

UNIVERSIDAD NACIONAL
AUTONOMA DE MEXICO

PROYECTO FINAL IMAGENES DE ENERGÍA DE MOVIMIENTO

Díaz Guerrero Alan Mauricio 316166496

García V. Karla Paulina 316173670

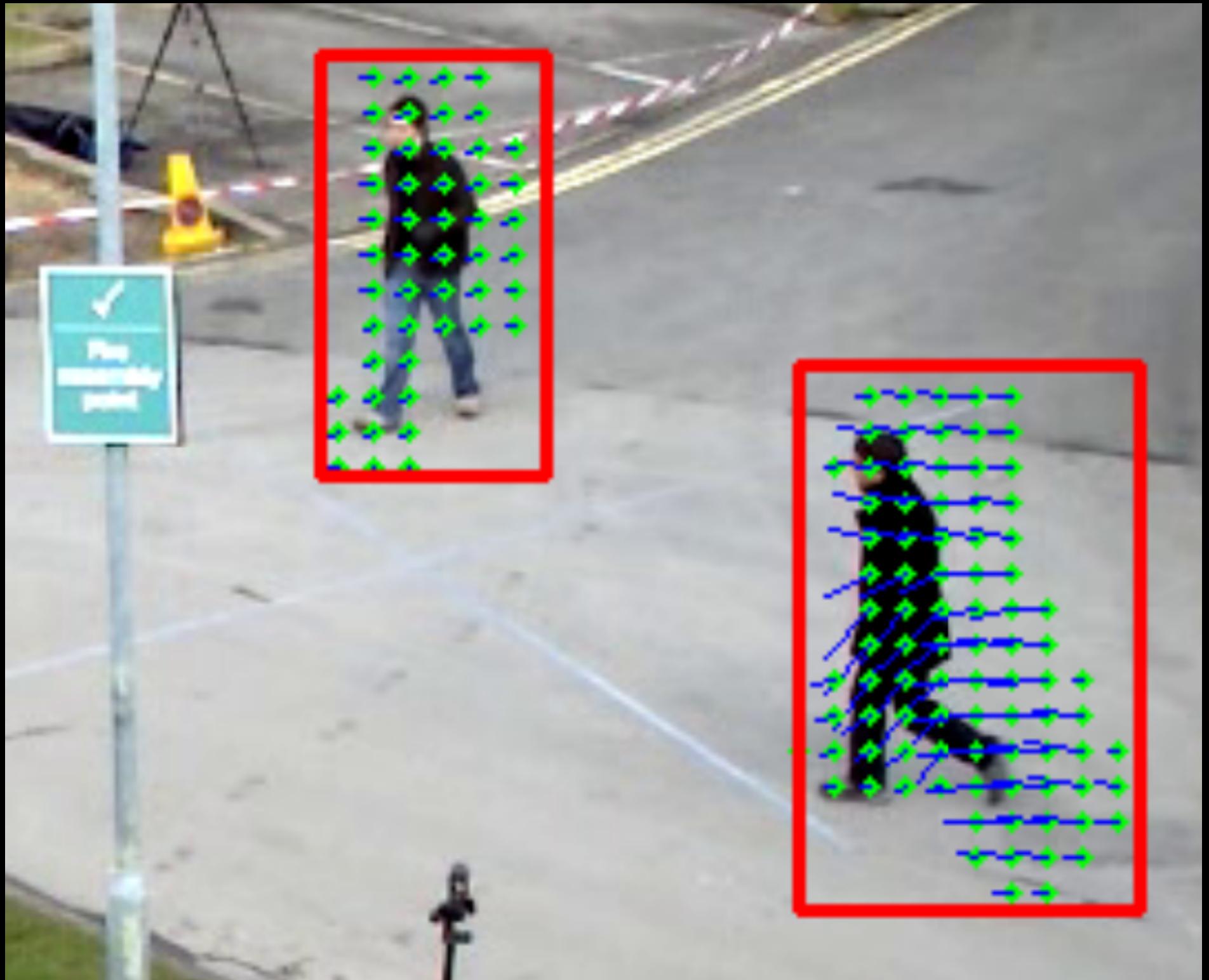
Zamora Morales Ángel Adrián 316252272

PROCESAMIENTO DIGITAL DE IMAGENES

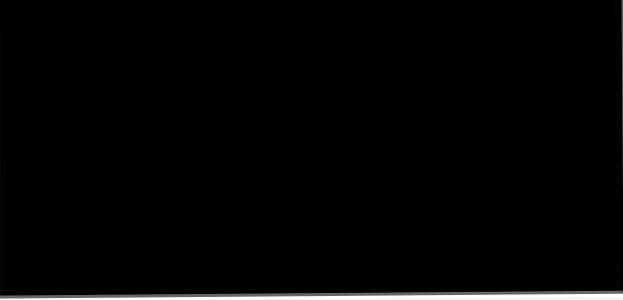


Flujo Óptico

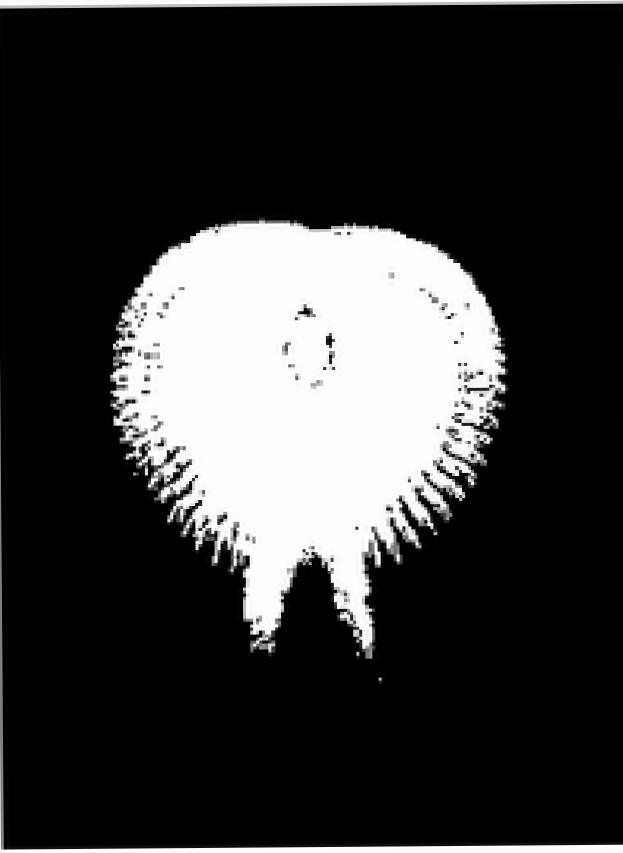
Es el patrón del movimiento aparente de los objetos, superficies y bordes en una escena causado por el movimiento relativo entre un observador (un ojo o una cámara) y la escena.



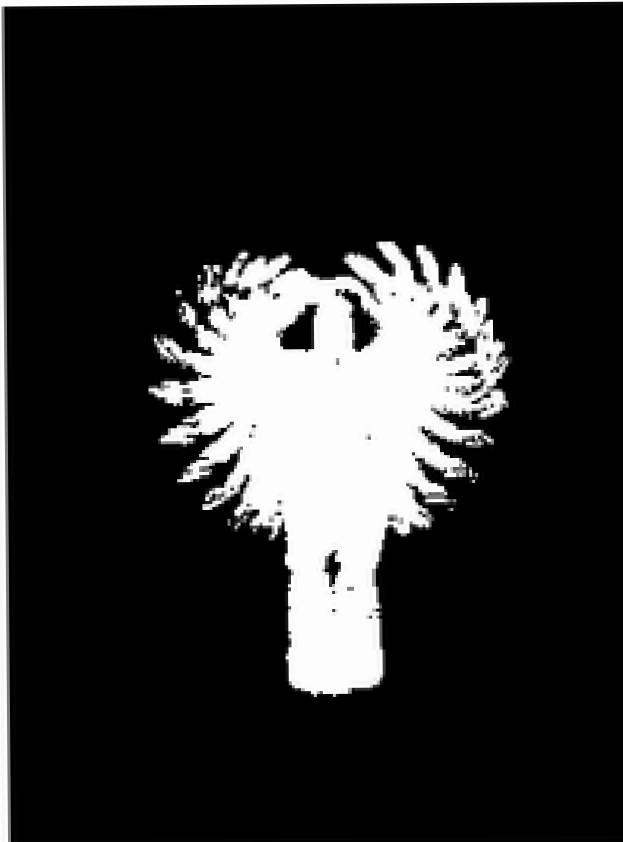
Move 2



Move 4



Move 17



Imágenes de energía de movimiento MEI

Se han consolidado como herramientas visuales esenciales que capturan de manera intrínseca y dinámica el movimiento presente en secuencias de video. Estas imágenes se distinguen por su enfoque en plantillas binarias, asignando a cada píxel valores discretos de 0 o 1.

