Diseño de un sistema de visión artificial para el reconocimiento de placas en el software Scilab

Edwin Geovani Mendoza Jiménez
Escuela Superior de Cómputo,
Instituto Politécnico Nacional
Ciudad de México, México
mendozajimenezedwin22@outlook.com

Elmar Montiel Jiménez
Instituto Tecnológico Superior de Libres,
Tecnológico Nacional de México
Puebla, México
elmar.mj@libres.tecnm.mx

***Resumen*— La visión por computadora ha tenido gran prevalencia en el campo de la inteligencia artificial, por ello numerosas aplicaciones se han desarrollado utilizándola. En este trabajo se presenta un diseño para reconocimiento de placas en el software de Scilab haciendo uso de la librería de OpenCV. Se manejó un procesamiento de imágenes que consistió en pasar una imagen a su escala de grises, segmentarla por el método de Otsu, búsqueda de sus contornos y así localizar las coordenadas de la placa, se reconocieron sus caracteres mediante los contornos y se obtuvo de cada uno: área, perímetro y centroide. Con estos últimos datos, se entrenó una red neuronal y se mostraron resultados apropiados cuando se manejaron datos de una misma placa.**

***Abstract*—Computer vision has been highly prevalent in the field of artificial intelligence, which is why numerous applications have been developed using it. In this work, a design for plate recognition is presented in Scilab software using the OpenCV library. Image processing was handled that consisted of transferring an image to its gray scale, segmenting it by the Otsu method, searching for its contours and locating the coordinates of the plate, its characters were recognized through the contours and it was obtained from each one: area, perimeter, and centroid. With these last data, a neural network was trained and appropriate results were shown when data from the same plate were handled.**

Keywords*— *procesamiento de imagen, reconocimiento de placas, Scilab, visión artificial.

I. Introducción

La presencia de la Inteligencia Artificial (IA) ha destacado por las numerosas aplicaciones que pueden desarrollarse gracias a ella. La visión artificial, también llamada visión por computadora, es un campo de la IA que tiene una gran relevancia al querer simular la visión humana, se basa principalmente en la recuperación de información de imágenes a través de algoritmos matemáticos y con ello cumplir un fin en específico [1]. Algunas de sus aplicaciones pueden ser: modelado 3D, reconocimiento de huellas dactilares, biométrica, vigilancia, coches autónomos, reconocimiento de rostros, entre otros.

Existen diversas herramientas que permiten utilizar funciones de visión artificial, una librería de código abierto es OpenCV que cuenta con una gran cantidad de algoritmos optimizados que pueden ser utilizados para tareas de identificación y reconocimiento [2], puede implementarse en distintos lenguajes de programación y softwares, siendo uno de ellos Scilab que es un software para análisis y cálculo numérico bastante potente y de código abierto [3].

Al entrar en una tarea especifica de visión por computadora como lo es el reconocimiento de placas de automóviles, se involucran cuestiones de seguridad, vigilancia y control de acceso para instituciones o lugares que lo requieran. Con esta situación, contar con un sistema capaz de realizar dicha tarea puede ser de gran beneficio para las organizaciones e instituciones, ya que ahorraría tiempo y trabajo humano, además realizarlo en un software como Scilab podría brindar un funcionamiento adecuado al no necesitar de muchos recursos computacionales para trabajar de forma óptima.

II. Trabajos anteriores

En esta sección se muestran trabajos desarrollados para el reconocimiento de placas de automóviles que utilizan técnicas similares al diseño propuesto.

Sistemas comerciales de ALPR y ANPR

Los sistemas ALPR y ANPR, por sus siglas en ingles *Automatic License Plate Recognition* y *Automatic Number Plate Recognition* respectivamente, son tecnologías que existen en el mercado para la detección de placas de automóviles. Proporcionan funciones de buena complejidad y eficacia, ya que trabajan con reconocimiento óptico de caracteres. Son principalmente utilizadas para cuestiones de vigilancia y pueden trabajar en conjunto con otras aplicaciones o bases de datos para cumplir fines más específicos. Al tener un gran campo de aplicación existe una buena cantidad de alternativas y fabricantes que ofrecen este tipo de sistemas [4], [5].

