UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

SISTEMA PARA REALIZAR EL CONTEO AUTOMÁTICO DE PERSONAS MEDIANTE TÉCNICAS DE IA.

DOCENTE:

Ing. Jorge Cordero

MATERIA:

INTELIGENCIA ARTIFICIAL AVANZADA

ESTUDIANTE:

Cristian Daniel Gaona Sánchez
Luis Alfredo Jaramillo Uday

ABRIL– AGOSTO/2020

Manual de usuario

Configuración del entorno de trabajo

El proyecto está desarrollado con el lenguaje de programación Python y ejecutado una máquina local con sistema operativo Windows 10 pro. Para el desarrollo del proyecto se realizó previamente la instalación de las siguientes herramientas:

Python:

- Descargar Python 3 para Windows 10 https://www.python.org/downloads/
- Ejecutamos el archivo descargado, seleccionamos la primera opción y verificamos que las casillas del final esten marcadas como se observa en la imagen.

Figura 1. Interfaz de instalación de Python

- Una vez que hacemos clin en **install now** empieza la instalación

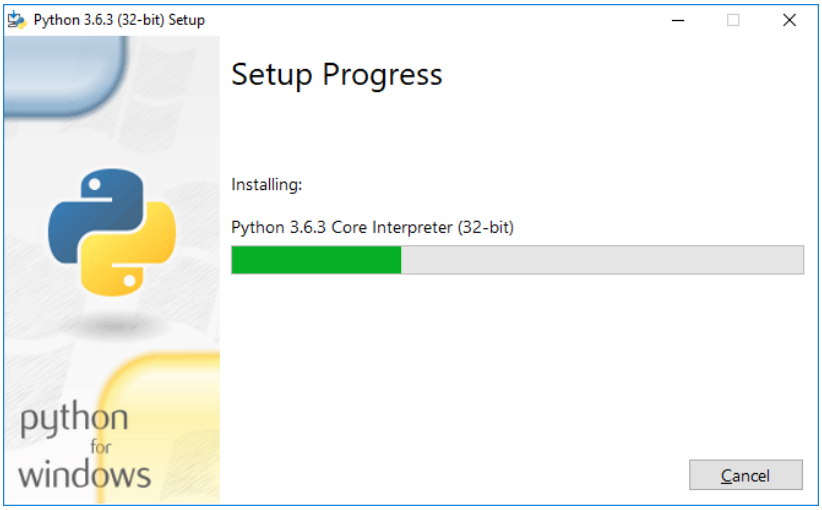

Figura 2. Instalación de Python

- Una vez finalizado la instalación se debe abrir la consola de Windows y se escribe los siguientes comandos para verificar la instalación de Python y Pip

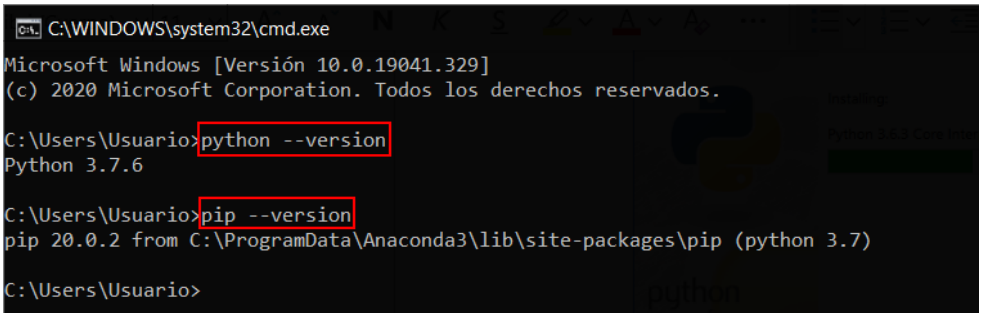

Figura 3. Versión de Python y Pip

Anaconda

- Descargar Anaconda desde el sitio oficial https://www.anaconda.com/products/individual#windows
- Una vez descargado ejecutar el archivo descargado como administrador.
- En el siguiente link se encuentra un tutorial completo para la instalación y configuración de anaconda https://www.datacamp.com/community/tutorials/installing-anaconda-windows
- Anaconda proporciona una suite de librerías instaladas, para visualizar todas las librerías instaladas se ejecuta el siguiente comando desde la shell de windows **conda list,** y se visualizara todas las librerías instaladas.

