Universidad de las Ciencias Informáticas

Facultad 7

Título

Localización y segmentación de matrículas de vehículos en imágenes digitales

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores

Rafael L. Cardero Álvarez

Jesmar E. Fajardo Martín

Tutor

Msc. Héctor R. González Diez

Ciudad de La Habana, Marzo 2008

"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

_____	_____	_____
Rafael L. Cardero Álvarez	Jesmar E. Fajardo Martín	**Msc**. Héctor R. González Diez

DATOS DE CONTACTO

TUTOR: Msc. Héctor Raúl González Diez (***hglez@uci.cu***)

Profesor graduado de Licenciatura en Física Nuclear. Ha impartido las asignaturas de Física I, Física II, Matemática 3 y Matemática 4. Es profesor de la facultad 7 y se desempeña actualmente como Jefe de proyecto dentro del Grupo de Procesamiento de Imágenes (GPI) de la Universidad de las Ciencias Informáticas (UCI).

AGRADECIMIENTOS

Agradecer es una cualidad distintiva de los seres de buena voluntad. Agradezco a todos los pedagogos que me han formado y educado: *a ustedes y a la Revolución me debo*. Eterno agradecimiento al Msc. Héctor R. González Diez por ser ejemplo de humildad y por invitarme a conocer el fascinante campo de las Técnicas de Visión Artificial. A mis compañeros de aula: *gracias por su amistad.*

— Rafael.

A mis padres: por haberme educado, con tanto esfuerzo, del modo que hoy soy.
— **Rafael.**

RESUMEN

Los vehículos se identifican en Cuba, generalmente, de modo manual. Esto afecta la economía del país y disminuye la calidad de vida de la población. En los países desarrollados se utilizan los “Sistemas de reconocimiento de los caracteres de la matrícula” (ANPR) para identificar automáticamente los vehículos. Nuestro país no puede adquirir estos sistemas debido a las dificultades económicas existentes.

En esta tesis se diseñan en implementan los componentes “Localización de la matrícula” y “Extracción de los caracteres de la matrícula” de un sistema ANPR cubano. La sensibilidad, la especificidad, el índice de falsos positivos y el índice de falsos negativos, del componente de localización, es de 0.92, 0.35, 0.65 y 0.08 respectivamente. El componente de extracción ofrece una tasa de acierto de 78 %.

El componente de localización permite localizar matrículas de cualquier formato, es robusto ante la presencia de matrículas inclinadas, opera correctamente bajo varias condiciones de iluminación y es relativamente invariante a la transformación de escala. El componente de extracción utiliza una técnica híbrida en su funcionamiento. Ambos componentes fueron implementados utilizando los lenguajes de programación C# 2.x y Matlab.

PALABRAS CLAVE

ANPR, componentes conectadas, detección de bordes, matrícula, morfología matemática, proyecciones, Transformada Radon.

Tabla de contenidos

“La exactitud en la comunicación científica no es un asunto de vida o muerte; es algo mucho más serio que eso.”

—Robert A. Day.

INTRODUCCIÓN

La identificación automática de los vehículos es un problema relevante en la actualidad. Esto se debe a la utilización masiva de los vehículos en los procesos de la sociedad y a los efectos de la urbanización, entre otros factores.

Los componentes que intervienen en un sistema de identificación automática de vehículos (AVI), están organizados en 3 niveles (Fig 1).

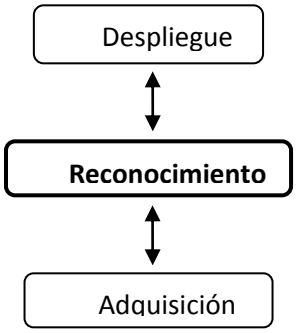


Figura 1. Estructura de un **Sistema de identificación automática de vehículos**.

El nivel “**Adquisición**” está compuesto por los dispositivos encargados de captar las señales y los datos de entrada. Algunos de estos dispositivos son las cámaras y los sensores. El nivel “**Reconocimiento**” contiene al artefacto capaz de identificar inequívocamente a un vehículo y el nivel “**Despliegue**” está compuesto por los elementos que permiten satisfacer las necesidades del usuario (e.g. control de acceso, monitoreo de autopistas).

El componente radicado en el nivel “**Reconocimiento**” constituye el núcleo del sistema de identificación y las bases de su funcionamiento son diversas. En algunos casos, el proceso de reconocimiento se logra mediante la incorporación de dispositivos especiales a los autos. Estos dispositivos emiten una señal única, que es leída por el componente de reconocimiento para identificar el

vehículo (Fig 2). Esta estrategia impone un alto costo de despliegue y no es robusta. Además, no permite que las personas, de modo natural, puedan identificar un vehículo (se necesita un dispositivo especial en el extremo receptor).

Figura 2. Estructura de un **AVI** basado en la incorporación de dispositivos especiales.

La matrícula (Fig 3), o patente de un vehículo, es una permutación de caracteres alfanuméricos que identifica e individualiza el vehículo respecto a los demás. Se representa en una placa metálica (que se conoce como placa de automóvil) en la que se graban o adhieren de forma inalterable los caracteres. En la mayoría de los países, los automóviles, así como los demás vehículos de una cilindrada de motor mínima, deben llevar sujeta una placa con la matrícula en la parte frontal y otra en la parte trasera. En algunas naciones solo se exige la placa trasera (Wikipedia 2008).

Figura 3. Ejemplos de matrículas utilizadas en algunos países.

La incorporación de matrículas, a los vehículos, constituye un método de identificación muy popular en la actualidad. El mismo permite que las personas puedan identificar los vehículos fácilmente Además, el desarrollo experimentado durante los últimos años en la Informática y en la Electrónica, ha facilitado la creación de componentes de reconocimiento que utilizan la matrícula para identificar, automáticamente, un vehículo. Estos componentes emplean técnicas de Visión artificial y son conocidos como "**Sistemas de reconocimiento automático del número de la matrícula**" (ANPR).

Los sistemas ANPR (Wikipedia s.f.) están compuestos, básicamente, por 3 módulos (Hansen, y otros 2002). El módulo **"Localización de la matrícula**" (Fig 4) es el encargado de localizar la matrícula del vehículo en una imagen digital de entrada. El módulo "**Extracción de los caracteres de la matrícula**" (Fig 5) recibe como entrada la imagen digital de una matrícula y produce un conjunto de imágenes digitales (cada imagen digital se corresponde con un carácter significativo de la matrícula).

Figura 4**.** Misión del módulo "**Localización de la matrícula**".

Figura 5**.** Misión del módulo "**Extracción de los caracteres de la matrícula**".

El módulo "**Reconocimiento de los caracteres de la matrícula**" (Fig 6) recibe la imagen digital de un carácter y retorna el código **ASCII** (American Standard Code for Information Interchange) de dicho carácter. Este módulo utiliza técnicas de reconocimiento óptico de caracteres (OCR).

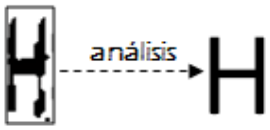


Figura 6. Misión del módulo "**Reconocimiento de los caracteres de la matrícula**".

El surgimiento de los sistemas ANPR se remonta al año 1976. En este año, el Departamento de Investigación y Desarrollo de la Policía de Inglaterra propuso un método para identificar automáticamente los vehículos, utilizando los caracteres de la matrícula. El primer prototipo de dicho método se comenzó a utilizar en el año 1979 (Wikipedia s.f.).

Los primeros fabricantes de sistemas ANPR industriales fueron **EMI Electronics** y **Computer Recognition Systems** (CRS s.f.). El último es un líder mundial y ha marcado hitos en la evolución de estos sistemas. Algunos de los hitos son el empleo de la detección de movimiento para estimar la posición del vehículo (año 1983) y la utilización de cámaras digitales **HDTV** [1](año 1997).

Los sistemas de identificación de vehículos, basados en ANPR, son muy utilizados en el presente. El límite de aplicación de los mismos, es igual al límite de la creatividad humana. Las aplicaciones más comunes son el control de acceso a las instalaciones, la localización de autos robados, el monitoreo del tráfico en las autopistas, y la detección de violaciones de las leyes del tránsito (e.g. violación de señal de pare, uso incorrecto del cinturón de seguridad, utilización de teléfonos celulares).

En Cuba, los vehículos son identificados, generalmente, de modo manual. El costo de adquisición y despliegue de los sistemas automatizados de identificación, resulta prohibitivo para un país con dificultades económicas como el nuestro. Además, las leyes dictadas por el Bloqueo (Cuba: Bloqueo) impiden u obstaculizan la obtención de estos novedosos sistemas. Lamentablemente, en el país no existe experiencia en el desarrollo de sistemas automatizados de identificación de vehículos.

La naturaleza manual, del proceso de identificación de los vehículos, está causando varios problemas en el país. Respecto al control de acceso, en las organizaciones se destinan varias personas para realizar esta actividad, que es sin duda muy tediosa. Además, a pesar del esfuerzo que realizan estos compañeros (responsables de seguridad), es frecuente la pérdida de los registros de datos y no existe un proceso de gestión de la información definido. Peor aún, en muchas organizaciones no se registran las entradas y las salidas. Esta información pudiese ser relevante ante la ocurrencia de hechos delictivos.

En cuanto al monitoreo de las autopistas, es justo señalar que se han adquirido sistemas automatizados para realizar esta actividad en las principales arterias, pero aún no se satisfacen las necesidades básicas. Hoy, en Cuba no existe un sistema integral que permita monitorear el tráfico en las

[1] Televisión de alta definición, por sus siglas en idioma Inglés.

autopistas y gestionar la información generada. Procesos de gestión esenciales como la localización de los vehículos y la estimación de la ocurrencia de embotellamientos, están ausentes.

Debido a los problemas expuestos y a la dificultades económicas existentes, en el país se necesita una solución nacional para identificar automáticamente los vehículos.

En esta tesis se aborda el problema ¿**Cómo localizar y segmentar matrículas de vehículos en imágenes digitales?** Las **Técnicas de procesamiento y análisis de imágenes digitales** constituyen el objeto de estudio de la investigación y se plantea como objetivo **Desarrollar los módulos "Localización de la matrícula" y "Extracción de los caracteres de la matrícula" de un sistema ANPR cubano**.

Para poder cumplir las metas trazadas, se han definido las siguientes tareas:

1. Analizar las tendencias actuales respecto a las técnicas de localización de matrículas en imágenes digitales.
2. Analizar las tendencias actuales respecto a las técnicas de extracción de los caracteres de la matrícula, en imágenes digitales.
3. Identificar las prestaciones que oferta el entorno Matlab para reconocer patrones en imágenes digitales.
4. Desarrollar un prototipo del componente "**Localización de la matrícula**", utilizando el entorno Matlab.
5. Validar la efectividad del prototipo "**Localización de la matrícula**", desarrollado en el entorno Matlab.
6. Desarrollar un prototipo del componente "**Extracción de los caracteres de la matrícula**", utilizando el entorno Matlab.
7. Validar la efectividad del prototipo "**Extracción de los caracteres de la matrícula**", desarrollado en el entorno Matlab.
8. Identificar las prestaciones que oferta la plataforma .NET para manipular imágenes digitales.
9. Implementar los componentes "**Localización de la matrícula**" y "**Extracción de los caracteres de la matrícula**", en un ensamblado .NET.
10. Demostrar que el ensamblado .NET creado es compatible con la plataforma libre Mono.

En la investigación se ha definido como campo de acción a las **Técnicas de localización y segmentación de matrículas de vehículos en imágenes digitales**, y la **imagen digital** constituye la unidad de estudio de la investigación.

La muestra es seleccionada aplicando las técnicas de muestreo no probabilísticas **Muestro intencional** y **Muestro accidental**. La naturaleza del objeto de estudio es abordada utilizando los métodos teóricos **Analítico-Sintético**, **Hipotético-Deductivo** y **Analítico-Histórico** y **Modelación**. Además se emplean los métodos empíricos **Observación** y **Entrevista**.

Lo que resta del documento está organizado como sigue. En el **Capítulo 1** se exponen las tendencias actuales en los métodos de localización y segmentación de matrículas de vehículos en imágenes digitales. Además, se fundamenta la utilización de los lenguajes C# 2.x y UML, se justifica la aplicación de los principios del **Proceso Unificado de Desarrollo** y se exponen los principales rasgos de las herramientas utilizadas durante la investigación.

En los **Capítulos 2** y **3** se presentan las características de los algoritmos de localización y segmentación de matrículas de vehículos, en imágenes digitales, propuestos en esta tesis. Luego en el **Capítulo 4**, se abordan las características del sistema.

El **Capítulo 5** está dedicado al análisis y el diseño del sistema. En el mismo se modelan las clases del análisis, las clases del diseño y los diagramas de interacción correspondientes a cada realización de casos de uso. A continuación, en el **Capítulo 6** se presentan los artefactos relacionados con la implementación del sistema.

En el **Capítulo 7** se exhiben los artefactos relacionados con el flujo de trabajo **Prueba**. Además, se reflejan la **sensibilidad**, la **especificidad**, el **índice de falsos positivos** y el **índice de falsos negativos**, de los métodos ANPR propuestos por los autores de esta tesis.

En la investigación se han utilizado varias funciones abstractas (Tabla 1). El identificador de cada una de ellas ha sido resaltado, en el documento, con el estilo de letra **negrita**.

Tabla 1. Funciones abstractas utilizadas en la investigación.

Identificador	**Sintaxis**	**Descripción**
height	height(**C**)	Retorna la altura del elemento **C**.
width	width(**C**)	Retorna la amplitud del elemento **C**.
length	length(**C**)	Retorna la cantidad de elementos contenidos en la colección **C**.
area	area(**C**)	Retorna la cantidad de puntos de dibujo, contenidos en el elemento **C**.
floor	floor(**C**)	Retorna el máximo entero, menor o igual que el elemento numérico **C**.
max	max(**C**)	Retorna el mayor elemento contenido en la colección **C**.

Capítulo 1

FUNDAMENTACIÓN TEÓRICA

En este capítulo se exponen las características lógicas de los sistemas de reconocimiento automático del número de la matrícula, y se demuestra la utilización de la segmentación por colores, la detección de discontinuidades y la Transformada Hough, para localizar la matrícula del vehículo en la imagen digital. Además, se explica cómo extraer los caracteres de la matrícula, mediante el análisis de las componentes conexas y de la proyección horizontal. Seguido, se justifica el uso de los lenguajes C# 2.x y UML 1.x. Posteriormente, se argumenta la elección del Proceso Unificado de Desarrollo como metodología de desarrollo. En el capítulo, además, se exponen los rasgos más significativos de las herramientas Visual Studio 2005 Team Suite, Rational Rose 2003, AQTime Profiler 4.x, MoMA 1.x, Sancastle 1.3.2.0 y Matlab 2007[b].

1.1. Sistemas de reconocimiento automático del número de la matrícula

Los sistemas de reconocimiento automático del número de la matrícula (ANPR) están constituidos, básicamente, por 3 módulos (MARTINSKY 2007) .

El módulo ***"*Localización de la matrícula*"*** (Fig 1.1) recibe como entrada una imagen digital y recorta, de dicha imagen, la región que se corresponde con la matrícula del vehículo. El módulo **"Extracción de los caracteres de la matrícula*"*** (Fig 1.2) recibe como entrada una imagen digital, que representa la matrícula, y retorna un conjunto de imágenes digitales (cada imagen digital del conjunto se corresponde con un carácter significativo de la matrícula). Por su parte, el módulo ***"*Reconocimiento de los caracteres de la matrícula*"*** (Fig 1.3) es el encargado de reconocer el carácter representado en una imagen digital. Este módulo recibe como entrada una imagen digital, que se corresponde con un carácter, y retorna el código **ASCII** *(American Standard Code for Information Interchange)* de dicho carácter.

Figura 1.1. Misión del módulo "**Localización de la matrícula**".

Figura 1.2. Misión del módulo "**Extracción de los caracteres de la matrícula**".

En algunos sistemas ANPR se implementa, además de los 3 módulos básicos, el módulo **"Análisis de la sintaxis de la matrícula"** (Shapiro, y otros 2003). El mismo recibe como entrada una cadena de caracteres y tiene el objetivo de aumentar la efectividad del sistema ANPR. Sus principales tareas son la corrección de los errores introducidos por los módulos anteriores (Fig 1.4) y la verificación de la sintaxis de la cadena de caracteres (la sintaxis es verificada después de corregir los errores en la cadena). Si la sintaxis es válida, la cadena es retornada hacia el contexto del usuario. En caso contrario, el sistema ANPR cede el control al módulo **"Localización de la matrícula"** y el proceso comienza nuevamente.

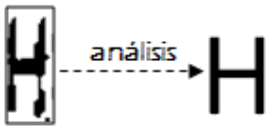


Figura 1.3. Misión del módulo "**Reconocimiento de los caracteres de la matrícula**".

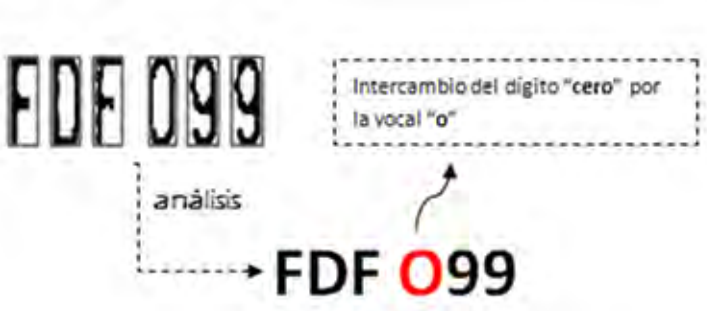


Figura 1.4. Error introducido durante el reconocimiento de los caracteres de la matrícula.

1.2. Tendencias actuales en las técnicas de localización de matrículas

En esta sección se demuestra la utilización de la segmentación por colores, la detección de discontinuidades y la Transformada Hough, para localizar la matrícula del vehículo en la imagen digital.

1.2.1. Localización mediante la segmentación por colores

La segmentación constituye la tercera etapa en un sistema de visión artificial (Fig 1.5). Los algoritmos de segmentación dividen la imagen digital en un conjunto de regiones **R**, donde los pixeles contenidos en la región $\mathbf{R_i}$ comparten atributos similares (Gonzalez y Woods 2002). Algunos de estos atributos son el color, el contraste y la textura (Acharya y Ray 2005).

Figura 1.5. Etapas de un sistema de Visión artificial.

Los algoritmos de segmentación por colores utilizan el atributo *color* para dividir la imagen digital en regiones. Las técnicas más comunes de segmentación por colores son la umbralización y la clusterización.

Los algoritmos de segmentación por clusterización ofrecen excelentes resultados, siendo k-means (MacQueen 1967) y mean-shift (Comaniciu y Meer 2002) unos de los más utilizados actualmente. Sin embargo, la complejidad temporal de estos algoritmos resulta prohibitiva para desarrollar sistemas ANPR.

Los algoritmos basados en la umbralización están caracterizados por el establecimiento de cotas o límites en el espacio de color[1]. Sean $\mathbf{P_A}$ y $\mathbf{P_B}$ dos vectores que delimitan un sub-espacio en el modelo de color, la región $\mathbf{R_i}$ estará compuesta por los pixeles de la imagen digital **I**, para los que se cumpla la expresión 1.1.

$$P_A \leq I(m.n) \leq P_B \qquad \textbf{(1.1)}$$

[1] Sub-espacio de un modelo de coordenadas, donde cada color se representa por un punto único (Molina 1998).

La umbralización es, generalmente, la técnica de segmentación por colores más utilizada para localizar matrículas de vehículos en imágenes digitales. Los algoritmos basados en está técnica, definen un espacio acotado del modelo de color (delimitado por los límites $\mathbf{P_A}$ y $\mathbf{P_B}$), que se corresponde con el color de la matrícula, y luego segmentan la imagen digital de acuerdo a la expresión 1.2.

$$\begin{aligned} I(m,n) = 1 &\longrightarrow P_A \leq I(m,n) \leq P_B \\ I(m,n) = 0 &\longrightarrow \text{caso contrario} \end{aligned} \qquad \textbf{(1.2)}$$

La imagen segmentada es sometida a un proceso de dilatación (Molina 1998) y luego se etiquetan las componentes conexas (Acharya y Ray 2005) existentes en la imagen resultante. A continuación, se seleccionan como posibles matrículas las componentes conexas que cumplan con los descriptores de las chapas. Algunos de estos descriptores son la relación de aspecto y la uniformidad de la proyección horizontal (Ron y Erez 2002).

Los métodos de localización de matrículas basados en la segmentación por colores poseen varios inconvenientes. Estos métodos causan que el sistema ANPR sea dependiente de la condiciones de iluminación (e.g. nublado, soleado) y que solo pueda reconocer matrículas de un único color. Además, bajo estas condiciones, el sistema ANPR es vulnerable en situaciones donde el color de la matrícula es similar al color de su contexto (e.g. vehículo de color blanco con matrícula de color blanco). A lo anterior, súmese que el costo de los dispositivos capaces de captar imágenes en color es elevado, comparado con el costo de los dispositivos que captan las imágenes en escala de grises.

1.2.2. Localización mediante la detección de discontinuidades

La detección de discontinuidades es, esencialmente, una operación de detección de los cambios locales significativos en la intensidad de la imagen (Molina 1998). Sea **I** una imagen digital, la magnitud y la dirección del gradiente, en el punto con coordenadas **I** (m, n), pueden ser calculados mediante las expresiones 1.3 y 1.4, respectivamente.

$$M = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad \textbf{(1.3)}$$

$$\theta = \operatorname{arctg}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad \textbf{(1.4)}$$

Los algoritmos de detección de discontinuidades están basados en el análisis del gradiente y en el análisis de la segunda derivada, de la intensidad de la imagen digital. Los métodos basados en el gradiente calculan la magnitud de la primera derivada **G** (m, n) en cada punto de la imagen digital. Luego, sea **T** un umbral, los puntos que cumplan la expresión $\boldsymbol{G}(m,n) \geq \boldsymbol{T}$, son seleccionados como puntos de discontinuidad. El problema principal de estos algoritmos radica en la dependencia del umbral **T** (el valor de **T** controla la cantidad de puntos de discontinuidad). Respecto a estos métodos, existen varios operadores para calcular la magnitud del gradiente (Gonzalez y Woods 2002). Algunos de ellos son el operador de Roberts, el operador de Prewitt y el operador de Sobel (Fig 1.6).

G_x

1	2	1
0	0	0
-1	-2	-1

G_y

1	0	-1
2	0	-2
1	0	-1

Figura 1.6. Máscaras del operador de Sobel.

Los algoritmos basados en la segunda derivada intentan suprimir la dependencia del umbral **T**, propia de los algoritmos basados en el gradiente. Es conocido que un máximo local en el gradiente se corresponde con un cruce por cero en la segunda derivada. Estos algoritmos seleccionan como unidades de discontinuidad, a los puntos en los que la segunda derivada de la intensidad experimenta un cruce por el eje horizontal. Los operadores de detección de discontinuidades —basados en la segunda derivada— más utilizados son el operador “Laplaciano” y el operador “Laplaciano del Gaussiano” (Molina 1998).

Los métodos de localización de matrículas mediante la detección de discontinuidades, están fundamentados en el principio de que los caracteres generan varios bordes verticales (Fig 1.7). En estos métodos, después de detectar los bordes verticales, se aplica un proceso de dilatación (Gonzalez y Woods 2002) y luego se etiquetan las componentes conexas (Acharya y Ray 2005). A continuación, las componentes conexas que cumplan con los descriptores de las chapas (e.g. relación de aspecto, uniformidad de la proyección horizontal) son seleccionadas como posibles matrículas.

Figura 1.7. Imagen de una matrícula. **a)** Original. **b)** Bordes verticales. **c)** Bordes verticales dilatados.

