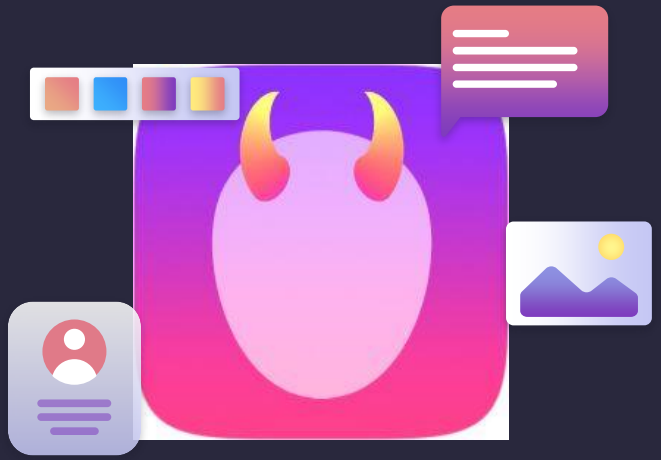
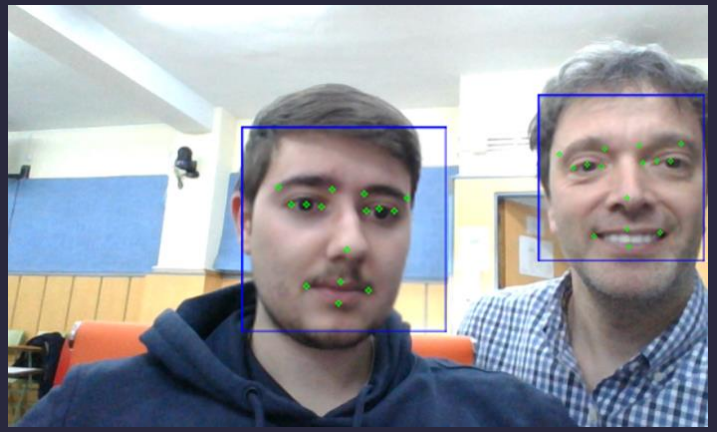
/PROYECTO 2.2
SELFIE FILTERS



IES Ribera del Tajo
Enseñanza bilingüe

/Enunciado

Con los modelos que hemos entrenado en el proyecto 1 queremos hacer una utilidad para una red social que, dependiendo de la emoción de la persona, permita visualizar filtros sobre los puntos faciales de cada persona.



/Pasos

/0 Carga en una lista de filtros imágenes con filtros divertidos, por ejemplo, unas gafas:

```
gafas = cv2.imread(ruta, cv2.IMREAD_UNCHANGED)
```

/1 Lee frames de la cámara de video con openCV

/2 Quédate con un frame en color y un frame en gris:

```
frame = cv2.flip(frame, 1)
frame2 = np.copy(frame)
hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

/3 Con el algoritmo de Viola/Jones, detecta la cara:

```
faces = face_cascade.detectMultiScale(…)
for (x, y, w, h) in faces:
    cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
    gray_face = gray[y:y+h, x:x+w]
    original_shape=gray_face.shape  #quédate con la forma original que te hará falta luego
    color_face = frame[y:y+h, x:x+w]
```

/Pasos

/4 Redimensiona la cara a 96x96 píxeles. Por cada cara detectada (gray_face), detecta su emoción y sus puntos faciales con los modelos entrenados en el proyecto 1

```
keypoints = modelo.predict(cara_redimensionada)
```

/5 Con la predicción de puntos faciales, empareja los puntos para que te sea más fácil averiguar cuales te interesan de cara a calcular la ubicación del filtro:

```
points = []
for i, co in enumerate(keypoints[0][0::2]):
    points.append((co, keypoints[0][1::2][i]))
```

/6 Muestra los puntos faciales con los números de las coordenadas. Fíjate que, por ejemplo, para ponerle unas gafas, te interesan los puntos 9 y 7 (ancho) y 8 y 10 (alto).

/Pasos

7/ Calcula las dimensiones que debería tener la imagen del filtro acorde a los puntos faciales y redimensiónalas a ese tamaño. Por ejemplo, si cargas las gafas, mide el ancho de las gafas y el alto de las gafas con los puntos correspondientes:

```
 ancho_gafas= int((points[7][0]-points[9][0]))
 alto_gafas = int((points[10][1]-points[8][1]))

