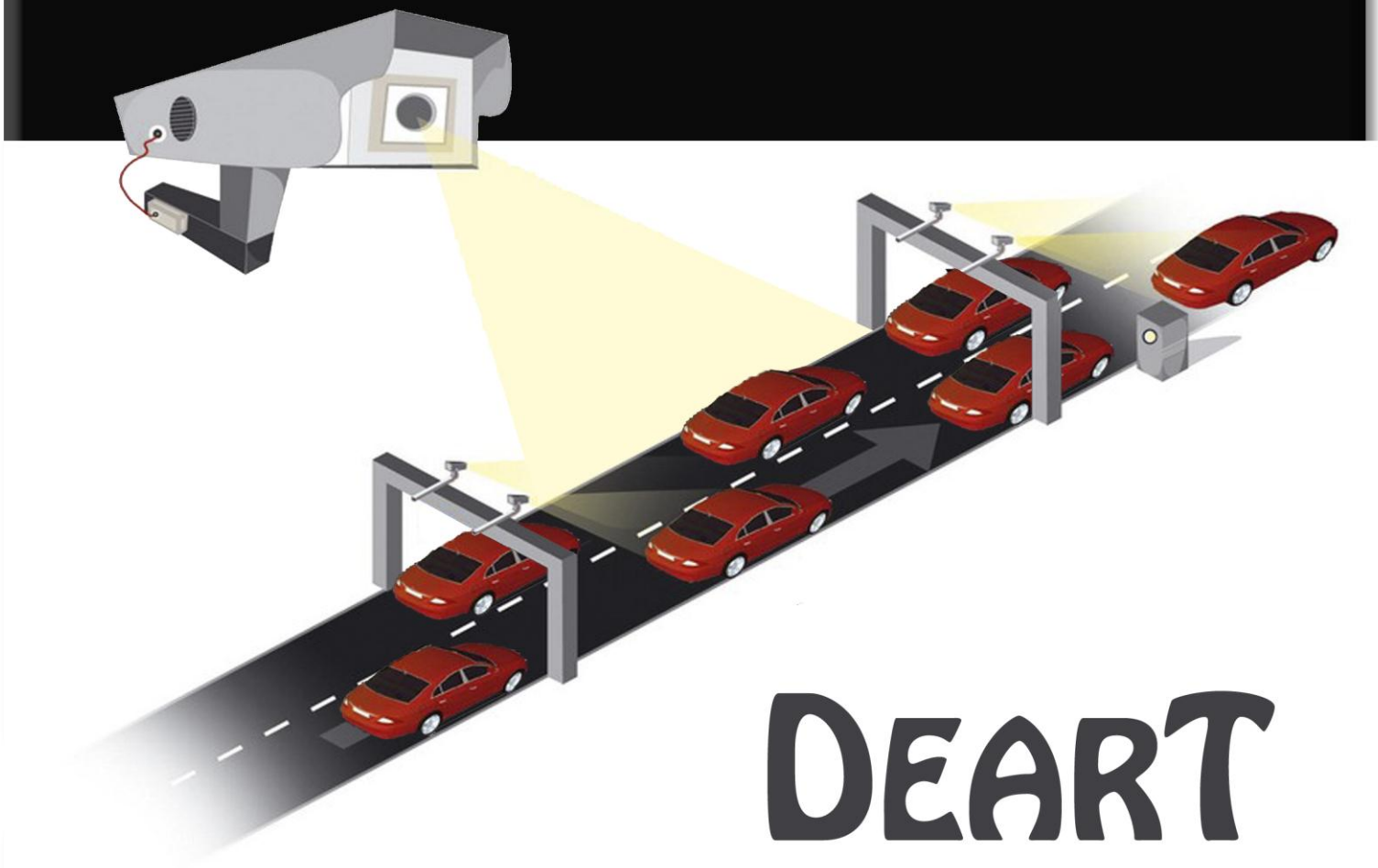
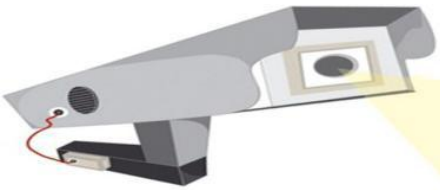
Detección y Eliminación Automática de Retenciones de Tráfico