Los métodos de localización basados en la detección de las discontinuidades, permiten que el sistema ANPR sea capaz de reconocer matrículas de cualquier formato, lo que facilita el despliegue del sistema en varios países. Además, la complejidad temporal de estos algoritmos es relativamente baja. Lamentablemente, estos métodos no constituyen una panacea. El proceso de detección de los bordes verticales —y su posterior dilatación— puede generar varias componentes conexas, por lo que se requiere utilizar un conjunto eficaz de descriptores para decantar las regiones ruidosas. Sobre lo anterior, es vital que el conjunto de descriptores garantice el equilibrio entre el índice de falsos positivos y el índice de falsos negativos (Álvarez y Martín 2007).

1.2.3. Localización mediante la Transformada Hough

La Transformada Hough (**HT**) es una técnica útil para localizar objetos de cierta forma en la imagen digital (HPIR s.f.). La misma está basada en la transformación de la imagen desde el espacio ***XY*** hacia el *espacio de los parámetros*. Este espacio está determinado por los parámetros que, inequívocamente, caracterizan las curvas que se desean localizar mediante la **HT**. La Tabla 1.1 muestra los parámetros que identifican a curvas utilizadas comúnmente (Molina 1998).

Tabla 1.1**.** Parámetros que identifican a curvas utilizadas comúnmente.

Curva	Ecuación	Parámetros
Línea[2]	$y = mx + c$	*m, c*
Línea[3]	$\rho = x\cos\theta + y\sin\theta$	ρ, θ
Círculo	$(x-a)^2 + (y-b)^2 = r^2$	*a, b, r*

[2] Ecuación “explícita” de la recta.
[3] Ecuación “normal” de la recta (Duda y Hart 1972).

El proceso de localización de curvas mediante la **HT** comprende 3 etapas. Primeramente, se detectan los bordes existentes en la imagen digital y luego se registran las evidencias en el arreglo de votos. La cantidad de dimensiones del arreglo de votos depende del tamaño del espacio de los parámetros, y los límites de cada dimensión están condicionados por los valores válidos cuantizados del parámetro asociado. En la tercera etapa se localizan los máximos en el arreglo de votos. Las coordenadas de estos picos, se corresponden con los parámetros que identifican a las curvas más visibles en la imagen digital (Wikipedia s.f.).

Los métodos de localización mediante la **HT** están fundamentados en el principio de que, en los vehículos, las matrículas son delimitadas de su contexto mediante un rectángulo (Fig 1.8). Estos algoritmos detectan los bordes existentes en la imagen digital y a continuación emplean la **HT** para calcular las coordenadas de las líneas más visibles. Luego, estas líneas son combinadas y las regiones rectangulares que cumplan con los descriptores de las chapas (e.g. alto, ancho, relación de aspecto) son seleccionadas como posibles matrículas.

Figura 1.8. Imagen de un vehículo. a) Original. b) Bordes detectados.

Los algoritmos de localización basados en la **HT** garantizan que el sistema ANPR sea invariante a las transformaciones de rotación y de escala. Además, posibilitan el reconocimiento de matrículas de cualquier formato. Lamentablemente, la complejidad temporal del cálculo de la **HT** es elevada, lo que obstaculiza la implementación de estos métodos en sistemas ANPR diseñados para funcionar en circunstancias reales.

1.3. Tendencias actuales en las técnicas de extracción de los caracteres de la matrícula

En esta sección se describe cómo extraer los caracteres de la matrícula, mediante el análisis de las componentes conexas y de la proyección horizontal.

1.3.1. Extracción de los caracteres de la matrícula mediante el análisis de las componentes conexas

Los algoritmos de análisis de las componentes conexas permiten determinar las propiedades (e.g. centroide, perímetro, área) de los objetos presentes en la imagen digital. Estos métodos están compuestos por 3 etapas. Durante la fase de ***etiquetado***, cada pixel de la imagen recibe una etiqueta de acuerdo a las propiedades de su vecindad. La etapa de ***resolución de las equivalencias*** tiene el propósito de combinar las etiquetas equivalentes en una única etiqueta. Posteriormente, en la fase de ***re-etiquetado***, cada pixel de la imagen recibe una etiqueta de acuerdo al objeto al que pertenezca. Hasta la actualidad, se han desarrollado varios algoritmos para etiquetar las componentes conexas. Los más utilizados son "**Flood fill**", "**Union-find**" y "**Run length encoding**" (Eddins 2007).

El análisis de las componentes conexas, constituye una solución natural al problema de la extracción de los caracteres de la matrícula ya que, normalmente, cada carácter de la matrícula constituye un objeto aislado en la misma (Fig 1.9).

Figura 1.9. Resultado del análisis de las componentes conexas en la imagen de una matrícula.

La principal desventaja de los métodos de extracción de los caracteres de la matrícula, basados en el análisis de las componentes conexas, radica en la alta sensibilidad al ruido (un punto ruidoso puede causar la unión de 2 caracteres y por tanto el fracaso del proceso de extracción). Además, el proceso de resolución de las equivalencias, puede ser complicado y costoso en algunos algoritmos de análisis de las componentes conexas.

1.3.2. Extracción de los caracteres de la matrícula mediante el análisis de la proyección horizontal

Las proyecciones, o perfiles, constituyen un instrumento útil para analizar las imágenes digitales. Las proyecciones más utilizadas, en las aplicaciones de Visión artificial, son la proyección vertical (Exp 1.5) y la proyección horizontal (Exp 1.6). La proyección diagonal es empleada con menor frecuencia en estas aplicaciones (Fig 1.10).

$$\boldsymbol{P_V}(m) = \sum_{n=1}^{\boldsymbol{width}} \boldsymbol{I}(m,n) \qquad m = 1 \ldots \boldsymbol{height} \qquad \textbf{(1.5)}$$

$$\boldsymbol{P_H}(n) = \sum_{m=1}^{\boldsymbol{height}} \boldsymbol{I}(m,n) \qquad n = 1 \ldots \boldsymbol{width} \qquad \textbf{(1.6)}$$

La proyección horizontal ($\mathbf{P_H}$) es utilizada en varios algoritmos de extracción de los caracteres de la matrícula (Shapiro, y otros 2003). Estos algoritmos están fundamentados en el principio de que la separación vertical, existente entre los caracteres de la matrícula, genera elevaciones en $\mathbf{P_H}$. Dichas elevaciones, generalmente, comparten la misma magnitud y son equidistantes (Gráf 1.1).

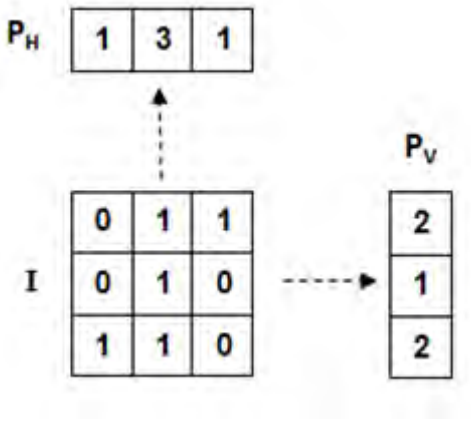


Figura 1.10. Proyecciones de una imagen digital.

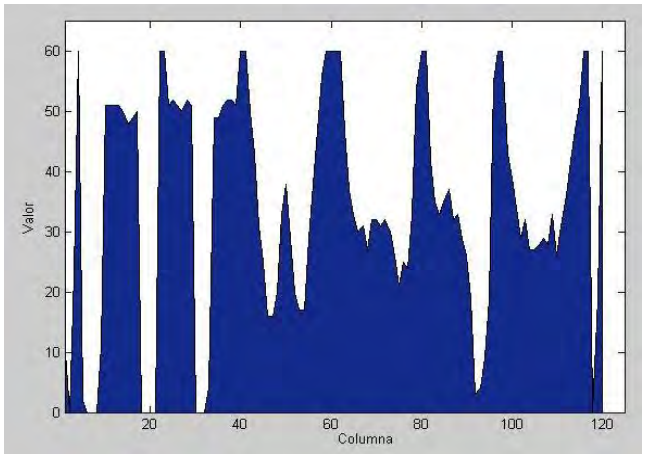


Gráfico 1.1. Proyección horizontal de la imagen de una matrícula.

Los algoritmos de extracción de caracteres, basados en el análisis de la proyección horizontal, utilizan métodos de detección de picos (MARTINSKY 2007) para calcular los índices de las elevaciones existentes en el perfil. Luego, estos índices son empleados para recortar la matrícula verticalmente.

El análisis de la proyección horizontal es una alternativa eficiente para extraer los caracteres de la matrícula. Su principal desventaja radica en la dependencia de la exactitud del algoritmo de detección de picos. Unido a lo anterior, téngase en cuenta que la detección de picos, en señales digitales, no constituye una tarea trivial.

1.4. Lenguajes de programación

En esta sección se especifican las características más significativas de los lenguajes de programación C++, Java y C#. Luego, se fundamenta la elección de C# 2.x, como lenguaje a utilizar para generar los artefactos *software* durante esta investigación.

1.4.1. Lenguaje de programación C++

C++ es un lenguaje de programación de propósito general inspirado en los lenguajes C y Simula67. Este lenguaje fue inventado por Bjarne Stroustrup en el año 1979 e inicialmente fue llamado "*C with classes*". El nombre actual, C++, fue adoptado en el año 1983 con el objetivo de indicar que el lenguaje era un "*C mejorado*" (Stroustrup 1997).

El lenguaje C++ soporta varios estilos de programación (e.g. procedural, orientado a objetos) e intenta ser tan eficiente y portable como lo es el lenguaje C. Además, permite la sobrecarga de los operadores, la inclusión de directivas del preprocesador y los tipos genéricos, entre otras funcionalidades. En C++, el encapsulamiento puede ser controlado mediante las palabras reservadas **private**, **protected public** y **friend**. Por otro lado, el lenguaje soporta plenamente el polimorfismo y la herencia, destacándose la existencia de la herencia múltiple (Stroustrup 1997).

El lenguaje C+ es utilizado para desarrollar, prácticamente, sistemas de cualquier tipo (e.g. sistemas operativos, sistemas incrustados, aplicaciones). Algunos de ellos son los sistemas operativos Unix, Linux y Windows, los juegos Quake y Starcraft, y el programa de edición gráfica Photoshop.

1.4.2. Lenguaje de programación Java

Java es un lenguaje de programación de propósito general creado en la compañía Sun Microsystems y tiene su antecedente en el lenguaje "Oak". Este lenguaje (Oak) se comenzó a desarrollar en 1990 por James Gosling y Bill Jay, entre otros, con el objetivo de desarrollar sistemas incrustados. Sin embargo, en 1993, ante la explosión de las aplicaciones de Internet y el escaso mercado para los sistemas incrustados, el grupo decidió enfocar el lenguaje "Oak" hacia el desarrollo de sistemas distribuidos y sustituir el nombre "Oak" por la cadena "Java" (Java Tech 2004).

El lenguaje Java es orientado a objetos y robusto. Entre sus características sobresalen, la ausencia de los punteros (la memoria es manejada automáticamente) y de las directivas del preprocesador, el soporte para la programación asincrónica y la utilización del enlace dinámico. Además, se destaca la naturaleza compilado-interpretado del código del lenguaje (el código Java es compilado a bytecodes[4] y luego es interpretado por una máquina virtual) (Java Tech 2004). Este lenguaje oferta, también, un mecanismo robusto de manejo de excepciones.

Java se ha desarrollado unido a Internet. Es por ello que el lenguaje es utilizado, ampliamente, en el desarrollo de sistemas distribuidos (e.g. sistemas de gestión, aplicaciones web). Además, durante los últimos años, el lenguaje se ha utilizado crecientemente en el desarrollo de sistemas incrustados.

1.4.3. Lenguaje de programación C#

C# es un lenguaje de programación orientado a objetos desarrollado por la compañía Microsoft como parte de la plataforma .NET. Su desarrollo fue guiado por Anders Hejlsberg y está basado en los lenguajes Delphi, Visual Basic, C++ y Java, entre otros (Wikipedia 2006).

El lenguaje C# ha sido diseñado para ser robusto, moderno y sencillo. El mismo promueve las prácticas sanas de programación (e.g. chequeo de tipos) y las reglas de globalización. Además, funciona bajo un entorno de recolección automática de la memoria, lo que aumenta la productividad del desarrollador (Butow y Ryan 2002). En este lenguaje no existe el nivel de visibilidad global ni la herencia múltiple. Por otro lado, los punteros solo pueden ser utilizados en contextos no manejados y la memoria manejada no

[4] Tipo de código intermedio más abstracto que el código máquina.

puede ser liberada explícitamente. Se destaca, además, la existencia de tipos genéricos, indexadores, delegados, eventos, clases parciales, estructuras y interfaces, entre otros rasgos (Microsoft 2006).

El lenguaje de programación C# es utilizado para desarrollar sistemas de gran diversidad. Sobresalen entre ellos las aplicaciones web, las aplicaciones de escritorio y los componentes reutilizables (toolkits y frameworks).

1.4.4. Fundamentación de la elección

En esta investigación se utiliza el lenguaje de programación C# 2.x, de la plataforma .NET, para generar los artefactos tipo *software*. La elección se debe a la elegancia del lenguaje y a las prestaciones que oferta .NET para desarrollar aplicaciones que involucren un gran volumen de cálculos.

La plataforma .NET incluye un recolector de basura para manejar automáticamente la memoria, pero no impide que el desarrollador asuma esta responsabilidad. El desarrollador puede manipular directamente la memoria manejada, utilizando punteros en un contexto inseguro. Además, se brinda la oportunidad de reservar recursos en la memoria no manejada, mediante miembros existentes en el tipo **System.Runtime.InteropServices.Marshal**. Esta facilidad ha sido expuesta en la plataforma con el objetivo de minimizar la cantidad de recolecciones de memoria (téngase en cuenta que durante una recolección de memoria, todos los hilos en ejecución son detenidos).

Las prestaciones que oferta .NET para exponer comportamiento asincrónico fueron tomadas en cuenta durante el proceso de toma de decisiones. La infraestructura para la programación asincrónica, existente en el espacio de nombres **System.Threading** del framework .NET, es robusta y fácil de utilizar. La misma incluye mecanismos de sincronización (e.g. monitores, semáforos) y una piscina de hilos, entre otros servicios. Todo ello es vital para explotar al máximo las potencialidades del sistema de cómputo.

Por otro lado, .NET constituye una preferencia actual para desarrollar componentes reutilizables. Esto se debe a las facilidades de interoperabilidad entre lenguajes que proporciona, a la consistencia del framework y a las excelentes herramientas de desarrollo que la acompañan, por solo mencionar algunos factores.

A todos los elementos mencionados, súmense las facilidades que brinda la plataforma .NET para procesar y analizar imágenes digitales. El espacio de nombres **System.Drawing** brinda acceso a las

funcionalidades básicas de GDI+. (Librería de tipos utilizada en el sistema operativo Windows para interactuar con los dispositivos gráficos.) Además, el espacio de nombres **System.Drawing.Drawing2D** contiene tipos relacionados con los gráficos vectoriales y las imágenes bidimensionales. Por su parte, el espacio de nombres **System.Drawing.Imaging** expone funcionalidades avanzadas de GDI+. No menos útil es el espacio de nombres **System.Drawing.Text**. El mismo, permite acceder a los servicios tipográficos de GDI+ (MSDN 2008).

La principal desventaja de utilizar el lenguaje de programación C# 2.x, de la plataforma .NET, radica en la naturaleza privativa de la plataforma. Para contrarrestar este problema, se adoptó una estrategia de migración hacia la plataforma libre Mono.

1.5. Lenguaje de modelación utilizado

En esta investigación se utiliza el Lenguaje Unificado de Modelado (UML) para representar las abstracciones de la realidad. La elección se basa en que UML 1.x es un lenguaje formal, conciso, comprensivo y escalable (Hamilton y Miles 2006) . Además, UML 1.x concentra las mejores prácticas de la comunidad y ha sido adoptado masivamente por la industria.

El lenguaje unificado de modelado está constituido por varios diagramas. Estos diagramas son utilizados para especificar, visualizar y documentar las características de un sistema (Chonoles y Schardt 2003). Algunos de ellos son el diagrama de actividades y el diagrama de estados.

Los autores de la investigación no desconocen la existencia de UML 2.0 ni sus mejoras respecto a UML 1.x. Sin embargo, se ha seleccionado UML 1.x ya que la herramienta CASE[5] a utilizar (Rational Rose 2003) no soporta UML 2.0.

1.6. Metodología de desarrollo utilizada

En la investigación, la generación de los artefactos estará guiada por los principios del Proceso Unificado de Desarrollo (RUP, por sus siglas en idioma Inglés). Dicha metodología está basada en componentes y utiliza el Lenguaje Unificado de Modelado para preparar todos los esquemas de un sistema software (Jacobson, Booch y Rumbaugh 1999).

[5] Computer Aided Software Engineering, por sus siglas en idioma Inglés.

RUP es iterativo e incremental, centrado en la arquitectura y dirigido por casos de uso (Jacobson, Booch y Rumbaugh 1999). Sus flujos de trabajo fundamentales son Modelación del negocio, Requerimientos, Análisis y diseño, Implementación y prueba, Despliegue, Gestión de la configuración, Gestión de proyectos, y Entorno. El esfuerzo dedicado a cada uno de estos flujos está distribuido en las fases Inicio, Elaboración, Construcción y Transición (IBM corporation 2003).

Todos los aspectos, mencionados anteriormente, permitirán obtener artefactos de calidad óptima, realizando un esfuerzo mínimo.

1.7. Herramientas utilizadas

En esta sección se describen las principales características de las herramientas Visual Studio 2005 Team Suite, Rational Rose 2003, AQTime Profiler 4.x, MoMA 1.x, Sancastle 1.3.2.0 y Matlab 2007[B].

1.7.1. Visual Studio 2005 Team Suite

La herramienta Microsoft Visual Studio 2005 Team Suite brinda servicios útiles para todos los miembros (e.g. arquitectos, diseñadores, desarrolladores) del equipo de desarrollo. Esta herramienta permite modelar la arquitectura y el diseño del sistema. Además incluye herramientas para, medir el rendimiento de la aplicación, garantizar la calidad del producto y optimizar la interacción de los miembros del equipo, entre otros servicios (Microsoft 2007).

1.7.2. Rational Rose 2003

Rational Rose 2003 es una herramienta CASE creada por la corporación Rational (IBM). Esta herramienta promueve el desarrollo basado en componentes, iterativo e incremental. Además, soporta los lenguajes de modelado UML (Unified Modeling Language), COM (Component Object Modeling), OMT (Object Modeling Technique) y Booch´93. Rational Rose incluye soporte para la generación automática de código a partir de los modelos. Algunos lenguajes de programación soportados son C++, Java y Visual Basic (Rational 2000).

1.7.3. AQTime Profiler

AQTime 4.x es una herramienta profesional, desarrollada por la compañía AutomatedQA, para realizar *profiling*[6] a las aplicaciones. AQTime puede analizar archivos ejecutables (e.g. exe, dll, ocx, bpl, cpl), servicios NT, aplicaciones ISAPI, aplicaciones ASP.NET, y servidores COM, DCOM y COM+. Esta herramienta puede ser integrada fácilmente con varios ambientes integrados de desarrollo. Algunos de ellos son Visual Studio 2008, Visual Studio 2005, Visual C++, C++ Builder e Intel C++ (AQTime 2007).

1.7.4. MoMA

La herramienta Mono Migration Analyzer (MoMA) permite analizar si una aplicación .NET puede ser ejecutada satisfactoriamente en la plataforma Mono. Lo anterior disminuye el esfuerzo a desarrollar durante la fase de migración. Nótese que el resultado retornado por MoMA es una estimación; el resultado real se obtiene al ejecutar la aplicación en la plataforma Mono (Mono s.f.).

1.7.5. Sancastle

Sandcastle (v3) es una herramienta que utiliza los mecanismos de **Reflection** (Archer 2001) y los comentarios Xml, para generar automáticamente la documentación de proyectos .NET (Codeplex 2008).

1.7.6. Matlab

Matlab 2007[b] es una ***excelente*** herramienta, con lenguaje de programación propio, orientada al desarrollo de aplicaciones científicas. Las prestaciones de Matlab cubren, prácticamente, todas ramas de la ciencia (e.g. estadística, bioinformática, análisis de imágenes digitales). Además se destacan las potencialidades de Matlab en la visualización de datos. Los paquetes **Filter Design**, **Fuzzy Logic**, **Image Acquisition, Image Processing**, **Neural Network** y **Statistics,** existentes en Matlab, facilitan el desarrollo de aplicaciones que utilicen técnicas de Reconocimiento de patrones y de Visión Artificial. Lo anterior, permite disminuir el esfuerzo y obtener un producto final de gran calidad (Mathworks 2007).

[6] Actividad que consiste en analizar cómo una aplicación realiza su tarea.

CONCLUSIONES

En este capítulo se han expuesto las características lógicas de los sistemas de reconocimiento automático del número de la matrícula y se demostró la utilización de la segmentación por colores, la detección de discontinuidades y la Transformada Hough, para localizar la matrícula del vehículo en la imagen digital. Además, se explicó cómo extraer los caracteres de la matrícula, mediante el análisis de las componentes conexas y de la proyección horizontal. Seguido, se justificó la utilización de los lenguajes C# 2.x y UML 1.x, y se argumentó la elección del Proceso Unificado como metodología de desarrollo. En el capítulo, además, se especificaron las características más significativas de las herramientas Visual Studio 2005 Team Suite, Rational Rose 2003, AQTime Profiler 4.x, MoMA 1.x, Sancastle 1.3.2.0 y Matlab 2007[b].

REFERENCIAS

[1] Acharya, Tinku, y Ajoy K Ray. *Image processing:Principles and applications.* New Jersey: Wiley-Interscience, 2005.

[2] Álvarez, Rafael L Cardero, y Jesmar E. Fajardo Martín. «Localización y segmentación de matrículas en imágenes de vehículos.» *RecPat´2007.* La Habana, 2007. 8.

[3] AQTime. «AQTime Profiler.» *Automated profiling and debugging.* 2007. http://www.automatedqa.com/products/aqtime/index.asp.

[4] Archer, Tom. *C# a fondo.* Redmond, Washington: Mc Graw Hill, 2001.

[5] Butow, Eric, y Tommy Ryan. *C#: Your visual blueprint for building .NET applications.* New York: Hungry Minds, Inc, 2002.

[6] Chonoles, Michael Jesse, y James A. Schardt. *UML 2 for dummies.* New York: Wiley, 2003.

[7] Codeplex. «Sandcastle.» *Sandcastle- Codeplex.* 15 de 1 de 2008. http://www.codeplex.com/Sandcastle.

[8] Comaniciu, Dorin, y Peter Meer. «Mean shift: A robust approach toward feature space analysis.» *IEEE Transactions on pattern analysis and machine intelligence*, 2002.

[9] Duda, R O, y P E Hart. «Use of the Hough Transformation to Detect Lines and Curves in Pictures.» *ACM*, 1972: 4.

[10] Eddins, Steve. «Connected component labeling - Wrapping up.» *Mathworks.* 2007. http://blogs.mathworks.com/steve/2007/06/13/connected-component-labeling-wrapping-up/ (último acceso: 2007 de 12 de 11).

[11] Gonzalez, Rafael C, y Richard E Woods. *Digital Image Processing.* New Jersey: Prentice Hall, 2002.

[12] Hamilton, Kim, y Russell Miles. *Learning UML 2.0.* Sebastopol: O'Reilly, 2006.

[13] Harvey, Deitel M, Paul J Deitel, Jeffrey A Listfield, Tem R Nieto, Cheryl H Yaeger, y Marina Zlaktina. *C# How to program.* Prentice Hall, 2001.

[14] HPIR. «Image transforms- Hough transform.» *HPIR.* http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm (último acceso: 10 de 12 de 2007).

