Disertación

Por

Andrés Albán C.

Facultad de Ingeniería.

Escuela de Sistemas.

Dedicatoria

El Desarrollo de esta disertación está dedicada para mí, demostrándome que puedo hacer lo que me interesa. También se la dedico a mi familia y a mis profesores que colaboraron para el desarrollo de la misma.

Tabla de Contenidos

Contenido

Resumen

En la presente disertación se hablará de temas relacionados con procesamiento de imágenes, uso de redes neuronales artificiales y desarrollo de software con el propósito de alcanzar a tener una aplicación capaz de reconocer imágenes de rostros humanos.

Capítulo 1

- Comprende una introducción de lo que trata y sirve el procesamiento de imágenes dando a conocer algunos conceptos, definiciones y funciones importantes como también algunas operaciones que se manejan con imágenes.

Capítulo 2

- Enlista y conoce sobre diferentes programas para el procesamiento de imágenes; contiene características de herramientas de software libre y software privado y el uso de software relacionado a RNA (redes neuronales artificiales).

Capítulo 3

- Propone ciertas soluciones a implementar para el reconocimiento de imágenes.

Capítulo 4

- Procede con el desarrollo de la aplicación propuesta en el capítulo anterior así como también realiza el análisis de resultados.

Capítulo 5

- Define las conclusiones y recomendaciones del proyecto realizado.

Introducción

Motivación

El motivo por la cual voy a desarrollar esta disertación, es para elaborar una aplicación relacionada al reconocimiento de imágenes de rostros humanos aprendiendo del uso de soluciones avanzadas, como por ejemplo, el uso de redes neuronales artificiales o algoritmos de detección de características en imágenes.

Planteamiento

El tema de disertación va a tratar sobre el desarrollo de un software que permita hacer reconocimiento de imágenes de rostros humanos. El reconocimiento de rostros humanos dentro de un grupo de imágenes no es una tarea trivial, requiere de análisis de ciertas directrices que permitan primero establecer una probabilidad de existencia de un rostro y luego confirmar de su existencia dentro de un rango aceptable de incertidumbre, el presente trabajo tiene como alcance hallar un mecanismo de reconocimiento de rostros en varias imágenes de prueba y determinar el grado de eficiencia.

Objetivos

General

- Desarrollar una aplicación para el reconocimiento de imágenes de rostros humanos a través del uso de procesamiento de imágenes.

Específicos

- Establecer si una imagen contiene rostros.
- Determinar la localización de los rostros en la imagen a analizar.

Metodología

Lectura de imágenes desde archivos, manipulación de imágenes, técnicas de morfología y acondicionamiento para que con el uso de técnicas de mejoramiento de imágenes, filtros, procesamiento de señales y el uso de redes neuronales para el reconocimiento de patrones sea la base en la identificación de rostros.

Capítulo I

Procesamiento de imágenes

Introducción

El procesamiento de imágenes es un conjunto de técnicas con el objetivo de mejorar la apariencia visual de las imágenes para un observador humano, incluyendo su impresión, transmisión y preparación para la medición de las características y estructuras que presentan las mismas.

En particular, el procesamiento de imágenes es una tecnología práctica para:[i]

- Clasificación
- Extracción de características
- Reconocimiento de patrones
- Proyección
- Análisis de señales multiescala.

El procesamiento de imágenes comprende actividades como editar una imagen, restaurarla, modificarla o hacer reconocimiento. Procesar una imagen entonces, no es solamente modificar una de sus propiedades como por ejemplo su brillo, el procesamiento comprende el “entendimiento” de la imagen, es decir, determinar su significado, o extraer la información contenida en la misma.

Una imagen puede ser considerada como un arreglo bidimensional de puntos, los mismos que poseen una determinada coloración y en conjunto muestran lo que en ella se ve. Cada uno de los puntos que componen la imagen se denomina pixeles y además de su coloración, un pixel está identificado por su posición dentro de la imagen. Dicho de otra manera, una imagen es una matriz de puntos denominados pixeles, de modo que la imagen es una matriz de pixeles.

Para efectos del procesamiento de imágenes, siempre se refiere a imágenes rectangulares, de modo que la definición de imagen como matriz será siempre válida.

La forma de representar una imagen es: $f(x, y)$ la misma que muestra la amplitud f en las coordenadas x y y. La magnitud de $f(x, y)$ se halla limitada a valores válidos de los pixeles, por ejemplo para una imagen que posee 4 tonalidades o niveles de gris los valores permitidos de $f(x, y)$ son el 0, 1, 2 y el 3.

En general, se tiene que:

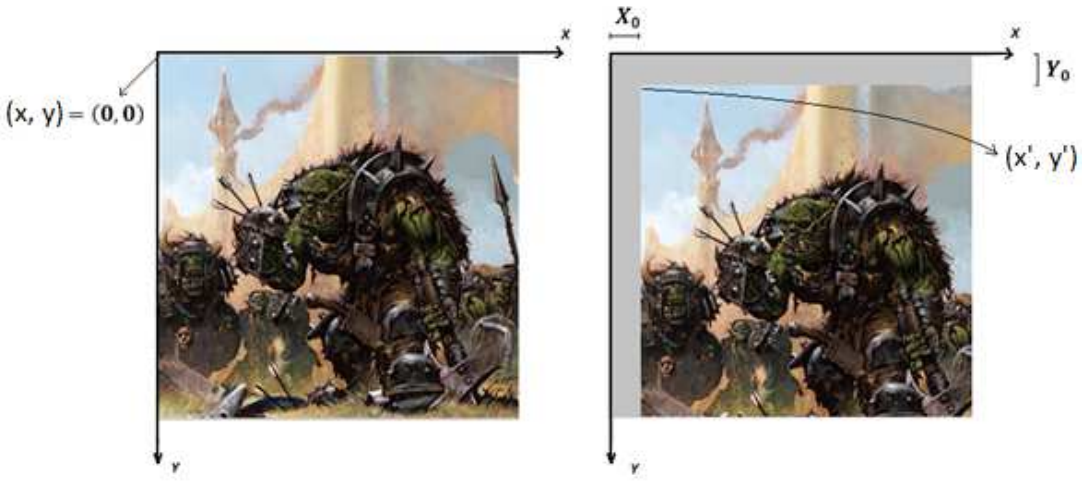
$$0 \leq f(x, y) \leq (Niveles\ de\ gris - 1)$$

Los factores que determinan los niveles de gris o el color de los pixeles son la Iluminación $i(x, y)$ y la Reflectancia $r(x, y)$, definidos como la cantidad de luz aplicada sobre el punto y la capacidad de reflejar o absorber la luz recibida. A mayor reflectancia mayor es la luz reflejada o lo que es lo mismo, menor es la luz absorbida. Entonces una imagen será: $f(x, y) = i(x, y)\, r(x, y)$, o sea pixel por pixel, el resultado de la iluminación y la reflectancia.[ii]

Geometría de imágenes

Traslación o desplazamiento

Se define como el desplazamiento de una imagen que puede ser tanto en el eje horizontal como vertical. Sea x' la nueva posición del pixel dada la imagen desplazada x_0 pixeles desde el origen, entonces $x' = x + X_0$, de manera similar, $y' = y + Y_0$.[iii]



$$x' = x + X_0$$

$$y' = y + Y_0$$

$$(x', y') = (x + X_0, y + Y_0)$$

Expresándolo en forma matricial, la traslación está dada por:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & X_0 \\ 0 & 1 & Y_0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

