



IPN  
UNIDAD PROFESIONAL INTERDISCIPLINARIA EN  
INGENIERÍA Y TECNOLOGÍAS AVANZADAS

NÚMERO DE LA PRÁCTICA: 3

RECONOCIMIENTO DE PLACAS

---

## Reconocimiento de patrones

---

*Autores:*

Alejo Cerezo Braulio Israel

Pérez Monje Juan Pablo

*Grupo: 4BM3*

*Profesor:*

Anzueto Ríos Álvaro

*Fecha de entrega:* 31 de marzo de 2021

# 1. Introducción Teórica

Hoy en día, el reconocimiento de patrones, junto a otras disciplinas como la inteligencia artificial, forman cada vez más parte de la vida cotidiana. No hace falta pensar que este tipo de tecnología se encuentra en sistemas costosos y complejos, como robots, androides, sistemas de seguridad de gama alta, etc., basta con revisar aplicaciones que proveen empresas como Google en teléfonos comunes para, por ejemplo, buscar una canción con sólo reproducir algunos segundos de la misma, o desbloquear el propio teléfono celular utilizando nuestro rostro. Todo esto se ha podido lograr con el desarrollo de algoritmos cada vez más eficientes, de la mano de mejoras en la arquitectura de dispositivos de uso cotidiano, que permiten imitar la capacidad del cerebro humano para identificar y clasificar objetos.

Entonces, se puede definir al reconocimiento de patrones como la disciplina encargada de extraer información específica de objetos o fenómenos, independientemente de si estos son definidos o abstractos, que nos permita establecer propiedades para relacionarlos con conjuntos o clases del mismo tipo.

## Sistema de reconocimiento de patrones

Un sistema de reconocimiento de patrones se caracteriza por tener las siguientes partes y/o procesos:

- Sensor

El sensor es el dispositivo encargado de la adquisición de datos. Son un tipo de transductor o mecanismo que traduce algún tipo de energía (normalmente mecánica) en diferencias de potencial. Las dificultades del uso o diseño de sistemas con sensores dependen de las características y las limitaciones del sensor, su ancho de banda, resolución, sensibilidad, distorsión, radio de señal y ruido, latencia, etc.

- Modelo

La meta del modelo es hipotetizar cada clase con sus cualidades, procesar los datos recogidos por el sensor, eliminar errores, ruido o defectos por cada patrón percibido y generar un modelo que describa exactamente la semántica del patrón.

- Extracción de características

Es el proceso de generar características que puedan ser usadas en la clasificación de los datos.

- Selección de variables

Consiste en seleccionar cual es el tipo de características o rasgos adecuados para describir los objetos. Para ello, se deben localizar los patrones que inciden en el problema de manera determinante.

- Clasificación

La clasificación trata de asignar las diferentes partes del vector de características a grupos o clases, basándose en los rasgos extraídos. Es usualmente en esta parte en donde se implementan algoritmos de aprendizaje automático.

## Distancia Euclidiana

La distancia Euclidiana se utiliza para medir la separación entre dos puntos, ya que se interpreta como un espacio de  $n$  dimensiones definido mediante la siguiente ecuación:

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

Aunque esta distancia es útil para medir la separación entre dos puntos en el mundo físico, muestra algunas desventajas cuando se utiliza en un espacio de características, como lo es la dependencia con las unidades de cada una de las coordenadas. A la hora de medir la separación en el mundo físico todas las dimensiones se miden con las mismas unidades (metros, pies, etc.). En un espacio de características generalmente no es así, pues existirán algunas características que tengan más peso (importancia) que otras.

## 2. Desarrollo

El programa realizado considera solucionar la problemática que se presenta en el momento en que un automovil desea ingresar a un estacionamiento privado, el cual tiene un sistema de acceso controlado mediante un sistema de reconocimiento de patrones, que identifica la terminación numérica de las placas del vehiculo en cuestión. A continuación, se muestra el algoritmo de reconocimiento de perfiles numéricos implementado como solución al problema planteado.

