Polo del Conocimiento

Pol. Con. (Edición núm. 47) Vol. 5, No 07
Julio 2020, pp. 60-76
ISSN: 2550 - 682X
DOI: 10.23857/pc.v5i7.1495

Diseño de un prototipo para cobro de peajes con visión artificial

Design of a prototype for collection of tolls with artificial vision

Projeto de um protótipo para cobrança de pedágio com visão artificial

Mario Andrés Palma-Jaramillo [I]
mapalma@tecnologicoloja.edu.ec
https://orcid.org/0000-0003-4588-843X

Ramiro Vladimir Vaca-Moscoso [II]
rvvacam@unl.edu.ec
https://orcid.org/0000-0003-0146-2162

Yeferson Mauricio Torres-Berru [III]
ymtorres@tecnologicoloja.edu.ec
https://orcid.org/0000-0003-3784-3493

Daniel Engiberto Granda-Gutiérrez [IV]
degranda@tecnologicoloja.edu.ec
https://orcid.org/0000-0001-8726-0806

Luis Fernando León-Pinzón [V]
fernando.leon@tecnologicoloja.edu.ec
https://orcid.org/0000-0001-8342-0673

Correspondencia: mapalma@tecnologicoloja.edu.ec

Ciencias de la computación
Artículo de investigación

*__Recibido:__ 20 de mayo de 2020 *__Aceptado:__ 27 de junio de 2020 * __Publicado:__ 22 de julio de 2020

I. Máster en desarrollo de aplicaciones para dispositivos móviles, Máster en Gerencia y Liderazgo Educacional, Ingeniero en Sistemas, Docente del Instituto Superior Tecnológico Loja, Loja, Ecuador.
II. Ingeniero en sistemas, Universidad nacional de Loja, Loja, Ecuador
III. Máster Universitario en Ingeniería de Software y Sistemas Informáticos, Ingeniero en Sistemas, Docente del Instituto Superior Tecnológico Loja, Loja, Ecuador.
IV. Ingeniero Eléctrico, Docente del Instituto Superior Tecnológico Loja, Loja, Ecuador.
V. Ingeniero en sistemas, Docente del Instituto Superior Tecnológico Loja, Loja, Ecuador.

Resumen

El presente artículo muestra la creación de un prototipo para cargar peajes, es decir, un sistema base para el reconocimiento de placas de vehículos y, según el tipo de vehículo, se realiza una carga, utilizando las bibliotecas de visión artificial OpenCV, el lenguaje de programación C ++ y el entorno de desarrollo QT integrado. Para la implementación del sistema se utilizó la metodología ICONIX de peso pesado, ya que nos permitirá continuar haciendo iteraciones del sistema, además de la reutilización de los componentes debido al modularidad que establece. Las etapas más importantes del desarrollo incluyen el análisis, diseño e implementación del sistema.

Palabras claves: OpenCV; visión artificial; QT Ide; ICONIX, haarcascade.

Abstract

This article shows the creation of a prototype to charge tolls, that is, a base system for the recognition of vehicle plates and, depending on the type of vehicle, a charge is made, using the OpenCV machine vision libraries, the language of C ++ programming and the integrated QT development environment. For the implementation of the system, the heavyweight ICONIX methodology was used, since it will allow us to continue making iterations of the system, in addition to reusing the components due to the modularity it establishes. The most important stages of development include the analysis, design and implementation of the system.

Keywords: OpenCV; Artificial vision; QT Ide; ICONIX, haarcascade.

Resumo

Este artigo mostra a criação de um protótipo para cobrar pedágio, ou seja, um sistema básico para o reconhecimento de placas de veículos e, dependendo do tipo de veículo, é feita uma cobrança usando as bibliotecas de visão de máquina OpenCV, a linguagem de Programação C ++ e o ambiente de desenvolvimento QT integrado. Para a implementação do sistema, foi utilizada a metodologia pesada ICONIX, pois nos permitirá continuar fazendo iterações do sistema, além de reutilizar os componentes devido à modularidade que estabelece. Os estágios mais importantes do desenvolvimento incluem a análise, o design e a implementação do sistema.

