

Vehículo reconocedor de señales de tráfico

Manuel López Urbina



Director: Arturo Morgado Estévez

Universidad de Cádiz
Escuela Superior de Ingeniería

Índice

- 1 Índice
- 2 Presentación
- 3 Planificación
- 4 Desarrollo
- 5 Software de reconocimiento
 - Herramienta utilizada
 - Elementos reconocibles
 - Prototipo 1
 - Prototipo 2
 - Prototipo 3
- 6 Software de control
- 7 Interfaz
 - Herramienta
 - Elementos incorporados
- 8 Conclusiones
- 9 Referencias

Introducción

- En la actualidad se emplean importantes esfuerzos por parte de grupos investigadores y fabricantes de automóviles para dotar sus vehículos de un sistema de conducción autónoma. Campo en la actualidad en pleno desarrollo y presentando multitud problemas aún por resolver.
- Entre los problemas existentes en los sistemas de conducción autónoma destaca su escasa adaptabilidad en carreteras reales donde existen multitud de imperfecciones, como pueden ser el desgaste o la falta de las señales viales, falta de iluminación o existencia de señalización provisional no fija como las indicadoras de obras en la vía. Estas circunstancias especiales impiden el correcto funcionamiento de los vehículos de dentro de unos niveles de seguridad mínimos aceptables para su implantación en carreteras transitables junto con vehículos conducidos de modo tradicional.

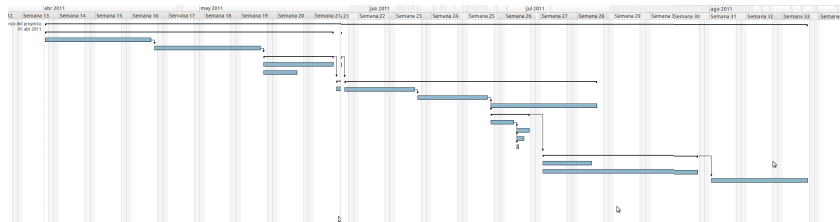
Objetivos

- El objetivo principal del proyecto es elaborar un software de detección de señales de tráfico mediante el análisis de imágenes.
- Junto al software irá asociado a un vehículo dotado de una cámara para permitir la visualización del terreno además de ser capaz de reconocer las señales de tráfico mostrando en el ordenador la última señal reconocida. Por otra parte, permitirá al vehículo actuar en consecuencia a la señal detectada.
- En resumen, se desea dotar a un vehículo de un sistema de conducción autónoma a partir de señales de tráfico.

Tareas

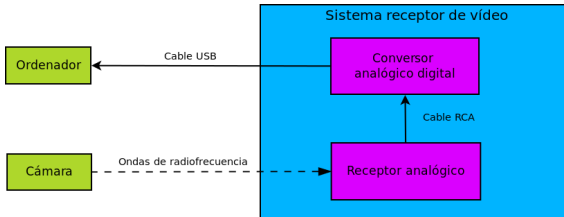
WBS	Nombre	Trabajo
1	SRV Traffic Console	132d
1.1	Investigación - aprendizaje previos	45d
1.1.1	Matlab pruebas	15d
1.1.2	Octave - C++ pruebas	15d
1.1.3	OpenCV	15d
1.1.3.1	OpenCV aprendizaje	10d
1.1.3.2	OpenCV pruebas	5d
1.2	Implementación algoritmo reconocedor OpenCV	35d
1.2.1	Prototipo 1 Reconocimiento	10d
1.2.2	Prototipo 2 Reconocimiento	10d
1.2.3	Versión final	15d
1.3	SRV-1 Puesta en funcionamiento	9d
1.3.1	Configuración vehículo	3d
1.3.2	Comunicación - control vehículo	3d
1.3.3	Implementación pad	2d
1.3.4	Implementación eventos teclado	1d
1.4	Desarrollo Interfaz gráfica	28d
1.4.1	QT aprendizaje	8d
1.4.2	Desarrollo	20d
1.5	Documentación	15d

Diagrama de Gantt

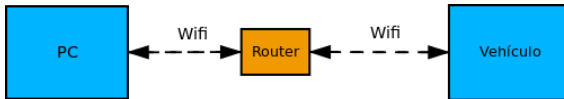


Comunicación

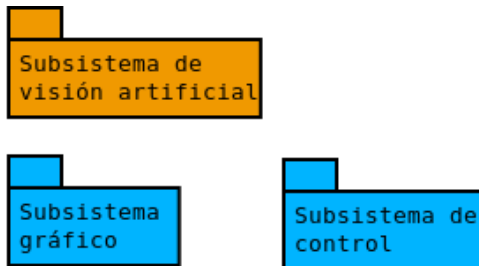
Las imágenes son captadas por el ordenador a partir de la cámara para su procesamiento.