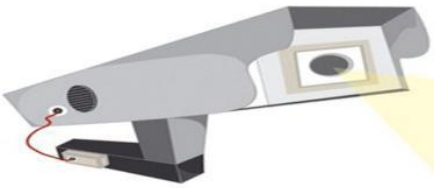
DEART

Premio nacional don bosco

Colegio: Salesiano "San Ignacio" Cádiz
Autores: Salvador Cordero Rando
Ricardo Romero Benítez
Tutor: Raúl Reyes Mangano



DEART

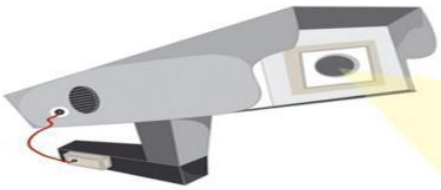


Introducción

Hoy en día, el tiempo de cada persona es muy preciado. El ritmo de trabajo, reuniones, etc. es muy elevado y mucho de ese tiempo se malgasta en las carreteras. Obras, vacaciones y otras situaciones cotidianas provocan retenciones de tráfico que conlleva pérdida no solo de tiempo, sino que también hablamos de un gasto innecesario de gasolina, que en los tiempos que corren es un bien casi de lujo. Más consecuencias negativas de todo esto son la contaminación y el desgaste mental y físico.

También, y adelantándonos a un futuro no muy lejano, nuestro sistema puede ser crucial para los nuevos coches eléctricos, ya que debido a su poca autonomía, el encontrarse en un atasco puede conllevar el gasto total de la energía del vehículo.

Por eso, con nuestro proyecto pretendemos crear una serie de medidas positivas tanto para los seres humanos, como para el medio ambiente.



Nuestro sistema busca reaprovechar las herramientas utilizadas actualmente por la Dirección General de Tráfico puesto que a través de sus cámaras, colocadas en sitios estratégicos, controlaremos en todo momento el flujo de tráfico existente, permitiendo interactuar con los paneles informativos dispuestos en las carreteras.

Inicialmente, y con los medios de los que disponemos, nuestro sistema serviría para carreteras compuestas de un carril en cada sentido más un carril central que se habilitaría en el sentido donde se produjese la retención.

Nos encontramos con un gran problema, y es la imposibilidad de probar nuestro sistema en un escenario real debido, evidentemente, a la falta de permisos, medios, etc. Por ello, hemos desarrollado un simulador muy básico de una carretera que nos posibilita formar un atasco para probar nuestro sistema. La metodología sería la misma, sólo que las instrucciones se enviarían a los paneles informativos reales.

Para desarrollar nuestro sistema hemos utilizado el lenguaje JAVA utilizando una serie de librerías de OpenCV diseñadas específicamente para el tratamiento de imágenes.

Posteriormente, analizaremos la posibilidad de instaurar este sistema para combatir otro tipo de retenciones.

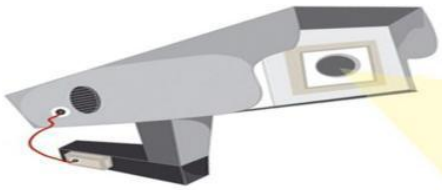
Objetivos

Como hemos comentado en la introducción, nuestro objetivo principal es detectar y eliminar las retenciones de tráfico mediante cámaras de tráfico y los paneles informativos.

Para ayudarnos a conseguir nuestra meta, antes es necesario cubrir otros objetivos específicos, como son:

- Desarrollar, mediante visión artificial, un software capaz de detectar cada uno de los vehículos que están circulando por la carretera.
- Establecer la densidad de tráfico actual a fin de determinar cuándo se está formando una retención.
- Ser capaz de, a través de los datos obtenidos de la densidad del tráfico, habilitar (mediante un panel informativo) un carril extra a fin de agilizar el tráfico rodado.