Palavras-chave: OpenCV; visão artificial; QT Ide; ICONIX, haarcascade.

Introducción

La visión artificial o visión por computador intenta emular la capacidad de algunos seres vivos para ver una escena (imagen), entenderla y actuar en consecuencia. Existe un crecimiento en el número y tipo de aplicaciones industriales que demandan el uso de técnicas de visión artificial [1]

El propósito de este proyecto es la detección y reconocimiento del objeto placa vehicular en imágenes, mediante el reconocimiento de patrones y procesamiento de imágenes.

Se aplica la visión artificial a la automatización de cobros de peaje, ya que el modelo de negocio actual no es óptimo, en el caso del sistema manual (el más utilizado en el país), se coligió que: "El sistema de cobro de peajes en las carreteras del país no es el adecuado, generando congestionamiento vehicular e interrupción del tráfico vehicular". Esta solución todavía no ha sido implementada en ningún peaje del territorio nacional; en su primera versión efectúa el cobro de peaje y hace que los cobros sean transparentes al usuario a través de un sistema web, donde cualquier usuario con su número de placa puede consultar su historial de pagos por peaje; además, a los administradores de las estaciones se le permite el acceso a información estadística por flujo vehicular e ingresos por recaudaciones de peaje.

El prototipo construido realiza la detección y reconocimiento de placas. En la parte de detección se utiliza a la librería OpenCV que: captura la imagen de la escena natural, obtiene la región de interés (ROI), mediante el clasificador en cascada HAAR detecta la placa, con el método basado en ER (Extremal regions) del algoritmo basado en componente conexas se realiza la detección de regiones estables, se realiza el filtrado y agrupación utilizando en método de Neumann; finalmente se realiza la detección de bordes con el umbral adaptativo. En la siguiente parte referente al reconocimiento de placas se utiliza la librería Tesseract que: se obtiene una imagen procesada, se la binariza para analizar la estructura de la placa, se realiza la segmentación para finalmente el reconocimiento de caracteres.

Todo este trabajo y carga de procesamiento es transferida al ordenador, donde con mejoras de la programación se logra el reconocimiento óptico de caracteres de la placa, dejando así que el acondicionamiento del entorno de peaje se enfoque a una cámara de alta resolución específica para sistemas de visión artificial, correcta iluminación y un ordenador en la cabina de peaje.

Este prototipo de peaje tiene un software de escritorio programado en el lenguaje C++ que será ejecutado en cada ordenador de las diferentes cabinas de peaje que tenga la estación, las personas encargadas de su utilización son los usuarios: administrador y operador; básicamente este software automatiza todo el modelo de negocio de los peajes, aplicando las tarifas que se encuentran en el sistema de peajes del Ecuador, y la consulta de la placa se la hace directa al sistema de la Agencia Nacional de Tránsito del Ecuador [2] [3].

HERRAMIENTAS UTILIZADAS EN EL DESARROLLO

Los avances tecnológicos que ha tenido la informática son enormes, dándonos la capacidad de encontrar las herramientas para prácticamente cualquier cosa que deseemos construir, pero una cosa muy importante que debemos tener en cuenta o considerar al momento de usar una herramienta o librería es su licencia, aunque también es imperceptible porque siempre hay empresas u organizaciones además de desarrolladores en todo el mundo que contribuyen al mejoramiento y soporte de herramientas libres, como lo es el caso de OpenCV una librería que contiene miles de algoritmos para visión artificial, y tiene una licencia abierta, es decir que lo podemos utilizar, modificar o usar como nosotros lo necesitemos.