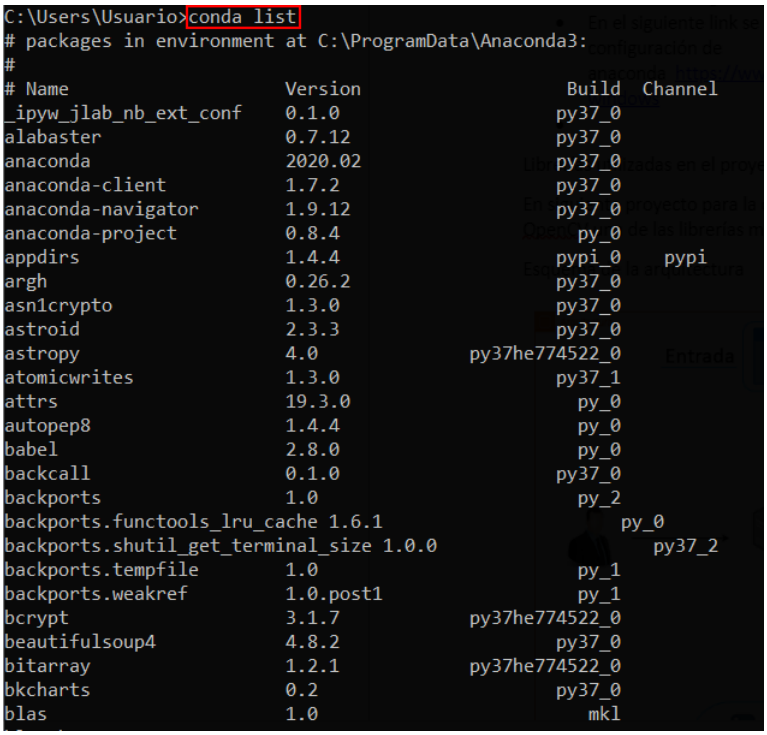

Figura 4. Liberias de Anaconda

- En el caso de no estar una librería dentro de la suite de la anaconda se la puede instalar con el siguiente comando **pip install {nombre_librería}**

Visual Studio Code

- Editor de código multiplataforma para el desarrollo del proyecto, se sugiere descargarlo de su sitio oficial https://code.visualstudio.com/
- Una vez descargado se sugiere ejecutar el instalador como administrador para la configuración automática de las variables de entorno.