Imágenes de Historia de Movimiento (MHI)

Técnica utilizada en visión por computadora para capturar y representar la información de movimiento en secuencias de imágenes. Al igual que el flujo óptico, el MHI se centra en resaltar patrones de movimiento en el tiempo.



METODO DE FARNEBACK

El método de Farneback es un algoritmo utilizado en el procesamiento digital de imágenes para calcular el flujo óptico entre dos imágenes consecutivas en una secuencia de video.

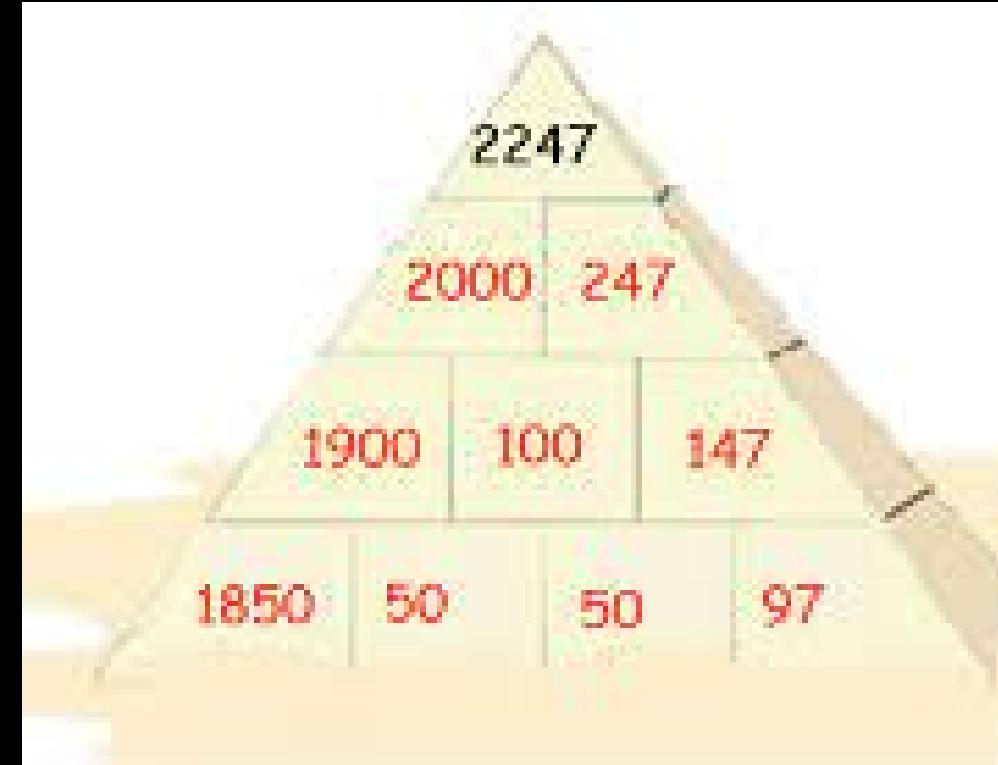


El flujo óptico se refiere al patrón de movimiento aparente de los puntos en una secuencia de imágenes. El método de Farneback es un enfoque denso, lo que significa que estima el flujo óptico para cada píxel en la imagen.



Cálculo de Pirámides:

- El algoritmo opera en múltiples escalas para manejar diferentes niveles de detalle en la imagen.
- Crea pirámides de imágenes mediante reducción y expansión sucesivas.



Cálculo de Campos de Desplazamiento:

- Estima el campo de desplazamiento (flujo) entre las imágenes de la pirámide.
- Utiliza una aproximación polinómica para modelar las variaciones locales de intensidad entre las imágenes.



Interpolación y Refinamiento:

- Interpolación del campo de desplazamiento a una resolución más alta.
 - Refina iterativamente el campo de desplazamiento mediante un enfoque de corrección.

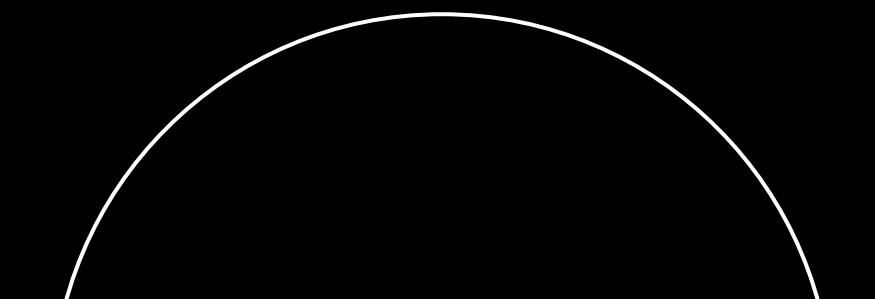
Resultados:

- Proporciona como salida el campo de desplazamiento para cada píxel en términos de cambios en las coordenadas x e y.

DISTRIBUCIÓN DEL PROYECTO

MÉTODO OPENCV

Video original 1





IMPORTACIÓN DE MEDIOS Y DECLARACIÓN DE VARIABLES

```
# Importando el video original
vc = cv2.VideoCapture("dance2.mp4")

fourcc = cv2.VideoWriter_fourcc(*'mp4v') # Generando el archivo donde se guardará el video
out = cv2.VideoWriter('dense_optical_flow_output.mp4', fourcc, 20.0, (1000, 562))

"""Leyendo el primer frame y sacando medidas"""
# Read first frame
_, first_frame = vc.read() #Leyendo primer frame
# Redimensionando fotograma
resize_dim = 1000
max_dim = max(first_frame.shape)
scale = resize_dim/max_dim
first_frame = cv2.resize(first_frame, None, fx=scale, fy=scale)
# Convierte el primer cuadro a escala de grises
prev_gray = cv2.cvtColor(first_frame, cv2.COLOR_BGR2GRAY)
# Creando mascara
mask = np.zeros_like(first_frame)
# Poniendo la saturacion en el valor maximo
mask[..., 1] = 255
i=0

"""Generando variables varias"""
output_folder="frames_opticflow/" #Carpeta para fotogramas
kernel=np.ones((11,11),np.uint8) #Kernel de limpieza
mei_accumulator = np.zeros_like(prev_gray, dtype=np.float32)
mei_accumulator2 = np.zeros_like(prev_gray, dtype=np.float32)
flow2 = np.zeros((562, 1000)) #Matriz para el MEI
flow3 = np.zeros((562, 1000)) #Matriz para el MHI
```

GENERACION DEL OPTICAL FLOW

```
while(vc.isOpened() and _==True): #Bucle mientras el video tenga contenido
    # Leyendo cada frame
    _, frame = vc.read()

    # Redimensionando el frame y convirtiendo a escala de grises
    gray = cv2.resize(frame, None, fx=scale, fy=scale)
    gray = cv2.cvtColor(gray, cv2.COLOR_BGR2GRAY)

    cv2.imshow('gray',gray)      #Desplegando imagen original en gris
    """Metodo de Farneback"""
    # Calcular el flujo optico usando el metodo de Farneback
    flow = cv2.calcOpticalFlowFarneback(prev_gray, gray, None, pyr_scale = 0.5,
                                         levels = 3, winsize = 15,
                                         iterations = 3, poly_n = 5,
                                         poly_sigma = 1.2, flags = 0)

    filename = f'{output_folder}opticflow_{i}.png' #Guardando cada fotograma como imagen
    cv2.imwrite(filename, flujo_to_image(flow))

    """Calculo del MEI RGB"""
    # Calcular la magnitud y angulo de cada vector del Flujo Optico
    magnitude, angle = cv2.cartToPolar(flow[... , 0], flow[... , 1])

    # Ajustando el color del pixel dependiendo de la direccion
    mask[... , 0] = angle * 180 / np.pi / 2

    # Normalizando la imagen
    mask[... , 2] = cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX)

    # Convirtiendo el valor HSV a RGB
    rgb = cv2.cvtColor(mask, cv2.COLOR_HSV2BGR)
```

```
# Calcular la magnitud y angulo de cada vector del Flujo Optico
magnitude, angle = cv2.cartToPolar(flow[... , 0], flow[... , 1])

# Ajustando el color del pixel dependiendo de la direccion
mask[... , 0] = angle * 180 / np.pi / 2

# Normalizando la imagen
mask[... , 2] = cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX)

# Convirtiendo el valor HSV a RGB
rgb = cv2.cvtColor(mask, cv2.COLOR_HSV2BGR)

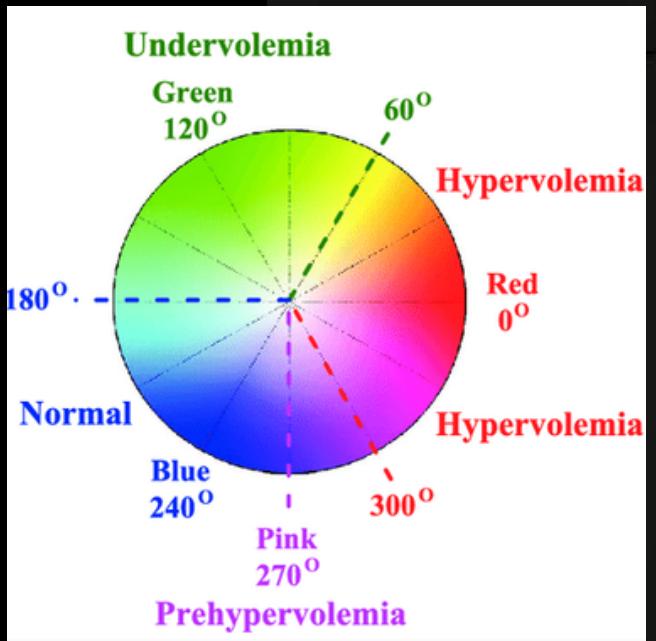
# Ajustar la intensidad de la imagen original para hacerla más oscura
factor_atenuacion = 0.1 # Ajusta este valor según tus preferencias
frame_oscuro = frame.astype(float) * factor_atenuacion

# Redimensionando frame
frame = cv2.resize(frame_oscuro.astype(np.uint8), None, fx=scale, fy=scale)

# Desplegando imagenes/video
dense_flow = cv2.addWeighted(frame, 1,rgb, 2, 0)
cv2.imshow("Dense optical flow", dense_flow)
```