Object Detection: Automatic License Plate Detection using Deep Learning and OpenCV [6].

En este trabajo se muestra el desarrollo de un sistema ALPR utilizando software libre como Python y OpenCV. El procedimiento consistió en la captura de la imagen a través de una cámara, se preprocesó con ayuda de OpenCV pasándola a su escala de grises, umbralizándola y obteniendo sus contornos. Para extraer la región en la que se encuentra la placa, se utilizó la relación de aspecto y densidad de los bordes. Finalmente, en Python se utilizó la librería TensorFlow que permitió realizar un mecanismo de Deep Learning que fue el Reconocimiento Óptico de Caracteres u OCR (*Optical Character Recognition*) para reconocer los caracteres de la placa.

License Plate Recognition System using OpenCV and PyTesseract [7].

Se propone un sistema alternativo a los sistemas ALPR con herramientas de bajo costo. Esta propuesta fue desarrollada en Python, con ayuda de las librerías externas de OpenCV y Tesseract. El procesamiento de la imagen tuvo como finalidad la obtención de la región de la placa, para ello se pasó la imagen a su escala de grises, se realizó un filtro de suavizado, se identificaron sus contornos, se seleccionaron aquellos de forma rectangular y se extrajo el referente a la placa. Para la identificación de los caracteres de la placa se realizó un OCR con ayuda de Tesseract dando a la salida un número de placa.

Automatic License Plate Recognition Using OpenCV and Neural Network [8].

Se realizó una implementación de reconocimiento de placas en el lenguaje de programación de Java e implementando funciones de la librería OpenCV. Se capturó la imagen de una cámara, se redimensionó dicha imagen y se localizó la región de la placa del automóvil a través de la densidad de los bordes y la relación de aspecto. Obteniendo la región de la placa, se segmentó cada caracter por el método de Otsu, se recuperó, y se reconoció con una red neuronal previamente entrenada, guardando la salida en una base de datos SQL.

III. METODOLOGÍA

El desarrollo del diseño propuesto consistió en tres etapas generales: adquisición de la imagen, procesamiento de imagen y el entrenamiento de una red neuronal. El diagrama a bloques se muestra en Fig. 1.

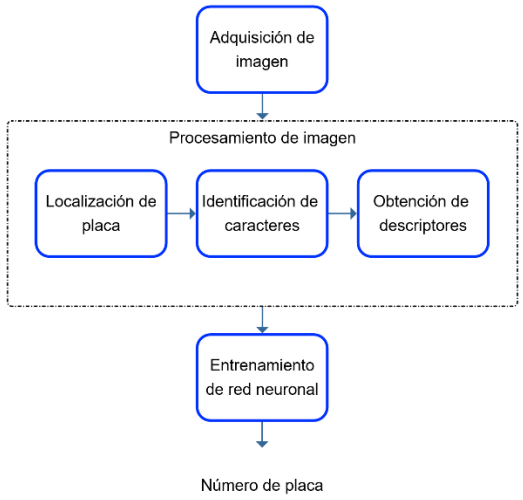

Fig. 1. Diagrama a bloques del diseño propuesto.

Se utilizó el toolbox de Scilab Computer Visión Module (*scicv*), para obtener las funciones e interfaces de la librería OpenCV en Scilab.

1. Adquisición de imagen.

La entrada al sistema propuesto consiste en una serie de imágenes de la parte trasera o delantera de un automóvil en donde se pueda apreciar su placa. Para ello se tomaron imágenes de un repositorio de Google con dichas características y fueron leías en Scilab con la función imread para su posterior tratamiento y procesamiento.

2. Procesamiento de imagen.

Este apartado está enfocado en las tareas de localización de la placa, identificación de caracteres y obtención de los descriptores de cada caracter.

- Localización de la placa.

Una vez leída la imagen, se redimensionó usando resize y se convirtió a su escala de grises (niveles de 0 a 255 por pixel) con la función cvtColor (véase Fig. 2).

Fig. 2. Imagen original e imagen en escala de grises.

Se utilizó con la función GaussianBlur un filtro de suavizado gaussiano para eliminar ruido de la imagen, y en seguida, con la función threshold se realizó una umbralización por el método de Otsu para que existiera un umbral específico para cada imagen a procesar y así obtener una imagen binaria (0 – negro, 255 - blanco), lo anterior se muestra en la Fig. 3.

