PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

RECONOCIMIENTO FOTOGRÁFICO DE PATENTES

BASTIÁN NICOLÁS CARVAJAL AHUMADA

Profesor guía: Rafael Mellado Silva
Profesor co-referente: Dr. Claudio Cubillos Figueroa

MEMORIA PARA OPTAR
AL TÍTULO PROFESIONAL DE
INGENIERO DE EJECUCIÓN EN INFORMÁTICA

Marzo, 2018

Dedicatoria

Dedico este proyecto a mi mujer Natalia, quien es una de las personas más importantes junto al regalo más hermoso que me ha podido dar, mi hija Trinidad, dos personas que me dan la fuerza de seguir adelante siempre, que han estado incondicionalmente conmigo apoyando y formando una vida juntos. A mis padres y hermanos que me formaron como el hombre que hoy soy, agradecer todo su esfuerzo por salir adelante e inculcarme todos sus valores, quienes siempre han estado a mi lado cuidándome y enseñándome. Gracias por darme todo el apoyo y las ganas de terminar lo que empecé.

Bastián Nicolás Carvajal Ahumada.

Resumen

El reconocimiento óptico de patentes a través de imágenes digitales, es un algoritmo OCR implementado, capaz de reconocer la patente del vehículo a partir de una imagen. La implementación de este algoritmo, está bajo una herramienta llamada Openalpr, la cual es capaz de reconocer los caracteres de las patentes, dando un abanico de configuraciones que pueden ser útiles en este proyecto. Esta herramienta da la posibilidad de poder integrarla en nuestro sitio web, implementado con Nodejs, el cual nos brinda la posibilidad de levantar un servidor e interactuar con el Back-End del sitio, por medio del lenguaje de programación Javascript, generando la integración y comunicación de Openalpr. La implementación de esta herramienta, nos permite extraer los caracteres de manera más efectiva, con el debido entrenamiento y configuración de las placas del país a trabajar. El resultado del análisis, son los caracteres de la patente sugerida, mostrada en una tabla, con el nivel de confiabilidad en porcentaje y patentes candidatas sugeridas.

Palabras claves: Openalpr, patente, algoritmos, herramienta, Back-End, programación, sitio web, PPU.

Abstract

OCR patent through digital imaging, OCR algorithm is implemented, able to recognize the patent of the vehicle from an image. The implementation of this algorithm, is under a tool called Openalpr which is able to recognize the characters of patents, giving a range of configurations that can be useful in this project. This tool gives the possibility to integrate it into our web site, implemented with nodejs, which gives us the ability to lift a server and interact with the back end of the site, through the JavaScript programming language, generating integration and communication of Openalpr. The implementation of this tool allows us to extract the characters more effectively, with proper training and configuration of the plates in the country to work. The result of the analysis, are the characters of the suggested patent, shown in a table, with the confidence level percentage and patents candidates suggested.

Keywords: Openalpr, patent, algorithms, tools, Back-End, programming, website, PPU.

Índice

Lista de Figuras

Lista de Tablas

1. Definiciones, siglas y abreviaturas

En la siguiente tabla se aprecian las definiciones y siglas a utilizar en el presente informe.

PPU	Placa patente única de vehículos
OCR	Reconocimiento Óptico de Caracteres

Tabla 1.1: Siglas y abreviaturas.

En la siguiente tabla, se definirán conceptos o palabras, que serán utilizados recurrentemente en este informe.

Patrón	Refiere a sucesos u objetos recurrentes en la imagen.
Algoritmo	Conjunto de operaciones que permite hallar una solución.
Prototipo	Un Prototipo es un ejemplar o primer molde en que se fabrica una figura u otra cosa.
Software o sistema	Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.
Herramienta	Son programas, aplicaciones o simplemente instrucciones usadas para efectuar otras tareas de modo más sencillo.

Tabla 1.2: Conceptos o palabras

2. Presentación del tema

A continuación,se da inicio y explica a fondo el Reconocimiento Fotográfico de Patentes Chilenas.

2.1. Introducción

El presente trabajo es un estudio sobre la aplicación práctica de técnicas de aprendizaje profundo en el desarrollo de un sistema para el reconocimiento automático de patentes chilenas de vehículos. Estos sistemas son denominados generalmente ALPR (Automatic License Plate Recognition) que son capaces de reconocer el contenido de las patentes de los vehículos a partir de las imágenes capturadas por una cámara fotográfica. El sistema propuesto en este trabajo se basa en un clasificador de imágenes desarrollado mediante técnicas de aprendizaje supervisado con redes neuronales artificiales llamado Openalpr. Estas redes son una de las arquitecturas de aprendizaje profundo más populares, y están diseñadas específicamente para resolver problemas de visión artificial, como el reconocimiento de patrones y la clasificación de imágenes.

2.1.1. Antecedentes de la Industria (Chile)

Actualmente en la industria ANPR de Chile, existe una gama importante de software que cumplen con las expectativas de rendimiento al reconocimiento de una patente nacional. Muchos de estos software son vendidos con el fin de implementar seguridad en el transito vehicular tomando un valor monetario considerable respecto a las especificaciones técnicas de cada vendedor. Nuestra prueba muestra que podemos obtener resultados similares con un software opensource a bajo costo con especificaciones y prestancias similares o de iguales características. Respecto a esto Carabineros de Chile acompañado del Gobierno de turno en el año 2012 lanzan un sistema de vigilancia que opera en autopistas con telepeaje, con el fin de detectar autos con encargo por robos. Gracias a estas medidas, desde agosto del año 2011 los robos mensuales de vehículos han disminuido, marcando un quiebre en la tendencia que exhibía hasta ese momento, y logrando que los delitos mensuales en 2012 ya estén bajo el promedio del año pasado. Esto quiere decir que en lo que va del 2012 el robo ha disminuido en un 5,4 % con respecto al 2011. [6]

2.1.2. Formato PPU vigente desde el 2007

En septiembre de 2007 se comenzó a utilizar el nuevo formato, que se conforma por 4 letras y 2 números (BB-BB·10). Este sistema utiliza 18 letras,1 que son: B, C, D, F, G, H, J, K, L, P, R, S, T, V, W, X, Y, Z [23]. No se emplean las letras M, N (las

2 anteriores de seguro para no tener combinaciones que sean CTM o PN), Ñ, Q (por su parecido con la O y el número 0), y las vocales (para evitar combinaciones que forman palabras y así incomodar a los conductores de los vehículos) [23]. Si bien las placas de vehículos nuevos usan el formato BB·BB-10, las placas anteriores no fueron reemplazadas, por lo que actualmente coexisten los dos formatos 2.1. En la siguiente figura se muestran algunos tipos de PPU utilizadas en Chile, cabe destacar que este informe se centra en la PPU de vehículo particular.

Fig. 2.1: Tipos de PPU. [23]

2.1.3. Problema

El problema se presenta en la capacidad de reconocer una patente chilena por medio de técnicas OCR en tiempo real de manera exacta. Generalmente la capacidad de reconocimiento es baja frente al estado de la imagen que se trata de analizar.

2.1.4. Justificación de la necesidad

El sistema web de reconocimiento de patentes que se plantea busca facilitar la extracción de la información de las patentes vehiculares dentro del territorio nacional, ya que para lograr su reconocimiento es necesario implementar una herramienta capaz de realizar este objetivo y ser lo suficientemente efectivo a la hora del reconocimiento. Se busca una alta tasa de reconocimiento y brindar una respuesta rápida.

2.2. Objetivos

En este apartado se presentan los objetivos generales y secundarios de esta primera etapa del proyecto.

2.2.1. Objetivo general

Se establece como objetivo general del proyecto la implementación de Openalpr para la creación de un prototipo web, el cual permita desarrollar un prototipo web para el reconocimiento de los caracteres de una patente utilizando motor OCR, Tesseract, previamente configurado y entrenado para las patentes de Chile. De esta manera se permite la automatización de los procesos de extracción de los caracteres de una patente chilena.

2.2.2. Objetivos específicos

En este apartado se mencionan los objetivos específicos o secundarios del proyecto.

- Análisis de herramienta Openalpr e instalación.
- Levantar servidor web con lenguaje NodeJs.
- Generar sitio web, con integración de Openalpr.
- Establecer tabla con el reconocimiento de las patentes analizadas.
- Integrar configuración para el reconocimiento de patentes chilenas.
- Generar una tendencia, con niveles de entrenamiento del Openalpr.

2.3. Criterios de éxito del proyecto

El sistema tiene que abarcar todos los requerimientos presentados. Debe ser capaz de reconocer la patente en una imagen digitalizada, luego reconocer los caracteres para procesar y extraer la información. Finalmente ser mostrados en una tabla con posibles candidatas semejantes a la patente y confianza en porcentaje (1 - 100) [21].

2.4. Propuesta de Solución

Se desarrolla un software que permite el reconocimiento óptico de los caracteres contenidos en un documento escaneado o fotografía, tratando de automatizar la extracción de esta información. La solicitud de un software que sea capaz de extraer la información de una patente de vehículo, a través de una imagen, ha llegado a nuestra investigación de algoritmos OCR, quienes son capaces de reconocer los patrones de una imagen digital [19]. Las patentes de vehículos son conocidas actualmente en Chile como Placa Patente Única (PPU)[23]. Esta denominación se introduce para unificar los registros de patentes a nivel nacional, donde cada vehículo lleva su patente única en el país. De acuerdo a esto podemos obtener, a partir de sus siglas, información sobre el propietario y vehículo. Uno de los posibles usos de este algoritmo, es verificar por medio del OCR[2] sus caracteres y extraer la información. Openalpr es una herramienta que nos permite reconocer patentes de vehículos. Un entrenamiento y configuración de Openalpr, respectivo a la patente del país, en este caso Chile, es el paso más costoso e importante, ya que nos permite realizar los reconocimientos de la patente con un nivel de confianza alto. Esta herramienta permite integrarse en distintos lenguajes de programación, dada esta característica el entorno de programación multiplataforma NodeJs [14] permite comunicarnos directamente a través de Javascript en el Back-End del servicio web. Montaremos un sistema web con la plataforma Nodejs, quien nos da la capacidad de integrar de manera directa Openalpr por su lenguaje nativo en JavaScript. Este sistema será capaz de cargar una imagen del vehículo o patente a consultar, luego se pondrá en marcha Openalpr, el cual reconocerá los caracteres de la patente. Esta herramienta tendrá como motor fundamental el motor OCR llamado TESSERACT [11], quien finalmente con un entrenamiento y configuración previa de la patente, en este caso chilena, será reconocida mostrando consigo el nivel de confianza y los caracteres de la patente.

2.5. Planificación

La siguiente tabla Gantt nos muestra de manera gráfica la dedicación en tiempo previsto para la realización de las diferentes tareas encomendadas hasta la fecha, además de eso nos indica las relaciones entre distintas tareas como reuniones, investigación y envío de avances.