O de la siguiente forma:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & X_0 \\ 0 & 1 & Y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Si v es el vector $[X\ Y\ 1]^T$, entonces v' es el formado por $[X', Y'\ y\ 1]^T$, luego se puede escribir

$$v' = Ty$$

Donde T es la matriz de transformación:

$$\begin{bmatrix} 1 & 0 & X_0 \\ 0 & 1 & Y_0 \\ 0 & 0 & 1 \end{bmatrix} = T$$

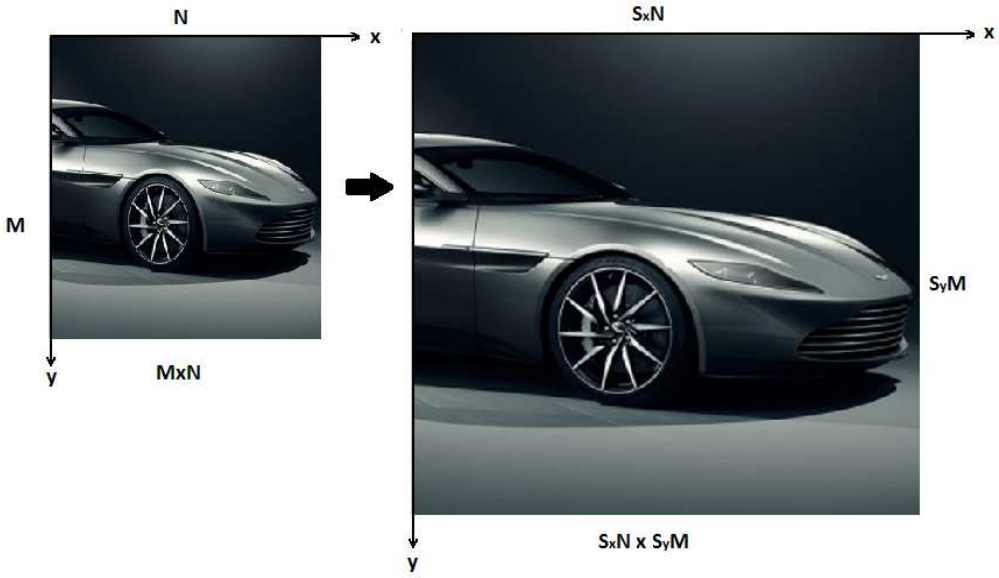
Escalamiento

Al escalar una imagen se cambia de tamaño a la misma, esto significa que si la imagen original posee dimensiones de N x M pixeles, la imagen resultado tendrá dimensiones de $\boldsymbol{S_x}N$ x $\boldsymbol{S_y}M$, donde $\boldsymbol{S_x}$ y $\boldsymbol{S_y}$ son los factores de escalamiento. La transformación de escalamiento expresada como[iv]

$$v' = Tv$$

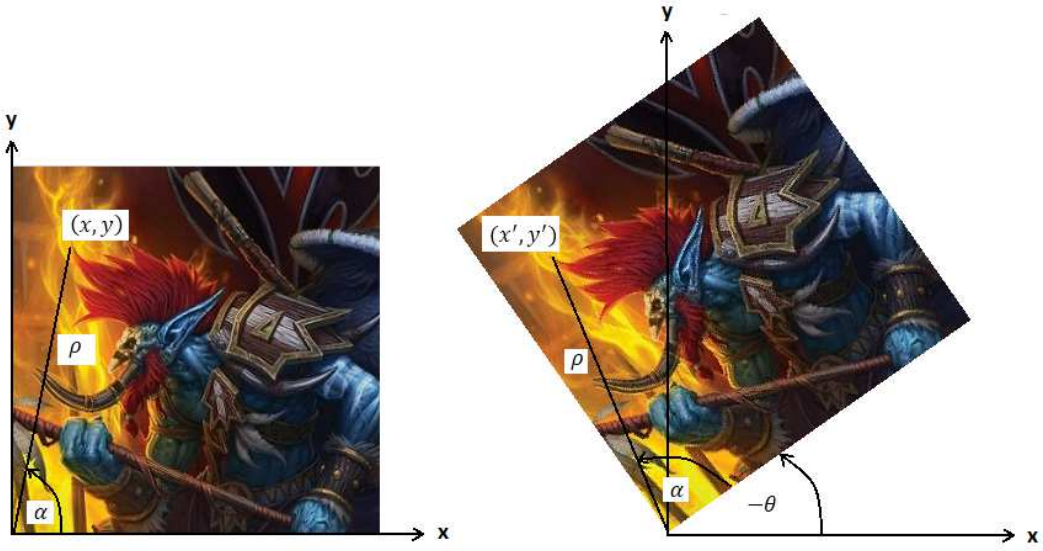
Con $\boldsymbol{S_x}$ como el factor de escala en el eje x y $\boldsymbol{S_y}$ en el eje y es:

$$T = \begin{bmatrix} \boldsymbol{S_x} & 0 & 0 \\ 0 & \boldsymbol{S_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Rotación

Rotar una imagen significa hacer girar la imagen sobre su punto de origen $(0,0)$ un ángulo θ. Debido a que el origen de la imagen está en su esquina superior izquierda, y que el eje y crece hacia abajo, se complican los cálculos para determinar la matriz de transformación, por ello, se usa una nueva representación en la que el eje y crece hacia arriba, permitiendo el cálculo de esta matriz. Note que el ángulo girado es $-\theta$, y que el origen es el usado en un sistema de coordenadas cartesianas. [v]



$$x = \cos(\alpha)\,\rho$$

$$y = sen(\alpha)\rho$$

$$y' = \rho\, sen(\alpha - \theta) \rightarrow \rho[sen(\alpha)\cos(\theta) - \cos(\alpha)\, sen(\theta)]$$

$$x' = \rho\, cos(\alpha - \theta) \rightarrow \rho[cos(\alpha)\cos(\theta) - \text{sen}(\alpha)\, sen(\theta)]$$

