UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA YARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Modelo de reconocimiento automático y detección de matrículas basado en OpenCV y Machine Learning

Tesis para obtener el Título Ingeniero de Sistemas

Por:

Elias Ccoto Huallpa

Asesor:

Dr. Jorge Alejando Sánchez Garcés

Juliaca, julio de 2022

DECLARACIÓN JURADA DE AUTORÍA DEL INFORME DE TESIS

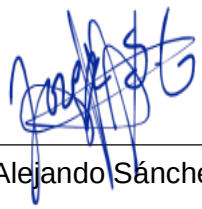
Dr. Jorge Alejando Sánchez Garcés, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que la presente investigación titulado: **"MODELO DE RECONOCIMIENTO AUTOMÁTICO Y DETECCIÓN DE MATRÍCULAS BASADO EN OPENCV Y MACHINE LEARNING"** constituye la memoria que presenta el Bachiller **Elias Ccoto Huallpa** para obtener el título de Profesional de Ingeniero de Sistemas, cuya tesis ha sido realizada en la Universidad Peruana Unión bajo mi dirección.

Las opiniones y declaraciones en este informe son de entera responsabilidad delautor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en la ciudad de Juliaca, a los 12 días del mes de noviembre de 2022.



Dr. Jorge Alejando SánchezGarcés
Asesor

161

ACTA DE SUSTENTACIÓN DE TESIS

En Puno, Juliaca, Villa Chullunquiani, a 04 día(s) del mes de Julio del año 2022 siendo las 14:00 horas, se reunieron en el Salón de Grados y Títulos de la Universidad Peruana Unión, Filial Juliaca, bajo la dirección del Señor Presidente del jurado: Msc. Benazir Francis Herrera Yucra, el secretario: Msc. Fredy Abel Huanca Torres y los demás miembros: Mg. Jorge Eddy Olazo Luque Mg. Abel Angel Sullon Macalupu y el asesor Dr. Jorge Alejandro Sanchez Garces con el propósito de administrar el acto académico de sustentación de la tesis titulada: Modelo de reconocimiento automático y detección de matriculas basado en OpenCV y Machine Learning

de el(los)/la(las) bachiller/es: a) Elias Ycoto Huallpa

b)

conducente a la obtención del título profesional de Ingeniero de Sistemas

(Nombre del Título Profesional)

con mención en

El Presidente inició el acto académico de sustentación invitando al (los)/a(la)(las) candidato(a)/s hacer uso del tiempo determinado para su exposición. Concluida la exposición, el Presidente invitó a los demás miembros del jurado a efectuar las preguntas, y aclaraciones pertinentes, las cuales fueron absueltas por el(los)/la(las) candidato(a)/s. Luego, se produjo un receso para las deliberaciones y la emisión del dictamen del jurado.

Posteriormente, el jurado procedió a dejar constancia escrita sobre la evaluación en la presente acta, con el dictamen siguiente:

Candidato (a): Elias Ycoto Huallpa

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	
Aprobado	18	A-	Muy Bueno	Sobresaliente

Candidato (b):

CALIFICACIÓN	ESCALAS			Mérito
	Vigesimal	Literal	Cualitativa	

(*) Ver parte posterior

Finalmente, el Presidente del jurado invitó al(los)/a(la)(las) candidato(a)/s a ponerse de pie, para recibir la evaluación final y concluir el acto académico de sustentación procediéndose a registrar las firmas respectivas.

Presidente

Secretario

Asesor

Miembro

Miembro

Candidato/a (a)

Candidato/a (b)

Índice General

Índice de Tablas

Índice de Figuras

Modelo de reconocimiento automático y detección de matrículas basado en OpenCV y Machine Learning

Automatic recognition and license plate detection model based on OpenCV and Machine Learning

Elias Ccoto Huallpa
Escuela Profesional de Ingeniería de Sistemas, Universidad Peruana Unión, Carretera Salida ArrequipaKm 6 Chullunquiani, Juliaca, Puno, Perú
elias.ch@upeu.edu.pe

Angel Abel Sullon Macalupu
Escuela Profesional de Ingeniría de Sistemas, Universidad Peruana Unión, Carretera Salida ArrequipaKm 6 Chullunquiani, Juliaca, Puno, Perú
angeli@upeu.edu.pe

Jorge Eddy Otazu Luque
Escuela Profesional de Ingeniría de Sistemas, UniversidadPeruana Unión, Carretera Salida ArrequipaKm 6 Chullunquiani, Juliaca, Puno, Perú
jorgeol@upeu.edu.pe

Jorge Sánchez-Garces
Escuela Profesional de Ingeniría de Sistemas, UniversidadPeruana Unión, Carretera Salida Arrequipa Km 6 Chullunquiani, Juliaca, Puno, Perú
jasg@upeu.edu.pe

Resumen

El reconocimiento automático de matrículas (ALPR) es una tarea importante con muchas aplicaciones en los sistemas inteligentes de transporte y vigilancia. Muchos de los sistemas de reconocimiento de matrículas automatizados existentes, solo funcionan en un entorno controlado donde las imágenes se capturan desde un ángulo recto con buena iluminación y claridad. Esta investigación presenta un modelo de procesamiento de imágenes para la detección y el reconocimiento de matrículas en Perú, que se puede manejar matrículas de fuentes ruidosas, con poca iluminación, en ángulo cruzado y no estándar. Este trabajo emplea varias técnicas de procesamiento de imágenes como, transformación morfológica, suavizado gaussiano y umbral gaussiano en la etapa de procesamiento. Una vez realizado el procesamiento de imagen se usa 3 algoritmos diferentes K-NN, SVM y Tesseract para el reconocimiento de caracteres, cada algoritmo con sus respectivos hiperparámetros para su optimización. Las imágenes fueron separadas en dos grupos, la primera en 80 imágenes tomadas de diferentes ángulos y distancia donde se obtuvo SVM con el mejor mode oiiiiiiiiiiiiiiiiiioiolo con un accuracy de 86% y en el segundo grupo con imágenes tomadas de un Angulo recto y distancia similar, en este grupo obtuvo un accuracy de 95.5%

***Palabras clave**: KNN; SVM; Tesseract; OpenCV; Machine Learning; hiperpárametros.*

Abstract

Automatic Number Plate Recognition (ALPR) is an important task with many applications in intelligent transportation and surveillance systems. Many of the existing automated license plate recognition systems only work in a controlled environment where images are captured from a right angle with good lighting and clarity. This research presents an image processing model for license plate detection and recognition in Peru, which can handle license plates from noisy, low-light, cross-angle, and non-standard sources. This work employs various image processing techniques such as morphological transformation, Gaussian smoothing and Gaussian thresholding in the processing stage. Once the image processing is done, 3 different algorithms K-NN, SVM and Tesseract are used for character recognition, each algorithm with its respective hyperparameters for its optimization. The images were separated into two groups, the first in 80 images taken from different angles and distances where SVM was obtained with the best model with an accuracy of 86% and in the second group with images taken from a right angle and similar distance, in this group obtained an accuracy of 95.5%

Keywords: KNN; SVM; Tesseract; OpenCV; Machine Learning y hiperpárametros.