Una vez procesadas las imágenes se envía la respuesta al vehículo.



Metodología de desarrollo



Modelo de ciclo de vida empleado:

- Desarrollo en cascada.
- Desarrollo por prototipos.

OpenCV

El software de visión artificial ha sido elaborado haciendo uso de la biblioteca OpenCV para el lenguaje C. Esta biblioteca incluye los elementos necesarios para el tratamiento de imágenes para proporcionar al vehículo de un sistema de reconocimiento de señales de tráfico.



Señales de tráfico

Las señales de tráfico detectables por el sistema son las siguientes:



Entre las señales incluidas disponemos las diferentes señales indicadoras de velocidad máxima, existiendo desde la de 20 km/h hasta la de 100 km/h con el objetivo de poder ajustar automáticamente la velocidad del vehículo en función de la señal detectada. Por otro lado existen las señales indicadoras de dirección obligatoria, con el fin de efectuar giros y la señal de stop con el propósito de efectuar paradas de manera automática.

Paso 1: segmentación por color

- Como paso inicial se realizó una segmentación por color siguiendo el modelo tradicional RGB:



Imagen de entrada (a).

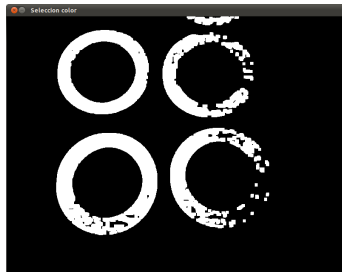
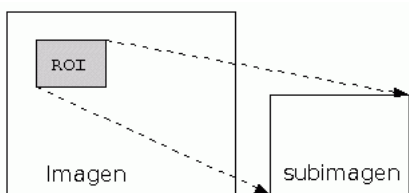


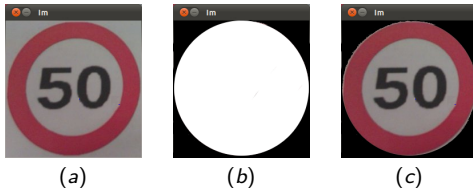
Imagen resultante (b).

Paso 2: extraer objetos

- Posteriormente se establecen las regiones de interés:

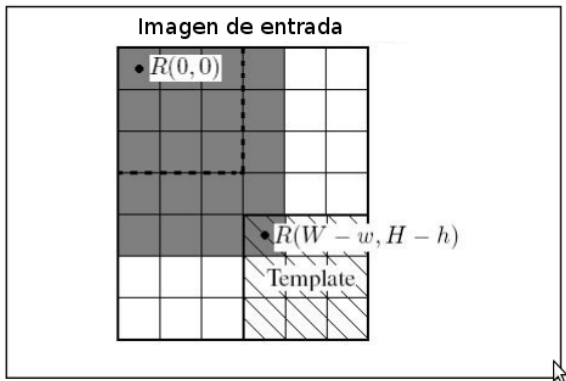


- Posteriormente se copia la imagen empleando una máscara:



Paso 3: clasificación

Para la clasificación se ha utilizado una técnica denominada Template Matching o Comparación de Plantillas:



Problemas detectados

Los problemas que han llevado a cabo el rechazo de este primer prototipo han sido los siguientes:

- El problema de la utilización de una segmentación por color siguiendo el modelo RGB es que sufre una extrema sensibilidad de variación de los valores de un determinado color a las condiciones lumínicas del entorno. Esto ocasiona que el prototipo desarrollado solo sea funcional en un único entorno, por ejemplo, la sala donde se ha desarrollado y probado, dejando de ser funcional en otras situaciones y lugares siendo necesario idear una nueva metodología de segmentación para el siguiente prototipo.
- El problema del Template Matching es que ha de comparar muchas características (para él, un píxel es una característica), y si tenemos en cuenta que en la base de datos encontramos N señales, observamos que con este método el resultado no ofrece garantías de funcionamiento en tiempo real, condición indispensable del proyecto.