### Perfiles de números

Primeramente, se generó una base de datos llamada *dataBasePerfiles*, la cual contiene imágenes de números del 0 al 9, que sirvió como referencia para establecer los perfiles de cada número, y así determinar qué dígito es el detectado a través de un recorte en la imagen de la placa. Posterior a esto, se creó una lista que contiene la secuencia de 5 placas llamada *dataBasePlaca*, con la cual se comprobó mediante comparación si la placa analizada se encuentra o no registrada en la base de datos del estacionamiento.

```
ic = io.imread_collection('BaseDatos/*.jpg', conserve_memory=False)
dataBasePerfiles = []
for w in range(10):
    imaDB = (color.rgb2gray((ic[w]))*255<80).astype(int)
    perfilDB = []
    for i in range(imaDB.shape[0]):
        for j in range(imaDB.shape[1]):
            if (imaDB[i,j] == 1):
                perfilDB.append(j)
                break

    for i in range(imaDB.shape[0]):
        for j in range(imaDB.shape[1]-1,0,-1):
            if (imaDB[i,j] == 1):
                perfilDB.append(j)
                break
    dataBasePerfiles.append(np.var(perfilDB))
dataBasePlaca = [[4,2,8,6],[6,8,2,9],[3,1,7,3],[9,3,3,2],[9,3,8,9]]
```

### Lectura y binarización de la imagen

Después, en el programa principal, se leyó la imagen que conforma la placa completa del auto, se binarizó con un umbral de 80 unidades para los tonos de gris, y se mostró por medio de un *figure*. Una vez realizado este paso, se efectuó la suma de los valores de los pixeles de cada fila

por medio del comando `np.sum(ima2,axis=1)`, y se asignó el resultado a la variable `sumaFila`, que corresponde a un vector columna. Con esta variable se indicó cómo es la distribución de los tonos de gris a lo largo de la imagen y se determinó qué filas contienen los datos de interés en la placa (números de la placa).

```
#-----Programa principal-----
ima = color.rgb2gray(color.rgb2rgb(io.imread('Prueba2.png')))*255
plt.figure()
plt.imshow(ima,cmap='gray')

ima2 = ((ima)<80).astype(int) # Binarizar a menores de 80

sumaFila = np.sum(ima2,axis=1)
```

## Primer corte, matrícula de la placa

Posterior a esto, para realizar el corte de la región de interés de la placa, el programa se auxilió de variables bandera, que sirvieron para indicar en qué región de la imagen se encuentran los datos de interés. La lógica de este ciclo `for` se basa en encontrar y guardar los índices inicial y final para realizar el corte. No obstante, antes de eso, el programa seleccionará la región de píxeles con la mayor longitud, pues esta refiere, en la mayoría de los casos, a la zona de la placa donde se encuentra la matrícula. De forma experimental, se determinó que las tonalidades de gris de interés en la placa deben estar en el intervalo en que la suma de las filas sea mayor a 25 unidades. De esta manera, al terminar el ciclo `for` de recorrer el histograma de tonos de gris, se tendrá ya un vector con los índices inicial y final del `corte1` que se debe realizar a las filas de la imagen. `sumaColumna` ahora almacena la suma de los tonos normalizados de los píxeles a lo largo de las columnas de `corte1`

```
intervInter = []
indices = []
temp = 0
Max = 0
bandera=2
# # ----- Corte Filas -----
for i in range(len(sumaFila)):
    if (sumaFila[i]>25):
        bandera = 1

    if (bandera == 0):
        temp = len(intervInter)
        if(temp>Max):
            Max = temp
            corteFinal = intervInter
            indicesFinal = indices
```

```
        intervInter = []
        indices = []

    if bandera == 1:
        intervInter.append(sumaFila[i])
        indices.append(i)
        bandera = 0

    if(i==len(sumaFila)-1):
        intervInter = []
        indices = []

corte1 = ima2[indicesFinal[0]:indicesFinal[len(indicesFinal)-1],:]
sumaColumna = np.sum(corte1,axis=0)
```