		Nombre	Duración	Inicio	Fin
1		**Inicio sistema de reconocimiento**	**81d?**	**01/07/2016**	**21/10/2016**
2		Instalación Nodejs	2d	01/07/2016	04/07/2016
3		Instalación Tesseract con Opencv (Compilación)	14d	05/07/2016	22/07/2016
4		Instalación Openalpr en sistema Linux 14.04	12d	01/08/2016	16/08/2016
5		**Desarrollo de sitio web con NodeJs**	**40d**	**29/08/2016**	**21/10/2016**
6		Integración de módulos para subir imagenes	10d	29/08/2016	09/09/2016
7		Integración Openalpr con reconocimiento standart	24d	20/09/2016	21/10/2016
8		Inicio de entrenamiento básico de prueba (no oficial)	1d?	21/09/2016	21/09/2016

Fig. 2.2: tabla gantt entrega 1.

		Nombre	Duración	Inicio	Fin
1			1d?	19/05/2016	19/05/2016
2			1d?	19/05/2016	19/05/2016
3		**Entrenamiento**	**65d**	**01/09/2016**	**30/11/2016**
4		Construcción de set de patentes	8d	21/09/2016	30/09/2016
5		Configuración de patente Openalpr	27d	01/09/2016	07/10/2016
6		Implementación de entrenamiento	13d	12/10/2016	28/10/2016
7		Realización de pruebas	5d	31/10/2016	04/11/2016
8		Corrección de configuración de entrenamiento	14d	01/11/2016	18/11/2016
9		Realización de pruebas exitosas	4d	25/11/2016	30/11/2016
10		Presentación de Código	1d	07/12/2016	07/12/2016
11		Presentación final de proyecto 2	1d	09/12/2016	09/12/2016

Fig. 2.3: tabla gantt entrega 2.

2.6. Métodos

La capa de métodos se centra en las actividades técnicas que se deben realizar para conseguir las tareas de ingeniería. Proporciona el "cómo" y cubre las actividades de ingeniería fundamentales. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Los métodos de la ingeniería del software dependen de un conjunto de principios básicos que gobiernan cada una de las áreas de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas. La construcción de software implica una amplia colección de actividades técnicas. La capa de métodos contiene los métodos definidos para realizar esas actividades de forma eficiente. Se centra en cómo se han de realizar las actividades técnicas. Los personas involucradas usan los métodos para realizar las actividades de ingeniería fundamentales necesarias para construir el software. Para varias actividades de proceso, la capa de métodos contiene el correspondiente conjunto de métodos técnicos para usar. Esto abarca un conjunto de reglas, los modos de representación gráficos o basados en texto, y las guías relacionadas para la evaluación de la calidad de la información representada [8].

2.6.1. Metodología Cascada

Es un enfoque metodológico que ordena rigurosamente las etapas del ciclo de vida del software, de forma que el inicio de cada etapa debe esperar a la finalización de la inmediatamente anterior. El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo se ve fluyendo hacia abajo (como una cascada) sobre las fases que componen el ciclo de vida. La primera descripción formal del modelo en cascada se cree que fue en un artículo publicado en 1970 por Winston W. Royce, aunque Royce no usó el término cascada en este artículo. Irónicamente, Royce estaba presentando este modelo como un ejemplo de modelo que no funcionaba, defectuoso. En el modelo original de Royce, existían las siguientes fases [8]:

- Especificación de requisitos.
- Diseño.
- Construcción (Implementación o codificación).
- Integración.
- Pruebas.
- Instalación.

- Mantenimiento.

Para seguir el modelo en cascada, se avanza de una fase a la siguiente en una forma puramente secuencial.

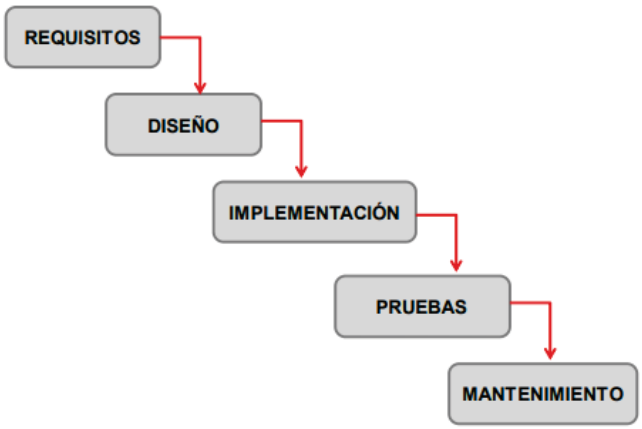

Fig. 2.4: Método Cascada [8].

2.6.2. Metodología Incremental

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software [8].

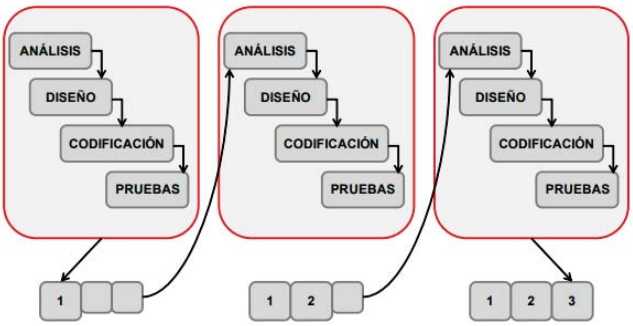

Fig. 2.5: Método Incremental [8].

Cuando se utiliza un modelo incremental, el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Este modelo se centra en la entrega de un producto operativo con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

2.6.3. Metodología Iterativa Incremental

La metodología que se utilizará para el desarrollo del software será la del modelo iterativo incremental, debido a que utiliza las ventajas del modelo en cascada y el modelo incremental. Este modelo se adapta de forma más eficiente para el desarrollo de este software en comparación con otros modelos ya que está basado en la filosofía de desarrollo en incrementos, en la que cada incremento proporciona un subconjunto de funcionalidades requeridas por el cliente.

3. Reconocimiento de patente

En este capítulo se expondrán todos los aspectos técnicos del software paso a paso para llegar a su ejecución.

3.1. Descripción del problema

El software del sistema se ejecuta sobre un hardware de PC estándar y puede ser enlazado a otras aplicaciones o bases de datos. Primero utiliza una serie de técnicas de manipulación de la imagen para detectar, normalizar y realzar la imagen del número de la matrícula [24], y finalmente reconocimiento óptico de caracteres para extraer la información de la patente. El entrenamiento de esta herramienta es fundamental en el reconocimiento, ya que un entrenamiento óptimo y configuración puede elevar la confiabilidad del resultado.

Hay seis algoritmos principales, cada uno representa una función matemática donde el software las necesita para identificar una patente [13]:

- 1- Localización de la patente, responsable de encontrar y aislar la patente en la imagen.
- 2- Orientación y tamaño de la patente, compensa los ángulos que hacen que la patente parezca torcida y ajusta las dimensiones al tamaño requerido.
- 3- Normalización, ajusta el brillo y el contraste de la imagen.
- 4- Segmentación de los caracteres, encuentra los distintos caracteres presentes en la patente.
- 5- Reconocimiento óptico de los caracteres.
- 6- Análisis sintáctico y geométrico, comprueba los caracteres encontrados y sus posiciones con las reglas específicas del país al que pertenece la patente.

Fig. 3.1: Fases del algoritmo [13].

La complejidad de cada una de estas subdivisiones del programa determina la exactitud del sistema. Durante la tercera fase (normalización) algunos sistemas utilizan técnicas de detección [13] de borde para aumentar la diferencia en la imagen entre las letras y el fondo de la placa. Existen varias aplicaciones donde puede utilizarse el reconocimiento de patentes y cada diferente aplicación puede tener diferentes sistemas en términos de implantación, hardware y tecnologías e incluso fabricantes de las mismas aplicaciones proveen sistemas de reconocimiento con funcionalidades similares pero estructuras totalmente diferentes [17].

3.1.1. Motores de OCR

Son los algoritmos correspondiente al OCR, los cuales funcionan de distintas formas y capacidades.

GOCR: Es el motor OCR por defecto, Convierte el texto en imágenes a archivos de texto Joerg Schulenburg inicio el programa y ahora lidera el equipo de desarrolladores. GOCR puede ser usado con diferentes front-end y back-end, con lo que hace muy fácil el portarlo a diferentes SOs y arquitecturas[5].

OCRAD: GNU Ocrad es un motor de Reconocimiento de Caracteres Ópticos(OCR) basado en un método de extracción de características. Ocrad lee una imagen en formato pbm (mapa de bits), pgm (escala de grises) o ppm (color), y produce texto en formato byte (8-bit) o UTF-8. También incluye un analizador de composición capaz de separar las columnas o bloques de texto que forman normalmente las páginas impresas. Ocrad puede ser usado como aplicación autónoma en modo texto, o como complemento (backend) de otros programas. Ocrad ha sido desarrollado por Antonio Diaz Diaz desde el 2003. La Versión 0.7 fue lanzada en febrero de 2004 2004, la 0.14 en febrero del 2006 y la 0.18 en mayo de 2009. [22]

TESSERACT: Es un motor de Reconocimiento Óptico de Caracteres (OCR) que está disponible para múltiples sistemas operativos. En éste motor de OCR se basan múltiples programas, que son los que realmente proporcionarán una interfaz al usuario para sacar partido a Tesseract-OCR. Por tanto, conviene tener claro que si sólo se instalara el motor Tesseract-OCR sólo se podría interactuar con él a base de instrucciones que se deberían escribir desde la línea de comandos de una consola.

El motor Tesseract se desarrolló en los laboratorios de Bristol de Hewlett Packard en Greeley (Colorado) entre 1985 y 1994. En 1996 se realizaron las modificaciones necesarias para su portabilidad en Windows, y más tarde, en 1998 se migró el sistema de C a C++.[11]

3.1.1.1. Comparación entre motores OCR

Comparación de varios motores OCR en sus tasas de reconocimiento y tiempo de espera. [1]

	Abbyocr	Cuneiform	Gocr	Ocrad	Tesseract
curier/black	txt 100 % (2.92s)	txt 61 % (1.11s)	txt 67 % (0.09s)	txt 21 % (0.02s)	txt 81 % (0.63s)
curier/grey	txt 100 % (2.85s)	X	txt 67 % (0.09s)	txt 21 % (0.03s)	txt 81 % (0.63s)
justy/black	txt 11 % (3.62s)	txt 3 % (1.14s)	txt 31 % (0.11s)	txt 1 % (0.02s)	txt 15 % (0.61s)
justy/grey	txt 14 % (3.45s)	X	txt 31 % (0.10s)	txt 1 % (0.02)	txt 15 % (0.60s)
times/black	txt 100 % (2.80s)	txt 96 % (1.07s)	txt 76 % (0.16s)	txt 82 % (0.03)	txt 92 % (0.74)
times/grey	txt 100 % (2.87s)	X	txt 76 % (0.16s)	txt 82 % (0.03)	txt 92 % (0.74)
verdana/black	txt 100 % (2.90s)	txt 95 % (1.07s)	txt 98 % (0.10s)	txt 98 % (0.03)	txt 98 % (0.45)
verdana/grey	txt 100 % (2.95s)	X	txt 98 % (0.10s)	txt 98 % (0.02)	txt 98 % (0.46)

Tabla 3.1: Comparación rendimiento motores OCR [1]

Los motores OCR son mas de una treintena, la mayoría de distribución libre, tienen además otras características que las diferencian unas de otras como los soft-

ware que requieren para su ejecución, los sistemas operativos compatibles, lenguajes en que este disponible el software, lenguajes de programación en que se implementan, etc.