$$y' = ycos(\theta) - xsen(\theta)$$

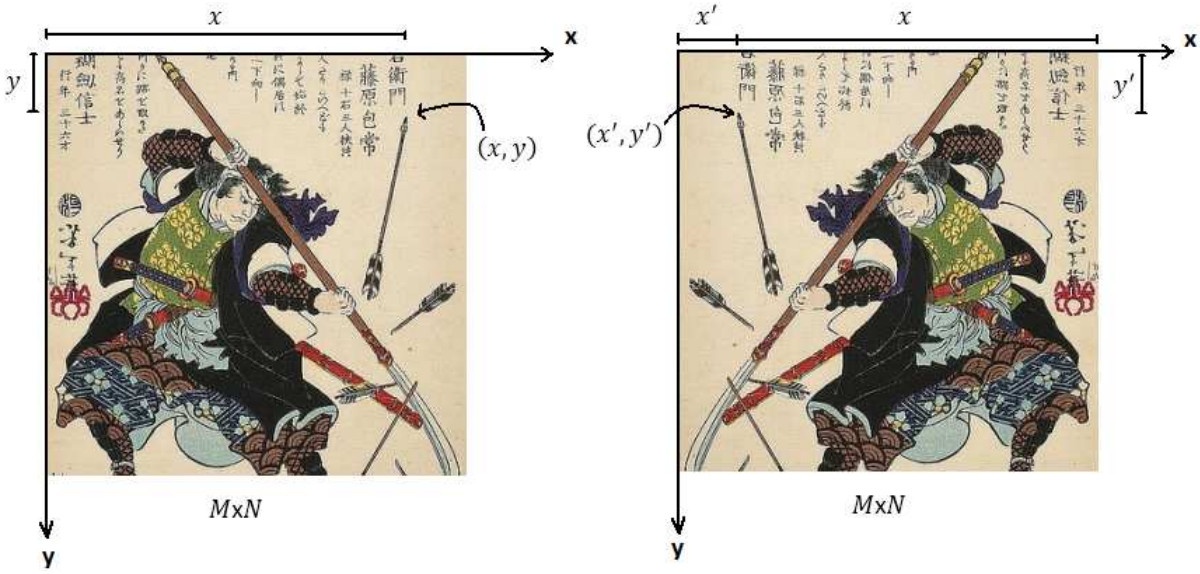
$$x' = xcos(\theta) + ysen(\theta)$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & sen(\theta) & 0 \\ -sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

La matriz de transformación es $T = \begin{bmatrix} \cos(\theta) & sen(\theta) & 0 \\ -sen(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Reflejo

El reflejo de una imagen consiste en darle la vuelta simétricamente a la imagen sin desplazamiento o traslación, ya sea verticalmente u horizontalmente.[vi]



$$y' = y$$

$$x' = N - x$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & N \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

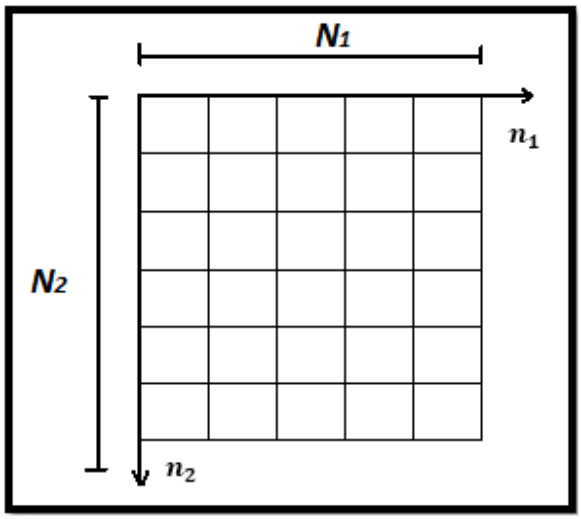
Funciones sobre imágenes

Las funciones más utilizadas sobre imágenes son: cardinalidad, suma de pixeles, media, desviación estándar, igual, entre, truncado, umbral, restricción, y producto interno o escalar. Existen además otras funciones un poco más complicadas que se utilizan para reconocimiento de imágenes, como la autocorrelación.

A continuación se dan las definiciones de estas funciones usando una nueva nomenclatura en al que la imagen $f(x, y)$ se representa como $f\ (n_1, n_2)$, con dimensiones de N_1 puntos sobre el eje horizontal (n_1) y N_2 puntos en el eje vertical (n_2). [vii]

Cardinalidad

Dentro de la función de cardinalidad se realiza la multiplicación del número de pixeles N_1 y N_2 donde N_1 y N_2 son el largo y ancho de la imagen.



$$N_1 = 5$$
$$N_2 = 6$$

$$card(f) = N_1 N_2$$
$$card(f) = 30$$

La cardinalidad brinda muy poca información acerca de la imagen.

Suma de pixeles

La suma de pixeles o también conocido como $sumpix$, es la función en donde se toma cada valor de cada pixel el cual calcula el total de la suma de pixeles conjuntamente.

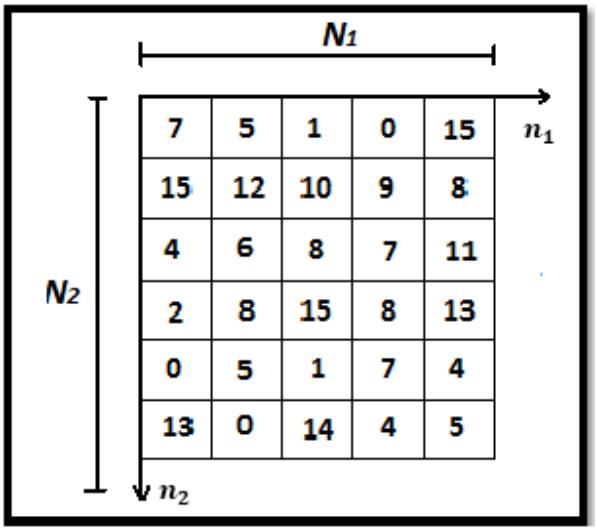


Ilustración 1.1 Imagen de 4 bits

$$sumpix(f) = \sum_{n1,n2} f(n_1, n_2)$$

$$sumpix(f) = 7 + 5 + 1 + 15 + 15 + 12 + 10 + 9 + 8 + 4 + 6 + 8 + 7 + 11 + 2 + 8 + 15 + 8 + 13 + 5 + 1 + 7 + 4 + 13 + 14 + 4 + 5$$

$$sumpix(f) = 217$$

La suma de pixeles, al igual que la cardinalidad, no brinda mucha información acerca de la imagen.

Media

El brillo promedio de una región está definido como el cociente entre la suma de pixeles de una imagen y su cardinalidad.

$$media = u(f) = \frac{\sum_{n_1,n_2} f(n_1, n_2)}{N_1 N_2}$$

$$media = u(f) = \frac{sumpix(f)}{card(f)}$$

Desviación estándar

La estimación sesgada de la desviación estándar, $\sigma(f)$, del brillo junto con una región con N pixeles es llamado la muestra de la desviación estándar y está dada por:

$$\sigma^2(f) = \frac{1}{card(f) - 1} \sum_{n_1 n_2} [f(n_1, n_2) - u(f)]^2$$

Igual

Dado dos imágenes del mismo tamaño f y g, la función Igual tiene como resultado una imagen en donde tiene valores de 1 si y solo si $f(n_1, n_2) = g(n_1, n_2)$, 0 caso contrario.

$$Igual(f, g) = \begin{cases} 1, & f(n_1, n_2) = g(n_1, n_2) \\ 0, & caso\ contrario \end{cases}$$

