Análisis de Movimiento.

J.F. Sánchez Castillo, ITNM.

I. INTRODUCCIÓN.

El siguiente documento contiene los pasos seguidos durante el estudio movimiento de videos en los cuales el fondo permanece estático y se encuentran personas en movimiento, las etapas del análisis de forma general son:

1- Obtención del fondo del video.
2- Segmentación de los objetos en movimiento.
3- Binarizacion de los cuadros que contienen los objetos en movimiento.
4- Detección de contornos.

Para es el estudio de los videos se generaron 2 programas escritos en Python 3 con las librerías de Opencv4. El primer programa prueba los métodos contenidos en Opencv para extracción del fondo de videos y representación de contornos, mientras que el segundo programa aplica el promedio a los cuadros del video y aplica la técnica de minimos cuadrados para el modelado del fondo, posteriormente se aplica el análisis de componentes principales PCA para obtener el gradiente y la orientación de los contornos los cuales son representados mediante líneas en el video de origen.

Cada programa contiene un menú que permite seleccionar el nombre del video o un video de prueba preestablecido, así como el método para extracción de fondo.

```
Este programa Modela el fondo de un video, por medio de los metodos
      de substraccion  disponibles en Opencv4
Ingrese el video a analizar, seleccione una opcion:
Opcion 1: video preestablecido,  presione --> '1'
Opcion 2: Utilizar la camara de la computadora, presione --> '2'
Opcion 3: Ingresar el nombre del video, presione--> '3'
Ingrese una opcion:__3
Selecciono la opcion 3
, ingrese el nombre del video con extension --->vtest.avi
presione una tecla para continuar
Seleccine el metodo de substraccion de fondo:
  4)MOG 5)MOG2 6)KNN 7)GMG 8)CNT --->>4
selecciono el metodo MOG
presione una tecla para continuar
```

Figura1. Menú Programa 1.

```
Este programa Modela el fondo de un video, por medio del promedio
      de cuadros de video, utilizando el metodo de minimos cuadrados
Ingrese el video a analizar, seleccione una opcion:
Opcion 1: video preestablecido,  presione --> '1'
Opcion 2: Utilizar la camara de la computadora, presione --> '2'
Opcion 3: Ingresar el nombre del video, presione--> '3'
Ingrese una opcion:__2
Selecciono la opcion 3
, ingrese el nombre del video con extension --->vtest.avi
presione una tecla para continuar
presione una tecla para continuar
```

Figura2. Menú Programa 2.

II. EXTRACCIÓN DE FONDO.

A. Método GMG

Este algoritmo utiliza la estimación estadística del fondo y la segmentación Bayesiana por pixel. Las estimaciones son adaptativas, las nuevas observaciones tienen mayor peso que las observaciones anteriores para ajustar la iluminación variable. Se aplican varios filtros morfológicos como apertura y cerradura para remover ruido no deseado. Durante los primeros cuadros solo se apreciará una pantalla negra.

B. Método MOG.

Es un algoritmo de segmentación de fondo basado en una mezcla gaussiana. Utiliza un método para modelar cada pixel de fondo utilizando una mezcla de distribución Gaussiana (K=3 a 5). El peso de la mezcla representa la proporción de tiempo que los colores permanecen en escena. Los colores de fondo son los únicos que permanecen más tiempo y más estáticos. El objeto se inicia con la función: createBackgroundSubtractorMOG(), sus parámetros son longitud de historia, numero de mezclas gaussianas, umbral.

C. Método MOG2.

Es un algoritmo de segmentación de fondo basado en mezcla gaussiana. Una característica de este algoritmo es que selecciona el numero apropiado de distribuciones gaussianas para cada pixel. Permite una mejor adaptabilidad a la variación de escenarios causadas por los cambios de iluminación.
El objeto se inicia con la función createBackgroundSubtractorMOG2().

D. Método CNT

Se caracteriza por ser más rápido que MOG2 en sistemas con recursos reducidos, y en sistemas de gama alta su velocidad es similar. El objeto se inicia con la función createBackgroundSubtractorCNT().

E. Método KNN

Es un método para segmentación de fondo basado en k-vecindarios cercanos, se caracteriza por ser eficiente cuando el número de pixeles del fondo es bajo. El objeto se inicia con la función createBackgroundSubtractorKNN().