VIDEO OPTICAL FLOW RGB



GENERACION DEL MEI

```
"""Generacion MEI imagen"""
if 50 <= i <= 80:
    flow2=magnitude+flow2 # Acumulando el valor del OF
    imagen_escalada = cv2.normalize(flow2, None, 0, 255, cv2.NORM_MINMAX) #Normalizando

    # Convertir la imagen a tipo uint8
    imagen_uint8 = imagen_escalada.astype(np.uint8)

    # Mostrar y guardar la imagen con cv2
    cv2.imshow('Imagen', imagen_uint8)
    cv2.imwrite('MEI.png',imagen_uint8)

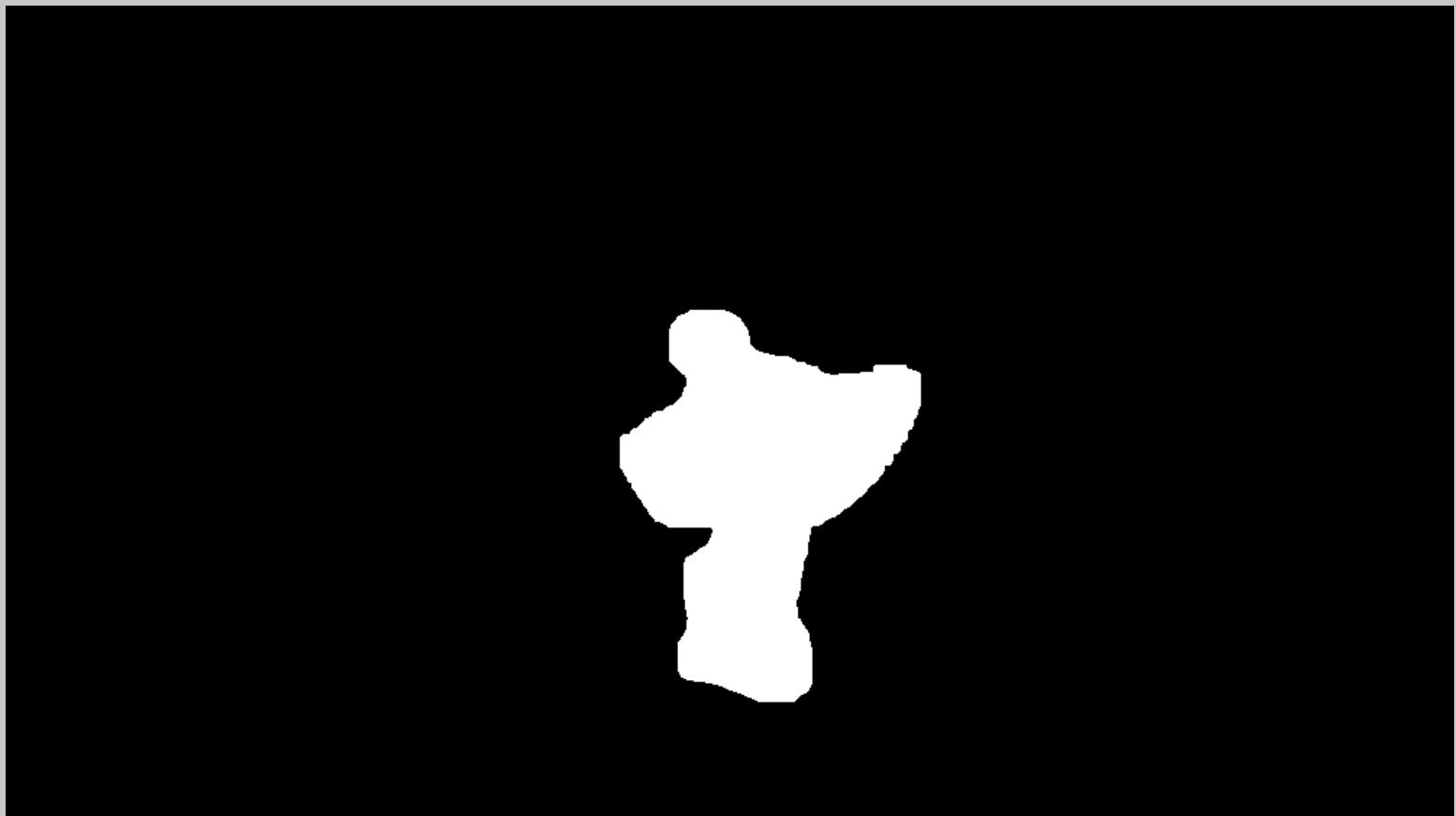
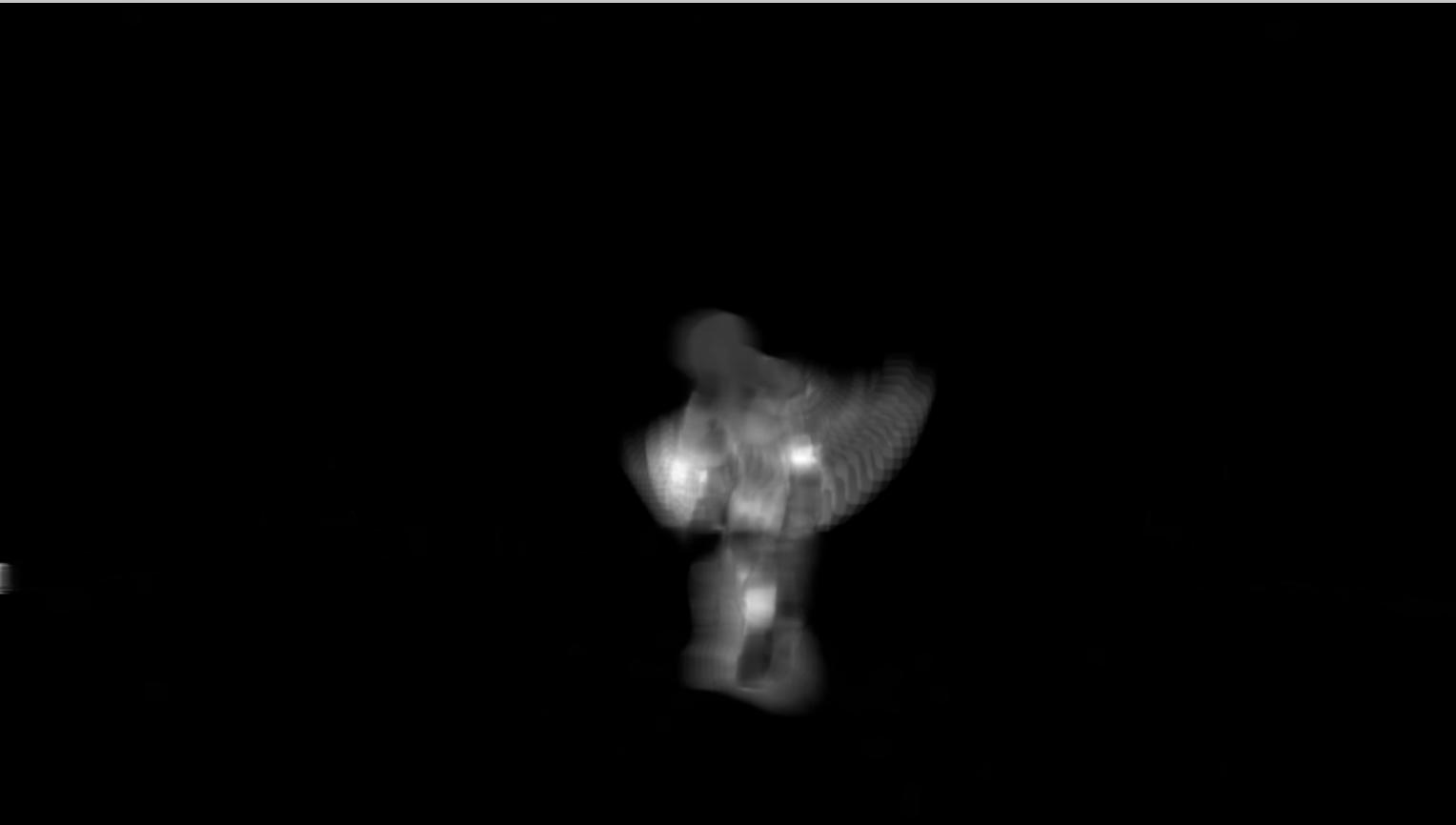
"""Fuera del bucle"""

"""Limpieza del MEI"""
imagen_erodida = cv2.erode(imagen_escalada, kernel, iterations=2) #Erosion del pixel
imagen_escalada = cv2.dilate(imagen_erodida, kernel, iterations=2) #Dilatacion del pixel
for x in range(562): #Altura
    for y in range(1000): #Ancho
        if imagen_escalada[x,y] >= 23: #Conversion de pixeles mayores a 23 a blanco
            imagen_escalada[x,y] = 255

        else: #El resto se hace 0
            imagen_escalada[x,y] = 0

cv2.imwrite('MEI binaria2.png',imagen_escalada) # Desplegando MEI
```