- Utilizar nuestro sistema para determinar cuáles son las vías más propensas a la formación de atascos y utilizar nuestro sistema para dar prioridad a los carriles.
- Obtener datos estadísticos sobre el tiempo de espera antes y después de implantar nuestro sistema.



Marco Teórico

¿Cómo se forman los atascos?

➡ http://atascos.com/about/

En la web vemos como un grupo de investigadores del MIT han desarrollado un modelo para saber en qué circunstancias se suelen formar un atasco. Estos matemáticos han llegado a la conclusión que las ecuaciones para determinar el movimiento y formación de los atascos es casi idéntica que el usado en la mecánica de fluidos en física.

Otros investigadores han establecido que la principal causa por la que se formen atascos es la aparición de frenadas prolongadas en las carreteras. Esto fuerza a los automóviles que se encuentran en una posición posterior al automóvil que ha tenido que frenar, a reducir su velocidad aún más que el coche que ha frenado en primer lugar.

OpenCV. ¿Cómo trabaja?

➡ http://alereimondo.no-ip.org/OpenCV/uploads/41/tema6.pdf

Una descripción amplia de cómo trabaja OpenCV para captar las imágenes cuando lo utilizamos a la hora de crear nuestro código fuente en C++. Su índice se compone en la segunda viñeta de la web, que nos irá explicando con detenimiento la elaboración de sus funciones:

- 6.1. Búsqueda de patrones.
- 6.2. Flujo óptico.

- 6.3. Integrales proyectivas.
- 6.4. Análisis del color.
- A.6. Análisis de imágenes en OpenCV.

Por último que añadir, al final del documento, nos explicará algunas funciones utilizables de la librería.

OpenCV. ¿Cómo usamos la librería?

➡ http://danielmonteropfc.wordpress.com/category/opencv/

Para empezar a utilizar la librería OpenCV tenemos esta web en la que nos explica con ejemplos paso a paso lo que debemos realizar para empezar a entender la biblioteca. Nos incluye, además, dos vínculos externos hacia otras web en la que nos explica aún más detalladamente las funciones.

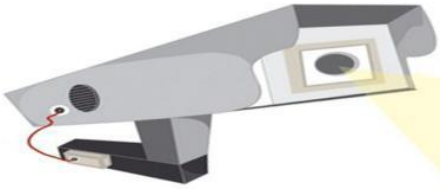
Sus apartados vienen divididos en 4, que son lo más básico de cada parte:

1. Primeros pasos
2. Convertir una imagen a escala de grises
3. Resta de imágenes y umbralizado
4. Detección de movimiento

➡ http://www.sebest.com.ar/?q=node/79

Más ejemplos de cómo usar nuestra librería, de lo más simple a algo relativamente más avanzado. Estos ejercicios son:

1. Abrir una foto con la librería OpenCV.
2. Dibujar un rectángulo en una imagen.



3. Crear una región de interés y mostrarla en otra ventana.
4. Cambiar el tamaño de una imagen.
5. Imprimir texto dentro del dibujo.

Comparación con otros trabajos

➡ http://www.robesafe.com/personal/pablo.alcantarilla/papers/Alcantarilla06pfc.pdf

Trabajo con el que compararemos el nuestro, siendo éste, parecido a la idea que tenemos en mente para nuestro proyecto. Utiliza la misma librería que las que usaremos (aunque nosotros no la usaremos nativamente en C++, si no en JAVA). Explica de qué forma lo ha realizado y trabaja OpenCV.

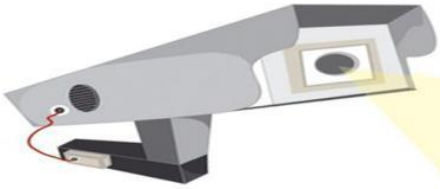
Viabilidad

En caso de obtener unos resultados satisfactorios, el sistema se podría implantar en las carreteras dado que se aprovecharían las infraestructuras existentes con las cámaras de la DGT y adaptar los sistemas a la recepción de la señal.

