



Visión Artificial

Adrián Racero Serrano
Juan Manuel Cardeñosa Borrego



Conceptos teóricos

Índice

01

Introducción

02

Cómo funciona

03

Aplicaciones

04

Beneficios

05

Ejemplos

01

Introducción

Definición, técnicas empleadas y sectores



Introducción

- La **visión artificial** es un campo de la inteligencia artificial que se enfoca en desarrollar métodos para que los ordenadores puedan interpretar y comprender imágenes del mundo real.

Este campo combina técnicas de procesamiento de imágenes, aprendizaje automático y análisis de datos para permitir que las máquinas realicen tareas que simulen la visión humana.

Visión artificial en sectores



Energía



**Servicios
públicos**



Automoción



Industria

02

Cómo funciona

Machine learning y CNN



Funcionamiento

- La visión artificial necesita muchos datos. Se basa en ejecutar análisis de datos una y otra vez hasta que percibe diferencias y finalmente reconoce imágenes.

Tecnologías básicas



**Redes neuronales
convolucionales**



**Machine
learning**

03

Aplicaciones

My moments, Google Translate, etc.



Aplicaciones



Google Translate



IBM con My Moments y la IA en edge computing



Vehículos autónomos

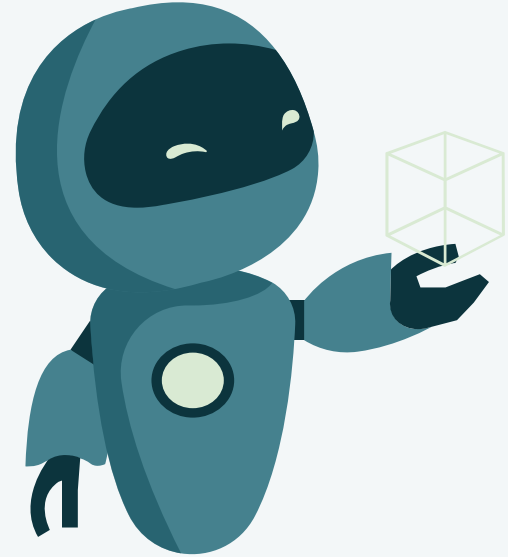
04

Beneficios



Beneficios

- Mejora en la precisión.
- Mejora en la productividad.
- Reducción de costes.
- Mejora en la seguridad.



05

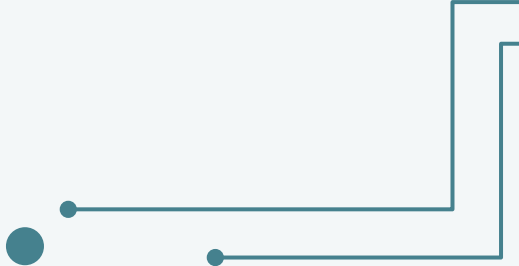
Ejemplos

Ejemplos y algunas técnicas usadas



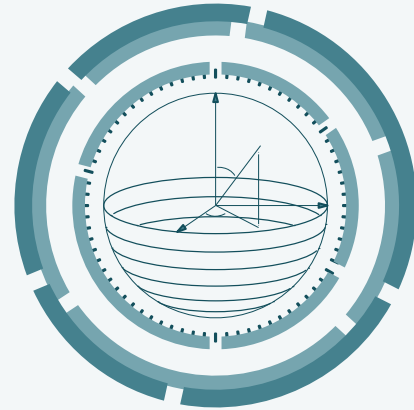


Ejemplos

- Clasificación de imágenes.
 - Detección de objetos.
 - Rastreo de objetos.
 - Recuperación de imágenes basada en contenido.
- 

Técnicas empleadas

- Filtrado.
- Transformación.
- Segmentación.
- Mejora de calidad.
- Compresión.
- Detección de texto.