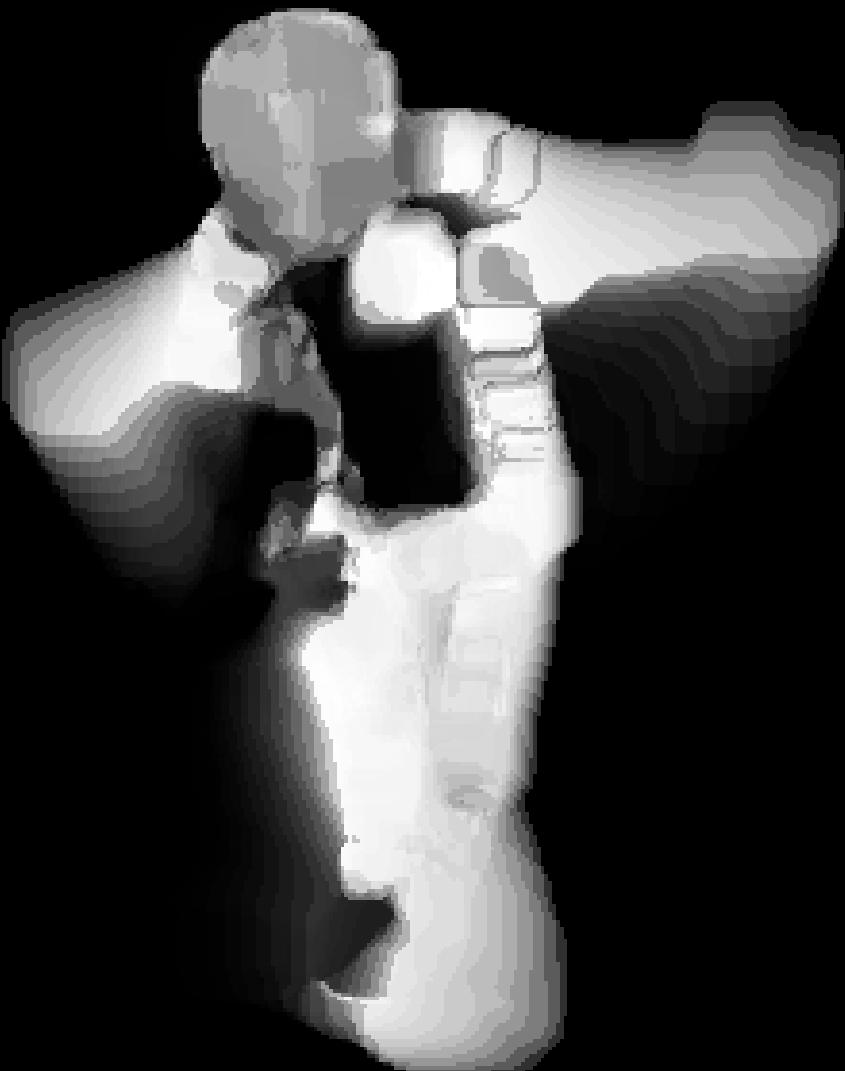
ACUMULACIÓN DEL FLUJO OPTICO



**MOTION ENERGY
IMAGE**

GENERACIÓN DEL MOTION HISTORY IMAGES (MHI)

```
if 50 <= i <= 80:  
    magnitude2=cv2.normalize(magnitude, None, 0, 255, cv2.NORM_MINMAX)  
  
    # Aplica la condición a cada elemento  
    condicion = magnitude2 >= 18  
    condicion2 = magnitude2 < 18  
    # Establece a 255 solo los elementos que cumplen la condición  
    magnitude2[condicion] = 255  
    magnitude2[condicion2] = 0  
  
#Acumulacion de los arreglos  
flow3=magnitude2+flow3*.8  
imagen_escalada2 = cv2.normalize(flow3, None, 0, 255, cv2.NORM_MINMAX)  
  
# Convertir el arreglo a imagen  
imagen_uint82 = imagen_escalada2.astype(np.uint8)  
  
# Mostrar la imagen con cv2  
cv2.imshow('Imagen2', imagen_uint82)  
cv2.imwrite('MHI.png',imagen_uint82)  
  
out.write(dense_flow) #Creacion de video de imagen original + optic flow rgb  
prev_gray = gray # Actualizando frame
```





