# Tarea 09. Seguimiento de Rostros con Viola y Jones.

Dr. Francisco Javier Hernández
Alumno: Mario Alberto Tapia de la Cruz

Maestría en Matemáticas Aplicadas

## 1. Seguimiento de Rostros.

### 1.1. Marco Teórico

Para esta tarea se usarán algunas funciones de la librería `OpenCV2` para implementar un programa que haga seguimiento de rostros en Python. A continuación se mencionarán las funciones a utilizar.

- `cv2.CascadeClassifier` - Con esta función cargaremos un clasificador entrenado para hacer la detección de rostros, usaremos un clasificador Haar.
- `cv2.data.haarcascades` - Esto va como argumento en la función `CascadeClassifier`, nos ayudará a cargar nuestro archivo entrenado `.xml` sin necesidad de escribir rutas de forma manual.
- `detectMultiScale` - Esta función escanea nuestra imagen en múltiples tamaños y nos ayudará a detectar objetos (en nuestro caso, rostros), y como resultado nos generará una lista de tuplas $(x, y, w, h)$ donde se hicieron las detecciones; cada una representa un rectángulo. A la función se le dará de entrada los siguientes parámetros:
  - `img_grises` - Una imagen en escala de grises.
  - `scaleFactor` - Cuánto reduce el tamaño de la ventana de detección de escala.
  - `minNeighbors` - Vecinos que necesita una región candidata para ser aceptada como rostro.
  - `minSize` - Tamaño mínimo en píxeles que puede tener una cara.
- `cv2.matchTemplate` - Esta función compara la imagen con el template obtenido, y usaremos el método `TM_CCOEFF_NORMED` para generar un mapa de correlación normalizada.
- `cv2.minMaxLoc` - Esta función la usaremos para encontrar el mínimo y el máximo de la matriz generada por `matchTemplate`, además de las coordenadas donde ocurren dichos valores.

Se probará el programa con un vídeo vertical de 27 segundos tomado con un celular.

Para la detección de rostros cargamos el clasificador Haar pre-entrenado `haarcascade_frontalface_default.xml`. Este clasificador fue entrenado para detectar rostros frontales.

### 1.2. ¿Cómo funciona el código?

Primero creamos nuestro `face_cascade` usando Viola & James con el clasificador entrenado `haarcascade_frontalface_default` además de que inicializamos `found_face = False` y `template = None`, para que al entrar al bucle while, directamente vaya al `if not found_face` para detectar rostros. Si sí se detectaron rostros se manda a llamar la primer entrada de la variable faces (el primer rostro detectado) y se le dibuja un rectángulo y regresa al inicio del while, si no se encuentra un rostro se avanza de frame y se reintenta la re-detección.

Además cada una cierta cantidad de frames se guarda una imagen en formato .jpg. Todos los primeros se van mostrando de manera consecutiva.

Se le dio una tolerancia de 0.7 para el template matching para la condicional de `found_face`, si no, se mandan a hacer las variables `found_face = False` y `template = None` nuevamente.

Todo dentro del bucle hasta que se terminen todos los frames, o el usuario presioné la tecla `q`.

## 1.3. Detectar una sola cara a la vez

El código esta escrito de manera en que se detecte un solo rostro, por ejemplo, en uno de los experimentos aparecen dos personas (Sarah y Mario Tapia), pero el primer rostro que detecta es al que le realiza el seguimiento, hasta que lo pierde y vuelve a buscar un rostro nuevo (entonces puede volver a encontrar a la persona que seguía, o detectar a la otra persona).

Esto se puede ver en la parte de `if len(faces) >0`, al guardar las variables $x, y, w, h$, está mandando a llamar solamente a `faces[0]`.

El código es bastante sensible a cambios de movimentos bruscos, por lo que un ligero movimiento en la cabeza puede hacer que tenga que volver a recalcular el template del rostro a seguir (Esto se puede ver visualmente al dejar de comentar las lineas de `cv2.putText` para ver como frecuentemente en movimientos bruscos se ve por un frame la leyenda de "*Rostro detectado*", "*Rostro Re-detectado*".

A continuación se presentan capturas de pantalla de lo mencionado anteriormente.

