



Practica 002

## **Captura de video con cámara y edición de imagen**

visión artificial

Funcionamiento, mejoras y resultados

Leonardo Isaac Gonzalez Yañez

18-03-2025

# 001\_AbrirCamarasGrises

Este código captura video en tiempo real desde la cámara, lo convierte a escala de grises y lo muestra en una ventana de OpenCV.

## Explicación línea por línea

### 1 Importación de librerías

```
import numpy as np
import cv2
```

#### Importa las bibliotecas necesarias:

- **OpenCV (cv2)** → Para capturar y procesar video.
  - **NumPy (numpy)** → Aunque se importa, en este código no se usa directamente.
- 

### 2 Iniciar captura de video

```
cap = cv2.VideoCapture(0)
```

#### Abre la cámara para capturar video en tiempo real.

- El número **0** indica que se usará la **cámara principal** (si hay más cámaras, 1 o 2 podría referirse a cámaras adicionales).
- 

### 3 Bucle para capturar y procesar video

```
while(True):
```

#### Inicia un bucle infinito para capturar cuadros de la cámara.

```
    ret, frame = cap.read()
```

#### Captura un cuadro (frame) de la cámara.

- **ret** → Devuelve True si la captura fue exitosa.
- **frame** → Contiene la imagen del cuadro capturado.

---

## 4 Convertir el cuadro a escala de grises

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

✦ **Convierte la imagen de color BGR a escala de grises.**

- OpenCV carga imágenes en **BGR (Blue, Green, Red)** en lugar de **RGB**.
- Convertir a escala de grises **reduce el tamaño de los datos y mejora la velocidad de procesamiento**.

---

## 5 Mostrar el video en tiempo real

```
cv2.imshow('frame', gray)
```

✦ **Muestra el cuadro en una ventana de OpenCV.**

- Se usa `gray`, por lo que la imagen se verá en **blanco y negro**.

💡 **Si quisieras verlo en color, reemplaza `gray` por `frame`:**

```
cv2.imshow('frame', frame)
```

---

## 6 Permitir salir del bucle

```
if cv2.waitKey(1) & 0xFF == ord('q'):  
    break
```

✦ **Permite cerrar la ventana presionando la tecla 'q'.**

- `cv2.waitKey(1)` espera **1 milisegundo** por una tecla.
- `ord('q')` detecta si la tecla presionada es 'q'.
- Si se presiona 'q', el bucle **se detiene** con `break`.

---

## 7 Liberar recursos y cerrar ventanas

```
cap.release()  
cv2.destroyAllWindows()
```

## **Cierra la cámara y las ventanas de OpenCV.**

- `cap.release()` → Libera la cámara para que otros programas puedan usarla.
- `cv2.destroyAllWindows()` → Cierra todas las ventanas de OpenCV.

## **Resumen Final**

Parte del código	Función
<code>cap = cv2.VideoCapture(0)</code>	Abre la cámara principal.
<code>cap.read()</code>	Captura un cuadro de video.
<code>cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)</code>	Convierte el cuadro a escala de grises.
<code>cv2.imshow('frame', gray)</code>	Muestra el video en blanco y negro.
<code>cv2.waitKey(1) == ord('q')</code>	Permite salir al presionar la tecla 'q'.
<code>cap.release()</code>	Libera la cámara.
<code>cv2.destroyAllWindows()</code>	Cierra las ventanas de OpenCV.

## **Mejoras que puedes hacer**

### **Verificar si la cámara se abre correctamente antes de capturar:**

```
if not cap.isOpened():  
    print("Error: No se pudo abrir la cámara.")  
    exit()
```

### **Ajustar la velocidad de captura para reducir la carga del procesador:**

```
cv2.waitKey(30) # Reduce la velocidad de actualización del video
```

### **Guardar el video en un archivo:**

```
fourcc = cv2.VideoWriter_fourcc(*'XVID')  
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 480))  
out.write(frame)
```

## 002\_CapturarVideo

Este código captura video en tiempo real desde la cámara, lo muestra en dos versiones (color y escala de grises) y lo guarda en un archivo de video (output.avi).

