo, resulta una buena aproximación considerando que se desea detectar caminos para una conducción a velocidades altas. Ésta consta de 3 partes: la transformación de la imagen binaria a una representación run-length (ver sección 2.7), una proyección de la imagen al plano del suelo (ver sección 2.6.1), y la detección de líneas basada en RANSAC (ver sección 2.8).

Representación run-length de la imagen: Se utiliza esta representación en la imagen binaria para disminuir el tiempo de cálculo, y facilitar la búsqueda de los candidatos de bordes. En esta representación la imagen queda como una lista de líneas horizontales, donde cada línea esta descrita como las coordenadas del borde izquierdo, y el largo de la línea. Con esta información es rápido encontrar todos los candidatos a bordes del camino, además de permitir diferenciar los candidatos de borde izquierdo de los

candidatos de borde derecho.

Proyección de la imagen sobre el suelo: Utilizando la información de la posición de la cámara en el auto, y suponiendo que la superficie del camino es plana, se proyecta cada pixel de borde sobre el plano tridimensional que representa al suelo en el que se encuentra el vehículo. Esta transformación de la imagen permite buscar líneas rectas en el camino, y no en su imagen. Además facilita la búsqueda de líneas al ser menos significativo el error en pixeles de la región mas cercana, y mas significativo al acercarse al horizonte.

Detección de líneas basado en RANSAC: Usando separadamente los candidatos de cada borde, se busca la mejor representación posible de una recta en la imagen proyectada. Este método elige aleatoriamente un par de pixeles candidatos a borde, y a partir de ellos se calcula un modelo de línea recta. Luego se calcula la distancia de cada pixel candidato a borde a esta recta, y si la distancia es menor que un cierto umbral, entonces se considera que este candidato es parte de esta línea. Si la cantidad de candidatos es mayor que un cierto porcentaje, se considera que el conjunto de candidatos es suficientemente representativo del borde del camino, y se calcula una regresión lineal para obtener un modelo definitivo del borde del camino (ver figura 3.4(c)) . Pero si la cantidad de candidatos no es suficiente, se eligen aleatoriamente otros dos candidatos y se repite el procedimiento. Si después de una cierta cantidad de intentos no se obtiene un buen modelo de línea, se asume que no existen bordes bien delimitados.

3.3.2. Validación de la detección del camino

Debido al ruido natural del sistema, es común que se encuentren bordes erróneos y representaciones incorrectas del camino, por lo que es necesario incorporar una etapa de validación de la representación del camino, en donde se evalúe si el modelo de camino detectado es válido en función de la información disponible. Esta etapa se compone de un conjunto de reglas que discriminan distintos aspectos que definen el camino, y por lo tanto es necesario que se aprueben todas para que se considere que el modelo de camino es válido.

Filtrado por punto de fuga Si el camino es recto, entonces el punto de fuga definido por la intersección de los bordes debe estar situado en la línea del horizonte. Debido a que el sistema está sujeto a ruido, esta regla exige que el punto de fuga del camino este a menos de una distancia máxima en pixeles de la línea del horizonte.

Filtrado por completitud de la región En algunos casos la segmentación permite que se detecten bordes de camino que en realidad no lo son. Como solución a este problema se mide el porcentaje de la región definida como camino que esta segmentada. Si este porcentaje está por debajo que cierto umbral, se rechaza la definición de camino. Este filtro también evita que se definan como camino zonas incompletas o curvas.

Filtrado de coherencia temporal Utilizando el hecho de que el sistema está pensado para funcionar a 15 imágenes por segundo, es posible exigir que las definiciones de camino para imágenes consecutivas sean relativamente parecidas. Es por esto que se exige la intersección entre las regiones definidas como camino según el modelo actual y el modelo de la imagen anterior tengan una diferencia pequeña, medida como la razón del área de la intersección y el área anterior. Si esta razón es mayor que un cierto umbral, se considera que el modelo es coherente temporalmente.

