

Reporte Proyecto Final PDI

Cristian David Ospina Primero

201556123

I. RESUMEN

El uso de sistemas que se basan en la visión por ordenador se han convertido en alternativas eficientes en la cuestión de análisis de datos, esto lo podemos ver aplicado a sistemas de vigilancia y sistemas de control y monitoreo de tráfico de vehículos en ciudades o carreteras. Mediante este documento se dará a conocer el proceso que se llevó a cabo, así como la explicación del algoritmo y las librerías que se usaron para realizar la aplicación descrita en el enunciado del proyecto, en este caso una aplicación de reconocimiento de placas para carros, se tiene que nuestro sistema recibe una imagen, la cual procesa, para posteriormente aplicar el algoritmo de reconocimiento a través de su placa.

II. INTRODUCCIÓN

El procesamiento digital de imágenes (PDI) es un **área importante de la computación**, y sus aplicaciones no son limitadas, hoy en día podemos ver como se utilizan en diversos sectores de la industria, comercio, sistemas médicos, seguridad, etc.

En el enunciado del proyecto final de PDI se plantea desarrollar una aplicación de reconocimiento de placas, se plantea poder identificar un vehículo mediante su placa, la idea del enunciado, es el uso de imágenes precargadas de vehículos las cuales sean visibles sus placas o matrículas, con la intención de poder segmentar estas, obtener su placa, para posteriormente lograr interpretar esta.

Para poder alcanzar los objetivos previamente descritos, se definieron ciertas etapas las cuales se explicarán a continuación:

- Carga de la imagen.
- Pre procesamiento de la imagen, usando filtros con el fin de mejorar la imagen, para obtener el resultado deseado (placa o matrícula).
- Procesamiento de la imagen, se segmenta la imagen, se realiza la conversión a los canales RGB y se aplica la umbralización.
- Localización de la placa usando algoritmos de segmentación de imágenes a través de rectángulos.
- Reconocimiento de los caracteres de la imagen, mediante la librería Tesseract-OCR (Optical Character Recognition).

III. TRABAJO PREVIO

Para desarrollar la aplicación se usa un lenguaje de programación multiplataforma, conocido como Python, en su versión 2.7.0.

El objetivo es clasificar un vehículo mediante su matrícula o placa, el enunciado se plantea para realizar una aplicación para el reconocimiento de imágenes, se decide usar la librería de OpenCV para realizar los procesos de filtrados y procesar la imagen que nos darán como entrada para así poder mejorarla y poder llegar al objetivo deseado, que es obtener la parte de la placa para así hacer el proceso de reconocimiento de caracteres, para el reconocimiento o clasificación de caracteres, se decide usar la librería de Tesseract-OCR para el reconocimiento de caracteres, la función que usaremos tiene como entrada una imagen, la cual separa en bloques de texto, líneas y figuras, donde estos objetos se analizan delimitando líneas alrededor de los objetos sobre el **texto**.

IV. DESCRIPCIÓN DEL ALGORITMO

A continuación, explicaremos los pasos o etapas que se llevaron a cabo para la implementación del algoritmo, esto se hizo basado en las indicaciones del enunciado del proyecto.

1. El primer paso que se lleva a cabo es el de la obtención de las imágenes, para este ejercicio se usaron imágenes de vehículos donde fuera visible sus placas, este contenido fue descargado directo desde internet, se tomaron como ejemplo 5 imágenes para probar la aplicación, dicha aplicación decidió hacerse por comandos por consola, en nuestra aplicación lo que se hace es cargar la imagen que se encuentra en la carpeta de *resource* de nuestro fichero fuente.

Nombre	Fecha	Tipo	Tamaño	Etiquetas
output	22/06/2018 1:39 a. m.	Carpeta de archivos		
resource	22/06/2018 1:39 a. m.	Carpeta de archivos		
FinalProjectPDI.py	22/06/2018 1:39 a. m.	Archivo PY	4 KB	
Report.docx	22/06/2018 2:19 a. m.	Documento de Micro...	0 KB	

Figure 1. Carpeta resource.