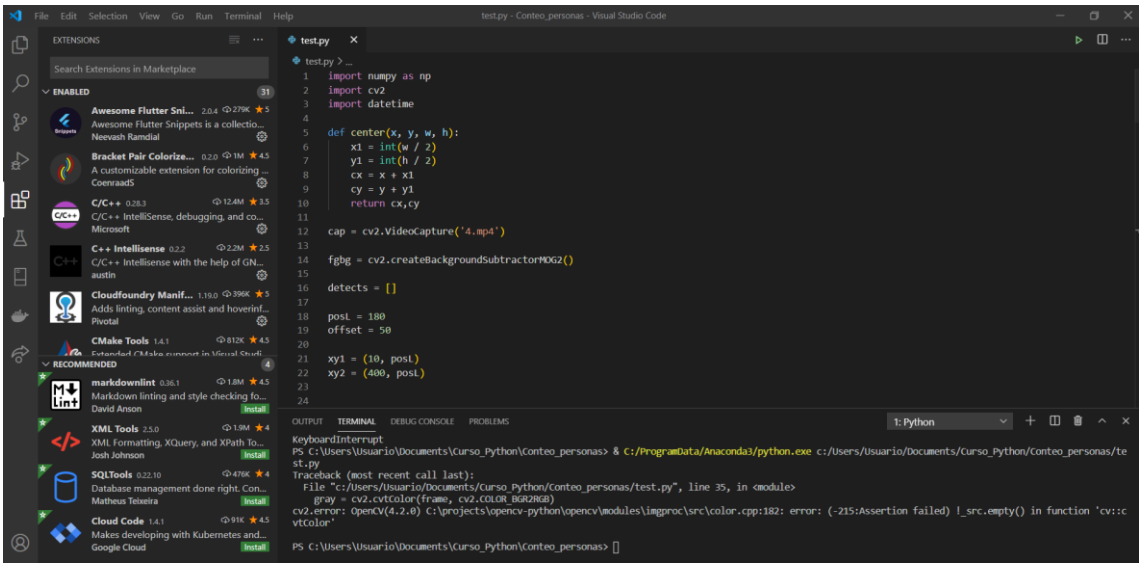

Figura 5. Visual Studio Code - Editor de código

Arquitectura

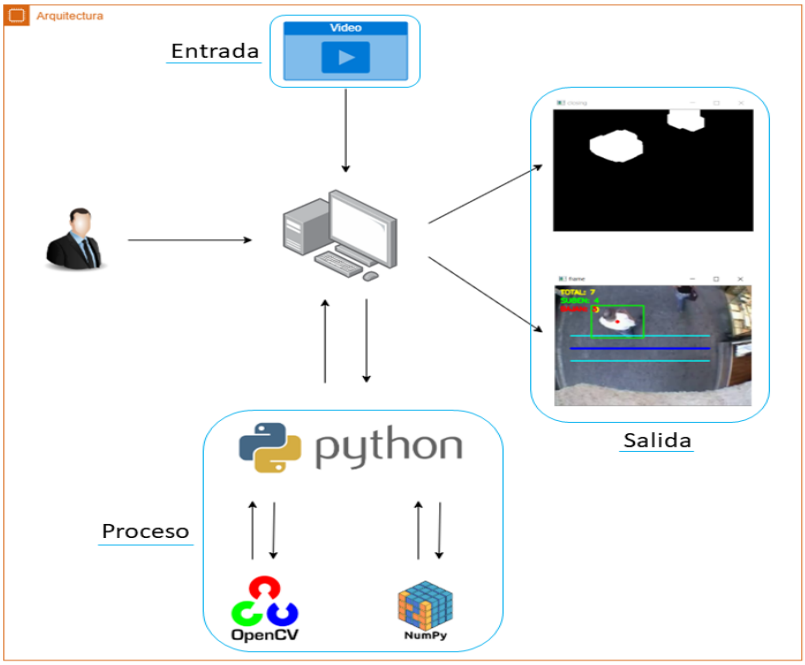

Figura 6. Arquitectura del sistema

Entrada

```python
import numpy as np
import cv2
import datetime

def center(x, y, w, h):
    x1 = int(w / 2)
    y1 = int(h / 2)
    cx = x + x1
    cy = y + y1
    return cx,cy

cap = cv2.VideoCapture('4.mp4')

fgbg = cv2.createBackgroundSubtractorMOG2()

detects = []
```

Figura 7. Datos de entrada.

En esta parte de código se puede observar la importación de las siguientes librerías:

- Numpy es una librería principal para la informática científica, proporciona potentes estructuras de datos.
- OpenCV es una librería que proporciona una suite de algoritmos que con un par de líneas permite hacer uso de los mismos en este caso se ha utilizado substraccioBackground para obtener una imagen binaria que contiene los pixeles que pertenecen a objetos en movimiento, mediante el uso de cámaras estáticas.
- Datatime Liberia que permite obtener el la fecha y hora en tiempo real.

La parte relevante en cuanto a la *entrada* es la lectura del video, este procedimiento lo realizamos por medio de la librería **OpenCV *cv2,*** empleando una de sus propiedades, ***VideoCapture*** la cual nos proporciona la captura de vídeo desde secuencias de imágenes, cámaras o archivos de vídeo.

El video a emplear durante la ejecución del sistema fue tomado desde una perspectiva alta para lograr un mejor campo de visión y además de un mejor desempeño, en cuanto al conteo de personas se refiere.