	Founde year	ONLINE	Windows	Linux	Mac	Lenguaje de programación	SDK?	Lenguajes
Tesseract	1985	NO	SI	SI	SI	C++, C	SI	100+
Gocr	2000	SI	SI	SI	SI	C++	-	-
Ocrad	?	SI	SI	SI	SI	C	SI	Latin alphabet
Cuneiform	1996	NO	SI	SI	SI	C/C++	SI	28
Abbyocr	1989	SI	SI	SI	SI	C/C++	SI	198

Tabla 3.2: Propiedades y comparación de software [1]

3.2. Técnica Utilizada

A continuación, se explica en que consiste la técnica a utilizar LBP.

3.2.1. LBP

El operador LBP es un operador que describe el entorno de un píxel generando un código de bits a partir de las derivadas binarias de un píxel. El operador se aplica generalmente a imágenes en escala de grises y la derivada de las intensidades[12].

La formula utilizada esta dada por [18]:

$$H_i = \sum_{x,y} I\left\{f_l(x,y) = i\right\}, i = 0, \ldots, n-1 \tag{1}$$

Después de obtener la imagen fl (x, y) etiquetada como LBP, el histograma de LBP se puede definir como [18]:

$$N_i = \frac{H_i}{\sum_{j=0}^{n-1} H_j} \tag{2}$$

En su forma más simple, el operador LBP toma el 3 por 3 que rodea un píxel y genera un 1 binario, si el vecino del píxel central tiene un valor mayor que el píxel central y en el caso contrario el operador genera un 0 binario, si el vecino es menor que el centro [16]. Los ocho vecinos del centro pueden entonces representarse con un entero de 8 bits, por el cual lo convierte en una descripción muy compacta. La figura 3.2 muestra un ejemplo de un operador LBP. A menudo, la distribución de código LBP sobre una imagen se utiliza para describir la textura como un histograma de una imagen [10].

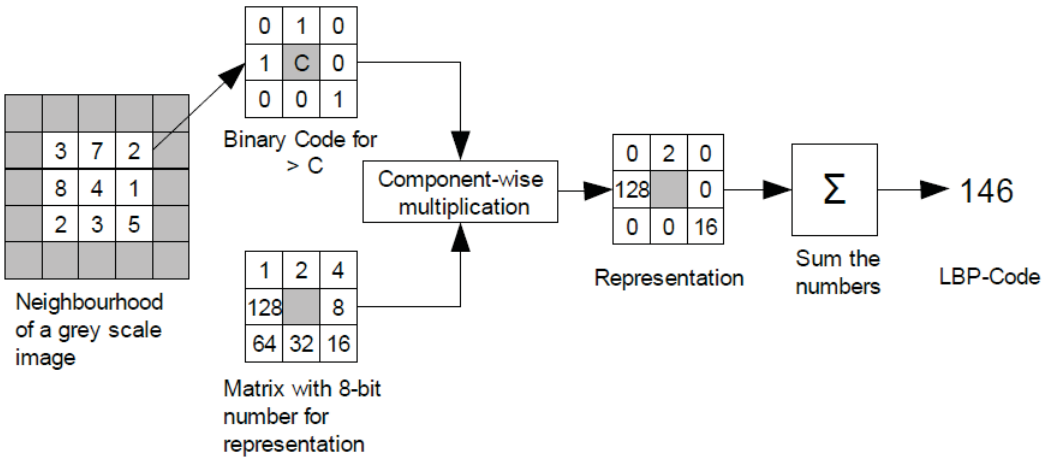

Fig. 3.2: Funcionamiento LBP [10].

El operador LBP se introduce por primera vez como una medida complementaria al contraste en el vecindario de un píxel llamado LBP / C [15]. El componente LBP se calcula como el que se muestra en la figura 3.2. A continuación se calcula el componente de contraste C como el promedio de los píxeles sobre el umbral menos el promedio de los píxeles bajo el umbral. En el caso mostrado anteriormente en la figura 3.2, el resultado sería:

$$(7+8+5)/3-(3+2+2+3+1)/5 \approx 4,47 \tag{3}$$

3.2.2. Derivación

Una derivación circular arbitraria para el operador LBP con cualquier radio y número de vecinos con el centro como umbral es dada por T. En esta sección se presenta una derivación basada en la de Ojala [15]. Se define la vecindad local de un píxel por la ecuación [10]:

$$T = t(g_c, g_0, ..., g_{p-1}) \tag{4}$$

- g_c: Valor de grises del píxel central.
- g_0 - g_{p-1}: Valor de P del número de vecinos grises.

donde g_c es el valor de grises del píxel central y g_0 - g_{p-1} corresponde al valor de P del número de vecinos gris. Las muestras tienen la misma distancia a su próxi-

mo vecino y se colocan a la distancia R desde el píxel central. Mientras que las coordenadas de los vecinos están dadas por la ecuación [10]:

$$[x_p, y_p] = [x_c + Rcos(\frac{2\pi p}{P}), y_c + Rsin(\frac{2\pi p}{P})] \tag{5}$$

- R: Distancia desde el píxel central.
- $[x_p, y_p]$: Coordenadas del píxel.
- P: Número de vecinos.

Las coordenadas a nivel de gris que se encuentran entre los valores de píxeles se interpolan, sugiriendo T como resultado. Ojala [15] sugiere la interpolación bilineal pero en muchas aplicaciones la interpolación más cercana del vecino es suficiente y también más rápida. El operador LBP en la sección 3.2 corresponde a la interpolación de vecinos más próximos. Por definición, la textura es el cambio de valores que corresponde matemáticamente a una derivada. Rescatando a los vecinos con el valor central R obteniendo la primera derivada discreta en cada dirección como se muestra en la siguiente ecuación [10]:

$$T = [\frac{g_0 - g_c}{R}, ..., \frac{g_{p-1} - g_c}{R}] \tag{6}$$

El valor g_c no contiene ninguna información sobre la textura, ya que sólo contiene el nivel de gris local de la región, por lo que la pérdida de g_c no significa ninguna pérdida de información de textura. Los datos del vecindario con valores mayor que cero describe un aumento y menos de cero una disminución. Esto entrega información de qué tipo es el píxel del centro. Toda otra información puede ser considerada como escalamiento de esta información. Para saber que datos son invariantes a esta información introducimos la función escalonada en la siguiente ecuación [10]:

$$s(x) = \begin{cases} 1, x \leq 1 \\ 0, x < 1 \end{cases} \tag{7}$$

Esta función es aplicada a cada valor y dado que el radio R también es un escalado y no tendrá ningún efecto sobre el resultado después de la función de escalón, también se puede ignorar dando como resultado la siguiente ecuación [10]:

$$T = t(S(g_0 - g_c), ..., S(g_{p-1} - g_c)) \tag{8}$$

Para asignar un número único a cada relación se puede asignar un peso binario a cada vecino. Esto da como resultado la siguiente ecuación que define un código LBP [10].

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} S(g_p - g_c)2^p \tag{9}$$

3.3. Openalpr

OpenALPR es un código abierto de reconocimiento automático de patentes, librería escrita en C ++ con fijaciones en C , Java, Node.js: Go, y Python, lo que brinda una integración relativamente fácil en el proyecto creado. La biblioteca analiza las imágenes y secuencias de vídeo para identificar las patentes. El resultado es una representación de texto de los caracteres de la patente.

3.3.1. Diseño de Openalpr

Openalpr funcciona como una tubería. La entrada es una imagen donde se le aplican varios procesamientos a esta las cuales se producen en etapas, y la salida son los posibles caracteres que contenga la patente contenida en la imagen ingresada.

3.3.2. Detección

La etapa de detección ocurre una vez para cada imagen de entrada. Utiliza el algoritmo LBP (generalmente usado para la detección de rostros)[3] para encontrar las posibles regiones de la patente. Cada una de estas regiones se envía a las siguientes fases de la tubería para su posterior procesamiento. Esta fase suele ser la más intensa en el procesamiento [9].

3.3.3. Binarización

Esta fase, y todas las siguientes fases ocurren varias veces para cada región posible detectada. La binarización crea múltiples imágenes binarias para cada región de la patente. La razón de utilizar varias imágenes binarias es para darnos la mejor oportunidad posible de encontrar todos los personajes. Una sola imagen binarizada puede perder caracteres si la imagen es demasiada oscura o demasiada clara [9] [4].

3.3.4. Análisis de caracteres

Esta fase de análisis de caracteres intenta encontrar los caracteres en la región de la patente. Para ello, primero encuentra todos los blobs (Objetos Binarios Grandes) conectados en la región de la patente. Entonces busca todas las figuras que son aproximadamente similar a los tamaños de los caracteres de la patente. Este análisis se realiza varias veces en la región. Comienza buscando los caracteres pequeños para luego buscar, gradualmente, los caracteres más grande. Si no se encuentra nada en la región, esta es expulsada y no se realiza ningún procesamiento adicional, de lo contrario se guarda para continuar con los siguientes procesamientos de la región [9] [4].

3.3.5. Bordes de la patente

En esta fase se encuentran los bordes de la patente. Se tiene que tener en cuenta que la ffase de detección sólo es responsable de identificar una posible región donde puede existir una patente. A menudo va a proporcionar una región que es un poco más grande o más pequeña que la dimensión real. En esta fase se trata de encontrar los bordes exactos de la patente ingresada [9].

3.3.6. Corrección

Dado los bordes de la patente, en esta etapa trata de corregir la región de la patente a un tamaño estándar. Idealmente, esto dará una imagen de la patente correctamente orientada, sin rotación o inclinación [9].

3.3.7. Segmentación de caracteres

En esta fase de segmentación de caracteres, se trata de aislar todos los caracteres que componen la imagen de la patente. Utiliza un histograma vertical para encontrar los espacios que existen entre los caracteres de la patente. En esta fase también se limpia las cajas de personajes mediante la eliminación de puntos pequeños y desconectados de la región de los caracteres. También intenta eliminar las regiones del borde de la pantete, para que no se clasifique de manera inapropiada como carácter [9] [4].