Para cargar la imagen se usa la función *imread* de OpenCV y la asignamos a la variable *imgOriginal*, como se muestra a continuación.

```
#imagen original
imgOriginal = cv2.imread('resource/carro1.jpg')
cv2.imshow("Image", imgOriginal)
```

Figure 2. Carga de la imagen.

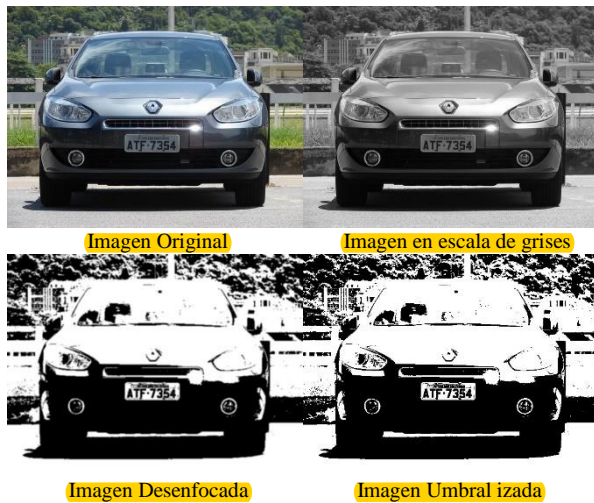
2. Luego de obtener o cargar la imagen, es necesario poder segmentar la parte de la cual necesitamos obtener información, es decir donde se hará el reconocimiento e interpretación de caracteres, en pocas palabras, obtener el área donde exactamente se encuentra la placa, pero antes de esto debemos obtener una imagen la cual pueda ser segmentada por esta región, ya que normalmente las imágenes vienen con ruido e imperfecciones las cuales podemos corregir usando filtros, entonces antes de poder obtener la imagen deseada debemos hacer un pre procesamiento de esta, para esto usamos 3 filtros, se hizo una conversión a escala de grises, **una umbralización (Threshold)** y **un filtro gaussiano**, respectivamente, a continuación se muestra como se implementó esto en el código fuente usando las funciones de OpenCV:

```
#escala de grises
imgScaleGray = cv2.cvtColor(imgOriginal, cv2.COLOR_BGR2GRAY)
#cv2.imshow("Image gray", imgScaleGray)

#threshold
ret, thresh = cv2.threshold(imgScaleGray, 90, 255, cv2.THRESH_BINARY)
#cv2.imshow("Image threshold", thresh)

#filtro gaussiano
gaussian = cv2.GaussianBlur(thresh, (5,5), 0)
#cv2.imshow("Image gaussian", gaussian)
```

Figure 3. Aplicación de filtros.



Luego de haber concluido con la etapa de pre procesamiento, sigue aplicar la segmentación, para así obtener la imagen de la placa.

- Al haber concluido la etapa de pre procesamiento de la imagen se pasa a la etapa de segmentación de la imagen, la segmentación consiste en poder particionar una imagen por regiones, basado en similitudes, proximidad y continuidad, para esto hacemos uso de **clasificadores**, ya que al tener una mejora de la imagen, gracias al pre procesamiento, usamos la función *findContours* de OpenCV, que permite encontrar patrones en una imagen y separarla por regiones, esta función se utiliza para encontrar contornos y bordes.

```
#contornos de la imagen
img, contours, hier = cv2.findContours(gaussian, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
```

Figure 4.Función find contours.



Figure 5. Imagen con sus contornos.

Pero como se puede apreciar con la función *drawContours* nos trae todos los contornos que posee la imagen, pero para nuestro caso estamos buscando una forma particular, un rectángulo para ser más exactos, para esto necesitamos decirle a la función *drawContours* que contornos quiero dibujar para que en este caso me formen un rectángulo, por ultimo usamos otras 2 funciones de la librería de OpenCV, *arcLength* y *approxPolyDP*.

