

Reconocimiento Automático del Número de Placa Vehicular

Jorge Ramos Kenny Méndez Adrián Rodríguez

CIMAT

13 de Diciembre de 2018



Agenda

- 1 Introducción**
 - Planteamiento
 - Objetivos
- 2 Detección de placas**
 - Métodos de detección de placas
- 3 Detección de caracteres**
 - Métodos OCR
- 4 Implementación del Sistema ANPR**
 - Obtención de las imágenes
 - Preproceso
 - Experimentos
 - ANPR 0.9 (versión beta)
- 5 Resultados**
 - ¿Cómo evaluamos?
 - Lo bueno y lo no tan bueno
- 6 Conclusiones**
- 7 Referencias**

- 1 Introducción**
 - Planteamiento
 - Objetivos
 - 2 Detección de placas**
 - Métodos de detección de placas
 - 3 Detección de caracteres**
 - Métodos OCR
 - 4 Implementación del Sistema ANPR**
 - Obtención de las imágenes
 - Preproceso
 - Experimentos
 - ANPR 0.9 (versión beta)
 - 5 Resultados**
 - ¿Cómo evaluamos?
 - Lo bueno y lo no tan bueno
 - 6 Conclusiones**
 - 7 Referencias**

Introducción

El reconocimiento automático de placas vehiculares es un problema importante debido a que el crecimiento del parque vehicular en los países desarrollados y en desarrollo va en aumento, por lo que mantener control de velocidades para evitar accidentes y la capacidad de encontrar vehículos robados se han convertido en problemas comunes y que pueden prevenirse con un buen uso de esta aplicación.

Se han establecido varias soluciones a este problema, pero no existe una solución universal debido a que se pueden presentar problemas como vehículos en movimiento, cambios de iluminación, movimientos de la cámara, obstrucciones, cercanía-lejanía de la foto al vehículo, fondo cambiante.



Figura: Sistema de detección de placas en tiempo real.

Objetivos

- Explorar los distintos algoritmos que existen para esta tarea (algunos de mayor popularidad) y evaluar su desempeño.
 - Descubrir limitantes y/o como atacarlas.
 - Implementar la base para un Sistema Automático de Reconocimiento del Número de Placa Vehicular.

Diseño del Sistema ANPR

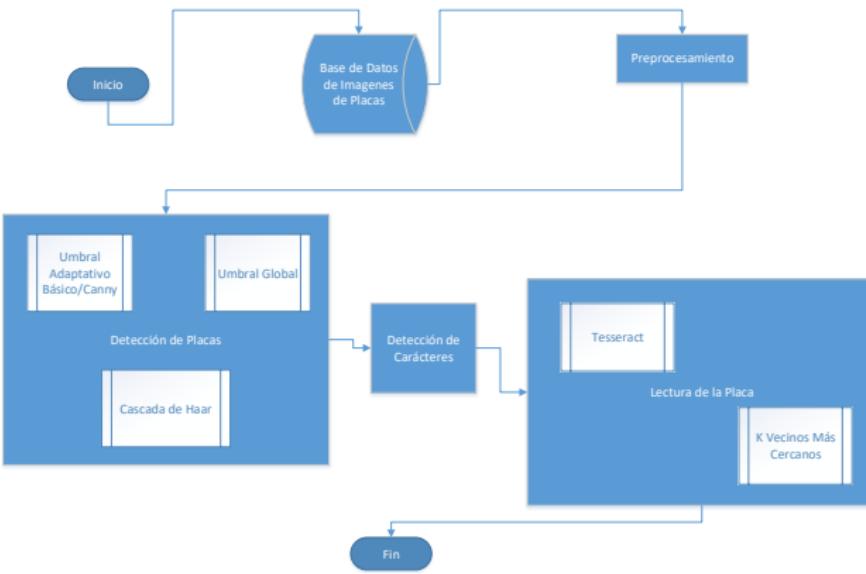


Figura: Diagrama general para el Sistema Automático de Reconocimiento del Núm. de Placa

- 1** Introducción
 - Planteamiento
 - Objetivos
 - 2** Detección de placas
 - Métodos de detección de placas
 - 3** Detección de caracteres
 - Métodos OCR
 - 4** Implementación del Sistema ANPR
 - Obtención de las imágenes
 - Preproceso
 - Experimentos
 - ANPR 0.9 (versión beta)
 - 5** Resultados
 - ¿Cómo evaluamos?
 - Lo bueno y lo no tan bueno
 - 6** Conclusiones
 - 7** Referencias

Detector de bordes de Canny

- 1 **Preprocesamiento** Debido a que los detectores son sensibles al ruido, es aconsejable aplicar un poco de suavizado con un desenfoque (blur), usualmente un desenfoque gaussiano.
 - 2 **Cálculo de gradientes** Se calculan las magnitudes y direcciones en cada punto de la imagen. Donde la magnitud del gradiente en cada punto determina si posiblemente yace o no un borde.