## Segundo corte, clasificador

El código del clasificador se muestra a continuación. Al principio, se volvieron a inicializar variables y banderas del proceso anterior, pero esta vez con la finalidad de realizar un corte por columnas, mismo que se hizo de derecha a izquierda para que, mediante un contador, se recorten sólo los 4 números referentes a las terminaciones de las placas. El conteo comienza cada que la suma de una columna es menor a 10 unidades (pues esto refiere a que esa zona corresponde a un espacio entre cada dígito), o lo que es lo mismo, cuando se detecta un *valle* en el histograma generado por *sumaColumna*. Al mismo tiempo en que cada número en la placa es aislado, se llama a la función *detectarNumero*, la cual identifica el número en el recorte empleando la comparación entre los perfiles del recorte y la base de datos de los perfiles de los números.

La variable *idPlaca* es un arreglo que guarda los números que detectó la función previa en el ciclo *for*. Sin embargo, ya que estos se leyeron de derecha a izquierda, cuando termina el proceso de identificación, se utiliza el comando *reverse* para dar la secuencia real de los números en la placa. Por último, mediante un ciclo *for* se compara *idPlaca* con la base de datos de las placas registradas para encontrar si ésta tiene o no acceso al estacionamiento.

```
start = 0
bandera = 2
indicesCol = []
vecColumnas = []
cont = 0
idPlaca = []
for j in range(len(sumaColumna)-1,0,-1):
    if sumaColumna[j] < 10 :
        start = 1
```

```
if (sumaColumna[j]>10 and start == 1):
    bandera = 1

if (bandera == 0):
    if(max(vecColumnas)>20):
        result = detectarNumero(indicesCol,corte1,dataBasePerfiles)
        print('Número identificado %d'%result)
        idPlaca.append(result)
    indicesCol = []
    cont += 1
    bandera = 2
    vecColumnas = []
    if cont == 6:
        break

if bandera == 1:
    vecColumnas.append(sumaColumna[j])
    indicesCol.append(j)
    bandera = 0

# # -----
idPlaca.reverse()
print('El vehículo tiene placas con terminación: %a'%idPlaca)
noEntra=0
for k in range(len(dataBasePlaca)):
    if idPlaca == dataBasePlaca[k]:
        print('Vehículo aceptado')
        k=len(dataBasePlaca)
        break
    elif (noEntra==1 and k==len(dataBasePlaca)-1):
        print('Vehículo no aceptado')
    else:
        print('Verificando placa...')
        noEntra=1
```

## Funciones

A continuación, se describen las funciones empleadas en el programa.

### 1. Distancia Euclidiana

Esta función es similar a la utilizada en la práctica 1, diferenciándose de esta en la operación que se lleva a cabo para el cálculo de las distancias, puesto que, para fines de este algoritmo, se empleó para establecer la distancia mínima entre la varianza del perfil del número procesado, y la varianza de los perfiles de los números que se encuentran almacenados en la base de datos.

```
def distanciaEu (dato, centros, clases):  
    vecEu = []  
    for i in range(clases):  
        D = np.sqrt((centros[i]-dato)**2)  
        vecEu.append(D)  
    solEu = [np.argmin(vecEu), vecEu[np.argmin(vecEu)]]  
    return solEu
```

### 2. Detectar número

La función *detectarNumero* realiza, en primera instancia, un recorte por columnas sobre la imagen obtenida en *corte1*, a partir de los índices de *VecColumnas*. Posterior a esto, se emplearon dos ciclos *for*, el primero de ellos se encarga de almacenar los índices de la columnas en las que se encuentran *unos* en la imagen, realizando un recorrido sobre ésta de izquierda a derecha, recordando que estos representan el espacio de la imagen que pertenece a un número, y, por lo tanto, dibujando el perfil del contorno izquierdo del número en cuestión. El segundo de ellos se utiliza basado en el mismo principio, pero recorriendo la imagen de derecha a izquierda, con la finalidad de dibujar el perfil restante del número que está siendo procesado.

