Prototipo para la Medición Automática de la Velocidad de un Automóvil con Cámara de Video Mediante Procesamiento de Imágenes

Héctor I. ARISTIZABAL, José V. RESTREPO

Universidad Pontificia Bolivariana, Cir. 1 #70-01, B11, Medellín, Colombia.
hectorivan.aristizabal@alfa.upb.edu.co

Resumen: El propósito de este proyecto es el desarrollo de un prototipo para un dispositivo que permita medir la velocidad de un vehículo en tiempo real por medio del análisis de imágenes capturadas usando una cámara de video. Para lograr esto se aplica un algoritmo de sustracción de fondo y detección de contornos, para detectar el vehículo; A continuación se crea una plantilla del vehículo, para que su detección a lo largo del video sea eficiente. Por medio de análisis geométrico de la posición de la cámara, podemos medir el desplazamiento del vehículo y con el cálculo del tiempo por cada *frame*, se obtiene finalmente la velocidad a la que el vehículo transita. *Copyright © UPB 2013*

Palabras clave: OpenCV, Vehicle Tracking, cámara web, detección de objetos, seguimiento de imágenes.

Abstract: The purpose of this project is to develop a prototype for a real time vehicle speed meter based on the analysis of webcam captured images. The vehicle is detected by a background subtraction algorithm and edge detection. After that a Template of the vehicle is created to make the tracking more efficient. Using geometrical analysis of the camera position, the distance and displacement of the vehicle can be measured, and finally with the time difference between events, the speed can be calculated.

Keywords: OpenCV, Vehicle Speed, Image processing.

1. INTRODUCCIÓN

Antes de comenzar a hablar acerca de lo que veremos en esta tesis, quiero que pensemos un momento sobre cuáles son los principales problemas de movilidad vehicular que tiene su ciudad?, ¿Cuáles son las causas que incrementan los índices de accidentalidad?, ¿Qué se ha hecho para mejorar esto?

El número de vehículos se ha incrementado en los últimos años en las grandes capitales del mundo, estudios como el que hace anualmente BBVA aseguran que en Colombia en el sector automotriz por cada 100 habitantes hay 7,4 carros, lo que equivaldría a tener 3,4 millones de vehículos en todo el territorio nacional, cifra que este año (2013) está alrededor de los 4,6 millones y que tiene una tendencia de crecimiento considerable para los próximos años. [1][2]

A lo anterior sumamos otros factores que incurren en una desorganización de nuestro sistema de movilidad vial, como la falta de vías alternas, el alto flujo de carros en horas pico lo cual se ha venido controlando por medio del pico y placa, y uno de los más importantes, los incidentes de tránsito, donde se hace especial énfasis en este documento, ya que la alta velocidad de las personas al conducir causa los mayores problemas de circulación en las ciudades, es así, como mejorando estándares de seguridad que se han aplicado en varias ciudades del país y desarrollando procesos bajo nuevas tecnologías verán cómo se puede disminuir los accidentes viales.

Para el control y regulación de la velocidad existen diferentes tecnologías como los radares microondas, radares infrarrojos, de ultrasonidos, de captador magnético, y de lazos inductivos, sin embargo en Colombia adquirir tecnología de punta resulta muy costoso, y no se tiene una infraestructura claramente desarrollada para la ejecución de estas.

Medellín es la ciudad pionera en la implementación de foto-multas o comparendos electrónicos. Actualmente Sabaneta y Cali están incursionando en esta modalidad de control vehicular.

El inconveniente de este sistema es su costo, no solo la tecnología es de un precio considerable, sino que en el aprovisionamiento se deben llevar a cabo obras en las vías, ya que la técnica utiliza lazos inductivos en el asfalto para la detección de los vehículos. Son estos gastos lo hace que las demás ciudades del país tomen esta iniciativa como un proyecto a largo plazo.

En este documento conocerán una propuesta basada en desarrollo de software por medio de visión artificial, o también llamada visión por computadora, sub campo de la inteligencia artificial.

Este modelo estará al alcance de muchas ciudades, permitiendo que su implementación sea más económica, con un software más completo, que cumplirá con los estándares de medición de la velocidad. Los procesos están desarrollados en Java ® con la ayuda de la librería OpenCV, la cual contiene diversas funciones para el procesamiento y análisis de imágenes, dando un resultado interesante que recopilará datos que va a ayudar a tener indicadores del estado de las vías.