Detector de bordes de Canny

La dirección del gradiente muestra hacia donde está orientado el borde. Para calcularlos se utiliza **el detector de bordes Sobel**, el cual trabaja en primeras derivadas de manera separada para los ejes X y Y.

La magnitud del gradiente está dada por

$$m = \sqrt{G_x^2 + G_y^2}$$

y la dirección

$$\theta = \arctan \frac{G_y}{G_x}$$

donde G_x y G_y son las derivadas X y Y en el punto considerado. Una vez ya calculadas se inicia con la detección de bordes.

Detector de bordes de Canny

- 3 **Supresión de no máximos** Si un píxel no es un máximo, este es suprimido. Para hacer esto, se hace una iteración sobre todos los píxeles de la imagen. Asimismo, la orientación de cada pixel se coloca dentro de cuatro contenedores, los cuales rodean al pixel en cuestión, ya que hay cuatro posibles bordes; ir de norte a sur, de este a oeste, o en diagonal de arriba hacia abajo o de abajo hacia arriba.
Asimismo, las cuatro posibilidades necesitan ser consideradas por separado para revisar la supresión de no máximo.
- 4 **Umbral con histéresis** Despues de realizar la supresión de no máximos, se obtienen los llamados “bordes delgados”. Utilizando la información de dirección y el umbral inferior, “se amplifican” estos bordes.

Detector de bordes de Canny

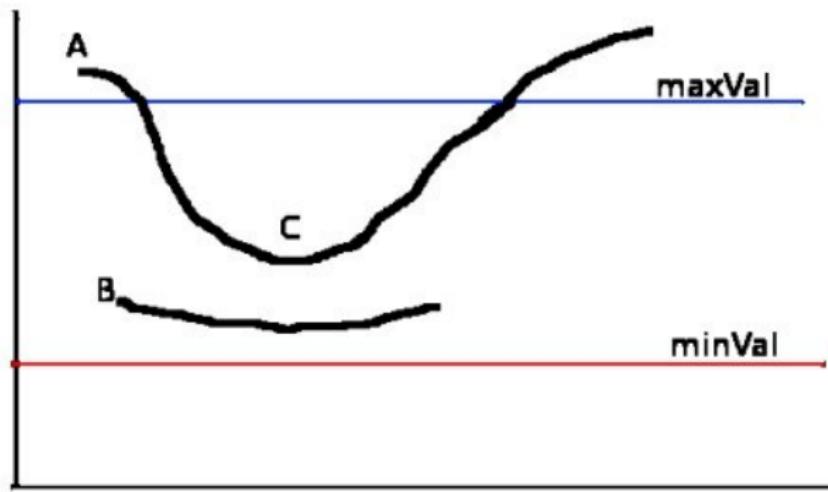


Figura: Proceso de histeresis

Detector de bordes de Canny

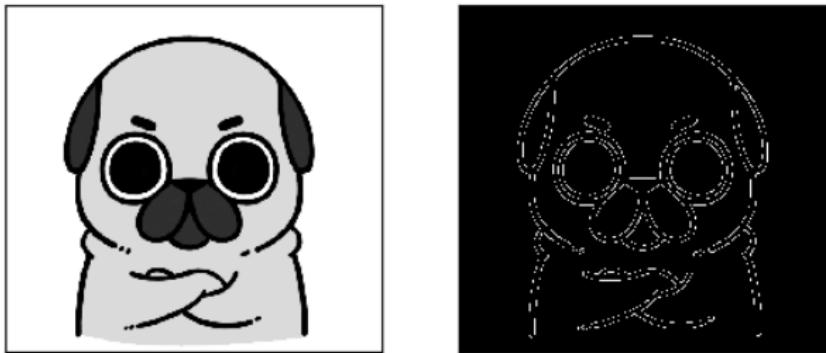


Figura: Imagen original e imagen aplicando el filtro de Canny

Umbral Adaptativo Básico (Basic Adaptive Thresholding)

Hay factores que transforman un histograma perfectamente segmentable por un umbral global en uno que no, así para manipular tales situaciones se divide la imagen en subimagenes y entonces se utiliza un umbral diferente para segmentar cada subimagen. Debido a que el umbral utilizado para cada píxel depende de donde esté localizado dentro de la subimagen, por lo que este tipo de umbral es adaptativo.