### Explicación línea por línea

#### Importación de librerías


```
import numpy as np
import cv2
```

#### Importa las bibliotecas necesarias:

- **OpenCV (cv2)** → Para capturar y procesar video.
  - **NumPy (numpy)** → Aunque se importa, no se usa directamente en este código.
- 

#### Iniciar captura de video

```
cap = cv2.VideoCapture(0)
```

 **Abre la cámara principal** (0 indica la cámara predeterminada del sistema).

- Si tienes más de una cámara, puedes probar con 1, 2, etc.
- 

#### Configurar el formato de grabación

```
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640,480))
```

#### Configura la grabación del video en un archivo (output.avi).

- `fourcc = cv2.VideoWriter_fourcc(*'XVID')` → Define el códec **XVID** para comprimir el video.
  - `cv2.VideoWriter('output.avi', fourcc, 20.0, (640,480))`
    - 'output.avi' → Nombre del archivo de salida.
    - 20.0 → Cuadros por segundo (**fps**).
    - (640,480) → Resolución del video.
- 

#### 4 Bucle para capturar y procesar video

`while(True):`

📌 **Inicia un bucle infinito** para capturar y procesar el video en tiempo real.

`ret, frame = cap.read()`

📌 **Captura un cuadro (frame) de la cámara.**

- `ret` → True si el cuadro se capturó correctamente.
  - `frame` → Contiene la imagen del cuadro.
- 

#### 5 Convertir el cuadro a escala de grises

`gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)`

📌 **Convierte la imagen de color BGR a escala de grises.**

- Se usa `cv2.COLOR_BGR2GRAY` para eliminar la información de color.
- 

#### 6 Guardar el video

`out.write(frame)`

## ✚ Escribe cada cuadro en el archivo de video (output.avi).

- Se está guardando en **color**, no en escala de grises.
- **Si quisieras guardar en escala de grises, usa esto:**

```
out.write(cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR))
```

OpenCV requiere **imágenes en color** para guardar videos, por eso hay que reconvertirlas.

---

## 🗲Mostrar el video en pantalla

```
cv2.imshow('frame', frame)  
cv2.imshow('gray', gray)
```

## ✚ Muestra dos versiones del video:

- 'frame' → En **color**.
  - 'gray' → En **escala de grises**.
- 

## 🗲Salida del programa

```
if cv2.waitKey(1) & 0xFF == ord('q'):  
    break
```

## ✚ Permite cerrar el programa presionando 'q'.

- cv2.waitKey(1) espera 1 milisegundo por una entrada de teclado.
  - ord('q') detecta si se presionó la tecla 'q'.
  - Si se presiona 'q', el programa **se detiene** con break.
- 

## 🗲Liberar recursos y cerrar ventanas

```
cap.release()
```

```
out.release()
cv2.destroyAllWindows()
```

### **Cierra la cámara y las ventanas de OpenCV.**

- `cap.release()` → Libera la cámara para que otros programas puedan usarla.
- `out.release()` → Guarda y cierra el archivo de video.
- `cv2.destroyAllWindows()` → Cierra todas las ventanas abiertas.

---

### **Resumen Final**

Parte del código	Función
<code>cap = cv2.VideoCapture(0)</code>	Abre la cámara principal.
<code>cv2.VideoWriter()</code>	Configura la grabación del video.
<code>cap.read()</code>	Captura un cuadro de video.
<code>cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)</code>	Convierte el cuadro a escala de grises.
<code>cv2.imshow('frame', frame)</code>	Muestra el video en color.
<code>cv2.imshow('gray', gray)</code>	Muestra el video en escala de grises.
<code>out.write(frame)</code>	Guarda el video en color en un archivo AVI.
<code>cv2.waitKey(1) == ord('q')</code>	Permite salir al presionar 'q'.
<code>cap.release()</code>	Libera la cámara.
<code>out.release()</code>	Guarda y cierra el archivo de video.
<code>cv2.destroyAllWindows()</code>	Cierra las ventanas de OpenCV.

---

### **Mejoras que puedes hacer**

- ✓ **Verificar si la cámara se abre correctamente antes de capturar:**



```
if not cap.isOpened():  
    print("Error: No se pudo abrir la cámara.")  
    exit()
```

✓ **Reducir la carga del procesador aumentando el tiempo de espera:**

```
cv2.waitKey(30) # Reduce la velocidad de actualización del video
```

✓ **Guardar el video en escala de grises en lugar de color:**

```
out.write(cv2.cvtColor(gray, cv2.COLOR_GRAY2BGR))
```

## 003\_EfectosEnCamara

✚ **Efectos añadidos al video en tiempo real**

### 1 Conversión a escala de grises

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

✚ **Convierte el cuadro a escala de grises** para resaltar los detalles de la imagen sin color.