I. Introducción

Con el aumento de la población, el número de vehículos en las carreteras aumenta según [1] hay más de 701 mil vehículos registrados en el Perú el año 2019 y hubo un aumento del 6.4 %, el 2018 lo que significa que aumentó los robos de e infracciones. Según [2] Instituto Nacional de Estadística e Información informa que, hasta el 2019 se registró más de 20000 denuncias de vehículos robados en todo el Perú con un promedio de crecimiento de 5 % anual, tan solo en Lima se registró 17000 denuncias, siendo el 85 % de las denuncias del país, la mayor parte de los vehículos recuperados fueron hallados abandonados, estos fueron abandonados después de cometer un delito como robos o saltos. Otro problema mencionado sería la clonación de placas. Más de 41000 vehículos en todo el Perú describen una alerta registral, que consiste en el duplicado de la placa sin el conocimiento del propietario [3]. Por lo cual, se hace muy necesario la identificación de los vehículos. A continuación, se describe la tabla 1 de los principales antecedentes usados en la investigación para justificar los vacíos.

A comparación de otras investigaciones descritas en la tabla 1, en esta investigación se ha utilizado 2 framework para hacer una comparativa y obtener un modelo OCR (optical character recognition) para la detección de placas. Estos frameworks utilizados son el Tesseract con su algoritmo LSTM(Long short-term memory) y el OpenCv con sus algoritmos tanto para el procesamiento de imágenes y de Machine Learnig; específicamente su algoritmo K-nn(K-Nearest Neighbors) y SVM(Support-vector machine); así mismo para hacer la validación del modelo de detección de placas se construyeron muestras para cada uno de los algoritmos, combinando hiperparámetros de entrada con el total de placas tomadas, para KNN se usó un hiperparámetros con 10 valores distinto , SVM 6 hiperparámetros obteniendo una combinación de 144 y para Tesseract un hiperparámetro con 13 valores distintos. Se hizo las pruebas para obtener el mejor hiperparámetro para el reconocimiento de caracteres con 80 imágenes tomadas de distintos ángulos y distancias obteniendo 800 resultados para K-NN, 11520 para SVM y 960 para Tesseract, obtenido el mejor hiperparámetro para cada algoritmo se hizo otra prueba con una muestra de 100 imágenes tomadas de un ángulo recto de y distancias que varían de 3 a 5 metros del cual se obtuvo el mejor accuracy de 95.5% determinando el mejor modelo. Por lo tanto, se vuelve absolutamente necesario identificar a los vehículos en accesos a principales centros comerciales tales como supermercados, universidades y zonas de aparcamiento para reducir los robos e infracciones de los conductores. Esta investigación pretende ayudar en la identificación de los vehículos mediante el reconocimiento de matrículas, mejorando el sistema de seguridad y así disminuir los robos, clonación y de vehículos infractores. Este modelo tendrá un aporte importante para las empresas de seguridad, para la policía, población en general y la economía del país.

Tabla 1. *Antecedentes de investigación*

	Técnica	Técnica de reconocimiento de placas	Resultados	Referencias
La implementación de ALPR (reconocimiento automático de matrículas)	OCR con CNN	Conversión a escala de grises, la dilatación de la escala de grises, procesamiento morfológico	Obtuvo un accuracy del 93 %	Agarwal et al. [4].
La implementación de ALPR (reconocimiento automático de matrículas) Robusta y con un tiempo de respuesta corta.	Keras, CNN.	Operación morfológica, Suavizado con gaussiano	Se redujo los errores en placas en colores, matices distintas, tuvo una mejor del 98%	Sathya et al. [5].
Implementación de ALP (Automaticlicense plate) al Norte de Iraq.	End-to-end CNN.	operaciones morfológicas.	Se segmentó adecuadamente las tres partes de las placas del Norte de Iraq, con un accuracy del 95%.	Farhat et al. [6].
Reducción de errores en imágenes de fuentes ruidosas, falta de iluminación y ángulos cruzados.	OpenCV y K-NN.	Transformaciones morfológicas, suavizado gaussiano.	El sistema ha detectado con éxito el 98% de las placas de matrícula con ruido y baja luminosidad.	Varma et al. [7].
Propone un marco para generar imágenes sintéticas de placas para luego entrenarlas	CNN.	Generación de imágenes de entrenamiento sintéticas.	Resulto tener una eficiencia del 98% con imágenes sintéticas para el entrenamiento de CNN.	Björklund et al. [8].
Propone proporcionar técnicas para la detección de caracteres que tengan mejores resultados.	OCR, Tesseract.	Texto impresos y manuscritos.	El modelo OCR propuesto tiene una precisión del 93% y funciona muy rápido con recuperación efectiva del texto que se ha pasado al modelo. Funciona de manera muy productiva para la literatura inglesa con una respuesta muy rápida.	Divya et al. [9].
Mejorar la localización de las placas en vehículos motorizados. .	No especifica.	GrabCut, escala de grises y binarizacion	Se logró una precisión del 99,8% para la localización de la placa.	Salau et al. [10].
Desarrollar un Sistema automático de reconocimiento de placas en Malasia.	Algoritmos de Support Vector Machine (SVM).	Algoritmo de suavizado de longitud de ejecución de clúster.	Los resultados alcanzados en la detección y reconocimiento de caracteres de un 80%.	Abdullah et al. [11].
El ajuste de hiperparámetros para la clasificación de huellas.	SVM.	Histogramas.	El ajuste de hiperparámetros muestra una mejora mediante el aumento en la puntuación de precisión hasta un 27,43%.	Chougule and Shah [12].
Estimación y predicción del estado de tráfico mediante reconocimiento de matrículas.	Modelo dynamic linear-Gaussian (DLG).	No especifica.	El modelo es aplicable en los sistemas ALPR.	Zhan et al. [13].
Detección y reconocimiento de matrículas en tiempo real.	Redes neuronales.	YOLO.	Tuvo un promedio alto en reconocimiento de matrículas teniendo problemas con las imágenes con ruido y baja luminosidad.	Silva and Jung [14].

II. MATERIALES Y MÉTODOS

A. Muestra

Se consideró dos grupos de muestras, para la primera muestra se utilizaron 80 imágenes con distinto ángulos y distancias. El propósito fue encontrar el mejor hiperparámetro para el modelo de predicción; a continuación, se detallan las muestras por cada algoritmo, teniendo un total de 800 datos para la prueba de KNN, 11520 datos para SVM y 900 para tesseract.

Tabla 2. *Muestras por algoritmo para la primera ejecución del modelo*

Frameword	Algoritmo	hiperparámetros	Cantidad hiperparámetros	Muestra Total
OpenCV	**KNN**	**K**	**10**	**80*10=80**
OpenCV	**SVM**	**C, Gamma, Kernel, Type1, degree y P**	**144**	**80*144=11520**
Tesseract	**LSMT**	**psm**	**13**	**80*13=960**

Para la segunda muestra se utilizaron 100 imágenes con un ángulo recto y con distancias de entre 3 a 5 metros. El propósito fue probar el mejor hiperparámetro para el modelo de predicción y encontrar el accuracy.