Una vez obtenidos los perfiles completos del número analizado, se calcula la varianza entre los valores del perfil y este dato se guarda en *varPerfil*. Posteriormente, se calculan las similitudes con cada perfil de la base de datos por medio de la función para distancia Euclideana. La variable *posMin[0]* retorna el valor del índice que corresponde al perfil con la varianza que más se asemeja a nuestro número. Para finalizar con este análisis, se compara dicho índice con valores del 0 al 9 para determinar de qué dígito se trata.

**NOTA:** Obsérvese que si el número identificado corresponde al 6 o al 9, se realiza una segunda clasificación, esto como producto de que, en la gran mayoría de pruebas, estos dos valores eran confundidos por el programa, debido a que la varianza entre sus datos era semejante. Sin embargo, se descubrió en cada perfil analizado, que el histograma del número 6 presentaba un valle mayor a una longitud de 70 unidades, mientras que el del 9 apenas y llegaba a las 50, por esto se determinó esta condicional para diferenciar estos números.



```

def detectarNumero (indicesCol, corte1n,Perfiles):

    corte2 = corte1[:,indicesCol[len(indicesCol)-1]:indicesCol[0]]
    # ----- Perfil de un numero -----
    perfil = []
    for i in range(corte2.shape[0]):
        for j in range(corte2.shape[1]):
            if (corte2[i,j] == 1):
                perfil.append(j)
                break

    for i in range(corte2.shape[0]):
        for j in range(corte2.shape[1]-1,0,-1):
            if (corte2[i,j] == 1):
                perfil.append(j)
                break

    numPlaca=-1
    varPerfil = np.var(perfil)
    print('Distancia Euclidiana: %d'%varPerfil)
    posMinEu = distanciaEu(varPerfil,dataBasePerfiles,10)
    print('Número más cercano: %a'%posMinEu[0])
    acum = []
    if (posMinEu[0] == 0):
        return 0
    elif(posMinEu[0] == 1):
        return 1
    elif(posMinEu[0] == 2):
        return 2
    elif(posMinEu[0] == 3):
        return 3
    elif(posMinEu[0] == 4):
        return 4
    elif(posMinEu[0] == 5):
        return 5
    elif(posMinEu[0] == 6 or posMinEu[0] == 9):
        for o in range(len(perfil)):
            if(perfil[o]<20):
                acum.append(perfil[o])
            elif(len(acum)>70):
                return 6
        else:

```

```
        return 9
    elif(posMinEu[0] == 7):
        return 7
    elif(posMinEu[0] == 8):
        return 8
```

### 3. Resultados

Como resultado de la extracción de perfiles para la base de datos se obtuvo la siguiente relación gráfica (figura 1):

