César Soltero Pérez

18310460

7E1

Sistemas de procesamiento de imágenes y visión artificial

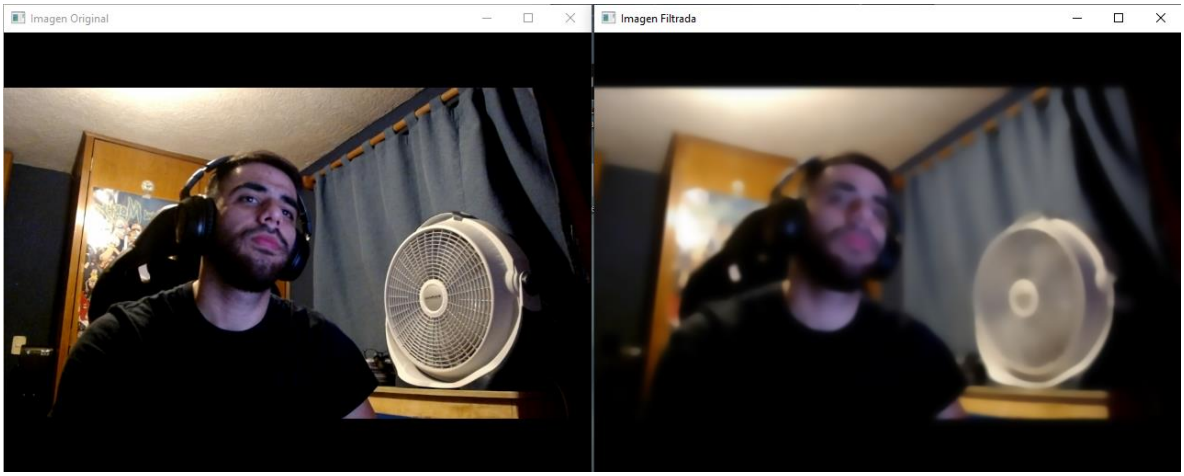
Práctica 7

Remover Ruido – Lineal y morfológicamente en VIDEO.

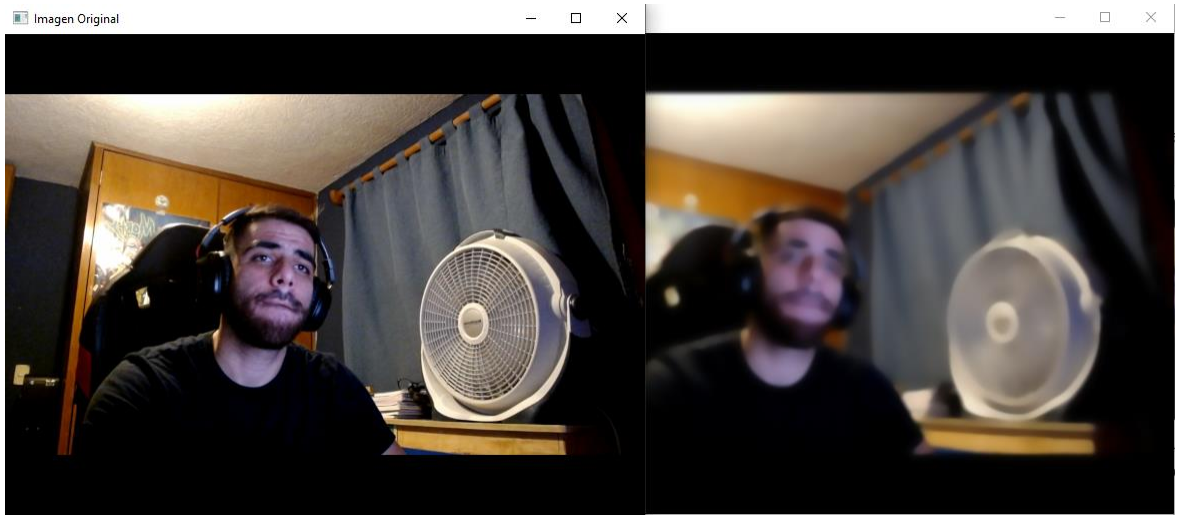
Objetivo: Remover ruido de la detección F+ y F-

Desarrollo:

Para limpiar el ruido uno de los filtros mas comunes es el lineal, usando una matriz de kernel para representar los coeficientes del filtro.



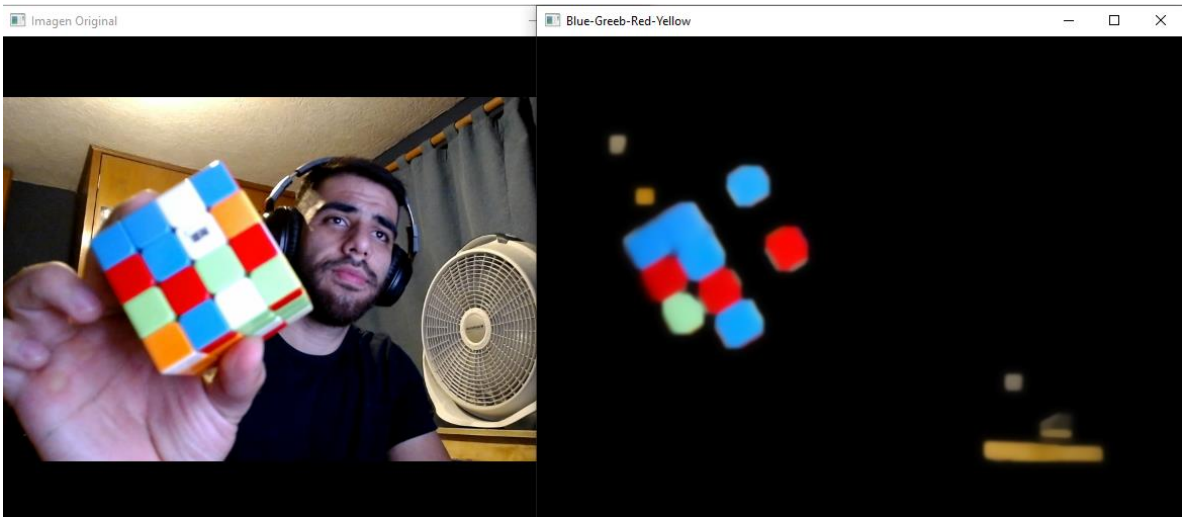
El filtro Gaussiano es uno de los más utilizados. Se realiza convolución a cada píxel de entrada con un núcleo Gaussiano y luego los suma todos para crear la matriz de salida.