[15] IBM corporation. *The Rational Unified Process.* 2003.

[16] Jacobson, Ivar, Grady Booch, y James Rumbaugh. *El Proceso Unificado de Desarrollo de Software.* Addison Wesley, 1999.

[17] Java Tech. «Java course.» 2004. http://www.particle.kth.se/~lindsey/JavaCourse/Book/courseMap.html (último acceso: 16 de 12 de 2006).

[18] MacQueen, J B. «Some Methods for classification and Analysis of Multivariate Observations.» *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[19] MARTINSKY, ONDREJ. «ALGORITHMIC AND MATHEMATICALPRINCIPLES OF AUTOMATIC NUMBER PLATE RECOGNITION SYSTEMS.» B.Sc. Thesis, Brno , 2007.

[20] Mathworks. *The Mathworks.* 2007. http://www.mathworks.com.

[21] Microsoft. *C# language specification.* Redmond, 2006.

[22] —. «Visual Studio 2005 Team Suite.» *MSDN.* 2007. http://msdn2.microsoft.com/en-us/teamsystem/aa718822.aspx.

[23] Molina, Rafael. *Introducción al procesamiento y análisis de imágenes digitales.* Granada, 1998.

[24] Mono. «MoMA.» *Mono.* http://www.mono-project.com/MoMA.

[25] MSDN. «System.Drawing namespace.» *Microsoft Developer Network- MSDN.* 2008. http://msdn2.microsoft.com/en-us/library/system.drawing.aspx.

[26] Rational. *Using Rose.* 2000.

[27] Ron, Bar-Hen, y Johanan Erez. «A Real-time vehicle License Plate Recognition System.» Conference Report, Tel Aviv, 2002.

[28] Shapiro, Vladimir, Dimo Dimov, Stefan Bonchev, Veselin Velichkov, y Giorgi Gluhchev. «Adaptive license plate image extraction.» Conference Report, 2003.

[29] Stroustrup, Bjarne. *The C++ programming language.* 3rd. New Jersey, 1997.

[30] Wikipedia. «C Sharp.» *Wikipedia.* 2006. http://en.wikipedia.org/wiki/C_Sharp (último acceso: 16 de 12 de 2007).

[31] —. «Hough transform.» *Wikipedia, the free encyclopedia.* http://en.wikipedia.org/wiki/Hough_transform (último acceso: 10 de 12 de 2007).

Capítulo 2

ALGORITMO DE LOCALIZACIÓN DE MATRÍCULAS PROPUESTO

En este capítulo se describen las características del algoritmo de localización de matrículas de vehículos, en imágenes digitales, propuesto en esta tesis. Los candidatos fueron generados a partir de los bordes verticales y se utilizó la Transformada Radon para calcular la inclinación de la matrícula. La conformidad de los candidatos con la matrícula se determinó mediante los descriptores relación de aspecto, longitud de la mayor línea horizontal y uniformidad de la proyección horizontal. La matrícula fue segmentada utilizando una técnica de binarización adaptativa.

INTRODUCCIÓN

Los métodos de localización de la matrícula (Fig 2.1), reciben como entrada la imagen digital de un vehículo y devuelven, como resultado, una imagen que se corresponde con la matrícula del automóvil.

Figura 2.1. Misión del módulo "**Localización de la matrícula**".

En este capítulo se describe el algoritmo, para localizar la matrícula del vehículo, propuesto por los autores de esta tesis. En este algoritmo, se utilizan los bordes verticales para generar los candidatos y la inclinación de la matrícula es calculada a partir del análisis de la Transformada Radon. La conformidad de los candidatos, con la matrícula, es determinada mediante los descriptores relación de aspecto, longitud de la mayor línea horizontal y uniformidad de la proyección horizontal. Además, se utiliza una binarización adaptativa para segmentar la matrícula.

La figura 2.2 muestra la imagen digital de un vehículo. Dicha imagen será utilizada, a lo largo de este capítulo, para demostrar la evolución del algoritmo de localización propuesto.

Figura 2.2. Imagen de un vehículo.

2.1. Generación de los candidatos

Se detectaron los bordes verticales mediante el operador de Sobel (Molina 1998). El umbral fue seleccionado de acuerdo al método propuesto en (Pratt 2001). Después de eliminar el ruido mediante el análisis de las componentes conexas (Acharya y Ray 2005), la imagen resultante fue dilatada con un elemento estructurante rectangular de amplitud **RECTWIDTH** puntos y de altura **RECTHEIGHT** puntos (Fig 2.3). Luego se etiquetaron las componentes conexas existentes en la imagen dilatada.

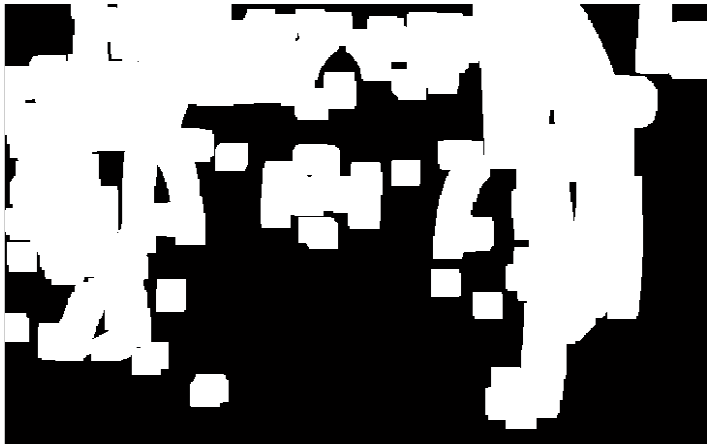


Figura 2.3. Candidatos generados a partir de la detección de los bordes verticales.

2.2. Verificación inicial de los rasgos

La región **R**, correspondiente a la componente conexa **C**, es admitida como candidata a matrícula si se cumple la expresión 2.1.

$$area(C) \geq \varphi_{Min} \quad \textbf{(2.1)}$$

donde:

φ_{Min}: Área mínima de las matrículas a localizar.

2.3. Detección de la inclinación

El candidato (Fig 2.4a) puede estar inclinado respecto a la horizontal debido a varios factores (e.g. inclinación natural, posición de la cámara), lo que puede impedir el éxito de la localización de la matrícula.

Para solucionar el problema se detectaron los bordes horizontales presentes en el candidato y se sometió la imagen obtenida a un proceso de dilatación (Gonzalez y Woods 2002), con un elemento estructurante en forma de diamante (The Mathworks 2007) de radio 1 (Fig 2.4b). Luego se calculó la Transformada Radon (Toft 1996)—de la imagen dilatada— a lo largo de los ángulos [90^0 - ϕ_{Max} ; 90^0 +

ϕ_{Max}]; donde ϕ_{Max} representa el ángulo de la máxima inclinación horizontal permitida en las matrículas a detectar.

A partir de la trasformada Radon (RT) se calculó el ángulo de la inclinación, los límites horizontales de la matrícula y la longitud de la mayor línea que delimita la matrícula.

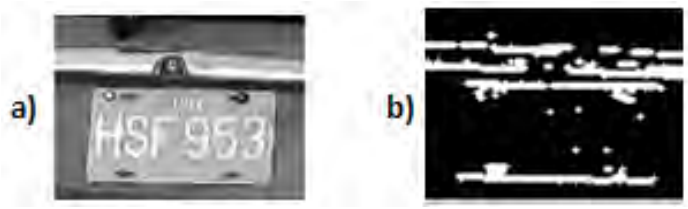


Figura 2.4. Región seleccionada como posible matrícula.
a) Original. b) Bordes horizontales dilatados.

2.3.1. Cálculo del ángulo de la inclinación

Sea **R** (rho, theta) la coordenada del valor máximo en la RT, el ángulo de la inclinación (en contra de las manecillas del reloj) fue calculado mediante la expresión 2.2.

$$\theta = \text{theta} - 90^0 \quad \textbf{(2.2)}$$

2.3.2. Cálculo de los límites horizontales de la matrícula

Sean **P** la proyección de la RT en el ángulo ***theta***, ***a*** el índice del mayor pico en **P**, ***b*** el índice del pico más alejado de ***a*** en **P** tal que $\frac{\mathbf{P(b)}}{\mathbf{P(a)}} \geq \frac{\mathbf{1}}{\mathbf{2}}$ y ***h*** la altura de la imagen dilatada (**D**), el límite superior y el límite inferior de la matrícula fueron calculados mediante las expresiones 2.3 y 2.4, respectivamente (Gráf 2.1).

$$\boldsymbol{up = C_{DIL} - (a - C_P)} \quad \textbf{(2.3)}$$

$$\boldsymbol{down = C_{DIL} - (b - C_P)} \quad \textbf{(2.4)}$$

donde:

$$C_{DIL} = floor\left(\frac{h+1}{2}\right)$$

$$C_P = \frac{length\,(P) - 1}{2}$$

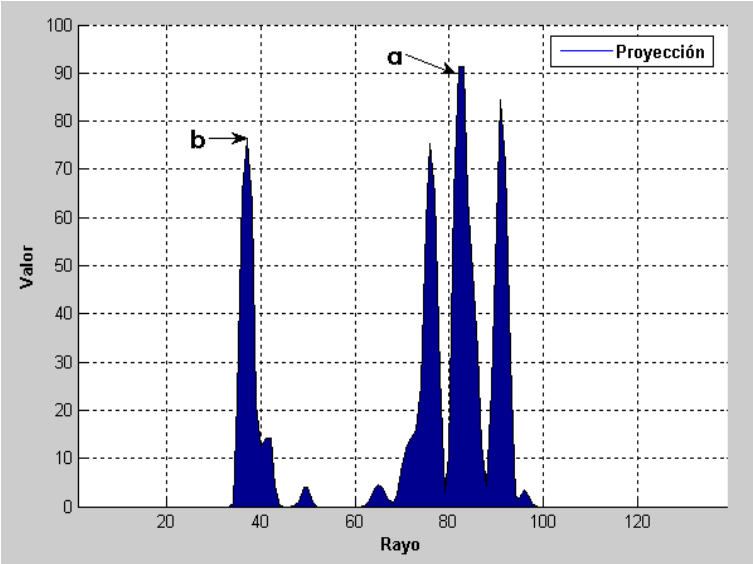


Gráfico 2.1. Proyección de la Transformada Radon en el ángulo ***theta***.

2.3.3. Cálculo de la longitud de la mayor línea que delimita la matrícula

Sean $\mathbf{P_1}(x_1, y_1)$ y $\mathbf{P_2}(x_2, y_2)$ los puntos de intersección del rayo ***rho*** con 2 lados de **D**, **R** el menor rectángulo que contiene a los puntos $\mathbf{P_1}$ y $\mathbf{P_2}$, y **S** la región de **D** encerrada por el rectángulo **R**, la longitud de la mayor línea que delimita la matrícula fue calculada mediante la expresión 2.5.

$$L = \max\,(l_i) \qquad i = 1 \ldots n \qquad \textbf{(2.5)}$$

donde:

n: Cantidad de componentes conexas **C** en la región **S**.

$h = height\,(C)$

$w = width\,(C)$

$l = \sqrt{h^2 + w^2}$

2.3.4. Verificación de la longitud de la línea

El candidato es admitido para futuro análisis si contiene una línea horizontal de longitud **L** mayor o igual que el valor del parámetro $\mathbf{L_{MIN}}$ (longitud mínima de la línea horizontal que delimita las matrículas a

localizar). Si el candidato no cumple con la condición, es adicionado a una estructura tipo **FIFO** (First-in First-out) condicionada por prioridad. (La longitud **L** indica la prioridad del candidato.)

2.4. Supresión de la inclinación

Conocido el ángulo de la inclinación θ, el candidato fue rotado para alinear la matrícula respecto a la horizontal (Fig 2.5).

Figura 2.5. Región seleccionada como posible matrícula alineada respecto a la horizontal. (corregida la inclinación de $\mathbf{-2^0}$)

2.5. Recorte horizontal

Se recortó la región del candidato comprendida entre los límites ***up*** y ***down*** calculados a partir de la RT (Fig 2.6a). Si la región recortada no se corresponde con la matrícula, los límites son calculados mediante el análisis de la proyección vertical y la región es recortada nuevamente.

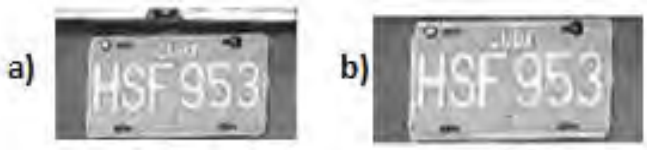


Figura 2.6. Región seleccionada como posible matrícula, recortada horizontalmente.
a) Analizando la Transformada Radon b) Analizando la proyección vertical.

2.5.1. Análisis de la proyección vertical

Se detectaron los bordes verticales presentes en el candidato y luego se dilató la región resultante con una línea de longitud **RECTWIDTH** puntos. Luego se calculó la proyección vertical $\mathbf{P_V}$ de la imagen dilatada (Gráf 2.2).

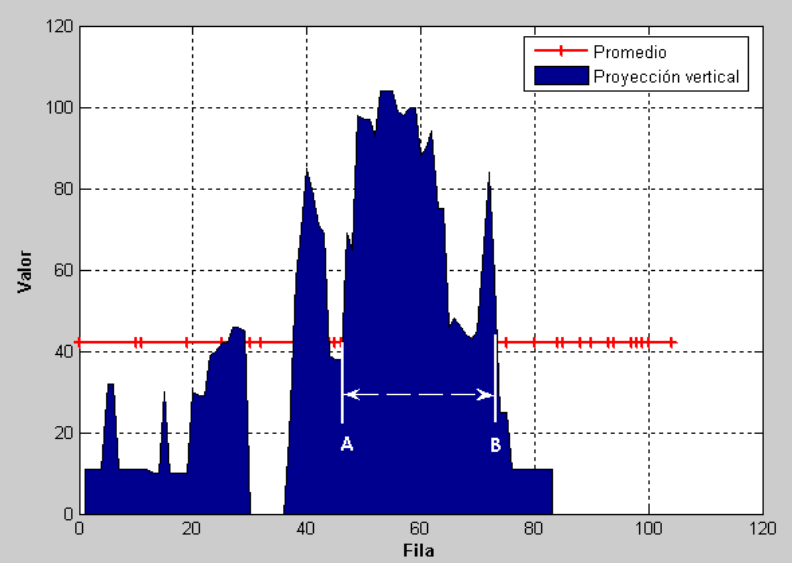


Gráfico 2.2. Proyección vertical de la imagen dilatada.

Sean **A** y **B** las cotas de la elevación más ancha en $\mathbf{P_V}$ —compuesta por los elementos con valor superior al promedio de $\mathbf{P_V}$ —, los límites **up** y **down** fueron calculados mediante las expresiones 2.6 y 2.7 respectivamente (Fig 2.6b).

$$up = A \quad \textbf{(2.6)}$$

$$down = B \quad \textbf{(2.7)}$$

2.6. Recorte vertical

Se detectaron los bordes verticales existentes en la región recortada (Fig 2.7). Luego, se calculó y se suavizó la proyección horizontal (P_H) de la imagen resultante.

Figura 2.7. Bordes verticales existentes en la región recortada.

Se utilizó el método *peaks-to-valleys* (MARTINSKY 2007) para localizar las elevaciones existentes en la proyección horizontal (Gráf 2.3). (El umbral que requiere el método fue calculado mediante la expresión 2.8). Seguido, se eliminaron las elevaciones ruidosas.

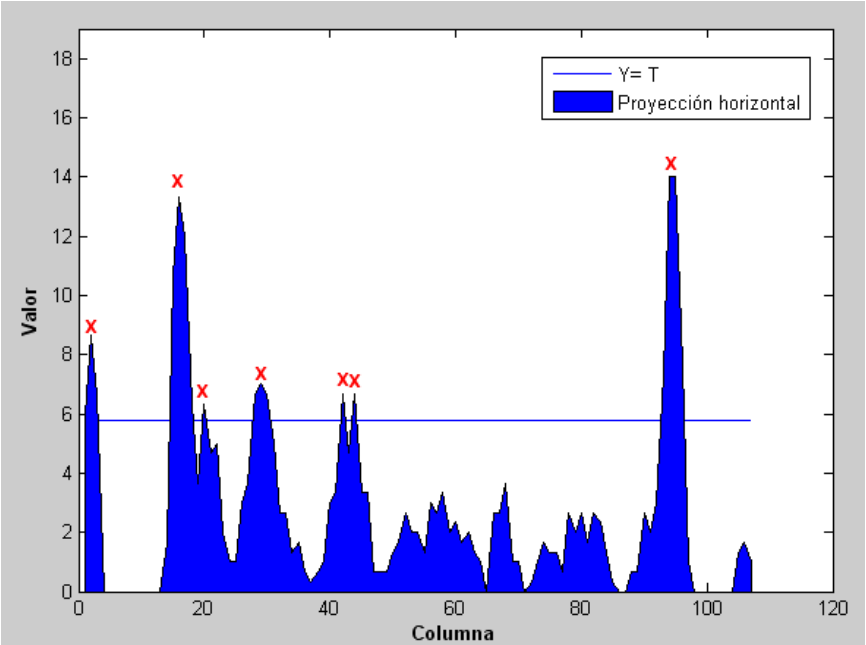


Gráfico 2.3. Elevaciones existentes en la proyección horizontal.

Supóngase **E** como el conjunto de elevaciones (no ruidosas) existentes en la proyección horizontal. Sean **a**∈**E** y **b**∈**E** dos elevaciones consecutivas para las cuales se maximiza la expresión 2.9, el candidato fue recortado verticalmente por los límites ***left*** y ***right*** de acuerdo a la expresión 2.10.

$$T = \bar{x} + s \qquad (2.8)$$

donde:
$\bar{x}$: Media aritmética de P_H.
s: Desviación estándar de P_H.

$$\boldsymbol{cost}(a,b) = \sum_{i=j}^{k} P_H(i) \qquad \textbf{(2.9)}$$

donde:
j: Índice del primer elemento en la elevación **a**.
k: Índice del último elemento en la elevación **b**.

$$left = \boldsymbol{mainIndex}(a) \qquad \textbf{(2.10)}$$

$$right = \boldsymbol{mainIndex}(b)$$

donde:
mainIndex (l): Índice principal de la elevación **l**.

2.7. Eliminación de ruido

El candidato (Fig 2.8) fue sometido al algoritmo expuesto en la **Sección** 2.5.1. Luego se recortó, horizontalmente, la región del candidato comprendida entre las cotas ***up*** y ***down*** (Fig 2.9).

Figura 2.8. Candidato recortado vertical y horizontalmente.

Figura 2.9. Resultado de la eliminación de ruido

2.8. Verificación de rasgos

El análisis del candidato continua si y solo si se cumple la expresión 2.11.

$$\gamma_{Min} \leq \frac{\boldsymbol{width}(C)}{\boldsymbol{height}(C)} \leq \gamma_{Max} \qquad \textbf{(2.11)}$$

donde:

γ_{Min}**:** Mínima relación de aspecto de las matrículas a localizar.

γ_{Max}**:** Máxima relación de aspecto de las matrículas a localizar.

2.9. Binarización del candidato

Se ajustó el contraste del candidato y se binarizó el resultado. Luego se normalizó el color del fondo del candidato (Fig 2.10).

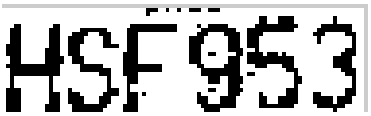


Figura 2.10. Resultado de la binarización del candidato.

2.9.1. Selección del umbral para la binarización

El umbral adaptativo fue calculado mediante una variación realizada al método propuesto en (Ron 2002). (El umbral inicial fue calculado a partir de la moda de los niveles de grises del candidato.)

2.9.2. Normalización del color del fondo de la matrícula

El resultado de la binarización del candidato se normalizó para contribuir al éxito de etapas posteriores (e.g. extracción de los caracteres). El color de los puntos del fondo del candidato se estableció a blanco y el color de los demás puntos del candidato se estableció a negro.

La detección del color, del fondo del candidato, se realizó calculando la moda del candidato binario. Lo anterior está fundamentado en el principio de que, generalmente, las matrículas poseen más puntos de fondo que puntos de caracter.

2.10. Verificación de la firma del candidato

La proyección horizontal de una matrícula se caracteriza por la existencia de varias elevaciones espaciadas uniformemente (Shapiro, y otros 2003). La figura 2.11 muestra la proyección horizontal de una matrícula y la figura 2.12 muestra la proyección horizontal del rostro de una persona.

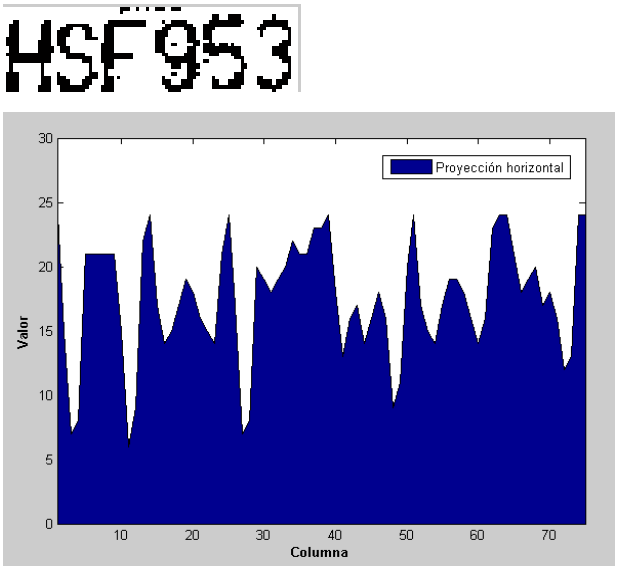


Figura 2.11. Imagen de una matrícula y su correspondiente proyección horizontal.

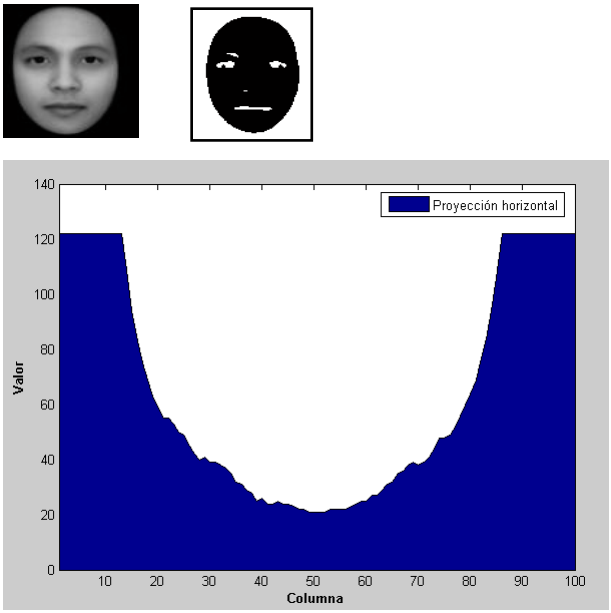


Figura 2.12. Rostro de una persona y su correspondiente proyección horizontal.

Sea $\mathbf{P_H}$ la proyección horizontal del candidato binario, el candidato es aceptado como matrícula si las (**N-1**) mayores elevaciones están espaciadas uniformemente. Las elevaciones fueron localizadas utilizando el método *peaks-to-valleys* (MARTINSKY 2007).

2.11. Normalización del tamaño de la matrícula

Se normalizó el tamaño de la matrícula detectada. Las figuras 2.13a y 2.13b muestran el resultado final del algoritmo de localización de matrículas de vehículos en imágenes digitales, descrito en este capítulo.

Figura 2.13. Resultado final del algoritmo propuesto.
a) Matricula en escala de grises. b) Matrícula binaria.