Tabla 3. *Valor de hiperparámetros por algoritmo para la primera ejecución del modelo*.

Frameword	Algoritmo	hiperparámetros	Muestra Total
OpenCV	**KNN**	**K= 4**	**100**
OpenCV	**SVM**	**C=1, Gamma=1, Kernel=lineal, Type1=SVM_C_SV C, degree=1 y P=0**	**100**
Tesseract	**LSMT**	**psm=9**	**100**

B. Metodología

La metodología propuesta que consta de tres fases principales: la primera por el procesamiento de imágenes, la segunda por la segmentación de placa y por último el reconocimiento de caracteres muestra en la figura 1

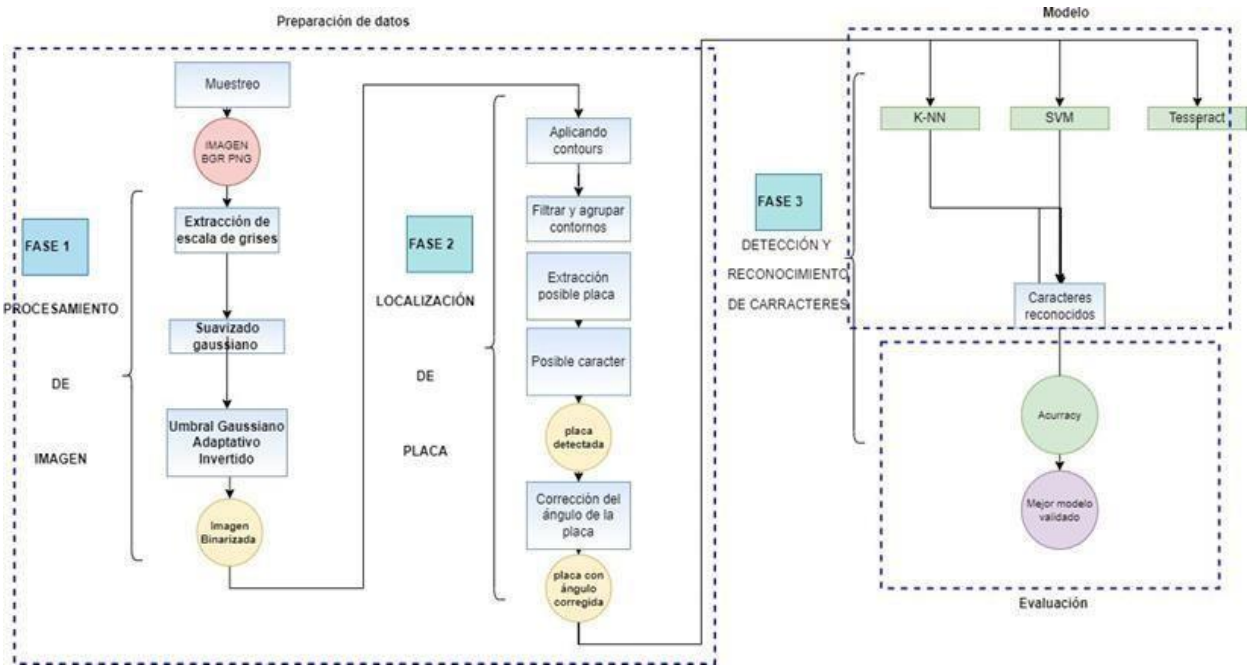


Figura 1. Metodología de la investigación

C. Nivel de investigación

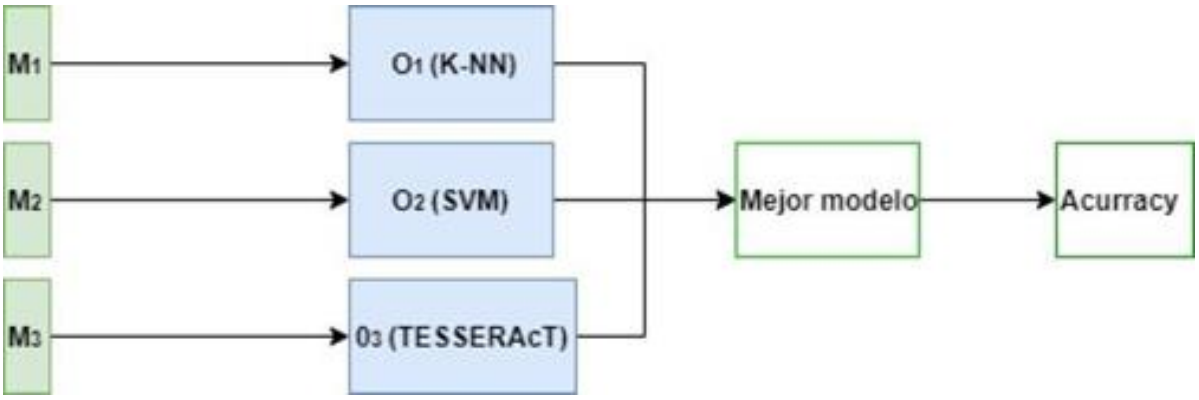


Figura 2. Nivel de Investigación

Donde:

M = Muestra

O = Observación

La investigación es cuantitativa de nivel descriptiva comparativo porque se realizaron análisis comparativos entre algoritmos computacionales determinando el algoritmo prevalente en función al mejor accuracy así obteniendo el mejor modelo computacional.

D. Ejecución

Procesamiento de imagen

Para el procesamiento y tratamiento de imágenes se usa las herramientas de OpenCv(The Open Computer Vision). Según [15], la librería OpenCV proporciona un marco de trabajo de alto nivel para el desarrollo de aplicaciones de visión por computador en tiempo real: estructuras de datos, procesamiento y análisis de imágenes, análisis estructural, etc.

La entrada puede ser una imagen RGB, en este caso utilizamos imágenes en formato PNG, antes empezar con la detección de la placa, la fuente de la imagen debe adaptarse para su posterior procesamiento. La figura 3 es la imagen de entrada de muestra utilizada para mostrar el proceso. El siguiente es el orden en el que se aplican las técnicas de procesamiento de imágenes.