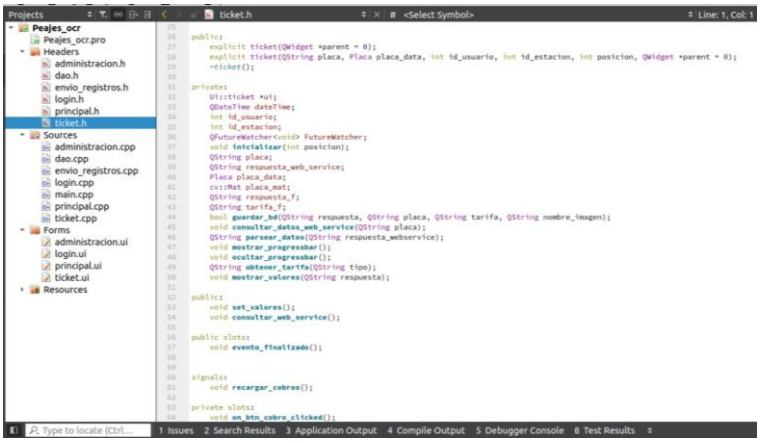
Para la construcción del sistema es necesario poseer conocimientos sólidos en programación, ya que todo el sistema está basado en el lenguaje de programación C++ y la mayoría de las herramientas usadas están basadas en él, como el entorno integrado de desarrollo QT Creator. Además de poseer conocimientos en bases de datos MySQL.

OpenCV se integra perfectamente con el lenguaje de programación C++ luego de realizar una compilación al mismo, en conjunto se logran obtener excelentes resultados, por lo que ha llegado a ser una de las librerías más usadas en lo que respecta a visión artificial, dada la efectividad y resultados obtenidos sobre procesamientos en tiempo real [3].

Es importante notar que para el desarrollo de este tipo de sistemas el rendimiento siempre es un factor importante y este se obtiene de la integración de las diferentes herramientas y lenguajes de programación, pero cabe recalcar que su importancia radica en saber aplicarlos de la manera correcta y acorde al resultado que se requiere obtener de ellos.

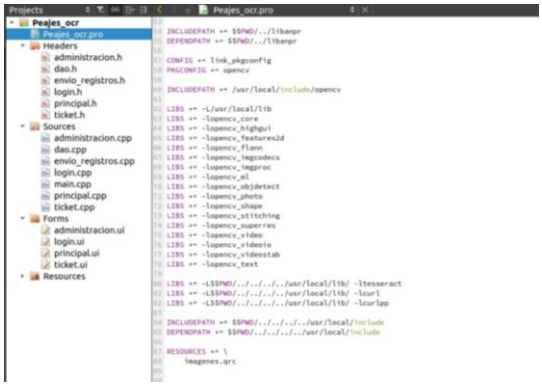
En la Fig. 1 se muestra la interfaz e inicio del entorno integrado de desarrollo integrado [*] utilizado para el desarrollo del sistema de cobro de peajes, en la cual también se muestra los formularios del sistema.

Figura 1. Interfaz principal del IDE QT Creator



Las librerías que contribuyeron en el desarrollo del sistema fueron llamadas como se muestran en la Fig. 2 previamente se debe tener instaladas todas las dependencias y las librerías OpenCV, además de un gestor de base de datos como lo es MySQL para la persistencia de la información generada, cabe indicar que el sistema operativo utilizado para el desarrollo e implementación es Ubuntu 16.04.