3.3.8. OCR

En la fase de OCR, se analiza cada carácter independientemente. Para cada imagen de carácter, se calcula todos los caracteres posibles y sus confidencias [9].

3.3.9. Post- Procesamiento

Dada una lista de todos los caracteres de OCR y confidencias posibles, el post-procesamiento determina las mejores combinaciones de letras de la patente. Está organizado como una lista N superior. El procesamiento posterior deshabilita todos los caracteres por debajo del umbral determinado [9].

3.4. Calculo de Confianza de la patente

Como se menciona anteriormente, la confianza de la patente irá de 0 - 100 %. En este proyecto se logra un umbral de corte del 85 %, pero el caso óptimo bordea el 90 % mostrando así los caracteres correctos de la patente analizada.

En el siguiente extracto de código que forma parte de este proceso, se puede apreciar la función evalúa los puntajes de los caracteres seleccionados y procesados en la etapa del OCR, donde se suman todos los puntajes y se divide por el número de los caracteres posibles. Con esta primer función se logra calcular la media.

Listing 1: Extracto de código para calcular media. [21]

```
float PostProcess::calculateMaxConfidenceScore()
 {
   // Take the best score for each char position and
      average it.

   float totalScore = 0;
   int numScores = 0;
   // Get a list of missing positions
   for (int i = 0; i < letters.size(); i++)
   {
     if (letters[i].size() > 0)
     {
       totalScore += (letters[i][0].totalscore / letters[i
          ][0].occurrences) + min_confidence;
       numScores++;
     }
   }

   if (numScores == 0)
     return 0;

   return totalScore / ((float) numScores);
 }
```

En el siguiente extracto de código, una vez obtenida la media, se ajustan los resultados existentes a porcentaje, lo que nos brinda la confianza de la patente.

Listing 2: Extracto de código para porcentaje de confianza. [21]

```
  // Now adjust the confidence scores to a percentage value
      float maxPercentScore = calculateMaxConfidenceScore()
         ;
```

```
3         float highestRelativeScore = (float) allPossibilities
              [0].totalscore;
4
5         for (int i = 0; i < allPossibilities.size(); i++)
6         {
7           allPossibilities[i].totalscore = maxPercentScore *
                (allPossibilities[i].totalscore /
                highestRelativeScore);
8         }
9       }
10
11      if (this->config->debugPostProcess)
12      {
13        // Print top words
14        for (int i = 0; i < allPossibilities.size(); i++)
15        {
16          cout << "Top " << topn << " Possibilities: " <<
                allPossibilities[i].letters << " :\t" <<
                allPossibilities[i].totalscore;
17          if (allPossibilities[i].letters == bestChars)
18            cout << " <--- ";
19          cout << endl;
20        }
21        cout << allPossibilities.size() << " total
              permutations" << endl;
22      }
```

A continuación se explica cada variable:

- totalScores: es la suma total de los puntajes asociados a las caracteres.
- numScores: es el número de los posibles caracteres.
- maxPercentScore: es la media resultante con la operación totalScores dividido numScores.
- highestRelativeScore: es el puntaje total de todos los posibles caracteres asociados a la patente.

3.5. Reconocimiento de patentes

Si bien existen variadas aplicaciones que brindan el mismo servicio, con distintos esquemas en la programación y desempeño, el reconocimiento de las patentes que se postula en este informe es gracias a la herramienta Openalpr, que funciona

principalmente con el motor OCR Tesseract, donde con un buen entrenamiento al motor OCR Tesseract y configuración de Openalpr puede generar los resultados esperados.

4. Requerimientos del proyecto

La ingeniería de requerimientos trata de establecer lo que el sistema debe hacer, sus propiedades emergentes deseadas y esenciales, y las restricciones en el funcionamiento del sistema y los procesos de desarrollo de software. Por lo tanto se debe considerar a la ingeniería de requerimientos como el proceso de comunicación entre los clientes, usuarios del software y los desarrolladores del mismo. “La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema”.

4.1. Requerimiento General

La finalidad del proyecto a realizar es un prototipo web en el cual se integre un software capaz de reconocer patentes chilenas por medio de una imagen digital, a través de una herramienta previamente configurada y entrenada.

4.2. Requerimientos funcionales

- Implementación de Openalpr para el reconocimiento de patentes.
- Ingreso de imágenes digitales de patentes.
- Entregar resultados en un formato de texto ligero Json.
- Mostrar resultados a partir del Json entregado para mostrarlos en una tabla de resultados.
- Mostrar la imagen ingresada al sistema en un recuadro cerca de la tabla de resultados.

4.3. Requerimientos no funcionales

- Diseño de una interfaz sencilla para el uso intuitivo del usuario principiante.
- Completa documentación del código.
- Rendimiento óptimo del sistema y sus funcionalidades.
- Conexión a Internet.

4.4. Casos de Uso

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el diagrama de casos de uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema. Los diagramas de casos de uso que se presentan a continuación tienen como finalidad modelar la interacción existente entre los distintos usuarios del sistema con éste, en conjunto con los actores involucrados para la ejecución de las funcionalidades diseñadas.

4.4.1. Caso de uso general

El siguiente diagrama nos muestra el caso de uso general del sistema y como el usuario se desenvuelve en él.

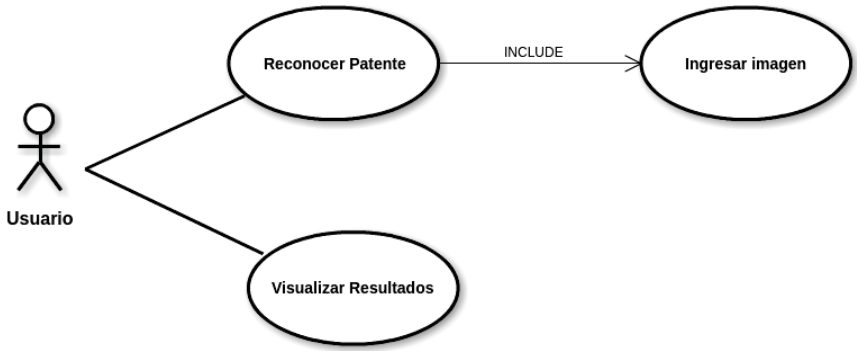

Fig. 4.1: Caso de uso general

4.4.2. Caso de uso narrativo

Las siguientes tablas muestran detalladamente cada uno de los casos de usos en su versión extendida narrativa, para una mayor comprensión del funcionamiento del sistema.

4.4.3. Reconocer patente

Caso de uso	Reconocer patente
Actores	Usuario
Propósito	Reconocer número de patente a partir de una fotografía.
Descripción	Un cliente ingresa a sistema de reconocimiento de matrículas de automóviles o página web e ingresa la imagen digital de la placa del automóvil. El sistema entrega los caracteres contenidos en la patente del auto.
Pre-Condición	
Post-Condición	
Curso normal de los eventos	
Acción actores	**Respuesta sistema**
1.- Este caso de uso comienza cuando cliente ingresa a sistema web y sube a dicho sistema una fotografía con la patente del automóvil(botón ingresar fotografía).	2.- Envía una interfaz con el resultado (caracteres de la patente reconocida por el sistema).
Curso alternativo de los eventos	
Acción actores	**Respuesta sistema**
1.- Este caso de uso comienza cuando cliente ingresa a sistema web y sube a dicho sistema una fotografía con la patente del automóvil(botón ingresar fotografía).	2.- El sistema le envía un mensaje al usuario informando incompatibilidad del formato de la imagen.
	3.- El sistema envía un mensaje al usuario con los formatos compatibles de imágenes en el sistema.

Tabla 4.1: Caso de uso narrativo/Reconocer patente

4.4.4. Desplegar Resultados

Caso de uso	Desplegar resultados
Actores	Usuario
Propósito	Desplegar resultados de una patente.
Descripción	Un usuario ingresa al sistema para ejecutar el reconocimiento de caracteres de la patente.
Pre-Condición	Haber cargado la, imagen digital de la patente.
Post-Condición	
Curso normal de los eventos	
Acción actores	**Respuesta sistema**
1.- Usuario ingresa a sistema web y selecciona la opción reconocer patente.	2.- Envía interfaz de reconocer patente.
3.- El usuario carga la imagen digital a analizar	4.- Procesa la patente.
en una tabla con la confianza y candidatas más cercanas.	5.- Retorna el resultado de los caracteres
Curso alternativo de los eventos	
Acción actores	**Respuesta sistema**
1.- Usuario ingresa a sistema web y selecciona opción reconocer patente".	2.- Envía interfaz de reconocer patente.
3.- El usuario carga la imagen digital de la patente.	4.- Procesa la patente.
	5.- Notifica al usuario que hubo un problema.

Tabla 4.2: Caso de uso narrativo/Desplegar resultados

5. Implementación de la Herramienta

A continuación, se describe detalladamente el trabajo realizado.

5.1. Desarrollo del Sistema

Para el desarrollo del sistema de reconocimiento de patentes, se ha estado estudiando los lenguajes de programación y herramientas necesarias para el reconocimiento de esta. A continuación se mostrará una guía de pasos con los diferentes elementos necesarios para el desarrollo del sistema de reconocimiento de patentes. Cabe recordar que este sistema se ha estado trabajando bajo la arquitectura de software Linux 14.04, pero se puede trabajar en otras plataformas de sistemas operativos, cada elemento que se menciona tiene sus dependencias para otros tipos de sistemas.

5.2. NodeJs

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor (pero no limitándose a ello) basado en el lenguaje de programación ECMAScript, asíncrono, con I/O de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web. Fue creado por Ryan Dahl en 2009.

5.2.1. Instalación

Para la instalación de Nodejs, en Linux, abrimos la terminal y copiamos el siguiente comando con tenido en la siguiente figura:

Listing 3: Instalación de NodeJs.

```
1  curl -sL https://deb.nodesource.com/setup_4.x | sudo -E
       bash -
2  sudo apt-get install -y nodejs
```

De manera opcional y más bien recomendable es instalar las herramientas de compilación para poder hacer uso de extensiones nativas desde NPM, el cual este último es un manejador de paquetes de datos que viene por defecto integrado en la instalación de NodeJs.

Listing 4: Comando de instalación herramienta de compilación.

```
sudo apt-get install -y build-essential
```

La instalación de Nodejs nos permite encontrar variadas herramientas para el desarrollo de un sistema web, módulos que podemos utilizar y sacar un buen rendimiento al comportamiento de nuestro sitio web a construir para el reconocimiento de patentes chilenas.

5.3. Instalación y configuración de Openalpr

Las dependencias de Openalpr nos muestra una guía completa de como instalar e integrar esta herramienta en nuestro sistema web [20].