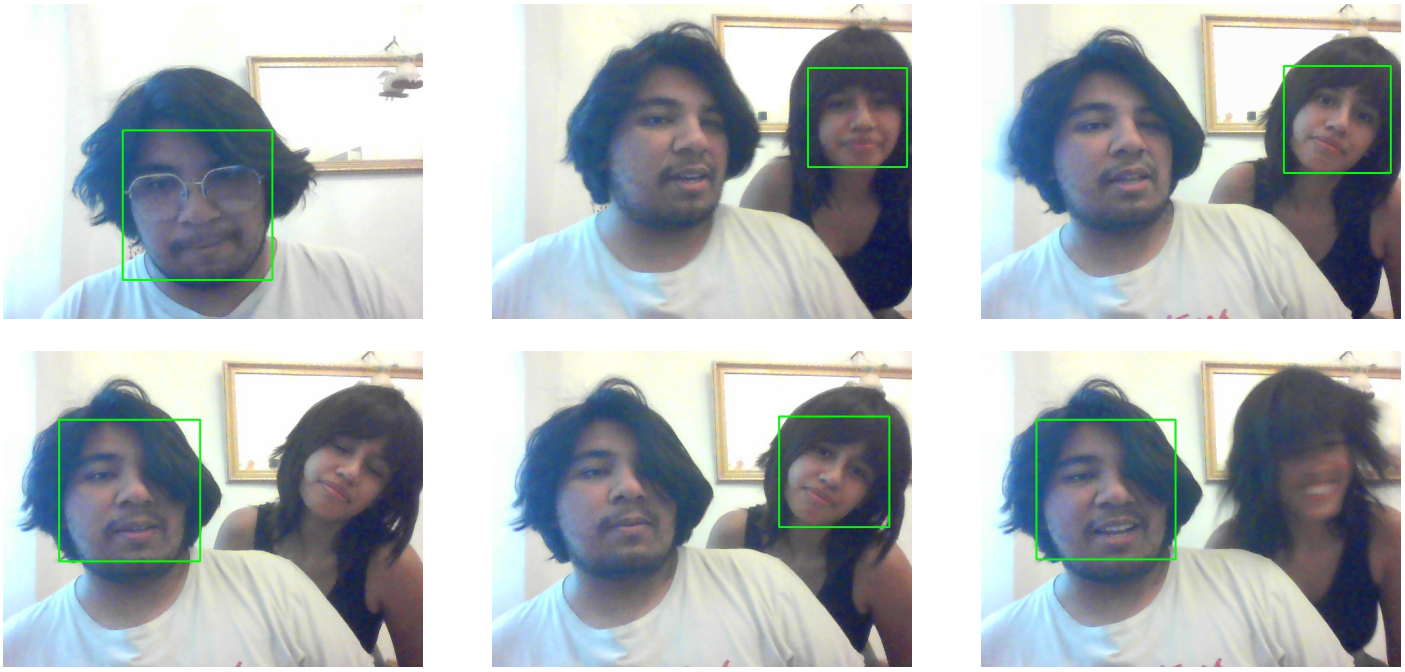

Figura 1: Dos personas en la misma escena, una sola detección de cara.

## 1.4. Experimentos

A continuación se mostrarán dos experimentos, en el que sale una sola persona (Sarah) y se detecta una sola cara en movimiento a lo largo del vídeo.

Y el experimento 2 en el que se pasa un vídeo en el que entran y salen 3 personas (Sarah, Samuel y Mario Tapia) a la escena, y se detectan sus rostros en las tres ocasiones conforme avanza la escena.

Figura 2: Experimento 1 (Sarah moviéndose)

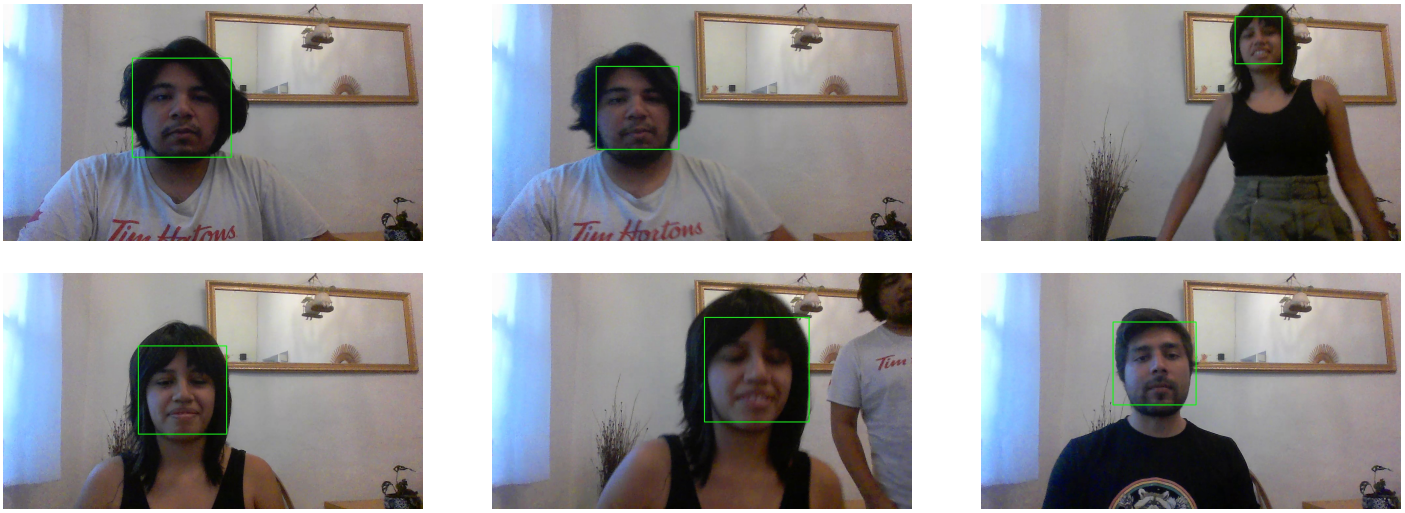

Figura 3: Experimento 2. Sarah, Samuel y Mario Tapia entrando y saliendo de escena.

## 1.5. Conclusiones

Se podría extender para detección de múltiples rostros e impresión de las múltiples cajas, además de que se pudiera revisar la sensibilidad del programa para que no recalcule ante movimientos bruscos en la escena.

También se pierde la detección cuando la persona que está en la escena inclina su cabeza hacia la izquierda y la derecha, además de que si solo se ve parcialmente la cara de la persona, esto es más notable si se está probando la detección con la webcam.

Se puede probar también con la webcam (la línea comentada está en el archivo .py)