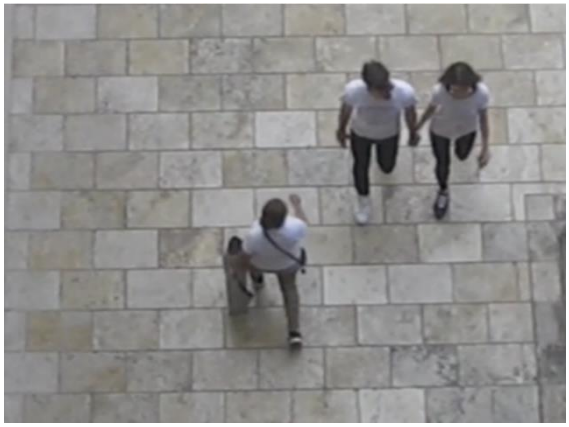

Figura 8. Perspectiva empleada para la ejecución del sistema

Proceso

```
fgbg = cv2.createBackgroundSubtractorMOG2()

detects = []

posL = 180
offset = 30

xy1 = (20, posL)
xy2 = (400, posL)
```

Figura 9. Creación del fondo de sustracción

Durante el proceso del sistema, se realizará en una primera instancia la sustracción de fondo la cual nos permite generar una máscara de primer plano, es decir, una imagen binaria del video que contiene los píxeles que pertenecen a objetos en movimiento dentro de la escena captada, esto lo realizamos por medio de la propiedad de OpenCV denominada ***createBackgroundSubtractorMOG2()***

```
cv2.line(frame,xy1,xy2,(0,0,0),3)

cv2.line(frame,(xy1[0],posL-offset),(xy2[0],posL-offset),(255,255,0),2)

cv2.line(frame,(xy1[0],posL+offset),(xy2[0],posL+offset),(255,255,0),2)

contours, hierarchy = cv2.findContours(dilation,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
```

Figura 10. creación de umbrales

Seguido a esto realizamos la implementación de un total de tres líneas de reconocimiento, las cuales nos permitirán realizar el conteo de las personas, que se desplazan de arriba hacia abajo o viceversa, además de un total de personas reconocidas, esto lo realizamos por medio de varios

parámetros, en primer lugar, especificamos la locación de estas líneas, esto se logra por medio de ***(posL, offset, xy1, xy2)***, una vez establecida su ubicación realizamos su trazado, especificando tanto su color, como su ancho, por medio de la propiedad ***line()***.

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

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)

fgmask = fgbg.apply(gray)

retval, th = cv2.threshold(fgmask, 200, 255, cv2.THRESH_BINARY)

kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))

opening = cv2.morphologyEx(th, cv2.MORPH_OPEN, kernel, iterations = 2)

dilation = cv2.dilate(opening,kernel,iterations = 8)

closing = cv2.morphologyEx(dilation, cv2.MORPH_CLOSE, kernel, iterations = 8)
cv2.imshow("closing", closing)
```

Figura 11. Método de substracción de fondo

La función cv.threshold se usa para aplicar el umbral. El primer argumento es la imagen de origen, que debería ser una imagen en escala de grises. El segundo argumento es el valor umbral que se utiliza para clasificar los valores de píxeles. El tercer argumento es el valor máximo que se asigna a los valores de píxeles que exceden el umbral. OpenCV proporciona diferentes tipos de umbrales que viene dado por el cuarto parámetro de la función. El umbral básico como es cv.THRESH_BINARY.

Para una compresión mas clara acerca del kernel, opening, dilation y closing se sugiere revisar la siguiente documentación de OpenCV https://bit.ly/3e99EdT

```
(x,y,w,h) = cv2.boundingRect(cnt)

area = cv2.contourArea(cnt)

if int(area) > 3000 :
    centro = center(x, y, w, h)

    cv2.putText(frame, str(i), (x+5, y+15), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255),2)
    cv2.circle(frame, centro, 4, (0, 0,255), -1)
    cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
    if len(detects) <= i:
        detects.append([])
    if centro[1]> posL-offset and centro[1] < posL+offset:
        detects[i].append(centro)
    else:
        detects[i].clear()
    i += 1
```

Figura 12. detección y reconocimiento de movimiento

La detección y reconocimiento de movimiento, es una de las piezas fundamentales para lograr un conteo de personas optimo dentro del desarrollo del sistema, su representación gráfica se logró obtener por medio de la propiedad ***boundingRect*** de ***OpenCv***, que nos permite dibujar un rectángulo aproximado alrededor de la imagen binaria, resaltando la región de interés.

Además de poder representar gráficamente el rectángulo, también se procedió al cálculo de su área, el cual nos permitirá obtener el centroide que nos ayudará a saber si se trata de una o varias personas en un mismo lugar, esto gracias a la propiedad ***contourArea*** de ***OpenCV.***

Para un mejor desempeño en la detección se optó por limitar el reconocimiento del contorno del rectángulo, ignorando reconocimientos encontrados que posean un tamaño inferior al especificado.