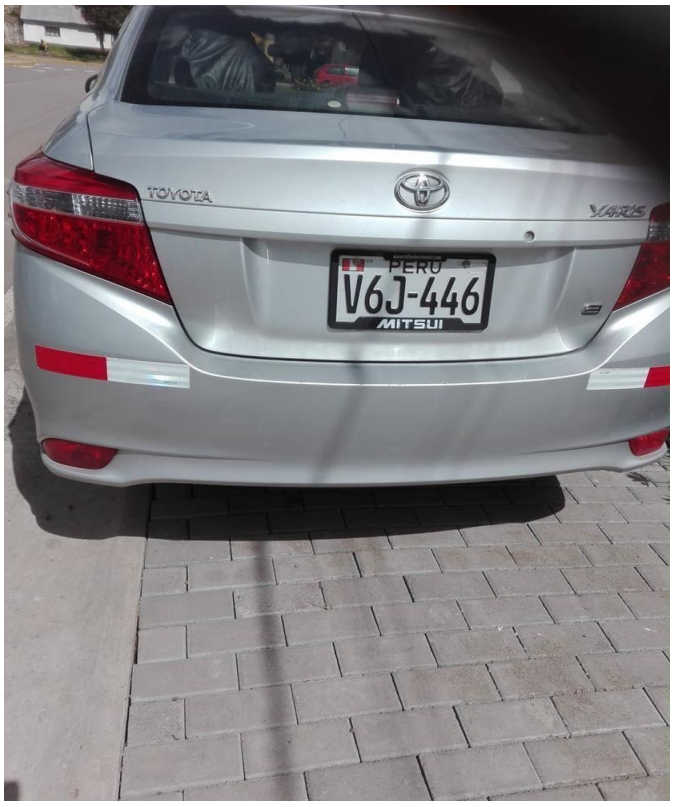


Figura 3. Vehículo a detectar

1) *Extracción de escala de grises*: A la imagen cargada se procede a realizar la escala de grises, así como se muestra en la figura 4.

imgGrayscale. shape

Figura 4. Aplicando escala de Grises

2) *Suavizado gaussiano:* Es un filtro de suavizado lineal que puede eliminar el ruido gaussiano y se usa ampliamente en el proceso de eliminación de ruido del procesamiento de imágenes. La operación especifica del filtrado gaussiano es usar una plantilla para escanear cada pixel en la imagen y usar la escala de grises promedio ponderada de los pixeles en la zona determinada por la plantilla para reemplazar el valor del pixel central de la plantilla, [16].

El filtrado gaussiano o suavizado gaussiano utiliza una función gaussiana lineal. El objetivo del filtrado gaussiano es reducir el ruido y los detalles. Esto servirá para otros pasos de procesamiento de imágenes. La aplicación de un filtro gaussiano a una imagen tiene la ventaja adicional de evitar los artefactos de alias. Matemáticamente, una función gaussiana unidimensional se puede expresar como: [11]

$$G(x) = \frac{1}{\sqrt{2\pi\theta^2}} e^{\frac{-x^2}{2\theta^2}}$$

En OpenCV, el suavizado gaussiano se puede aplicar usando la siguiente función:

Cv2. GaussianBluear(image. (5, 5), 0)

Aquí, el segundo argumento (5, 5) es el tamaño del kernel gaussiano, cuanto mayor es el tamaño, mayor es la intensidad de suavizado.

3) *Umbral Gaussiano Adaptativo Invertido:* El propósito de la creación de umbrales, como parte de la segmentación de imágenes, es crear una imágen binaria a partir de una imágen en escala de grises. Este proceso se conoce como binarización de imágenes. Sin embargo, este metodo puede no ser adecuado en condiciones en las que la iluminación no sea uniforme. Se toma una ventana de tamaño predefinido y se encuentra una suma ponderada de pixeles vecinos para realizar el umbral adaptativo, [17]. Matemáticamente, cada pixel en la imagen del umbral de salida se puede calcular como:

$$output\_image(x,y) = \begin{cases} 0, input\_image(X,y) > T(x,Y) \\ 255, otherwise \end{cases}$$

Donde, T (x, y) es una función de umbral que calcula el umbral individualmente para cada pixel. OpenCV proporciona una función útil para realizar el umbral Gaussiano adaptativo, [13].

cv2.adaptiveThreshold(image,255,
cv2.ADADTIVETHRESHGAUSSIANC,
cv2.THRESHB INARYINV,
cBLOCKS IZE, WEIGHT)

Aquí:

- ADADTIVE_THRESH_GAUSSIAN_C El umbral será en función de Gaussian;
- BINARY_INV donde se binariza y se invierte la binarización;
- BLOCK_SIZE es el tamaño de la ventana de umbral.
- WEIGHT se usa para calcular la suma ponderada de los valores en la zona.

Detección de placa

Al final de la etapa anterior de procesamiento de imágenes, Umbral Gaussiano Adaptativo Invertido, devuelve una imagen binarizada, con valores de 0 o 255. La imagen binarizada servirá como entrada para las etapas posteriores en las fases de detección y reconocimiento.

Figura 5. Imagen binarizada

1) *Aplicando contours:* El seguimiento de contorno, es el algoritmo utilizado para generar contornos. Un contorno es un vínculo de puntos de igual intensidad a lo largo del límite, [17]. En OpenCV, encontrar contornos es como encontrar un objeto blanco del negro fondo, por lo tanto, durante la etapa de Umbral Gaussiano Adaptativo, se tuvo que aplicar la operación de Inversión. La figura 6 muestra el resultado de aplicar contornos a una imagen binarizada.

cv2. drawContours()

Figura 6. Aplicando Contours

2) *Filtrar y agrupar contornos:* Para regiones pequeñas, especialmente bordes afilados y valores atípicos de ruido, se aplican contornos. Así, el ojo humano puede calcular fácilmente que tales contornos son innecesarios, pero esto debe ser incorporado en el algoritmo, [18]. Inicialmente, se aplicaron cuadros delimitadores a cada contorno. Luego, para cada contorno, se consideraron los siguientes factores, como el área mínima del contorno, el ancho y la altura mínimos del contorno, las relaciones de aspecto mínima y máxima posibles.

cv2.findContours(imgThreshCopycv2.RETR_LISTcv2.HAIN_APPROX_SIMPLE)

Esto resultó en el filtrado de la mayoría de los contornos innecesarios, acercándonos a nuestro objetivo: detectar una placa. La segunda etapa del filtrado consta en comparar cada contorno con cualquier otro contorno en parámetros como la distancia entre los contornos. La tercera etapa es agrupar los contornos, un grupo de contornos satisface todas estas condiciones, se agrupan en uno. Es posible que se puedan obtener dos o más de tales grupos.

3) *Extracción de posible placa*: En esta etapa nos enfocamos en la detección de la posible placa. Para ello usamos una función de OpenCV esta función nos ayudará a la detección de un rectángulo cerrado.

cv2. boundingRect(self. contour)

cv2.boundingRect(selfcontour) ayuda a detectar todos los rectángulos que se encuentre en nuestra imagen de prueba, en función contour que fue trabajado anteriormente; esta función nos devolverá: coordenada X, coordenada Y, ancho y altura.

$$\begin{gathered} \text{IntboundingRectX} = \text{intX} \\ \text{IntboundingRectY} = \text{intY} \\ \text{IntboundingRectWidwt} = \text{intWidwt} \\ \text{IntboundingRectHeight} = \text{intHeight} \end{gathered}$$

Figura 7. Rectángulo detectado

Una vez obtenida las coordenadas y medidas se procederán a extraer la posible placa recortando y delimitando del resto de la imagen.