F. Método mínimo cuadrados.

L.Wang et al. [4] utiliza el método de mínimos cuadrados para calcular el fondo de la imagen mediante la fórmula:

$$b_{xy} = {}_{p}minmed \quad {}_{t}(I_{xy}^{t} - p)^2$$

Donde I representa una secuencia de n imágenes, p es el valor de brillo del pixel en la ubicación xy y t representa el número de cuadro de video actual.

En esta práctica la función de mínimos cuadrados es aproximada em las siguientes líneas de código:

Calculo del promedio de todos los cuadros para cada pixel:

```
cv.accumulateWeighted(frame_ave, avg, 0.005)
```

calculo del valor absoluto del promedio:

```
result = cv.convertScaleAbs(avg)
```

para cada nuevo cuadro, el cálculo del promedio conserva el valor mínimo encontrado, en la variable result_min:

```
np.array(result)
result_min = np.min(result)
```

calculo del valor del fondo por medio de la resta del promedio general y el promedio mínimo calculado en cada pixel:

```
result_bg=result-result_min
```

III. TRANSFORMACIONES MORFOLÓGICAS

Para ambos programas, es necesario aplicar las operaciones de apertura y cerradura para reducir los elementos del fondo que pudieran haber permanecido en las orillas del objeto, así como cerrar los contornos que pudieron haber quedado incompletos, este se hace mediante las funciones: cv.morphologyEx(_, cv.MORPH_OPEN, kernel) , cv.morphologyEx(_, cv.MORPH_CLOSE, kernel).

IV. REPRESENTACIÓN DE SILUETA

Una vez que el fondo del video se ha modelado y se ha hecho la resta con el cuadro de video actual para segmentar los objetos en movimiento, el siguiente paso es representar el contorno del objeto para obtener los datos de orientación objeto.

Los métodos incluidos en Opencv utilizan la funciones cv.findContours() y cv.drawContours() para encontrar y posteriormente dibujar los contornos detectados.

El método de mínimos cuadrados utiliza el análisis de componentes principales mediante la función: cv.PCACompute2()

V. RESULTADOS.

A. Métodos incluidos en Opencv4

Figura 3. Objetos segmentados y representación del contorno para método MOG.

Figura 4. Objetos segmentados y representación del contorno para método MOG2.

Figura 5. Objetos segmentados y representación del contorno para método KNN.

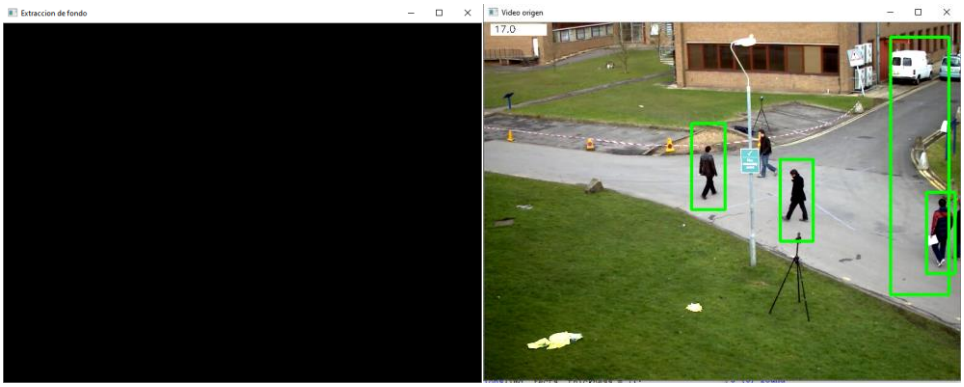

Figura 6. Objetos segmentados y representación del contorno para método GMG.

Figura 7. Objetos segmentados y representación del contorno para método CNT.