Reconocimiento de patrones

- Para este segundo prototipo se emplearon técnicas propias del reconocimiento de patrones.
- Para poder clasificar satisfactoriamente los elementos, es necesario un proceso de aprendizaje previo en el cual el sistema crea un modelo de cada una de las clases a partir de una secuencia de entrenamiento o conjunto de vectores de características de cada una de las clases.
- Debido a esta razón, debemos previamente, antes de realizar cualquier tarea de clasificación, elaborar un conjunto de vectores de características o también denominado matriz de modelos con el fin de cotejar las características del elemento a identificar con los datos de la matriz modelos. A la fase de obtención de la matriz de modelos se le denomina etapa de entrenamiento en un sistema de reconocimiento de patrones.
- Al conjunto de características de una clase se denomina patrón.

Reconocimiento de patrones

- Antes de proceder con la extracción de los datos se realiza una segmentación por color, aunque en esta ocasión se ha realizado siguiendo el modelo de color HSV.
- La aplicación de este modelo resulta mucho más ventajosa con respecto al modelo RGB debido a que HSV, (Matiz, Saturación, Valor) sólo precisa de un solo valor numérico para detectar el color, desde tonos relativamente ligeros hasta los tonos más oscuros centrándonos en el matiz y la saturación.
- Entre los datos extraídos disponemos:
 - Forma de la figura: círculo u octógono.
 - Perímetro del objeto/s interior/es.
 - Área del objeto/s interior/es.
 - Número de piezas interiores.
 - Número de vértices de las figuras interiores.

Reconocimiento de patrones

Todos estos datos obtenidos durante la etapa de entrenamiento conforman un patrón, en la tabla se puede apreciar los diferentes datos obtenidos para algunas de las clases del sistema:

Señal:				
Identificador:	1	2	3	4
Figura externa:	8	9	9	9
Perímetro:	506.877	1025.25	1565.18	1971.43
Área:	8384.5	12774.5	17323.5	21771
Número de vértices:	39	77	125	156
Piezas interiores:	4	2	2	2

Clasificación

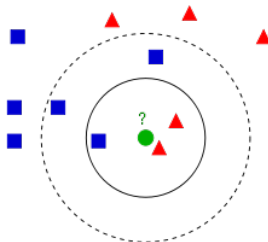
Por tanto la matriz de modelos obtenida queda de la siguiente manera:

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 8 & 9 & 9 & 9 \\ 506,877 & 1025,25 & 1565,18 & 1971,43 \\ 8384,5 & 12774,5 & 17323,5 & 21771 \\ 39 & 77 & 125 & 156 \\ 4 & 2 & 2 & 2 \end{pmatrix}$$

Y el patrón del elemento a identificar:

$$V_{desconocido} = \begin{pmatrix} 8 \\ 502,237 \\ 4022 \\ 34 \\ 4 \end{pmatrix}$$

Vecino más cercano



Se selecciona la clase correspondiente al elemento más cercano, las medidas son calculadas mediante la distancia euclídea.

$$d(x_i, y_j) = \sqrt{\sum_{r=1}^p (x_{ir} - x_{jr})^2}$$

Problemas detectados

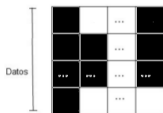
Los problemas que han llevado a cabo el rechazo de este segundo prototipo han sido los siguientes:

- Los datos obtenidos durante la extracción de características tales como área, perímetro, número de vértices, número de objetos interiores de la señal, entre otros, no resultaban determinantes y excluyentes a la hora de realizar la clasificación proporcionando un alto número de errores cuando el objetivo es elaborar un sistema lo más fiable y eficiente posible.
- Como punto positivo, se ha comprobado que el algoritmo del vecino más cercano presenta un funcionamiento más adecuado a las características exigibles del proyecto ya que proporciona una buena relación entre tiempo computacional y efectividad. Por otro lado, el algoritmo presenta la posibilidad de identificación de más de una clase.