[View all posts by **John Doe**](#) [View all posts in **Category A**](#) [View all posts in **Category B**](#)

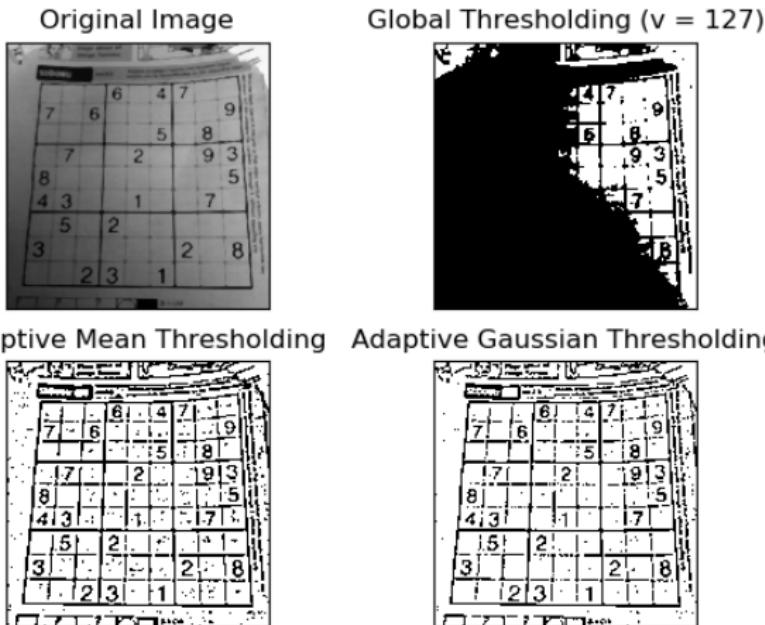


Figura: Imagen original con cambios de iluminación, imagen con umbral global, imagen con umbral adaptativo y kernel de media, imagen con umbral adaptativo y kernel gaussiano

Cascada de Haar

La cascada de Haar es un método de detección de objetos, la aportación principal de este método es su eficiencia en tiempo computacional ya que inclusive permite ser utilizado en tiempo real. Además de que se logra una muy buena tasa de detección con un nivel de falsos positivos bajo.

El aporte completo de dicho método se basa en tres ejes:

- Características de tipo Haar
- Imagen integral
- Cascada de clasificadores

Cascada de Haar

Este algoritmo es muy rápido y en general da muy buenos resultados al detectar objetos. En la figura 6 se presenta una idea básica de como funciona el algoritmo sobre un rostro.

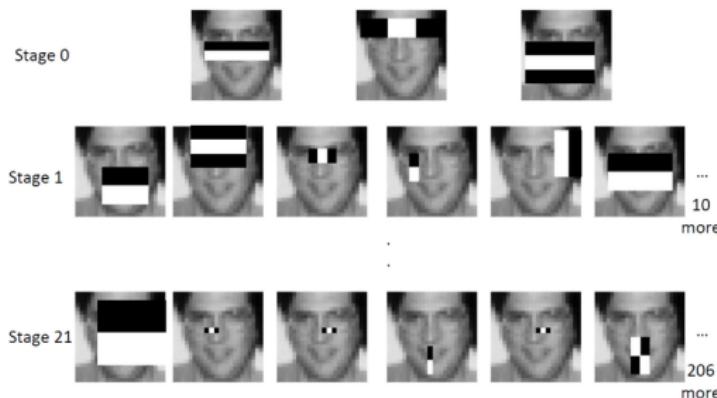


Figura: Aplicando una cascada de Haar sobre un rostro

Su utilización en OpenCV¹ mediante Python es sencilla y por tanto es factible entrenar nuestros propios modelos, como veremos más adelante.