Caso práctico

Detección de matrículas

Índice

01

Contexto y herramientas

02

Procesamiento de imágenes

03

Extracción del texto

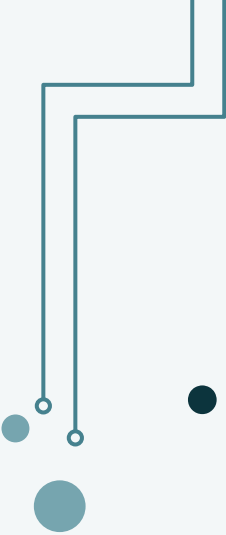
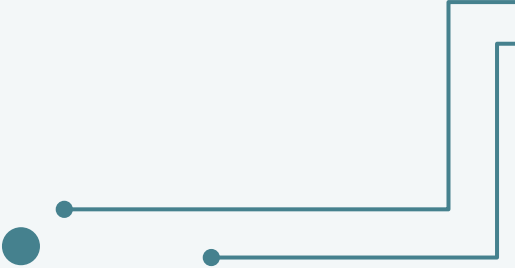
04

Resultados y conclusiones

01

Contexto y herramientas



- 
- Programa en Python. Librerías:
 - cv2
 - skimage
 - pytesseract
- 

02

Procesamiento de imágenes



Características de las imágenes

```
img = cv2.imread("./dataset/coche1.png")
```



Características de las imágenes

```
>>> print(img.shape)  
(512, 512, 3)
```

Canal Rojo



Canal Verde



Canal Azul



Imagen en escala de grises

```
imgBW = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
>>> print(imgBW.shape)  
(512, 512)
```



Imagen binaria

```
value = 170
```

```
imgBIN1 = cv2.threshold(imgBW, value, 255,  
cv2.THRESH_BINARY_INV)[1]
```

```
imgBIN2 = cv2.adaptiveThreshold(imgBW, 255,  
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,  
cv2.THRESH_BINARY_INV, 9, 7)
```

Imagen binaria

Umbral Global

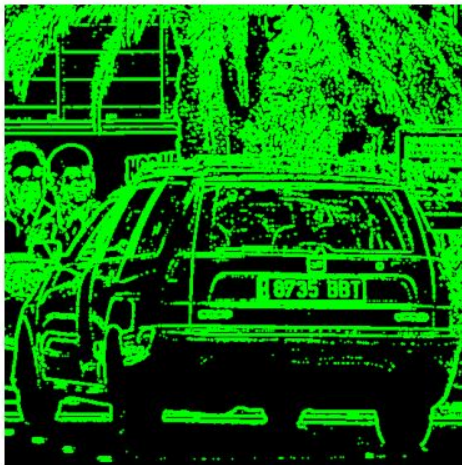


Umbral Adaptativo



Contornos

```
contornos = cv2.findContours(imgBIN,  
cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)[0]
```

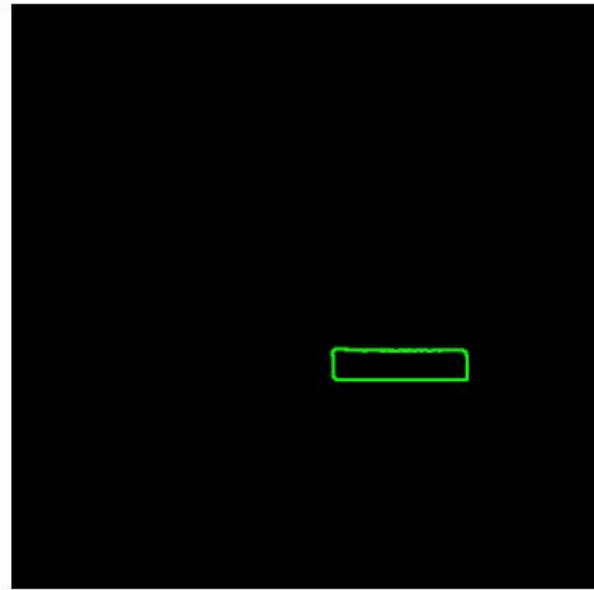


Filtrar contornos

```
ratio = 520.0/120.0
min_w = 64
max_w = 256
min_h = 16
max_h = 64
candidatos = []
    for c in contornos:
        x, y, w, h = cv2.boundingRect(c)
        aspect_ratio = float(w) / h

        if (np.isclose(aspect_ratio, ratio, atol=1.5) and
            (max_w > w > .min_w) and
            (max_h > h > min_h)):
            candidatos.append(c)
```