B. Método mínimos cuadrados y PCA

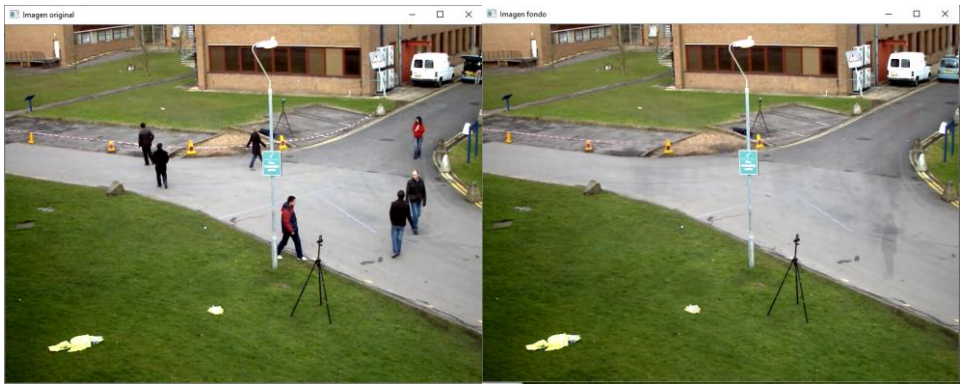

Figura 8. Video de origen y modelado del fondo obtenido para el método LMSR..

Figura 9. Objetos segmentados y representación del contorno para el método LMSR.

VI. Conclusiones

En esta práctica se pudo comprobar el funcionamiento de diferentes métodos para extracción de fondo aplicado a videos, los resultados arrojaron mayor exactitud a la hora de segmentar los objetos en movimiento con los métodos incluidos en Opencv, teniendo pocas variaciones comparados entre ellos, también se pudo observar el método para extracción basado en el cálculo de mínimos cuadrados, en este método fue menor la exactitud con la que se segmentaron los objetos en movimiento, pues aún se conservaron algunos partes del fondo, sin embargo tiene mucho valor pues permite observar el proceso de extracción de fondo de manera mas detallada.