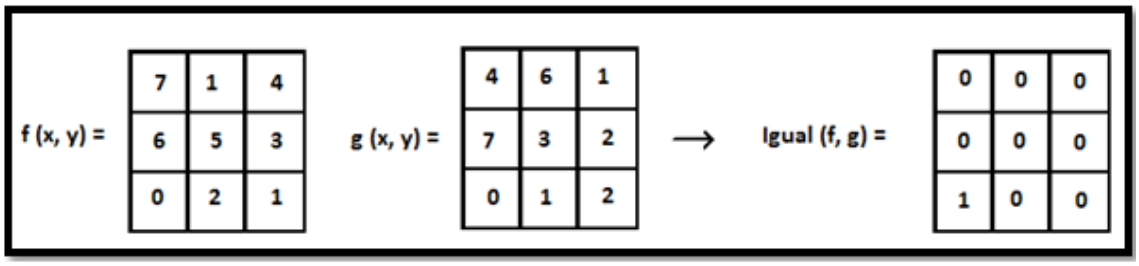


Ilustración 1.2 Operación Igual entre dos imágenes $f(x, y)$ y $g(x, y)$

Entre

Dado una imagen $f(x, y)$ y s,t, donde $s, t \in \mathbb{R}$, la función Entre tiene como resultado una imagen en donde hay valores de 1 si y solo si $f(n_1, n_2), \geq s \ \wedge f(n_1, n_2), \leq t$, 0 caso contrario.

$$Entre(f, s, t) = \begin{cases} 1, & s \leq f(n_1, n_2) \leq t \\ 0, & caso\ contrario \end{cases}$$

$$s, t \in \mathbb{R}$$

g (x, y) =

4	6	1
7	3	2
0	1	2

Entre (g, 1.3, 10) =

1	0	0
0	1	1
0	0	1

Truncado

Dado una imagen $f(x, y)$ y t donde $t \in \mathbb{R}$, la función Truncado da como resultado una imagen la cual tiene valores de $f(n_1, n_2)$, si y solo si $f(n_1, n_2), \geq t$, 0 caso contrario.

$$Truncado(f, t) = \begin{cases} 1, & f(n_1, n_2) \geq t \\ 0, & caso\ contrario \end{cases}$$

$$t \in \mathbb{R}$$

f (x, y) =

7	1	4
6	5	3
0	2	1

Truncado (f, 3) =

7	0	4
6	5	3
0	0	0

Umbral

Esta técnica está basado en un concepto simple. Un parámetro t llamado umbral de luminosidad es elegido y aplicado a la imagen $f(n_1, n_2)$, tal como sigue:

$$umbral(f,t) = \begin{cases} 1, & f(n_1,n_2) \geq t \\ 0, & caso\ contrario. \end{cases}$$

$\mathbf{f(n_1, n_2) = f(x, y) =}$

7	1	4
6	5	3
0	2	1

Umbral (f, 3) =

1	0	1
1	1	1
0	0	0

Restricción

Dada dos imágenes f y g, y g es binaria, la función Restricción está definida de la siguiente manera

$$Rest(f,g) = \begin{cases} f(n_1,n_2), & g(n_1,n_2) = 1 \\ 0, & caso\ contrario \end{cases}$$

f (x, y) =

7	1	4
6	5	3
0	2	1

g (x, y) = Umbral (f, 3) =

1	0	1
1	1	1
0	0	0

Rest [f, Umbral (f, 3)] =

7	0	4
6	5	3
0	0	0

Si se trabaja con la función Restricción y la función Truncado dado la misma imagen $f(x, y)$, se obtiene iguales resultados

$$Rest[f, Umbral(f, t)] = Truncado(f, t)$$

Producto interno

La función interno o también conocido como Producto punto, está definido de la siguiente manera:

$$f(x, y)$$

$$g(x, y)$$

$$Producto\ interno < f, g > = \sum_{n1,n2} f(n_1, n_2)\, g(n_1, n_2)$$

Autocorrelación

En el caso especial cuando se tiene las funciones de imágenes $f\ y\ g$ son lo mismo, la operación de correlación es llamada Autocorrelación. Esto es usado para combinar conjuntamente todas las partes de la imagen, para encontrar estructuras repetitivas. La interpretación de la Autocorrelación de una imagen puede ser entendido como que puede ser impresa transparentemente la imagen y siendo colocada arriba de sí misma, rotada por 180∘. Al deslizar la imagen superior lateralmente en cualquier dirección, el grado de coincidencia con el original subyacente se mide por la función de autocorrelación.[viii]

$$R_{f,g(\tau,\upsilon)} = \sum_{n_1+\tau,n_2+\upsilon \in R} f(n_1+\tau, n_2+\upsilon)\, g(n_1, n_2)$$

Mejoramiento de imágenes

El mejoramiento de imágenes consiste en cambiar algún atributo de ésta con el propósito de mejorar la apreciación de su contenido. El término mejoramiento es relativo a cada caso y lo que para una imagen puede resultar positivamente no es necesariamente bueno para otra.

Los métodos de mejoramiento se pueden aplicar en el dominio del espacio (trabajando pixel por pixel) o en el dominio de la frecuencia en donde se trabaja con las propiedades de la imagen completa.[ix]

Ruido

Las imágenes son propensos a una variedad de tipos de ruido. El ruido es el resultado de errores en el proceso de adquisición de imágenes que dan lugar a valores de píxeles que no reflejan las verdaderas intensidades de la imagen real.[x]

Ilustración 1.3 Imagen con ruido “Sal y Pimienta”

Tipos de ruido

Existen tres tipos de ruido:

- Gaussiano
- Sal y Pimienta
- Impulsivo

Ruido gaussiano

El ruido gaussiano es un ruido estático que tiene una función de densidad de probabilidad (PDF) igual a la de la distribución normal, que también se conoce como la distribución de Gauss. En otras palabras, los valores que el ruido puede asumir son Gaussianos distribuidos.

Imagen original

Ruido Gaussiano

Ruido ‘Sal y Pimienta’

Ocurrencias aleatorias de pixeles blancos y negros sobre la imagen.

Imagen original

Ruido Sal y Pimienta

Ruido Impulsivo

Ocurrencias aleatorias de pixeles blancos sobre la imagen.

Imagen original

Ruido Impulsivo

Filtros pasa bajos y pasa altos

La variabilidad de una imagen es su frecuencia, de modo que una imagen monótona posee frecuencias bajas (imagen cuyos pixeles tienen todos el mismo color, tienen frecuencia cero), en cambio una imagen con gran variabilidad de los niveles de gris de los pixeles poseen altas frecuencias. En términos generales, una imagen con mucha variabilidad, puede visualizarse como tosca al ojo humano, por ello es deseable que se eliminen las altas frecuencias de una imagen. Si se eliminan las altas frecuencias, se eliminan las variaciones bruscas. El proceso mediante el cual se eliminan frecuencias de una imagen se denomina filtrado.

Filtro pasa bajos (suavizado)

Los FPB son empleados para remover el ruido de alta frecuencia de una imagen. Un FPB es un filtro que hace pasar las frecuencias bajas, pero atenúa las más altas comparado con la frecuencia de corte.

Imagen original - 4 niveles

Imagen suavizada (FPB)

Filtro Pasa Altos (Detección de bordes, nitidez)

Un FPA puede ser usado para que una imagen se visualice nítida. Estos filtros destacan los detalles finos de la imagen – opuesto del FPB.