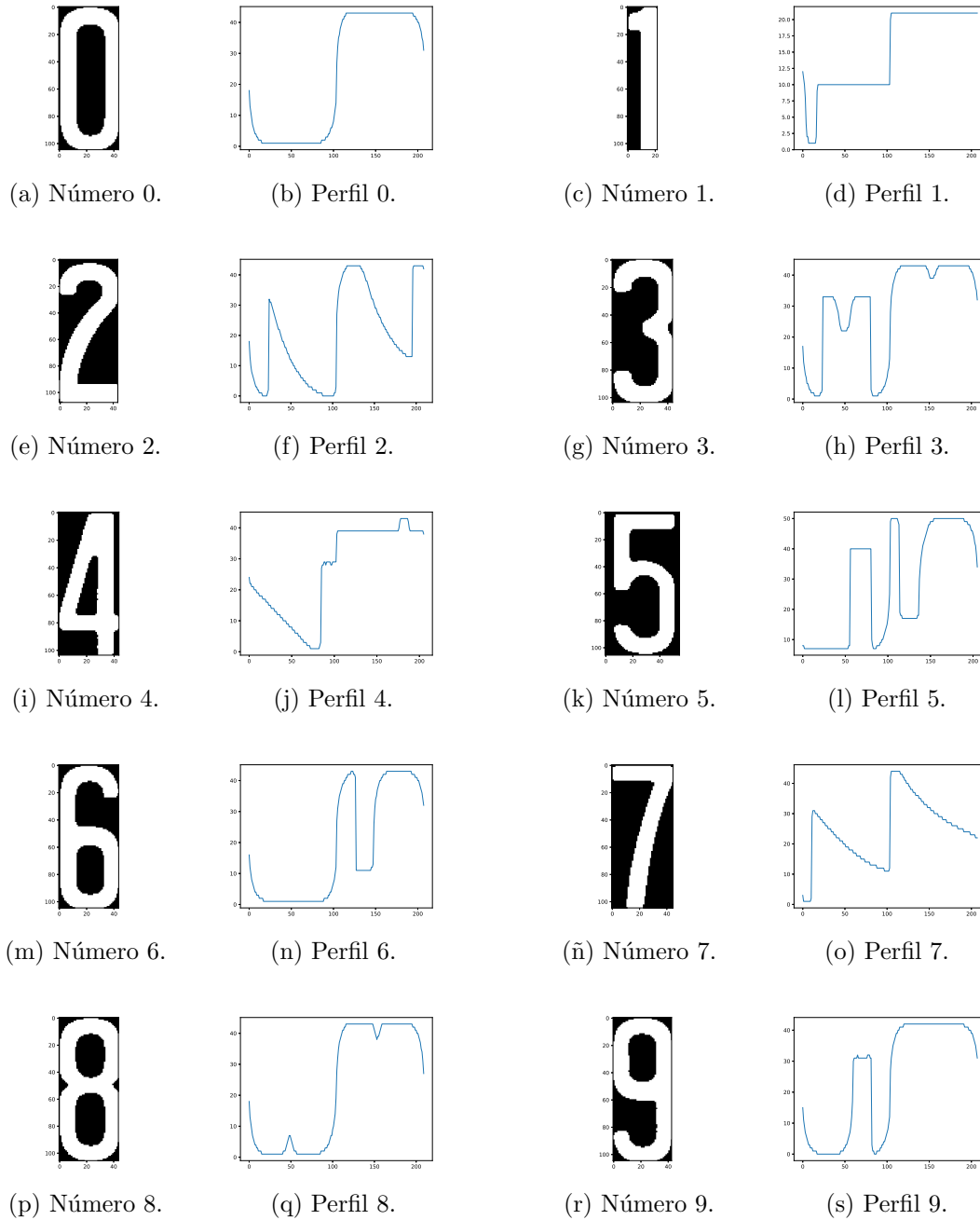
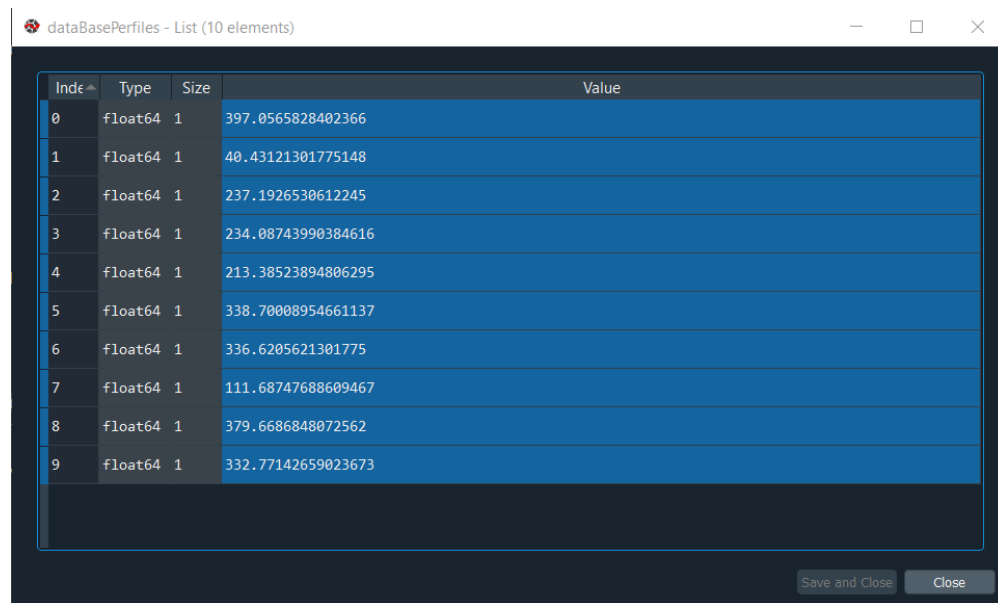


Figura 1: Gráfica comparativa de las relaciones obtenidas en Número-Perfil para la base de datos del programa.

