Gustavo Castro
Carné 11129

Reconocimiento facial mediante algoritmos genéticos

Se utilizaron algoritmos genéticos para el reconocimiento facial, específicamente se trata hacer encajar una elipse con los rostros dentro de la imagen.

Previo a la utilización del AG se realizó un procesamiento de imagen, que consistió en lo siguiente:

1. Convertir la imagen de RGB a escala de grises
2. Modificar el tamaño de la imagen a 256x256 pixeles
3. Eliminar el ruido de la imagen
4. Aplicar el algoritmo de Canny en la imagen para obtener los bordes de las imagenes (Escala de blanco y negro)

Para realizar estas tareas se utilizaron funciones de las librerías PIL y CV. En particular, para el paso 3 se utilizó la función cv2.fastNlMeansDenoisingColored. Finalmente, en el paso 4 se utilizó la función Canny de la librearía CV cuyo resultado se puede apreciar en la Fig 1.

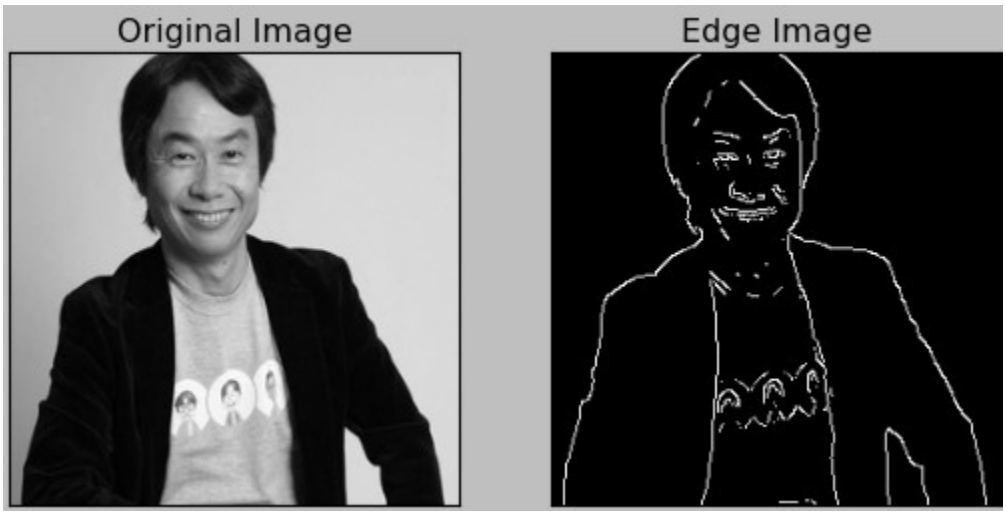

Fig 1 : Resultado de la imagen luego de aplicarle la función Canny

Posteriormente, al tener la imagen en blanco y negro se obtiene una matriz M de 256x256 pixeles 0 y 255 (negro y blanco respectivamente)

A partir de la matriz M recolectada anteriormente se puede iniciar con la implementación del algoritmo genético. La idea central de utilizar algoritmos genéticos es la siguiente:

Partimos generando 4 valores:

X0 : Coordenada en x del centro de la elipse (8 bits)
Y0: Coordenada en y del centro de la elipse. (8 bits)

b: Valor del eje mayor (8 bits)
a: Valor de semieje (8 bits)

Entonces cada cromosoma se compone de estos cuatro elementos (X0,Y0,a,b) y el objetivo es generar una ecuación de la elipse a partir de estos valores

(1) $$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$$

Por ejemplo si la población inicial fuera de 50 cromosomas, significa que se generarían 50 ecuaciones y por cada cromosoma se hace un test con los todos los valores (posición en *x*,*y* de cada pixel) de la matriz M para los los puntos blancos (los de valor 255) en la ecuación (1) , con la salvedad de que no debe ser necesariamente igual a 1 sino se cambió la restricción a que sea menor o igual 2, ya que evidentemente no estamos tratando con elipses perfectas. De esta manera se obtiene el fitness de cada cromosoma, gana quien más puntos cumpla con la restricción mencionada.