5.3.1. Requerimientos para el funcionamiento de Openalpr

Openalpr requiere la integración de las librerías de Tesseract y Opencv para su funcionamiento, en la siguiente figura se muestra el repositorio de estas librerías:

Listing 5: Repositorio Tesseract y Opencv.[20]

```
 - Tesseract OCR v3.0.4 (https://github.com/tesseract-ocr/
    tesseract)
- OpenCV v2.4.8+ (http://opencv.org/)
```

Después de clonar este repositorio GitHub, usted debe descargar y extraer Tesseract y OpenCV código fuente en sus propios directorios. Compilar las dos bibliotecas [20].

5.3.2. Instalación de Openalpr

La instalación es relativamente sencilla, en la siguiente figura se muestra el comando a ejecutar en nuestra terminal:

Listing 6: Líneas de comandos instalación Openalpr. [20]

```
sudo apt-get update && sudo apt-get install -y openalpr
    openalpr-daemon openalpr-utils libopenalpr-dev
```

Luego para comprobar el funcionamiento se sugiere escribir el siguiente comando, dando como resultado los caracteres reconocidos y posibles patentes que se mostraran en nuestra terminal:

Listing 7: Líneas de comandos comprobación de instalación Openalpr. [20]

```

wget http://plates.openalpr.com/ea7the.jpg alpr -c us
    ea7the.jpg
```

La siguiente figura muestra el resultado por el terminal de Linux, donde muestra 10 resultados la confianza correspondiente y el tiempo que duro el procesamiento al analizar la patente. La salida de estos datos puede ser en formato Json, el cual podemos utilizar de mejor manera en el proyecto.

Listing 8: Datos por consola. [20]

```

user@linux:~/openalpr$ alpr ./samplecar.png

plate0: top 10 results -- Processing Time = 58.1879ms.
    - PE3R2X     confidence: 88.9371
    - PE32X      confidence: 78.1385
    - PE3R2      confidence: 77.5444
    - PE3R2Y     confidence: 76.1448
    - P63R2X     confidence: 72.9016
    - FE3R2X     confidence: 72.1147
    - PE32       confidence: 66.7458
    - PE32Y      confidence: 65.3462
    - P632X      confidence: 62.1031
    - P63R2      confidence: 61.5089
```

El resultado de Openalpr se basa en técnicas de visión por computadora. Hace uso del motor OCR Tesseract y las librerías de visión artificial OpenCV, donde esta

última contiene más de 500 funciones que abarcan una gran gama de áreas en el proceso de visión.

5.3.3. Utilización detallada de línea de comandos

En las siguientes figuras se muestra la lista de comandos con su especificación correspondiente para la utilización de Openalpr a la hora de analizar la patente.

Listing 9: Líneas de comandos. [20]

```
1
2  user@linux:~/openalpr$ alpr --help
3
4  USAGE:
5
6     alpr  [-c <country_code>] [--config <config_file>] [-n <
          topN>] [--seek
7           <integer_ms>] [-p <pattern code>] [--clock] [-d]
               [-j] [--]
8           [--version] [-h] <image_file_path>
9
10
11 Where:
12
13    -c <country_code>,  --country <country_code>
14      Country code to identify (either us for USA or eu for
           Europe).
15      Default=us
16
17    --config <config_file>
18      Path to the openalpr.conf file
19
20    -n <topN>,  --topn <topN>
21      Max number of possible plate numbers to return.
           Default=10
22
23    --seek <integer_ms>
24      Seek to the specified millisecond in a video file.
           Default=0
25
26    -p <pattern code>,  --pattern <pattern code>
27      Attempt to match the plate number against a plate
           pattern (e.g., md
28      for Maryland, ca for California)
29
```

```
30      --clock
31        Measure/print the total time to process image and all
             plates.
32        Default=off
33
34      -d,  --detect_region
35        Attempt to detect the region of the plate image.  [
             Experimental]
36        Default=off
37
38      -j,  --json
39        Output recognition results in JSON format.  Default=
             off
40
41      --,  --ignore_rest
42        Ignores the rest of the labeled arguments following
             this flag.
43
44      --version
45        Displays version information and exits.
46
47      -h,  --help
48        Displays usage information and exits.
49
50      <image_file_path>
51        Image containing license plates
```

5.3.4. Análisis y resultado de la imagen con Openalpr

En la siguiente figura se muestra un posible resultado gráfico una vez analizada la imagen que contenga a la patente.

Plate	Confidence (%)	Processing Time (ms)
PE3R2X	88.94%	152.94 ms

Fig. 5.1: Imagen original. [20]

5.4. Configuración de Openalpr

Openalpr es un lector de placas de autos de código abierto que da interesantes posibilidades en el tema de vigilancia o en cualquier situación donde el conocer la placa de un auto podría llevar a sanciones para el propietario, este software es uno de los más populares en el mundo para el reconocimiento de placas ya que lleva consigo buenas herramientas que nos ayudaran a entrenar adecuadamente nuestro motor OCR para el reconocimiento de placas de cualquier parte del planeta, si bien Openalpr ya está configurado para varios países como Estados Unidos y varios de sus estados(utilizan placas diferentes en cada estado, caracteres y dimensiones), Brasil, Argentina, Reino Unido, etc. No hay ninguno que haya sido implementado ni entrenado para el reconocimiento de placas chilenas.

5.4.1. Calibración de la cámara

Al escribir en el terminal de Linux el comando **openalpr-utils-calibrate[IMAGEN]** se abre una ventana de configuración de la imagen donde se establece la posición de esta. El resultado de las coordenadas aparecerán en al terminal, estas coordenadas se deben copiar en en el archivo ubicado en **/etc/openalpr/openalpr.conf**, en la línea **prewarp**.

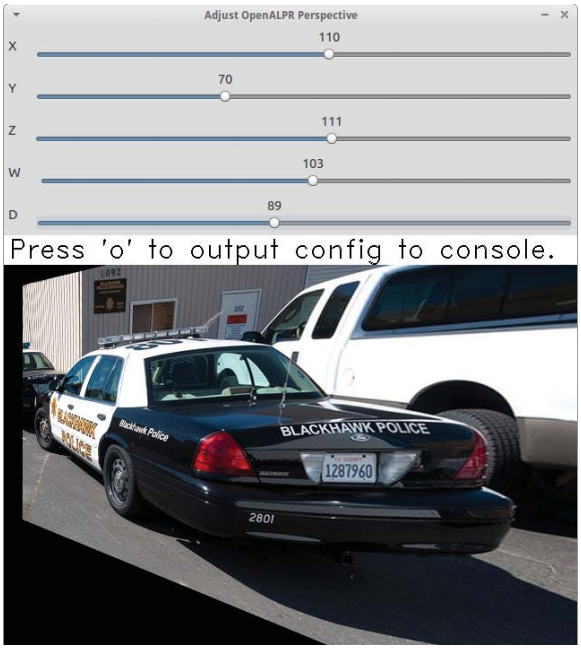

Fig. 5.2: Calibración de cámara. [20]

5.4.2. Coincidencia de patrones

Dirigirse a la dirección donde está instalado Openalpr **/usr/share/openalpr/-runtime_data/postprocess**. En esta dirección crear un archivo **ch.patterns** y modificarlo de la siguiente manera para que contenga, cabe mencionar que los @ representan las letras y los # representan los números:

base @@####
base @@@@##

5.4.3. Configuración de la patente

Configurar las dimensiones de la placa y los caracteres. Añadir un nuevo archivo en **runtime data / config /** con código de 2 dígitos de Chile. Se deben configurar estos valores:

- plate width mm = [Ancho de la patente en mm]
- plate height mm = [Altura de la patente en mm]
- char width mm = [Ancho de un solo carácter en mm]
- char height mm = [Altura de un solo carácter en mm]
- char whitespace top mm = [Espacio en blanco entre el carácter y la parte superior de la patente en mm]
- char whitespace bot mm = [Espacio en blanco entre el carácter y la parte fonde de la patente en mm]
- template max width px = [Anchura máxima de la placa antes del procesamiento. Debe ser proporcional a las dimensiones de la placa]
- template max height px = [Altura máxima de la placa antes del procesamiento. Debe ser proporcional a las dimensiones de la placa]
- min plate size width px = [El tamaño mínimo de una región de la patente que se considere válida (Anchura)]
- min plate size height px = [El tamaño mínimo de una región de la patente que se considere válida (Altura)]
- ocr language = [nombre del idioma de OCR - por lo general sólo la letra l seguido de su código de país]

Así sería el archivo para la configuración de Chile:

- plate width mm = 360
- plate height mm = 120
- char width mm = 35
- char height mm = 70
- char whitespace top mm = 38
- char whitespace bot mm = 38
- template max width px = 120
- template max height px = 60
- min plate size width px = 70
- min plate size height px = 35
- ocr language = lch

5.5. Entrenamiento Openalpr

El entrenamiento de Openalpr es una forma rápida de mejorar la precisión de un país en particular. Para ello, se necesita: Alrededor de 200 imágenes claras de patentes de su país. El desarrollo de este entrenamiento se puede hacer en el sistema operativo Linux 14.04, el cual se ha ocupado durante este desarrollo.

5.5.1. Train-OCR

En este ítem se muestra los pasos a seguir para generar el entrenamiento del banco de patentes. Para realizar este paso es necesario descargar el repositorio Train-OCR desde el enlace `https://github.com/openalpr/train-ocr` [20] y tener por lo menos 200 imágenes limpias sin fondo, solo la placa, para recortar las placas sin fondo véase " Recortar las patentes con imageclipper", contenido en la sección soluciones del anexo.

- 1- La carpeta puede ser almacenada en la ubicación que estimes conveniente, luego se debe crear una carpeta con el nombre **ch** y dentro una carpeta llamada **input**.

- 2- En la terminal de Linux debes ejecutar el comando **openalpr-utils-classifychars [directorio donde se encuentra la colección de imágenes de patentes] [directorio vacío de salida]**. Nota: el directorio de salida contendrá las imágenes de cada carácter reconocido por el comando anterior. Cabe destacar que este es un proceso manual, donde si los directorios están bien, se abrirá una ventana para reconocer los caracteres.

  La siguiente imagen muestra la ventana que se abre al ingresar el comando anteriormente mencionado.

Fig. 5.3: Clasificador de caracteres. [20]

El resultado, luego de la clasificación de los caracteres de cada patente se muestran a continuación.

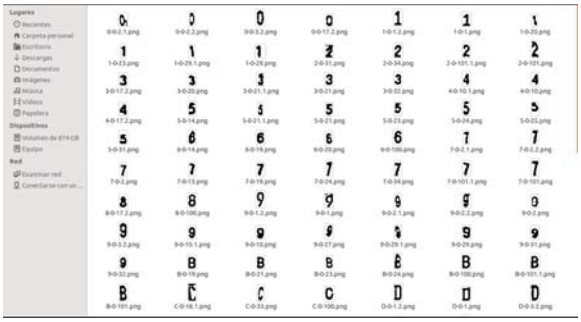

Fig. 5.4: Caracteres resultado. [20]

- 3- Una vez completada la clasificación se debe crear un archivo de extensión

tif y otro de extensión **box**, para ello se debe ejecutar el siguiente comando **openalpr-utils-prepcharsfortraining [directorio de salida de clasificación anterior]**.