Imagen original

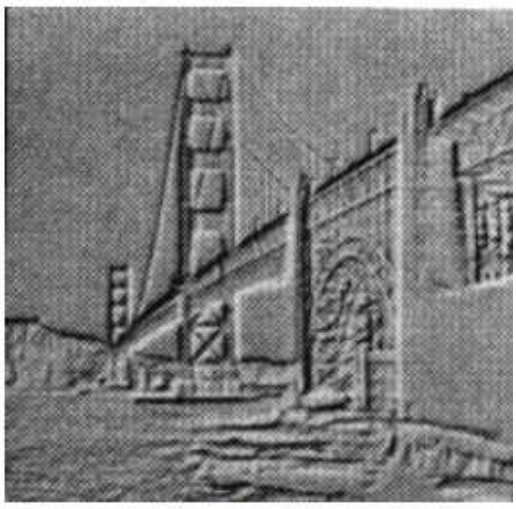


Imagen con FPA

Técnicas para reconocimiento de imágenes

Existen varias técnicas para el reconocimiento de imágenes, siendo más específico, reconocimiento facial de imágenes, de los cuales, los más relevantes son: Características Geométricas, Análisis de Componentes Principales (PCA), Análisis Lineal Discriminante (LDA), Correspondencia de Plantillas, Redes Neuronales Artificiales, Eigenfaces, Transformada Wavelet, Modelos de Cara en 3D, Line Edge Map, entre otras. [xi] Más adelante, en el Capítulo 3, se da a conocer a detalle sobre los métodos Eigenfaces y Line Edge Map ya que son posibles soluciones a implementar para el desarrollo de este proyecto. En este mismo Capítulo se habla acerca de las Redes Neuronales Artificiales (RNA) con el enfoque a reconocimiento de imágenes, ya que las RNA tienen varios usos, como por ejemplo, pronóstico, reconocimiento de texto, reconocimiento de caracteres manuscritos, entre otros.

Redes Neuronales Artificiales

La primera ola de interés en las redes neuronales (también llamadas modelos conexionistas o de procesamiento distribuido paralelo) surgió con la introducción del modelo simplificado de neurona propuesto por McCulloch y Pitts en el año 1943. Estas neuronas fueron presentadas como modelos de las neuronas biológicas y componentes conceptuales para circuitos que podían realizar tareas computacionales. Cuando Minsky y Papert publican

su libro “Perceptrones” en 1969, mostrando las deficiencias en el modelo perceptrón, la mayor parte de los grupos de investigación dirigen entonces sus esfuerzos hacia otros modelos. Solo unos pocos continúan investigando sobre las redes neuronales; los más notables son Teuvo Kohonen, Stephen Grossberg, James Anderson y Kunihiko Fukushima. El interés en las redes neuronales vuelve a emerger a principios de los años ochenta con nuevos avances teóricos (principalmente el algoritmo de la retro-propagación del error) y gracias a la mayor capacidad de computo por parte de los procesadores. Este interés se refleja en la gran cantidad de congresos, conferencias, revistas, libros, etc, centrados en las redes neuronales.

Las redes neuronales pueden ser descritas como “modelos computacionales” con una serie de propiedades tales como la habilidad de adaptarse o aprender, generalizar o agrupar u organizar datos, y cuya operación está basada en procesamiento paralelo. Algunas de estas propiedades pueden ser reclamadas por otros modelos; sin embargo, la pregunta clave es hasta dónde la aproximación basada en redes neuronales es mejor para una aplicación que otros modelos. Hasta la fecha no hay una respuesta clara a esta pregunta. En muchos libros, al hablar de redes neuronales se describe su paralelismo con modelos biológicos. Pero existe tan poco conocimiento sobre los sistemas biológicos que los modelos de redes neuronales artificiales son una “súper simplificación” de estos sistemas.

Una red neuronal artificial (RNA) está constituida por un conjunto de células de procesamiento llamadas neuronas. Una neurona es un dispositivo sencillo formado por una serie de entradas y una salida. Cada neurona acepta como entrada las salidas procedentes de otras neuronas, siendo la entrada efectiva a la neurona la suma ponderada de las entradas reales a dicha neurona. Cada neurona se caracteriza por su estado de activación, que es un valor que oscila entre 0 y 1. Si el estado de activación de una neurona es 0, la neurona no está activada, mientras que cualquier valor distinto de 0 corresponde a una neurona activa. La salida de la neurona es el estado de activación. Cada neurona realiza una tarea sencilla; recibe la información de entrada de los vecinos o del exterior y la usa para calcular una señal de salida que se propaga a otras unidades.

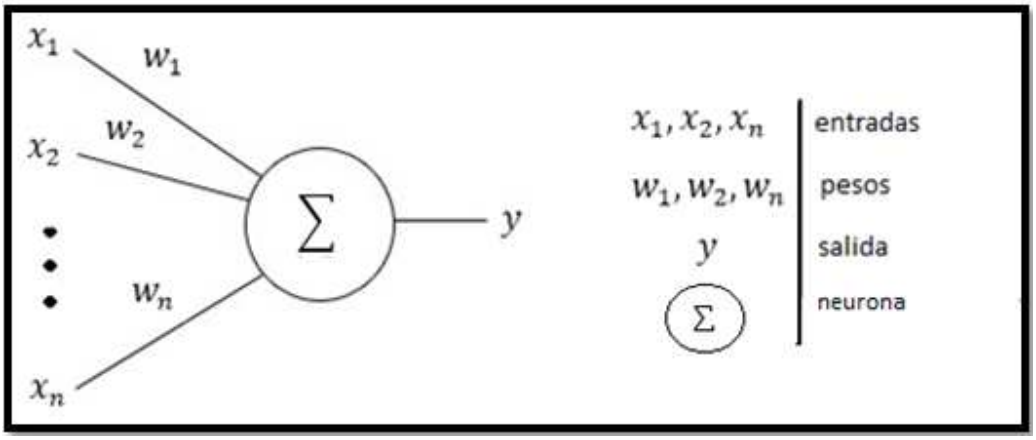


Ilustración 1.4 Modelo básico de una neurona artificial

En general, las neuronas se organizan en capas. Dependiendo de su función en la red, se distinguen tres tipos de unidades: las unidades cuya activación son los datos de entrada del problema (unidades de entrada); las unidades cuya salida es la salida del problema (unidades de salida); y el resto de unidades, llamadas unidades ocultas (ya que no son "visibles" desde el exterior).[xii]

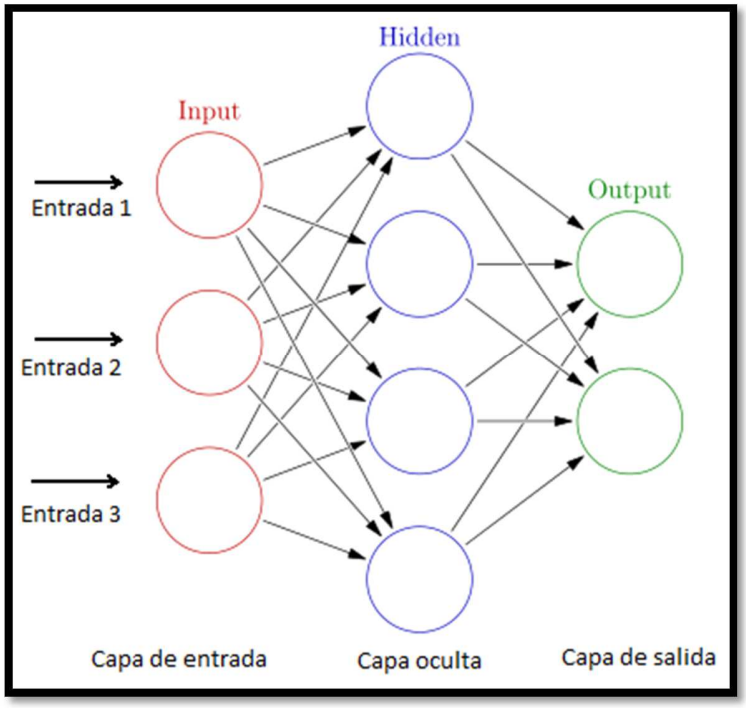


Ilustración 1.5 Modelo de una red neuronal

Redes de Retropropagación (Backprogragation). Aplicación al procesamiento de imágenes