- Se usa en muchas aplicaciones de **Visión Artificial**, como reconocimiento de objetos y detección de rostros.

---

### 2 Efecto de negativo

```
negative = cv2.bitwise_not(frame)
```

✚ **Invierte los colores de la imagen.**

- Funciona aplicando la operación bit a bit NOT, lo que convierte los píxeles claros en oscuros y viceversa.
- **Ejemplo:**
  - Un píxel blanco (255,255,255) se convierte en negro (0,0,0).

- Un píxel azul (0, 0, 255) se convierte en amarillo (255, 255, 0).
  - Útil en **análisis de imágenes médicas** o para mejorar la visibilidad de ciertos patrones.
- 

### 3 Desenfoque Gaussiano

```
blur = cv2.GaussianBlur(frame, (15, 15), 0)
```

#### 📌 Aplica un filtro de desenfoque a la imagen.

- Se usa para **suavizar bordes y reducir ruido** en una imagen.
  - (15, 15) es el tamaño del **kernel** (más alto = más desenfoque).
  - **Ejemplo de uso:**
    - Preprocesamiento en **detección de rostros**.
    - Eliminación de ruido antes de aplicar otros filtros.
- 

### 4 Detección de bordes con Canny

```
edges = cv2.Canny(frame, 100, 200)
```

#### 📌 Detecta los bordes de los objetos en la imagen.

- Utiliza la técnica de **Canny Edge Detection**.
  - **Parámetros (100, 200)** → Define los umbrales para detectar bordes fuertes y débiles.
  - **Ejemplo de uso:**
    - Identificación de contornos en **reconocimiento de caracteres (OCR)**.
    - Segmentación de objetos en **visión artificial**.
- 

### 5 Mostrar los efectos en ventanas separadas

```
cv2.imshow('Original', frame)
```

```
cv2.imshow('Gris', gray)
cv2.imshow('Negativo', negative)
cv2.imshow('Desenfoque', blur)
cv2.imshow('Bordes', edges)
```

### 📌 Muestra el video en diferentes versiones:

- **"Original"** → Imagen sin modificaciones.
- **"Gris"** → Escala de grises.
- **"Negativo"** → Inversión de colores.
- **"Desenfoque"** → Imagen suavizada.
- **"Bordes"** → Solo contornos detectados.

Esto permite **comparar los efectos en tiempo real** y entender cómo afectan la imagen.

---

### 📌 Resumen de los efectos aplicados

Efecto	Función	Ejemplo de uso
<b>Escala de grises</b>	Convierte la imagen a blanco y negro	Reducción de procesamiento en Visión Artificial
<b>Negativo</b>	Invierte los colores de la imagen	Análisis de patrones en imágenes médicas
<b>Desenfoque Gaussiano</b>	Suaviza la imagen y reduce el ruido	Preprocesamiento en detección de rostros
<b>Detección de bordes (Canny)</b>	Resalta los contornos de los objetos	Segmentación de imágenes y OCR

---

### 📌 ¿Cómo mejorar el código?

✓ **Agregar un filtro sepia** para darle un efecto de foto antigua:

```
sepia = np.array(frame, dtype=np.float64) # Convertir a flotante
sepia = cv2.transform(sepia, np.matrix([[0.393, 0.769, 0.189],
```

```
[0.349, 0.686, 0.168],  
[0.272, 0.534, 0.131]]))  
sepia = np.clip(sepia, 0, 255).astype(np.uint8) # Limitar valores y  
convertir a entero  
cv2.imshow('Sepia', sepia)
```

- ✓ **Permitir cambiar de efecto con el teclado (1, 2, 3, etc.).**
- ✓ **Reducir la carga en el procesador aumentando cv2.waitKey(30).**

## 004\_RotacionDeVideo

Este código captura video en tiempo real desde la cámara y **rota cada cuadro 45°** antes de mostrarlo en pantalla.

### Explicación del código

#### Captura de video

```
cap = cv2.VideoCapture(0)
```

#### Abre la cámara principal para capturar video.

- Si hay más cámaras disponibles, puedes cambiar 0 por 1, 2, etc.