2. LENGUAJE DE PROGRAMACIÓN

2.1. Java®

En 1995 Sun Microsoft creo su primera plataforma en lenguaje Java®. Actualmente se ejecuta Java® en más de 850 millones de computadoras y en miles de millones de dispositivos. [3]

Es un lenguaje de propósito general, orientado a objetos y basado en clases, permite escribir programas en interfaz gráfica o textual. También se pueden correr programas de manera incorporado en navegadores de internet en forma de *Java® applets*.

Java® es un lenguaje abierto, no propietario, que puede ser implementado por muchas compañías y venderse como un paquete.

2.1. OpenCV

OpenCV es una librería de código abierto orientada principalmente para visión artificial en tiempo real. Esta librería corre en diferentes plataformas como Windows, Linux y Mac OS X, enfocada en C y C++ aunque existen desarrollos en otros lenguajes de programación. [4]

3. ÓPTICA GEOMÉTRICA IPHONE4

3.1. Problema

El objetivo principal es determinar el valor de conversión de pixeles a cm, el cual depende de la distancia de la cámara al objeto, un valor alfa el cual vamos hallar y el la distancia focal de la cámara, en este caso de un iPhone4.

3.2. Captura de imagen

Cuando se toma la imagen del objeto con una cámara, la distancia del objeto en la imagen con respecto a la lente no se conoce. Según el principio de formación de una imagen virtual se puede observar en la Figura 1.

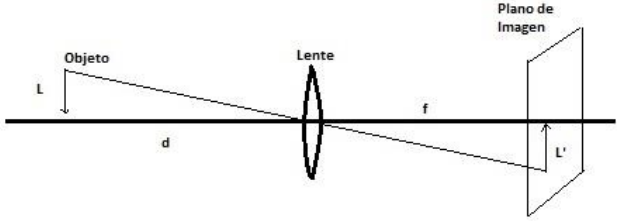

Figura 1. Formación imagen

Para obtener el tamaño del objeto real, es necesario conocer la distancia del objeto, la distancia focal y el tamaño del objeto proyectado en el plano de imagen. Estas variables se pueden se relacionan con la ley básica de la óptica

$$\frac{L'}{f} = \frac{L}{d} \quad (1)$$

donde L es el tamaño del objeto real, f la distancia focal, L' es el tamaño del objeto proyectado y d es la distancia al objeto.

En la Figura 1 se observa como es la proyección de la imagen en el plano proyectado, y su relación entre variables. [5]

3.3. Toma de datos

Para realizar pruebas y encontrar las constantes que son necesarias para encontrar el tamaño del objeto real, o el valor α de conversión entre pixeles y cm, se utilizó un vehículo como modelo, debido a su gran tamaño y pruebas a distancias donde normalmente estará la cámara de detección.

Se realizó varias tomas del vehículo a diferentes distancias con la cámara trasera del iPhone4, los resultados se observan en la Figura 2.

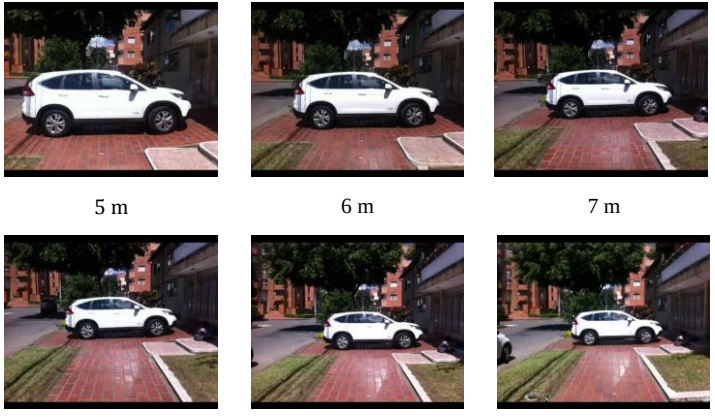

Figura 2. Pruebas de calibración

Para encontrar el factor de conversión se evalúa los datos en (1) para distintas distancias conocidas. Para el cálculo del tamaño del objeto proyectado se utiliza

$$L' = \propto \cdot n \quad (2)$$

donde n es el número de pixeles del objeto y α es el factor de conversión. Se remplaza (2) en (1) y se despeja el factor de conversión, obteniendo

$$\propto = \frac{L\,f}{d\;n} \qquad (3)$$

3.4. Resultados

Para las diferentes distancias encontramos valores del ancho del objeto *L'*. Con estos datos y utilizando (3), en donde se despeja el factor de conversión, se obtienen los resultados de la Tabla 1.