Las redes de retropropagación incluyen elementos ocultos no lineales entre las unidades de entrada y de salida. La habilidad de las redes neuronales para extraer información útil fue inicialmente aplicada a caras humanas por Cottrell y Fleming. Entrenaron una red de tres capas con 4096 unidades de entrada, 80 unidades en la capa oculta de tipo sigmoidal y 4096 unidades de salida, para comprimir un conjunto de imágenes de caras y de no-caras. El conjunto de entrenamiento consistía en 64 imágenes de caras (5 o 6 imágenes de 11 personas) y 13 imágenes de no-caras. La base de datos completa estaba formada por 231 imágenes de las cuales 204 eran caras (entre 5 y 20 imágenes de 17 personas) y 27 no contenían caras. Todas las caras en el conjunto de entrenamiento se le suministraron a la red de comprensión. La representación formada en la capa oculta fue utilizada como entrada de una red de 80 unidades de entrada y 14 neuronas de salida con un rango [-1, 1], entrenada para clasificar "apariencia de cara", genero e identidad.

El método backpropagation (propagación del error hacia atrás), a pesar de sus limitaciones, ha ampliado de forma considerable el rango de aplicaciones de las redes neuronales. El funcionamiento de la red backpropagation (BPN) consiste en el aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo: primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada.

La importancia de la red backpropagation consiste en su capacidad de auto-adaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe entre

un conjunto de patrones de entrada y sus salidas correspondientes. Es importante la capacidad de generalización, facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento. La red debe encontrar una representación interna que le permita generar las salidas deseadas cuando se le dan entradas de entrenamiento, y que pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje para clasificarlas.[xiii]

Estructura y aprendizaje de la Red Backpropagation

En una red Backpropagation existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas. Cada neurona de una capa (excepto las de entrada) recibe entradas de todas las neuronas de la capa anterior y envía su salida a todas las neuronas de la capa posterior (excepto las de salida). No hay conexiones hacia atrás feedback ni laterales entre las neuronas de la misma capa. La aplicación del algoritmo tiene dos fases, una hacia delante y otra hacia atrás. Durante la primera fase el patrón de entrada es presentado a la red y propagado a través de las capas hasta llegar a la capa de salida. Obtenidos los valores de salida de la red, se inicia la segunda fase, comparándose estos valores con la salida esperada para así obtener el error. Se ajustan los pesos de la última capa proporcionalmente al error. Se pasa a la capa anterior con una retropropagación del error, ajustando los pesos y continuando con este proceso hasta llegar a la primera capa. De esta manera se han modificado los pesos de las conexiones de la red para cada patrón de aprendizaje del problema, del que conocíamos su valor de entrada y la salida deseada que debería generar la red ante dicho patrón. La técnica Backpropagation requiere el uso de neuronas cuya función de activación sea continua, y por lo tanto, diferenciable. Generalmente, la función utilizada será de tipo sigmoidal.[xiv]

Historia

En el año de 1920 se tenía la técnica “Sistema de transmisión de imágenes por cable Bartlane” que fue usado para transmitir imágenes de periódicos a través del Atlántico. Las imágenes fueron codificadas, enviadas por telégrafo, impreso por un telégrafo especial. Tomaba alrededor de 3 horas enviar una imagen, los primeros sistemas soportaban 5 niveles

de gris. En 1964, el laboratorio de la NASA Jet Propulsion comenzó a trabajar en algoritmos computacionales para mejorar imágenes de la luna. Las imágenes fueron trasmitidas por la sonda Ranger 7.[xv]

Muchas de las técnicas de procesamiento de imágenes, cuando fueron desarrollados en los 60 en el laboratorio Jet Propulsion, MIT, Bell Laboratories, Universidad de Maryland, y algunas otras instituciones de investigación, con aplicaciones para imágenes de satélites, medicina, reconocimiento de caracteres y mejoramiento de fotografías. El costo del procesamiento fue alta teniendo los equipos de computación de esa época. Eso cambió en el año de 1970, cuando el procesamiento digital de imágenes fue proliferado como las computadoras y el hardware se tornó disponible. Entonces las imágenes pudieron ser procesadas en tiempo real, para algunos problemas puntuales como conversión estándar de televisión. Como propósito general, las computadoras se volvieron rápidas, comenzaron a asumir el papel de un hardware, y para las operaciones más especializadas y en la informática intensiva.

Con los ordenadores rápidos y procesadores de señal disponibles en la década de 2000, el procesamiento de imagen digital se ha convertido en la forma más común de tratamiento de la imagen y, en general, se utiliza porque no sólo es el método más versátil, pero también el más barato. La tecnología de procesamiento digital de imágenes para aplicaciones médicas se instaló en el Space Foundation Space Technology Hall of Fame en 1994. En 2002 Raanan Fattal introdujo Gradiente de procesamiento de imágenes de dominio, una nueva manera para procesar imágenes en las que las diferencias entre pixeles se manipulan en lugar de los propios valores del píxel.[xvi]

Propiedades

Existen varias formas de clasificar y caracterizar las operaciones en las imágenes. La razón para hacerlo es para entender que tipos de resultados esperaríamos lograr dado un tipo de operación o cual sería la carga computacional asociado a una operación dada.

Aplicaciones

El procesamiento de imágenes no está solo limitado a ajustar la resolución espacial de imágenes capturadas por una cámara. No está limitada al ajuste del brillo de una fotografía, etc. Más bien, es mucho más que eso.