Donde la función *arcLength* nos permite determinar si los contornos que hemos encontrado, forman un perímetro, es decir si es cerrado. Para la función *approxPolyDP* nos permite obtener una aproximación al contorno, es decir si al momento

de obtener el contorno no forma la figura que deseamos, esta función nos aproximara los contornos a la figura deseada. Para la segmentación de la imagen se hace un *for* basado en los contornos de la imagen a segmentar, pero para este ejercicio se validó que el perímetro tenga un valor constante pensando que el tamaño de la forma que buscamos (rectángulo) tiene un tamaño fijo, luego validamos que la cantidad de contornos que se unen para el perímetro sean 4 es decir los lados del rectángulo, se ubica la región donde se encuentra la placa y se dibujan los contornos para proceder a segmentar la imagen.



Figure 6.Región de la placa

Luego de haber encontrado la región en donde se encuentra la placa se procede a segmentar la imagen para que solo tengamos la parte de la imagen donde se encuentra la placa, y esta será guardada en carpeta output.



Figure 7. Región de la imagen que posee la placa.



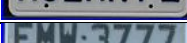


- Después de obtener la imagen, aplicar el pre procesamiento y segmentar la placa mediante los rectángulos, se obtiene una imagen procesada al punto de que puede ser llevada para la interpretación de los caracteres, y al usar la función *image_to_string* de la **librería Tesseract-OCR**. Esta imagen la obtenemos de nuestra carpeta output que se guardó previamente, luego llamamos a la función de la librería Tesseract, y usamos una función auxiliar para eliminar los caracteres que se crean por el ruido que aún queda en la imagen, para al final mostrar el resultado en una interfaz de Tkinter, o por consola.



Figure 8.Resultado final.

V. RESULTADOS EXPERIMENTALES

Al final de pasar por las cada una de las etapas anteriormente mencionadas se hace una tabla mostrando los resultados obtenidos con 5 imágenes escogidas para este ejercicio.

nombre	Imagen	Placa Real	Placa Obtenida
carro1		ATF 7354	ATF7354
carro2		MCLRN F1	MELRNF1
carro3		FMW3777	FMH3777
carro4		OJJ3984	UJVJEBBA
carro5		CA191932	CA19L93A

VI. CONCLUSIONES

En este trabajo podemos ver aplicados algunos de los conceptos vistos en clase como la aplicación de filtros, segmentación y algunas características acerca de las imágenes, como nos dice el enunciado se utiliza una librería para el reconocimiento y clasificación de la imagen según sus caracteres, como se ha dicho a lo largo de este documento se usa la librería de Tesseract-OCR la cual cómo se pueden apreciar en los resultados no es 100% correcta pero aun así logra determinar la mayor parte de los caracteres con exactitud.

Tal vez se podría aplicar algún otro filtro ya sea en la etapa de pre procesamiento o la de procesamiento para mejorar aún más la imagen y al momento de clasificarla o aplicar el algoritmo de reconocimiento tenga una mayor exactitud.

VII. BIBLIOGRAFIA

1. Using Tesseract OCR with Python:
(<https://www.pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/>)
2. Documentation tesseract-ocr/tesseract Wiki GitHub:
(<https://github.com/tesseract-ocr/tesseract/wiki/Documentation>)
3. OpenCV: OpenCV modules:
(<https://docs.opencv.org/3.4.1/index.html>)
4. Pytesseract PyPI:
(<https://pypi.org/project/pytesseract/>)

5. How to Python Convert Image to Text using OCR with Tesseract:
(<https://www.youtube.com/watch?v=LRXS3mC0OKo>)
6. License Plate Recognition with OpenCV 2: OCR License Plate Recognition:
(<https://www.youtube.com/watch?v=S67ofitVsws>)
7. License Plate Recognition with OpenCV 3: OCR License Plate Recognition:
(<https://www.youtube.com/watch?v=nmDiZGx5mqU>)
8. License Plate Recognition with OpenCV 4: Open Automatic License Plate Recognition (OpenALPR):
(<https://www.youtube.com/watch?v=oBAOdj8HABc&t=91s>)