VII. Código Fuente

A. Métodos incluidos en Opencv4.

```
from __future__ import print_function
from glob import glob
import cv2 as cv
import argparse
import matplotlib.pyplot as plt
import numpy as np
import sys
import itertools as it

print("Este programa Modela el fondo de un video, por medio de los métodos\n\
     de substraccion disponibles en Opencv4")
print("Ingrese el video a analizar, seleccione una opcion:")
print("Opcion 1: video preestablecido,  presione --> '1'")
print("Opcion 2: Utilizar la camara de la computadora, presione --> '2'")
print("Opcion 3: Ingresar el nombre del video, presione--> '3'")
opcion=int(input("Ingrese una opcion:__"))

if opcion == 1:
   print ("selecciono opcion 1")
if opcion == 2:
```

```
    print ("selecciono opcion 1")
if opcion == 3:
    nombre=input("Selecciono la opcion 3 \n Ingrese el nombre del video con extension --->")
    input("presione una tecla para continuar")

metodo=int(input("Seleccine el metodo de substraccion de fondo: \n  4)MOG 5)MOG2 6)KNN 7)GMG
8)CNT --->>"))

if metodo==4 :
    print("selecciono el metodo MOG ")
if metodo==5 :
    print("selecciono el metodo MOG2")
if metodo==6 :
    print("selecciono el metodo KNN")
if metodo==7 :
    print("selecciono el metodo GMG")
if metodo==8 :
    print("selecciono el metodo CNT")
input("presione una tecla para continuar")


def inside(r, q):
    rx, ry, rw, rh = r
    qx, qy, qw, qh = q
    return rx > qx and ry > qy and rx + rw < qx + qw and ry + rh < qy + qh

def draw_detections(img, rects, thickness = 1):
    for x, y, w, h in rects:
        # the HOG detector returns slightly larger rectangles than the real objects.
        # so we slightly shrink the rectangles to get a nicer output.
        pad_w, pad_h = int(0.15*w), int(0.05*h)
        cv.rectangle(img, (x+pad_w, y+pad_h), (x+w-pad_w, y+h-pad_h), (0, 255, 0), thickness)


if metodo == 4:
    backSub =cv.bgsegm.createBackgroundSubtractorMOG()
if metodo == 5:
    backSub = cv.createBackgroundSubtractorMOG2()
if metodo == 6:
    backSub = cv.createBackgroundSubtractorKNN()
if metodo == 7:
    backSub =cv.bgsegm.createBackgroundSubtractorGMG()

if metodo == 8:
    backSub =cv.bgsegm.createBackgroundSubtractorCNT()

#Inicia captura de video
if opcion == 1:
    capture = cv.VideoCapture('vtest.avi')
if opcion == 2:
    capture = cv.VideoCapture(0)
if opcion == 3:
    capture = cv.VideoCapture(nombre)

if not capture.isOpened:
    print('Unable to open video ')
```

```
    exit(0)

#Se declara detector de personas
hog = cv.HOGDescriptor()
hog.setSVMDetector( cv.HOGDescriptor_getDefaultPeopleDetector() )


while True:
    ret, frame = capture.read()
    if frame is None:
        break
    #Se obtiene la imagen de fondo y se extrae de la imagen.
    fgMask = backSub.apply(frame)

    #Se aplican operaciones apertura y cerradura a imagen binaria con objetos segmentados.
    kernel = np.ones((5,5),np.uint8)
    fgMask = cv.morphologyEx(fgMask, cv.MORPH_OPEN, kernel)
    fgMask = cv.morphologyEx(fgMask, cv.MORPH_CLOSE, kernel)
    cv.imshow('Extraccion de fondo', fgMask)

    #Se imprime el numero actual de cuadro que esta siendo procesado
    #en la parte superior de el video resultante.
    cv.rectangle(frame, (10, 2), (100,20), (255,255,255), -1)
    cv.putText(frame, str(capture.get(cv.CAP_PROP_POS_FRAMES)), (15, 15),
               cv.FONT_HERSHEY_SIMPLEX, 0.5 , (0,0,0))

    #Se identifican los contorno en la imagen sin fondo.
    contours, hierarchy = cv.findContours(fgMask, cv.RETR_TREE, cv.CHAIN_APPROX_SIMPLE)
    #Se dibujan los contornos en el video de origen.
    image = cv.drawContours(frame, contours, -1, (0, 255, 0), 2)


    #Procesa la deteccion de perosonas y coloca un cuadro en cada posicion detectada
    found, w = hog.detectMultiScale(frame, winStride=(8,8), padding=(32,32), scale=1.05)
    found_filtered = []
    for ri, r in enumerate(found):
        for qi, q in enumerate(found):
            if ri != qi and inside(r, q):
                break
        else:
            found_filtered.append(r)
    draw_detections(frame, found)
    draw_detections(frame, found_filtered, 3)
    print('%d (%d) found' % (len(found_filtered), len(found)))

    #Muestra imagen de origen con los contornos detectados
    cv.imshow('Video origen', frame)


    keyboard = cv.waitKey(30)
    if keyboard == 'q' or keyboard == 27:
        capture.release()
        cv.destroyAllWindows()
        break
```