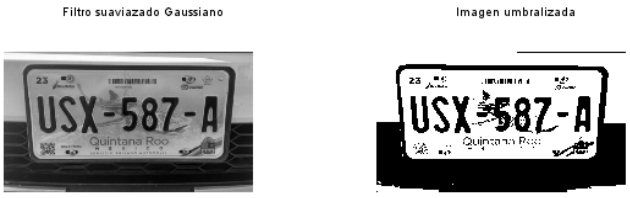

Fig. 3. Imagen con filtro gaussiano y umbralizada.

Finalmente, se realizó la búsqueda de los contornos de la imagen con la función canny y findContour. A cada uno se aplicó una aproximación de curva polinomial usando approxPolyDP, se identificaron aquellos que tuvieran aproximadamente 4 contornos o lados, y se rescató el de mayor longitud que correspondía al contorno de la placa (véase Fig. 4). De esta forma se obtuvo una imagen donde se visualizaba únicamente la placa del automóvil, como se muestra en Fig. 5.

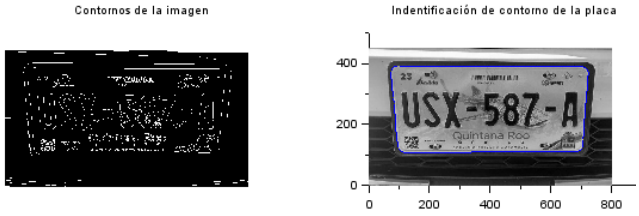

Fig. 4. Búsqueda de contornos e identificación de contorno de la placa.

Fig. 5. Resultado de la localización de la placa.

- Identificación de caracteres.

Con la placa identificada, se redimensionó y se realizó un procedimiento similar para identificar cada caracter. Primero, se hizo una umbralización por el método de Otsu, y se invirtió la imagen para que la información de interés correspondiera a pixeles blancos (véase Fig. 6).

Fig. 6. Imagen umbralizada e invertida.

Al observar la imagen, se visualiza que resaltan áreas que no corresponden a los caracteres, por lo que se aplicó un filtro de erosión y dilatación para mostrar lo más limpio posible cada caracter, estos fueron realizados con la función erode y dilate respectivamente (véase Fig. 7).

Fig. 7. Imagen erosionada y dilatada.

Al tener más limpia la imagen, el siguiente paso fue encontrar los contornos, excluir aquellos que eran de longitud muy pequeña y que se encontraban en los bordes de la imagen, de esta forma quedaron únicamente los contornos que identificaban a cada caracter de forma individual como se observa en Fig. 8 y Fig. 9.

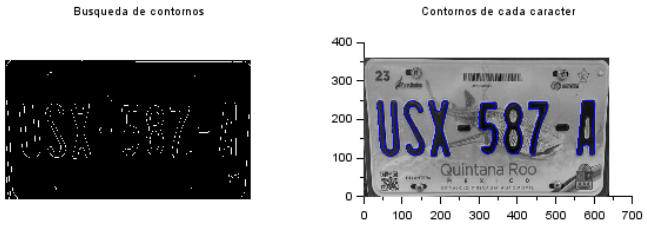

Fig. 8. Búsqueda de contornos e identificación de cada caracter.

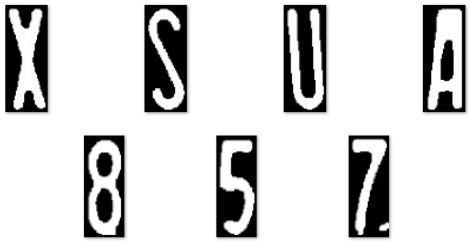

Fig. 9. Obtención de cada caracter individual.

- Obtención de descriptores.

Lo siguiente fue conocer las características que describieran cada caracter. Por cada imagen recortada se realizó una búsqueda de objetos binarios que contienen información de posible utilidad, conocidos como BLOB (*Binary Large Object*), a través de la identificación de los pixeles blancos mediante la conectividad 8. El resultado se puede observar en Fig. 10.