Cuando una detección de camino pasa estos filtros se genera una imagen con el modelo de camino detectado. Para compensar lo ruidoso de las detecciónes, esta imagen es filtrada temporalmente (ver figura 3.4(d)). Esta imagen filtrada temporalmente se utiliza como nivel de certeza sobre la forma del camino, y entrega información clara sobre su orientación, pero al suponer radio de curvatura fijo, genera errores considerables sobre los 20 metros. Es por esto que solamente la sección de la imagen que representa los primeros 20 metros se utiliza como retroalimentación para el entrenamiento del histograma de camino.

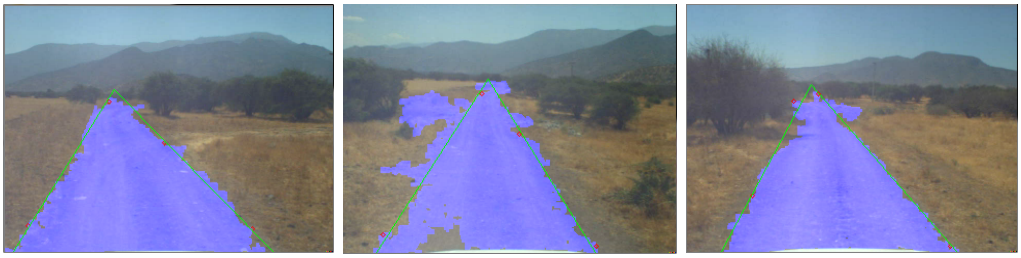
Para obtener la sección de la imagen que se utiliza como retroalimentación se omiten todos los pixeles que estén por sobre los 20 metros, y al resto se compara con un umbral fijo. La imagen binaria resultante de esta operación se une con la región de entrenamiento fija y se genera la región de camino requerida para el cálculo en línea de histograma de camino.

En la figura 3.4 se muestra imagenes de las distintas etapas del sistema, (a) son la imágenes originales tomadas desde el vehículo, (b) es la salida de la etapa de segmentación, (c) muestra la detección realizada con RANSAC sobre la imagen original y su región segmen-

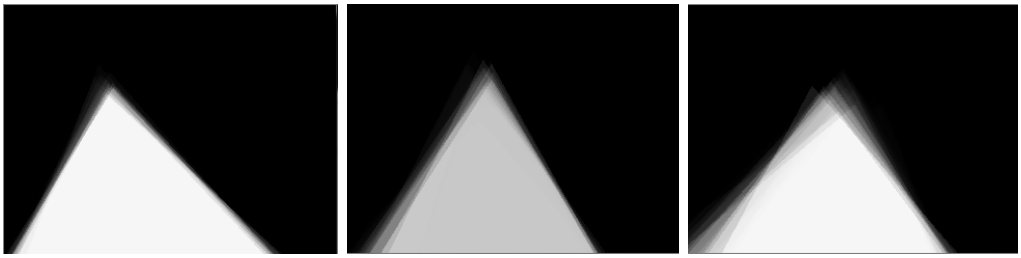
tada en color azul y (d) es la salida de la etapa de detección de camino tras la aplicacin de las reglas.

(a) Imagen original.

(b) Salida de la etapa de segmentación.



(c) Resultado de RANSAC.



(d) Salida del bloque de validación.

Figura 3.4: Etapa de detección de camino.

Capítulo 4

Experimentos

Tras haber implementado el sistema propuesto, resultó necesario medir su comportamiento para corroborar que cumpliera con los requisitos establecidos, como tiempo de procesamiento por cuadro, rendimiento y robustez del sistema.

La primera implementación del sistema completo estaba muy lejos de cumplir los requisitos de funcionamiento en tiempo real, por lo que fue necesario optimizar las distintas etapas del código y calibrar los parámetros relacionados. Dentro de las medidas mas importantes en este ámbito están la utilización de la librería estándar de C++ para el manejo de datos y estructuras, la disminución al mínimo del uso de memoria dinámica y la correcta selección de la discretización de los histogramas. La discretización utilizada fue de 32 segmentos por canal de color.