Figura 8. Placa extraída

4) *Posibles caracteres:* Se presenta otro reto el de identificar la placa correcta de los n resultados que nos proporciona la función anterior. En esta etapa se ubica la placa correcta, para esto detectaremos los posibles caracteres de cada rectángulo, ya que el que contenga más posibles caracteres será la placa correcta para esto necesitamos de la siguiente función.

```
Cv2.findContours(imgThresehCopyCv2.RETRLIST.
cv2.CHAIN_APPROX_SIMPLE)
```

La función findContours mostrar contornos de la placa, para estado de dos parámetros: la primera Cv2.RETR_LIST ayuda a solo mostrar los contornos que no sean dependientes de otros; el segundo parámetro que usamos es cv2.CHAIN_APPROX_SIMPLE que mostrará los puntos finales del contorno, así poder segmentarlo y contar los caracteres; una vez hecho el conteo tendremos la placa correcta.

Figura 9. Caracteres detectados

5) *Corrección del ángulo de la placa:* En esta etapa, se aplica un cuadro delimitador a cada placa. Si alguna de las placas sufre distorsión de ángulo, se realiza una transformación de su ángulo. El proposito de la transformación es un mapeo entre dos espacios, que se usa para conservar puntos y líneas. Después de la transformación, la línea paralela conserva su paralelismo. Se mantiene la relación de longitudes entre los puntos que residen en una línea recta. Sin embargo, los ángulos dentro de las líneas y las longitudes dentro de los puntos no se conservan. En OpenCV, la corrección del ángulo de la placa se puede lograr mediante la función getRotationMatrix2D.

```
Cv2. getRotationMatrix2D(tupleCenter)
fltCorrectionAnglenDeg 1.0)
```

Donde este devuelve una matriz:

$$\begin{bmatrix} \alpha & \beta & (1-\alpha)*x-\beta*y \\ -\beta & \alpha & \beta*x+(1-\alpha)*y \end{bmatrix}$$

Dónde, α = escala ∗ cos(ngulo), β = escala

X es la coordenada X del centro de la placa, Y es la coordenada Y del centro de la placa.

Esta matriz es usada como entrada a warpAffine de la siguiente manera:

Cv2. warpAffine(imgOriginal totationMatrix (width heigth)

Donde ha sido corregida la imagen.

Reconocimiento de caracteres

Para el reconocimiento de caracteres se realizan varios subprocesos donde se aplican los algoritmos de OpenCv y los de Tesseract.

1) *Transformación de características y predicción:* Una vez que se eliminan los caracteres superpuestos, se cambia el tamaño de cada contorno, que representa un carácter en la placa o un número en una imagen de 20x30. Esta operación se realiza para garantizar la coherencia con el formato de entrada del modelo de aprendizaje, donde cada carácter de fuente que se alimenta durante el proceso de entrenamiento es una imagen redimensionada de 20x30. Una vez que la imagen redimensionada se envía al modelo.

$$\text{RISIZED\_CHAR\_IMAGE\_WIDTH} = 20$$
$$\text{RISIZED\_CHAR\_IMAGE\_HEIGHT} = 30$$

2) *K-NN:* Según [19] el algoritmo más simple de la técnica de aprendizaje automático es k-NN. Viene bajo la metodología de aprendizaje supervisado. La clasificación de los caracteres se realiza mediante el cálculo de la distancia que se mide implementando la distancia euclidiana como se muestra en la ecuación.

$$D(a,b) = \sqrt{\sum_{i=1}^{n}(b_i - a_i)^2}$$

Donde "a" y "b" son dos puntos en un espacio n- dimensional representado por:

$$a = (a_1, a_2, a_3, \dots, a_b)$$

Para extraer los mejores hiperparámetros posibles, para el modelo, se utilizó un testeo manual. Buscando un hiperparámetro que, de un mejor resultado para el reconocimiento de caracteres, se utilizó 80 imágenes donde se consideró el mejor acurracy para el hiperparámetro utilizado.

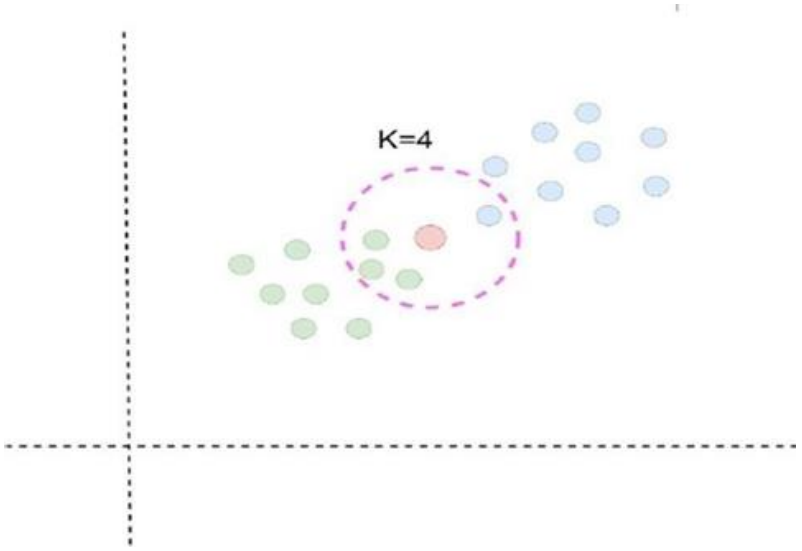


Figura 10. Knn con hiperparámetro de K=4

Para la implementación de este modelo se utilizó scikit- learn. Scikit-learn proporciona KNeighborsClassifier, donde k es un número entero. La ventaja de un valor de k mayor es que se suprime el ruido, pero los límites de los caracteres se volverán menos distinguibles. La búsqueda aleatoria toma como entrada el modelo y una lista de hiperparámetros que se prueban durante el entrenamiento. Para este proceso, se probaron diferentes valores positivos para los parámetros "K", siendo los valores de "K" del "1" al "10" así logrando el mejor valor para "K" para este modelo.

3) *SVM:* Support Vector Machine es un método de aprendizaje supervisado para el análisis de clasificación y regresión. [20]. En SVM, los datos se representan en un espacio n-dimensional donde se puede predecir si un nuevo ejemplo de entrenamiento cae en la misma categoría o en otra diferente. El objetivo principal de SVM es encontrar un hiperplano en el espacio n-dimensional que pueda clasificar los puntos de datos. [21] Se podrian elegir varios hiperplanos potenciales para distinguir las dos clases de puntos de datos. Pero el hiperplano ideal es el que maximiza el margen entre los puntos de datos de ambas clases. Los hiperplanos son límites para la toma de decisiones que ayudan a

distinguir los puntos de datos. Los puntos de datos que caen a ambos lados del hiperplano pueden atribuirse a varias clases. La dimensión del hiperplano depende del número de características. El hiperplano se puede encontrar usando la siguiente ecuación.

$$\vec{w} * \vec{x} + b = 0$$

Donde w es el vector normal al hiperplano y es el conjunto de puntos.