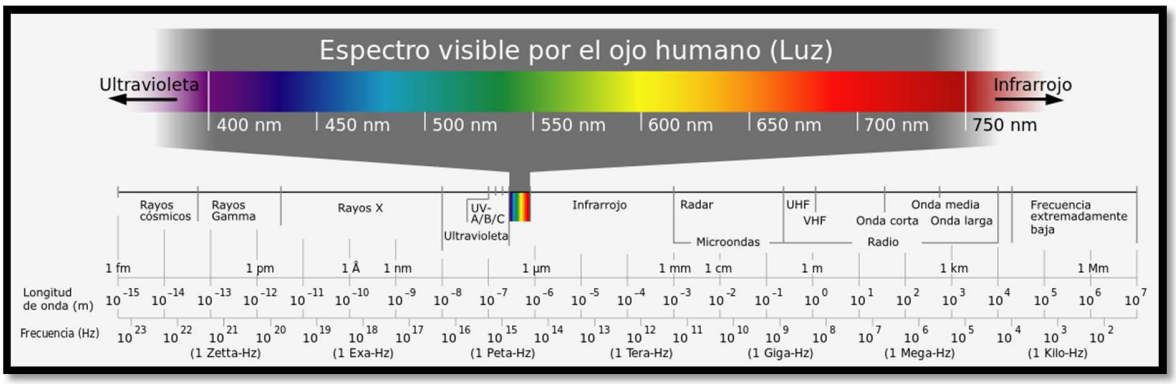


Ilustración 1.6 Espectro Electromagnético

El espectro visible principalmente incluye siete colores diferentes que son comúnmente limitados como (VIBGOYR). VIBGOYR significa violet, indigo, blue, green, orange, yellow y red. Pero eso no anula la existencia de otras cosas en el espectro. Nuestro ojo humano sólo puede ver la parte visible, en la que vemos a todos los objetos. Pero una cámara puede ver las otras cosas que a simple vista no es capaz de ver. Por ejemplo: rayos X, rayos gamma, etc. Por lo tanto el análisis de todo eso también se hace en el procesamiento digital de imágenes. Algunos de los principales campos en donde el procesamiento digital de imágenes es ampliamente mencionado y usado

- Nitidez y restauración de imágenes
- Medicina
- Teledetección
- Transmisión y codificación
- Visión de máquinas/robots

Nitidez y restauración de imágenes

Se refiere aquí a procesar las imágenes que han sido capturadas por la cámara moderna para que sean una mejor imagen o manipular esas imágenes en forma de lograr el resultado deseado. Esto incluye zoom, desenfoque, nitidez, escala de grises para la conversión de color, detección de bordes y viceversa, de recuperación de la imagen y de reconocimiento de imágenes. Los ejemplos más comunes son:

Imagen original

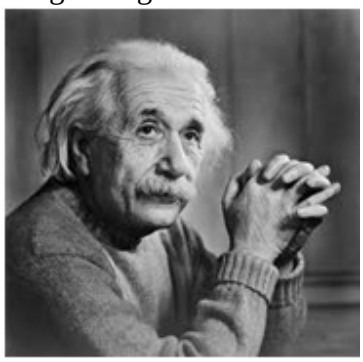


Imagen amplificada (zoom)

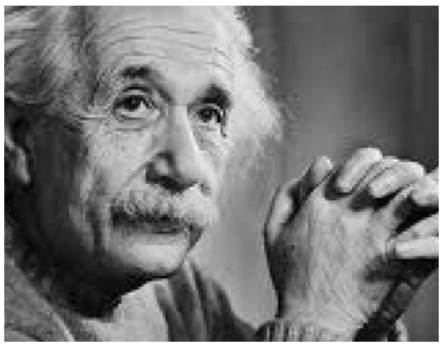


Imagen borrosa

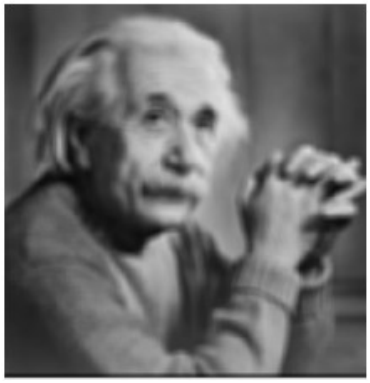
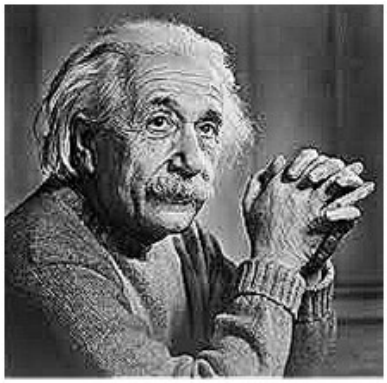


Imagen nítida



Bordes

Medicina

Las aplicaciones comunes en el campo de la medicina son

- Imágenes rayos Gamma
- Escáner de Tomografía por Emisión de Positrones (PET)
- Imágenes rayos X
- Tomografía Computarizada
- Imágenes Ultravioletas

Teledetección

El área de la tierra es escaneada por un satélite o aeroplano y luego se analiza para obtener información sobre él. Una aplicación particular de procesamiento de imágenes en el campo de la teledetección es detectar daños a la infraestructura causados por un terremoto. Así, una solución a esto se encuentra en el procesamiento de imagen digital. Una imagen de la zona afectada es capturada desde arriba y luego se analiza para detectar los diversos tipos de daños causados por el terremoto.

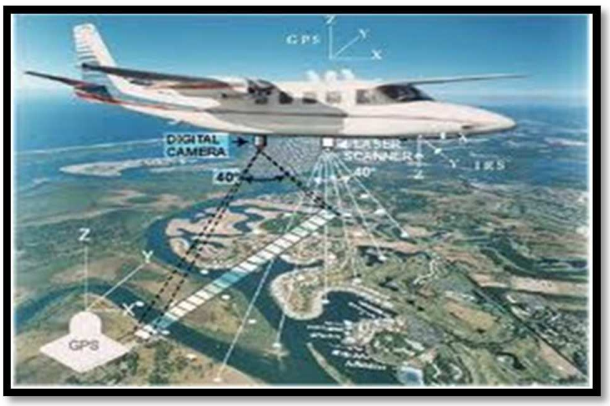


Ilustración 1.7 Teledetección desde un aeroplano

Los pasos claves incluidos en el análisis son

1. La extracción de bordes
2. Análisis y mejoramiento de varios tipos de bordes

Transmisión y codificación

La primera imagen que ha sido transmitido a través del cable desde Londres a Nueva York a través de un cable submarino. La imagen que se envió se muestra a continuación.

Ilustración 1.8 Imagen transmitida por un cable submarino

La imagen que fue enviada tomó 3 horas en alcanzar su destino. Hoy en día somos capaces de ver videos en vivo, o programas desde otro continente con solo unos segundos de retraso en vivo. Esto significa que se ha hecho bastante trabajo en esta área. Este campo no solo se enfoca a la transmisión sino también en codificación. Diferentes formatos han sido desarrollados para el ancho de banda para codificar fotos y luego transmitir a través de Internet.

Visión máquina/robot

Aparte de los muchos desafíos que tienen las máquinas, uno de ellos es el incremento de la visión, el cual permitiría ver objetos con mayor alcance, identificarlos, ver obstáculos, entre otros. Mucho trabajo se ha aportado por este campo y existe otra área de visión computacional que ha sido introducida para trabajar en ella.

Detección de obstáculos

La detección de obstáculos es una tarea común que ha sido realizado a través de procesamiento de imágenes, identificando diferentes tipos de objetos en la imagen para luego calcular distancias entre las máquinas/robots y los obstáculos.

Ilustración 1.9 Identificación de objetos en una calle principal a través de cámaras de vigilancia

Seguidor de líneas

La mayoría de los robots de hoy tienen tareas que se logran a través del seguimiento de líneas. Esto ayuda a un robot a moverse en su camino y realizar simples tareas. Esto también se ha logrado a través de procesamiento de imágenes.