Otro problema presente en la primera implementación del programa fue que el tiempo de procesamiento aumentaba en el tiempo, hasta que dejaba de funcionar. El problema que presentaba era debido al mal uso de la memoria dinámica. Esto fue reparado al incorporar el uso de la librería estándar y al reducir el uso de la memoria dinámica. Actualmente el programa funciona a una frecuencia estable, y utiliza una cantidad fija de memoria.

Para medir el tiempo de procesamiento se utilizó una base de datos existente de 2500 imágenes y un computador de escritorio con un procesador Intel Core i5 de 2.27 [GHz]. Se calculó el tiempo total de procesamiento de la base de datos, el tiempo promedio obtenido fue de 58[ms], lo que permite un funcionamiento de aproximadamente 17[fps]. Cabe destacar que el programa no incluye ningún tipo de paralelismo en su programación, por lo que este

desempeño puede ser mejorado.

Para medir el rendimiento del sistema, fue necesario generar una base de datos donde estuviera marcado lo que se considera camino. A continuación se explica el método utilizado para lograr esto, y los experimentos utilizados para medir el rendimiento del sistema.

4.1. Generación de base de datos

Para poder obtener una medición cuantitativa del desempeño del sistema fue indispensable la construcción de una base de datos de imágenes donde el camino esté marcado. Para esto se utilizó una grabación tomada durante diciembre del año 2009, en el predio de Laguna Carén. Esta grabación consta de 2500 imágenes tomadas a una frecuencia promedio de 10[fps], con una resolución de 320x240, y con una cámara con un campo de visión horizontal de 45°.

Para poder marcar en cada imagen la región que representa el camino se desarrolló un programa en Matlab (ver figura 4.1), el cual muestra cada imagen, y permite que se agreguen, eliminen o muevan puntos que marcan el borde de la región. Una vez fijados los puntos, el programa inunda la región delimitada, y guarda una imagen binaria con la región descrita. Finalmente muestra la imagen siguiente, manteniendo los puntos anteriores. Esto facilita el marcado, ya que al ser imágenes tan cercanas temporalmente, suelen ser muy similares, y por tanto la región que define el camino es prácticamente la misma, o solo requiere mover levemente los bordes.

4.2. Desempeño de la etapa de segmentación.

Para medir el desempeño de la etapa de segmentación, se utiliza la curva ROC (Receiver Operating Characteristic, o Característica Operativa del Receptor). Para generar esta curva se calcula el TPR(True Positive Rate, o Razón de Verdaderos Positivos) y el FPR(False Positive Rate, o Razón de Falsos Positivos) de cada imagen como se muestra a continuación:

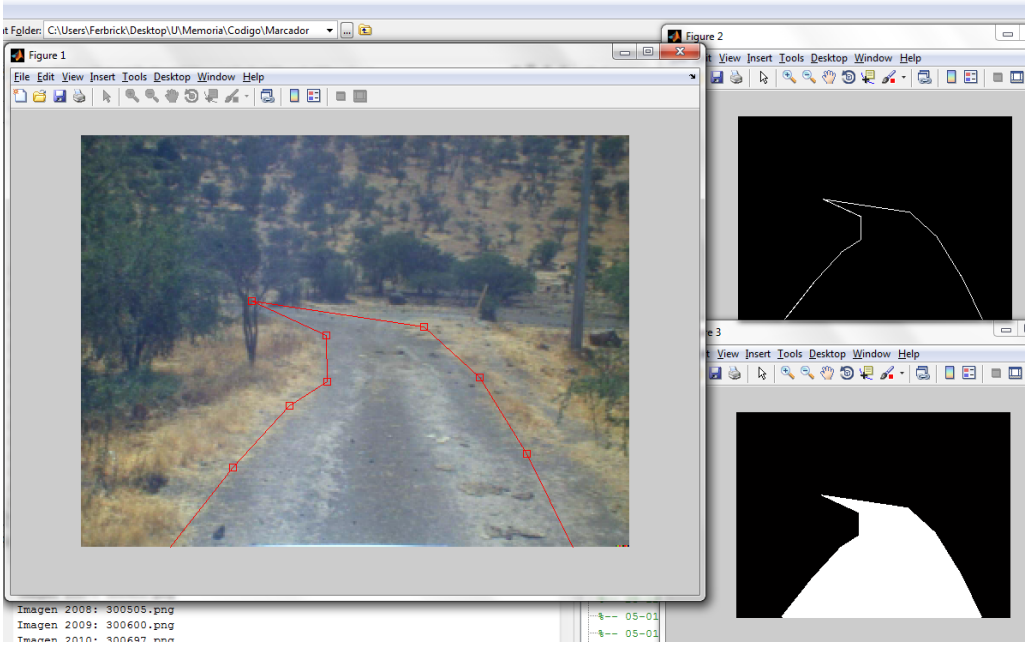


Figura 4.1: Programa para marcar caminos.

$$\text{TPR} = \frac{\sum (I_{GT} \ \& \ I_S)}{\sum I_{GT}} \tag{4.1}$$

$$\text{FPR} = \frac{\sum \left(\overline{I_{GT}} \ \& \ I_S\right)}{\sum \overline{I_{GT}}} \tag{4.2}$$

Donde I_{GT} es la imagen donde está marcada la región de camino e I_S es la imagen segmentada entregada por el sistema. Para obtener una descripción final del desempeño del sistema, el valor final del TPR y FPR en todo el video se calcula como el promedio de cada imagen y se genera una curva ROC usando como parámetro de umbral de binarización T de la ecuación (3.1).

Los resultados de este experimento se muestran en la figura.4.2. Ahí se observa una curva con el sistema implementado en lazo abierto (*Open Loop*), es decir, sin la retroalimentación de la etapa de detección de camino, y una con el sistema en lazo cerrado (*Closed Loop*).

4.3. Utilización para toma de decisiones.

Este segundo experimento esta diseñado para evaluar cuan confiable es este sistema para tomar decisiones. Para esto se utiliza un método de selección de ruta simple alimentado con la salida del sistema, y se compara con el mismo método alimentado con la base de datos marcada. Este método de selección de ruta utiliza un modelo de ruta con curvatura fija [15], donde cada ruta posible tiene cuatro parámetros: curvatura, orientación, ancho y largo (ver figura 4.3). Dado que se desea identificar el camino que seguiría el vehículo, el ancho de la ruta es el ancho del vehículo, y el largo de la ruta queda determinado por la segmentación del camino, de forma tal que la ruta esté totalmente segmentada. Por lo tanto, cada ruta tiene dos variables independientes: curvatura y orientación. El método consiste en calcular para cada imagen la distancia de todas las rutas posibles, en un conjunto discreto de curvaturas y orientaciones, y considerar como decisión a la ruta mas larga.

Para comparar el desempeño del sistema, se mide el porcentaje de imágenes donde el sistema es incapaz de encontrar una ruta posible, y la tasa de detección, definida como el porcentaje de la ruta que se encuentra dentro del camino marcado en la base de datos. Los resultados de este experimento se muestran en la tabla 4.1.

