CC5508 - Procesamiento y Análisis de Imágenes

Reconocimiento automático de patentes vehiculares

Integrantes: Iván Rojas
Víctor Rojas

12 de Julio del 2013

Motivación

La identificación de la patente de un vehículo puede ser muy útil en distintas situaciones de la vida en una ciudad. Por ejemplo, control de entrada y salida en estacionamientos, flujo vehicular en una avenida, detección de autos robados, etc.

Es por ello, que este proyecto busca crear un sistema de reconocimiento automático de patentes, usando las herramientas que OpenCV ofrece y los conocimientos aprendidos en el curso.

Objetivos

- Identificación y recorte de la zona que contiene a la patente.
  - Segmentación.
  - Clasificación de patentes y no patentes.
- Eliminación de ruido de la imagen de la patente.
- Extracción de caracteres.
- Reconocimiento de caracteres.

Trabajo Realizado

Segmentación

Se probó con distintos métodos de segmentación, en busca de poder aislar la patente del resto de la imagen. Los métodos de binarización escasamente fueron de ayuda. Aunque eran capaces de dejar aislados los caracteres de la patente, como es el AdaptativeThreshold de OpenCV, eso no fue suficiente, porque no sabemos cuáles de todos los componentes en negro (o blanco) que deja la binarización corresponden a caracteres. Además, no en el 100% de los casos funciona bien, habiendo casos en que las letras simplemente pasan a ser parte del fondo (patentes sucias, foto con malos contrastes o malas iluminaciones).

Con respecto a los métodos de segmentación por color, probamos con MeanShift y Segmentación basada en grafos, siendo esta última la más útil. Aunque la segmentación no nos permite decir a ciencia cierta dónde está la patente, decidimos aprovechar su separación en componentes, y modificamos un poco el método basado en grafos. Haciendo la suposición de que la patente en la imagen no es tan grande, por lo tanto siempre habrá componentes de mayor tamaño que los que forman parte de la patente. Entonces, aquellos componentes de cierto tamaño en adelante se pintarán totalmente de negro. De esta manera, cuando se busque la patente, si se encuentra una zona con una cantidad de negro superior a un porcentaje, por ejemplo 70%, se descarta.

*Esta segmentación está implementada considerando una tolerancia al momento de unir o no componentes.

Ejemplos usando Segmentación basada en grafos, pintando en negro las componentes más grandes:

Al final, luego de todas las pruebas realizadas usando segmentación, se decidió no utilizarlo dado el tiempo excesivo que agrega al algoritmo.

Entrenamiento de Patentes y No Patentes

Selección de imágenes

El entrenamiento de las patentes se realizó mediante el uso de la librería de SVM de OpenCV.

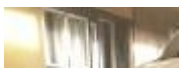
Para empezar, se recolectaron:

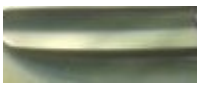
- 110 imágenes de autos.
- 110 patentes.
- 2330 Imágenes de NO patentes.

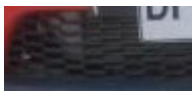
Esto con el fin de clasificar patentes usando dos clases (es o no es patente).

Patentes

No patentes







Primeros intentos de entrenamiento

Al conjunto de imágenes se les obtuvo distintas características, como:

- histograma de colores
- histograma de la binarización

tanto de la imagen completa como partes de ésta (particiones). Estos tuvieron mala precisión, debido a que no todas la imágenes de patentes se ven del mismo color, afectando las sombras, brillo, mala calidad, etc.

Desición final: HOG

En definitiva, se probó con una nueva alternativa: HOG. La imagen de entrada es escalada, para que todas tengan el mismo tamaño al momento de comparar. Los datos usados fueron:

window: 120x40

bloque: 20x20

celda: 10x10

stride: 5x5

n° bins: 9

El largo del descriptor es de 3780.

Búsqueda de patente usando *sliding window*

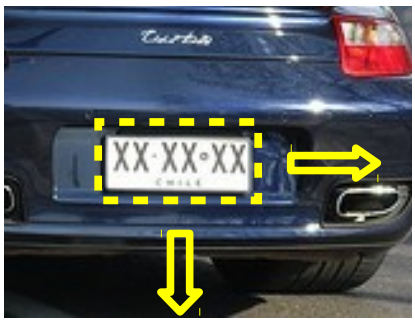
Teniendo ya un modelo entrenado, capaz de identificar si una imagen es patente o no, pasamos a la parte de encontrar la patente. Para esto utilizamos la metodología de *sliding window*. Recorremos la imagen, usando una ventana rectangular de un tamaño proporcional a una patente. Como no sabemos a priori el tamaño de la patente con respecto a la imagen de entrada, el recorrido se hace usando un rectángulo de distinto tamaño cada vez. La primera vez, el ancho del rectángulo es un tercio del ancho de la imagen, y la altura es de un tercio de su ancho. Si no encuentra la patente, el factor de división va aumentando en 0.5, con un máximo de 9, siempre suponiendo que la patente no será tan pequeña dentro de la imagen.