- 4- Luego ir al repositorio descargado anteriormente y ejecutar el archivo **python train.py** (no olvidar que se debe arreglar los directorios dentro de este archivo por los tuyos), si todo va bien saldrá un archivo llamado **lch.traineddata**, que se debe pegar en el directorio de openalpr **/usr/share/openalpr/runtime_data/ocr/tessdata**.

5.5.2. Train-Detector

Este ítem muestra como entrenar el detector de patentes. Es importante saber que para el caso favorable del entrenamiento se debe contar con al menos 3000 imágenes de patentes limpias. Al igual que en el anterior ítem se debe descargar un repositorio, esta vez descargaremos el de Train-Detector desde el siguiente enlace `https://github.com/openalpr/train-detector` [20].

Este repositorio contiene la secuencias de comandos que le ayudarán a entrenar el detector de patentes para un país, en este caso en particular Chile.Una vez entrenado su detector puede ser utilizado en Openalpr. El detector de patente utiliza el algoritmo local patrón binario (LBP). Con el fin de capacitar el detector, tendrá muchas imágenes positivas y negativas. Este repositorio ya contiene una colección de imágenes negativas. Usted tendrá que añadir sus propias imágenes positivas.

- 1- Usted debe tener un banco de imágenes que contengan solo la placa sin fondo en una carpeta llamada **ch** dentro de este repositorio.
- 2- Dentro del repositorio hay un archivo llamado **prep.py**, el cual debe ser editado y cambiar la altura, anchura y las variables del país que entrenaremos (Chile).
- 3- Editar, también las rutas de accesos a OpenCv instalado en nuestro computador.
- 4- Una vez listo se debe ejecutar los siguientes comandos en este orden:
  1) **./prep.py neg**
  2) **./prep.py pos**
  3) **./prep.py train**
- 5- Se debe copiar el archivo generado, llamado **cascade.xml** y pegarlo en el directorio **runtime_data / region / [código de país (ch)] .xml)**

Con estos pasos el sistema debiera ser capaz de utilizar la región para la detección de la patente. En el caso de presentarse problemas, al ejecutar el comando que resulta luego de hacer el procedimiento anterior, véase en el anexo, el apartado posible solución train-detector.

5.6. Desarrollo del Sistema

El resultado esperado de reconocimientos para patentes de vehículos particulares en Chile, se ha llevado a cabo con el análisis de la herramienta Openalpr, la cual nos ha dado una alta capacidad de entrenamiento, influyendo en la cantidad de imágenes positivas que se tengan de las patentes nacionales y se ha logrado cumplir el objetivo principal de reconocer patentes chilenas. El planteamiento con imágenes luego el entrenamiento de patentes, ha sido con un total de 347 imágenes positivas, de las cuales al terminar el entrenamiento e implementación de la configuración al sistema de reconocimiento de patentes chilenas, da una cantidad aleatoria de imágenes de las cuales no es capaz de reconocer el sistema, bajo este motivo en el repositorio del sistema entregado, se hace entrega de un Script de Python creado por nosotros, que funciona como clasificador de imágenes la cual el corte de patentes que arrojan un dato es del 80 % y casos favorables es del 90 % el cual nos indica que todos los caracteres de la patente se han podido reconocer.

5.6.1. Código Back-End del sistema

En este extracto del código se hace la interacción con Openalpr para el análisis de la imagen subida al servidor que funciona bajo la arquitectura de Nodejs. En primera instancia para compilar y hacer funcionar el código se necesita la implementación descritas en los puntos anteriores de fase de entrenamiento, luego de comprobar que todo este funcionando se tiene que abrir la carpeta a través de la terminal de tu sistema operativo e ingresar el comando de iniciación de servidor de node y se abrirá una pantalla en tu navegador con el sistema corriendo.

En la siguiente imagen se muestra el extracto del código que ejecuta toda la funcionalidad del sistema para el reconocimiento de las patentes chilena.

```
if(tipo_imagen == 'image/jpeg' || tipo_imagen == 'image/png'){

    var imagen = solicitud.body.archivo.path;
    var auxiliar= imagen.split('/');
    var imagen_mostrar = auxiliar[1]+'/'+auxiliar[2];
    //console.log(imagen_mostrar);
    var exec = require('child_process').exec, child;
    //console.log(imagen);
    child = exec('alpr -c ch -j '+imagen,function
                    (error, stdout, stderr)
    {

    // Esta variable almacena el resultado del comando
```

```
ingresado hacia openalpr.
 var objeto_patentes = stdout;

if (error !== null) {

    console.log('exec error: ' + error);
}else{
// Parseamos el json de una manera mas estructurada.
    var patentes = JSON.parse(objeto_patentes);

// Asignamos los resultados a la variable.
    var results = patentes['results'];

// bandera para ver que trae realmente la variable.
    console.log(patentes);

// En el siguiente if compruebo si existe algun resultado
a mostrar para no hacer caer el servidor.
    if(results[0] != null){

    // Asignamos a la variable las posibles patentes que mas
    se acercan.
         var candidatas =results[0].candidates;

    // creamos una función de respuesta desde el servidor
    a la vista con las candidatas y la imagen que se ingreso.
         respuesta.render('menu/tabla_patente',{

            lista: candidatas,
            imagen: imagen_mostrar
        });

    // si no se encuentra algún resultado se ejecuta
        la accion para informar al usuario.
    }else{

        respuesta.render('menu/new');
        console.log("No se encontraron coincidencias.");
    }
```

5.6.2. Reconocimiento

Una vez entrenado el detector de patentes chilenas con 342 muestras positivas, se ponen a prueba los reconocimientos y confianza para detectar los caracteres contenidos en estas placas vehiculares. Cabe destacar que el reconocimiento habla de la capacidad de detectar las características de la patente chilena y no necesariamente reconocer los caracteres dispuestos en esta.

5.6.3. Porcentajes de reconocimiento

El porcentaje de reconocimiento efectivo del espacio muestral de posibilidades asciende al 26 % de rendimiento en cuanto a las 342 imágenes dispuestas, esto quiere decir que de las 342 patentes, solo 89 han podido ser reconocidas.

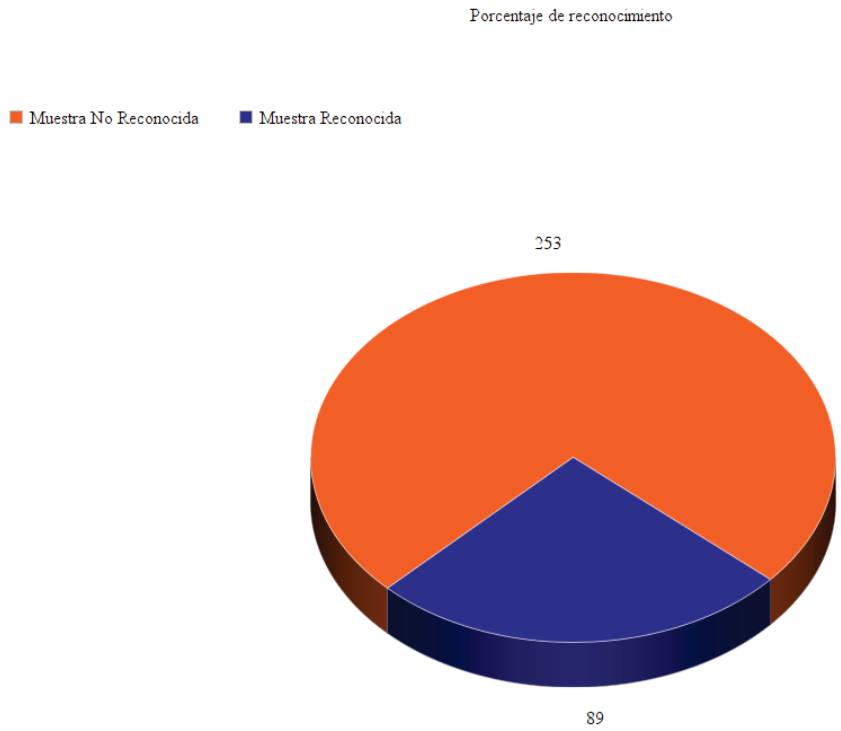

Fig. 5.5: Reconocimiento Total de Patentes. [20]

5.6.4. Segmentos de Reconocimiento

Para el reconocimiento efectivo se ha dispuesto una nueva muestra de patentes las cuales son el 26 % anterior mencionado como patentes reconocidas. Se ha dispuesto de un segmento corte el cual se expresa por la confianza calculada por el sistema web. Un 85 % de confianza será el identificador de patentes reconocidas, pero no acertadas y patentes reconocidas con resultado óptimo cuya confianza se eleva al 90 %. Por lo tanto, de las 89 patentes reconocidas (26 % del total muestral), las dividiremos en dos segmentos:

- 1- Patentes Reconocidas con confianza mayor a 85 %, con un 56 % de la muestra total y una cantidad de 50 patentes acertadas.
- 2- Patentes Reconocidas con confianza menor a 85 %, con un 44 % de la muestra total y una cantidad de 39 patentes.

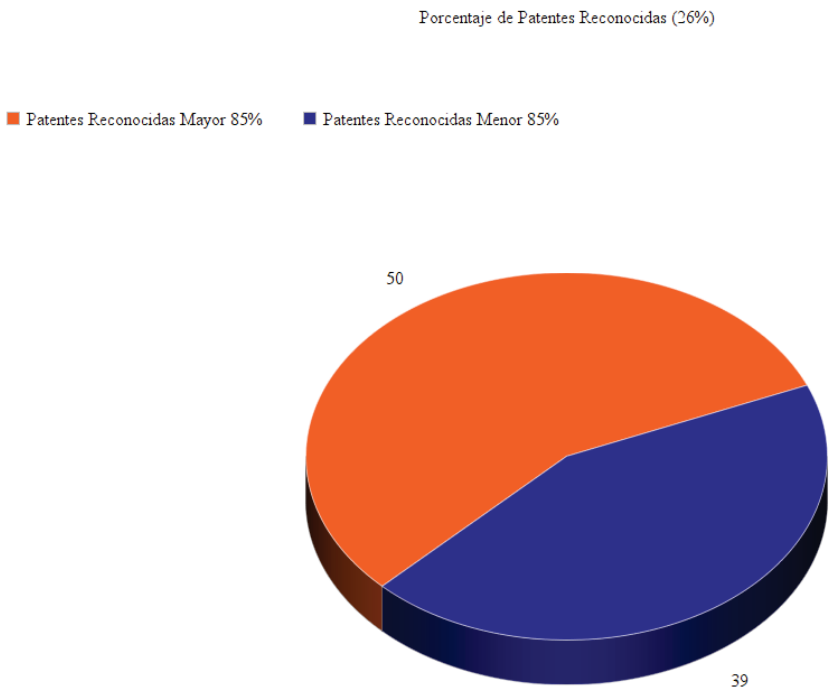

Fig. 5.6: Reconocimiento de patente con 90 %. [20]

5.6.5. Pruebas con clasificador de imágenes total

Gracias a esta clasificación podemos contrastar en un gráfico que por el nivel de entrenamiento, en cuanto a cantidad de imágenes positivas, podemos encontrar el número de imágenes sin resultado y con resultados , la cual se divide en dos categorías: con resultado menor al 85 % de confianza, el cual arroja datos no certeros y mayor a 85 % arroja datos más certeros.