Los costes no serían demasiados elevados, puesto que sólo necesitaríamos un ordenador con una buena tarjeta gráfica y bastante memoria RAM para hacer el tratamiento de las imágenes.

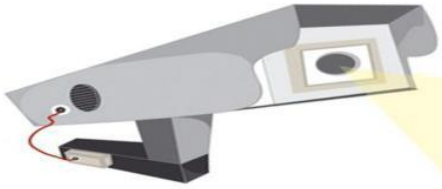
Justificación

Hemos escogido este proyecto porque pensamos que subsanar los atascos tendría consecuencias excepcionales tanto a nivel personal como medioambiental.



Consecuencias

Las consecuencias que esperamos obtener con este proyecto son, además de eliminar los atascos, reducir el consumo de gasolina con sus respectivas consecuencias en materia de contaminación y ahorro. También esperamos mejoras a nivel personal, como disminución del tiempo de trayecto, influencia en el ánimo, etc.



Software y Hardware

Para realizar nuestro proyecto hemos utilizado un ordenador con las siguientes especificaciones:

- Modelo: Toshiba Satellite L655
- Procesador: Intel I3-M380 a 2.53GHz
- RAM: 4GB DDR3
- Tarjeta gráfica: ATI RADEON HD5740 512MB

Los sistemas operativos utilizados han sido Windows 7 64bits y Ubuntu 10.10. El programa ha sido desarrollado en NetBeans bajo lenguaje JAVA. NetBeans es un entorno de desarrollo gratuito y JAVA es un lenguaje de programación orientado a objetos. También hemos tenido que utilizar unas librerías específicas para el tratamiento de las imágenes, que son las librerías OpenCV adaptadas para JAVA (JavaCV).

En cuanto al Hardware, hemos utilizado una Webcam Trust WB-6250X.

INSTALACIÓN

La instalación de NetBeans es trivial y no procederemos a su explicación. En cuanto a JavaCV, es necesario descargar las librerías OpenCV y JavaCV. Estas librerías no se instalan, sino que se descomprimen y se debe especificar las variables de entorno para que el sistema las localice en el caso de OpenCV; y en caso de las de JavaCV, en nuestro proyecto tenemos que añadir los .jar.

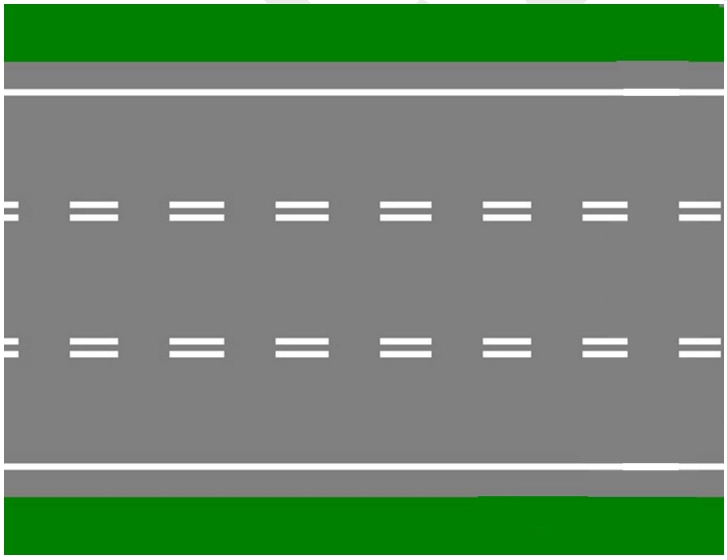
En cuanto a la Webcam, solamente tenemos que instalar los drivers que vienen en el CD de instalación y posteriormente calibrarla adecuadamente a nuestras necesidades.

Desarrollo

Inicialmente, nuestra idea fue la de hacer un software que fuese capaz de ir adaptando la velocidad de los coches progresivamente de manera que nunca se llegase a detener el tráfico o éste estuviese el menor tiempo posible parado. Nos encontramos pues ante el primer gran problema: la imposibilidad de probarlo en el mundo real. Por tanto, decidimos crear un simulador para intentar recrear lo que pasaría en la realidad.