Figura 2. Archivo .pro incluyendo todas las librerías necesarias



Pol. Con. (Edición núm. 47) Vol. 5, No 07, julio 2020, pp. 60-76, ISSN: 2550 - 682X

IMPLEMENTACIÓN DEL SISTEMA

Para el desarrollo del sistema se utilizó la metodología ICONIX, debido a que su enfoque iterativo e incremental permite la integración de nuevas funcionalidades. A continuación, se detallan cada una de las etapas correspondientes a la metodología:

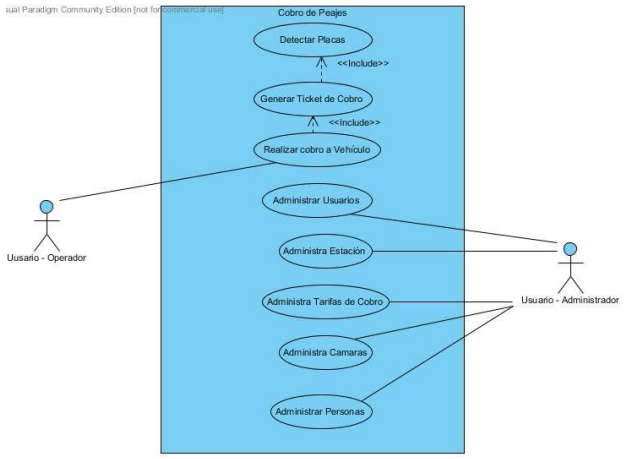
a) Análisis de requisitos: Al tratarse el sistema de un prototipo **se** debió cumplir con los requerimientos base de un sistema de este tipo, es decir se debió recopilar la información necesaria para poder cumplir con este objetivo y de esta manera determinar los requerimientos con los que debe cumplir el sistema. Esto fue posible debido a la documentación bibliográfica y la observación directa.

b) Análisis y diseño preliminar: Luego de haber realizado un primer análisis para el desarrollo del sistema, el siguiente paso es la determinación de los requerimientos. Entre los cuales tenemos los funcionales y no funcionales:

- **Requerimientos Funcionales**
  - Reconocer placas utilizando una cámara.
  - Reconocimiento de Caracteres de la placa detectada (OCR)
  - Mostrar la placa detectada.
  - Generar un cobro según el tipo de vehículo.
- **Requerimientos no funcionales**
  - Poseer una interfaz amigable e intuitiva.
  - Fácil uso por parte del usuario.

c) Diseño: Una vez determinados los requerimientos con los que deberá cumplir el sistema, se procedió a realizar el reconocimiento de todos los elementos que serán parte del sistema. Para ello se construyó el diagrama de casos de usos final en base a los requerimientos y análisis respectivos analizados, del cual partirá todo el sistema, como se muestra en la figura 3.

Figura 3. Diagrama de casos de uso



Además de este diagrama se construyeron artefactos de esta fase como son el diagrama de secuencia como lo muestra en la figura 4 y el modelo del dominio de la figura 5.

Figura 4. Diagrama de secuencia cobrar ticket

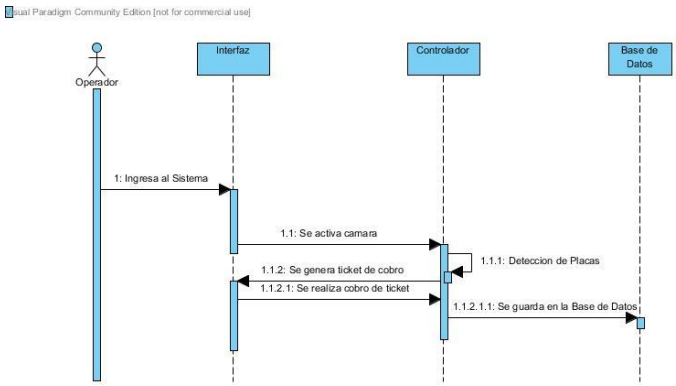
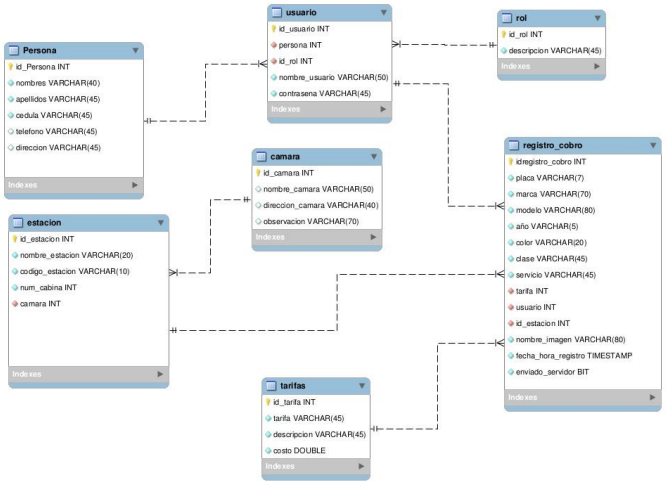