CONCLUSIONES

En este capítulo se ha presentado un algoritmo para localizar la matrícula de los vehículos, en imágenes digitales. Los candidatos fueron generados a partir de los bordes verticales y la inclinación fue calculada analizando la Transformada Radon (RT).

La conformidad de cada candidato con la matrícula fue verificada mediante los descriptores relación de aspecto, longitud de la mayor línea horizontal y uniformidad de la proyección horizontal.

Las áreas ruidosas presentes en cada candidato fueron eliminadas analizando los rayos de la RT, la proyección vertical y la proyección horizontal. Posteriormente, el objeto fue segmentado mediante una técnica de binarización adaptativa.

REFERENCIAS

[1] Acharya, Tinku, y Ajoy K Ray. Image processing: Principles and applications. New Jersey: Wiley-Interscience, 2005.

[2] Gonzalez, Rafael C, y Richard E Woods. Digital Image Processing. New Jersey: Prentice Hall, 2002.

[3] MARTINSKY, ONDREJ. «ALGORITHMIC AND MATHEMATICALPRINCIPLES OF AUTOMATIC NUMBER PLATE RECOGNITION SYSTEMS.» B.Sc. Thesis, Brno , 2007.

[4] Molina, Rafael. Introducción al procesamiento y análisis de imágenes digitales. Granada, 1998.

[5] Pratt, William K. Digital Imaging Processing. Wiley-Interscience, 2001.

[6] Ron, Bar-Hen. «A Real-time vehicle License Plate Recognition (LPR) System.» Conference Report, 2002.

[7] Shapiro, Vladimir, Dimo Dimov, Stefan Bonchev, Veselin Velichkov, y Giorgi Gluhchev. «Adaptive license plate image extraction.» Conference Report, 2003.

[8] The Mathworks. Image Processing Toolbox User´s Guide- strel function. 2007.

[9] Toft, Peter. «The Radon Transform - Theory and Implementation.» PhD Thesis, 1996.

Capítulo 3

ALGORITMO DE EXTRACCIÓN DE CARACTERES PROPUESTO

En este capítulo se describe el algoritmo de extracción de los caracteres de la matrícula propuesto en esta tesis. Primero, se intenta extraer los caracteres, mediante el análisis de las componentes conectadas. Si el resultado obtenido es incorrecto, se intenta extraer los caracteres mediante el análisis de las elevaciones existentes en la proyección horizontal. Estas elevaciones son detectadas utilizando el método *peaks-to-valleys*.

INTRODUCCIÓN

Los métodos de extracción de los caracteres de la matrícula (Fig 3.1), reciben como entrada la imagen digital de una matrícula y devuelven como resultado un conjunto de imágenes digitales. Cada imagen digital, del conjunto, se corresponde con un carácter significativo presente en la matrícula (Hansen, y otros 2002).

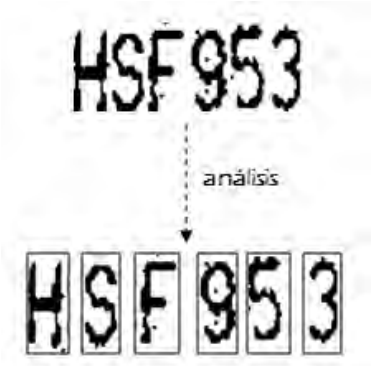


Figura 3.1. Misión del módulo "**Extracción de los caracteres de la matrícula**".

En este capítulo se describe el algoritmo, para extraer los caracteres de la matrícula, propuesto por los autores de esta tesis. El mismo está basado en una estrategia híbrida. Primeramente, se intenta recortar los caracteres, mediante el análisis de las componentes conexas (Acharya y Ray 2005). Luego, si el resultado es correcto, el proceso termina; en caso contrario se intenta recortar los caracteres de la matrícula, mediante el análisis de la proyección horizontal.

La figura 3.2 muestra la imagen digital de una matrícula (segmentada y con las áreas ruidosas eliminadas). Dicha imagen será utilizada, a lo largo de este capítulo, para demostrar la evolución del algoritmo de extracción propuesto.

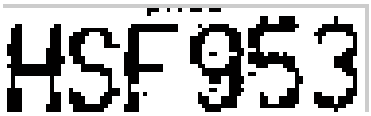


Figura 3.2. Imagen digital de una matrícula.

3.1. Extracción mediante el análisis de las componentes conectadas

Se aplicó un filtro tipo "negativo" (Molina 1998), a la imagen digital de la matrícula, con el objetivo de establecer el color de los puntos de caracter a blanco (Fig 3.3). Luego, se etiquetaron las componentes conexas (C_i) existentes en la imagen resultante y se seleccionaron aquellas en las que se cumple la expresión 3.1. Llámese β al conjunto compuesto por estas componentes conexas.

$$\boldsymbol{height}(C_i) \geq \frac{4}{5}\boldsymbol{height}(I) \qquad \textbf{(3.1)}$$

donde:

C_i: i-ésima componente conexa.

I: Imagen digital de la matrícula.

Se establecieron como posibles caracteres a las componentes conexas existentes en el conjunto β. Este resultado se admite como "*correcto*", si se cumple la expresión 3.2 (Fig 3.4).

$$length(\beta) \geq N \qquad \textbf{(3.2)}$$

donde:

N: Cantidad de caracteres significativos existentes en las matrículas bajo estudio.

Figura 3.3. Resultado de la aplicación del filtro tipo “negativo”.

Figura 3.4. Regiones recortadas mediante el análisis de las componentes conexas

3.2. Extracción mediante el análisis de la proyección horizontal

Se calculó el negativo (Molina 1998) de la imagen de la matrícula. Luego, se dilató (Gonzalez y Woods 2002) el resultado con un elemento estructurante lineal (L= 5 puntos, θ= 90^0) y se aplicó una negación (Fig 3.5).

Figura 3.5.Preprocesamiento de la imagen de la matrícula.

Se calculó la proyección horizontal de la imagen de la matrícula y se utilizó el método *peaks-to-valleys* (MARTINSKY 2007) para detectar las elevaciones existentes en dicha proyección (Gráf 3.1).

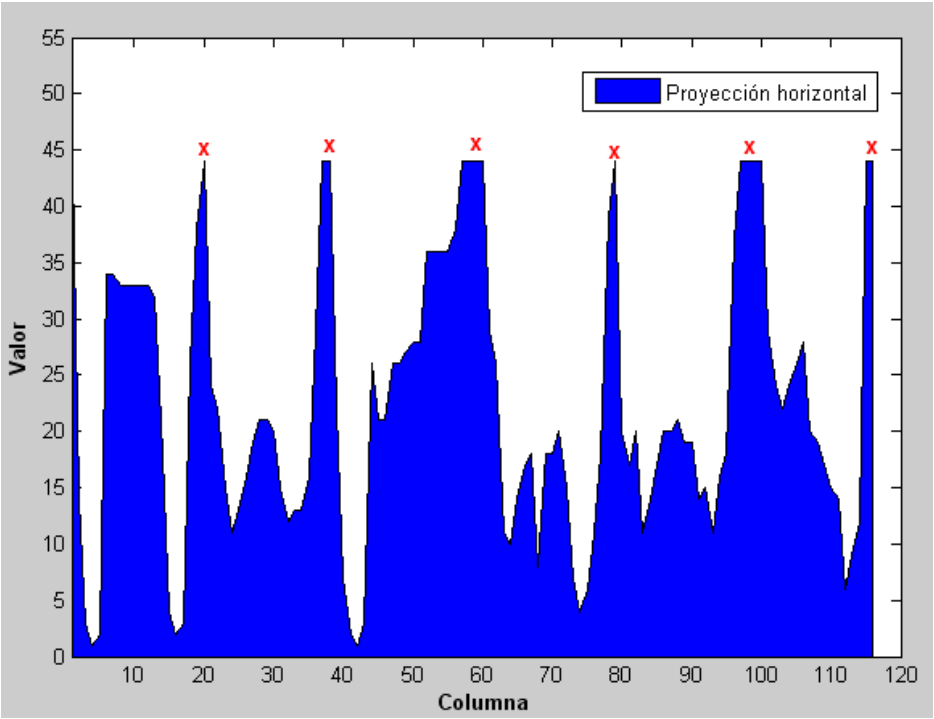


Gráfico 3.1. Elevaciones existentes en la proyección horizontal de la matrícula.

Seguido, se aplicó una negación (Gonzalez y Woods 2002) a la imagen de la matrícula y se recortó la misma, verticalmente, por el índice principal de cada elevación detectada (Fig 3.6).

Figura 3.6. Regiones recortadas mediante el análisis de la proyección horizontal.

CONCLUSIONES

En este capítulo se ha presentado un algoritmo para extraer los caracteres significativos, de la imagen digital de una matrícula. El método propuesto estuvo basado en una estrategia híbrida. Primero, se intentó recortar los caracteres de la matrícula mediante el análisis de las componentes conexas. Si el resultado obtenido fue incorrecto, se intentó extraer los caracteres, mediante el análisis de las elevaciones existentes en la proyección horizontal. Estas elevaciones fueron detectadas utilizando el método *peaks-to-valleys*.

REFERENCIAS

[1] Acharya, Tinku, y Ajoy K Ray. Image processing: Principles and applications. New Jersey: Wiley-Interscience, 2005.

[2] Gonzalez, Rafael C, y Richard E Woods. Digital Image Processing. New Jersey: Prentice Hall, 2002.

[3] Hansen, Henrik, Anders Wang Kristensen, Morten Porsborg Kohler, Allan Weber Mikkelsen, Jens Mejdahl Pedersen, y Michael Trangeled. «Automatic recognition of license plates.» Thesis, 2002.

[4] MARTINSKY, ONDREJ. «ALGORITHMIC AND MATHEMATICALPRINCIPLES OF AUTOMATIC NUMBER PLATE RECOGNITION SYSTEMS.» B.Sc. Thesis, Brno , 2007.

[5] Molina, Rafael. Introducción al procesamiento y análisis de imágenes digitales. Granada, 1998.

[6] The Mathworks. Image Processing Toolbox User´s Guide- strel function. 2007.

Capítulo 4

CARACTERÍSTICAS DEL SISTEMA

En este capítulo se describe el negocio, se detalla la situación problemática y se presenta el problema científico. Además se muestran los requisitos funcionales, los requisitos no funcionales y la definición de los casos de uso, del sistema propuesto.

INTRODUCCIÓN

El objetivo del flujo de trabajo **Modelación del negocio**, del Proceso Unificado de Desarrollo, es comprender la estructura y la dinámica de la organización en la que se implantará el sistema (IBM corporation 2003). Los trabajadores que intervienen en el mismo son el Analista de procesos del negocio, el Diseñador del negocio y el Revisor del Modelo del negocio (IBM 2003). El la modelación del negocio se desarrolla el Modelo de casos de uso de negocio. (En lo adelante Modelo de negocio.) Un Modelo de negocio describe los procesos de negocio de una empresa en términos de casos de uso del negocio y actores del negocio, que se corresponden con los procesos del negocio y los clientes, respectivamente (Jacobson, Booch y Rumbaugh 2000). Además, en este flujo de trabajo se especifican las Reglas del negocio, la Lista de riesgos, el Documento Visión y el Glosario de Términos (IBM 2003).

En el flujo de trabajo **Requerimientos**, del Proceso Unificado de Desarrollo, se identifican, enuncian y clasifican los requerimientos (IBM 2003). Dichos requisitos se clasifican en funcionales y no funcionales (Jacobson, Booch y Rumbaugh 2000). Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los requisitos no funcionales son propiedades o cualidades que el producto debe tener (Pressman 2002). Los trabajadores que intervienen en esta etapa (Requerimientos) son el Analista del sistema, el Especificador de casos de uso, el Diseñador de interfaz de usuario y el Arquitecto (IBM 2003).

En este capítulo se describe el negocio relacionado con el tema que se aborda en la tesis, y se exponen las características del sistema propuesto. Lo que resta del mismo está organizado como sigue. En la **Sección 4.1** se describe el negocio. En la **Sección 4.2** se detalla la situación problemática y se presenta el problema científico. Luego, en la **Sección 4.3**, se especifica la propuesta de sistema.

4.1. Descripción del negocio

Los vehículos se identifican en nuestro país (Cuba), generalmente, de modo manual. Cuando un vehículo arriba a un punto de control y solicita la entrada a la instalación, el custodio (Responsable de seguridad) verifica que al vehículo se le ha concedido permiso de acceso. Para ello, el custodio lee la matrícula del vehículo y localiza la misma en el Registro de vehículos autorizados. Si la cadena de la matrícula es encontrada, el custodio acepta la entrada del vehículo y actualiza el Registro de entradas. (El Registro de entradas contiene la hora de entrada de los vehículos.) En caso de que la cadena de la matrícula no sea encontrada, el custodio rechaza el acceso y el vehículo debe abandonar el punto de control.

En el punto de control también se controlan las salidas de los vehículos. Cuando un vehículo arriba al punto de control, solicitando la salida, el custodio lee la matrícula del vehículo y actualiza el Registro de salidas (El Registro de salidas contiene la hora de salida de los vehículos.). Luego, el vehículo abandona la instalación.

La descripción anterior demuestra la existencia del actor de negocio **Vehículo** y de los procesos de negocio **Registrar entrada** y **Registrar salida** (Diagrama 4.1). Además, se verifica la presencia del trabajador **Custodio** y de las entidades **Registro de vehículos autorizados**, **Registro de entradas, Registro de salidas** y **Matrícula**. (Diagrama 4.2).

Diagrama 4.1. Diagrama de casos de uso del negocio.

En el “**Anexo 1. Modelo de negocio**” se justifican los actores y los trabajadores del negocio. Además, se muestra el diagrama de casos del uso del negocio, el diagrama de clases del Modelo de objetos y el diagrama de actividades asociado a cada caso de uso del negocio.

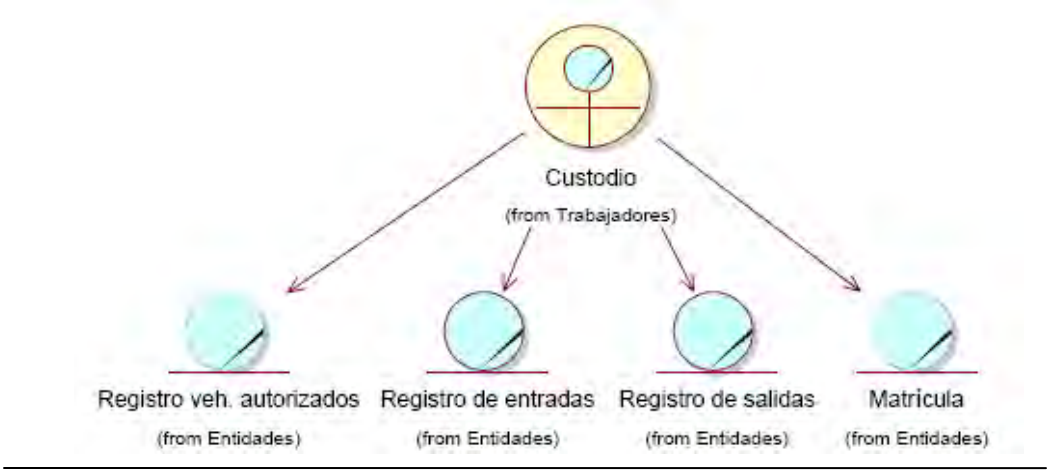


Diagrama 4.2. Diagrama de clases del Modelo de objetos.

4.2. Situación problemática y problema científico

La naturaleza manual, del proceso de identificación de los vehículos, está causando varios problemas en Cuba. En las empresas se destinan varios trabajadores (Custodios) para desarrollar esta tediosa actividad. Lo anterior afecta la economía de la institución y disminuye la calidad de vida de los custodios. Además, a pesar del esfuerzo que se realiza, es frecuente la pérdida de los registros de datos y la corrupción de los mismos. A todo ello, súmese la ausencia de un proceso de gestión de la información. Como consecuencia, los datos generados durante las entradas y las salidas de los vehículos, se convierten en un repositorio ocioso dentro de la empresa.

En los países desarrollados se utilizan los sistemas basados en ANPR para identificar automáticamente los vehículos (CRS s.f.). Esta solución, completamente no supervisada, es robusta en disímiles condiciones y posibilita la implantación de un sistema informático de gestión de la información. Además, facilita el envío de avisos y la emisión de mensajes personalizados (Fig 4.1).

La aplicación de los sistemas basados en ANPR no está restringida al control de acceso. También son utilizados en la localización de autos robados, la detección de violaciones de las leyes del tránsito, el control de peajes y el monitoreo de autopistas, principalmente.

Figura 4.1. Control de acceso utilizando un sistema basado en ANPR.

La utilización, en Cuba, de sistemas basados en ANPR para identificar automáticamente los vehículos es la solución ideal, no la real. El costo de adquisición y despliegue de estos novedosos sistemas es prohibitivo para un país con dificultades económicas como el nuestro. Además, en el país no existe experiencia en el desarrollo de sistemas de identificación basados en ANPR.

Esta tesis se rige por el problema científico ¿Cómo localizar y segmentar matrículas de vehículos en imágenes digitales? En la misma, se diseñan e implementan los componentes "**Localización de la matrícula**" y "**Extracción de los caracteres de la matrícula**", de un sistema ANPR cubano. Lo anterior, ayuda a solucionar el problema de la identificación automática de los vehículos en Cuba.

4.3. Propuesta de sistema

El sistema que se propone es una librería de tipos (elemento reutilizable) que permita localizar y segmentar matrículas de vehículos en imágenes digitales (Diagrama 4.3). El sistema debe ser compatible con la plataforma libre Mono y los tipos expuestos en la librería deben estar documentados de acuerdo a los estándares vigentes (Tablas 4.1 y 4.2).

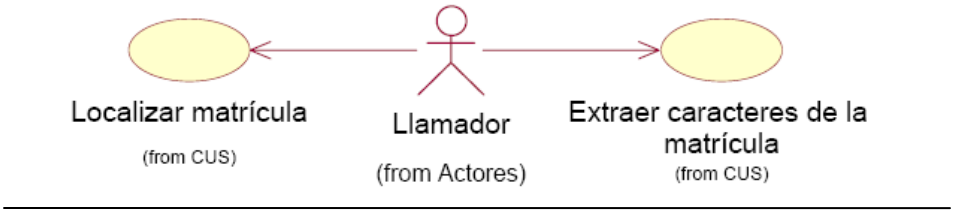


Diagrama 4.3. Diagrama de casos de uso del sistema.

En el "**Anexo 2. Artefactos de requerimientos**" se describen los actores y los casos de uso. Además, se muestra el diagrama de casos de uso del sistema propuesto.

Tabla 4.1. Listado de requisitos funcionales.

Número	Requisito
RF-1	El sistema debe ser capaz de localizar matrículas en la imagen digital de un vehículo.
RF-2	El sistema debe ser capaz de extraer los caracteres significativos existentes en la imagen digital de una matrícula (segmentada y no ruidosa).

Tabla 4.2. Listado de requisitos no funcionales.

Número	Requisito	Tipo
RNF-1	El sistema deber ser portable a la plataforma libre Mono.	Portabilidad
RNF-2	El sistema es propiedad intelectual del Grupo de Procesamiento de Imágenes y Señales Digitales (GPI), perteneciente a la Universidad de las Ciencias Informáticas.	Legal
RNF-3	Los tipos expuestos en la librería deben estar documentados de acuerdo al estándar vigente.	Documentación

CONCLUSIONES

En este capítulo se describió el negocio, se detalló la situación problemática y se presentó el problema científico. Además, se reflejaron los requisitos funcionales, los requisitos no funcionales y la definición de los casos de uso, del sistema propuesto.

REFERENCIAS

[1] Boogs, Wendy, y Michael Boogs. *Mastering UML with Rational Rose 2002.* California: SYBEX, 2002.

[2] CRS. *Computer Recognition Systems.* http://www.crs-traffic.co.uk/index.html (último acceso: 2 de 12 de 2007).

[3] IBM. *Rational Unified Process Online Help.* 2003.

[4] Jacobson, Ivar, Grady Booch, y James Rumbaugh. *El Proceso Unificado de Desarrollo.* Addison Wesley Logman, 2000.

[5] Pressman, Roger S. *Ingeniería de software: Un enfoque práctico.* Addison-Wesley, 2002.

Capítulo 5

ANÁLISIS Y DISEÑO DEL SISTEMA

En este capítulo se presentan las clases de análisis y las clases de diseño, que participan en la realización de los casos de uso del sistema propuesto. Además, se reflejan los diagramas de colaboración entre las clases de diseño y se describe, detalladamente, cada una de estas clases.

INTRODUCCIÓN

El objetivo del flujo de trabajo **Análisis y Diseño**, del Proceso Unificado de Desarrollo, es traducir los requisitos en una especificación que describe cómo se debe implementar el sistema (UCI 2008). Los roles que intervienen en el mismo son el Arquitecto, el Diseñador y el Revisor Técnico (IBM 2003).

En el **Análisis** se tienen en cuenta los requisitos funcionales, para generar las clases de análisis. Estas clases se clasifican en Entidad, Control e Interfaz (Jacobson, Booch y Rumbaugh 2000). Por su parte, el **Diseño** es un refinamiento del análisis, tiene en cuenta los requisitos no funcionales y su salida debe ser suficiente para que el sistema pueda ser implementado, en un entorno de programación dado, sin ambigüedades (UCI 2008).

En este capítulo se aborda el análisis y diseño del sistema propuesto. Lo que resta del mismo está organizado como sigue. En la **Sección 5.1** se presentan las clases de análisis, las clases de diseño y los diagramas de colaboración, correspondientes al caso de uso *Localizar matrícula*. Luego, en la **Sección 5.2** se presentan los artefactos relacionados con el caso de uso *Extraer caracteres de la matrícula*.

5.1. Análisis y diseño. Caso de uso "*Localizar matrícula*"

En esta sección se presentan las clases de análisis, las clases de diseño y los diagramas de colaboración, correspondientes a la realización del caso de uso *Localizar matrícula* (Diagrama 5.1).

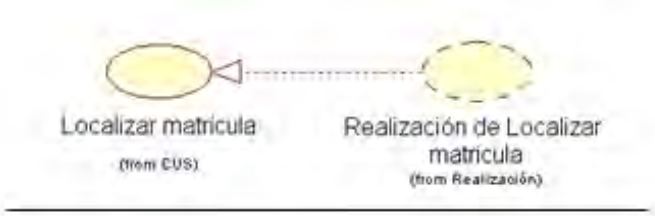


Diagrama 5.1**.** Realización del caso de uso **Localizar matrícula**.