```
for (c,l) in enumerate(detect):


    if detect[c-1][1] < posL and l[1] > posL :
        detect.clear()
        up+=1
        total+=1
        cv2.line(frame,xy1,xy2,(0,255,0),5)
        continue

    if detect[c-1][1] > posL and l[1] < posL:
        detect.clear()
        down+=1
        total+=1
        cv2.line(frame,xy1,xy2,(0,0,255),5)
        continue

    if c > 0:
        cv2.line(frame,detect[c-1],l,(0,0,255),1)
```

Figura 13. Número de personas suben, bajan y total

Una vez que hemos logrado detectar a los individuos, dentro de las especificaciones anteriormente explicadas, procedemos a la actualización y cálculo de las personas que pasen por las tres líneas explicadas anteriormente, de estas líneas dos nos ayudaran a saber si una de persona entra o sale (sube o baja) de un lugar en específico y una última línea que nos ayudara al conteo total.

```
cv2.putText(frame, "TOTAL: "+str(total), (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255),2)

cv2.putText(frame, "SUBEN: "+str(up), (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0),2)
cv2.putText(frame, "BAJAN: "+str(down), (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255),2)
cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p"),
            (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

cv2.imshow("IA Avanzada Luis - Cristian", frame)

if cv2.waitKey(30) & 0xFF == ord('q'):
    break
```

Figura 14. Presentación de datos

Para finalizar se realizó la representación en pantalla de los resultados obtenidos durante el reconocimiento, especificando el total de personas que entran (suben) y que salen (suben), además de un total de personas reconocidas.

Salida

Figura 15. Detección de personas

La salida de datos se la realiza por el momento de manera automática durante la ejecución del video en análisis, no se la almacena en ningún archivo o base datos, sin embargo, se tiene previsto su almacenamiento en una próxima entrega del sistema.

RESULTADOS

Tabla de Resultados Videos Propios

Tabla 1. Resultados videos obtenidos por el estudiante

Vídeos	Conteo Manual			Conteo Automático			Precisión
	Entrada	Salida	Total	Entrada	Salida	Total	%
Vídeo 1	6	1	7	6	1	7	100%
Vídeo 2	19	15	34	19	12	31	91%

Tabla videos propuestos por el docente

Tabla 2. Videos propuestos por el docente

Vídeos	Conteo Manual			Conteo Automático			Precisión
	Entrada	Salida	Total	Entrada	Salida	Total	%
Vídeo 1	3	7	10	3	6	9	90%
Vídeo 2	12	11	23	12	11	23	85%
Vídeo 3	4	4	8	4	3	7	87.5%
Video 4	3	2	5	2	2	4	80%

Tanto en la tabla 1 y 2 se presenta los resultados de los videos puestos a prueba con el programa desarrollado para detectar la precisión que presenta el programa con cada video puesto a prueba.

Para la ejecución del programa se debe ejecutar el archivo demo.py el mismo que contiene un menú para seleccionar el video que se desea ejecutar, mismo que ya contiene la respectiva configuración para alcanzar una precisión alta. En la figura 16 se observa el menú del programa.

```
def menu():
    print("----------- Vídeos de prueba -----------")
    print("1. Prueba video 1")
    print("2. Prueba video 2")
    print("3. Prueba video 3")
    print("4. Prueba video 4")
    print("----------------------------------")
```

Figura 16. Menú principal del programa

Para la prueba con cada uno de los videos se ha realizado modificación en las variables, tanto para la separación de líneas como y la posición de estas para adaptarlas de acuerdo con el escenario de cada video.