Tabla 1. Resultados de calibración

d (cm)	n (px)	L (mm)	f (mm)	α (px/cm)
500	281	4150	30	0.8861
600	241	4150	30	0.8609
700	205	4150	30	0.8675
800	179	4150	30	0.8694
900	165	4150	30	0.8383
1000	149	4150	30	0.8357
Promedio				0.8596

Haciendo un promedio de los valores de α, se obtiene un valor de conversión de $\alpha = 0.86$.

4. IMPLEMENTACIÓN

El desarrollo se implementó en *Java*®, con ayuda de las librerías de procesamiento de imágenes OpenCV. Las funciones que se utilizaron están incluidas en las librerías cv.h donde están los principales algoritmos, cxcore.h que contiene las estructuras básicas y funciones para graficar, higui.h para el manejo de ventanas además de entrada y salida de video.

4.1. Configuración inicial

En la configuración inicial, se obtienen variables como son el ancho y alto del *frame*. Se calcula la distancia euclidiana del punto donde está la cámara, hasta el punto de la vía donde van a pasar los vehículos. Además se configuran algunas variables como el ancho y altura máximo y mininos que tendrá aproximadamente los vehículos que van a transitar en la vía, esto se obtiene según la posición de la cámara.

4.2. Pre Procesamiento

El pre procesamiento es un paso fundamental en tratamiento de imágenes, es aquí donde se utilizan funciones para lograr la segmentación del objeto de la imagen. Esta parte del desarrollo se realiza la sustracción de fondo dinámica. Además se utilizan algunas funciones morfológicas para segmentar, y obtener una buena detección del vehículo como se observa en la Figura 3.

Figura 3. Pre Procesamiento

4.3. Procesamiento de Contornos

Después de haber obtenido la sustracción del fondo, y al haber aplicado algunas funciones de segmentación, se realiza un proceso de reconocimiento de contornos, para lograr detectar la silueta del carro en la imagen.

Para esto se utiliza la función de OpenCV llamada cvFindCountours, la cual encuentra todos los contornos del objeto, esta secuencia de contornos se dibujan mediante el ciclo y la función cvDrawContours.

Solo se dibuja los contornos mayores a un área mínima y menores a un área máxima, esto se realiza para evitar dibujar elementos u objetos que no son de interés. El área máxima y mínima se calcula dependiendo de la posición de la cámara esto se realiza en la configuración inicial.

Figura 4. Procesamiento Contornos

4.4. Detección

Este es el proceso más importante de todo el algoritmo, pues aquí es donde se debe detectar el vehículo que se va a seguir.

Para realizar esto calculamos de nuevos los contornos de la imagen y con una función que se llama cvBoundingRect buscamos en la imagen figuras semejantes a rectángulos.

Solo se acepta rectángulos con una altura y un ancho que sea aproximadamente igual al de un vehículo que pase por la vía. Estas variables se definen al inicio en la configuración inicial.

Cuando detectamos el rectángulo con las medidas correctas, creamos una plantilla del vehículo, para poder en los próximos *frame* seguir al vehículo.

Figura 5. Detección del vehículo

4.5. Seguimiento

Después de haber obtenido la plantilla, se dispone a seguir el vehículo en los siguientes *frames*, utilizando una función de OpenCV llamada cvMatchTemplate que compara el siguiente frame con la plantilla y el resultado lo almacena en una matriz.

Esta técnica se utiliza para encontrar áreas que coincidan o sean similares a en las siguientes imágenes. El método que se utilizó para la comparación fue el de Diferencia Cuadrática Normalizada

$$R(x,y) = \frac{\sum_{x'y'} [\,T(x',y') - I(x+x',y+y')]^2}{\sqrt{\sum_{x'y'} T(x',y')^2 \cdot \sum_{x'y'} I(x+x',y+y')^2}} \quad (4)$$

donde I es la imagen de entrada, T es la plantilla y R el resultado. Esta técnica arroja cero si la coincidencia es exacta, y uno cuando no existe relación alguna con la imagen. El valor varía entre cero y uno.