Con lo anterior, se obtuvo a manera de datos de análisis, una tabla con la cual se calculó la varianza en cada conjunto de datos correspondiente al perfil de cada número. Una vez obtenido este conjunto de valores, se generó una matriz para almacenarlos (véase figura 2) y facilitar su comparación con la placa evaluada.



Inde	Type	Size	Value
0	float64	1	397.0565828402366
1	float64	1	40.43121301775148
2	float64	1	237.1926530612245
3	float64	1	234.08743990384616
4	float64	1	213.38523894806295
5	float64	1	338.70008954661137
6	float64	1	336.6205621301775
7	float64	1	111.68747688609467
8	float64	1	379.6686848072562
9	float64	1	332.77142659023673

Figura 2: Variable dataBasePerfiles, donde se almacenan las varianzas correspondientes a cada conjunto de datos por dígito conforme a la figura 1.

Una vez obtenida la base de datos, el programa analizó la placa en cuestión, convirtiéndola primero en escala de grises (figura 3a), su posterior binarización con un rango de tonos de grises menores a 80 unidades (figura 3b), y extracción del histograma de la imagen contabilizando los tonos de blanco por filas (figura 4):



(a) Imagen en gris.



(b) Imagen binarizada.

Figura 3: Procesamiento de la placa evaluada.

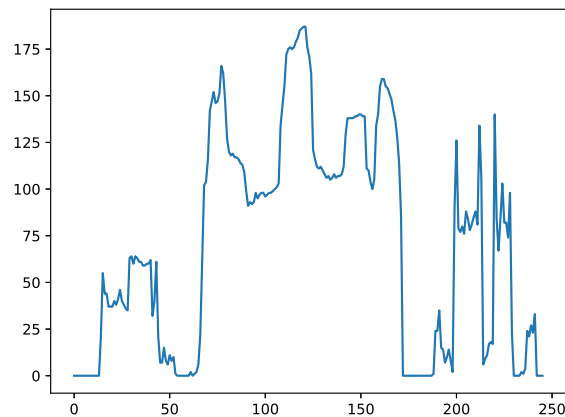


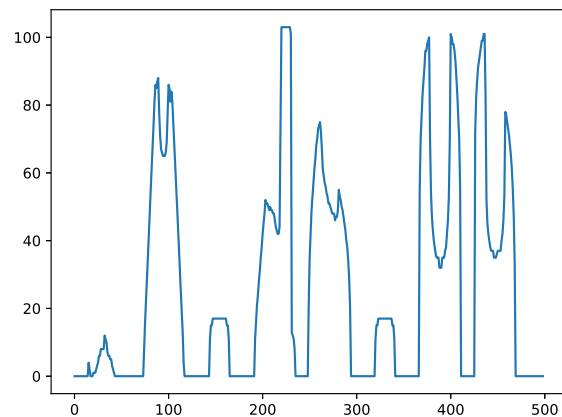
Figura 4: Histograma de las relaciones obtenidas en pixeles por fila seleccionar el área de interés de la fotografía (matrícula).

### Primer corte, matrícula de la placa

El primer corte en la placa se obtuvo mediante el código correspondiente a *Primer corte, matrícula de la placa*, con el cual se obtuvo la imagen de la figura 5. Dada esta imagen, el resultado de la suma de tonos de gris por columna se observa en la figura 5b, en esta figura se puede identificar ya diversos patrones que se relacionan con los dígitos de la placa, sin embargo, se consideran aun aquellos que corresponden a letras y guiones.



(a) Matrícula aislada de la placa.



(b) Histograma por columnas.

Figura 5: Corte y obtención del histograma por columnas para la matrícula de la placa.