 gafas_resized =
   cv2.resize(gafas, ancho_gafas, alto_gafas), interpolation = cv2.INTER_CUBIC)
```

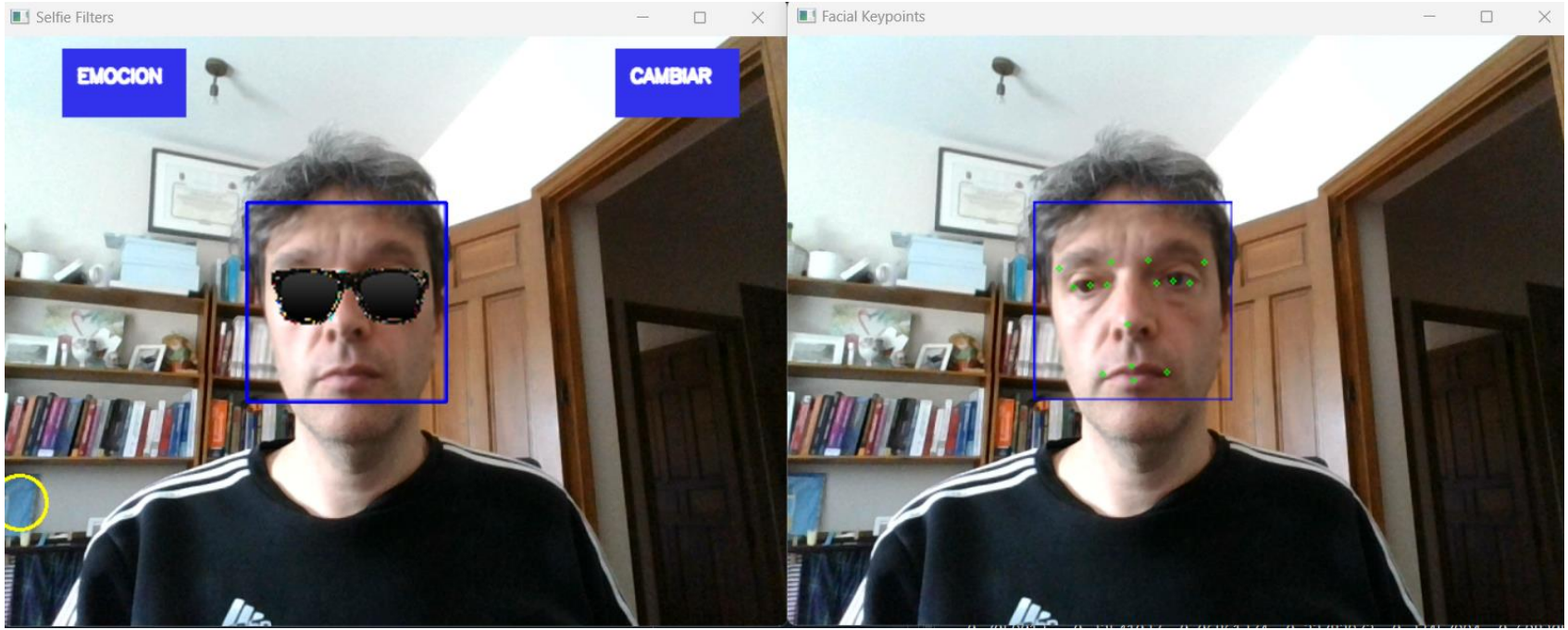
8/ Coloca el filtro en su posición (boolean indexing)

```
region_no_transparente = gafas_redim[:,:,:3] != 0
cara_redim_color[int(points[9][1]):int(points[9][1])+gafas_alto,
                 int(points[9][0]):int(points[9][0])+gafas_ancho,:][region_no_transparente] =
                       gafas_redim[:,:,:3][region_no_transparente]
```

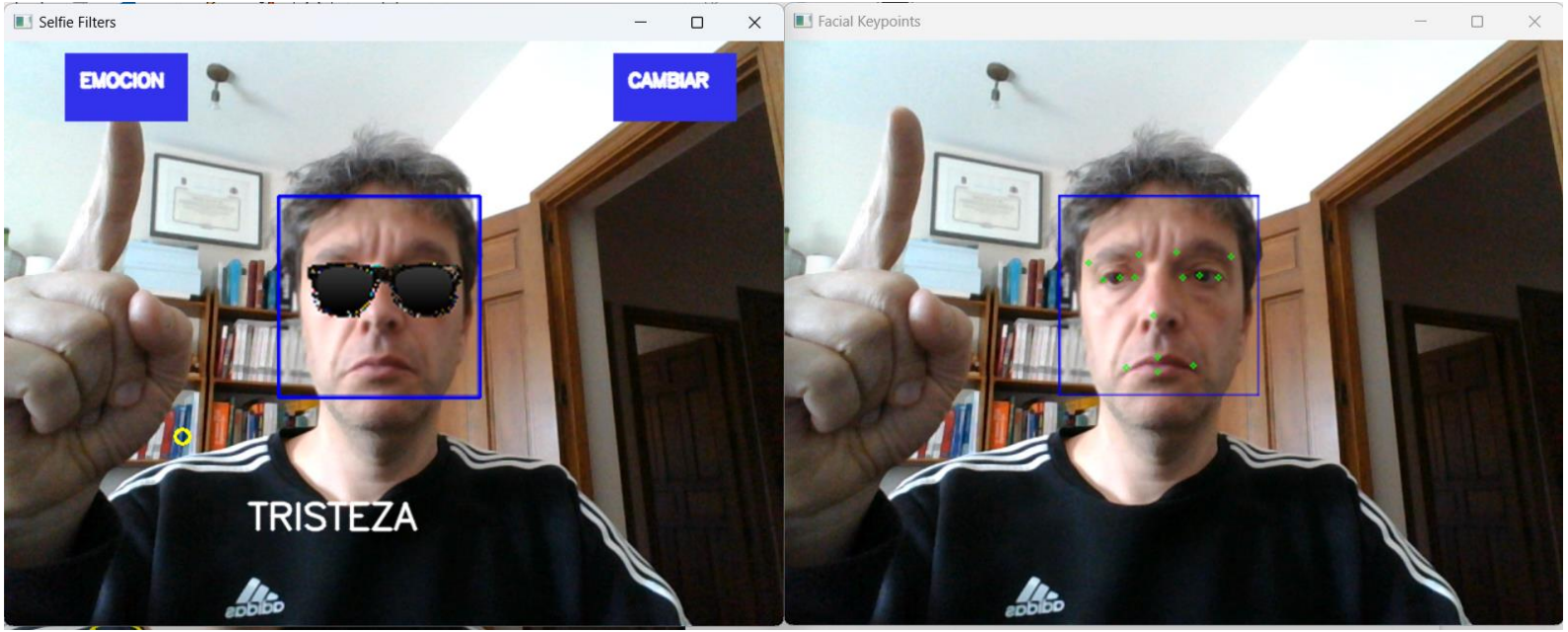
9/ redimensiona cara_redim_color a su tamaño original (original_shape)