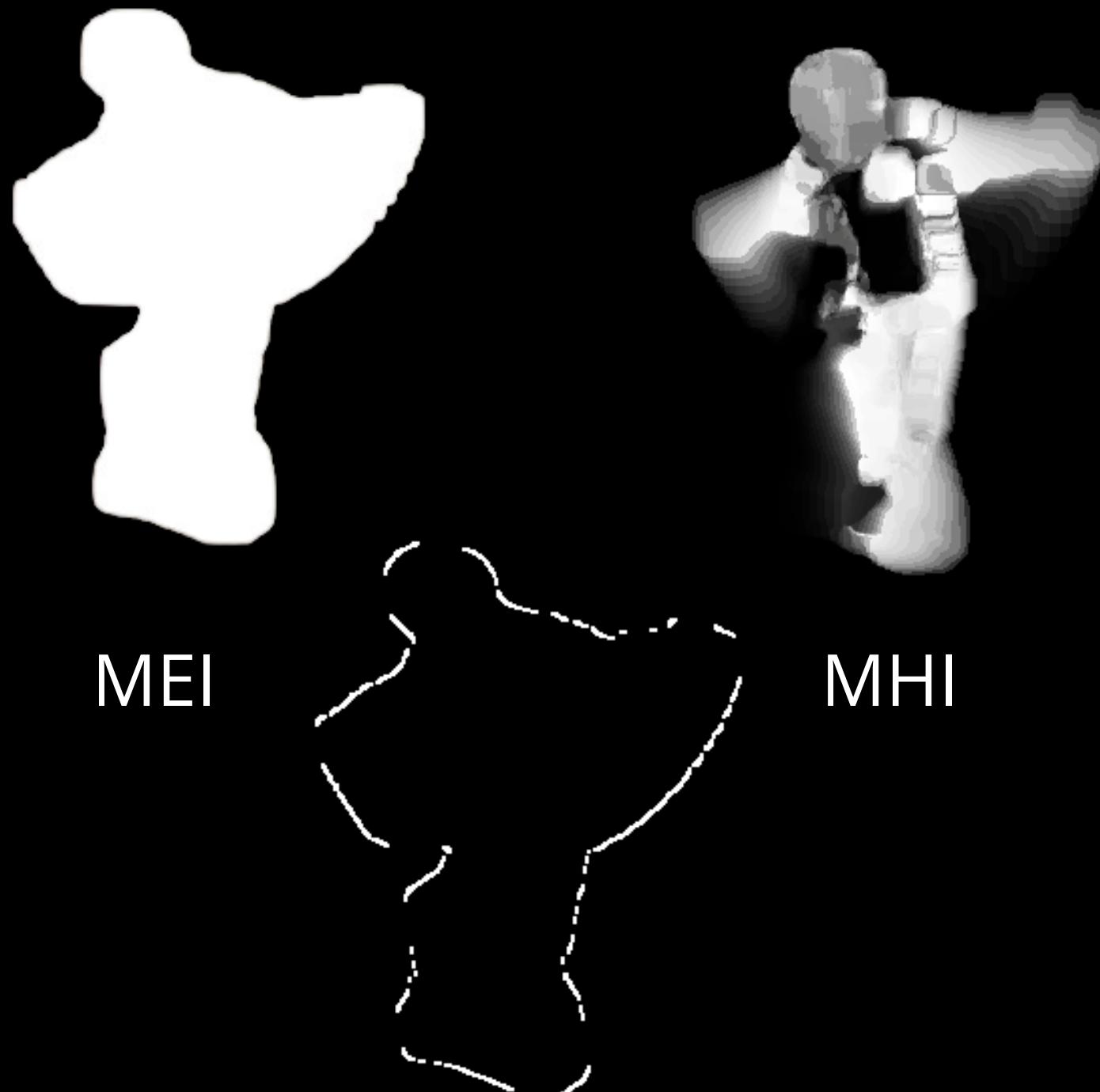
CONTORNO DEL MEI USANDO FILTRO DE SOBEL

```
"""Contorno del MEI/SOBEL"""
sobel=cv2.Sobel(imagen_escalada, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5)
cv2.imwrite('ContornoDelMEI.png',sobel)

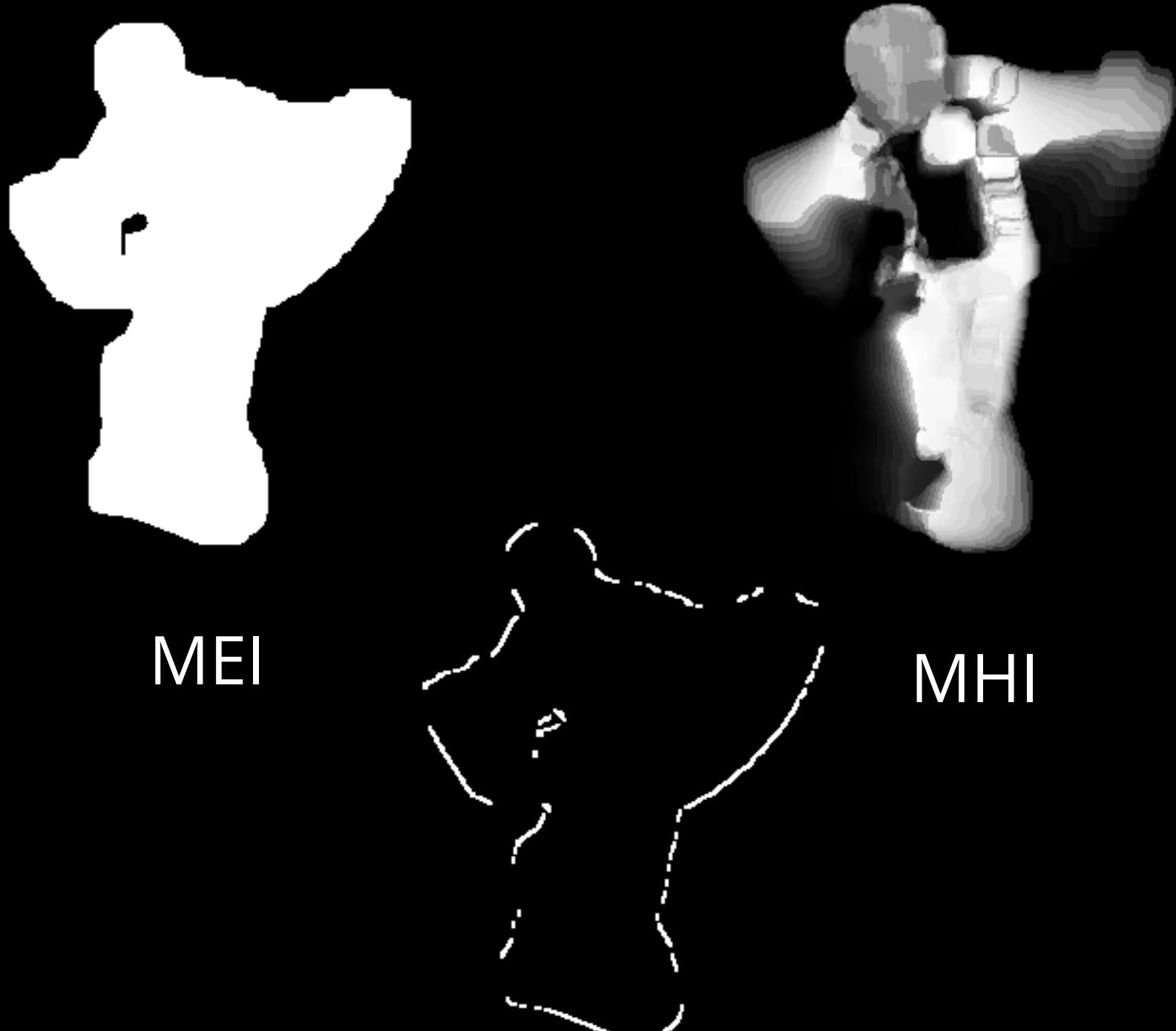
#Liberando el video y cerrando pestañas
vc.release()
cv2.destroyAllWindows()
```