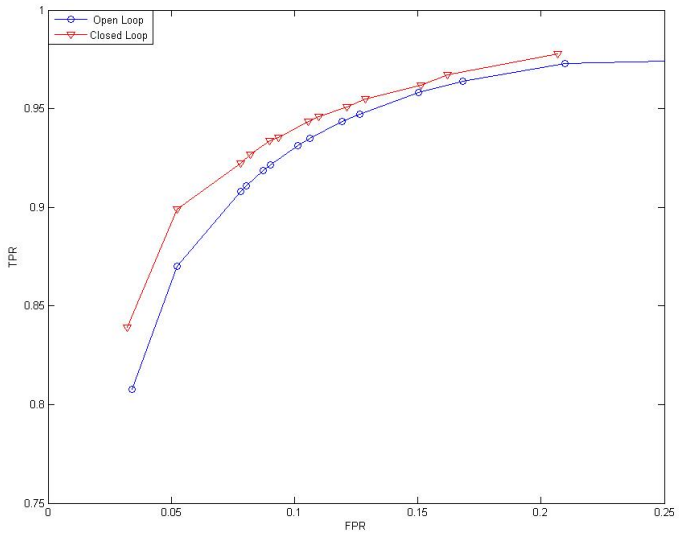


Figura 4.2: Curva ROC del sistema de segmentación.

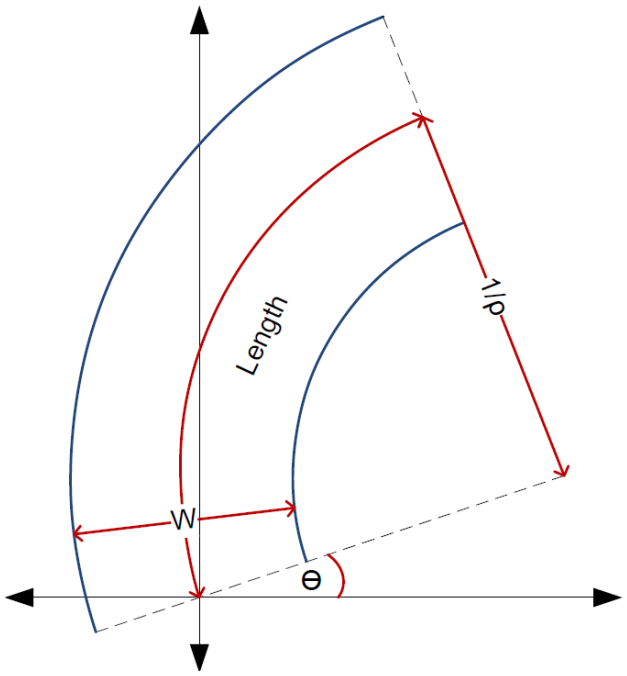


Figura 4.3: Modelo geométrico de una ruta posible en 2D. θ es la orientacion, $\frac{1}{\rho}$ es el radio de curvatura, W es el ancho y *Lenght* es el largo de la ruta.

Tabla 4.1: Resultados de toma de decisiones.

	Sistema diseñado	Base de datos
Imágenes sin ruta posible	9.57 %	7.84 %
Tasa de detección	96.92 %	100 %

La figura 4.4 muestra nueve posibles rutas desde la cámara. De izquierda a derecha se cambia la curvatura y de arriba a abajo se cambia la orientación.

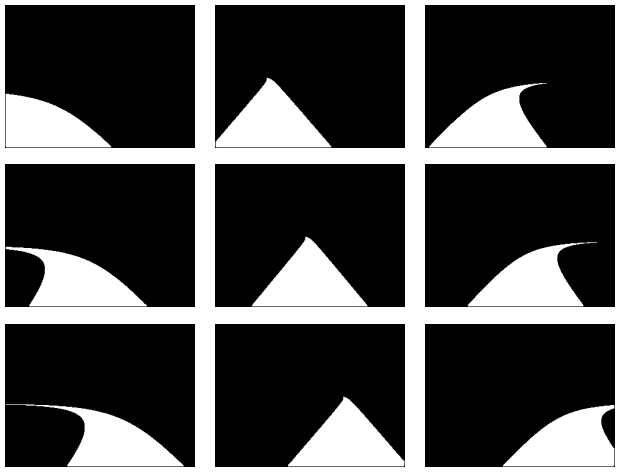


Figura 4.4: Ejemplos de posibles caminos con largo infinito.