5.1.1. Diagrama de clases de análisis

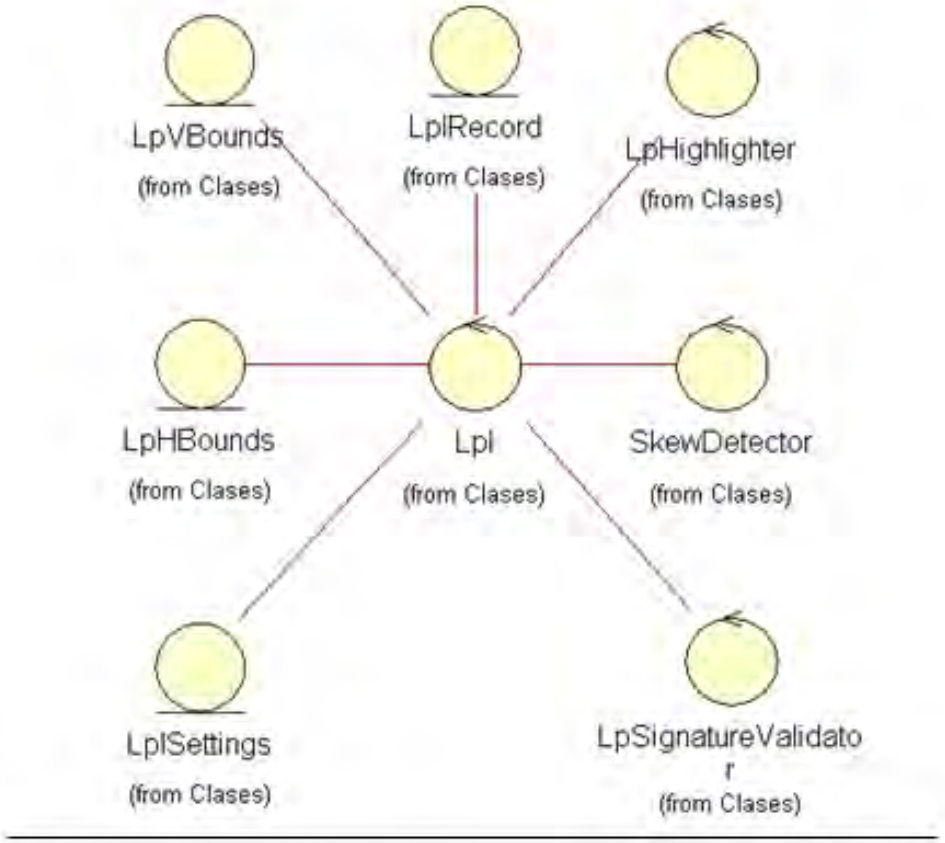


Diagrama 5.2. Diagrama de clases de análisis, correspondiente al caso de uso **Localizar matrícula**.

5.1.2. Diagrama de clases de diseño

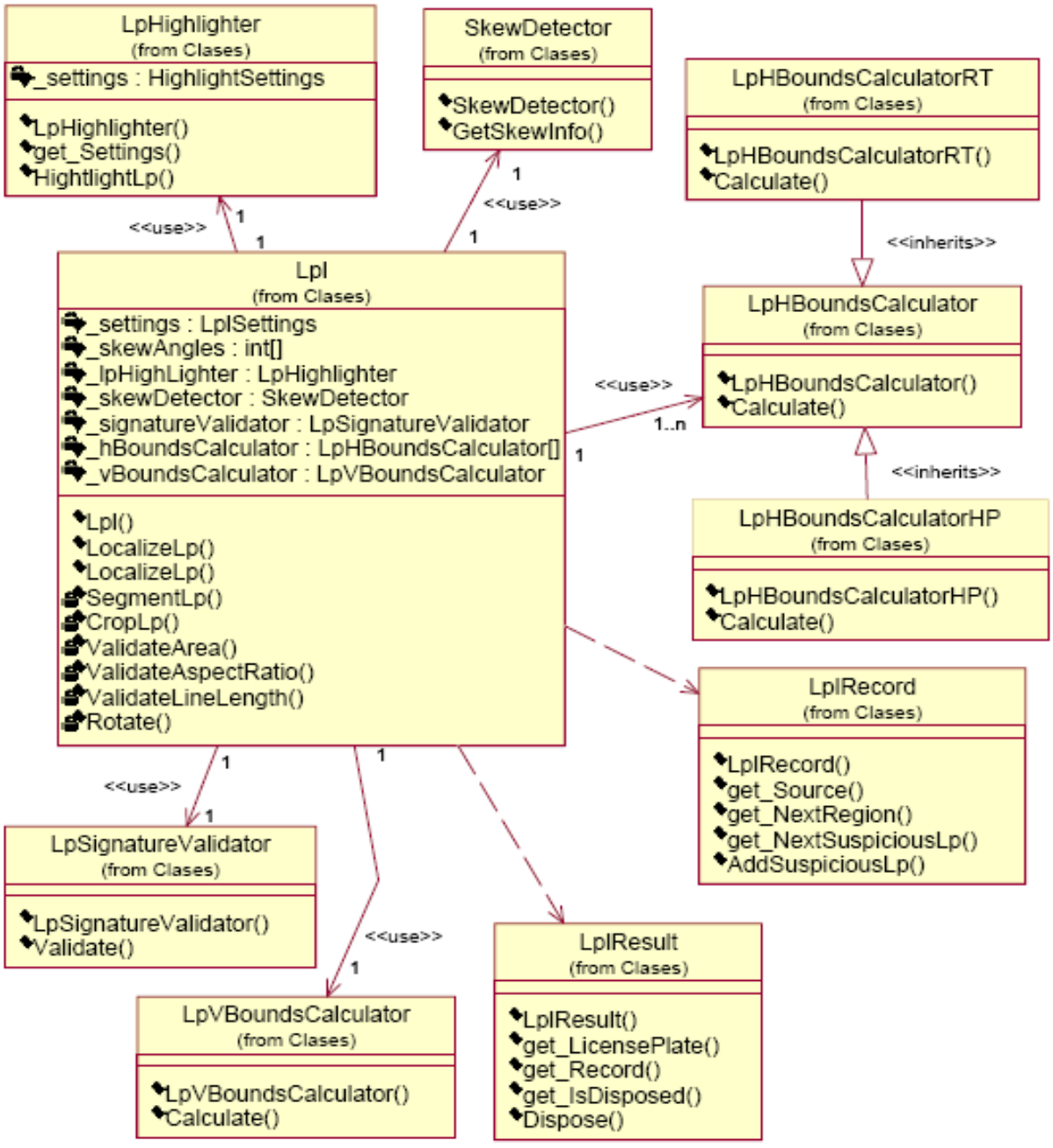


Diagrama 5.3**.** Diagrama de clases de diseño, correspondiente al caso de uso **Localizar matrícula**.

5.1.3. Diagramas de interacción entre las clases de diseño

En esta sección se muestra el diagrama de colaboración, correspondiente a cada escenario del caso de uso *Localizar matrícula* (Diagramas 5.4 y 5.5).

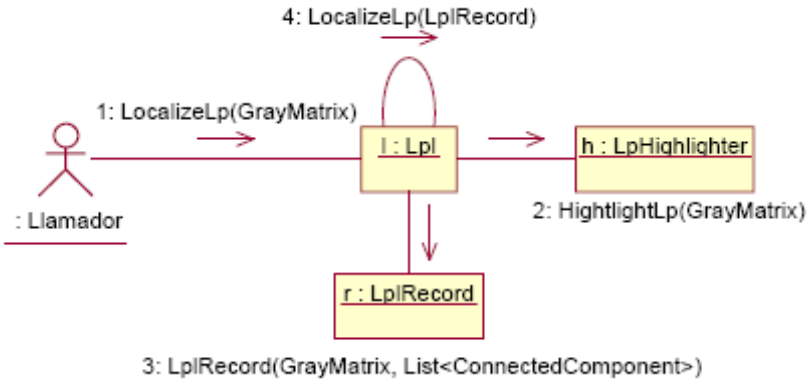


Diagrama 5.4**.** Diagrama de colaboración entre las clases de diseño, correspondiente al **Escenario 1** del caso de uso **Localizar matrícula**.

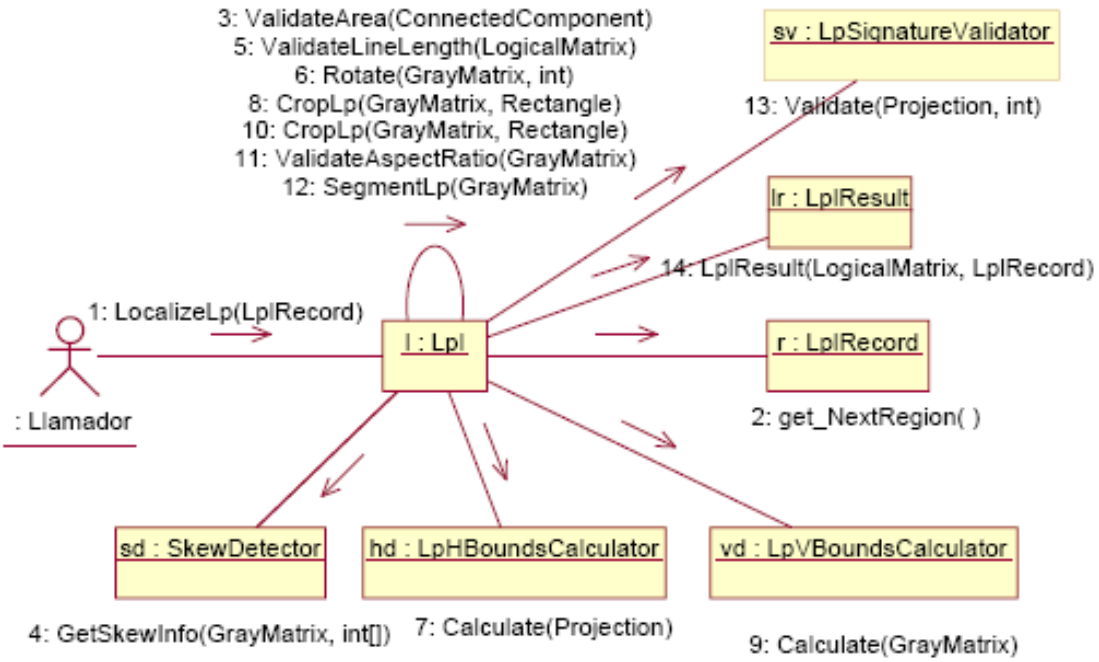


Diagrama 5.5**.** Diagrama de colaboración entre las clases de diseño, correspondiente al **Escenario 2** del caso de uso **Localizar matrícula**.

5.1.4. Descripción textual de las clases de diseño

En el "**Anexo 3. Descripción textual de las clases de diseño**" se describen las clases de diseño que intervienen en la realización del caso de uso *Localizar matrícula*. Las mismas están en correspondencia con el lenguaje de programación C# 2.x, y han sido agrupadas en clases **entidad**, clases **control** y clases **interfaz**.

5.2. Análisis y diseño. Caso de uso "*Extraer caracteres de la matrícula*"

En esta sección se presentan las clases de análisis, las clases de diseño y los diagramas de colaboración, correspondientes a la realización del caso de uso *Extraer caracteres de la matrícula* (Diagrama 5.6).

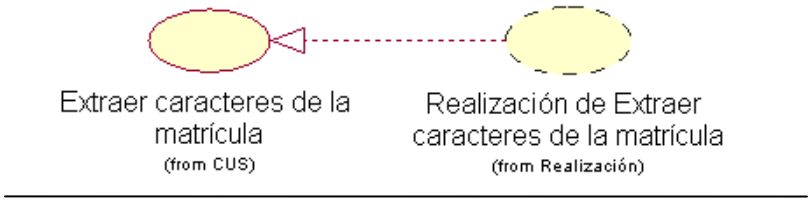


Diagrama 5.6. Realización de l caso de uso **Extraer caracteres de la matrícula**.

5.2.1. Diagrama de clases de análisis

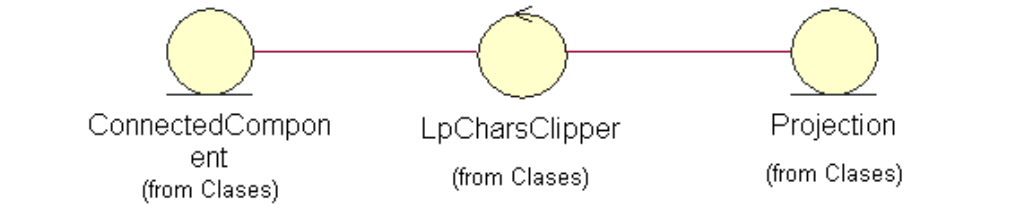


Diagrama 5.7. Diagrama de clases de análisis, correspondiente al caso de uso **Extraer caracteres de la matrícula**.

5.2.2. Diagrama de clases de diseño

En esta sección se expone el diagrama de clases de diseño, correspondiente al caso de uso *Extraer caracteres de la matrícula* (Diagrama 5.8). Nótese que en este diagrama no se muestran las clases de diseño correspondientes a las clases de análisis **Projection** y **ConnectedComponent**. Dichas clases serán incorporadas desde un artefacto reutilizable, existente en el grupo de proyecto.

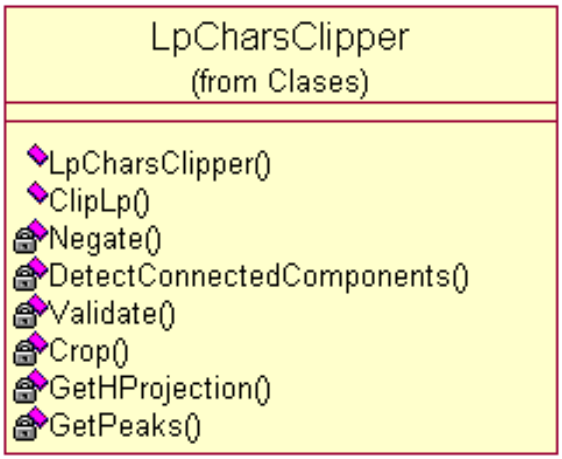


Diagrama 5.8. Diagrama de clases de diseño, correspondiente al caso de uso **Extraer caracteres de la matrícula**.

5.2.3. Diagramas de interacción entre las clases de diseño

En esta sección se muestra el diagrama de colaboración, correspondiente a cada flujo (normal y alterno) del caso de uso *Extraer caracteres de la matrícula* (Diagramas 5.9 y 5.10).

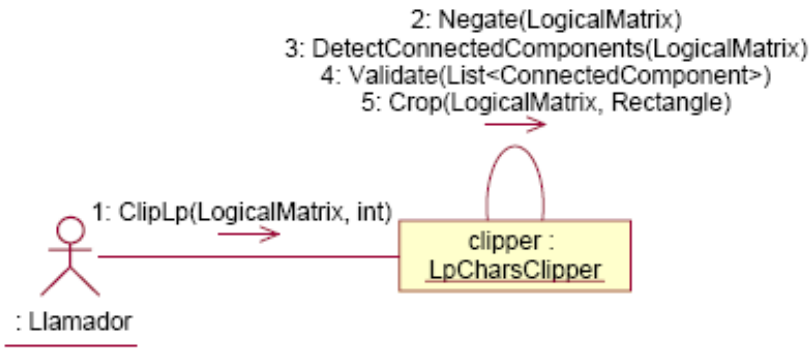


Diagrama 5.9. Diagrama de colaboración entre las clases de diseño, correspondiente al **Flujo normal** del caso de uso **Extraer caracteres de la matrícula**.

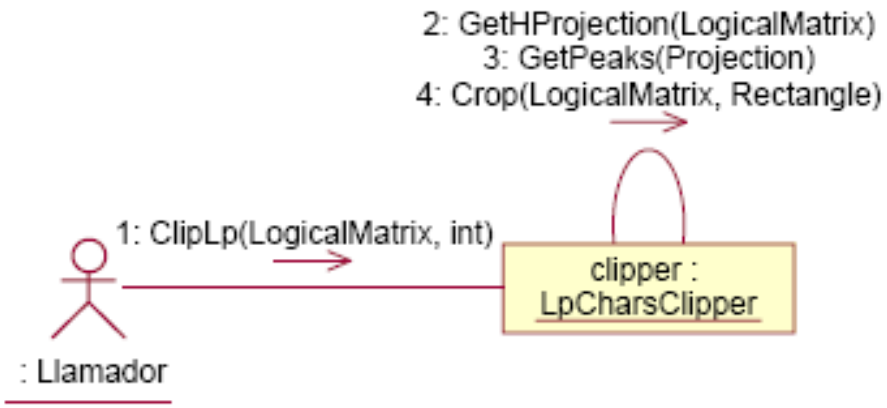


Diagrama 5.10**.** Diagrama de colaboración entre las clases de diseño, correspondiente al **Flujo alterno** del caso de uso **Extraer caracteres de la matrícula**.

5.2.4. Descripción textual de las clases de diseño

En el "**Anexo 3. Descripción textual de las clases de diseño**" se describen las clases de diseño que intervienen en la realización del caso de uso *Extraer caracteres de la matrícula*. Las mismas están en correspondencia con el lenguaje de programación C# 2.x, y han sido agrupadas en clases **entidad**, clases **control** y clases **interfaz**.

CONCLUSIONES

En este capítulo se han presentado las clases de análisis y las clases de diseño, que participan en la realización de los casos de uso del sistema propuesto. Además, se reflejaron los diagramas de colaboración entre las clases de diseño y se describieron, detalladamente, cada una de estas clases.

REFERENCIAS

[1] Archer, Tom. *C# a fondo.* Redmond, Washington: Mc Graw Hill, 2001.

[2] Boogs, Wendy, y Michael Boogs. *Mastering UML with Rational Rose 2002.* California: SYBEX, 2002.

[3] Cooper, James W. *Introduction to Design Patterns in C#.* 2002.

[4] Cwalina, Krzysztof, y Brad Adams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries.* Redmond, Washington: Addison-Wesley Professional, 2005.

[5] Gamma, Erich, Richard Helm, Ralph Jonhson, y John Vlissides. *Design Patterns.* 1994.

[6] Harvey, Deitel M, Paul J Deitel, Jeffrey A Listfield, Tem R Nieto, Cheryl H Yaeger, y Marina Zlaktina. *C# How to program.* Prentice Hall, 2001.

[7] IBM. *Rational Unified Process Online Help.* 2003.

[8] Jacobson, Ivar, Grady Booch, y James Rumbaugh. *El Proceso Unificado de Desarrollo.* Addison Wesley Logman, 2000.

[9] Meier, J. D, Srinath Vasireddy, Ashish Babbar, y Alex Mackman. *Improving .NET application performance and scalability.* Redmond, Washington, 2004.

[10] UCI. «Fase de elaboración. Flujo de análisis y diseño. Modelo de análisis.» Conferencia de Ingeniería de Software I, La Habana, 2008.

Capítulo 6

IMPLEMENTACIÓN

En este capítulo se describen los componentes ejecutables involucrados en el sistema propuesto. Además, se refleja el diagrama de componentes de dicho sistema.

INTRODUCCIÓN

El objetivo del flujo de trabajo **Implementación**, del Proceso Unificado de Desarrollo, es desarrollar la arquitectura y el sistema como un todo (Jacobson, Booch y Rumbaugh 2000). Los trabajadores que intervienen en el mismo son el Arquitecto, el Programador y el Integrador (IBM 2003).

En el flujo de trabajo **Implementación**, se implementan las clases y los subsistemas encontrados durante el **Diseño**. Además se distribuye el sistema, asignando componentes ejecutables a nodos en el diagrama de despliegue, y se prueban e integran los componentes individuales (Jacobson, Booch y Rumbaugh 2000).

En este capítulo se aborda la implementación del sistema propuesto. Lo que resta del mismo está organizado como sigue. En la **Sección 6.1** se describen los componentes ejecutables involucrados en el sistema. Luego, en la **Sección 6.2**, se refleja el diagrama de componentes del sistema.

6.1. Descripción de los componentes ejecutables

El código fuente, que se corresponde con los algoritmos propuestos en esta tesis, ha sido expuesto en el ensamblado (Archer 2001) **Anpr.dll.** Este ensamblado (Anpr.dll) está constituido por los espacios de nombres (Ferguson, Patterson y Beres 2003) **Anpr.LpLocalization** y **Anpr.LpCharsClipping**.

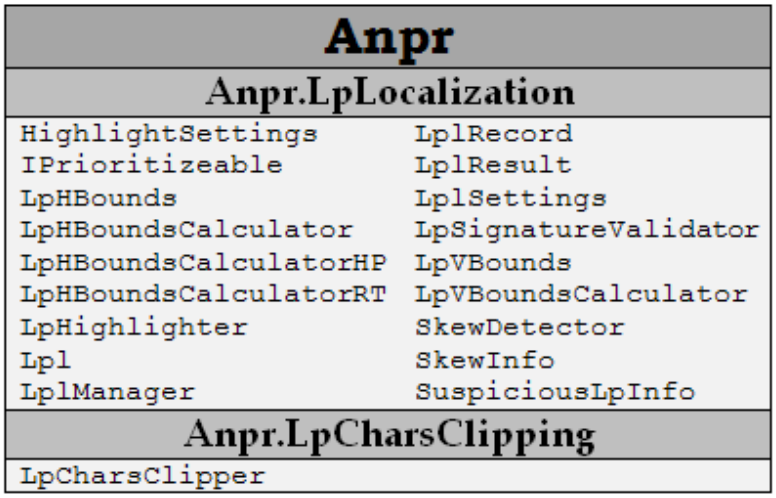


Figura 6.1. Espacios de nombres y tipos existentes en el ensamblado **Anpr.dll**.

El espacio de nombres **Anpr.LpLocalization** contiene los tipos que permiten localizar la matrícula del vehículo, en la imagen digital. Por su parte, el espacio de nombres **Anpr.LpCharsClipping** contiene los tipos que permiten segmentar (en caracteres) la imagen digital de la matrícula (Fig 6.1).

El elemento reutilizable **Mtk.dll** ha sido utilizado durante el desarrollo del sistema propuesto. Este elemento reutilizable está constituido por tipos que permiten procesar y analizar imágenes digitales. La detección de discontinuidades (mediante el operador de Sobel), el etiquetado de las componentes conexas y el análisis de proyecciones, constituyen funcionalidades expuestas en dicho artefacto.

6.2. Diagrama de componentes del sistema

Diagrama 6.1. Diagrama de componentes del sistema propuesto.

CONCLUSIONES

En este capítulo se han descrito los componentes ejecutables involucrados en el sistema propuesto. Además, se reflejó el diagrama de componentes de dicho sistema.

REFERENCIAS

[1] Archer, Tom. *C# a fondo.* Redmond, Washington: Mc Graw Hill, 2001.

[2] Boogs, Wendy, y Michael Boogs. *Mastering UML with Rational Rose 2002.* California: SYBEX, 2002.

[3] Cwalina, Krzysztof, y Brad Adams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries.* Redmond, Washington: Addison-Wesley Professional, 2005.

[4] Ferguson, Jeff, Brian Patterson, y Jason Beres. *La biblia de C#.* Madrid: Anaya Multimedia, 2003.

[5] Harvey, Deitel M, Paul J Deitel, Jeffrey A Listfield, Tem R Nieto, Cheryl H Yaeger, y Marina Zlaktina. *C# How to program.* Prentice Hall, 2001.

[6] IBM. *Rational Unified Process Online Help.* 2003.

[7] Jacobson, Ivar, Grady Booch, y James Rumbaugh. *El Proceso Unificado de Desarrollo.* Addison Wesley Logman, 2000.

Capítulo 7

PRUEBA, RESULTADOS Y DISCUSIÓN

En este capítulo se especifican los casos de prueba y los procedimientos de prueba del sistema propuesto. Además se demuestra la compatibilidad del sistema con la plataforma libre Mono y se discuten la sensibilidad, la especificidad, el índice de falsos positivos, el índice de falsos negativos y la complejidad temporal, de los algoritmos presentados.

INTRODUCCIÓN

El objetivo del flujo de trabajo **Prueba**, del Proceso Unificado de Desarrollo, es verificar el resultado de la **Implementación** (Jacobson, Booch y Rumbaugh 2000). Los trabajadores que intervienen en el mismo son el Administrador de prueba, el Analista de prueba, el Diseñador de prueba y el Probador (IBM 2003).

En las pruebas se definen los casos de prueba y los procedimientos de prueba. Un caso de prueba especifica una forma de probar el sistema; incluye las entradas, los resultados esperados y las condiciones bajo las que se debe verificar el sistema. Por su parte, un procedimiento de prueba indica cómo realizar uno o varios casos de prueba (Jacobson, Booch y Rumbaugh 2000).

En este flujo de trabajo (Prueba) se aplican pruebas de especificación (*caja negra*) y de estructura (*caja blanca*). Las pruebas de caja negra se ejecutan sobre la interfaz del sistema mientras que las pruebas de caja blanca se ejecutan sobre la estructura interna del software (Pressman 2002). Unido a los métodos de caja blanca y de caja negra, súmense las técnicas de caja gris (Braude 2003). Estás técnicas son el resultado de la combinación de los 2 métodos anteriores (caja blanca y caja negra).