Figura 5. Modelo del dominio



d) Implementación: Con el análisis, el diseño y las funcionalidades del sistema claras, se puede empezar la construcción del software, se siguieron varias fases para la implementación del sistema, las cuales se detallan a continuación.

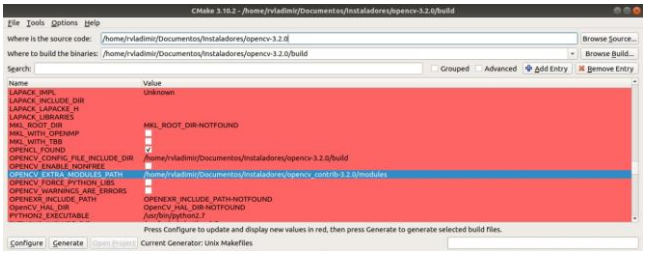
A. Integración de las herramientas

Como se mencionó anteriormente, para la construcción del software se usaron varias herramientas, entre las cuales se encuentra principalmente la librería OpenCV. Esta librería se encuentra escrita en el lenguaje de programación C++, pero para poder realizar la integración con el IDE Qt Creator se deben realizar una serie de pasos, y además para también poder utilizarlo en nuestro sistema Ubuntu, que fue el que se utilizó para el desarrollo [3].

La herramienta que nos permitió realizar la compilación de las librerías fue Cmake, un conjunto de librerías que nos permiten compilar librerías escritas en C++ para ser instaladas en nuestro sistema ya este Linux, Mac o Windows. Además de compilar nos permite integrar los módulos extra de OpenCV Contrib, estos módulos son desarrollados en paralelo, pero no son integrados de manera directa ya que son módulos aun no conocidos, o son desarrollados por la comunidad que mantienen el código.

A continuación, en la figura 6 se muestra la interfaz de Cmake con la inclusión de los módulos de OpenCV Contrib

Figura 6. Interfaz de Cmake con path de módulos extra



B. **Clasificador Haarcascade**

Una de las herramientas más importantes que provee OpenCV es la posibilidad de crear, entrenar y utilizar un clasificador en cascada. Este clasificador es parte fundamental ya que este contendrá la información necesaria que necesitaremos para realizar la detección de los objetos deseados en una imagen o secuencia de imágenes como lo es en un video. Se pueden utilizar varios clasificadores dentro de un mismo sistema, permitiéndonos ubicar no solo uno sino varios objetos de nuestro interés.

Figura 7. Carga del clasificador haarcascade en el proyecto

```
cargar_imagen_label();
this->progress->setValue(60);
QThread::msleep(500);
this->progress->setValue(70);
this->lib_api = new Libanpr("/home/ironvlack/Documentos/Tesis/Clasificadores/clasificadorPlacas.xml",
                            "spa","ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789",7,8,7);

this->progress->setValue(80);
QThread::msleep(250);
this->progress->setValue(90);
QThread::msleep(250);
ui->btn_pause->setEnabled(false);
this->progress->setValue(100);
```

Para la construcción del clasificador utilizado en este sistema, se utilizaron para el entrenamiento 1200 imágenes positivas que son las que encontrara positivamente en nuestras imágenes, y también se utilizaron 2000 imágenes negativas que caso contrario tomara como objetos erróneos dentro de nuestra escena y no las reconocerá.