Los resultados de la sección *Segundo corte, clasificador*, mostraron los dígitos de interés (figura 6), los cuales se compararon con los perfiles previamente analizados y así se reonoció, número a número, la terminación de la placa.

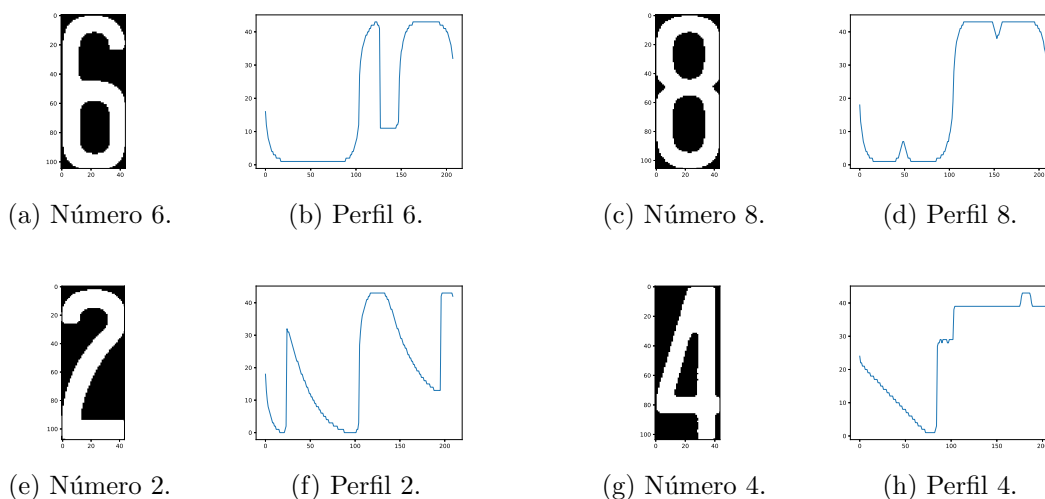


Figura 6: Dígitos obtenidos en la placa, comenzando con el 6 debido al recorrido inverso de la imagen de la matrícula.

## Funciones

Una vez comparado cada dígito de la placa con su correspondiente perfil almacenado en la base de datos, el programa comparó la placa formada con el registro de placas guardadas previamente en él, de esta manera, el algoritmo arrojó los siguientes resultados:

- Número 6.  
Con una distancia euclidiana mínima de 328 unidades, el número más cercano detectado para este valor fue correspondiente al 9, según la base de datos, sin embargo, gracias al código de corrección/diferenciación de 6 y 9, se pudo obtener el valor correcto para el dígito analizado: *Número identificado {6}*.
- Número 8.  
Con una distancia euclidiana mínima de 383 unidades, el número más cercano detectado para este valor fue correspondiente al 8, según la base de datos, siendo este el valor correcto para el dígito analizado: *Número identificado {8}*.
- Número 2.  
Con una distancia euclidiana mínima de 236 unidades, el número más cercano detectado para este valor fue correspondiente al 2, según la base de datos, siendo este el valor correcto para el dígito analizado: *Número identificado {2}*.

- Número 4.

Con una distancia euclidiana mínima de 210 unidades, el número más cercano detectado para este valor fue correspondiente al 4, según la base de datos, siendo este el valor correcto para el dígito analizado: *Número identificado {4}*.

Por lo tanto, tal como lo indica la figura 7, el resultado para esta placa es que se encuentra dentro del registro de matrículas del estacionamiento, y por lo tanto el vehículo es aceptado.

```
In [8]: runfile('D:/Politécnico/7º Semestre/Materias/Reconocimiento de Patrones/
Práctica 3 Placas/reconocerPlacas.py', wdir='D:/Politécnico/7º Semestre/Materias/
Reconocimiento de Patrones/Práctica 3 Placas')
Distancia Euclidiana: 328
Número más cercano: 9
Número identificado {6}
Distancia Euclidiana: 383
Número más cercano: 8
Número identificado {8}
Distancia Euclidiana: 236
Número más cercano: 2
Número identificado {2}
Distancia Euclidiana: 210
Número más cercano: 4
Número identificado {4}
El vehículo tiene placas con terminación: [4, 2, 8, 6]
Vehículo aceptado
El desempeño del clasificador fue de: 100 %
```

Figura 7: Resultados finales arrojados en la consola.