---

#### Bucle de captura y procesamiento

```
while True:  
    ret, frame = cap.read()  
    if not ret:  
        break
```

#### Captura un cuadro (frame) en cada iteración del bucle.

- ret devuelve True si el cuadro se capturó correctamente.

- Si ret es False, significa que hay un problema con la cámara, y el bucle se detiene.
- 

### 3 Obtener el centro de la imagen

```
(h, w) = frame.shape[:2]  
center = (w // 2, h // 2)
```

#### 📌 Obtiene el tamaño de la imagen y calcula su centro.

- frame.shape[:2] devuelve el **alto (h) y ancho (w)** de la imagen.
  - center = (w // 2, h // 2) calcula el **centro de la imagen** para usarlo como punto de rotación.
- 

### 4 Matriz de transformación para rotar 45°

```
M = cv2.getRotationMatrix2D(center, 45, 1.0)
```

#### 📌 Genera una matriz de transformación para rotar la imagen.

- center → Punto alrededor del cual se rota la imagen.
- 45 → Ángulo de rotación en grados.
- 1.0 → Factor de escala (1.0 = tamaño original).

💡 Esta matriz se usa para transformar la imagen con **cv2.warpAffine()**.

---

### 5 Aplicar la rotación a la imagen

```
rotated = cv2.warpAffine(frame, M, (w, h))
```

#### 📌 Aplica la transformación de rotación a la imagen.

- `cv2.warpAffine(frame, M, (w, h))` usa la matriz M para girar la imagen **45°**.
- `(w, h)` mantiene el tamaño original de la imagen.

💡 Si quieres que la imagen rota se ajuste automáticamente al tamaño correcto sin recortes, usa `cv2.getRotationMatrix2D()` con corrección de bordes.

---

## 6. Mostrar la imagen rotada

```
cv2.imshow('Rotado 45°', rotated)
```

📌 Muestra la imagen rotada en una ventana llamada "Rotado 45°".

---

## 7. Cerrar el programa al presionar 'q'

```
if cv2.waitKey(1) & 0xFF == ord('q'):  
    break
```

📌 Permite salir del programa presionando la tecla 'q'.

---

## 8. Liberar la cámara y cerrar ventanas

```
cap.release()  
cv2.destroyAllWindows()
```

📌 Libera la cámara y cierra las ventanas de OpenCV.

---

## 📌 Resumen del proceso

Paso	Función
------	---------

Capturar el video	cap.read() obtiene un cuadro de la cámara.
Obtener el tamaño de la imagen	frame.shape[:2] obtiene alto y ancho.
Definir el centro de rotación	center = (w // 2, h // 2).
Crear la matriz de rotación	cv2.getRotationMatrix2D(center, 45, 1.0).
Aplicar la rotación	cv2.warpAffine(frame, M, (w, h)).
Mostrar el video rotado	cv2.imshow('Rotado 45°', rotated).
Salir con 'q'	cv2.waitKey(1) == ord('q').

## ✦ Mejoras que puedes hacer

### ✓ Evitar que se recorte la imagen tras la rotación

Cuando se rota una imagen en OpenCV, puede haber **recortes**. Para evitar esto, puedes recalcular el tamaño del lienzo usando cv2.warpAffine() con una nueva dimensión:

```
import cv2
import numpy as np

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    if not ret:
        break

    (h, w) = frame.shape[:2]
    center = (w // 2, h // 2)

    # Calcular la nueva dimensión del lienzo
    M = cv2.getRotationMatrix2D(center, 45, 1.0)
    cos = np.abs(M[0, 0])
    sin = np.abs(M[0, 1])

    # Nuevo tamaño de la imagen rotada
    new_w = int((h * sin) + (w * cos))
```

```
new_h = int((h * cos) + (w * sin))

# Ajustar la matriz de transformación para evitar recortes
M[0, 2] += (new_w / 2) - center[0]
M[1, 2] += (new_h / 2) - center[1]

rotated = cv2.warpAffine(frame, M, (new_w, new_h))

cv2.imshow('Rotado sin recortes', rotated)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break

cap.release()
cv2.destroyAllWindows()
```

- ✓ **Permitir cambiar el ángulo de rotación con las teclas a y d.**
- ✓ **Agregar un efecto de zoom mientras rota la imagen.**