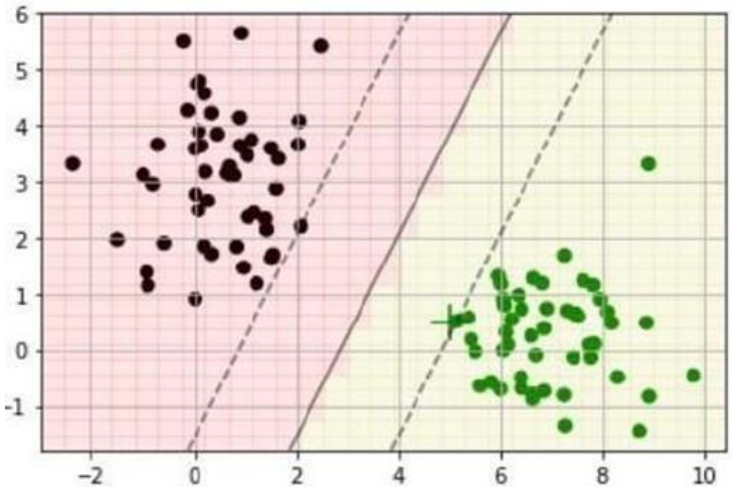


Figura 11 SVM con hiperparámetro C=1

Para hallar la mejor predicción para SVM se utilizaron varias combinaciones de hiperparámetros siendo: Kernel, gamma, C, type1, degree y P obteniendo 144 de posibles combinaciones el valor de estos hiperparámetros se detalla en la tabla 4.

Tabla 4. *Valores de los hiperparámetrosde SVM*

#	Hiperparámetros	Valores
1	C	1, 10,100, 1000 y 0.1
2	Gamma	0.001, 0.0001, 0.1, 0.01 y 1
3	Kernel	Lineal, Polynomial, Sigmoid, RBFy CHI2
4	Type	EPS_SVR, C_SVC, NU_SVC,ONE_CLASS y UN_SVR
5	Degree	1, 2 y 3
6	P	0 y 0.1

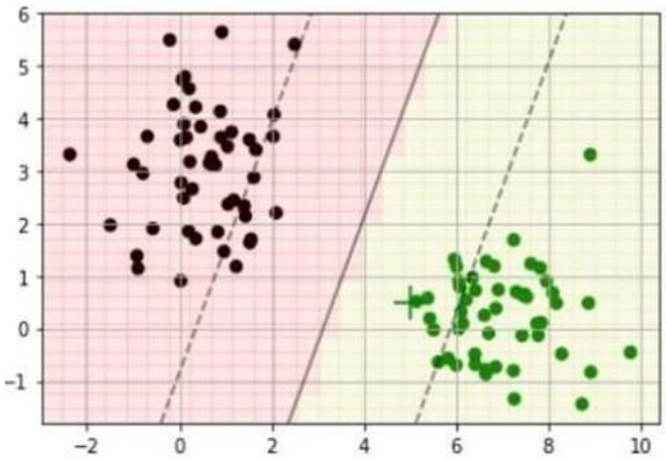


Figura 12. SVM con hiperparámetro C=0.01

4) *TESSERAC:* Tesseract es un motor OCR que es un código abierto que es lo más usados entre los ingenieros de OCR. [9]. Tesseract es un motor de búsqueda OCR potente que tiene la capacidad de ver más de 100 dialectos. Es muy posible que este educado para percibir diferentes dialectos. La variante más reciente (basada en LSTM) es estable cargada el 26 de diciembre de 2019.

   Se usó el parámetro de PSMque hace referencia a los modos de segmentación de las páginas (page segmentation modes, en inglés) de la librería Pytesseract. Cada número hace referencia a un modo de segmentación:

   0: Orientación y detección de script (OSD) únicamente.

   1: Segmentación automática de páginas con OSD.

   2: Segmentación automática de páginas sin OSD ni OCR.

   3: Segmentación completamente automática de páginas sin OSD.

   4: Supone una única columna de texto de tamaños variables.

   5: Supone un único bloque uniforme de texto alineado de forma vertical.

   6: Asume un único bloque uniforme de texto.

   7: Trata la imagen como una única línea de texto.

   8: Trata la imagen como una única palabra.

   9: Trata la imagen como una única palabra dentro de un círculo.

   10: Trata la imagen como un único carácter.

   11: Buscador de texto disperso. Encontrar la mayor cantidad de texto posible sin un orden en particular.

   12: Buscador de texto disperso con OSD.

   13: Trata el texto como una única línea, sin utilizar métodos específicos de Tesseract.

Estos parámetros se aplicaron en las pruebas para obtener el mejor modelo mediante el acurracy para la detección y reconocimiento de placas.

III. RESULTADOS

En esta sección, se presenta los resultados experimentales de la evaluación del proceso de localización de matrículas y, además, se presenta una comparación de los algoritmos SVM. KNN y el algoritmo de Tesseract, la finalidad es encontrar un modelo óptimo de reconocimiento. Para cada algoritmo se usó el mismo procesamiento de imagen utilizando las herramientas de OpenCV obteniendo una imagen segmentada y binarizada, esta imagen fue tomada como input para cada algoritmo, así como se muestra en la figura 9.

A. PRIMERA EJECUCIÓN DEL MODELO

KNN

Para este algoritmo se consideró un hiperparámetro con 10 diferentes valores, siendo "K vecinos" el hiperparámero y los valores considerados fueron positivos del 1 al 10, dando como resultados el hiperpaametro con valor de 4 con el mejor un accuracy de 0,841 tal como se detalla en la tabla 5.

Tabla 5. *Acurracy de los hiperparámetros de KNN.*

#	K	Accuracy
1	K4	0,841
2	K3	0,818
3	K1	0,807
4	K5	0,798
5	K2	0,782
6	K7	0,763
7	K6	0,760
8	K8	0,751
9	K9	0,747
10	K10	0,733

En las pruebas k-vecinos resulto tener el valor de 4, este resultado fue eficiente en el reconocimiento de caracteres,tiene sentido ya que cuando K es más cercano al valor de 1, la predicción puede ser más precisa, pero a la vez tiene más probabilidad de confundir caracteres similares como por ejemplo la "0" con la "O" esto porque no tiene más K-vecinos para validar

su predicción; al contrario, cuando K es más alejado de 1 tiende a errar la predicción ya que tiene demasiados K-vecinos que confunde al algoritmo, esto demostrado en los resultados detallados en la tabla 3; por lo cual el valor óptimo para K fue 4 siendo este un valor cercano a 1 y alejado del 10, teniendo los K-vecinos suficientes para validar su predicción como se puede ver en la figura 11.

SVM

Para Support-vector machine se usaron 6 distintos hiperparametros con distintos valores como se muestra en la tabla 4, obteniendo 144 la combinación de hiperparametros donde dio como resultado 11520 muestras para la evaluación. Realizando las pruebas se obtuvo la combinación más optima con los siguientes valores: C = 1, Gamma = 1, Kernel = Lineal, Type1 = C_SVC, Degree = 1 y P = 0 teniendo un accuracy de 0,864 como se detalla en la tabla 6.