B. Método mínimos cuadrados.

```
from __future__ import print_function
from __future__ import division
from math import atan2, cos, sin, sqrt, pi
import cv2 as cv
import math
import numpy as np
import argparse


print("Este programa Modela el fondo de un video, por medio del promedio \n\
     de cuadros de video, utilizando el metodo de minimos cuadrados")

print("Ingrese el video a analizar, seleccione una opcion:")
print("Opcion 1: video preestablecido,  presione --> '1'")
print("Opcion 2: Utilizar la camara de la computadora, presione --> '2'")
print("Opcion 3: Ingresar el nombre del video, presione--> '3'")


opcion=int(input("Ingrese una opcion:__"))

if opcion == 1:
   print ("selecciono opcion 1")
if opcion == 2:
   nombre=input("Selecciono la opcion 3 \n, ingrese el nombre del video con extension --->")
   input("presione una tecla para continuar")
input("presione una tecla para continuar")

##############PCA análisis, calculo gradiente y orientación#######################################
def drawAxis(img, p_, q_, colour, scale):
   p = list(p_)
   q = list(q_)
   ## visualización
   angle = atan2(p[1] - q[1], p[0] - q[0]) # Angulo en radianes
   hypotenuse = sqrt((p[1] - q[1]) * (p[1] - q[1]) + (p[0] - q[0]) * (p[0] - q[0]))

   # Se indica el tamaño de la flecha, por un valor de escala
   q[0] = p[0] - scale * hypotenuse * cos(angle)
   q[1] = p[1] - scale * hypotenuse * sin(angle)
   cv.line(img, (int(p[0]), int(p[1])), (int(q[0]), int(q[1])), colour, 1, cv.LINE_AA)

   # Se crea el gancho de la flecha
   p[0] = q[0] + 9 * cos(angle + pi / 4)
   p[1] = q[1] + 9 * sin(angle + pi / 4)
   cv.line(img, (int(p[0]), int(p[1])), (int(q[0]), int(q[1])), colour, 1, cv.LINE_AA)

   p[0] = q[0] + 9 * cos(angle - pi / 4)
   p[1] = q[1] + 9 * sin(angle - pi / 4)
   cv.line(img, (int(p[0]), int(p[1])), (int(q[0]), int(q[1])), colour, 1, cv.LINE_AA)


def getOrientation(pts, img):
   # Se crea un bufer usado durante el analisis PCA
```

```
    sz = len(pts)
    data_pts = np.empty((sz, 2), dtype=np.float64)
    for i in range(data_pts.shape[0]):
        data_pts[i,0] = pts[i,0,0]
        data_pts[i,1] = pts[i,0,1]

    # Se realiza análisis PCA
    mean = np.empty((0))
    mean, eigenvectors, eigenvalues = cv.PCACompute2(data_pts, mean)

    # Se almacena el centro del objeto
    cntr = (int(mean[0,0]), int(mean[0,1]))

    # Se dibujan los componentes principales
    cv.circle(img, cntr, 3, (255, 0, 255), 2)
    p1 = (cntr[0] + 0.02 * eigenvectors[0,0] * eigenvalues[0,0], cntr[1] + 0.02 * eigenvectors[0,1] *
eigenvalues[0,0])
    p2 = (cntr[0] - 0.02 * eigenvectors[1,0] * eigenvalues[1,0], cntr[1] - 0.02 * eigenvectors[1,1] *
eigenvalues[1,0])
    drawAxis(img, cntr, p1, (0, 255, 0), 1)
    drawAxis(img, cntr, p2, (255, 255, 0), 5)

    angle = atan2(eigenvectors[0,1], eigenvectors[0,0]) # orientacion en radianes

    return angle


#############Calculo fondo#########################################################################
#Load Video
if opcion == 1:
    capture = cv.VideoCapture('vtest.avi')

if opcion == 2:
    capture = cv.VideoCapture(nombre)

if not capture.isOpened:
    print('Unable to open: ' )
    exit(0)

#Obtener ancho, altura y #cuadros del video
Vwidth = capture.get(3)
Vheight = capture.get(4)
fps = round(capture.get(5),0)

print(str(Vwidth) + 'x' + str(Vheight) + ' , ' + str(fps) + 'fps')

#Obtener el numero total de cuadros
TotalFrames =  int(capture.get(cv.CAP_PROP_FRAME_COUNT))
print("Total Frames: " ,TotalFrames )


while True:
#Captura los cuadros del video
    ret_ave, frame_ave = capture.read()

#Se obtiene el numero del cuadro actual
```

```
    Current_Frame_POS = int( capture.get(1) )
    Learnstr = "Learning: " + str(Current_Frame_POS) + "/" + str(TotalFrames) + " " +
str(round(Current_Frame_POS/ TotalFrames*100,1)) +'%'
    print(Learnstr)
    if frame_ave is None:
        break

#Se inicia el calculo del promedio de pixeles en cada cuadro
    if Current_Frame_POS == 1:
        avg = np.float32(frame_ave)

#Cálculo del promedio acumulado entre cada cuadro para cada pixel.
    cv.accumulateWeighted(frame_ave, avg, 0.005)
    result = cv.convertScaleAbs(avg)
#Almacenamiento del valor mínimo de cada promedio calculado en la variable
#"result_min"
    np.array(result)
    result_min = np.min(result)
    result_bg=result-result_min
    cv.imshow('Imagen original', frame_ave)
    cv.imshow('Imagen fondo',result_bg)



###########Análisis objetos en movimiento (Segmentación)#####################################

if opcion == 1:
    capture = cv.VideoCapture('vtest.avi')

if opcion == 2:
    capture = cv.VideoCapture(nombre)

if not capture.isOpened:
    print('Unable to open: ' )
    exit(0)

while True:
    ret, frame = capture.read()
    if frame is None:
        break
    #Se convierte la imagen en cada cuadra a escala de grises, y formato binario.
    fgMask=frame-result_bg
    fgMask = cv.cvtColor(fgMask, cv.COLOR_RGB2GRAY)
    fgMask = cv.GaussianBlur(fgMask,(5,5),0)
    _, fgMask = cv.threshold(fgMask, 70, 150, cv.THRESH_BINARY_INV)
    cv.imshow('Eliminacion del fondo',fgMask)
    #Se procesan la imagen de los objetos segmentados mediante
    #funciones de apertura y cerradura.
    kernel = np.ones((5,5),np.uint8)
    fgMask_bin = cv.morphologyEx(fgMask, cv.MORPH_OPEN, kernel)
    fgMask_bin = cv.morphologyEx(fgMask, cv.MORPH_CLOSE, kernel)

    # Encuentra los contornos en la imagen binaria.
    contours, _ = cv.findContours(fgMask_bin, cv.RETR_LIST, cv.CHAIN_APPROX_NONE)

    for i, c in enumerate(contours):
```

```
    # Calcula el área de cada contorno
    area = cv.contourArea(c)
    # Ignora contornos que son demasiado pequeños o demasiado grandes
    if area < 1e2 or 1e5 < area:
        continue
    # Dibuja cada contorno
    cv.drawContours(frame, contours, i, (0, 0, 255), 2)
    # Encuentra la orientación de cada objeto
    getOrientation(c, frame)
#Muestra el video de entrada he incluye los contornos detectados.
cv.imshow('output',frame)

keyboard = cv.waitKey(30)
if keyboard == 'q' or keyboard == 27:
    capture.release()
    cv.destroyAllWindows()
    break
```