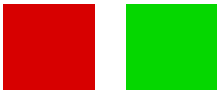
Debido al poco tiempo del que disponíamos y que el simulador estaba resultando muy dificultoso de realizar, decidimos reorientar el proyecto automatizando esta vez el funcionamiento de un carril reversible.

Para realizar nuestro simulador hemos utilizado los siguientes elementos:

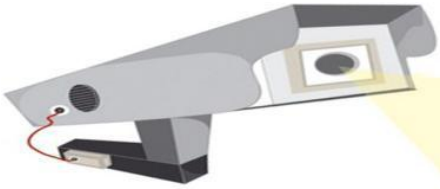


Carretera

Coches



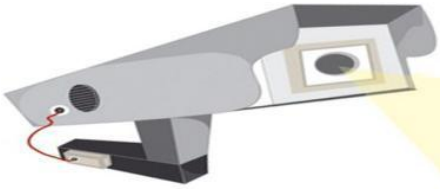
Semáforos



Para no complicar demasiado el simulador, dado que no es nuestro objetivo el desarrollarlo, hemos programado tres posibles situaciones. "Tráfico normal", en la que en ningún caso se producirá un atasco; "Atasco arriba", que provocará un atasco en el carril de arriba de forma que el primer coche se parará en un punto determinado y nuestro sistema a través de la cámara debe ser capaz de reconocer el número de coches y abrir el carril central; La última situación es la de "Atasco abajo" que hará lo mismo que la anterior pero en el carril de abajo.

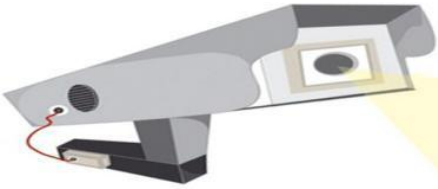
Como hemos dicho anteriormente, mediante la cámara nuestro sistema es capaz de reconocer el número de coches que hay en cada carril. A continuación explicaremos la forma en la que se hace:

- La cámara captura constantemente la imagen de la carretera.
- Mediante el código, vamos capturando frame a frame, por lo que obtenemos fotos del estado de la carretera.
- Ahora recorremos la imagen buscando cuantos cuadrados negros (coches) hay, y dependiendo de las coordenadas donde sean encontrados, pertenecerán al carril de arriba o al de abajo.
- Cuando el sistema detecta un número de coches determinado en un carril, interactúa con los paneles luminosos informativos abriendo el carril central en el sentido en el que se produce el atasco.



Evidentemente, el hecho de buscar el color en una imagen no sería una solución óptima a nuestro problema, pues en el mundo real hay coches de muchos colores, además de que por la noche sería difícil distinguir los colores. Para esto tenemos otra solución pero que no hemos podido implementar debido a fallos técnicos. Esta solución trata de capturar un frame y su siguiente, restar pixel a pixel de un frame y otro, y si hay píxeles que son diferentes se pintan de blanco, y si son iguales se pintarían de negro. De esta forma, un pixel blanco representaría un movimiento en la imagen. A este método se le conoce como binarización de imagen y solucionaría el problema de las cámaras durante la noche.

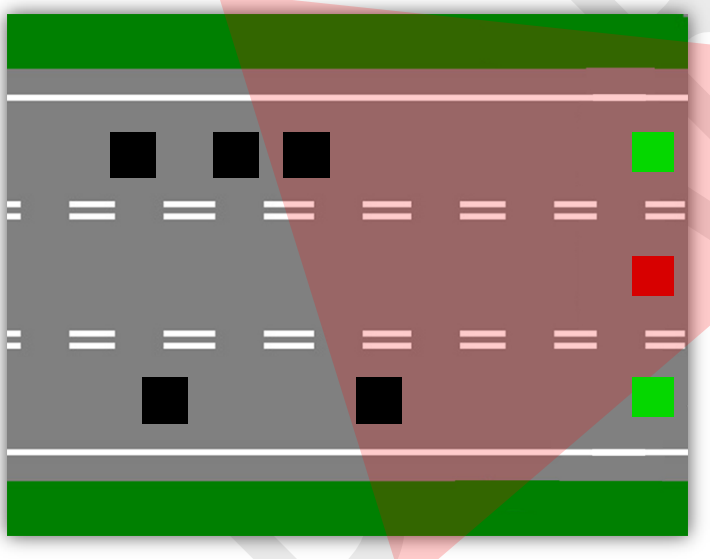
Además, nos encontramos con otro problema, que es qué pasaría en caso de lluvia, nieve o niebla.