## 4. Observaciones y/o Conclusiones

### Alejo Cerezo Braulio Israel.

Al realizar distintas pruebas sobre las imágenes de las placas, se pudo observar que es posible tener más de un indicador que permita segmentar la imagen para poder procesar la información. El más importante de ellos es la distancia que existe entre un número y otro, así como las regiones en blanco que pertenecen a los guiones de la placa. El haber desarrollado un algoritmo capaz de efectuar esta separación automática de dígitos permite delimitar, por medio de un contador, la cantidad de números que se desean leer, teniendo así un algoritmo que es capaz de adaptarse a múltiples regiones del país, independientemente de si las terminaciones en las placas constan de más o menos números.

La distancia Euclidiana probó ser un método efectivo para establecer la distancia mínima entre las clases de las varianzas de los números, puesto que se mantiene una relación aceptable entre los perfiles de cada número procesado cuando estos son comparados con los perfiles de la base de datos. El algoritmo presentado es una combinación de un clasificador probabilístico y un clasificador por distancias, ya que emplea recursos de ambos grupos para poder entregar una mejor respuesta.

Una de las mayores ventajas que tiene el patrón utilizado para determinar si la computadora ve un 6 o 9, es que permite trabajar con los mismos datos de los perfiles de cada número, haciendo innecesario recurrir a otro método de procesamiento de datos o imágenes, como momentos de Hu.

### Pérez Monje Juan Pablo.

Los resultados finales obtenidos en la presente práctica fueron satisfactorios conforme a la identificación de números en las placas de automóviles, No obstante, esto se debió en gran parte al uso de una tipografía correcta y un análisis de la misma mediante imágenes de características similares (tamaño principalmente). Lo anterior debido a las dificultades que presentó la escala de las imágenes a lo largo del desarrollo del algoritmo de reconocimiento.

Una vez en la etapa de reconocimiento de similitudes por distancia Euclidiana, se encontró con una nueva dificultad, ya que el programa presentó dificultades para la diferenciación del 6 y el 9. Ya que la varianza en sus valores era semejante con una relación 328-331 respectivamente, por lo que esta variable no permite realmente diferenciar ambos dígitos. Para lo anterior, se optó por diseñar un filtro para los valores de 6 y 9 que leyera directamente el histograma mostrado en las figuras [1n](#) y [1s](#) y evaluando la dimensión de su primer valle, pues en el caso del número 6, el valle corresponde a una región más extensa en comparación a la del 9 (pues los perfiles están invertidos).

Para evitar de manera más eficiente las dificultades antes señaladas, se optó también por utilizar un mismo tipo de escala de las placas, esto mediante fotografías de una sola fuente (una sola cámara de captura considerando un mismo ángulo y distancia al objeto), de esta



manera, la varianza correspondía de una forma más efectiva pues ahora los dígitos fueron reconocidos en cada evaluación sin errores. Por ello, basado en la idea original sobre un reconocedor de placas, esta práctica permite conocer la importancia no solo del algoritmo de reconocimiento, sino también, del sensor de captura de datos, debiéndose tener especial consideración para la etapa de adquisición de las matrículas.

## Referencias

Rodríguez, D. (2018). *Distancias y métricas en aprendizaje automático*. AnalyticsLANE. Consultado el 20 de marzo de 2021, en: <https://www.analyticslane.com/2018/08/24/distancias-y-metricas-en-aprendizaje-automatico/>

Christopher M. Bishop (2006). *Pattern Recognition and Machine Learning*, Reino Unido: Springer.

Sergio Theodoridis Konstantinos (2009). *Pattern Recognition*, Reino Unido: Elsevier.

Ludmila I. Kuncheva and Christopher J. Whitaker «Researchgate», *Pattern recognition and classification*. Consultado el 26 de marzo 2021. [En línea]. Disponible en: [https : //bit.ly/3g0QPLs](https://bit.ly/3g0QPLs).