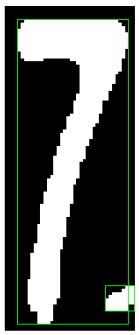

Fig. 10. Identificación de 2 BLOB en imagen de caracter 7.

Una vez identificados, se obtuvo el área de cada uno y se mantuvo el de mayor área, que correspondía al caracter. De este último, se guardaron los descriptores de área, coordenadas del rectángulo que forma el BLOB y centroide. El resultado final de este procesamiento se muestra en Fig. 11.

Fig. 11. Muestra de descriptores de caracter 7.

3. Entrenamiento de red neuronal.

Se realizó el proceso anterior con distintas placas y se guardaron en un archivo .csv los descriptores de distintos caracteres. Con ayuda del toolbox de Neural Network de Scilab, se entrenó una red neuronal. Primero se normalizaron todos los datos, se ocupó la función nn_onehot para codificar las salidas, la función initialize_parameters con las funciones de activación de rectificador (ann_relu) para las capas internas y distribución probabilística (ann_softmax) para la capa de salida, se entrenó con la función ann_train y se comprobaron los resultados con la función model_forward (propagación hacia adelante), esto se realizó con datos de entrada separando cada carácter y también por todos los caracteres de cada placa.

IV. Resultados

La localización de la placa e identificación de los caracteres con los que cuenta se realizó de forma adecuada, se obtuvo la totalidad de caracteres por cada placa de forma limpia y resaltada. Para placas que contenían cierto tipo de ruido, como logotipos, márgenes o dibujos, este proceso fue correcto, sin embargo, para imágenes con mala iluminación, inclinadas o borrosas la segmentación resultaba deficiente en unos casos y con ello una mala identificación de caracteres.

El entrenamiento de la red neuronal al realizarse con datos de entrada donde cada dato correspondía a un caracter (Por ejemplo, caracter A tiene salida 1, caracter B tiene salida 2), tuvo una aproximación cercana a la salida deseada pero no completamente correcta. Durante el entrenamiento se monitoreó el error de aprendizaje en cada época y en un inicio se tuvo un comportamiento adecuado, donde el error va disminuyendo, pero al llegar a cierta época, en específico la época 510, este error comienza a aumentar como se observa en Fig. 12. De igual forma, se puede apreciar este comportamiento en Fig. 13, donde se muestra gráficamente el valor del error desde la época 0 hasta la época 1000.

```
.Epoch 470 / 1000[.............>] loss: 1.083374
.Epoch 480 / 1000[.............>] loss: 1.082912
.Epoch 490 / 1000[.............>] loss: 1.082627
.Epoch 500 / 1000[.............>] loss: 1.082479
.Epoch 510 / 1000[.............>] loss: 1.082438
.Epoch 520 / 1000[.............>] loss: 1.082480
.Epoch 530 / 1000[.............>] loss: 1.082585
.Epoch 540 / 1000[.............>] loss: 1.082740
.Epoch 550 / 1000[.............>] loss: 1.082934
.Epoch 560 / 1000[.............>] loss: 1.083157
```

Fig. 12 Valores de error de aprendizaje entre época 470 y 560.

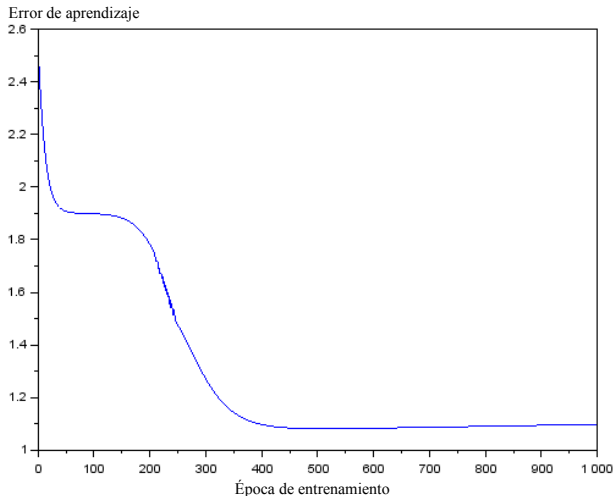

Fig. 13. Gráfica de error de aprendizaje donde cada dato de entrada corresponde a un caracter.