```
#Separación de lineas
    posL = 165
    offset = 20
    xy1 = (5, posL)
    xy2 = (600, posL)  # 400 600
```

El video 1 se obtuvo una mejor precisión con el 90%

Figura 17. Video de prueba 1

En relación con el video 1 se obtiene una precisión optima en donde 1 de 10 personas no es contada por el programa, en el video de prueba hay un escenario en donde pasas dos personas agarradas de la mano y el programa en la parte del método de substracción de fondo crea una sola imagen en escala de grises de una, cuando lo conveniente es crear dos imágenes y es por esa razón que el programa crea un solo centroide para las dos personas.

En el video 2 se obtiene una precisión del 85%

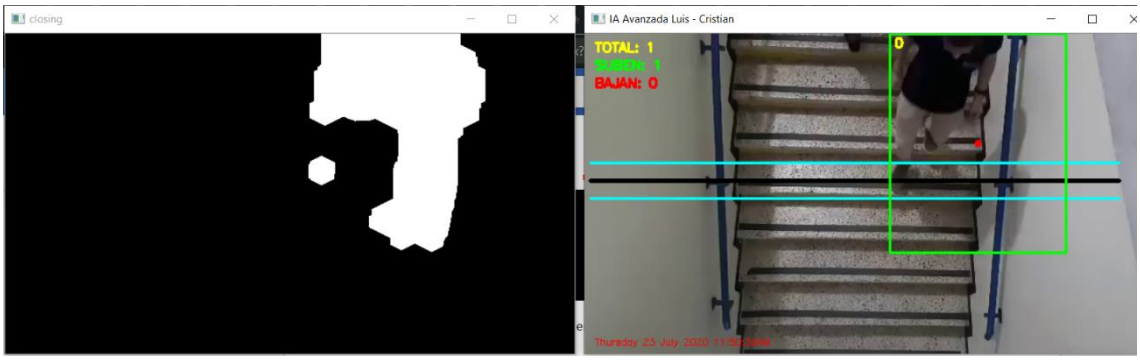

Figura 18. Video de prueba 2

El video 2 no se obtiene una buena precisión, porque la cámara se encuentra en constante movimiento además en este escenario se detecta falsos positivos, debido a que al detectar objetos en movimiento el programa detecta a la sobra de las personas como una persona más y la agrega al conteo de personas de acuerdo al umbral que pase el centroide que se detecta en la sombra, es por tal razón que para este tipo de escenarios, además de modificar las variables para la posición y desplazamiento de los umbrales, se debe modificar las variables de trasformación morfológicas y de este modo obtener una precisión optima con este tipos de

escenarios. A continuación, se coloca el bloque de código que debería ser modificado y el siguiente enlace explica en que cosiste cada una de estas variables https://bit.ly/3e99EdT.

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2LAB)

        fgmask = fgbg.apply(gray)

        retval, th = cv2.threshold(fgmask, 200, 255,
cv2.THRESH_BINARY)

        kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (4, 4))
# 4 4

        opening = cv2.morphologyEx(
            th, cv2.MORPH_OPEN, kernel, iterations=2)  # 2

        dilation = cv2.dilate(opening, kernel, iterations=1)  # 1

        closing = cv2.morphologyEx(
            dilation, cv2.MORPH_CLOSE, kernel, iterations=1)  # 1
        cv2.imshow("closing", closing)
```