C. **Algoritmo de detección**

En la figura 8 se puede apreciar los pasos a seguir para el tratamiento y detección de la placa vehicular y seguidamente el proceso para extraer el texto de la imagen.

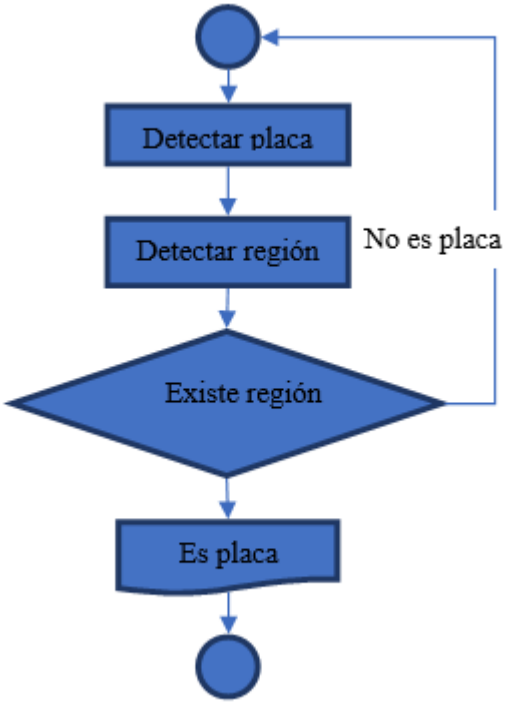


Figura 8. Diagrama de flujo empleado para la detección de placas

Es importante señalar que al ser un sistema en tiempo real se debió manejar los recursos del sistema de una manera adecuada, ya que todos los flujos se ejecutan muchas de las veces en paralelo al realizar procesamientos, y se puede llegar a provocar desbordamientos de memoria o ralentización en el funcionamiento del sistema.

Este tipo de sistemas por lo general generan gran cantidad de información, y esta debe ser almacenada de alguna manera, por lo que su utilizo una base de datos local que esta sincronizada con una base de datos en la nube lo que permitirá mantener un respaldo de toda la información, como también poner a disponibilidad de los usuarios que deseen acceder a ella a través de un sitio web.

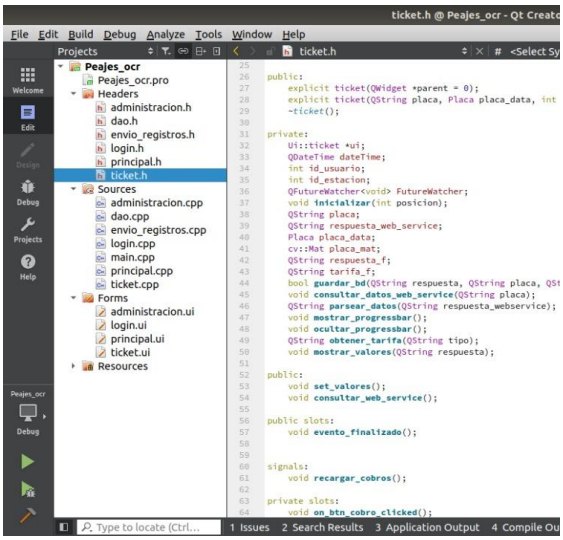
D. ***Implementación del código***

El sistema está escrito completamente en código de C++, el cual conjúntame con el framework Qt nos permitió crear todo el sistema y los algoritmos necesarios para que este funcione correctamente.

Es importante indicar que dada la metodología y la estructura que se llevó en el proyecto, las funciones base para el OCR y detección de texto así como funciones validadoras de las placas se realizaron en forma de librería, lo que permitirá que esta sea usada en los proyectos que sean necesarios sin comprometer la interfaz o al sistema aquí construido. Esto conlleva varias ventajas como el mejoramiento y optimizaciones posteriores del código y así mismo el mejoramiento de la interfaz o integración de nuevas funcionalidades.