Filtrar contornos



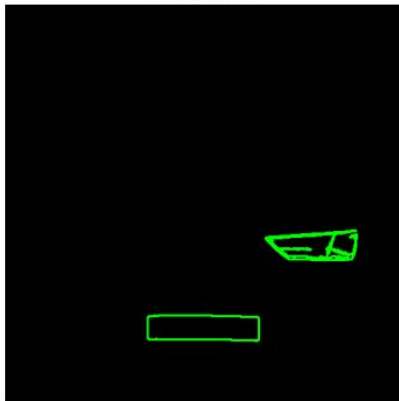
¿Qué ocurre si hay más de 1 candidato?

```
ys = [cv2.boundingRect(c)[1] for c in candidatos]  
candidatoMenor = candidatos[np.argmax(ys)]
```

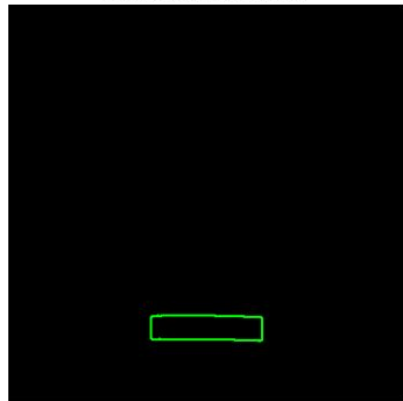
Imagen original



Candidatos



Candidato más abajo



Recortar matrícula

```
x, y, w, h = cv2.boundingRect(candidatoMenor)  
matricula = img[y:y+h,x:x+w]
```



Procesar matrícula

Pasar a escala de grises y binarizar.

A license plate with the text "8735 BBT" and a small icon on the left. The plate is black with white text and is centered within a white square frame. The text is in a bold, sans-serif font. The icon on the left is a small, stylized emblem.

Eliminar ruido en los bordes

```
matriculaSinBordes =  
skimage.segmentation.clear_border(matriculaBIN)
```



E 8735 BBT

Invertir imagen

```
matriculaFinal = cv2.bitwise_not(img)
```



1 8735 BBT

03

Extracción del texto



Configurar pytesseract

```
alphanumeric = "BCDFGHJKLMNPQRSTVWXYZ0123456789"
```

```
options = f"-c
```

```
tessedit_char_whitelist={alphanumeric} -- psm 7"
```

```
txt = pytesseract.image_to_string(img,  
config=options)
```

Estrategia empleada

```
txt = txt[::-1]
letras = ""
numeros = ""
counter = 0
for t in txt:
    if ((t in "BCDFGHJKLMNPSTVWXYZ") and (counter < 3)):
        letras += t
        counter += 1
    if ((t in "0123456789") and (3 <= counter < 7)):
        numeros += t
        counter += 1

letras += "?"*(3 - len(letras))
numeros += "?"*(4 - len(numeros))

matricula = numeros[::-1] + " " + letras[::-1]
```

Salida

Matrícula reconocida → 8735 BBT



04

Resultados y conclusiones



Resultados

ORIGINAL	----	RESULT	-----	SIMILITUD(%)
8735 BBT	----	8735 BBT	-----	100.0%
7206 KDF	----	7206 KDF	-----	100.0%
5683 JZG	----	5533 JZC	-----	62.5%
4386 LJP	----	?385 LJP	-----	75.0%
4991 KXN	----	?991 KXN	-----	87.5%
9723 LCP	----	9723 LCP	-----	100.0%
8586 KBY	----	8586 KBY	-----	100.0%
8206 MCS	----	8206 MCS	-----	100.0%
0378 LKF	----	0373 LKF	-----	87.5%
0798 MNC	----	0798 MNC	-----	100.0%

Coches totales: 10
Numero de aciertos: 6
Numero de fallos: 4
Porcentaje simbolos erroneos: 8.75%

Enlace al repositorio del trabajo