Capítulo 5

Análisis y Conclusiones

Se ha desarrollado e implementado exitósamente un sistema visual adaptativo para la detección de calzada, veloz y confiable, que cumple con los requerimientos para ser utilizado como parte del sistema de medición del entorno del proyecto Vehículo Autónomo.

La adaptabilidad del sistema es una de sus principales cualidades, y el hecho que no requiera entrenamiento previo es una clara muestra de esta capacidad del sistema. Si bien no se ha hecho una evaluación cuantitativa del comportamiento en este aspecto, se puede mencionar que, cualitatívamente hablando, el sistema se adapta rápidamente a cambios en el camino, como sombras o iluminación.

El tiempo de procesamiento del sistema, de 58.6[ms] en promedio, permite procesar aproximadamente $17,8$ imágenes por segundo, logrando el requerimiento inicial de 15 imágenes por segundo, lo que asegura un tiempo de respuesta adecuado en caso de ser necesario un frenado de emergencia. Cabe destacar que la principal innovación de el sistema diseñado es la etapa de detección de camino y su retroalimentación a la etapa de segmentación, y que el tiempo de procesamiento requerido por esta etapa es de $17,8$[ms] con un procesador Intel Core i5 de 2.27[GHz]. Por lo que se observa que el principal costo de la implementación de este método, su tiempo de procesamiento, es suficientemente bajo.

A través de los experimentos realizados se muestra que el sistema es confiable para esta aplicación. Del primer experimento se observa que el sistema tiene un rendimiento comparable con los sistemas encontrados en la literatura [9–16], pese a que se utilizan bases de datos distintas. Además, se observa que la utilización de la etapa de detección de camino

como retroalimentación genera un efecto positivo en el desempeño del sistema.

El segundo experimento muestra que pese a las diferencias entre el camino real y el detectado visualmente, es posible tomar decisiones a partir de la imagen segmentada y obtener decisiones parecidas a las tomadas a partir del camino real. Esto reafirma lo confiable del sistema.

Sobre los objetivos específicos, la investigación fue llevada a cabo correctamente, y entregó suficientes herramientas como para diseñar un sistema de detección que cumple con los requisitos preestablecidos. A partir de la investigacion, se implementaron tres sistemas distintos: El primero fue la segmentación a través de una gaussiana, la cual no se menciona en este documento puesto que los resultados obtenidos estaban muy por debajo de lo esperado y no amerita mayor análisis, el sistema en lazo abierto y el sistema en lazo cerrado, sin embargo queda como trabajo futuro la implementación del sistema de segmentación adaptativa a través de múltiples gaussianas. Los experimentos generados en el capitulo 4, se consideran como establecidos para generar comparaciones válidas entre sistemas de detección de calzada. La integración al vehículo estará terminada antes del 29 de Junio del presente año, ya que en esa fecha se ejecutará un experimento donde se conducirá el vehículo con un algoritmo de conducción basado principalmente en la detección de este sistema. Finalmente, la documentacion del trabajo incluye este documento y la información adjunta al código del programa.

Para el trabajo futuro se propone calibrar los parámetros del sistema a través de un algoritmo evolutivo, de manera que se obtenga el mejor desempeño posible del sistema. Además, se propone incluir información de textura en la etapa de segmentación, a través de la transformada mLBP. También se propone incorporar transformación para invarianza a la iluminación, lo que permitiría mejorar el funcionamiento del sistema con respecto a la posición del sol [14]. Otro trabajo propuesto es incorporar este sistema visual con el resto del sistema de medición de ambiente, de manera de obtener una definición dinámica de la región de entrenamiento de la imagen. También se propone implementar una segmentación adaptativa con gaussianas para poder comparar su funcionamiento con el del sistema implementado.