¹ https://docs.opencv.org/3.4.3/dc/d88/tutorial_traincascade.html

- 1 Introducción**
 - Planteamiento
 - Objetivos
- 2 Detección de placas**
 - Métodos de detección de placas
- 3 Detección de caracteres**
 - Métodos OCR
- 4 Implementación del Sistema ANPR**
 - Obtención de las imágenes
 - Preproceso
 - Experimentos
 - ANPR 0.9 (versión beta)
- 5 Resultados**
 - ¿Cómo evaluamos?
 - Lo bueno y lo no tan bueno
- 6 Conclusiones**
- 7 Referencias**

K-Nearest Neighbors (KNN)

Es uno de los algoritmos mas simples de clasificación, donde se tienen etiquetados casos en un conjunto de entrenamiento; se espera que los datos pertenecientes a una categoría estén bastante cerca mientras que se encuentran alejados aquellos que no pertenecen a su categoría. Por lo que al entrar una nueva observación se buscan sus k -vecinos más cercanos en base a alguna función de distancia predefinida anteriormente y se elige a que categoría pertenece generalmente en base a un criterio de mayoría.

KNN

En nuestro caso se genera una imagen de entrenamiento que contiene todos los dígitos y caracteres usando fuentes similares a las placas.

A partir de aqui se segmentan las imagenes usando un umbral global, y se localizan los contornos y de manera interactiva se establece a que digito o caracter pertenece al ser seleccionado cada caracter durante el entrenamiento.

KNN

0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Figura: Imagen de entrenamiento para KNN

Tesseract OCR

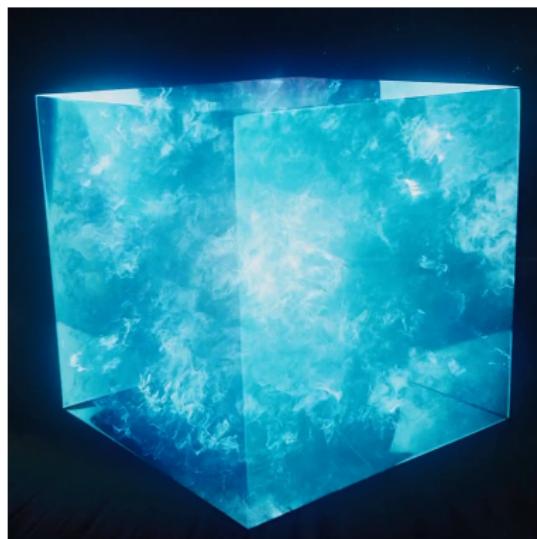


Figura: En geometría, un teseracto es una figura formada por ocho cubos tridimensionales ubicados en un espacio donde existe un cuarto eje dimensional.

El proyecto *Tesseract* fue desarrollado por Hewlett Packard Labs, pero en 2005 se liberó en colaboración con la Universidad de las Vegas, Nevada. A partir de 2006 debido a que se liberó varios contribuidores han aparecido para mejorar sus resultados.

Métodos OCB

Tesseract project

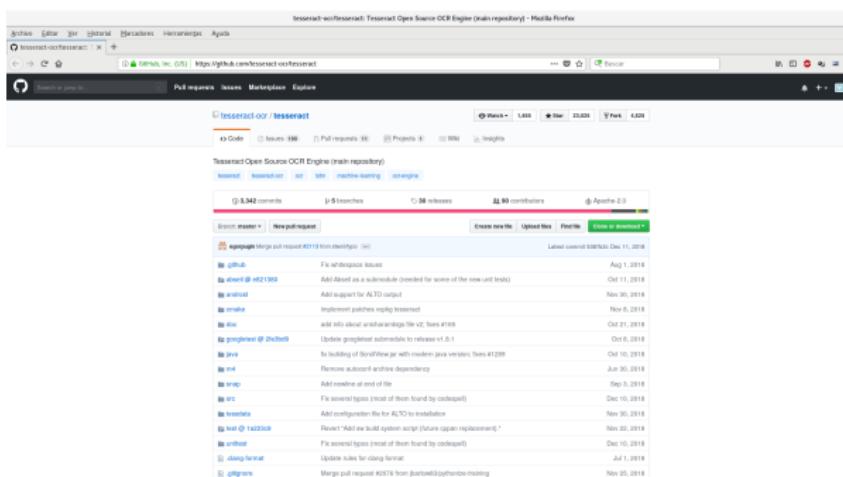


Figura: Proyecto *Tesseract* en Github.²

Tesseract 4.0 es la versión más nueva de este producto, y la novedad es que aparte del proyecto Legacy que se tenía en Tesseract 3.0 ahora se incorporó una red neuronal de tipo LSTM que es la usada por defecto para la lectura de caracteres en imágenes.