El filtro bilateral es especial ya que no se encarga específicamente de limpiar el ruido en las imágenes, sino que se encarga de suavizar los bordes de esta.

Falsos positivos - En el área negra lejos del objeto a detectar se ven pixeles azules porque estamos buscando el color azul.
Falsos negativos - Dentro del área azul se ven pixeles negros.

Filtros



Código:

```
import numpy as np
import cv2

cam = cv2.VideoCapture(-1)

amarilloBajo = np.array([20,70,20],np.uint8)
amarilloAlto = np.array([36,255,255],np.uint8)

azulBajo = np.array([100,100,40],np.uint8)
azulAlto = np.array([125,255,255],np.uint8)

verdeBajo = np.array([37,70,20],np.uint8)
verdeAlto = np.array([90,255,255],np.uint8)

rojoBajo1 = np.array([0,110,70],np.uint8)
rojoAlto1 = np.array([3,255,255],np.uint8)

rojoBajo2 = np.array([177,110,70],np.uint8)
rojoAlto2 = np.array([179,255,255],np.uint8)

kernel_1 = np.ones((5,5), np.uint8)
kernel_n = np.ones((15,15),np.uint8)
kernel_2 = np.ones((15,15),np.float32)/255

print('Presiona "espacio" para cerrar el video.')
```

```
while(cam.isOpened()):
    ready,imgVolteada = cam.read()
    imgOriginal = cv2.flip(imgVolteada,1)
    hsv = cv2.cvtColor(imgOriginal,cv2.COLOR_BGR2HSV)

    #establecemos los rangos de colores
    maskAmarillo = cv2.inRange(hsv,amarilloBajo,amarilloAlto)
    maskAzul = cv2.inRange(hsv,azulBajo,azulAlto)
    maskVerde = cv2.inRange(hsv,verdeBajo,verdeAlto)
    maskRojo = cv2.add(cv2.inRange(hsv,rojoBajo1,rojoAlto1),
                       cv2.inRange(hsv,rojoBajo2,rojoAlto2))

    #Falsos positivos
    maskAmarillo = cv2.dilate(maskAmarillo, kernel_1, iterations = 1)
    maskAzul = cv2.dilate(maskAzul, kernel_1, iterations = 1)
    maskVerde = cv2.dilate(maskVerde, kernel_1, iterations = 1)
    maskRojo = cv2.dilate(maskRojo, kernel_1, iterations = 1)

    #Falsos negativos
    maskAmarillo = cv2.morphologyEx(maskAmarillo, cv2.MORPH_OPEN, kernel_n)
    maskAzul = cv2.morphologyEx(maskAzul, cv2.MORPH_OPEN, kernel_n)
    maskVerde = cv2.morphologyEx(maskVerde, cv2.MORPH_OPEN, kernel_n)
    maskRojo = cv2.morphologyEx(maskRojo, cv2.MORPH_OPEN, kernel_n)

    #Extracción de colores
    imgAmarillo = cv2.bitwise_and(imgOriginal,imgOriginal,mask=maskAmarillo)
    imgAzul = cv2.bitwise_and(imgOriginal,imgOriginal,mask=maskAzul)
    imgVerde = cv2.bitwise_and(imgOriginal,imgOriginal,mask=maskVerde)
    imgRojo = cv2.bitwise_and(imgOriginal,imgOriginal,mask=maskRojo)

    #Construccion de imagenes de salida
    BGRY = cv2.add(imgAmarillo,cv2.add(imgAzul,cv2.add(imgVerde,imgRojo)))
    imFiltrada = cv2.subtract(imgOriginal,imgAmarillo)
    imFiltrada = cv2.subtract(imFiltrada,imgAzul)
    imFiltrada = cv2.subtract(imFiltrada,imgVerde)
    imFiltrada = cv2.subtract(imFiltrada,imgRojo)

    #Filtro para ruido
    BGRY = cv2.GaussianBlur(BGRY,(15,15),0)
    imFiltrada =  cv2.GaussianBlur(imFiltrada,(15,15),0)

    #Suavizado de bordes
    BGRY = cv2.bilateralFilter(BGRY, 15, 75, 75)
    imFiltrada = cv2.bilateralFilter(imFiltrada, 15, 75, 75)

    if ready == True:
        cv2.imshow('Imagen Original',imgOriginal)
        cv2.imshow('Blue-Greeb-Red-Yellow',BGRY)
        cv2.imshow('Imagen Filtrada',imFiltrada)
```

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

Repositorio:

https://github.com/Cesarsp41/Practica-7