En este capítulo se abordan las pruebas del sistema propuesto. Lo que resta del mismo está organizado como sigue. En las **Secciones 7.1** y **7.2** se especifican los casos de prueba y los procedimientos de prueba, respectivamente. Luego, en la **Sección 7.3** se exponen la especificidad, la sensibilidad, el índice de falsos positivos (FAR) y el índice de fasos negativos (FRR) de los algoritmos

propuestos. En la **Sección 7.4** se discute la efectividad de los métodos ANPR diseñados y en la **Sección 7.5** se estudia la complejidad temporal de los mismos. Seguido, en la **Sección 7.6** se demuestra la compatibilidad del sistema con la plataforma libre Mono. Para ello, se utilizó la herramienta **Mono Migration Analyzer** (MoMA)

7.1. Especificación de los casos de prueba

En el **"Anexo 4. Artefactos de prueba"**, se presentan los casos de prueba correspondientes a los casos de uso del sistema propuesto.

7.2. Especificación de los procedimientos de prueba

En el **"Anexo 4. Artefactos de prueba"**, se presentan los procedimientos de prueba correspondientes a los casos de uso del sistema propuesto.

7.3. Efectividad de los algoritmos propuestos

La efectividad de un algoritmo se calcula a partir de varios indicadores. Los más utilizados son la **sensibilidad**, la **especificidad**, el **índice de falsos positivos** (FAR) y el **índice de falsos negativos** (FRR). En esta sección se reflejan los indicadores de los algoritmos propuestos en esta tesis.

$$\boldsymbol{Sensibilidad} = \frac{VP}{VP + FN} \quad \textbf{(7.1)}$$

$$\boldsymbol{Especificidad} = \frac{VN}{VN + FP} \quad \textbf{(7.2)}$$

$$\boldsymbol{FAR} = \frac{FP}{VN + FP} \quad \textbf{(7.3)}$$

$$\boldsymbol{FRR} = \frac{FN}{VP + FN} \quad \textbf{(7.4)}$$

:

Donde:

- **VP**: Cantidad eventos verdaderos positivos.
- **VN**: Cantidad de eventos verdaderos negativos.
- **FP**: Cantidad de eventos falsos positivos.
- **FN**: Cantidad de eventos falsos negativos.

7.3.1. Localización de la matrícula

Se verificó la efectividad del algoritmo de localización en **145** imágenes digitales. (Véase "**Anexo 5. Resultados del algoritmo de localización de matrículas.**") Las mismas fueron capturadas bajo varias condiciones de iluminación (e.g nublado, soleado, oscuro) y se corresponden con vehículos de matrículas de diferentes formatos (e.g. estatal, particular).

En la **Tabla 7.1** se muestra la cantidad de eventos verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos ocurridos (Resonance Pub 2008). Posteriormente, se refleja el valor de la sensibilidad, la especificidad, el índice de falsos positivos y el índice de falsos negativos, del algoritmo propuesto (**Tabla 7.2**).

Tabla 7.1. Eventos ocurridos.

Tipo	**Cantidad**
Verdadero positivo (VP)	110
Verdadero negativo (VN)	27
Falso positivo (FP)	50
Falso negativo (FN)	10

Tabla 7.2. Indicadores del algoritmo de localización.

Indicador	**Valor**
Sensibilidad	0.916666667
Especificidad	0.350649351
Índice de falsos positivos (FAR)	0.649350649
Índice de falsos negativos (FRR)	0.083333333

7.3.2. Extracción de los caracteres de la matrícula

Se verificó la efectividad del algoritmo de extracción en **109** imágenes de matrículas. (Véase "**Anexo 6. Resultados del algoritmo de segmentación de matrículas.**") En la **Tabla 7.3** se resume el comportamiento del mismo.

Tabla 7.3. Indicadores del algoritmo de extracción.

Indicador	Valor
Total de aciertos	85
Total de fallos	24
% de aciertos	77.98165138
% de fallos	22.01834862

7.4. Discusión de la efectividad de los algoritmos propuestos

La sensibilidad y la especificidad indican la capacidad de un algoritmo para detectar el cumplimiento y el no cumplimiento de una condición, respectivamente. Además, el índice de falsos positivos es la probabilidad de ocurrencia de eventos falsos positivos, y el índice de falsos negativos es la probabilidad de ocurrencia de eventos falsos negativos. (Véase el Glosario de términos.)

El algoritmo de localización está caracterizado por una alta sensibilidad. (Consecuencia del bajo índice de falsos negativos.) Además, el índice de falsos positivos es elevado. Dicho comportamiento fue introducido en el diseño del algoritmo, para minimizar la cantidad de eventos falsos negativos. Nótese que un índice elevado de falsos negativos obstaculiza (o impide) el despliegue del componente de localización en circunstancias reales.

Es necesario destacar que el algoritmo de localización se diseñó de modo que, la mayoría de los resultados falsos positivos se generen después de los resultados verdaderos positivos (Gráf 7.1). Esta conducta minimiza la complejidad temporal (total) del sistema ANPR y asegura una alta tasa de acierto.

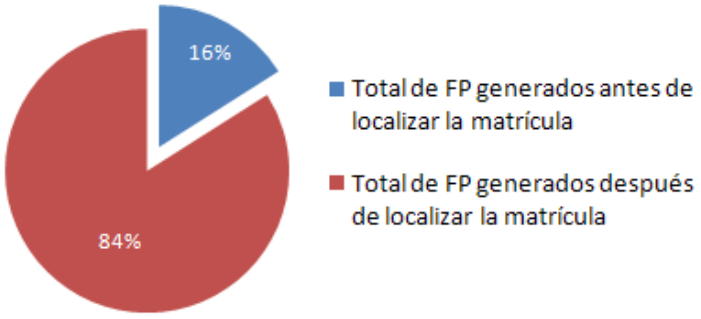


Gráfico 7.1. Distribución de los eventos falsos positivos.

La tasa de fallos, del algoritmo de extracción de los caracteres de la matrícula, es considerable. Lo anterior se debe a la incapacidad del algoritmo para segmentar, exitosamente, matrículas con caracteres unidos (Fig 7.1).

Figura 7.1. Resultado de la extracción en una matrícula con caracteres unidos.

7.5. Complejidad temporal de los algoritmos

Los algoritmos propuestos en esta tesis son no deterministas (i.e. el flujo de acciones depende de las características de los datos de entrada). Debido a ello (y a otros factores), en esta sección no se realiza un análisis formal de la complejidad temporal de dichos algoritmos. En su lugar, se muestra el impacto de cada etapa del algoritmo de localización y se compara la complejidad temporal de los métodos propuestos para extraer los caracteres de la matrícula.

La generación de los candidatos (Sección 2.1), la corrección de la inclinación (Secciones 2.3 y 2.4), y la segmentación (Sección 2.9), son las etapas de mayor impacto en la complejidad temporal del algoritmo de localización (Gráf 7.2). Recuérdese que durante la generación de los candidatos, se detectan los bordes verticales, se aplica una dilatación con un elemento estructurante rectangular y se etiquetan las componentes conectadas. Esta última acción, generalmente, es muy costosa.

Por su parte, la corrección de la inclinación involucra la detección de los bordes horizontales, el cálculo de la Transformada Radon y la operación de rotación (con interpolación bilineal). En cuanto a la segmentación del candidato, el cálculo del umbral adaptativo es el hito más costoso.

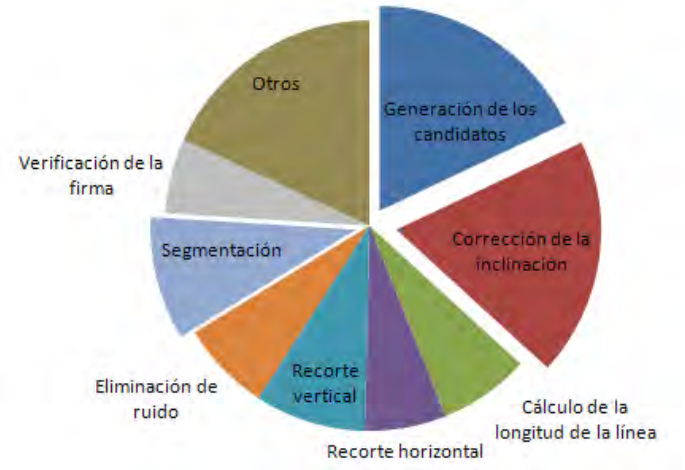


Gráfico 7.2. Distribución de la complejidad temporal del algoritmo de localización.

El método de extracción de los caracteres de la matrícula, basado en el análisis de las componentes conectadas (Sección 3.1), es más lento que el método basado en la proyección horizontal (Sección 3.2). Sin embargo, el algoritmo propuesto intenta extraer los caracteres analizando las componentes conectadas. Luego, si el resultado es incorrecto, se recortan los caracteres analizando la proyección horizontal. Se ha elegido este orden porque el método basado en el análisis de las componentes conectadas es la solución más confiable y natural (Gráf 7.3).

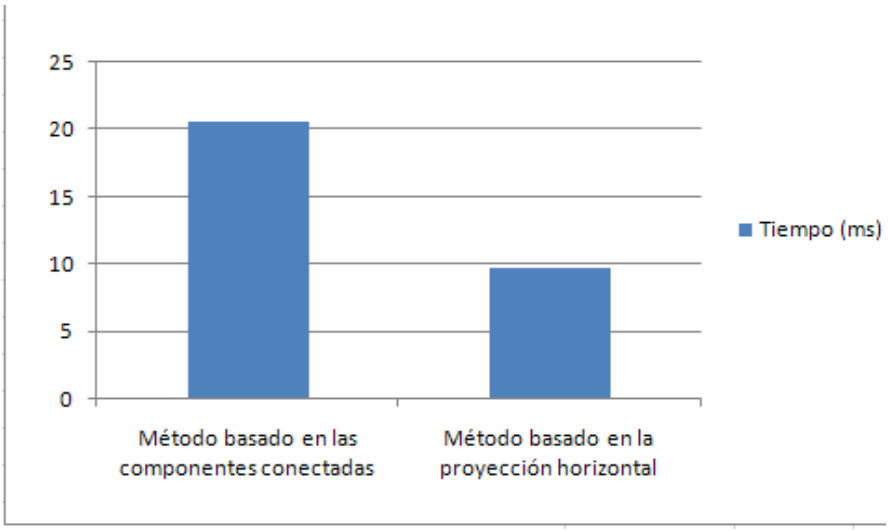


Gráfico 7.3. Complejidad temporal de los métodos de extracción de caracteres.

7.6. Compatibilidad con la plataforma libre Mono

La herramienta Mono Migration Analyzer (MoMA) permite analizar si una aplicación .NET puede ser ejecutada satisfactoriamente en la plataforma Mono (Mono s.f.). La compatibilidad de los ensamblados **Mtk.dll** y **Anpr.dll**, con la plataforma Mono, fue comprobada utilizando la herramienta MoMA (v. 1.2.6). Los resultados obtenidos (Fig 7.2) demuestran que el sistema propuesto en esta tesis es completamente portable hacia la plataforma libre.

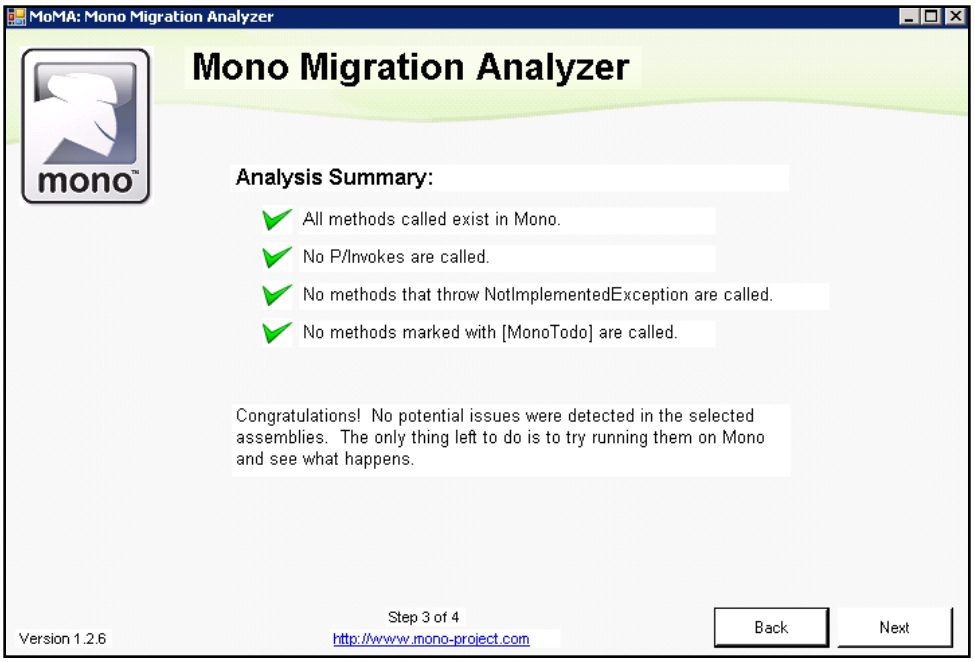


Figura 7.2. Compatibilidad de los ensamblados **Anpr.dll** y **Mtk.dll** con la plataforma libre Mono.

CONCLUSIONES

En este capítulo se han especificado los casos de prueba y los procedimientos de prueba del sistema propuesto. Además se demostró la compatibilidad del sistema con la plataforma libre Mono y se discutieron la sensibilidad, la especificidad, el índice de falsos positivos, el índice de falsos negativos y la complejidad temporal, de los algoritmos ANPR presentados.

REFERENCIAS

[1] Braude, J. Ingeniería de software: Una perspectiva orientada a objetos. Ra-ma, 2003.

[2] IBM. Rational Unified Process Online Help. 2003.

[3] Jacobson, Ivar, Grady Booch, y James Rumbaugh. El Proceso Unificado de Desarrollo. Addison Wesley Logman, 2000.

[4] Mono. «MoMA.» Mono. http://www.mono-project.com/MoMA.

[5] Pressman, Roger S. Ingeniería de software: Un enfoque práctico. Addison-Wesley, 2002.

[6] Resonance Pub. «Biometric Glossary.» Resonance Pub- The Portal to Science, Engineering and Technology. 2008. http://www.resonancepub.com/biometricgl.htm (último acceso: 3 de 3 de 2008).

CONCLUSIONES

En esta tesis se han expuesto las principales características de los **Sistemas de reconocimiento automático del número de la matrícula** (ANPR). Además, se analizaron las tendencias actuales respecto a las técnicas de localización y segmentación de matrículas de vehículos en imágenes digitales.

Unido a lo anterior, se diseñaron e implementaron los componentes "**Localización de la matrícula**" y "**Extracción de los caracteres de la matrícula**" de un sistema ANPR cubano. La sensibilidad, la especificidad, el índice de falsos positivos y el índice de falsos negativos, del componente de localización, es de 0.92, 0.35, 0.65 y 0.08 respectivamente. El componente de extracción ofrece una tasa de acierto de 78 %.

El componente de localización permite localizar matrículas de cualquier formato, es robusto ante la presencia de matrículas inclinadas, opera correctamente bajo varias condiciones de iluminación y es relativamente invariante a la transformación de escala. El componente de extracción utiliza una técnica híbrida en su funcionamiento. Ambos componentes fueron implementados utilizando los lenguajes de programación C# 2.x y Matlab. Además se demostró que el ensamblado .NET creado, es compatible con la plataforma libre Mono.

Los resultados mencionados anteriormente, sugieren que los componentes ANPR desarrollados en esta tesis, están listos para ser integrados a un sistema automático de identificación de vehículos cubano.

RECOMENDACIONES

Los componentes ANPR propuestos en esta tesis no constituyen una panacea; toda creación humana es imperfecta.

En el componente "**Localización de la matrícula**" (Sección 1.1) se pueden realizar mejoras potenciales. Actualmente, el algoritmo de localización es robusto ante la presencia de matrículas inclinadas, pero exige que las imágenes sean capturadas frente al vehículo. Para que el algoritmo opere con imágenes tomadas bajo cualquier ángulo, se podría utilizar la Transformada Radon para estimar los límites verticales de la matrícula, pero el costo temporal de esta solución es significativo.

Unido a lo anterior, el algoritmo de normalización del color del fondo del candidato (Sección 2.9.2) en ocasiones falla (Fig 7). Esta simple inversión de colores puede causar el fracaso del proceso de localización. Además, debe mejorarse el algoritmo de verificación de la firma del candidato (Sección 2.10). Es muy importante destacar que todas estas mejoras deben generar un balance adecuado entre el índice de falsos positivos y el índice de falsos negativos.

Figura 7. Error en la normalización del color del fondo del candidato.
a) Matricula en escala de grises. **b)** Normalización errónea del color del fondo de la matrícula.

Por su parte, el componente "**Extracción de los caracteres de la matrícula**" (Sección 1.1) no es robusto ante la unión de caracteres (Fig 8). La solución a este problema es vital para poder reconocer matrículas ruidosas.

Figura 8. Unión de los caracteres de la matrícula.

En cuanto a los detalles de implementación, se recomienda la paralelización de los algoritmos propuestos (Foster 1995). Si se elige un entorno con memoria distribuida, la Interfaz de paso de mensajes (**MPI**) constituye una buena opción (Indiana University 2008). Si por el contrario se elige un entorno con memoria compartida, se sugiere la implementación de los modelos basados en eventos y en la interfaz *System.IAsyncResult* (Meier, y otros 2004).

En el ensamblado .NET creado, toda la memoria que se utiliza es manejada (Meier, y otros 2004). Lo anterior afecta negativamente el rendimiento debido a las frecuentes recolecciones de memoria. Este problema puede ser atenuado implementando un tipo manejado **Matrix** (Diagrama 1). Este tipo debe almacenar los datos de los puntos, en la memoria no manejada (Meier, y otros 2004). Para controlar esta memoria no administrada, el tipo **Matrix** debería exponer los principios dictados por el patrón **Dispose** (Meier, y otros 2004).

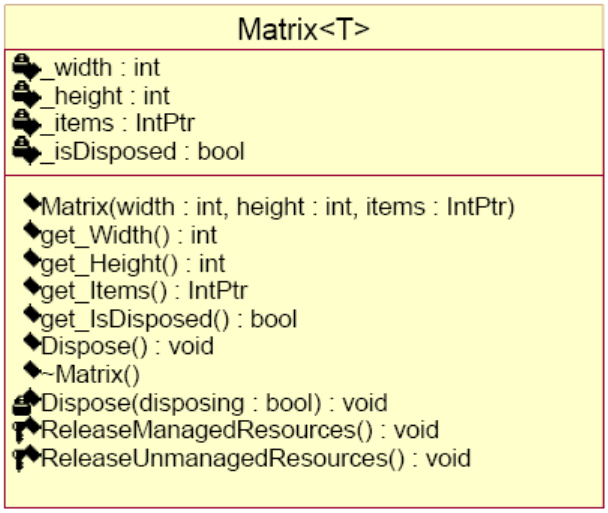


Diagrama 1. Definición de los miembros del tipo **Matrix**.

El componente de localización recorta la imagen de entrada, sucesivamente, hasta encontrar la matrícula. Esta característica ha sido introducida con el objetivo de optimizar el rendimiento del componente de localización y puede conseguirse más provecho de ella — el tiempo dedicado a la operación de recorte puede suprimirse completamente. Para ello, defínase el tipo **MatrixRegion** (Diagrama 2) y hágase que la operación de recorte retorne una instancia de este tipo. Además, para evitar

la corrupción de los datos, el tipo **MatrixRegion** debería exponer los principios de la técnica ***Copy-on-Write*** (Wikipedia 2008)***.***

A pesar de los señalamientos realizados, los componentes ANPR propuestos en esta tesis son funcionales. (Esta conclusión quedó demostrada en el Capítulo 7: Prueba, resultados y discusión.) Para alcanzar el sistema ANPR completo, se deben desarrollar los componentes "**Reconocimiento de los caracteres de la matrícula**" (Sección 1.1) y "**Análisis de la sintaxis de la matrícula**" (Sección 1.1).

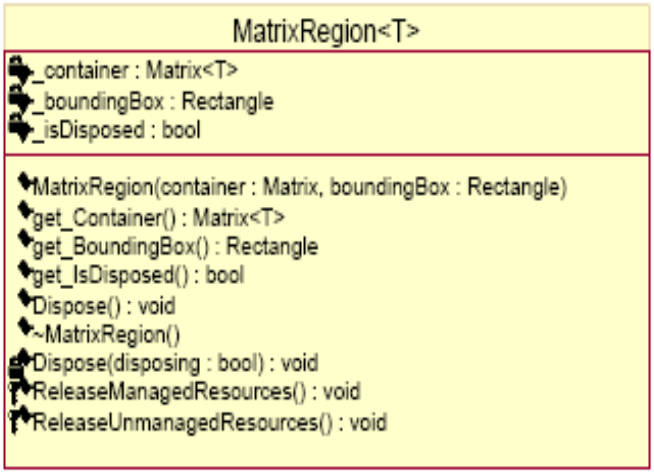


Diagrama 2**.** Definición de los miembros del tipo **MatrixRegion**.

El componente "**Reconocimiento de los caracteres de la matrícula**" utiliza técnicas de reconocimiento óptico de caracteres (OCR). En Internet existen sistemas OCR, de gran calidad, disponibles libremente. Algunos de ellos son (Softi Software 2007) y (Simple software 2007). Se recomienda utilizar alguno de estos sistemas OCR (u otro) para desarrollar el componente de reconocimiento. Respecto al componente "**Análisis de la sintaxis de la matrícula**", realícese un estudio profundo sobre las técnicas de limpieza de textos y de análisis sintáctico.

BIBLIOGRAFÍA

[1] Acharya, Tinku, y Ajoy K Ray. *Image processing:Principles and applications.* New Jersey: Wiley-Interscience, 2005.

[2] AQTime. «AQTime Profiler.» *Automated profiling and debugging.* 2007. http://www.automatedqa.com/products/aqtime/index.asp.

[3] Archer, Tom. *C# a fondo.* Redmond, Washington: Mc Graw Hill, 2001.

[4] Arora, Geetanjali, Balasubramaniam Aiaswamy, y Nitin Pandey. *Programación Microsoft C#.* Madrid: Anaya Multimedia, 2002.

[5] Batanga, Autos. «¿Cuántos vehículos hay en todo el mundo?» *Batanga autos.* http://autos.batanga.com/Cuantos_vehiculos_hay_en_todo_el_mundo?/44680588-6045-4C2D-B656-3A78BCF7A2CC.htm (último acceso: 9 de 11 de 2007).

[6] Boogs, Wendy, y Michael Boogs. *Mastering UML with Rational Rose 2002.* California: SYBEX, 2002.

[7] Braude, J. *Ingeniería de software: Una perspectiva orientada a objetos.* Ra-ma, 2003.

[8] Butow, Eric, y Tommy Ryan. *C#: Your visual blueprint for building .NET applications.* New York: Hungry Minds, Inc, 2002.

[9] Cardero Álvarez, Rafael L., y Jesmar E. Fajardo Martín. «Localización y segmentación de matrículas en imágenes de vehículos.» RECPAT´2007 Proceedings, La Habana, 2007.

[10] Chonoles, Michael Jesse, y James A. Schardt. *UML 2 for dummies.* New York: Wiley, 2003.

[11] CitySync. *CitySync- Leaders in Automatic Number Plate Recognition.* http://www.citysync.co.uk/ (último acceso: 29 de 11 de 2007).

[12] Codeplex. «Sandcastle.» *Sandcastle- Codeplex.* 15 de 1 de 2008. http://www.codeplex.com/Sandcastle.