²<https://github.com/tesseract-ocr/tesseract>

LSTM

Las redes LSTM son un tipo especial de RNN, capaces de aprender dependencias de plazo largo.

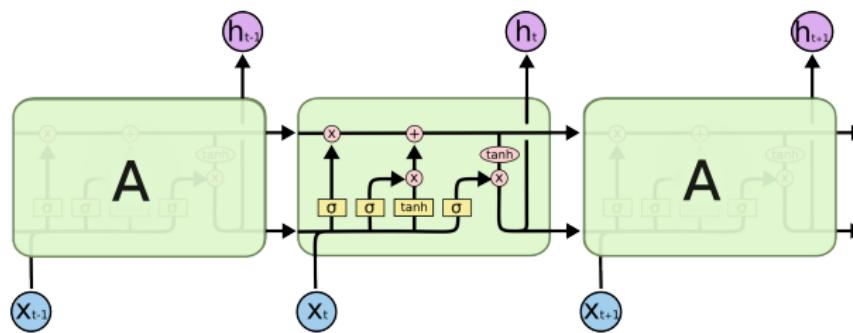


Figura: El modulo repetido en una red LSTM tiene cuatro capas interactuando.

- 1 Introducción**
 - Planteamiento
 - Objetivos
- 2 Detección de placas**
 - Métodos de detección de placas
- 3 Detección de caracteres**
 - Métodos OCR
- 4 Implementación del Sistema ANPR**
 - Obtención de las imágenes
 - Preproceso
 - Experimentos
 - ANPR 0.9 (versión beta)
- 5 Resultados**
 - ¿Cómo evaluamos?
 - Lo bueno y lo no tan bueno
- 6 Conclusiones**
- 7 Referencias**

Obtención de las imágenes

Base de datos

Para el entrenamiento y construcción de los modelos propuestos, se recolectaron los siguientes tipos de imágenes:

- Imágenes de vehículos. se hizo uso de una base de datos de vehículos³ que contiene **151 imágenes** (+).
- Imágenes que no contuvieran placas (-)

³ http://www.medialab.ntua.gr/research/LPRdatabase/Still_images/day/day_very_close_view.zip

Obtención de las imágenes

Imágenes positivas y negativas

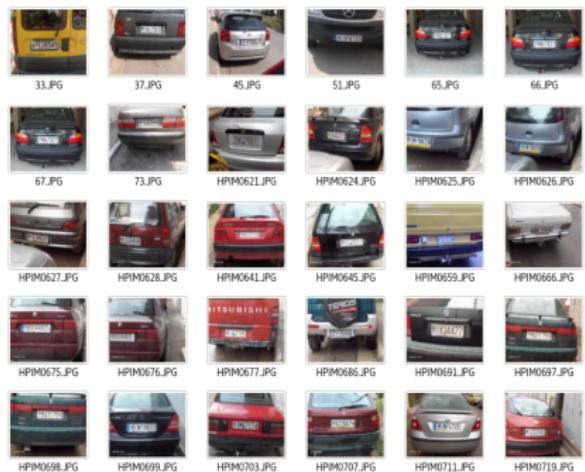


Figura: Muestra de imágenes **positivas** (autos con placas vehiculares).

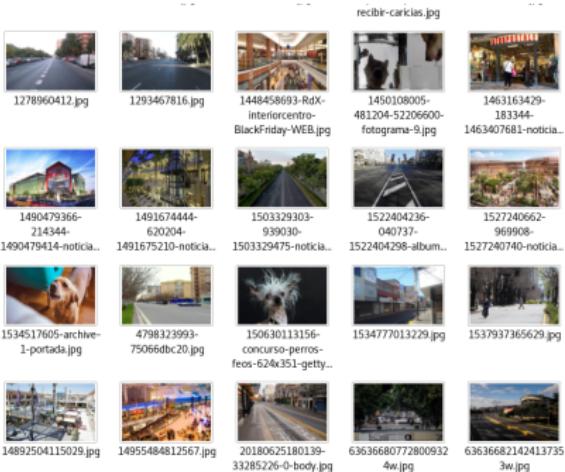


Figura: Muestra de imágenes **negativas** (no contiene placas vehiculares).