```
frame[y:y+h, x:x+w] = cv2.resize(cara_redim_color, original_shape, interpolation = cv2.INTER_CUBIC)
```

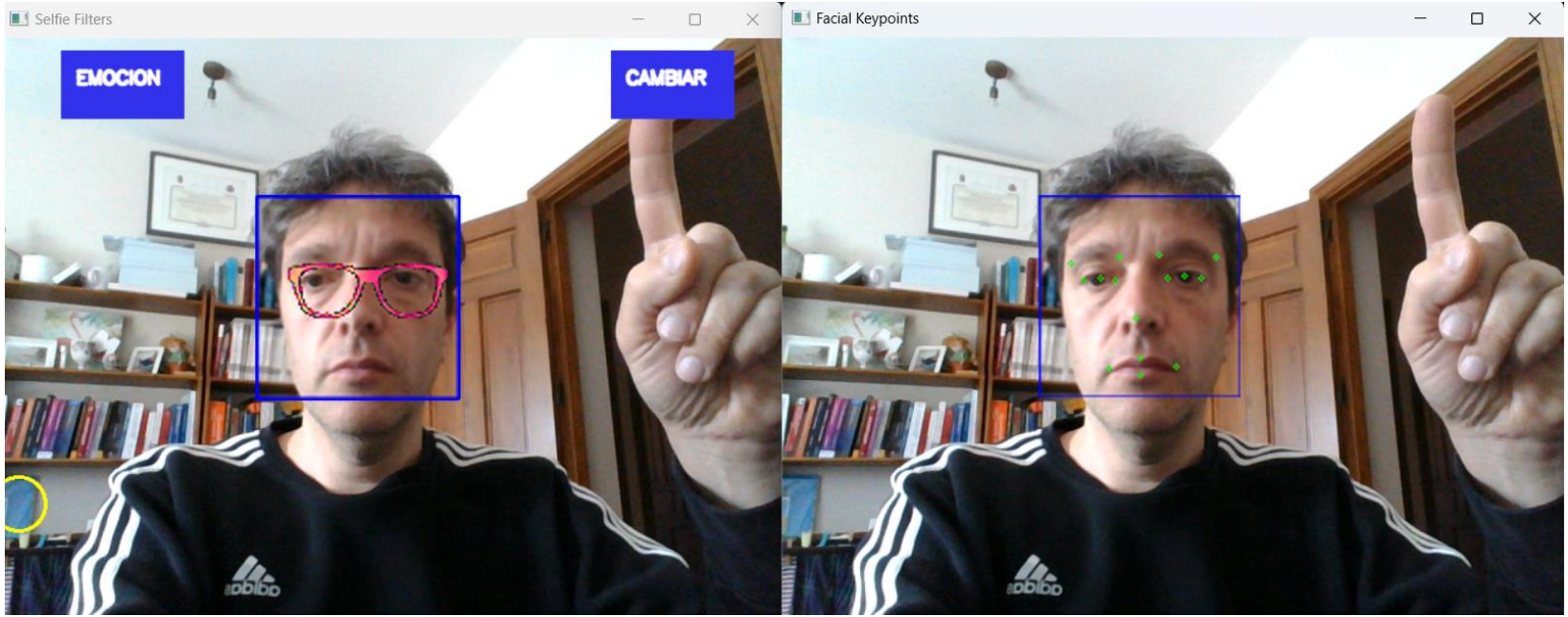
/Requisito 1: La aplicación mostrará dos ventanas, una con los puntos faciales y otra con dos botones, CAMBIAR (cambiar filtro) y EMOCIÓN (mostrar emoción)



/Requisito 2: Al señalar con un dedo el botón de EMOCIÓN, se mostrará la emoción en la pantalla de filtros



/Requisito 3: Al señalar con un dedo el botón de CAMBIAR, se cambiará el filtro al siguiente filtro de la lista



/Requisito 4: Al menos uno de los filtros, será un bigote