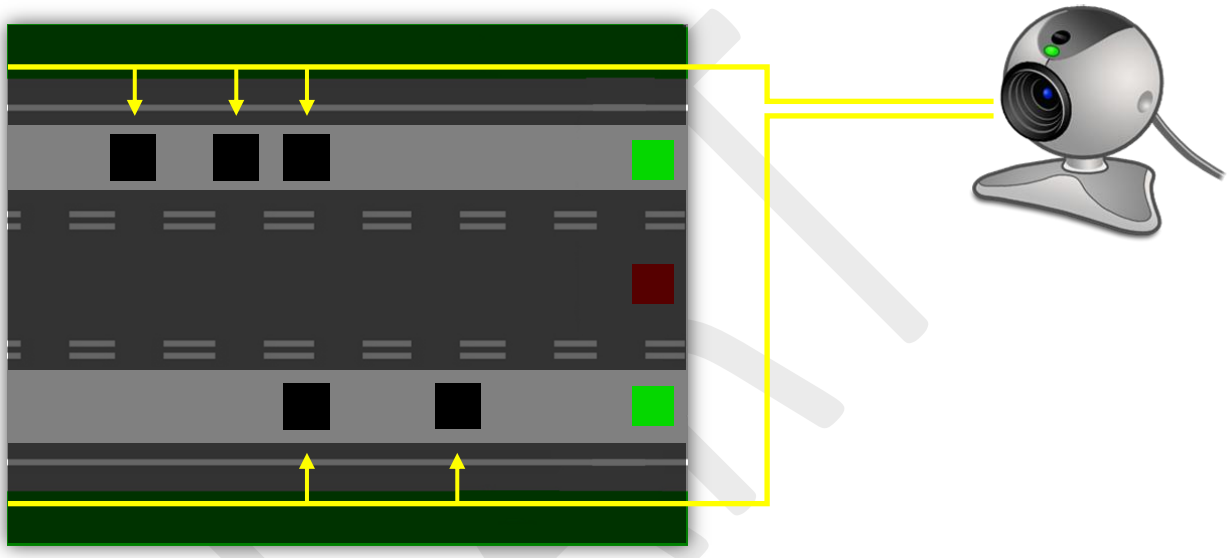
Funcionamiento

Como hemos comentado en el punto anterior, el sistema emula una carretera con tráfico en la que se pueden dar tres posibles situaciones: tráfico normal, atasco en el carril de arriba y atasco en el carril de abajo.

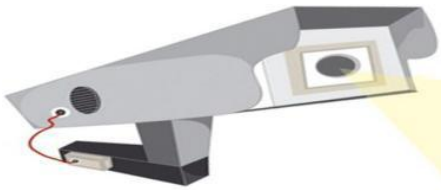
Nuestra webcam simula lo que sería la cámara de la DGT y estaría continuamente capturando la imagen de la carretera.



El programa va capturando imágenes de la carretera y recorre dicha imagen buscando el color negro. Para diferenciar un carril de otro, delimitamos las coordenadas que debe recorrer, porque si no necesitaríamos un ordenador más potente.



A medida que el programa reconoce los coches que hay en pantalla, va mostrando dicho número. Cuando el número de coches en un carril es lo suficientemente grande, o llevan un tiempo en la franja que captura la cámara, el programa pondrá a verde el semáforo en sentido del atasco y algunos coches se irán pasando al carril central, desapareciendo así la retención.