VIII. BIBLIOGRAFÍA.

[1] OPENCV, How to Use Background Subtraction Methods, www.opencv.org, https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html, [online], Ultimo acceso: 27 mayo 2020.

[2] OPENCV, Background Subtraction, www.opencv.org, https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_bg_subtraction/py_bg_subtraction.html, Ultimo acceso: 27 mayo 2020.

[3] OPENCV, Introduction to Principal Component Analysis (PCA), www.opencv.org, https://www.ccoderun.ca/programming/doxygen/opencv/tutorial_introduction_to_pca.html, [online], Ultimo acceso: 27 mayo 2020.

[4] Wang, L., Tan, T., Ning, H., & Hu, W. (2003). Silhouette analysis-based gait recognition for human identification. IEEE transactions on pattern analysis and machine intelligence.

[5] OPENCV, Contours : Getting Started, www.opencv.org, https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html, [online], Ultimo acceso: 27 mayo 2020.

[6] OPENCV, Structural Analysis and Shape Descriptors, www.opencv.org, https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=drawcontours, [online], Ultimo acceso: 27 mayo 2020.

[7] OPENCV, Operations on Arrays, www.opencv.org, https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#void%20addWeighted(InputArray%20src1,%20double%20alpha,%20InputArray%20src2,%20double%20beta,%20double%20gamma,%20OutputArray%20dst,%20int%20dtype, [online], Ultimo acceso: 27 mayo 2020.

[8] OPENCV, Motion Analysis and Object Tracking, www.opencv.org, https://docs.opencv.org/2.4/modules/imgproc/doc/motion_analysis_and_object_tracking.html#void%20accumulateSquare(InputArray%20src,%20InputOutputArray%20dst,%20InputArray%20mask, [online], Ultimo acceso: 27 mayo 2020.

[9] OPENCV, cv::PCA Class Reference, www.opencv.org, https://docs.opencv.org/master/d3/d8d/classcv_1_1PCA.html, [online], Ultimo acceso: 27 mayo 2020.

[10] OPENCV, More Morphology Transformations, www.opencv.org, https://docs.opencv.org/2.4/doc/tutorials/imgproc/opening_closing_hats/opening_closing_hats.html, Ultimo acceso: 27 mayo 2020.