Resultados

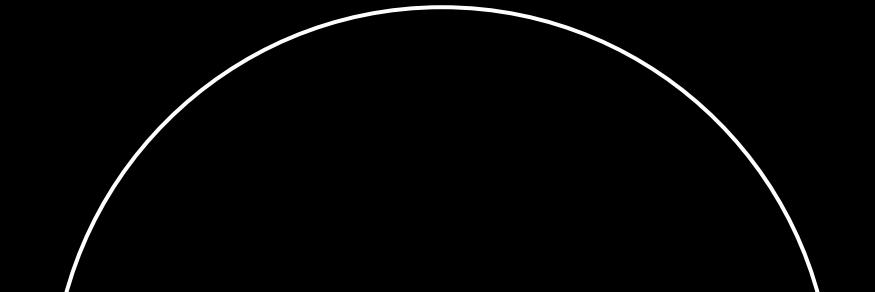
30 FPS



20 FPS



Video original 2



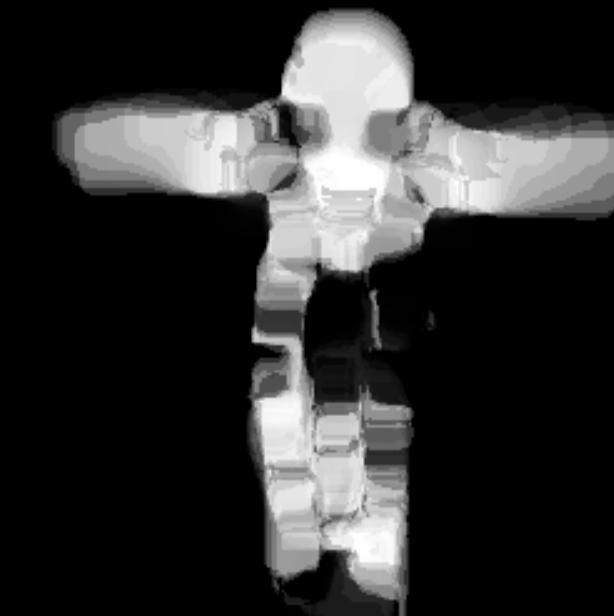


Resultados

30 FPS



MEI



MHI

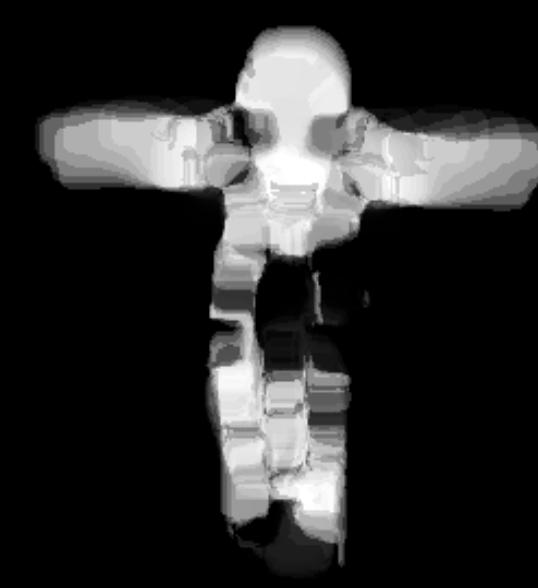


Contorno

20 FPS



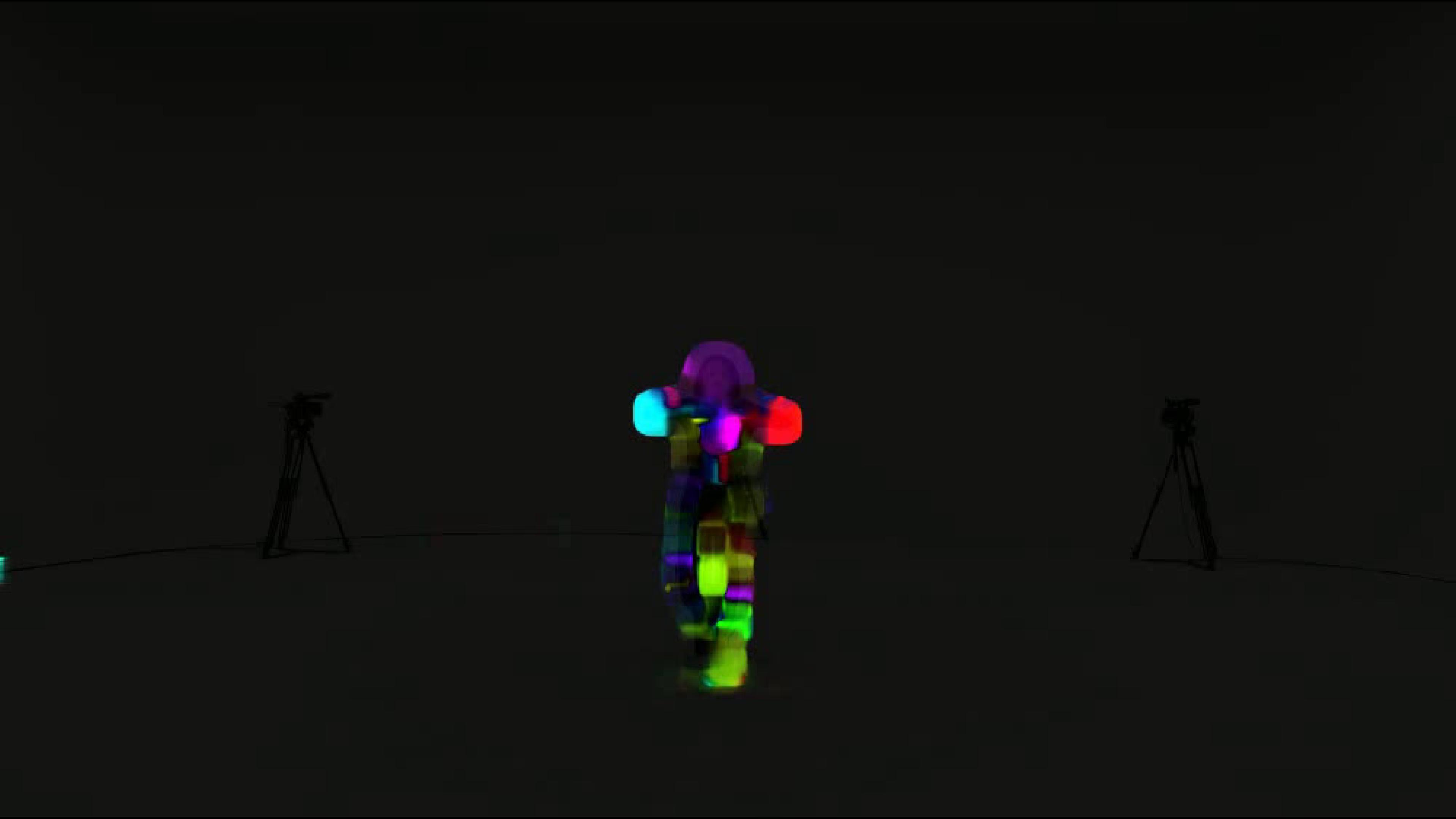
MEI



MHI



Contorno



MÉTODO PROPIO

IMPORTACIÓN DE MEDIOS Y DECLARACIÓN DE VARIABLES

```
# Importando el video
vc = cv2.VideoCapture("dance2.mp4")

#Crear medios para guardar el video del Optic Flow RGB
fourcc = cv2.VideoWriter_fourcc(*'mp4v') # Generando el archivo donde se guardará el video
out = cv2.VideoWriter('NuestroOFResultados/NuestroOpticalFlow.mp4', fourcc, 20.0, (1000, 562))

"""Leyendo el primer frame y sacando medidas"""
_, first_frame = vc.read() # Leer el primer frame

# Reescalando imagen y obteniendo el primer frame
resize_dim = 1000 #Ancho total
max_dim = max(first_frame.shape)
scale = resize_dim/max_dim
first_frame = cv2.resize(first_frame, None, fx=scale, fy=scale)
# Convierte el primer cuadro a escala de grises
prev_gray = cv2.cvtColor(first_frame, cv2.COLOR_BGR2GRAY)

"""Declaracion de variables varias"""
umbral=50 #Umbral de deteccion de movimiento
mask = np.zeros_like(first_frame) #Matriz de zeros para alojar O.F.
mask[..., 1] = 255 # Nos deshacemos de la saturacion de la imagen poniendo al maximo
output_folder2="NuestroOFResultados/"

i=0
flow2 = np.zeros((562, 1000))
flow3 = np.zeros((562, 1000))
kernel=np.ones((11,11),np.uint8)
```

GENERACION DEL OPTICAL FLOW

```
>while(vc.isOpened() and _==True):
    # Obtenemos los siguientes frames
    _, frame = vc.read()
    if frame is None:
        print("Fin del video.")
        break

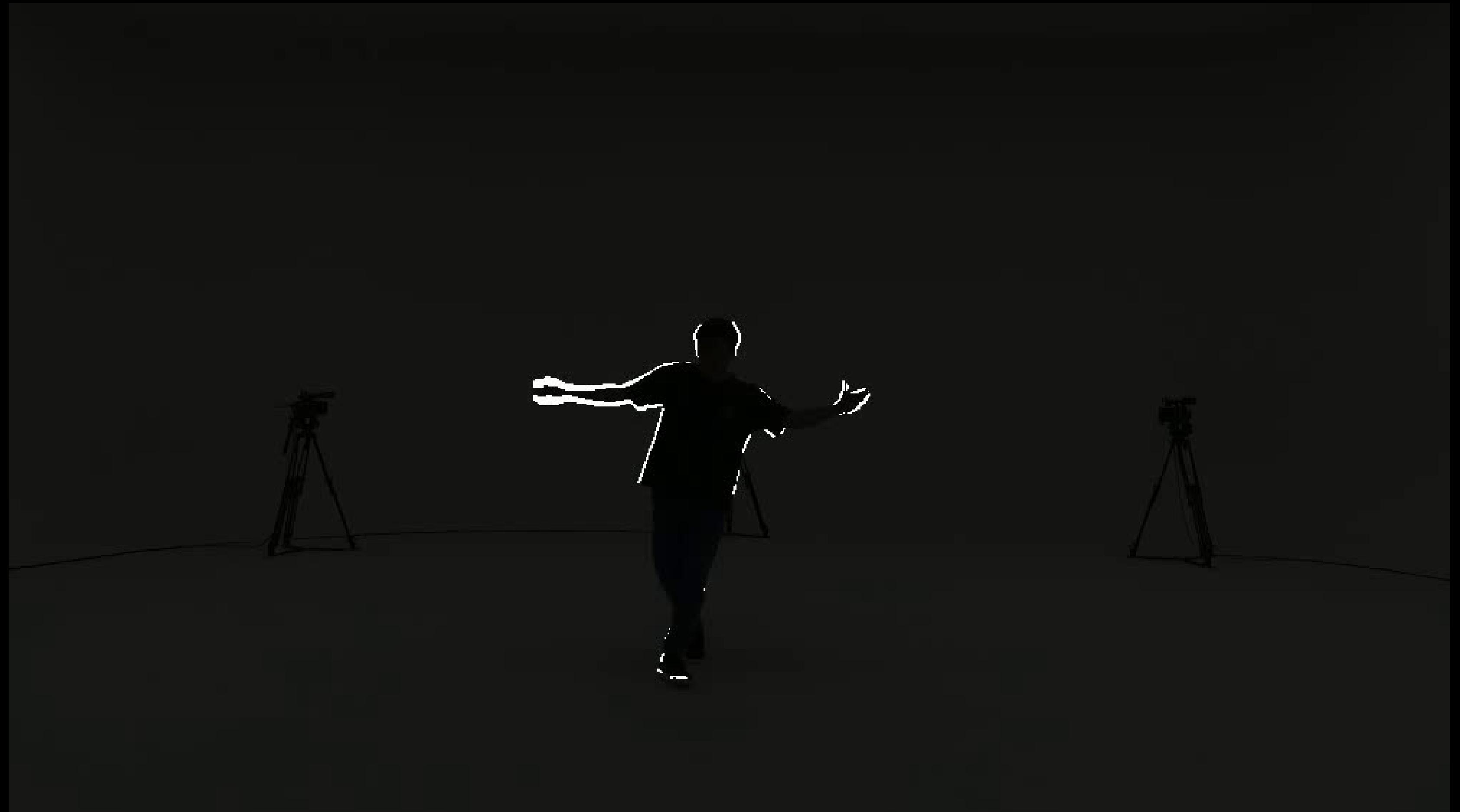
    # Redimension y filtrado
    gray = cv2.resize(frame, None, fx=scale, fy=scale)
    gray = cv2.cvtColor(gray, cv2.COLOR_BGR2GRAY)
    gray = cv2.normalize(gray,None, 0, 1.0, cv2.NORM_MINMAX, dtype=cv2.CV_32F)
    #Filtrado de fotograma para eliminar ruido
    gray = cv2.GaussianBlur(gray,(3,3),0)

    """Calculo del Optic Flow"""
    # Calculamos el flujo optico a traves de la diferencia de píxeles entre ambos frames
    # A traves de un umbral
    diff = abs(prev_gray-gray)
    cv2.imshow('GRIS-ORIGINAL', gray)
    condicion= diff> umbral/255
    condicion2= diff<= umbral/255
    diff[condicion] = 255
    diff[condicion2] = 0
    diff = diff.astype(np.uint8)
    cv2.imshow('OF+Original', diff)

    prev_gray=gray

    """Despliegue Optic Flow"""
    factor_atenuacion = 0.1 # Ajusta este valor según tus preferencias
    frame Oscuro = frame.astype(float) * factor_atenuacion
    frame = cv2.resize(frame_Oscuro.astype(np.uint8), None, fx=scale, fy=scale) # Redimension
    diff2=diff
    diff = cv2.cvtColor(diff, cv2.COLOR_GRAY2BGR)
    # Añadido y despliegue
    dense_flow = cv2.addWeighted(frame, 1,diff, 2, 0)
    cv2.imshow("Dense optical flow", dense_flow)
```