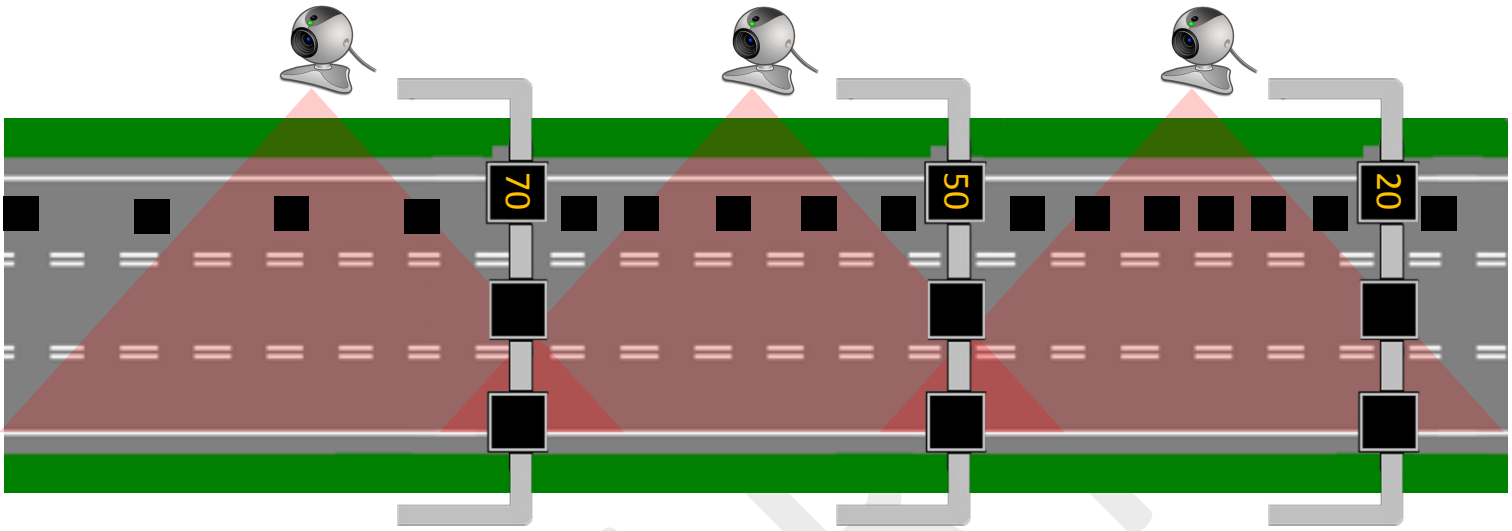
Mejoras

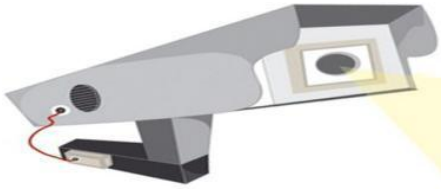
El sistema DEART tiene numerosas mejoras que se pueden aplicar en un futuro para aumentar su eficiencia.

La primera de ellas, y como hemos comentado anteriormente, existe la posibilidad de buscar el movimiento en la imagen. Es decir, binarizar la imagen de tal forma que todo movimiento que ocurra en el rango de la cámara, será captado por el programa pudiendo identificar todo tipo de vehículos.

En caso de inclemencias naturales, tales como lluvia o nieve, pensamos que se podría solucionar de forma que el programa capture el movimiento en un sentido en concreto. Es decir, con estas librerías nos permiten reconocer el sentido del movimiento, y suponiendo que la lluvia aparecería en nuestra cámara de arriba abajo, podríamos programar el código para que no capture dicho movimiento. De la misma forma ocurriría con la nieve.

Otra mejora sería ampliar el tipo de carretera en la que implantar dicho sistema, modificando también su comportamiento. Como nuestro planteamiento inicial, sería importante también implantar nuestro sistema en tramos largos de carretera, en los cuales, también mediante los paneles luminosos, ir adaptando la velocidad para evitar que los coches se paren.





Costes

Viéndolo desde un prisma objetivo, el sistema no resulta caro, puesto que ya aprovecha la red de cámaras DGT, que son de buena calidad.