En Fig. 14 se observa la salida que tuvo la red neuronal comparada con los resultados correctos, y a pesar de que el error de aprendizaje fue un valor pequeño, no garantizó la totalidad de los caracteres reconocidos, teniendo un porcentaje de efectividad de 53.84%.

```
Valores de salida

  8.   12.   10.   4.   5.   3.   6.   4.   12.   4.   12.   4.   4.

Resultados correctos

  8.   12.   10.   4.   5.   1.   6.   9.   7.   11.   3.   2.   4.
```

Fig. 14. Salida de la red neuronal comparada con el resultado esperado.

La baja efectividad puede deberse a ciertas situaciones, como puede ser la falta de imágenes que recibe de entrada la red neuronal y la arquitectura de la misma. Si se requiere mejorar esta efectividad podría ser adecuado aumentar la cantidad de datos de entrada diversificando los caracteres con los que cuentan las placas de automóviles, así como también, probar con otra arquitectura de red neuronal a la que aquí se propone.

La siguiente prueba fue tomando los datos de entrada como la totalidad de caracteres correspondientes a una sola placa, o bien, una red neuronal por cada placa. Los resultados obtenidos fueron que el error de aprendizaje converge al pasar las épocas de entrenamiento cómo se visualiza gráficamente en Fig. 15. Así mismo, al ver los resultados de la salida en Fig. 16 estos fueron más cercanos a la salida deseada.

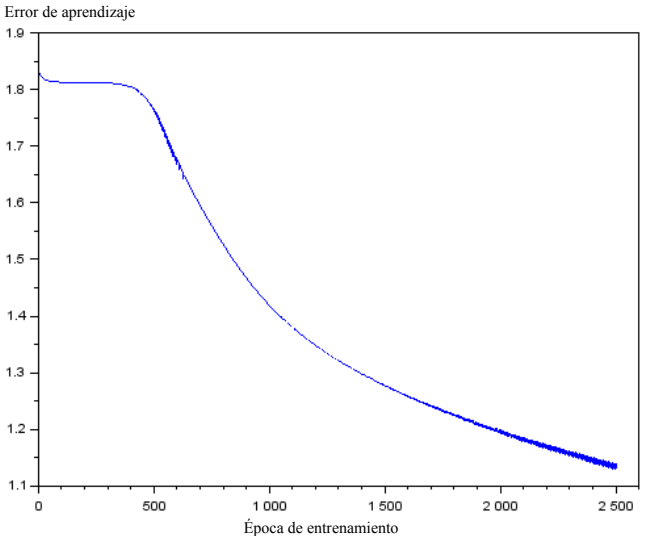

Fig. 15. Gráfica de error de aprendizaje de red neuronal para los datos de una sola placa.

```
Valores de salida de la red neuronal

  1.   2.   5.   4.   5.   6.

Resultados correctos

  1.   2.   3.   4.   5.   6.
```

Fig. 16. Salida de la red neuronal comparada con el resultado esperado para una sola placa.

Se realizó lo anterior con otras placas y se obtuvieron distintos resultados de efectividad como se observa en Tabla I. Algunas de las placas probadas contenían 6 o 7 caracteres, caracteres repetidos más de una vez y distintos tipos de iluminación, logotipos e inclinación.

TABLA I
PORCENTAJES DE EFECTIVIDAD PARA EL RECONOCIMIENTO DE DISTINTAS PLACAS

Imagen de placa	Identificación de contorno de sus caracteres	Porcentaje de efectividad en red neuronal
		83.33%
		71.42%
		85.71%
		57.14%
		57.14%
		57.14 %
		85.71%
		85.71%
		42.85%
		71.42%

En el diseño propuesto, el procesamiento y extracción de la placa y sus caracteres se realizó de manera adecuada, el inconveniente fue al momento de la detección a través de la red neuronal dónde la efectividad fue muy variante. En Tabla II se presenta una comparativa de la eficiencia del proyecto propuesto en comparación con otros trabajos relacionados, destacando el modelo utilizado para el reconocimiento del número de placa. Al comparar los modelos utilizados en otros trabajos de detección de placas, se observa que los algoritmos basados en Deep Learning como OCR o redes neuronales convolucionales tienen una alta efectividad, por lo que, cambiar el algoritmo de reconocimiento sería una posible mejora para este diseño.