VIDEO OPTICAL FLOW



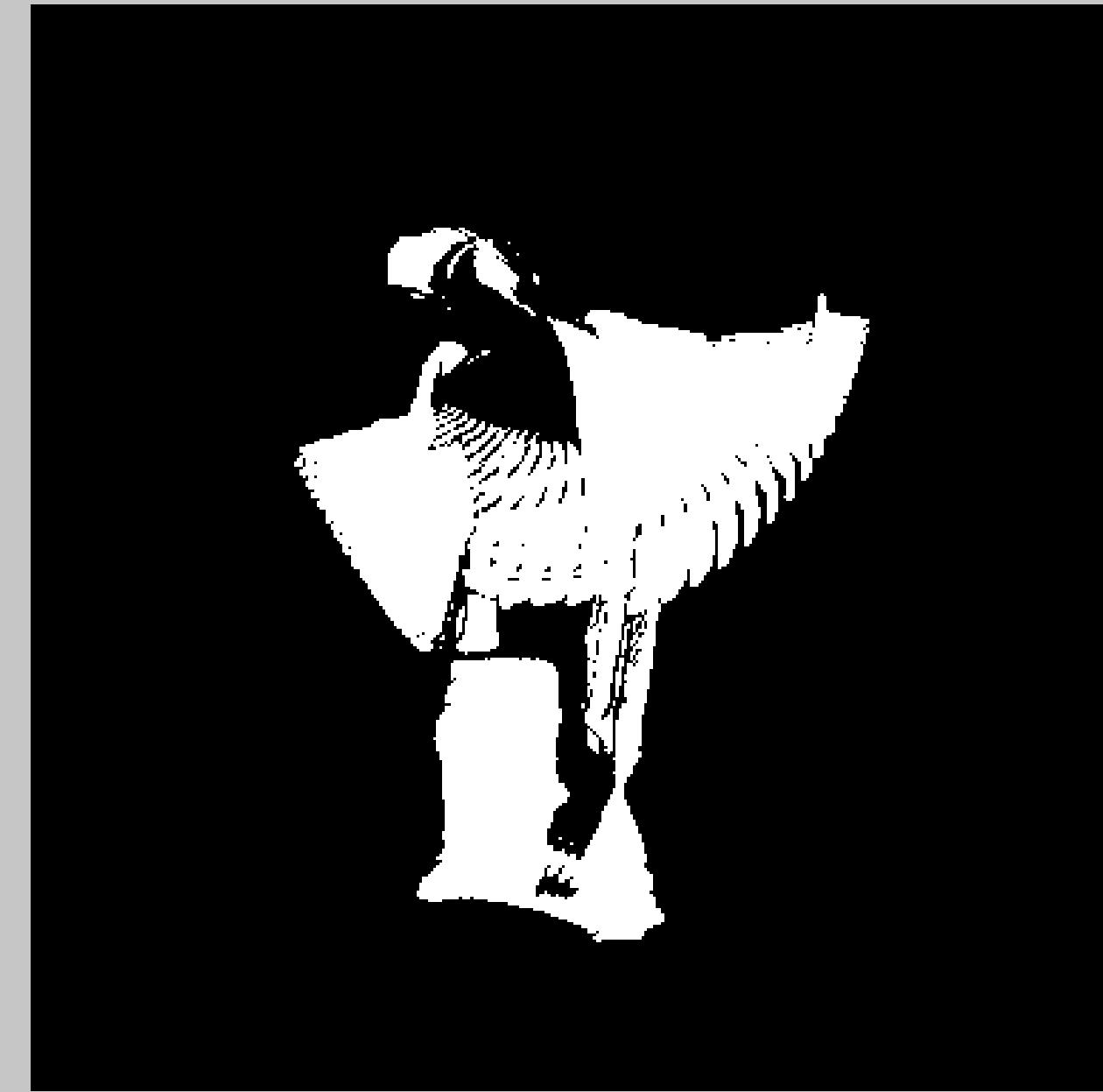


GENERACION DEL MEI

```
"""Generacion MEI imagen"""
if 50 <= i <= 80:
    flow2=diff2+flow2
    #imagen = flow2[:, :, 1]
    imagen_escalada = cv2.normalize(flow2, None, 0, 255, cv2.NORM_MINMAX)
    # Convertir la imagen a tipo uint8
    imagen_uint8 = imagen_escalada.astype(np.uint8)
    # Mostrar la imagen con cv2
    cv2.imshow('MEI', imagen_uint8)
    cv2.imwrite('NuestroOFResultados/MEI.png',imagen_uint8)
```



**ACUMULACIÓN DEL
FLUJO OPTICO**



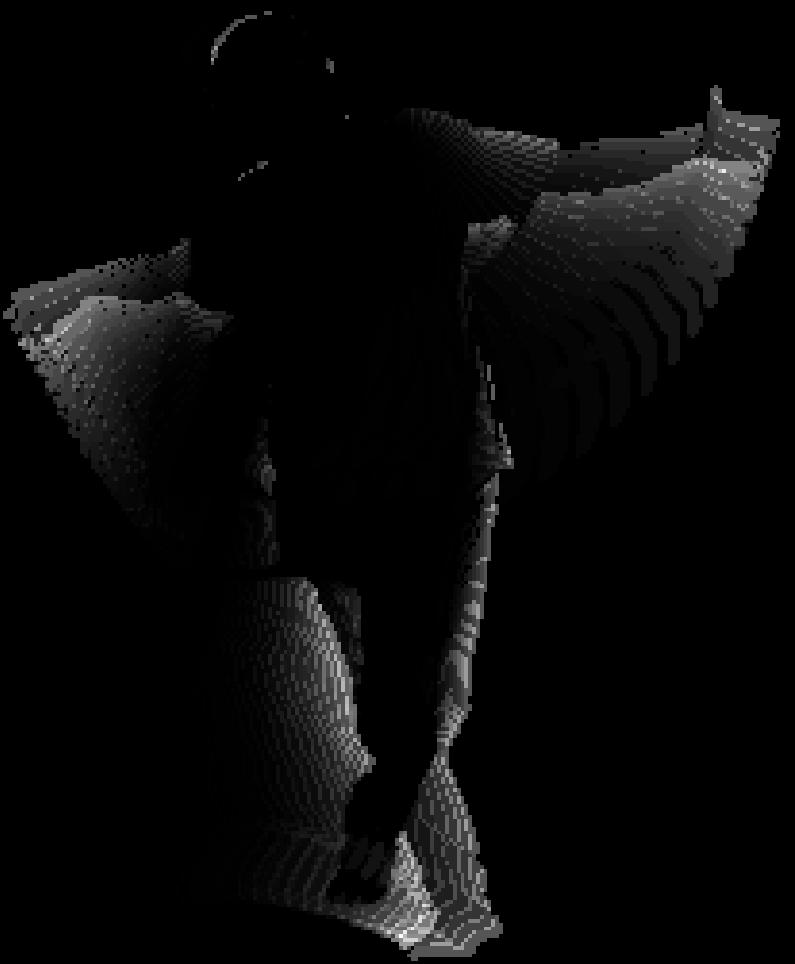
**MOTION ENERGY
IMAGE**

GENERACIÓN DEL MOTION HISTORY IMAGES (MHI)

```
"""Generacion MHI"""
if 50 <= i <= 80:
    magnitude2=cv2.normalize(diff2, None, 0, 255, cv2.NORM_MINMAX)

    # Aplica la condición a cada elemento
    condicion = magnitude2 >= 18
    condicion2 = magnitude2 < 18
    # Establece a 255 solo los elementos que cumplen la condición
    magnitude2[condicion] = 255
    magnitude2[condicion2] = 0

    flow3=magnitude2+flow3*.8
    imagen_escalada2 = cv2.normalize(flow3, None, 0, 255, cv2.NORM_MINMAX)
    # Convertir la imagen a tipo uint8
    imagen_uint82 = imagen_escalada2.astype(np.uint8)
    # Mostrar la imagen con cv2
    cv2.imshow('MHI', imagen_uint82)
    cv2.imwrite('NuestroOFResultados/MHI.png',imagen_uint82)
```