En cuanto a los equipos, haría que tener uno por cada centro de recepción de imágenes, y estimamos su coste en:

Procesador: Intel I7 3.4GHz ≈ 300€

Memoria RAM: 8GB DDR3 ≈ 50€

Tarjeta Gráfica: 1GB dedicado GDDR5 ≈ 120€

Total (incluyendo demás gastos de torre, monitor, etc...) ≈ 750€/Equipo

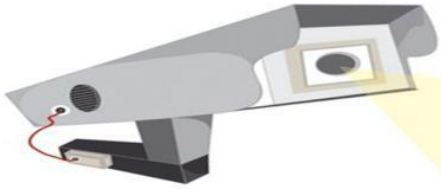
Todo el código utilizado en el programa es software libre, por lo que no lleva ningún coste asociado. Simplemente la mano de obra y el mantenimiento del sistema generarían los gastos. Estimamos estos gastos en:

Mano de obra: 40€/hora dedicada y persona (Estimamos unas 60 horas/persona)

Mantenimiento: 1.000€/año

Total ≈ 3.400€

Sumando las dos cantidades nos sale un total de 4.150€ aunque habría que incrementar en 750€ por cada equipo.



Conclusiones

Una vez conseguidos los primeros objetivos hemos obtenido una serie de conclusiones, algunas positivas y otras negativas. Las positivas son:

- Hemos conseguido el logro de poder automatizar el control de un carril a través de un semáforo, de forma que no haga falta tener una persona constantemente vigilando. Es cierto que el sistema puede fallar, por eso no elimina esos empleos ya que la figura de la persona sigue siendo importante.

- Aunque también debe haber concienciación de los conductores, es evidente que ayuda a eliminar el atasco.

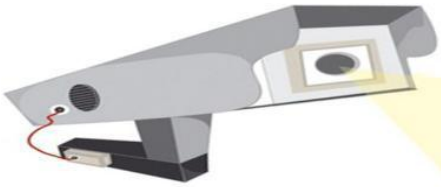
- Al eliminar el atasco, disminuye la contaminación por emisiones de los vehículos, así como ahorro de tiempo en carretera y demás connotaciones que hemos comentado anteriormente.

- Es un sistema relativamente barato.

DEART no es perfecto, y también hemos extraído una serie de conclusiones negativas:

- Al recorrer la imagen pueden haber ciertos errores a la hora de reconocer los objetos, puesto que esto también depende de sombras, luminosidad, etc.

- Hay que ajustar muy bien el algoritmo de reconocimiento, no pudiendo dejar prácticamente margen de error.



- Como dijimos anteriormente, también pueden ocurrir problemas si la carretera en cuestión se encuentra bajo fenómenos naturales tales como una lluvia, granizo, nieve o niebla, ya que la visibilidad disminuye considerablemente.

Bibliografía

- OpenCV (20/11/2011). *http://opencv.willowgarage.com/wiki/*. Instalación y manual de referencia de OpenCV
- OpenCV (20/11/2011). http://www.sebest.com.ar/?q=node/79. Ejemplos básicos
- Ceballos, Fco. Javier (1/12/2011). Java 2: Curso de Programación.
- JAVA (10/12/2011). http://zetcode.com/tutorials/javagamestutorial/. Tutorial de escenarios 2D
- OpenCV (21/11/2011). *http://opencv.willowgarage.com/*. Estructuras básicas.
- OpenCV (23/11/2011). http://opencv.jp/opencv-1.0.0_org/. HighGUI Reference Manual
- Montero, Daniel (23/11/2011). http://danielmonteropfc.wordpress.com/. PFC con OpenCV
- JavaCV (20/12/2011). http://code.google.com/p/javacv/. Instalación, manual de referencia de JavaCV y guía de transformación de OpenCV a JavaCV.
- Fernández Alcantarilla, Pablo (22/12/2011). PFC sobre monitorización del tráfico con OpenCV
- OpenCV (4/1/2012). http://opencvjaveriana.wikispaces.com/. PFC de conteo de vehículos con OpenCV