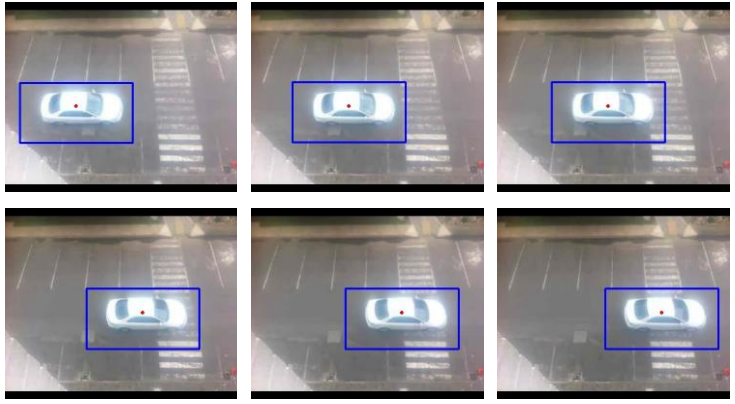

Figura 6. Seguimiento del vehículo

Después de obtener la matriz resultado, de la función cvMatchTemplate, se revisa que área tuvo mayor coincidencia con la plantilla. Para esto se utiliza la función cvMinMaxLoc, que encuentra el valor máximo y mínimo de la matriz resultado, además como la posición donde se encuentran estos valores. El valor que se utiliza para el seguimiento depende del método utilizado, en este caso utilizamos el de Diferencias Cuadráticas, por lo tanto el valor que utilizamos es el mínimo global que indica la probabilidad máxima de similitud.

4.6. Cálculo de velocidad

Para obtener la velocidad en la que pasa el vehículo por la cámara es necesario tener un punto de referencia del vehículo, por lo menos en dos *frames*. Se calcula la distancia euclidiana entre los dos puntos, que es la distancia recorrida. El tiempo transcurrido entre los *frames*, con un tiempo aproximadamente entregado por la cámara de 15 cuadros por segundo, se obtiene la velocidad del vehículo instantánea en cada *frame*. Se utiliza la fórmula básica de velocidad

$$v = \frac{d}{t} \tag{5}$$

donde *v* es la velocidad, *d* es la distancia recorrida entre dos puntos, y *t* es el tiempo transcurrido en recorrer ésta distancia.

La distancia entre dos puntos se obtiene aplicando

$$dist_{rec} = \frac{\propto n\, d}{f} \tag{6}$$

donde *n* indica el número de pixeles entre los dos puntos, $\propto$ es el valor de conversión obtenido en el capítulo 3, *f* es la distancia focal del iPhone4 y *d* es la distancia de la cámara al punto donde pasa el vehículo en la vía.

La distancia recorrida se encuentra en decímetros. Cada *frame* se muestra aproximadamente cada 66,67 ms, como se puede observar

$$t_{frame} = \frac{1000\ \text{ms}}{15\ \text{frame}} = 66{,}67\ \text{ms/frame} \tag{7}$$

esto debido que en un segundo se recorren 15 *frames*.

Por último la fórmula de velocidad

$$V = \frac{\propto n\, d}{f * t_{frame}} * 360 \tag{8}$$

entrega el valor de velocidad requerido, realizando conversión de distancia y tiempo, y así poder obtener las unidades en kilómetros por hora.

4.7. Mecanismo de alerta por infracción de velocidad

La aplicación representara con un recuadro de color rojo a los vehículos que sobrepasen la velocidad permitida, dependiendo del lugar donde se encuentre la cámara (Avenida, calle escolar, calle dentro de un barrio, etc.), además se guarda las imágenes como evidencia de la velocidad transitada.