Tabla 6. *Acurracy de los hiperparámetros de SVM***.**

#	C	gamma	kernel	type1	degree	P	Accuracy
1	1	1	Lineal	C_SVC	1	0	0,864
2	1	0,001	Lineal	C_SVC	3	0	0,844
3	100	1	Lineal	C_SVC	1	0	0,842
4	1	0,001	Lineal	C_SVC	1	0	0,842
5	1	1	Lineal	C_SVC	2	0	0,841
6	100	1	Lineal	C_SVC	2	0	0,840
7	100	1	Lineal	C_SVC	3	0	0,840
8	10	0,0001	Lineal	C_SVC	1	0	0,840
9	10	0,0001	Lineal	C_SVC	2	0	0,840

En el resultado el Kernel fue lineal, siendo este másrápido y simple a la hora de mapeo de muestras, también obtuvimos como resultados que “type1” fue C_SVC, este hiperparámetro trabaja adecuadamente con un Kernel Lineal y depende de los valores del hiperparámetro “C”; si “C” es cercano a cero no se consideraran puntos cercanos del núcleo en el mapeo así como se muestra en la figura 10 y cuanto más alejado de cero los puntos más cercanos serán considerados

en el mapeo se ve en la figura 9; El hiperparámetro “C” nos dio como valor 1 siendo el óptimo para nuestro modelo ya que este no cuenta con muchas penalizaciones en el mapeo siendo más preciso en los resultados; en cambio el hiperparámetro Gamma , P, Degree no trabaja con un Kernel Lineal por lo cual en nuestro modelo no afectará sus valores.

TESSERACT

Para Tesseract se consideraron su un hiperparametro (psm), siendo sus valores números enteros positivos del 1 al 13, donde se obtuvo 1040 muestras para su evaluación. Donde como resultado del hiperparametro psm fue valor de 9 con un accuracy de 0.479 siendo el más óptimo para este frameword así como se puede ver en la tabla 7.

Tabla 7. *Acurracy del hiperparámetros de TESSERACT*

#	psm	Accuracy
1	9	0,479
2	12	0,471
3	7	0,470
4	5	0,467
5	6	0,466
6	8	0,431
7	13	0,376
8	11	0,371
9	10	0,346
10	3	0,249
11	1	0,236
12	2	0,234
13	4	0,000

En esta prueba PSM tuvo un valor de 9 como es más óptimo, este PSM considera a la imagen extraída como una sola palabra en un círculo, este parámetro fue más preciso para el reconocimiento de caracteres, pero no tan óptimo como para conseguir un accuracy alto, esto porque Tesseract fue desarrollado para detección de textos, en una imagen de una placa se encuentra varias distracciones para este algoritmo dándonos como resultados datos erróneos, esto se detalla en los resultados de la tabla 7.

B. SEGUNDA EJECUCIÓN DEL MODELO

Obtenida los hiperparámetros adecuados para el OCR de cada algoritmo, se realiza otra prueba con las imágenes que fueron capturadas de un mismo ángulo(recto) y una distancia de 3 a 5 metros obteniendo un nuevo accuracy para cada algoritmo como se detalla en la tabla 8.

Tabla 8. *Accuracy de los mejores hiperparámetros*

Algoritmo	Muestras	Accuracy
SVM	100 con un solo ángulo y distancia	0,955
K-NN	100 con un solo ángulo y distancia	0,863
Tesseract	100 con un solo ángulo y distancia	0,292

Como se ve en la tabla los resultados variaron especialmente con el algoritmo de SVM donde se obtuvo un 95.5% en el reconocimiento de caracteres a comparación con imágenes de ángulos distintos de fue de 86.4%, SVM tuvo una mayor precisión a la hora de distinguir caracteres similares(0,O, I, 1, M, V, L,), en cambio KNN tuvo más confusión con esos caracteres, aun así hubo un aumento al 86.3% pero, en Tesseract se observa una disminución en su accuracy, esto debido a que este algoritmo detectó caracteres especiales o basura(!,",, $, %, , /,(,),=,?, !) , porque los algoritmos de Tesseract fueron desarrollados para la detección de manuscritos o textos en superficies homogéneos. Los algoritmos de KNN y SVM tuvieron un aumento con las muestras que tenga un ángulo y distancias similares.

Tabla 9. *Comparación de algoritmos*

Algoritmo	Hiperpárametro	Accuracy	Observaciónes
SVM	C=1, Gamma=1, Kernel=lineal, Type1=SVM_CSVC; degree =1y P = 0	95.5%	Este algoritmo tuvo mejor resultado en caracteres confusos y similares como son: 0, O, L, I,1, J entre otros. Por lo cual este algoritmo obtuvo un accuracy de 95.5% y siendo el mejor para ser usado por este modelo.
KNN	K=4	86.3%	El algoritmo de KNN tuvo algunas dificultades a la hora de reconocer y clasificar caracteres similares como por ejemplo la 0 con la O obteniendo un accuracy de 86.3% esto depende mucho del ruido y distancia donde se capture la imagen.
Tesseract	psm=9	29.2%	Tesseract tuvo muchos problemas con los pequeños ruidos que tiene las placas, ya que cualquier pequeña mancha o caracteres no claros de las placas confunde con caracteres especiales como son ¨$.+, *, -¨ , entre otros por lo cual se obtuvo un accuracy de 29.2%

IV. DISCUSIONES

Según los resultados mostrados se halló el mejor modelo para el reconocimiento de placas, este modelo fue con el algoritmo de SVM, también se halló los mejores hiperparámetros, siendo el Kernel lineal como el más óptimo para este modelo, según [20] el Kernel lineal es uno de los más sencillos, pero, el más rápido a la hora del mapeo de muestras cuando se tiene una gran cantidad como en las pruebas realizadas, también menciona que kernel lineal es utilizado cuando las muestras son linealmente separables siendo el caso de las muestras utilizadas. También los resultados nos muestran que el hiperparámetro C tuvo un valor de 1, el uso de este valor del hiperparámetro fue comprobado por [12] obteniendo un acierto de 98% en la clasificación de huellas dactilares; por otra parte otro factor que influye según[10] es la forma de captura de las muestras ya que influye en gran manera la ubicación de la cámara, de la misma manera [22] indica que otro factor que influye es situación en que se toma las fotos, esto podría ser el clima(lluvias, nevada, neblina).

V. CONCLUSIONES

Como seres humanos necesitamos un alto grado de precisión cuando realizamos la tarea de reconocimiento de las placas de los vehículos en una carretera muy transitada o en un área de estacionamiento que puede no ser posible manualmente, al no ser identificado un vehículo robado o infractor influye en la seguridad, en la economía del país [23]. Para evitar este problema, los investigadores de todo el mundo han simulado muchos intentos durante los últimos años para la detección de placas en vehículos, hallando dificultados en el reconocimiento en condiciones no óptimas como: luminosidad no adecuada, placas dañadas o caracteres similares [24].