Consideraciones



Para probar nuestros métodos de vehículos tomadas con suficiente luz y cercanía. Sin embargo, muchas de las fotos presentan muchas variaciones como ángulos distintos, perspectivas de toma diferentes, presencia de sombras sobre las placas, así como algunas fotografías en donde las placas se ven borrosas.

Redimensionamiento de imágenes

Se aplicó para

- Cascada de Haar
- Canny

Umbralización

- Sobre las placas directamente
- Se probó con distintos métodos
- Aplicando en secuencia algún método de OCR (evaluación)

Ejemplo usando el Filtro de Canny



Figura: Imagen original de uno de los automóviles e imagen al aplicarsele el filtro de Canny.

Entrenamiento con Haar

```
kenny@hotmachine:~/Documents/MCE-CIMA/Semestre-3/datos-complejos/dp-modulo/prj/plate-detect/opencv-haar-classifier-training
```

```

Archivo Editor Ver Buscar Terminal Ayuda
Precalculation time: 47
=====
| N | HR | FA |
| 1| 1| 1|
| 2| 1| 1|
| 3| 1| 1|
| 4| 1| 0.913828|
| 5| 1| 0.9699|
| 6| 1| 0.721443|
| 7| 1| 0.619238|
| 8| 1| 0.58162|
| 9| 1| 0.264529|
=====
END>
Training until now has taken 0 days 2 hours 10 minutes 21 seconds.

=====
names TRAINING 19-stage *****
=====
real    78m15.046s
user   137m00.613s
sys    3m00.318s
(kenny@hotmachine opencv-haar-classifier-training)8

```

Entrenamiento con Haar



ANPR 0.9

```
1 import platerecog as pr # Own Plate-OCR library
2 import os
3
4 path = "../FUENTES/day_very_close_view_adrian/"
5 placas_method = "haar"
6 ocr_method = "tesseract"
7 outputfile = "results/" + placas_method + "_" + ocr_method + ".txt"
8 myf = open(outputfile, "w")
9
10 files = sorted(os.listdir(path)) # get files from input directory
11 plates_detected = 0 # set descriptors
12 cont = 1
13
14 for filename in files:
15     print("Processed ", round(100 * cont / len(files), 2), "%")
16     recog = pr.ReconocePlaca(path + filename) # Creamos una instancia
17     # Buscamos posibles placas
18     recog.encuentra_placas(tipo_prep = placas_method)
19     plates_detected += (len(recog.lista_placas) > 0)
20     # Buscamos posibles textos
21     recog.placa_ocr(tipo_ocr = ocr_method)
22
23     line = ""
24     for j in range(len(recog.posibles_textos)):
25         if j > 0: line = line + ","
26         text = recog.posibles_textos[j]
27         line = line + str(text)
28     myf.write(filename + " : " + line + "\n")
29     cont += 1
30 print("Número de posibles placas detectadas: ", plates_detected)
31 myf.close()
```

- 1 Introducción**
 - Planteamiento
 - Objetivos
- 2 Detección de placas**
 - Métodos de detección de placas
- 3 Detección de caracteres**
 - Métodos OCR
- 4 Implementación del Sistema ANPR**
 - Obtención de las imágenes
 - Preproceso
 - Experimentos
 - ANPR 0.9 (versión beta)
- 5 Resultados**
 - ¿Cómo evaluamos?
 - Lo bueno y lo no tan bueno
- 6 Conclusiones**
- 7 Referencias**

¿Cómo evaluamos?

Métricas utilizadas

- Para obtener la precisión en lo que respecta al reconocimiento de los caracteres de la placa se optó por usar una métrica de similaridad, haciendo uso de la distancia *Levenshtein* (para normalizar la distancia entre 0 y 1)

$$\text{Similaridad} = \frac{\text{largo placa} - \text{dist levensthein}}{\text{largo placa}}$$

- Para una segunda métrica se considero tomar la media de los valores obtenidos para cada lectura de la placa.

