

UNIVERSIDAD DE GUADALAJARA



CENTRO UNIVERSITARIO DE CIENCIAS EXACTAS E INGENIERÍAS

Análisis de Algoritmos

Reporte de práctica

Nombre del alumno:	Samuel David Pérez Brambila
Código del alumno:	222966286
Profesor:	Erasmus Gabriel Martínez Soltero
Título de la práctica:	"Filtro de TikTok"
Fecha:	23 de Octubre de 2024

Metodología

Código:

```
Spyder (Python 3.12)
Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Proyectos Herramientas Ver Ayuda

C:\Users\samuel\OneDrive\Escritorio\Archivos UDG\S5\Análisis de Algoritmos\Tarea7.py

practica6_SVM.py X Tarea7.py X

1 import cv2 as cv
2
3 # Cargar los archivos PNG (sombrero y bigote)
4 sombrero = cv.imread("sombrero.png", cv.IMREAD_UNCHANGED)
5 bigote = cv.imread("bigote.png", cv.IMREAD_UNCHANGED)
6
7 # Cargar clasificadores
8 face_cascade = cv.CascadeClassifier("haarcascade_frontalface_default.xml")
9 smile_cascade = cv.CascadeClassifier("haarcascade_smile.xml") # Usamos el clasificador de sonrisas para colocar el bigote
10
11 # Obtener acceso a la webcam
12 video_capture = cv.VideoCapture(0)
13
14 # Configuración para grabar el video
15 fourcc = cv.VideoWriter_fourcc(*'XVID') # Codec para el video
16 out = cv.VideoWriter('video_filtro.avi', fourcc, 20.0, (640, 480))
17
18 # Función para superponer la imagen PNG
19 def superponer_png(fondo, overlay, x, y, w, h):
20     overlay_resized = cv.resize(overlay, (w, h)) # Redimensionar el PNG
21     for i in range(h):
22         for j in range(w):
23             if y + i < fondo.shape[0] and x + j < fondo.shape[1]:
24                 if overlay_resized[i, j][3] != 0: # Verificar el canal alfa
25                     fondo[y + i, x + j] = overlay_resized[i, j][:3] # Superponer si no es transparente
26
27 while True:
28     ret, frame = video_capture.read()
29     frame = cv.flip(frame, 1)
30     imagenGrises = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
31
32     # Detectar rostros
33     faces = face_cascade.detectMultiScale(imagenGrises, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
34
35     for (x, y, w, h) in faces:
36         # Calcular el centro del rostro para centrar el sombrero
37         face_center_x = x + w // 2
38
39         # Ajustar el tamaño del sombrero dinámicamente en función del ancho del rostro
40         sombrero_width = int(1.2 * w)
41         sombrero_height = int(0.8 * h)
42
43         # Calcular la nueva posición para centrar el sombrero sobre la cabeza
44         sombrero_x = face_center_x - sombrero_width // 2
45         sombrero_y = y - int(0.65 * h)
46
```

```
Spyder (Python 3.12)
Archivo Editar Buscar Código fuente Ejecutar Depurar Terminales Proyectos Herramientas Ver Ayuda

C:\Users\samuel\OneDrive\Escritorio\Archivos UDG\S5\Análisis de Algoritmos\Tarea7.py

practica6_SVM.py X Tarea7.py X

30 imagenGrises = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
31
32 # Detectar rostros
33 faces = face_cascade.detectMultiScale(imagenGrises, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
34
35 for (x, y, w, h) in faces:
36     # Calcular el centro del rostro para centrar el sombrero
37     face_center_x = x + w // 2
38
39     # Ajustar el tamaño del sombrero dinámicamente en función del ancho del rostro
40     sombrero_width = int(1.2 * w)
41     sombrero_height = int(0.8 * h)
42
43     # Calcular la nueva posición para centrar el sombrero sobre la cabeza
44     sombrero_x = face_center_x - sombrero_width // 2
45     sombrero_y = y - int(0.65 * h)
46
47     # Superponer el sombrero
48     superponer_png(frame, sombrero, sombrero_x, sombrero_y, sombrero_width, sombrero_height)
49
50     # Detectar sonrisas (aproximación para la boca)
51     roi_gris = imagenGrises[y:y+h, x:x+w]
52     roi_color = frame[y:y+h, x:x+w]
53     sonrisas = smile_cascade.detectMultiScale(roi_gris, scaleFactor=1.3, minNeighbors=22, minSize=(25, 25))
54     if len(smiles) > 0:
55         (sx, sy, sw, sh) = sonrisas[0]
56         bigote_height = int(sw * 0.5) # Determinar la altura del bigote basado en su ancho
57         # Ajustar la posición vertical para que el bigote esté justo sobre la sonrisa
58         superponer_png(roi_color, bigote, sx, sy - int(bigote_height / 2) + 10, sw, bigote_height)
59
60 # Guardar el frame en el video de salida
61 out.write(frame)
62
63 # Mostrar la imagen
64 cv.imshow('Video', frame)
65
66 # Salir al presionar 'q'
67 if cv.waitKey(1) & 0xFF == ord('q'):
68     break
69
70 # Liberar la cámara y cerrar ventanas
71 video_capture.release()
72 out.release()
73 cv.destroyAllWindows()
74
```

Resultados

Link del video: [Ver video](#)