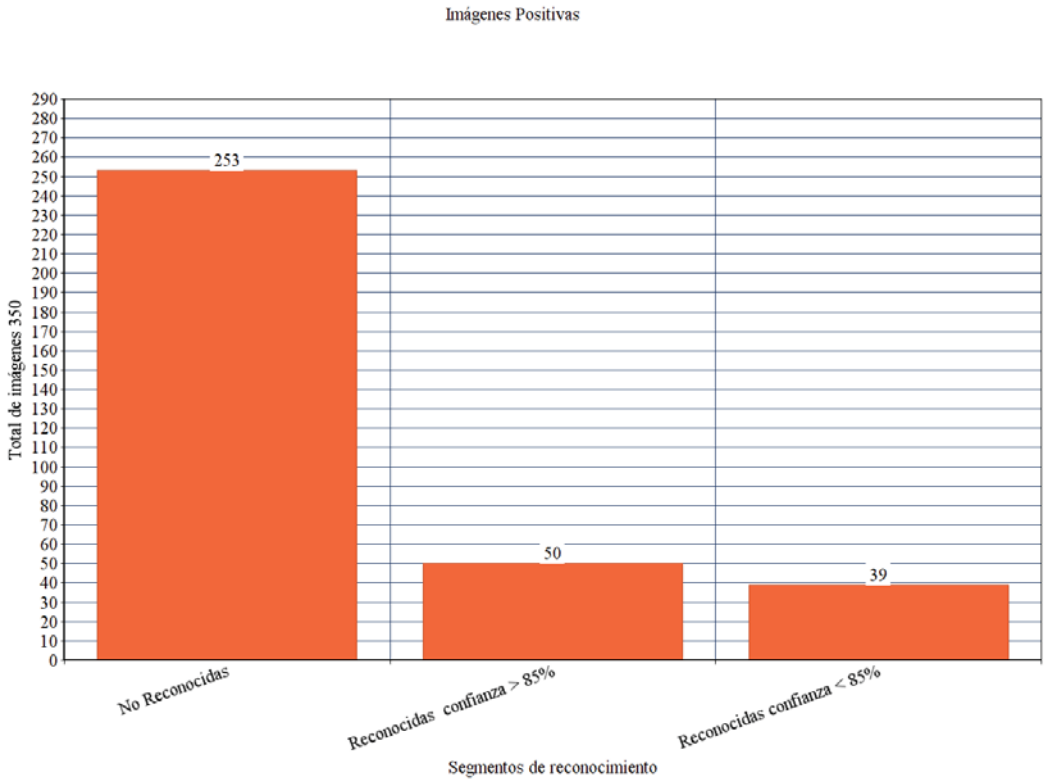

Fig. 5.7: Gráfico de segmentos clasificador de imágenes

5.6.6. Comparación Proyecto con Patentes MercoSur

Se ha evaluado contra un proyecto hecho en Argentina bajo la misma herramienta y el mismo entrenamiento de acuerdo al estudio realizado [7]. Donde se postula que los resultados obtenidos sobre la base de datos de imágenes fueron los siguientes:

- 1-De un total de 200 imágenes fueron reconocidas correctamente 195, contando positivas y negativas.
- 2- Esto nos da un nivel de confianza sobre esta base de datos de 97,50 %.

En comparación al proyecto realizado, se debe aclarar que el 26 % de confianza en este estudio está muy por debajo del rendimiento capaz de entregar esta herramienta de software libre. Esto pone en evidencia que la interacción del clasificador de patentes y calidad de imagen debe ser la adecuada para llegar a un reconocimiento y confianza óptimo de los casos.

6. Conclusión

El reconocimiento óptico de caracteres, puede presentar dificultades a la hora de extraer efectivamente los caracteres ya que hay varios factores que impiden la correcta visualización, tales como: luminosidad, daño de patentes y distancia, estos son factores que intervienen el tratamiento de la imagen. OCR es una tecnología que trata de emular la capacidad del ojo humano para reconocer objetos, concretamente este software permite extraer los caracteres contenidos en una imagen digital, de tal manera que esta información pueda ser comprendida por el ordenador.

Openalpr ejecuta técnicas de las librerías de Opencv, respecto al tratamiento de las imágenes, para lograr la extracción de los caracteres de la patente. Un entrenamiento y configuración para la nacionalidad de la patente es la base fundamental en la precisión de esta herramienta, mostrando la información con una alta confianza, es por esto que la colección de patentes debe ser gigantesca, desde 3000 patentes limpias, para lograr el reconocimiento óptimo. Este proyecto nos ha dado la capacidad de analizar el comportamiento de la visión artificial contenida en la informática, donde se pueden ocupar distintas herramientas para poder crear un conjunto de esta y entregar un sistema capaz de entregar un resultado favorable.

Este software puede tener diferentes funcionalidades dada la alta capacidad de reconocimiento, puede llegar a ser muy eficiente en la utilidad que se le quiera dar.

6.1. Trabajo realizado

La etapa principal de investigación acabada del OCR, ya se encuentra estudiada donde vimos distintos métodos y algoritmos para el reconocimiento de objetos dentro de una imagen digital. El estudio mostró claramente cómo trabaja y compara las técnicas de reconocimiento que existen en la actualidad. Además se mostraron distintos software de reconocimiento, que trabajan con distintas técnicas y métodos para el reconocimiento de las patentes [2].

En esta etapa se analiza el comportamiento de Openalpr, generando la integración de esta herramienta en un sitio web desarrollado con Nodejs y variados paquetes de programación, para que la comunicación fuera en el lenguaje de programación Javascript, donde este brinda una interacción directa con la herramienta Openalpr. Un correcto entrenamiento y configuración nos demuestra que es posible reconocer las patentes chilenas, y en definitiva cualquier tipo de patente a nivel mundial. El entrenamiento de esta herramienta se debe acompañar de un banco de patentes de al menos 3000 o más imágenes, en el caso ideal. En nuestro caso el banco de patentes con el cual se hizo el entrenamiento fue de 343 imágenes positivas, con las cuales se efectuaron los pasos descritos anteriormente, para poder reconocer las patentes chilenas. El prototipo web diseñado, incorpora las funciones

que devuelven un resultado y discrimina entre patentes nuevas y viejas, con respecto a la estandarización de Chile, donde las patentes nuevas constan de cuatro letras y dos número, mientras que las viejas constan de dos letras y cuatro números.

Se ha obtenido un porcentaje de confianza del 26 % respecto a las 342 imágenes de patentes chilenas, comparando con el estudio hecho de patentes MercoSur (patentes argentinas) con la misma herramienta de software libre (OpenAlpr), está muy por debajo de la confianza alcanzada de un 97.5 % obtenido [7] en el estudio citado.

6.2. Trabajo a futuro

El principal objetivo es aumentar el rendimiento del entrenamiento, con el fin de comparar resultados con otros estudios de la misma índole.

Se plantea incorporar el análisis de secuencias de vídeos, donde se capturan las patentes. Esta funcionalidad viene implementada en Openalpr y se le puede sacar provecho para seguir desarrollando este proyecto. Uno de los comportamientos esperados es el reconocimiento óptimo como las patentes europeas o norteamericanas, es por esto que se necesita crear un banco de patentes de al menos 3000 imágenes positivas para hacer este proyecto más efectivo y generar el entrenamiento para Chile.

La opción de tener un panel de configuración y ampliar el banco de patentes ya obtenidos para mejorar el reconocimiento y poder así generar una estadística con respecto a la tendencia que se obtenga, es el principal objetivo del trabajo a futuro esperado.

Referencias

[1] A.G. Linux OCR Software Comparison. https://es.wikipedia.org/wiki/UTF-8/, 2010. [Fecha última visita; 12 de Junio del 2016].

[2] B.C. A.G. *Reconocimiento de patentes.* Pontifica Universidad Católica de Valparaíso., 2016.

[3] Jo Chang-yeon. Face Detection using LBP features. http://cs229.stanford.edu/proj2008/Jo-FaceDetectionUsingLBPfeatures.pdf/. [Fecha última visita; 6 de Diciembre del 2016].

[4] Campos A.N. Cuttitta J.C., De Giovanni G. *RECONOCIMIENTO AUTOMÁTICO DE NÚMERO DE PATENTE* . Universidad Tecnológica Nacional, Facultad Regional Buenos Aires, Dpto. de Electrónica, 2011.

[5] GOCR. GOCR open-source character recognition. http://jocr.sourceforge.net/, 2000. [Fecha última visita; 16 de Abril del 2016].

[6] Gov. Reconocimiento de patentes Chile. http://www.seguridadpublica.gov.cl/sitio-2010-2014/n100_12-04-2012.html, 2014. [Fecha última visita; 30 de Enero del 2018].

[7] José Amado Cristian Caniglia Daniel Puntillo Marcos Blasco Ignacio Moretti, Javier Jorge. *Software libre para reconocimiento automático de las nuevas patentes del Mercosur.* NSTITUTO NACIONAL DE TECNOLOGÍA INDUSTRIAL Avenida Vélez Sársfield 1561 X5000JKC - Córdoba – Argentina.

[8] INTECO. *INTRODUCCIÓN A LA INGENIERÍA DEL SOFTWARE.* Instituto Nacional de Tecnologías de la comunicación, 2009.

[9] Diego Alberto Aracena-Pizarro Juan Francisco Rojas-Henríquez. *Segmentación de Patentes Vehiculares Mediante Técnicas de Agrupamiento en Ambientes Externos.* EUIIIS: Área de Ingeniería en Computación e Informática Universidad de Tarapacá Arica, Chile, 2012.

[10] Tobias Lindahl. *Study of Local Binary Patterns.* Institutionen for teknik och naturvetenskap, 2007.

[11] Books LLC. *Optical Character Recognition: Optical Character Recognition, Machine-Readable Medium, Automatic Number Plate Recognition, Ocr-A Font.* General Books, 2010.