Referencias

[1] Michael J. Jones , James M. Rehg, *Statistical color models with application to skin detection*, International Journal of Computer Vision, v.46 n.1, p.81-96, January 2002.

[2] Correa, M., Ruiz-del-Solar, J., Verschae, R., Lee-Ferng, J. Castillo, N. (2010). *Real-Time Hand Gesture Recognition for Human Robot Interaction*, Lecture Notes in Computer Science 5949 (RoboCup Symposium 2009), pp. 46-57.

[3] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A. and Mahoney, P. (2006), *Stanley: The robot that won the DARPA Grand Challenge*, Journal of Field Robotics, 23: 661692. doi: 10.1002/rob.20147

[4] Domingos, Pedro & Michael Pazzani (1997). *On the optimality of the simple Bayesian classifier under zero-one loss*, Machine Learning, 29:103137.

[5] Jean Serra, *Image Analysis and Mathematical Morphology*, ISBN 0126372403 (1982)

[6] S. W. Golomb. *Run-length encodings*, IEEE Trans. Inform. Theory, vol. IT-12, pp. 399401, July 1966.

[7] Bevilacqua, A., Lanza, A., Baccarani, G. Rovatti, R. (2003), *A Single-Scan Algorithm for Connected Components Labelling in a Traffic Monitoring Application*. SCIA'03 Proceedings of the 13th Scandinavian conference on Image analysis.

[8] Martin A. Fischler and Robert C. Bolles (June 1981). *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. Comm. of the ACM 24: 381395. doi:10.1145/358669.358692

[9] Tan, C., Hong, T., Shneier, M., & Chang, T. (2006). *Color model-based real-time learning for road following*. In IEEE intelligent transportation systems conference (ITSC06) (pp. 939944). Toronto, Canada.

[10] Hui Kong; Audibert, J.-Y.; Ponce, J. (2010). *General Road Detection From a Single Image*. IEEE Transactions on Image Processing

[11] Shengyan Zhou, Yanhua Jiang, Junqiang Xi, Jianwei Gong, Guangming Xiong, Huiyan Chen. (2010). *A Novel Lane Detection based on Geometrical Model and Gabor Filter*. 2010 IEEE Intelligent Vehicles Symposium, University of California, San Diego, CA, USA.

[12] Jian Wang, Zhong Ji, Yu-Ting Su (2009). *Unstructured Road Detection Using Hybrid Features*. Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, 12-15 July 2009.

[13] ZuWhan Kim (2008). *Robust Lane Detection and Tracking in Challenging Scenarios*. IEEE Transactions on Intelligent Transportation System, Vol. 9, No. 1, March 2008.

[14] J.M. Álvarez, A. López and R. Baldrich (2008). *Illuminant-Invariant Model-Based Road Segmentation*. IEEE Intelligent Vehicles Symposium Eindhoven University of Technology Eindhoven, The Netherlands, June 4-6, 2008.

[15] Shengyan Zhou, Yanhua Jiang, Junqiang Xi, Jianwei Gong, Guangming Xiong, Huiyan Chen (2010). *A Novel Lane Detection based on Geometrical Model and Gabor Filter*. 2010 IEEE Intelligent Vehicles Symposium, University of California, San Diego, CA, USA, June 21-24, 2010

[16] Joel C. McCall and Mohan M. Trivedi (2006). *Video-Based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation*. IEEE Transactions on Intelligent Transportation Systems, Vol. 7, No. 1, March 2006.

[17] Guerrero, P., Ruiz-del-Solar, J., Fredes, J., and Palma-Amestoy,R.(2008). *Automatic On-Line Color Segmentation using Classes-Relative color Spaces*. Lecture Notes in Computer Science 5001 (RoboCup 2007) pp. 246-253

[18] M. J. Magee, J. K. Aggarwal (1984), *Determining vanishing points from perspective images*, Computer Vision, Graphics, and Image Processing, Volume 26, Issue 2, May 1984, Pages 256-267.