Extracción de características

Para el último prototipo se ha empleado la misma técnica de clasificación pero trabajando con diferentes datos en la matriz de modelos. La matriz de modelos contendrá la imagen de los elementos interiores de las señales de tráfico dispuestos en columnas. Posteriormente se aplica la misma clasificación con el algoritmo del vecino más cercano.

Elemento número 1



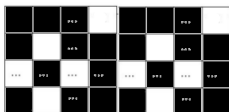
1. $n \times n$

Disposición de los datos en matriz columna



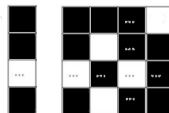
2. $n^2 \times 1$

Matriz de modelos resultante



4. $n^2 \times \text{número de clases}$

Añadido a la matriz modelos con el resto de clases



3. $n^2 * (m + 1)$

Clasificación

Para clasificar se realiza la llamada del algoritmo del vecino más cercano a cada objeto interno de la señal, en el caso mostrado son el número cero y el número cinco. Por tanto si dentro de un objeto circular de color rojo se localizan ambos elementos, determinaremos que se trata de una señal de 50 km/h.



Surveyor Robot Software

Para el control del vehículo se ha utilizado un software denominado Surveyor Robot Software, el cual ha sido desarrollado por John Cummins junto con los agentes de laboratorio de la Universidad de Brooklyn con la asistencia de M.P. Azhar, y la supervisión del profesor Sklar empleando el lenguaje de programación C++.

El código ha sido ligeramente modificado para adaptarlo a las necesidades del proyecto.



Qt

- Se consideró necesario proporcionar a la aplicación de una interfaz gráfica sencilla pero a la vez funcional teniendo como principal objetivo ofrecer al usuario, tras una visión rápida, la localización de los diferentes elementos para control del vehículo y deducir su funcionamiento.
- La herramienta empleada ha sido una biblioteca multiplataforma específica para el desarrollo de aplicaciones con una interfaz gráfica de usuario, dicha biblioteca es Qt, la cual emplea el estándar C++.



Interfaz

Los elementos incorporados en la ventana principal son los siguientes:

- Panel para el visionado de las imágenes captadas por la cámara.
- Conjunto de botones para manejo del vehículo, con imágenes representativas de su acción.
- Panel para el visionado de resultados donde se mostrarán aquellas señales de tráfico detectadas.
- Información acerca del detector de distancias.











Objetivos logrados

Durante la realización del proyecto *OpenTSR: Vehículo reconocedor de señales de tráfico* como trabajo de fin de carrera, he conseguido profundizar mi conocimiento en los siguientes campos:

- Ampliación de conocimientos sobre \LaTeX .
- Profundización sobre el lenguaje C y C++.
- Aprendizaje sobre programación por eventos.
- Diseño y creación de interfaces gráficas haciendo uso de la biblioteca Qt.
- Estudio de técnicas aplicables al procesamiento de imágenes.
- Estudio de técnicas y algoritmos propios del reconocimiento de patrones.

Referencias

-  Qt Reference Documentation
<http://doc.qt.nokia.com/4.7-snapshot/designer-manual.html>
-  Reference of OpenCV Language Library. <http://docs.opencv.org/>
-  Surveyor Corporation. http://www.surveyor.com/SRV_info.html
-  Francisco Palomo Lozano. Inmaculada Medina Buló. Gerardo Aburrizaga García. *Fundamentos de C++*. Servicio de Publicaciones de la Universidad de Cádiz, 2006.
-  Gerardo Aburrizaga García. *Make. Un programa para controlar la recompilación*
<http://www.uca.es/softwarelibre/publicaciones/make.pdf>
-  Gary Bradski. Adrian Kaehler. *Learning OpenCV Computer Vision with the OpenCV Library*. O'Reilly Media, 2008.
-  Ben Collins-Sussman. Brian W. Fitzpatrick. C. Michael Pilato. *Control de versiones con Subversion*. O'Reilly Media, 2006.
-  Wikibooks. *The Book of LaTeX*. <http://en.wikibooks.org/wiki/LaTeX>

Final

¿Preguntas?