4.8. Integración

Finalmente cada uno de las etapas del desarrollo es integrado para lograr la aplicación final. Esta aplicación detecta y sigue vehículos *frame* a *frame*, obteniendo la velocidad instantánea en cada punto en tiempo real, debido a que el video reproducido por el computador proviene de forma *streaming*. Debido a que el flujo de datos es por esta vía es posible que algunos datos o *frames* se pierdan, es por eso que se decide obtener la velocidad del vehículo, haciendo un promedio de las velocidad instantáneas y desechando posible valores errados donde la velocidad encontrada no coincide con la velocidad encontrada en los demás puntos

5. RESULTADOS

5.1. Pruebas en balcón Residencial

Se realizaron varias pruebas desde un balcón residencial. La cámara se encontraba a una distancia aproximada de 10 m horizontalmente y 4.5 m de altura. Se utilizó como velocidad máxima 40 km/h para verificar detección de infracción del vehículo. V_r es la velocidad real del vehículo, V_m la velocidad medida y E_r es el error relativo.

Tabla 2. Pruebas velocidad casa

V_r (km/h)	V_m (km/h)	E_r (%)	Infracción
30	33.26	10.8	NO
40	46.17	16.7	SI
50	52.62	5.2	SI
58	57.7	0.5	SI

Como se puede observar en la Tabla 2, la velocidad obtenida por la aplicación es muy cercana a la real en velocidades altas, aproximadamente en 58 km/h donde el error relativo es cerca del 0.5 %.

La Figura 7 muestra la alerta que se genera en el sistema al detectar un carro a velocidades superiores a la permitida, en este caso, a 40 km/h. Estas imágenes son almacenadas en una carpeta que indica la fecha, hora y velocidad en la cual paso el vehículo, como prueba de la infracción.

Figura 7. Pruebas de velocidad 50 km/h

Tabla 3. Pruebas velocidad Universidad

V_r (km/h)	V_m (km/h)	E_r (%)	Infracción
20	21	5	NO
30	28.8	4	NO
35	33.1	5.4	SI
40	54	35	SI
40	42	5	SI

5.1. Pruebas en la Universidad Pontificia Bolivariana

Estas pruebas se realizaron desde el 4 piso de Ingeniería de la Universidad Pontifica Bolivariana, en donde la cámara encontraba a una distancia aproximada de 6.6 m horizontalmente y 11.94 m de altura. Se utilizó como velocidad máxima 30 km/h para verificar detección de infracción del vehículo.

Las velocidades obtenidas mediante la aplicación son generalmente buenas, obteniendo un error relativo aproximadamente del 5 %. En una de las mediciones de velocidad el error fue del 35 %, esto sucedió debido a que algunos *frame*s se perdieron cuando intentamos enviar el video en tiempo real al pc desde el celular iPhone4 vía *streaming*, por el sistema hace una detección errónea de la velocidad.

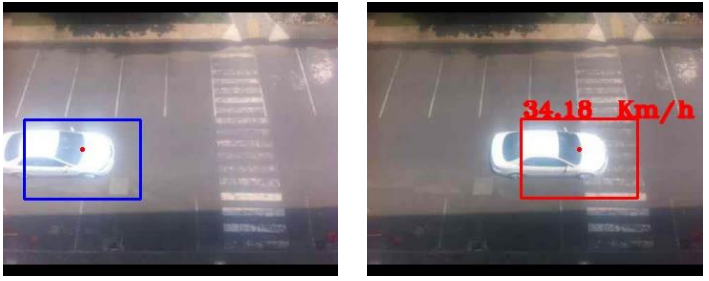

Figura 8. Pruebas de velocidad 35 km/h

En la Figura 8 se observa la imagen del vehículo desde el 4 piso del bloque de Ingeniería. La velocidad obtenida es de 35 km/h, por lo tanto se encierra en un cuadro rojo, indicando la alerta de infracción por violar el límite de velocidad.

6. CONCLUSIONES

El prototipo desarrollado en este trabajo es una herramienta útil para detectar, seguir y obtener la velocidad de los vehículos que transitan por una determinada vía. La aplicación se utiliza realizando una calibración sencilla que depende de la posición de la cámara respecto a la vía, además cuenta con un perfil de bajo costo, que solo requiere de un hardware donde se pueda correr la aplicación, junto a una cámara web, sin necesidad de hacer algún trabajo sobre la vía.

Al realizar las pruebas en dos ambientes y condiciones de posicionamiento de la cámara diferente se encontró un error relativo promedio del 6.5 % por encima de la velocidad real. Dentro de este valor existe un 5 % de incertidumbre dado que la velocidad del vehículo varía entre pequeños rangos. El 1.5 % restante se debe a la precisión de la distancia real de la cámara al punto de captura. Para un futuro se tendrán planes de mejora para precisar la instalación y utilización del sistema.