TABLA II
PORCENTAJES DE EFECTIVIDAD DE DISTINTOS TRABAJOS

Referencia	Modelo para detección de número de placa	Porcentaje de efectividad
Diseño propuesto	Red Neuronal Backpropagation (Neural Network Scilab Toolboox)	78.73%
[6]	Attention OCR (TensorFlow - Python)	93%
[7]	OCR (Tesseract - Python)	72.5%
[7]	Support Vector Machine (SVM)	79.84%
[9]	Optimal K-Means (OCM) y redes neuronales convolucionales (CNN)	98%
[10]	CNN	94%
[11]	Redes neuronales convolucionales basadas en regiones (RCNN)	97.93%
[11]	CNN	91.67%

V. CONCLUSIONES

El utilizar técnicas de visión artificial puede ser bastante útil en la solución de diversas problemáticas, además, ha sido una tendencia la inclusión de esta tecnología en distintos campos. El desarrollo de este sistema ayuda como una guía para el desarrollo de un sistema más complejo, el detalle relevante es el entrenamiento de la red neuronal, ya que sería apropiado utilizar una mayor cantidad de datos y de esa forma tener un mejor desempeño del diseño propuesto, e inclusive, modificar el algoritmo de reconocimiento por alguno basado en Deep Learning. A su vez, se destaca la importancia de dar como entrada imágenes con una buena iluminación y resolución, ya que con esto se puede garantizar un mejor procesamiento de la misma, o bien, implementar otro tipo de algoritmos para admitir imágenes con ese tipo de ruido.

Finalmente, al contar con un sistema diseñado con software de código abierto puede ser muy útil para la creación de sistemas de bajo costo, además, al no utilizar grandes recursos computacionales, puede ser implementado en un sistema embebido y cumplir adecuadamente esta función de detección de placas de automóviles.

REFERENCIAS.

[1] D. Yin., R. Lopes., J. Shlens., E. Cubuk., J. Gilmer. “A fourier perspective on model robustness in computer visión,” 3rd Conference on Neural Information Processing Systems, 2019.

[2] OpenCV, <<About OpenCV,>> 2021, [Online]. Available: https://opencv.org/about/

[3] Scilab, <<About Scilab,>> 2021, [Online]. Available: https://www.scilab.org/about

[4] ALPR101, <<ALPR101 Home,>> 2021, [Online]. Available: http://alpr.com/espa%C3%B1ol.html

[5] MAV, <<Understanding ANPR,>> 2021, [Online]. Available: https://www.anprcameras.com/about-us/understanding-anpr/

[6] M. Vishal., D. Maram., K. Chaitanya., R. Angeline. “Object Detection: Automatic License Plate Detection using Deep Learning and OpenCV,” International Journal of Engineering and Advanced Technology (IJEAT), vol. 9, no. 1, p. 6022-6028, 2019.

[7] A. Chadha., S. Kashyap., M. Gupta., V. Kumar., “License Plate Recognition System using OpenCV & PyTesseract,” CSI Journal of Computing, vol. 3, no. 3, p. 31-35, 2020.

[8] S. Kumari., L. Gupta., P. Gupta. “Automatic license plate recognition using OpenCV and neural network,” International Journal of Computer Science Trends and and Technology (IJCST), vol. 5, no. 3, p. 114-118, 2017.

[9] I. Pustokhina., D. Pustokhin., J. Rodrigues., D. Gupta., A. Khanna., K. Shankar., C. Seo., G. Prasad. “Automatic Vehicle License Plate Recognition Using Optimal K-Means With Convolutional Neural Network for Intelligent Transportation Systems,” IEEE Access, vol. 8, p. 92907-92917, 2020.

[10] G. Goncalves., M. Diniz., R. Laroca., D. Menotti., W. Schwartz. ‘‘Real-time automatic license plate recognition through deep multi-task networks,’’ 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI), p. 110–117, 2018.

[11] W. Weihong., T. Jiaoyang. “Research on License Plate Recognition Algorithms Based on Deep Learning in Complex Environment,” IEEE Access, vol. 8, p. 91661-91675, 2020.