En esta investigación se ha realizado un intento similar para desarrollar un modelo de reconocimiento de matrículas específico y automático, pero tomando tres algoritmos distintos con sus respectivos hiperparámetros donde se obtuvo el mejor algoritmo e hiperparámetro. La precisión del modelo propuesto no es del 100% en general, pero aun así esta investigación puede ser muy útil para la identificación de vehículos, ya que los algoritmos de SVM con los hiperparámetros mostrados en la tabla 6 tuvieron una tasa de reconocimiento con promedio de aproximadamente el 86.4% con imágenes tomadas de distintos ángulos y distancias, pero con las muestras con un ángulo recto y distancia similares hubo un aumento considerable obteniendo un accuracy del 95.5%. Se puede considerar que el proyecto haya sido muy fructífero para para este modelo de detección y reconocimiento de caracteres como también proporciona datos importantes para estudios futuros ya que esta investigación muestra el algoritmo y la combinación de hiperparámetros con mejores resultados.

VI. REFERÊNCIAS BIBLIOGRÁFICA

[1] SUNARP. Sunarp: número de autos que circulan en el país acumula una década de crecimiento continuo, 2020.

[2] INEI. PERU Instituto Nacional de Estadística e Informática INEI, 2021.

[3] Sunarp. Cuidado con los ‘ clonadores ’ de vehículos 2019.

[4] Agarwal, P.; Chopra, K.; Kashif, M.; Kumari, V. Implementing ALPR for detection of traffic violations: A step towards sustainability. Procedia Computer Science. Elsevier B.V., 2018, Vol. 132, pp. 738– 743. doi:10.1016/j.procs.2018.05.085.

[5] Sathya, K.B.; Vasuhi, S.; Vaidehi, V. Perspective Vehicle License Plate Transformation using Deep Neural Network on Genesis of CPNet. Procedia Computer Science. Elsevier B.V., 2020, Vol. 171, pp. 1858–1867. doi:10.1016/j.procs.2020.04.199.

[6] Farhat, A.; Hommos, O.; Al-Zawqari, A.; Al-Qahtani, A.; Bensaali, F.; Amira, A.; Zhai, X. Optical character recognition on heterogeneous SoC for HD automatic number plate recognition system. Eurasip Journal on Image and Video Processing 2018, 2018. doi:10.1186/s13640-018-0298-2.

[7] Varma, P.R.K.; Ganta, S.; Hari Krishna, B.; Svsrk, P. A Novel Method for Indian Vehicle Registration Number Plate Detection and Recognition using Image Processing Techniques. Procedia Computer Science. Elsevier B.V., 2020, Vol. 167, pp. 2623–2633. doi:10.1016/j.procs.2020.03.324.

[8] Björklund, T.; Fiandrotti, A.; Annarumma, M.; Francini, G.; Magli, E. Robust license plate recognition using neural networks trained on synthetic images. Pattern Recognition 2019, 93, 134–146. doi:10.1016/j.patcog.2019.04.007.

[9] Divya, P.; Varma, M.; Mouli, U.R. Materials Today : Proceedings Web based optical character recognition application using flask and tesseract. Materials Today: Proceedings 2020. doi:10.1016/j.matpr.2020.10.850.

[10] Salau, A.O.; Yesufu, T.K.; Ogundare, B.S. Vehicle plate number localization using a modified GrabCut algorithm. Journal of King Saud University - Computer and Information Sciences 2021, 33, 399–407. doi:10.1016/j.jksuci.2019.01.011.

[11] Abdullah, S.N.H.S.; Omar, K.; Sahran, S.; Khalid, M. License plate recognition based on support vector machine. Proceedings of the 2009 International Conference on Electrical Engineering and Informatics, ICEEI 2009 2009, 1, 78–82. doi:10.1109/ICEEI.2009.5254811.

[12] Chougule, A.; Shah, M. Local binary pattern with hyperparameter tuned support vector machine for fingerprint classification. 2019 International Conference on Intelligent Computing and Control Systems, ICCS 2019 2019, pp. 1084–1087. doi:10.1109/ICCS45141.2019.9065509.

[13] Zhan, X.; Li, R.; Ukkusuri, S.V. Link-based traffic state estimation and prediction for arterial networks using license-plate recognition data. Transportation Research Part C: Emerging Technologies 2020, 117. doi:10.1016/j.trc.2020.102660.

[14] Silva, S.M.; Jung, C.R. Real-time license plate detection and recognition using deep convolutional neural networks. Journal of Visual Communication and Image Representation 2020, 71. doi:10.1016/j.jvcir.2020.102773.

[15] Arévalo, V.; González, J.; Ambrosio, G. La librería de visión artificial OPENCV 2016. pp. 1–6.

[16] Liu, Y.Q.;Wei, D.; Zhang, N.; Zhao, M.Z. Vehicle-license-plate recognition based on neural networks. 2011 IEEE International Conference on Information and Automation,

ICIA 2011 2011, pp. 363– 366. doi:10.1109/ICINFA.2011.5949018.

[17] Phangtriastu, M.R.; Harefa, J.; Tanoto, D.F. Comparison between Neural Network and Support Vector Machine in Optical Character Recognition. Procedia Computer Science 2017, 116, 351–357. doi:10.1016/j.procs.2017.10.061.

[18] Abedin, M.Z.; Nath, A.C.; Dhar, P.; Deb, K.; Hossain, M.S. License plate recognition system based on contour properties and deep learning model. 5th IEEE Region 10 Humanitarian Technology Conference 2017, R10-HTC 2017 2018, 2018-Janua, 590–593. doi:10.1109/R10- HTC.2017.8289029.

[19] Kumar Sahoo, A. Automatic recognition of Indian vehicles license plates using machine learning approaches. Materials Today: Proceedings 2020. doi:10.1016/j.matpr.2020.09.046.

[20] Tabrizi, S.S.; Cavus, N. A Hybrid KNN-SVM Model for Iranian License Plate Recognition. Procedia Computer Science. Elsevier B.V., 2016, Vol. 102, pp. 588–594. doi:10.1016/j.procs.2016.09.447.

[21] Singh, K.R.; Neethu, K.P.; Madhurekaa, K.; Harita, A.; Mohan, P. Parallel SVM model for Forest Fire Prediction. Soft Computing Letters 2021, p. 100014. doi:10.1016/j.socl.2021.100014.

[22] Khare, V.; Shivakumara, P.; Chan, C.S.; Lu, T.; Meng, L.K.; Woon, H.H.; Blumenstein, M. A novel character segmentationreconstruction approach for license plate recognition. Expert Systems with Applications 2019, 131, 219–239. doi:10.1016/j.eswa.2019.04.030.

[23] Thangallapally, S.K.; Maripeddi, R.; Banoth, V.K.; Naveen, C.; Satpute, V.R. E-Security System for Vehicle Number Tracking at Parking Lot (Application for VNIT Gate Security). 2018 IEEE International Students' Conference on Electrical, Electronics and Computer Science, SCEECS 2018 2018, pp. 1–4. doi:10.1109/SCEECS.2018.8546903.

[24] Momin, B.F.; Mujawar, T.M. Vehicle detection and attribute based search of vehicles in video surveillance system. IEEE International Conference on Circuit, Power and Computing Technologies, ICCPCT 2015 2015, pp. 7–10. doi:10.1109/ICCPCT.2015.7159405