Lo bueno y lo no tan bueno



Lo bueno y lo no tan bueno

Resultados

	placa	Canny-placa	thresh1	thresh2	Haar	global	Unnamed: 6	global+Haar	texto	pred	pred_token	predkenny
0	33.JPG	1.0	0.0	1.0	1.0	BOOL99AN	1	1	APE-8549	True	APE8549	0.875000
1	37.JPG	1.0	1.0	1.0	1.0	BOOL99AN	1	1	MIB-7969	True	MIB7989	0.750000
2	45.JPG	1.0	0.0	0.0	1.0	BOOL99AN	1	1	MIN-8985	False	g585	0.250000
3	51.JPG	1.0	0.0	1.0	0.5	BOOL99AN	1	1	MIN-8793	False	HILL8788	0.375000
4	65.JPG	1.0	1.0	1.0	0.5	BOOL99AN	1	1	YMN-7927	True	YMN7927	0.875000
5	66.JPG	1.0	1.0	1.0	1.0	BOOL99AN	1	1	YMN-7927	True	YMN7927	0.875000
6	67.JPG	1.0	1.0	1.0	1.0	BOOL99AN	1	1	YMN-7927	True	YMN27927	0.875000
7	73.JPG	1.0	0.0	1.0	1.0	BOOL99AN	1	1	MIE-5948	False	None	0.000000
8	HPIM0621.JPG	1.0	0.0	1.0	1.0	BOOL99AN	1	1	ZHE-7081	True	ZHE7081	0.750000
9	HPIM0924.JPG	1.0	0.0	1.0	0.0	BOOL99AN	1	1	MIK-R371	True	MIKFR37	0.625000

Figura: Evaluación de las pruebas sobre nuestro sistema ANPR en un Notebook de Python.

cd CODIGO
jupyter-notebook

http://localhost:8888/notebooks/resultados_deteccion.ipynb

Lo bueno y lo no tan bueno

Lo bueno



Figura: Placas detectadas usando el método de Cascada de Haar.

Lo bueno y lo no tan bueno

Detección de placa

Precisión obtenida			
Canny	Thresholding 1	Thresholding 2	Haar
89.3 %	40.9 %	64.7 %	61.0 %

Cuadro: Resumen de resultados sobre la detección de placas.

Al combinarse todos los métodos previos se puede obtener una precisión de detección de la placa del **99.10 %**, lo cual equivale a solo no obtener una placa completa en las 122 imágenes.

OCR para las placas

Para la similaridad parcial entre los OCR y los caracteres verdaderos de las placas se obtuvo una precisión del **61.48 %**.

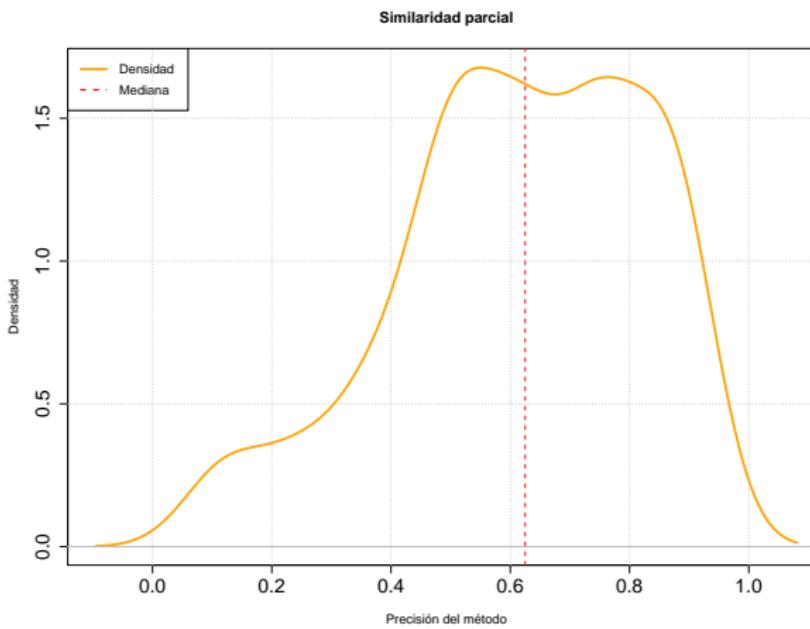


Figura: Precisión obtenida bajo la métrica de similaridad parcial.

OCR para las placas

Imagen	Núm. placa	OCR	Similaridad
HPIM1118.JPG	YHT5335	YHT5335	1.000000
HPIM1249.JPG	YHB3424	YHB3424	1.000000
HPIM0645.JPG	MIE8817	MIE8817	1.000000
HPIM0676.JPG	YBX4482	YBX4482	1.000000
HPIM0707.JPG	YXZ5674	YXZ5674	1.000000
HPIM0677.JPG	MIA7795	MIA7795	1.000000
HPIM1242.JPG	YPE8932	YPE8932	1.000000
HPIM1244.JPG	YEA8054	YEA8054	1.000000
66.JPG	YMN7927	YMN7927	1.000000
65.JPG	YMN7927	YMN7927	1.000000
PHT00051.JPG	YKK2252	YKK2252	1.000000
HPIM0719.JPG	MIZ3540	MIZ3540	1.000000

Cuadro: Primeras 10 imágenes para las que se encontro la mayor similaridad en la lectura de las placas.