Todos los formularios de administración además del frame principal fueron diseñados con la misma herramienta, lo cual agilito el desarrollo e implementación. Teniendo en cuenta también la organización del código y la estructura del lenguaje mismo se obtuvo una estructura de fácil comprensión y mantenimiento, a continuación, en la figura 9 se presenta la estructura final del proyecto.

Figura 9. Estructura generada en la estructura del proyecto



Pol. Con. (Edición núm. 47) Vol. 5, No 07, julio 2020, pp. 60-76, ISSN: 2550 - 682X

Resultados

Una vez concluida las fases de análisis, diseño e implementación se debe realizar la verificación, obteniéndose los siguientes resultados:

1. **Simulación en tiempo real:** En el sur del país es evidente el no encontrar peajes que realicen sus actividades de concesión debido a las regulaciones del gobierno local. En vista de esto se vio la necesidad de construir una maqueta a escala, que nos permita realizar todas las pruebas necesarias. A continuación, en la figura 10 se muestra la fase de construcción de la maqueta.

Figura 10. Proceso de construcción de la maqueta a escala

Una vez terminada la construcción de la maqueta queda de la manera como lo indica la figura 11, permitiéndonos posteriormente realizar todas las pruebas necesarias.

Figura 11. Prototipo final de maqueta de peaje

Se utilizaron además vehículos a escala conjuntamente con placas, para realizar las pruebas de funcionalidad.

Figura 12. Vehículo y placa a escala utilizados para las pruebas

2. **Detección de placas:** Evidentemente para la detección de los objetos, en este caso placas cabe indicar que el clasificador implementado es el pilar fundamental de todas las operaciones realizadas posteriormente ya que sin la ayuda de este no se hubiera podido extraer de las imágenes las placas que hubieren encontrado en ella.
   Posteriormente se probó el nivel de confianza del sistema, que, en condiciones de poca luz, ángulo de la cámara, placas con polvo o muy alejadas de la cámara, disminuye en 2% el reconocimiento del OCR; mas no de la detección y reconocimiento de la placa. A continuación, se resumen las pruebas y resultados obtenidos:

Tabla 1. Predicción del sistema para la detección de placas vehiculares

Imágenes de muestra	Placas detectadas	Nivel de confianza	% Error
50	48	95%	5%

Para la obtención de las imágenes se utilizó una cámara de alta definición Logitech C920 de 1080px, que nos permitió acercarnos a la calidad de las cámaras usadas propiamente para los sistemas de reconocimiento de placas vehiculares.

Figura 13. Cámara utilizada en el prototipo de peaje

En el sistema se puede evidenciar la detección de las placas, encerrándola en un recuadro para su fácil ubicación en la pantalla. Para minimizar el impacto y uso de memoria del computador se determinó una zona de interés en la captura de video debido a que un peaje es un entorno controlado y podemos suponer en que zona estará siempre ubicada la placa del vehículo, así como lo muestra la figura 14.

Figura 14. Reconocimiento de placa y zona de interés

3. **Generación de cobros:** Con la información obtenida de cada placa se puede realizar una consulta a la página de la ANT donde podemos obtener la información de cada vehículo y así saber a qué tipo pertenece para poder realizar el respectivo cobro [4].

   Para este proceso se tomó en cuenta la clasificación actual de cada uno de los vehículos existentes, dándole nosotros un costo a cada uno en base al tipo de vehículo al que pertenece, el sistema también mostrara un ticket de cobro en la interfaz del usuario indicando los datos, el tipo de vehículo y el costo a cancelar. Cada uno de estos cobros

realizados es almacenado en la base de datos local, para posteriormente ser enviado a la base de datos en la nube.

En cuanto al entrenamiento del clasificador detallado en Clasificador HAAR Cascades. para evitar esos falsos positivos se entrenó al clasificador con un conjunto positivo de 1200 imágenes y un conjunto negativo de 2000 imágenes, con un tiempo de entrenamiento aproximado de 2 días.