En algunos casos se identificó perdida de *frames* en la transmisión vía *Streaming* de los datos tomados desde el Iphone4, al portátil mediante la aplicación *WebCamera®*, lo cual generó valores de velocidad más elevados que la velocidad real del vehiculó.

Para facilitar el proceso de extracción de características de un objeto en una imagen, es necesario realizar un pre procesamiento usando técnicas de segmentación y operaciones morfológicas, los cuales ayuda a eliminar ruido y conservando las principales características de forma de los objetos.

Se comprobó que la técnica de extracción de características de alto nivel usando sustracción de fondo fue efectiva a la hora de detectar el vehículo, debido a que se trabajó con una cámara estática, lo cual facilitó la detección de la silueta del vehículo.

La librería OpenCV sobre procesamiento de imágenes, es útil para las soluciones de diversos problemas gracias a la gran cantidad de funciones que posee para el análisis, detección o seguimiento de objetos, sus ventajas sobre otras librerías, es gracias a su código abierto, lo cual reduce costos y evita problemas de propiedad intelectual.

Se utilizó *Netbeans* para programar en *Java®*, ya que los tiempos de ejecución son más rápidos con esta herramienta, lo que resulta de gran ayuda para este proyecto porque se busca que el procesamiento de imágenes sea en tiempo real.

Para trabajos futuros que complemente este sistema, se puede utilizar una cámara adicional que pueda detectar y capturar la placa del vehículo y así relacionar velocidad con la placa, obteniendo datos más completos para identificar a los infractores.

AGRADECIMIENTO

Agradezco a todos aquellos que de alguna u otra forma me apoyaron durante todo este proceso de formación, familia, compañeros y docentes.

Primero quiero agradecer a mis padres Iván Aristizabal y Helda Margarita Gomez, hermanos Juan Jacobo Aristizabal y Jacke Aristizabal.

También quiero agradecer a mi novia Diana Salazar por el apoyo incondicional durante el proceso de desarrollo de este trabajo de grado. Igualmente quiero agradecer a Sebastián Salazar por ayudarme en la realización de las pruebas de velocidad del trabajo.

Para finalizar quiero agradecer a mi director Valentín Restrepo por plantear este reto, su dedicación y guía durante la realización de este trabajo de grado, y a Carlos Andrés Arango por su disposición y colaboración.

REFERENCIAS

[1] El Espectador. (2012). *El sector automotor seguirá de plácemes en 2012*. Recuperado el 15 de julio de 2013, de http://www.elespectador.com/noticias/economia/el-sector-automotor-seguira-de-placemes-2012-articulo-319643

[2] BBVA Research. (2011). Situación Automotriz Colombia Recuperado el 9 de julio de 2013, de http://serviciodeestudios.bbva.com/KETD/fbin/mult/1301_SitAutomotriz_Colombia_Ene13_tcm346-364303.pdf?ts=1422013

[3] Java®. (2013) *Tecnoloía Java y sus necesidades* Recuperado el 29 de septiembre de 2013, de http://www.Java.com/es/download/faq/whatis_Java.xml

[4] OpenCV Wiki. (2013) *Introduction* Recuperado el 29 de septiembre de 2013, de http://OpenCV.org/

[5] Grupo de Tratamiento de Imágenes. (2002) *Geometría del proceso de formación de la imagen* Recuperado el 12 de octubre de 2013, de http://iie.fing.edu.uy/investigacion/grupos/gti/cursos/egvc/material/tema-3.pdf

AUTOR

Héctor Iván *ARISTIZABAL GOMEZ*, nacido en Cali, Colombia. Estudiante de décimo semestre de Ingeniería Electrónica de la Universidad Pontificia Bolivariana. Diplomado en Gestión de Proyectos de Ingeniería de la Universidad Pontificia Bolivariana (2013). Curso de controladores programables de la Universidad Pontificia Bolivariana (2012).

José Valentín Antonio *RESTREPO LAVERDE*, M.Sc. Ingeniero Electrónico graduado en 1998 de la Universidad Pontificia Bolivariana. Magister en Finanzas en 2009 de la Universidad EAFIT. Actualmente se desempeña como Profesor - Investigador de la Universidad Pontificia Bolivariana.