El video 3 se obtiene un 87.5 %

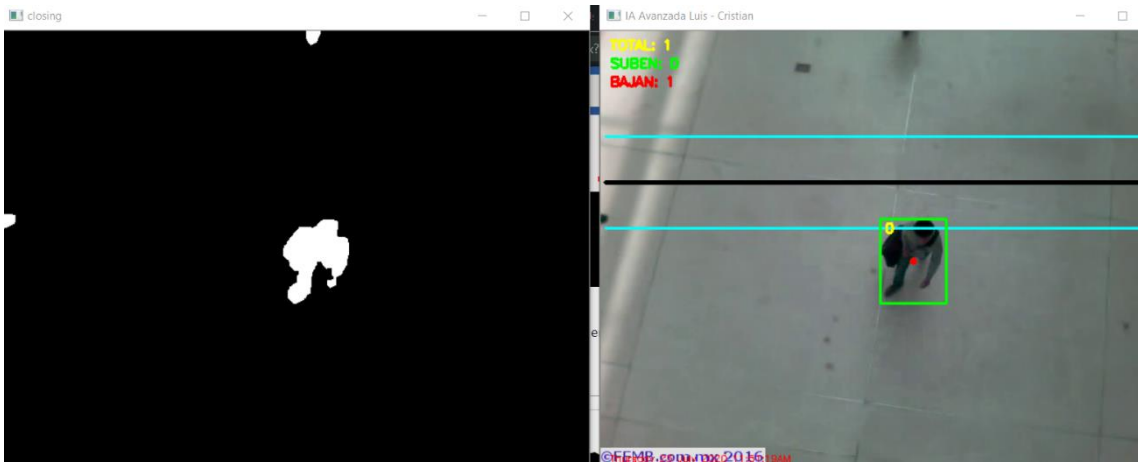

Figura 19. Video prueba 3

En este tipo de escenario de obtiene una precisión adecuada, la razón por la cual no se llega a obtener una precisión optima es porque el área del objeto a detectar debe cumplir un requisito que debe ser mayor a un valor de 2700 y el objeto a detectar no cumple con el requisito el programa le asignará un centroide y no lo tomará en cuenta en el conteo de personas.

El video 4 tiene una precisión del 80%

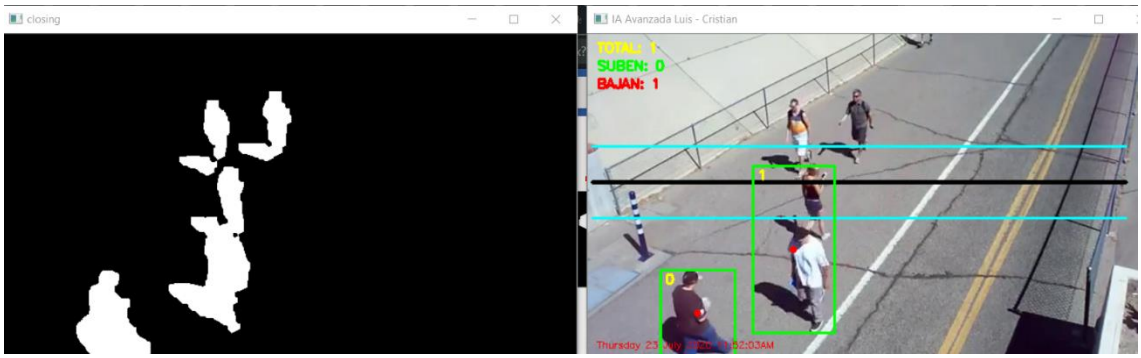

Figura 20. Video prueba 4

En este escenario se puede observar que existe un cierto grado de luz solar por lo que se refleja la sobra de las personas en la calle lo cual causa cierta confusión en el programa, la razón es porque genera imágenes en escala de grises las cuales se unen entre sí con otras imágenes y así el programa lo toma como una sola personas, pero todo esto puede ser modifica en las variables de trasformación morfológicas y en las variables de la definición de umbrales.

Todos los datos de las personas que entran y salen se almacena en un archivo .txt con el nombre del video correspondiente, en donde se almacena la fecha, hora, personas que entran, personas que salen y el total de personas que pasan por el umbral.

Nota: Todos los videos utilizados en el sistema de conteo de personas están alojados el siguiente repositorio
https://github.com/CristianGaona/Sistema_Conteo_Personas/tree/master/Videos

Conclusiones:

- Un sistema de conteo de personas implementado un local de eventos permite saber la capacidad disponible de un local y mediante el sistema controlar el numero de personas que entran y salen del lugar, de esta manera no sobrepasar la capacidad máxima que dispone un local de eventos, en caso de que supere el numero de personas configurar el sistema para que emita un tipo de alerta.
- En el sistema se ha implementado tecnología que se puede acceder fácilmente y de bajo costo, porque se ha utilizado tecnología OpenSource y para el conteo de personas solo se necesita una cámara.
- Los resultados obtenidos demuestran que el sistema puede ser implementado en cualquier lugar, en donde la cámara este ubicada en la parte superior de la entrada al local.
- Es necesario que, la luminosidad en la zona de interés sea la adecuada, ya que cualquier cambio en esta hace que el error aumente.