[13] Comaniciu, Dorin, y Peter Meer. «Mean shift: A robust approach toward feature space analysis.» *IEEE Transactions on pattern analysis and machine intelligence*, 2002.

[14] Cooper, James W. *Introduction to Design Patterns in C#.* 2002.

[15] CRS. *Computer Recognition Systems.* http://www.crs-traffic.co.uk/index.html (último acceso: 2 de 12 de 2007).

[16] Cubaweb. «El primer auto en La Habana.» *Cubaweb.* http://www.nnc.cubaweb.cu/historia/historia15.htm (último acceso: 3 de 12 de 2007).

[17] Cwalina, Krzysztof, y Brad Adams. *Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries.* Redmond, Washington: Addison-Wesley Professional, 2005.

[18] Day, Robert A. *How to Write & Publish a Scientific Paper.* Phoenix, Arizona.: ORYX PRESS, 1998.

[19] Duda, R O, y P E Hart. «Use of the Hough Transformation to Detect Lines and Curves in Pictures.» *ACM*, 1972: 4.

[20] Eddins, Steve. «Connected component labeling - Wrapping up.» *Mathworks.* 2007. http://blogs.mathworks.com/steve/2007/06/13/connected-component-labeling-wrapping-up/ (último acceso: 2007 de 12 de 11).

[21] Fábrega, Francisco Javier Thayer, y Joshua D. Guttman. «Copy on write.» Article, 1995.

[22] Ferguson, Jeff, Brian Patterson, y Jason Beres. *La biblia de C#.* Madrid: Anaya Multimedia, 2003.

[23] Foster, Ian. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Addison Wesley, 1995.

[24] Gamma, Erich, Richard Helm, Ralph Jonhson, y John Vlissides. *Design Patterns.* 1994.

[25] Gonzalez, Rafael C, y Richard E Woods. *Digital Image Processing.* New Jersey: Prentice Hall, 2002.

[26] Hamilton, Kim, y Russell Miles. *Learning UML 2.0.* Sebastopol: O'Reilly, 2006.

[27] Hansen, Henrik, Anders Wang Kristensen, Morten Porsborg Kohler, Allan Weber Mikkelsen, Jens Mejdahl Pedersen, y Michael Trangeled. «Automatic recognition of license plates.» Thesis, 2002.

[28] Harvey, Deitel M, Paul J Deitel, Jeffrey A Listfield, Tem R Nieto, Cheryl H Yaeger, y Marina Zlaktina. *C# How to program.* Prentice Hall, 2001.

[29] Hi-Tech solutions. *Welcome to Hi-Tech solutions.* http://www.htsol.com/ (último acceso: 29 de 11 de 2007).

[30] HPIR. «Image transforms- Hough transform.» *HPIR.* http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm (último acceso: 10 de 12 de 2007).

[31] IBM corporation. *The Rational Unified Process.* 2003.

[32] IBM. *Rational Unified Process Online Help.* 2003.

[33] Icaza, Manuel de. «Mono: la nueva plataforma de desarrollo para Unix, Linux y MacOS.» *UOC- La Universitat Oberta de Catalunya.* 10 de 2004. http://www.uoc.edu/dt/esp/deicaza0904.html.

[34] Indiana University. «MPI.NET: High-Performance C# Library for Message Passing.» *The Open Systems Laboratory - Indiana University Research.* 21 de 2 de 2008. http://www.osl.iu.edu/research/mpi.net/ (último acceso: 10 de 3 de 2008).

[35] InPeg. *License plate recognition system.* http://www.inpeg.com/index.php (último acceso: 29 de 11 de 2007).

[36] Jacobson, Ivar, Grady Booch, y James Rumbaugh. *El Proceso Unificado de Desarrollo.* Addison Wesley Logman, 2000.

[37] —. *El Proceso Unificado de Desarrollo de Software.* Addison Wesley, 1999.

[38] Java Tech. «Java course.» 2004. http://www.particle.kth.se/~lindsey/JavaCourse/Book/courseMap.html (último acceso: 16 de 12 de 2006).

[39] MacQueen, J B. «Some Methods for classification and Analysis of Multivariate Observations.» *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

[40] MARTINSKY, ONDREJ. «ALGORITHMIC AND MATHEMATICALPRINCIPLES OF AUTOMATIC NUMBER PLATE RECOGNITION SYSTEMS.» B.Sc. Thesis, Brno , 2007.

[41] Mathworks. *The Mathworks.* 2007. http://www.mathworks.com.

[42] Meier, J. D, Srinath Vasireddy, Ashish Babbar, y Alex Mackman. *Improving .NET application performance and scalability.* Redmond, Washington, 2004.

[43] Microsoft. *C# language specification.* Redmond, 2006.

[44] —. «Visual Studio 2005 Team Suite.» *MSDN.* 2007. http://msdn2.microsoft.com/en-us/teamsystem/aa718822.aspx.

[45] Molina, Rafael. *Introducción al procesamiento y análisis de imágenes digitales.* Granada, 1998.

[46] Mono. «MoMA.» *Mono.* http://www.mono-project.com/MoMA.

[47] MSDN. «About GDI+.» *Microsoft Developer Network- MSDN.* 2007. http://msdn2.microsoft.com/en-us/library/ms533797(VS.85).aspx.

[48] —. «System.Drawing namespace.» *Microsoft Developer Network- MSDN.* 2008. http://msdn2.microsoft.com/en-us/library/system.drawing.aspx.

[49] Pressman, Roger S. *Ingeniería de software: Un enfoque práctico.* Addison-Wesley, 2002.

[50] Rational. *Using Rose.* 2000.

[51] Resonance Pub. «Biometric Glossary.» *Resonance Pub- The Portal to Science, Engineering and Technology.* 2008. http://www.resonancepub.com/biometricgl.htm (último acceso: 3 de 3 de 2008).

[52] Ron, Bar-Hen, y Johanan Erez. «A Real-time vehicle License Plate Recognition System.» Conference Report, Tel Aviv, 2002.

[53] Santos, David A. «Consejos para una comunicación científica eficaz.» 2005. http://www.com.uvigo.es/~dsantos/guia_v1.pdf.

[54] Shapiro, Vladimir, Dimo Dimov, Stefan Bonchev, Veselin Velichkov, y Giorgi Gluhchev. «Adaptive license plate image extraction.» Conference Report, 2003.

[55] Simple software. «Download SimpleOCR SDK.» *Simple Software- Simple solutions for document management.* 2007. http://www.simpleocr.com/Download.asp.

[56] Softi Software. «Free OCR 2.0.» *Softi Software.* 3 de 8 de 2007. http://www.softdownload.org/fichas/downloads/hardware/scanners/dw_39790_free_ocr_2.0.asp (último acceso: 10 de 3 de 2008).

[57] Stroustrup, Bjarne. *The C++ programming language.* 3rd. New Jersey, 1997.

[58] UCI. «Fase de elaboración. Flujo de análisis y diseño. Modelo de análisis.» Conferencia de Ingeniería de Software I, La Habana, 2008.

[59] UNFPA. «UNFPA state of world population 2007. Unleashing the Potential of Urban Growth.» Report, 2007.

[60] Wikipedia. «C Sharp.» *Wikipedia.* 2006. http://en.wikipedia.org/wiki/C_Sharp (último acceso: 16 de 12 de 2007).

[61] —. «Copy-on-write.» *Wikipedia, the free encyclopedia.* 18 de 2 de 2008. http://en.wikipedia.org/wiki/Copy-on-write.

[62] —. «Hough transform.» *Wikipedia, the free encyclopedia.* http://en.wikipedia.org/wiki/Hough_transform (último acceso: 10 de 12 de 2007).

[63] —. «Matrícula- Automóviles.» *Wikipedia, la enciclopedia libre.* 2008. http://es.wikipedia.org/w/index.php?title=Matr%C3%ADcula_(autom%C3%B3viles).

[64] Zamir. *Zamir Vehicle Recognition.* http://www.zamir.co.il/index1.html (último acceso: 29 de 11 de 2007).

ANEXOS

En esta sección se muestran los artefactos que complementan el cuerpo de este documento.

Anexo 1. Modelo de negocio

En este anexo se justifican los actores y los trabajadores del negocio. Además, se muestra el diagrama de casos de uso del negocio, el diagrama de clases del Modelo de objetos y el diagrama de actividades asociado a cada caso de uso del negocio.

A1.1. Actores y trabajadores del negocio

En esta sección se justifican los actores y los trabajadores del negocio (Tablas A1.1 y A1.2).

Tabla A1.1. Actores del negocio.

Actor	Justificación
Vehículo	Agente que desencadena los procesos **Registrar entrada** y **Registrar salida**, al cruzar un punto de control.

Tabla A1.2. Trabajadores del negocio.

Trabajador	Justificación
Custodio	Agente interno al negocio que participa en los procesos **Registrar entrada** y **Registrar salida**. Interactúa con las entidades de negocio **Registro de vehículos autorizados**, **Registro de entradas, Registro de salidas** y **Matrícula**.

A1.2. Diagramas

En esta sección se muestra el diagrama de casos de uso del negocio (Diagrama A1.1), el diagrama de clases del Modelo de objetos (Diagrama A1.2) y el diagrama de actividades asociado a cada caso de uso del negocio (Diagramas A1.3 y A1.4).

Diagrama A1.1. Diagrama de casos de uso del negocio.

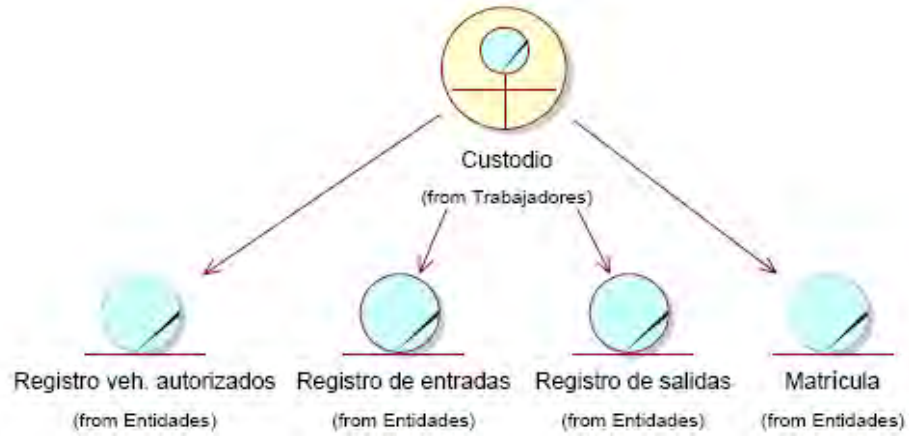


Diagrama A1.2. Diagrama de clases del Modelo de objetos.

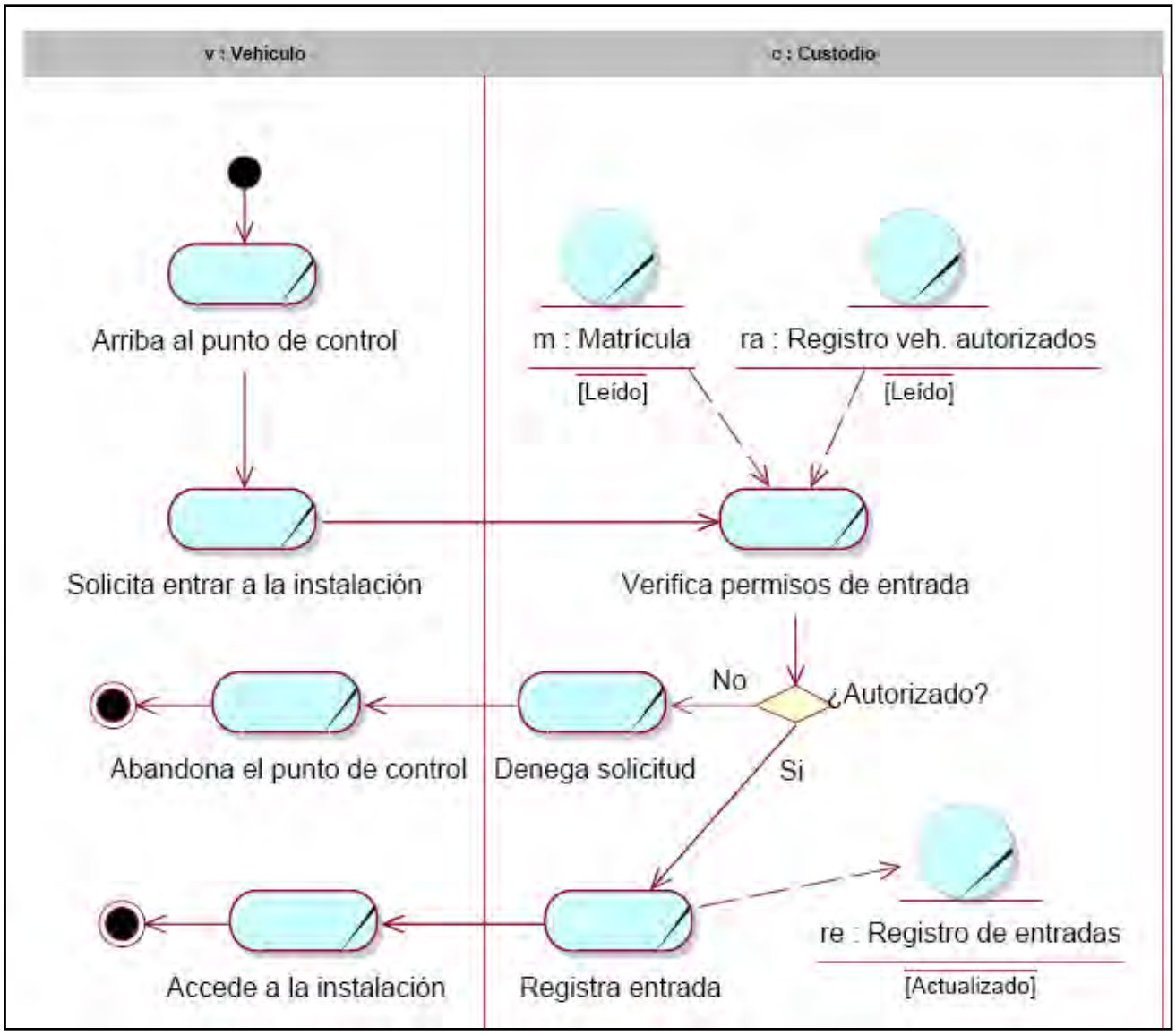


Diagrama A1.3**.** Diagrama de actividades correspondiente al caso de uso de negocio **Registrar entrada**.

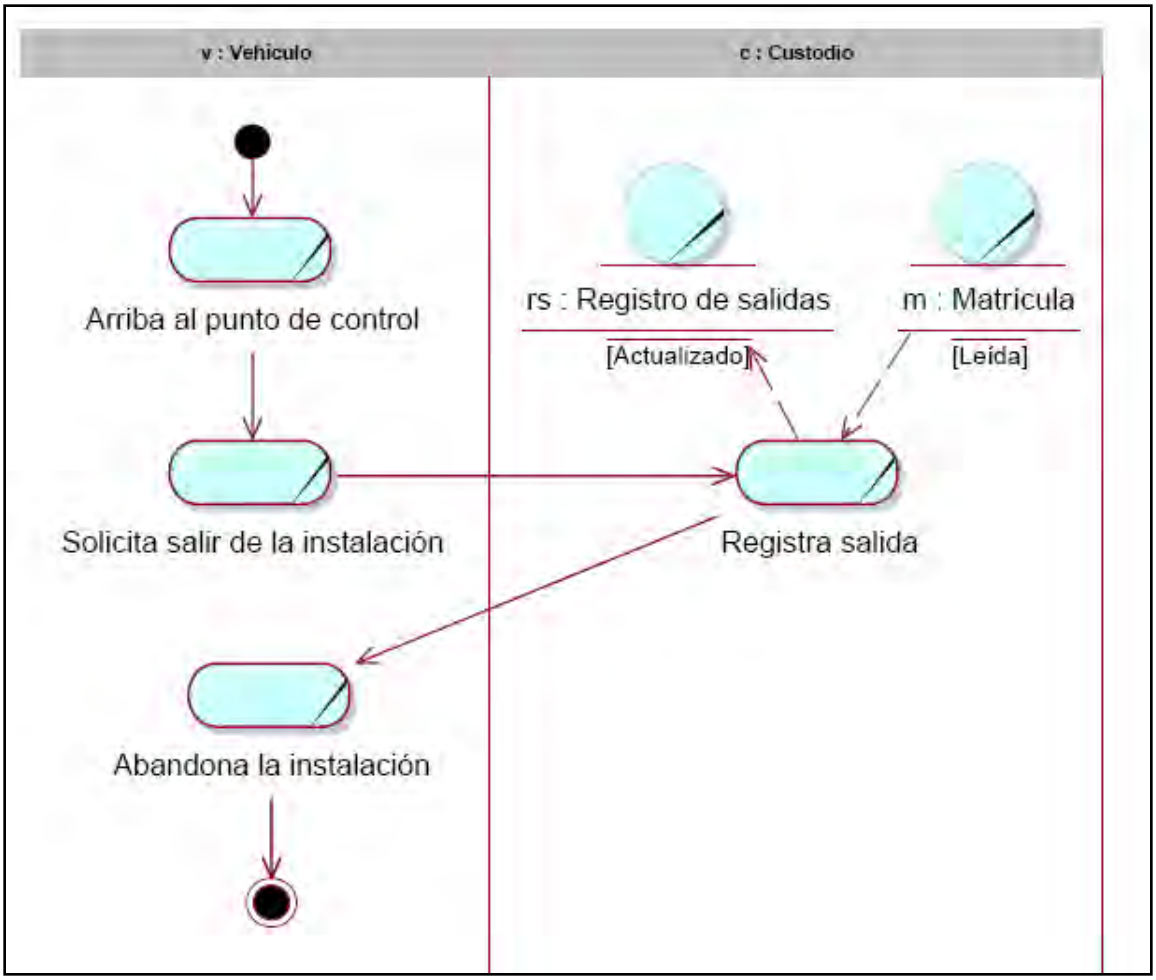


Diagrama A1.4**.** Diagrama de actividades correspondiente al caso de uso de negocio **Registrar salida**.

Anexo 2. Artefactos de requerimientos

En este anexo se muestra el diagrama de casos de uso del sistema propuesto. Además se describen los actores y los casos de uso, del sistema.

A2.1. Diagrama de casos de uso del sistema

En esta sección se muestra el diagrama de casos de uso del sistema propuesto (Diagrama A2.1).

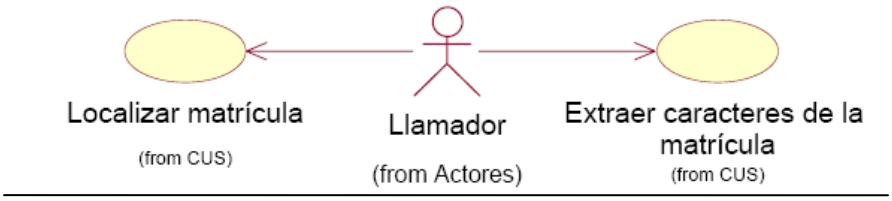


Diagrama A2.1. Diagrama de casos de uso del sistema.

A2.2. Actores y casos de uso del sistema

En esta sección se describen los actores (Tabla A2.1) y los casos de uso, del sistema propuesto (Tablas A2.2, A2.3,. A2.4 y A2.5).

Tabla A2.1. Actores del sistema.

Actor	**Justificación**
Llamador	Código o agente que invoca los tipos expuestos en la librería de tipos.

Tabla A2.2. Descripción del caso de uso **Localizar matrícula**.

CU1- Localizar matrícula	
Actores	Llamador
Descripción	El **Llamador** invoca el caso de uso para localizar la matrícula del vehículo. El sistema procesa y analiza los datos de entrada, utilizando técnicas de **Visión artificial**, y luego retorna el resultado.
Referencia	RF-1

Tabla A2.3. Descripción del caso de uso **Extraer caracteres de la matrícula**.

CU1- Extraer caracteres de la matrícula	
Actores	Llamador
Descripción	El **Llamador** invoca el caso de uso para recortar los caracteres de la matrícula. El sistema procesa y analiza los datos de entrada, utilizando técnicas de **Visión artificial**, y luego retorna el resultado.
Referencia	RF-2

Tabla A2.4**.** Descripción extendida del caso de uso **Localizar matrícula.**

CU-1	Localizar matrícula
Propósito	Localizar la matrícula en la imagen digital de un vehículo.
Actores	**Llamador**
Resumen	El **Llamador** invoca el caso de uso para localizar la matrícula del vehículo. El sistema procesa y analiza los datos de entrada, utilizando técnicas de **Visión artificial**, y luego retorna el resultado.
Referencias	RF-1
Tipo	Crítico
ESCENARIO 1	
FLUJO NORMAL DE ACCIONES	
Acción del actor	**Respuesta(s) del sistema**
1. El **Llamador** invoca al sistema, para localizar la matrícula a partir de la imagen digital de un vehículo.	1.1 Selecciona las posibles regiones de matrícula, siguiendo algún criterio. 1.2 Ordena las regiones seleccionadas, siguiendo algún criterio. 1.3 Construye un nuevo **Registro de Localización.** 1.4 Localiza la matrícula, a partir del **Registro de localización** construido *(Acciones del sistema en el escenario 2).*
ESCENARIO 2	
FLUJO NORMAL DE ACCIONES	
Acción del actor	**Respuesta(s) del sistema**
1. El **Llamador** invoca al sistema, para localizar la matrícula a partir de un **Registro de localización**.	1.1 Selecciona la próxima región. 1.2 Verifica la región de acuerdo al descriptor ***"área"***. 1.3 Calcula la inclinación de la matrícula.

	1.4 Verifica la región de acuerdo al descriptor ***"longitud de la mayor línea horizontal"***. 1.5 Rota la región. 1.6. Recorta la región horizontalmente. 1.7 Recorta la región verticalmente. 1.8 Verifica la región, de acuerdo al descriptor ***"relación de aspecto"***. 1.9 Segmenta la región. 1.10 Verifica la región de acuerdo al descriptor ***"uniformidad de la proyección horizontal".*** 1.11 Retorna la región.
FLUJO ALTERNO DE ACCIONES	
Acción del actor	**Respuesta(s) del sistema**
	1.2.1 Si la región actual no es válida, de acuerdo al descriptor ***"área"***, el sistema continúa la ejecución con la próxima región. Si no hay más regiones, el sistema retorna **NULL** y termina el caso de uso.
	1.4.1 Si la región actual no es válida, de acuerdo al descriptor "***longitud de la mayor línea horizontal***", el sistema continúa la ejecución con la próxima región. Además, el sistema adiciona la región actual al ***Registro de localización***.
	1.8.1 Si la región actual no es válida, de acuerdo al descriptor ***"relación de aspecto"***, el sistema continúa la ejecución con la próxima región. Si no hay más regiones, el sistema retorna **NULL** y termina el caso de uso.
	1.10.1 Si la región actual no es válida, de acuerdo al descriptor ***"uniformidad de la proyección horizontal"***, el sistema continúa la ejecución con la próxima región.

	Si no hay más regiones, el sistema retorna **NULL** y termina el caso de uso.
Post-condiciones	El **Registro de localización** queda actualizado.