[12] Cordelia Schmid Marko Heikkila a, Matti Pietikainen. *Description of Interest Regions with Local Binary Patterns* . Machine Vision Group, Infotech Oulu and Department of Electrical and Information Engineering, 2007.

[13] ONDREJ MARTINSKY. *ALGORITHMIC AND MATHEMATICAL PRINCIPLES OF AUTOMATIC NUMBER PLATE RECOGNITION SYSTEMS* . BRNO UNIVERSITY OF TECHNOLOGY, 2007.

[14] NodeJs. Documentación NodeJs. https://nodejs.org/docs/latest-v7.x/api/documentation.html, 2017. [Fecha última visita; 5 de Noviembre del 2017].

[15] Topi Maenpaa Matii Pietikainen. *Texture Analysis with Local Binary Patterns.* Electrical and Information Engineering, University of Oulu, Finland., 2004.

[16] Topi Maenpaa Matti Pietikainen. *Department of Electrical and Information Engineering, Infotech Oulu, University of Oulu* . Department of Electrical and Information Engineering, Infotech Oulu, University of Oulu, 2004.

[17] Rdns. Sistemas ANRP Argentina. http://www.rnds.com.ar/articulos/052/RNDS_124W.pdf. [Fecha última visita; 25 de Octubre del 2016].

[18] Yann Rodriguez and Sebastien Marcel. *Face Authentication Using Adapted Local Binary Pattern Histograms* . IDIAP Research Institute, Rue du Simplon 4, 1920 Martigny, Switzerland, -.

[19] Francisco José Núñez Sánchez-Agustino. *Diseño de un sistema de reconocimiento automático de matrículas de vehículos mediante una red neuronal convolucional.* Universitat Oberta de Catalunya, 2016.

[20] OpenALPR Technology. Reconocimiento de patentes con Openalpr. https://github.com/openalpr/openalpr/, 2013. [Fecha última visita; 16 de Junio del 2016].

[21] OpenALPR Technology. Documentación Openalpr. http://doc.openalpr.com/opensource.html#post-processing, 2017. [Fecha última visita; 05 de Noviembre del 2017].

[22] Gobierno Vasco. OCRAD el OCR de GNU. http://www.kultura.ejgv.euskadi.eus/contenidos/informacion/kultura2_0_prestakuntza/es_k20_form/adjuntos/pildora-OCR-2.pdf/, 2016. [Fecha última visita; 16 de Abril del 2016].

[23] Wikepdia. Matrículas Automovilísticas de Chile (PPU). https://es.wikipedia.org/wiki/Matriculas_automovilisticas_de_Chile/, 2016. [Fecha última visita; 16 de Abril del 2016].

[24] Wikipedia. Reconocimiento automático de matrículas. https://es.wikipedia.org/wiki/Reconocimiento_autom%C3%A1tico_de_matr%C3%ADculas, 2017. [Fecha última visita; 5 de Noviembre del 2017].

[25] Fei Zuo. FAST FACIAL FEATURE EXTRACTION USING A DEFORMABLE SHAPE MODEL WITH HAAR-WAVELET BASED LOCAL TEXTURE ATTRIBUTES. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.1878&rep=rep1&type=pdf/. [Fecha última visita; 9 de Diciembre del 2016].

7. Anexo

7.1. Manual de Usuario

En este apartado mostraremos y explicaremos las funcionalidades de manera individual y los posibles casos con respecto a los resultados del análisis de patentes chilenas mediante el prototipo web propuesto.

7.1.1. Vista Principal

Es la primera vista de interacción con el usuario, donde se ofrece tres opciones, representadas con un botón:

- Botón Reconocer Patente : El cual nos envía a la siguiente vista de interacción respecto al reconocimiento de la patente.
- Botón Contacto : Nos da la información básica del creador del prototipo.
- Botón Documentación : Nos redirige a la documentación de Openalpr Oficial.

Fig. 7.1: Vista Principal.

7.1.2. Vista Reconocimiento de Patente

En la siguiente vista se nos presenta un formulario capaz de subir la patente, para posteriormente analizarla. En este apartado existen dos posibles caminos:

- Subir la imagen y aceptar el reconocimiento, el cual nos envía a la tabla de resultados.
- Subir la imagen y no aceptar el reconocimiento, el cual nos envía a una vista de error.

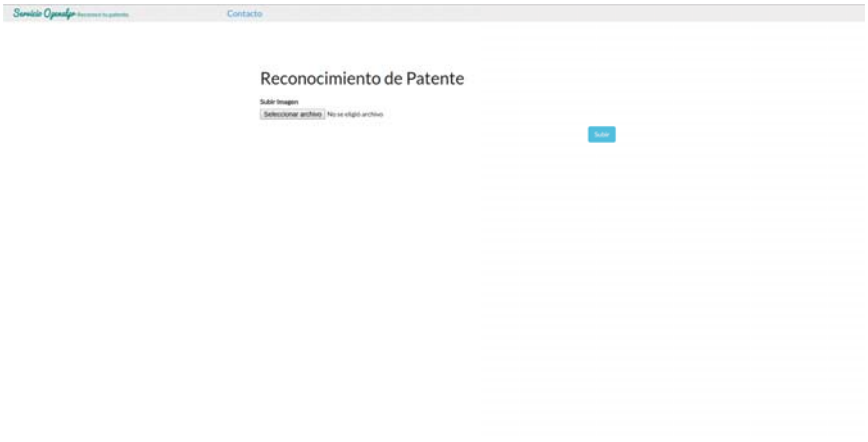

Fig. 7.2: Vista Reconocimiento de Patente.

Fig. 7.3: Vista Error de Reconocimiento de Patente.

7.1.3. Vista Tabla de resultados

En esta vista, se muestra una tabla con el resultado de la patente, con un set de candidatos más cercanos al reconocimiento, donde se expresa el óptimo gracias al porcentaje de confiabilidad (se entiende que la confiabilidad más alta es el resultado esperado). También se posiciona al lado derecho de esta tabla, la imagen de la patente subida a nuestro servidor, para realizar el reconocimiento de esta. La tabla presta funcionalidades de paginación y de busqueda dentro de esta misma coincidencias de lo ingresado en el buscador.

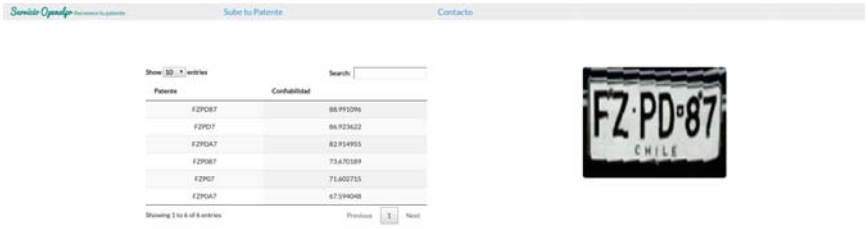

Fig. 7.4: Vista Tabla de resultados.

7.2. Soluciones

En este apartado, se contará con las posibles soluciones encontradas con distintos problemas sucitados en el transcurso de este proyecto.

7.2.1. Recortar las patentes con imageclipper

Esta herramienta nos permite cortar grandes cantidades de imágenes, el cual se encuentra en el siguiente repositorio https://github.com/openalpr/imageclipper. En esta ocación nos sirve para cortar todas las patentes con distintos angulos, puesto que cuenta con un panel de configuración la cual te permite ir editando la imagen para posicionarla de manera frontal, y realizar el corte para luego guardarla en tu dirección preferencial.

En la siguiente figura se muestra el funcionamiento de esta herramienta.

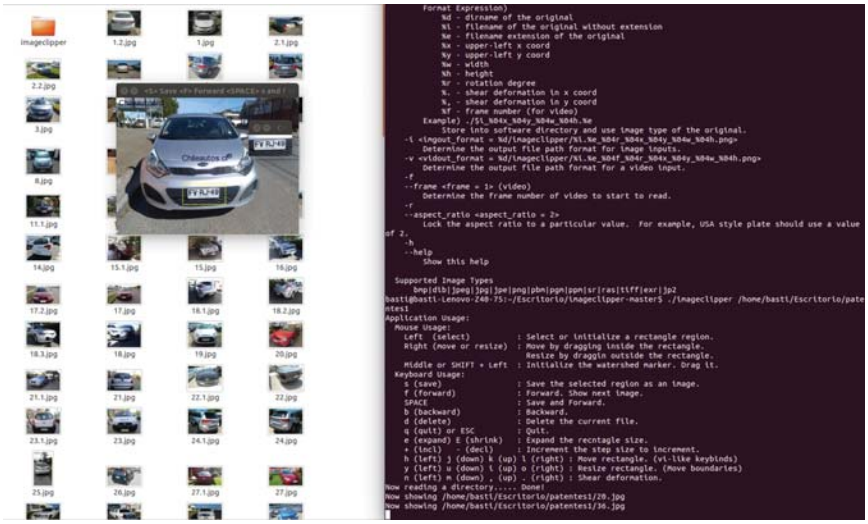

Fig. 7.5: Imageclipper

7.2.2. Posible solución Train-detector

En la siguiente imagen se muestra el error del comando resultante de ejecutar los comandos asociados al train-detector, en la fase de entrenamiento de este proyecto.

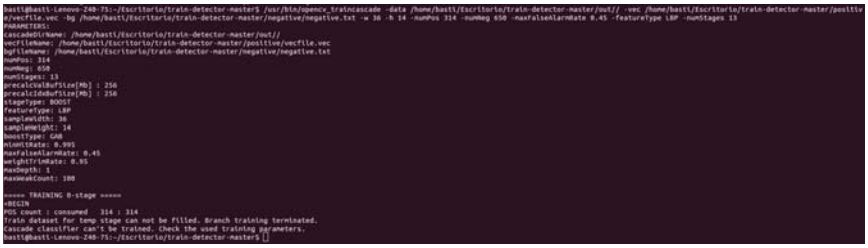

Fig. 7.6: Error Train-detector.

Una posible solución a este problema, es ocupar un sistema operativo OSx, ya que el problema consite en caracteres especiales creados al final de cada string de dirección de la imagen analizada en el sistema Linux. Dado este problema se recomienda ejecutar el proceso del entrenamiento Train-detector, en un equipo con sistema OSx, resultando el entrenamiento efectivo con la devolución del archivo cascade.xml esperado.

Obs : Si al ejecutar el siguiente proceso, ves que el resultado sigue produciendo el mismo error, puedes cambiar uno de los parametros del comando. LBP, de alguna manera no funciona con un repositorio de imágenes pequeñas, no obstante existe otro parametro de entrenamiento llamado HAAR [25], el cual hace funcionar el entrenamiento de train-detector, cabe destacar que es necesario poner como parametro de cantidad de imágenes positivas, un 20 % menos de estas para poder suplir los falsos positivos y no consumir todas las imágenes como casos favorables, de esta manera se puede resolver el error que nos ha estado saliendo.