**CONTORNO DEL
MEI USANDO
FILTRO DE SOBEL**

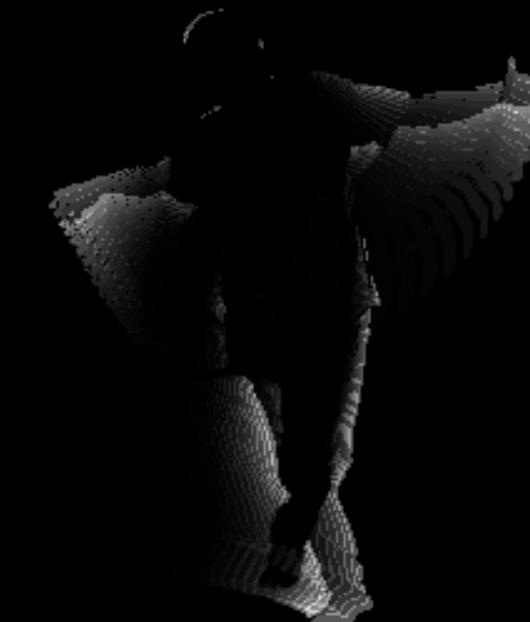


Resultados

30 FPS



MEI



MHI

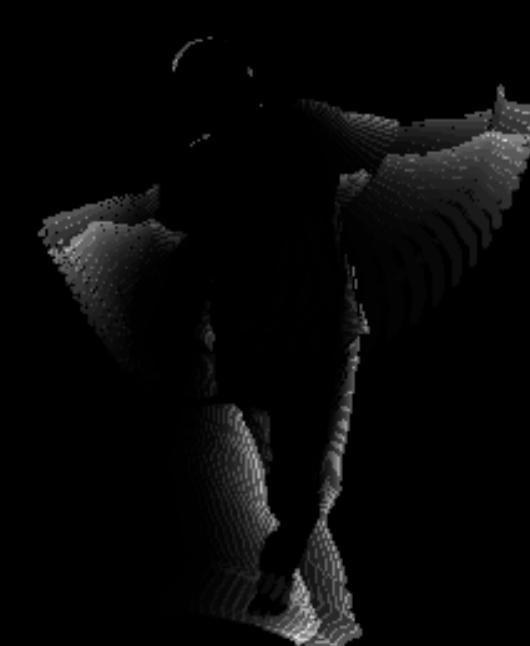


Contorno

20 FPS



MEI

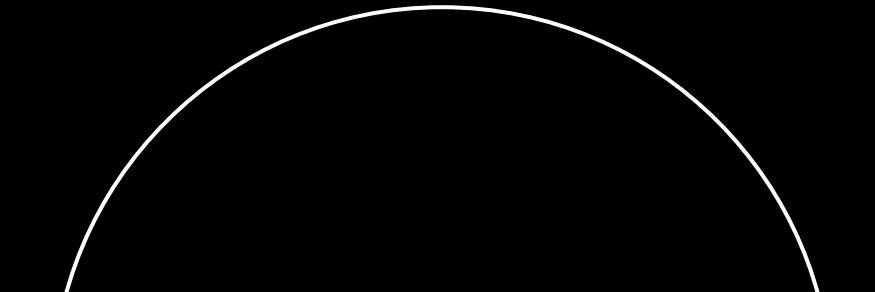


MHI



Contorno

Video original 2



Resultados

30 FPS



MEI



MHI



Contorno

20 FPS



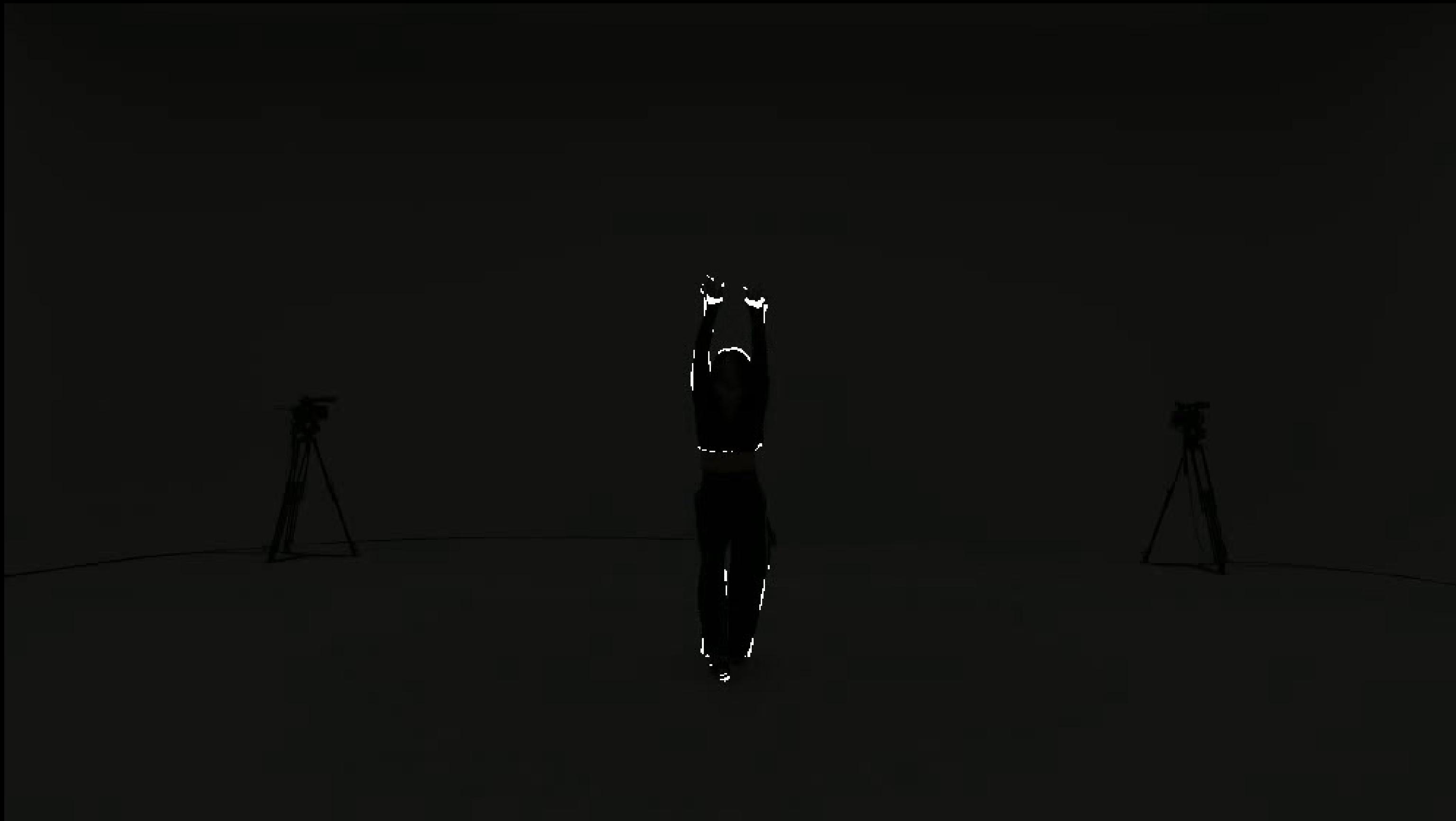
MEI



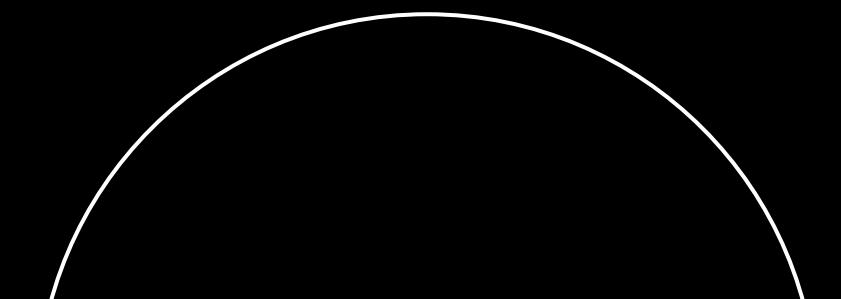
MHI



Contorno



Conclusiones



En términos de eficiencia, esta implementación fue un poco robusta y eficaz para la estimación de flujo óptico y la generación de imágenes de movimiento. Logrando un rendimiento en tiempo real, mostrandonos que existe un margen para explorar optimizaciones adicionales que puedan mejorar aún más la velocidad de procesamiento, asegurando una ejecución más fluida en diversas configuraciones.