Placas que no se pudieron detectar (Lo no tan bueno)



Figura: Imágenes con efectos de desgaste, oclusión, desenfoque e iluminación

Lo bueno y lo no tan bueno

Placas detectadas pero sin OCR

Imagen	Núm. placa	Similaridad
PHT00072.JPG	MIA2957	0.0
HPIM0784.JPG	MIB6822	0.0
HPIM0785.JPG	YYT9659	0.0
HPIM1170.JPG	IBZ6378	0.0
HPIM1211.JPG	MIM9621	0.0
HPIM0925.JPG	YBX4482	0.0
HPIM0933.JPG	TK1688	0.0
HPIM0965.JPG	MIZ7793	0.0
HPIM1004.JPG	ZMA2474	0.0
HPIM1039.JPG	IZT2313	0.0

Cuadro: Primeras 10 imágenes para las que se encontró la peor similaridad en la lectura de las placas.

Lo bueno y lo no tan bueno

Demo

- 1 Introducción**
 - Planteamiento
 - Objetivos
- 2 Detección de placas**
 - Métodos de detección de placas
- 3 Detección de caracteres**
 - Métodos OCR
- 4 Implementación del Sistema ANPR**
 - Obtención de las imágenes
 - Preproceso
 - Experimentos
 - ANPR 0.9 (versión beta)
- 5 Resultados**
 - ¿Cómo evaluamos?
 - Lo bueno y lo no tan bueno
- 6 Conclusiones**
- 7 Referencias**

Conclusiones

- Existen varias formas de resolver parcialmente el problema del reconocimiento debido a que se pueden presentar muchas variaciones en las imágenes como cambios de intensidad de luz en las placas así como la presencia de oclusiones que pueden llevar a los métodos tradicionales a fallar.
- Usando Adaptative Thresholding-Canny mostró los mejores resultados; aunque aun pueden alcanzarse a lograr mejoras con el método de Cascada de Haar pero debido a que toma un tiempo considerable para su entrenamiento no fue posible considerar mas combinaciones de parámetros.
- Por parte de los métodos de OCR se encontró que las variaciones por rotaciones, oclusiones, desvanecimientos e incluso variaciones en el tamaño de la placa pueden afectar demasiado su desempeño.
- Aun cuando se encontraron resultados no tan alentadores incluso combinando ambos métodos de OCR se debe considerar que debido a la variación en las imágenes de las placas obtenidas, el resultado es considerablemente bueno.

- 1** Introducción
 - Planteamiento
 - Objetivos
 - 2** Detección de placas
 - Métodos de detección de placas
 - 3** Detección de caracteres
 - Métodos OCR
 - 4** Implementación del Sistema ANPR
 - Obtención de las imágenes
 - Preproceso
 - Experimentos
 - ANPR 0.9 (versión beta)
 - 5** Resultados
 - ¿Cómo evaluamos?
 - Lo bueno y lo no tan bueno
 - 6** Conclusiones
 - 7** Referencias

-  **Teseracto.**
<https://es.wikipedia.org/wiki/Teseracto>, Diciembre 2018.
-  **Louka Dlagnekov.**
License plate detection using adaboost.
Computer Science and Engineering Department, San Diego, 2004.
-  **Rafael C Gonzalez, Richard E Woods, et al.**
Digital image processing, 2002.
-  **Wing Teng Ho, Wooi Hen Yap, and Yong Haur Tay.**
Learning-based license plate detection on edge features.
In *Proceedings of the 10th MMU International Symposium on Information and Communications Technologies*, 2007.
-  **Ana Riza F Quiros, Rhen Anjerome Bedruz, Aaron Christian Uy, Alexander Abad, Argel Bandala, Elmer P Dadios, Arvin Fernando, and De La Salle.**
A knn-based approach for the machine vision of character recognition of license plate numbers.
In *Region 10 Conference, TENCON 2017-2017 IEEE*, pages 1081–1086. IEEE, 2017.
-  **Paul Viola and Michael J Jones.**
Robust real-time face detection.
International journal of computer vision, 57(2):137–154, 2004.