En resumen, tenemos:

1. Generar población de cromosomas de forma aleatoria.
2. Realizar un conjunto de iteraciones para las generaciones
   2.1 Por cada generación evaluar el fitness de cada cromosoma
   2.2 Al finalizar cada generación realizar crossover de cromosomas con probabilidad p en base a valores fitness, correspondencia de cromosomas con mejor fitness. A partir de esto se genera nueva población.
   2.3 Realizar mutaciones en base a probabilidad q
   2.4 Guardar mejor cromosoma por cada generación según fitness
3. Evaluar los mejores cromosomas de cada generación y obtener el mejor

En cuanto a la mutación y crossover, cabe mencionar que la mutación es de un bit por cromosoma y el crossover se hizo intercambiando una sección entre dos cromosomas con límites elegidos de forma aleatoria.

En la siguiente tabla se muestran los datos para la ejecución del algoritmo.

Dato	**Valor**
Probabilidad de mutación	0.2
Probabilidad de cruce	0.8

Resultados de la ejecución:

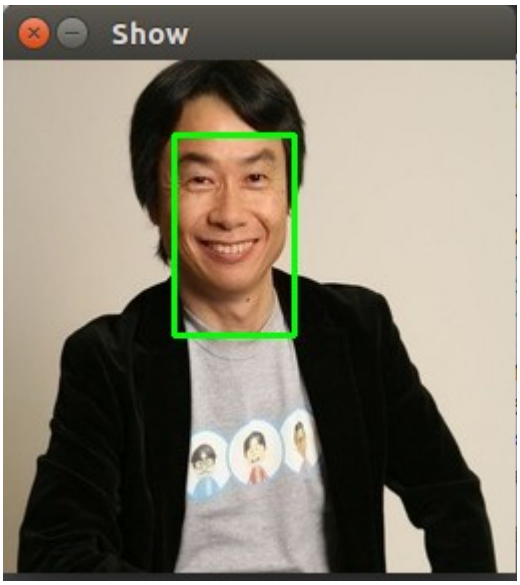

Fig 2: Ejecución del algoritmo con población 40 de individuos, y 10 generaciones.

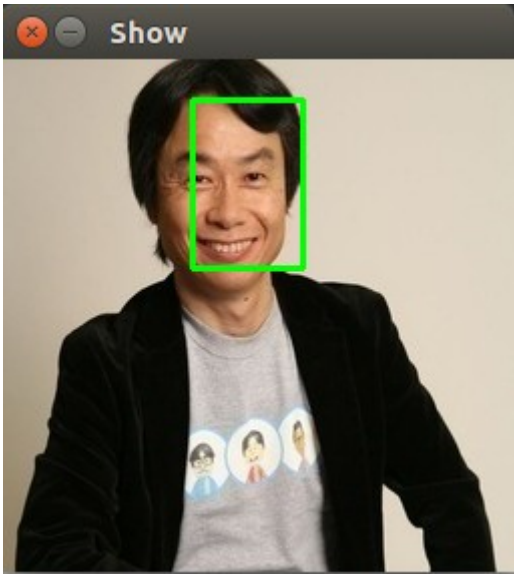

Fig 3: Ejecución del algoritmo con población 30 de individuos, y 15 generaciones.

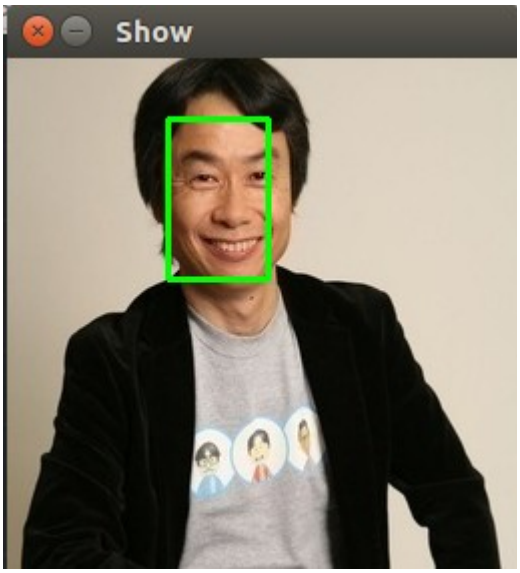

Fig 4: Ejecución del algoritmo con población 40 de individuos,
y 20 generaciones.