Se binariza cada trozo de imagen antes de descartarlo o no, lo que puede ser de igual ayuda. Se calcula el % de negro en el trozo, y si éste supera el 70%, se descarta el trozo.

Los trozos de la imagen no descartados por su porcentaje de negro, pasan a la siguiente prueba, ver si son o no patentes. Estos son contrastados con el modelo SVM calculado previamente, lo que nos dice a qué clase pertenece. Si pertenece a la clase de las patentes, se guarda ese trozo de imagen en un archivo y, al final del método, si ha encontrado una potencial patente, se detiene, sin probar con otros tamaños para la ventana.

* Si la imagen de entrada es directamente una patente, este método se reduciría a un simple contraste con el modelo SVM.

Ejemplo de búsqueda de patente usando *sliding window*:



Generación de dataset de caracteres

Para poder realizar el reconocimiento de caracteres, se eligieron 8 fuentes de letra disponibles en el sistema que eran las más similares a la fuente usada en las patentes chilenas. De estas fuentes, se usan tanto la letra normal como en el formato **bold** o **negrita**, para así aumentar y mejorar la variedad de los caracteres.

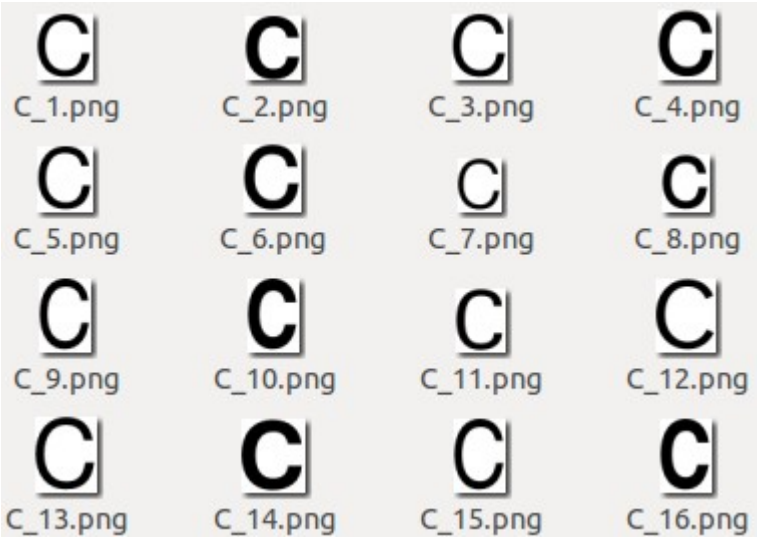
A B C D E F G H I J K L M

N O P Q R S T U V W X Y Z

1 2 3 4 5 6 7 8 9 0

A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z 1 2 3 4
5 6 7 8 9 0

Para la generación del set de fuentes, se tomó como entrada una imagen conteniendo las letras del abecedario y los números 0-9. Luego, se usó *findContours* y *bounding_box* para recortar cada letra y exportarla como archivo.



Reconocimiento de caracteres de la patente: KNearest

Entrenamiento de caracteres

Para el entrenamiento de reconocimiento de letras y números se usó el método KNearest que provee OpenCV.

Primero, todas las imágenes de los caracteres son escaladas a un tamaño definido. Luego a cada una de ellas se le obtiene el histograma que genera el método de HOG, y con ello se genera un conjunto de datos que se le entrega a Knearest, en conjunto con las clases de cada imagen.

Extracción de letras y números de la patente

Los pasos que se siguen son:

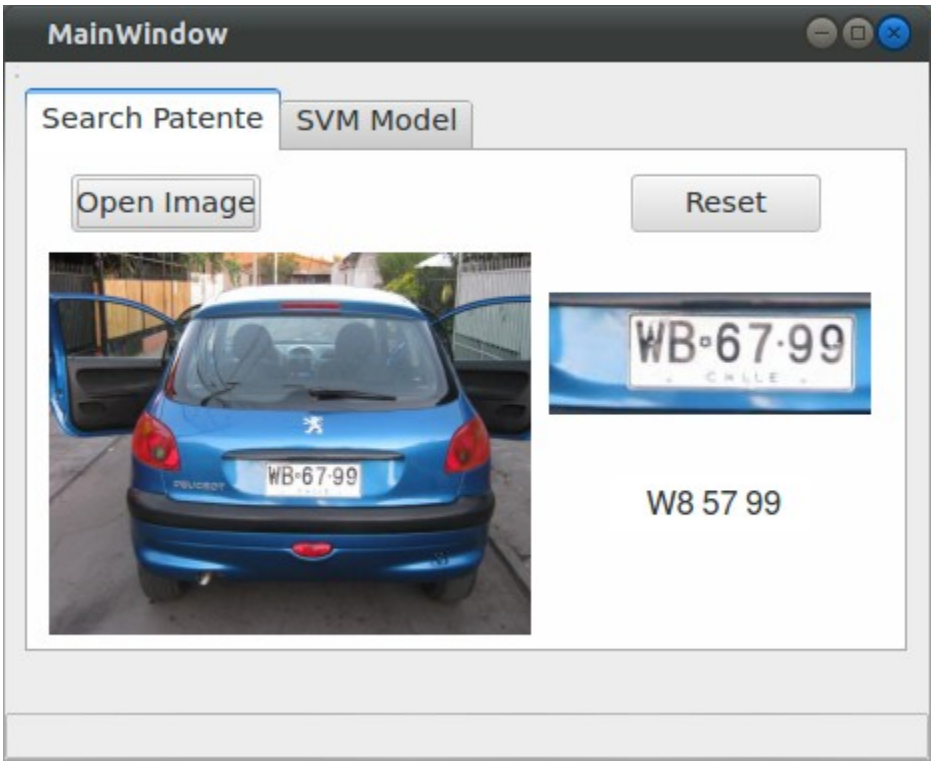
Se parte recorriendo la imagen de la patente con una ventana con un tamaño acorde para ir captando de a una letra.

Cada vez que se detiene la ventana, se obtienen el conjunto conexo más grande(ancho y alto) dentro de esa ventana. Para ello se aplica binarización con AdaptiveThreshold de OpenCV, el cual era el que entregaba el mejor resultado para las patentes en comparación a Otsu u otros métodos.

Obtenida la letra candidata, se pregunta a KNearest si pertenece a una de sus clases(caracteres). Si es así, se guarda y se prosigue.

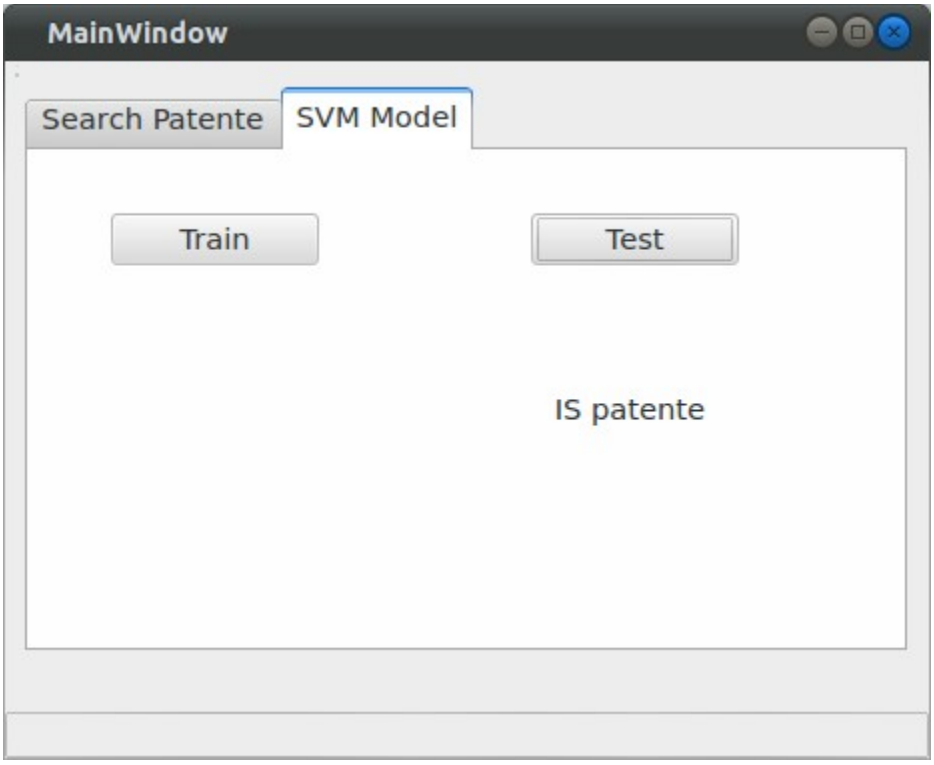
Interfaz Gráfica

Tab Search Patente



- Con Open Image se busca la imagen del auto.
- Al abrir la imagen, se muestra y se comienza de inmediato a buscar la patente.
- Cuando la encuentra la muestra en la parte derecha de la aplicación, como también las letras y números reconocidos.
- El resultado final no siempre es el correcto, pudiendo fallar en algunas letras.

Tab SVM Model



- Train entrena las patentes con SVM(modelo descrito en el archivo svm_model.cpp). Cuando termina imprime “Training finished”. Esto genera un archivo con los resultados.
- Test permite buscar una imagen y reconoce si es o no una patente.