Conclusiones

Una vez terminado el desarrollo del presente artículo es preciso puntualizar algunas conclusiones a las que se llegó luego de haber realizado todos los procesos necesarios para la construcción del sistema, las mismas se describen a continuación:

El prototipo para cobro de peaje con visión artificial resultó en un software de escritorio en C++, que detecta y reconoce placas vehiculares usando: una cámara web como hardware de adquisición de información de la imagen placa; y un equipo de cómputo con el software de escritorio en la cabina de peaje. Estos dos componentes hardware y software en un entorno controlado como lo es la maqueta de peaje a escala, permiten el OCR de placas con un 95% de confianza.

La metodología ICONIX es importante debido a su robustez y a la agilidad con la que se realizan cada una de las fases. Otro punto importante de haber seguido esta metodología es la documentación que nos deja y que además nos permite realizar posteriores iteraciones para inclusión de nuevos módulos o refactorización y mejora del código ya existente gracias a su estructura modular con la que nos sugiere la construcción del software.

Las herramientas libres como: el framework Django versión 1.11.0, aplica la arquitectura MVT (Modelo, Vista, Template), que generó código limpio y organizado, ayudando en la rápida construcción del sistema, modularidad y escalabilidad; conjuntamente con Django Rest versión 3.7.7 se levantó el API que consume datos de la aplicación de escritorio que fue programada en el framework QT versión 5.10.0 ya que con la librería de elección Opencv versión 3.2.0 y C++ tiene facilidad de integración.

Las pruebas de funcionalidad aseguraron el correcto cumplimiento de las necesidades del producto especificadas en los requisitos funcionales y casos de uso, además, demostraron que el

software para su ejecución no demanda más que: 20% de consumo de memoria y 36.1% de consumo de CPU, como requisitos aproximados de procesamiento

Referencias

1. J. e. a. Vélez, Visión por Computador., Madrid, España: Dykinson., 2004.
2. Panavial: La Concesión. [En línea]. Available: http://panavial.com/concesion-administracion-red-autopistas-vias-carreteras-panamericana-vial-ecuador.php. [Último acceso: 15 12 2015].
3. OpenCV, 25 12 2015. [En línea]. Available: http://opencv.org/.
4. IBM: Transporte Inteligente. [En línea]. Available: http://www-05.ibm.com/services/es/bcs/pdf/transporte-inteligente-como-mejorar-la-movilidad-en-las-ciudades.pdf [Último acceso: 25 12 2015].

References

1. J. e. to. Vélez, Computer Vision., Madrid, Spain: Dykinson., 2004.
2. Panavial: The Concession. [Online]. Available: http://panavial.com/concesion-administracion-red-autopistas-vias-carreteras-panamericana-vial-ecuador.php. [Last access: 15 December 2015].
3. OpenCV, December 25, 2015. [Online]. Available: http://opencv.org/.
4. IBM: Intelligent Transportation. [Online]. Available: http://www-05.ibm.com/services/es/bcs/pdf/transporte-inteligente-como-mejorar-la-movilidad-en-las-ciudades.pdf [Last access: 25 December 2015].

Referências

1. J. e. para. Vélez, Computer Vision., Madri, Espanha: Dykinson., 2004.
2. Panavial: A Concessão. [Em linha]. Disponível: http://panavial.com/concesion-administracion-red-autopistas-vias-carreteras-panamericana-vial-ecuador.php. [Último acesso: 15 de dezembro de 2015].
3. OpenCV, 25 de dezembro de 2015. [Online]. Disponível: http://opencv.org/.
4. IBM: Transporte Inteligente. [Em linha]. Disponível: http://www-05.ibm.com/services/es/bcs/pdf/transporte-inteligente-como-mejorar-la-movilidad-en-las-ciudades.pdf [Último acesso: 25 de dezembro de 2015].