Tabla A2.5. Descripción extendida del caso de uso **Extraer caracteres de la matrícula".**

CU-2	Extraer caracteres de la matrícula
Propósito	Extraer las regiones que representan los caracteres de la matrícula.
Actores	**Llamador**
Resumen	El **Llamador** invoca el caso de uso para recortar los caracteres de la matrícula. El sistema procesa y analiza los datos de entrada, utilizando técnicas de **Visión artificial**, y luego retorna el resultado.
Referencias	RF-2
Tipo	Crítico
FLUJO NORMAL DE ACCIONES	
Acción del actor	**Respuesta(s) del sistema**
1. El **Llamador** invoca al sistema para extraer los caracteres de matrícula, contenidos en una imagen digital de entrada.	1.1 Aplica un filtro tipo *negativo*. 1.2 Detecta las componentes conectadas. 1.3 Valida la cantidad de componentes conectadas. 1.4 Recorta las regiones delimitadas por las componentes conectadas. 1.5 Retorna las imágenes recortadas.
FLUJO ALTERNO DE ACCIONES	
Acción del actor	**Respuesta(s) del sistema**
	1.3.1 Si la cantidad de componentes conectadas no es correcta: — Se calcula la proyección horizontal de la matriz binaria. — Se calculan los picos existentes en la proyección horizontal. — Se recortan las regiones de la imagen digital, que representan los caracteres de la matrícula.
Post-condiciones	

Anexo 3. Descripción textual de las clases de diseño

En este anexo se describen las clases de diseño que participan en la realización de los casos de uso *Localizar matrícula* y *Extraer caracteres de la matrícula*. Las mismas están en correspondencia con el lenguaje de programación C# 2.x, y han sido agrupadas en clases **entidad**, clases **control** y clases **interfaz**.

A3.1. Caso de uso Localizar matrícula

A3.1.1. Clases entidad

Nombre: Anpr.LpLocalization.**SkewInfo**	
Estereotipo	Struct
Tipo	Entidad
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_beams	Projection
_maxRhoIndex	Int
_maxThetaIndex	int
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
SkewInfo(Projection, int, int)	Inicializa una nueva instancia del tipo **SkewInfo**.
Beams {get}: Projection	Retorna los rayos de la Transformada Radon.
MaxRhoIndex {get}: int	Retorna la coordenada rho de la celda con valor máximo en la Transformada Radon.
ThetaIndex {get}: int	Retorna la coordenada theta de la celda con valor máximo en la Transformada Radon.

Nombre: Anpr.LpLocalization.**LpVBounds**	
Estereotipo	Struct
Tipo	Entidad
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_left	Int
_right	Int
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**

LpVBounds(int, int)	Inicializa una nueva instancia del tipo **LpVBounds**
Left {get;set}:int	Retorna o modifica el valor del índice izquierdo.
Right {get;set}:int	Retorna o modifica el valor del índice derecho.
IsCorrupt {get}: bool	Retorna un valor, indicando si el valor de los campos **Left** y **Right** es incorrecto.

Nombre: Anpr.LpLocalization.**LpHBounds**	
Estereotipo	Struct
Tipo	Entidad
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_up	Int
_down	Int
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpHBounds(int, int)	Inicializa una nueva instancia del tipo **LpHBounds**
Up {get;set}:int	Retorna o modifica el valor del índice superior.
Down {get;set}:int	Retorna o modifica el valor del índice inferior.
IsCorrupt {get}: bool	Retorna un valor, indicando si el valor de los campos **Up** y **Down** es incorrecto.

Nombre: Anpr.LpLocalization.**HighlightSettings**	
Estereotipo	Class
Tipo	Entidad
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_strelSize	Size
_noiseHeight	Int
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
HighlightSettings(Size, int)	Inicializa una nueva instancia del tipo **HighlightSettings**.
StrelSize {get}:Size	Retorna el tamaño del elemento estructurante, rectangular, utilizado durante las operaciones morfológicas.
NoiseHeight {get}:int	Retorna el valor de la mínima altura de un borde.

Nombre: Anpr.LpLocalization.**LplSettings**	
Estereotipo	Class
Tipo	Entidad
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_plateHeight	Int
_plateWidth	Int
_minAspectRatio	double
_maxAspectRatio	double
_maxSkew	Int
_charsPerPlate	Int
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LplSettings(int, int, double, double, int, int)	Inicializa una nueva instancia del tipo **LplSettings**.
MinAspectRatio {get}: double	Retorna el valor de la mínima relación de aspecto.
MaxAspectRatio {get}:double	Retorna el valor de la máxima relación de aspecto.
MaxSkew {get}:int	Retorna el valor de la máxima inclinación permitida en las matrículas.
CharsPerPlate {get}:int	Retorna la cantidad de caracteres presentes en una matrícula.
MinEdgeHeight {get}:int	Retorna el valor de la mínima altura de un borde.
MinLineLength {get}:int	Retorna la mínima longitud de la línea horizontal presente en las matrículas.
MinArea {get}:int	Retorna el valor de la mínima área de una región.
StrelSize {get}: Size	Retorna el tamaño del elemento estructurante, rectangular, utilizado durante las operaciones morfológicas.

Nombre: Anpr.LpLocalization.**LplRecord**	
Estereotipo	Class
Tipo	Entidad
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_source	GrayMatrix
_regions	List<ConnectedComponent>
_currentIndex	Int
_suspiciousLps	PrioritizeableQueue<SuspiciousLpInfo>
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LplRecord(GrayMatrix, List<ConnectedComponent>)	Inicializa una nueva instancia del tipo **LplRecord**.

Source {get}: GrayMatrix	Retorna la matriz, a partir de la cual se inició el proceso de localización de la matrícula.
NextRegion {get}: ConnectedComponent	Retorna el próximo elemento, de la cola de componentes conexas por analizar.
NextSuspiciousLp {get}: SuspiciousLpInfo	Retorna el próximo elemento, de la cola de regiones por analizar.
AddSuspiciousLp(SuspiciousLpInfo):void	Adiciona un nuevo elemento, a la cola de regiones por analizar.

Nombre: Anpr.LpLocalization.**LplResult**	
Estereotipo	Class
Tipo	Entidad
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_licensePlate	LogicalMatrix
_record	LplRecord
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LplResult(LogicalMatrix, LplRecord)	Inicializa una nueva instancia del tipo **LplResult**.
LicensePlate {get}: LogicalMatrix	Retorna la matriz que representa la imagen digital de la matrícula.
Record {get}: LplRecord	Retorna el registro de las acciones efectuadas durante el proceso de localización de la matrícula.
IsDisposed {get}: bool	Retorna un valor, indicando si los recursos de la instancia actual han sido liberados.
Dispose():void	Libera los recursos de la instancia actual.

Nombre: Anpr.LpLocalization.**SuspiciousLpInfo**	
Estereotipo	Class
Tipo	Entidad
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_region	GrayMatrix
_skewInfo	SkewInfo
_lineLength	Int
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
SuspiciousLpInfo(region:GrayMatrix, skewInfo:SkewInfo, lineLength:int)	Inicializa una nueva instancia del tipo **SuspiciousLpInfo**.
Region {get}: GrayMatrix	Retorna la matriz que representa la región.
SkewInfo {get}: SkewInfo	Retorna información sobre la inclinación detectada en la región.
LineLength {get}: int	Retorna la longitud de la mayor línea horizontal existente en la región.

A3.1.2. Clases control

Nombre: Anpr.LpLocalization.**LpI**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_settings	LpISettings
_skewAngles	int[]
_lpHighLighter	LpHighlighter
_skewDetector	SkewDetector
_signatureValidator	LpSignatureValidator
_hBoundsCalculators	LpHBoundsCalculator[]
_vBoundsCalculator	LpVBoundsCalculator
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpI(LpISettings)	Inicializa una nueva instancia del tipo **LpI**.
LocalizeLp(GrayMatrix):LpIResult	Localiza la matrícula de un vehículo, a partir de una matriz.
LocalizeLp(LpIRecord):LpIResul	Localiza la matrícula de un vehículo, a partir de un registro de acciones

t	de localización.

Nombre: Anpr.LpLocalization.**LplManager**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
Create(LplSettings):Lpl	Retorna una nueva instancia del tipo Lpl.
Create(int, int, double, double, int, int):Lpl	Retorna una nueva instancia del tipo Lpl.

Nombre: Anpr.LpLocalization.**LpHighlighter**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
_settings	HighlightSettings
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpHighlighter(HighlightSettings)	Inicializa una nueva instancia del tipo **LpHighlighter**.
Settings {get}: HighlightSettings	Retorna los parámetros de configuración de la instancia actual.
HighlightLp(GrayMatrix):Logical Matrix	Resalta las regiones de la matrícula.

Nombre: Anpr.LpLocalization.**SkewDetector**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
SkewDetector()	Inicializa una nueva instancia del tipo **SkewDetector**.

GetSkewInfo(GrayMatrix, int[]):SkewInfo	Detecta la inclinación de la matrícula.

Nombre: Anpr.LpLocalization.**LpHBoundsCalculator**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpHBoundsCalculator()	Inicializa una nueva instancia del tipo **LpHBoundsCalculator**.
Calculate(Projection):LphBounds	Calcula los límites horizontales de la matrícula.

Nombre: Anpr.LpLocalization.**LpHBoundsCalculatorHP**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpHBoundsCalculatorHP ()	Inicializa una nueva instancia del tipo **LpHBoundsCalculatorHP**.
Calculate(Projection):LphBounds	Calcula los límites horizontales de la matrícula, utilizando la proyección horizontal.

Nombre: Anpr.LpLocalization.**LpHBoundsCalculatorRT**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpHBoundsCalculatorRT()	Inicializa una nueva instancia del tipo **LpHBoundsCalculatorRT**.
Calculate(Projection):LphBounds	Calcula los límites horizontales de la matrícula, utilizando la proyección de la Transformada Radon.

Nombre: Anpr.LpLocalization.**LpVBoundsCalculator**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpVBoundsCalculator()	Inicializa una nueva instancia del tipo **LpVBoundsCalculator**.
Calculate(GrayMatrix):LpVBounds	Calcula los límites verticales de la matrícula.

Nombre: Anpr.LpLocalization.**LpSignatureValidator**	
Estereotipo	Class
Tipo	Control
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpSignatureValidator()	Inicializa una nueva instancia del tipo **LpSignatureValidator**.
Validate(Projection, int)	Verifica la uniformidad de la proyección horizontal.

A3.1.3. Clases interfaz

Nombre: Anpr.LpLocalization.**IPrioritizeable**	
Estereotipo	Interface
Tipo	Interfaz
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
Priority {get}:int	Retorna la prioridad de un elemento, contenido en una estructura tipo **FIFO** (*First in, First out*) condicionada por prioridad.

A3.2. Caso de uso Extraer caracteres de la matrícula

A3.2.1. Clases control

Nombre: Anpr.LpCharsClipping.**LpCharsClipper**	
Estereotipo	Control
Tipo	Class
LISTADO DE ATRIBUTOS	
Nombre	**Tipo**
LISTADO DE RESPONSABILIDADES	
Firma	**Descripción**
LpCharsClipper()	Inicializa una nueva instancia del tipo **LpCharsClipper**.
ClipLp(source:LogicalMatrix, charsPerPlate:int): List< LogicalMatrix >	Recorta los caracteres de matrícula, contenidos en la imagen digital de entrada.

Anexo 4. Artefactos de prueba

En este anexo se presentan los casos de prueba y los procedimientos de prueba, correspondientes a los casos de uso del sistema propuesto.

A4.1. Especificación de los casos de prueba

Tabla A4.1**.** Casos de prueba correspondientes al caso de uso **Localizar matrícula.**

Nombre	**Localizar matrícula a partir de matriz nula.**
Entrada	*Null*
Salida	Ocurre una excepción de tipo *System.ArgumentNullException.*
Condiciones	
Nombre	**Localizar matrícula a partir de registro nulo.**
Entrada	*Null*
Salida	Ocurre una excepción de tipo *System.ArgumentNullException.*
Condiciones	
Nombre	**Localizar matrícula a partir de la imagen de un vehículo.**
Entrada	Matriz de valores escalares.
Salida	Conjunto de matrices que representan posibles matrículas de vehículos.
Condiciones	La matriz de entrada contiene una o varias matrículas.

Tabla A4.2**.** Casos de prueba correspondientes al caso de uso **Extraer caracteres de la matrícula**.

Nombre	**Extraer caracteres a partir de matríz nula.**
Entrada	*Null*
Salida	Ocurre una excepción de tipo *System.ArgumentNullException.*
Condiciones	
Nombre	**Extraer caracteres a partir de la matrícula de un vehículo.**
Entrada	Matriz de valores lógicos.
Salida	Conjunto de matrices que representan posibles caracteres de la matrícula.
Condiciones	1. La matriz lógica de entrada se corresponde con la matrícula (segmentada) de un vehículo. 2. El color de los puntos de fondo de la matrícula es blanco.

A4.2. Especificación de los procedimientos de prueba

Tabla A4.3**.** Procedimientos de prueba correspondientes al caso de uso **Localizar matrícula**.

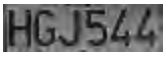
Nombre	**PP Localizar matrícula a partir de matriz nula.**
Caso de prueba asociado	Localizar matrícula a partir de matriz nula.
Acciones	
1. Construya una instancia del tipo *Anpr.LpLocalization.Lpl*. Para ello, invoque el método estático *Create* del tipo *Anpr.LpLocalization.LplManager.* 2. Con la instancia creada: 2.1. Invoque el método *LocalizeLp(Mtk.Imaging.GrayMatrix)*, suministrando como argumento *Null*. 2.2. Verifique la ocurrencia de una excepción de tipo *System.ArgumentNullException.*	
Nombre	**PP Localizar matrícula a partir de registro nulo.**
Caso de prueba asociado	Localizar matrícula a partir de registro nulo.
Acciones	
1. Construya una instancia del tipo *Anpr.LpLocalization.Lpl*. Para ello, invoque el método estático *Create* del tipo *Anpr.LpLocalization.LplManager.* 2. Con la instancia construida: 2.1. Invoque el método *LocalizeLp(Anpr.LpLocalization.LplRecord)*, suministrando como argumento *Null*. 2.2. Verifique la ocurrencia de una excepción de tipo *System.ArgumentNullException.*	
Nombre	**PP Localizar matrícula a partir de la imagen de un vehículo.**
Caso de prueba asociado	Localizar matrícula a partir de la imagen de un vehículo.
Acciones	
1. Construya una instancia del tipo *Anpr.LpLocalization.Lpl*. Para ello, invoque el método estático *Create* del tipo *Anpr.LpLocalization.LplManager.* 2. Con la instancia construida: 2.1. Invoque el método *LocalizeLp(Mtk.GrayMatrix),* suministrando como argumento la imagen digital de un vehículo. 2.2. Verifique que el valor de la propiedad *LicensePlate* (del resultado) se corresponde con la matrícula. 2.3. Invoque el método *LocalizeLp(Anpr.LpLocalization.LplRecord)*, suministrando como argumento el valor de la propiedad *Record* de la instancia retornada en el paso **2.1**. 2.4. Verifique que el valor de la propiedad *LicensePlate* (del resultado) se corresponde con la matrícula. 2.5. Si el valor de la propiedad *LicensePlate* (del resultado) es distinto de **Null**, **Regresar** al paso **2.3**; de lo contrario **Terminar**.	

Tabla A4.4**.** Procedimientos de prueba correspondientes al caso de uso **Extraer caracteres de la matrícula**.

Nombre	**PP Extraer caracteres a partir de matríz nula.**
Caso de prueba asociado	Extraer caracteres a partir de matríz nula.
Acciones	
1. Construya una instancia del tipo *Anpr.LpCharsClipping.LpCharsClipper*. Para ello invoque el método constructor de dicha clase. 2. Con la instancia creada: 2.1. Invoque el método *ClipLp(Mtk.LogicalMatrix),* suministrando como argumento *Null.* 2.2. Verifique la ocurrencia de una excepción de tipo *System.ArgumentNullException.*	
Nombre	**PP Extraer caracteres a partir de la matrícula de un vehículo.**
Caso de prueba asociado	Extraer caracteres a partir de la matrícula de un vehículo.
Acciones	
1. Construya una instancia del tipo *Anpr.LpCharsClipping.LpCharsClipper*. Para ello invoque el método constructor de dicha clase. 2. Con la instancia construida: 2.1. Invoque el método *ClipLp(Mtk.LogicalMatrix)*, suministrando como argumento la imagen digital de la matrícula de un vehículo. 2.2. Verifique que el resultado se corresponde con los caracteres de la matrícula.	

Anexo 5. Resultados del algoritmo de localización de matrículas

En este anexo se muestra un subconjunto de los resultados generados por el algoritmo de localización de matrículas, propuesto en esta tesis.



HGC701

HGC701

HSH014

HSH014


T 17520


HDG551


CUBA
HDG551


IVECO
Daily
HVY026


GDD064


GDD064


GDD064

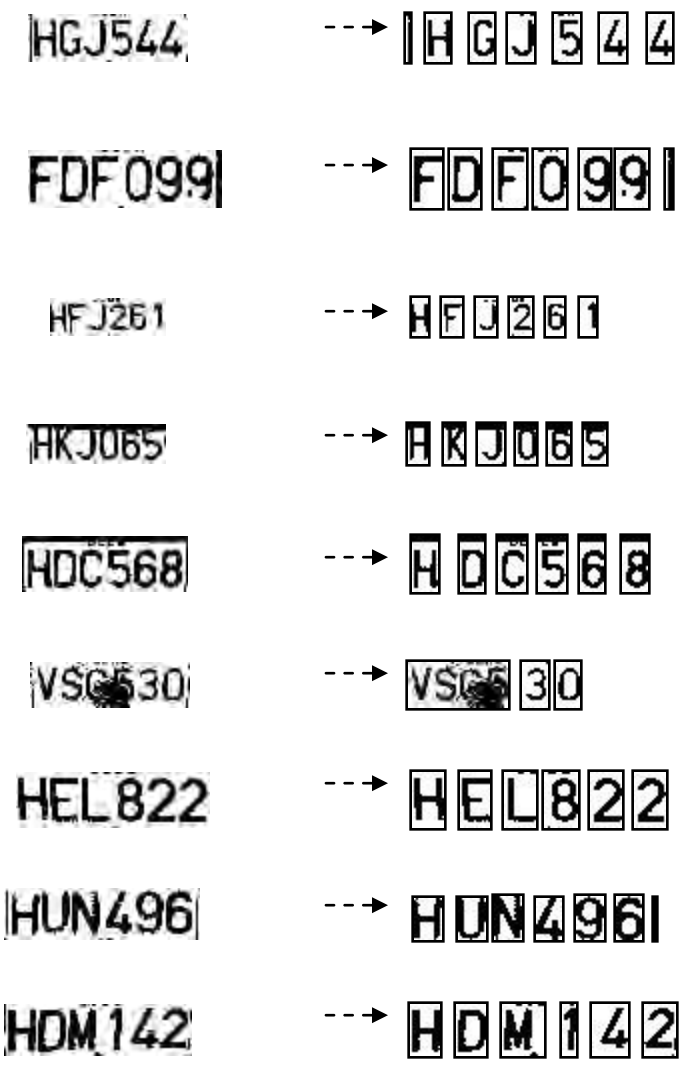

HDG275


HDG275


HDG275

Anexo 6. Resultados del algoritmo de segmentación de matrículas

En este anexo se muestra un subconjunto de los resultados generados por el algoritmo de segmentación de matrículas, propuesto en esta tesis.



GLOSARIO DE TÉRMINOS

En esta sección, se expone el significado de los términos relacionados con la temática abordada en esta tesis (Tabla 2).

Tabla 2. Glosario de términos.

Término	Significado
ANPR	ANPR (Automatic number plate recognition) es un método de vigilancia que utiliza técnicas de reconocimiento de patrones en imágenes digitales, para reconocer la matrícula de los vehículos (Wikipedia 2008).
CASE	Ingeniería de software asistida por computadora, por sus siglas en idioma Inglés.
Clase	Estructura de datos empleada para crear objetos (Arora, Aiaswamy y Pandey 2002).
Detección de discontinuidades	La detección de discontinuidades es, esencialmente, una operación de detección de los cambios locales significativos en la intensidad de la imagen (Molina 1998).
Dilatación	La dilatación es la transformación morfológica que combina dos vectores utilizando la suma. Si A y B son conjuntos de un n-espacio En, con elementos a= (a1,…, an) y b= (b1,…, bn), respectivamente, siendo ambos n-uplas, entonces la dilatación de A por B es el conjunto de todos los posibles vectores que son suma de pares de elementos, uno de A y otro de B (Molina 1998).
Ensamblado	Estructura lógica que contiene código compilado para la arquitectura .NET (Arora, Aiaswamy y Pandey 2002).
Erosión	La erosión es la transformación morfológica que combina dos conjuntos usando el concepto de inclusión. Si A y B son conjuntos en el espacio euclídeo n-dimensional, entonces la erosión de A por B es el conjunto de todos los elementos x para los que $x + b \in A$ para todo $b \in B$ (Molina 1998).
Especificidad	Capacidad de un sistema para detectar el no cumplimiento de una condición (Resonance Pub 2008).
Falso negativo	Suceso que indica el no cumplimiento de una condición en un individuo, cuando el individuo la cumple (Resonance Pub 2008).

Falso positivo	Suceso que indica el cumplimiento de una condición en un individuo, cuando el individuo no la cumple (Resonance Pub 2008).
GDI	Librería de tipos, utilizada en el sistema operativo Windows, que interactúa con los dispositivos gráficos en beneficio de las aplicaciones (MSDN 2007).
GDI+	Librería de tipos utilizada en el sistema operativo Windows para interactuar con los dispositivos gráficos. GDI+ es resultado de la perfección de GDI. (MSDN 2007)
Índice de falsos negativos	Probabilidad de ocurrencia de falsos negativos (Resonance Pub 2008).
Índice de falsos positivos	Probabilidad de ocurrencia de falsos positivos (Resonance Pub 2008).
Matrícula	Permutación de caracteres alfanuméricos que identifica e individualiza el vehículo respecto a los demás. Se representa en una placa metálica (que se conoce como placa de automóvil) en la que se graban o adhieren de forma inalterable los caracteres (Wikipedia 2008).
Modelo de color	Sub-espacio de un modelo de coordenadas, donde cada color se representa por un punto único (Molina 1998).
Mono	Mono es la plataforma de desarrollo de software libre basada en .NET que permite a los desarrolladores de software construir aplicaciones GNU/Linux y multiplataforma con una productividad sin precedentes (Icaza 2004).
Requisito funcional	Capacidad o condición que el sistema debe cumplir (Jacobson, Booch y Rumbaugh 2000).
Requisito no funcional	Propiedad o cualidad que el sistema debe tener (Jacobson, Booch y Rumbaugh 2000).
RUP	El Proceso Unificado de Desarrollo (RUP) es un proceso de desarrollo de software y un marco de trabajo genérico, que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organización, diferentes niveles de aptitud y diferentes tamaños de proyecto (Jacobson, Booch y Rumbaugh 2000).

Segmentación	Técnica que divide la imagen digital en un conjunto de regiones R, donde los pixeles contenidos en la región Ri comparten atributos similares (Gonzalez y Woods 2002).
Sensibilidad	Capacidad de un sistema para detectar el cumplimiento de una condición (Resonance Pub 2008).
Trabajador	Puesto que debe ser asignado a una persona o equipo, y que requiere responsabilidades y habilidades como realizar actividades o desarrollar determinados artefactos (Jacobson, Booch y Rumbaugh 2000).
Transformada Hough	La Transformada Hough (HT) es una técnica que permite localizar objetos de cierta forma en la imagen digital (HPIR s.f.)
UML	El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema software (Schmuller 2000).
Verdadero negativo	Suceso que indica el no cumplimiento de una condición en un individuo, cuando el individuo no la cumple (Resonance Pub 2008).
Verdadero positivo	Suceso que indica el cumplimiento de una condición en un individuo, cuando el individuo la cumple (Resonance Pub 2008).
Visión artificial	La Visión artificial o comprensión de imágenes describe la deducción automática de la estructura y propiedades de un mundo tridimensional, posiblemente dinámico, bien a partir de una o varias imágenes bidimensionales de este mundo (V.S. 1993).