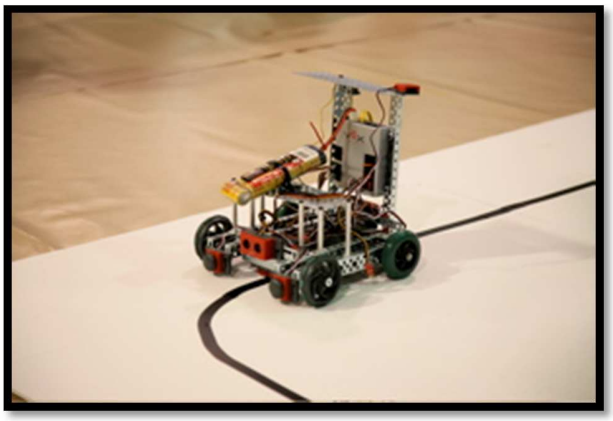


Ilustración 1.10 Seguimiento de líneas realizado por un robot

Capítulo 2

Herramientas de computación numérica

Las herramientas de computación numérica son aquellas que permiten a ingenieros y científicos a realizar tareas, cálculos complejos que son usados para investigaciones en diferentes disciplinas. Estas herramientas tienen diferentes usos, como por ejemplo permite trabajar con procesamiento de imágenes, procesamiento de señales, simulaciones de sistemas fluidos, análisis estadístico, mejoramiento de imágenes, optimización numérica, diseño de modelos, manipulación de matrices, entre otros. Herramientas como MATLAB, GNU Octave, FreeMat, dan gráficos de salida con visualizaciones en 2D/3D para un mejor estudio.

Cada herramienta de computación numérica tiene su propio lenguaje de programación en donde algunos de ellos soportan varios paradigmas de programación como programación imperativa, procedural, OOP, permitiendo de esa manera la escritura de código, scripts y posteriormente su ejecución; para ciertas herramientas de computación numérica, los lenguajes usados al momento de trabajar pueden llegar a ser lentos cuando se los ejecuta ya que son lenguajes interpretados, no compilados.

Herramientas de Software Libre y Software Privativo

A continuación se presenta un cuadro mostrando casi todas las herramientas de computación numérica relacionados al procesamiento de imágenes que sean libres y privativas:

Herramienta	Creador	Desarrollado desde	Primer lanzamiento público	Última versión estable	Costo (USD)	Licencia	Descripción
ADMB	D. Fournier, ADMB Project	N/A	N/A	nov-12	Free	BSD 3-clause (aka new) License	Diferenciación automática hace que sea muy adecuado para problemas de minimización complejos
Ch	SoftIntegration	N/A	2001	7.0/ 2012	$399 (commercial) / $199 (academic) / Free (student)	privado	C/C++ basado en computación numérica y trazado de gráficos
DADiSP	DSP Development	1984	1986	6.5 / 2010	$1995 (commercial) / $129 (academic) / Free (student)	privado	Cálculos numéricos para la ciencia y la ingeniería.
Dyalog APL	Dyalog Ltd.	1981	1983	13.2 / Enero 2013	£850/year or 2% royalty (non-commercial £50, edu / unreg Free)	privado	Un dialecto moderno de APL, mejorado con características de programación funcional y orientada a objetos.
Euler Math Toolbox	R. Grothmann	1987	1988	9.4 / 2010	Free	GPL	Sistema de álgebra computacional a través de la interfaz con Máxima
Fityk	M. Wojdyr	N/A	2002	1.0.1 / 2011	$115 (1.x binaries) / Free (source code and 0.x binaries)		gráficos interactivos, scripts, especializados en el ajuste de curvas y pico apropiado, sólo 2D
FreeMat	Samit Basu	N/A	N/A	4.1 / Noviembre 28, 2011	Free	GPL	Interfaz sin código de C externo, C ++ y Fortran. Parcialmente compatible con MATLAB.
GAUSS	Aptech Systems	N/A	1984	14-oct-13	No libre	privado	N/A
IGOR Pro	WaveMetrics	1986	1988	6.36 (6.3.6.4) / 2014	$595 (commercial) / $435 (academic) / $85 (student)	privado	Gráficos interactivos, programable, 2D / 3D, que se utiliza para la ciencia y la ingeniería, grandes conjuntos de datos.
J	Jsoftware	1989	1990	J802 / Julio 9, 2014	Free	GPL	Acceso en línea / Aplicación Library (JAL)

Herramienta	Creador	Desarrollado desde	Primer lanzamiento público	Última versión estable	Costo (USD)	Licencia	Descripción
Julia	Jeff Bezanson, Stefan Karpinski, Viral B. Shah,	2009	2012	0.3.1 / 2014	Free	MIT License	Lenguaje de código abierto y un entorno para computación científica. Las llamadas directas de C funciones de código (sin envoltorios o APIs especiales necesarias). Diseñado para la computación en nube paralelo en mente con LLVM JIT como backend. Ligero "verdes" threading (co-rutinas).
LabVIEW	National Instruments	N/A	1985	2010 / Agosto 2010	$1249 (commercial) / $79.95 (student)	privado	Enfoques gráficos y textuales (guión archivo .m) la programación
Maple	Maplesoft	1980	1982	18 / Marzo 5, 2014	$2275 (commercial) / $99 (student)	privado	Principalmente un sistema de álgebra computacional
Mathcad	Parametric Technology Corporation	1985	1986	15.0; Prime 3.0 / Octubre 1, 2013	$1195 (commercial) / $99 (student)	privado	Principalmente un sistema de álgebra computacional
Mathematica	Wolfram Research	1986	1988	10.0.2(Diciembre 10, 2014)	Free (Raspberry Pi), $2495 (commercial) / $145 (student) / $295 (personal)	privado	Principalmente un sistema de álgebra computacional
MATLAB	MathWorks	1970	N/A	2013a	$2150 (commercial) / $99 (student)	privado	Cálculo numérico y simulación con 2D / 3D de visualización ampliada
Maxima	MIT Project MAC y Bill Schelter et al	1967	1982	5.28.0 / Agosto 27, 2012	Free	GPL	Principalmente un sistema de álgebra computacional
GNU Data Language	Marc Schellens	N/A	2004	0.9.1 / 2011	Free	GPL	Reemplazo de IDL / PVWAVE
GNU Octave	John W. Eaton	1988	1993	3.8.2	Free	GPL	General de paquete de computación numérica con un montón de módulos de extensión. Sintaxis mayormente compatible con MATLAB

Herramienta	Creador	Desarrollado desde	Primer lanzamiento público	Última versión estable	Costo (USD)	Licencia	Descripción
Origin	OriginLab	N/A	1991	9.1 SR0 / Oct. 2013	$1095 (std.)/$1800 (Pro)	privado	Software integrado de gráficos de análisis de datos para la ciencia y la ingeniería. Marco de gráficos multi-capa flexible. 2D, 3D y tipos de gráficos estadísticos. Herramienta integrada de la digitalización. Análisis con cálculo automático y generación de informes. Built-in de secuencias de comandos y lenguajes de programación.
Perl Data Language	Karl Glazebrook	1990s	c. 1997	2.4.10 / 2012	Free	Artistic License	Se utiliza para la astrofísica, la física solar, la oceanografía, la biofísica y la simulación. 2D trazando a través PGPLOT, encuadernaciones PLplot; 3D a través de GL.
PSPP	Ben Pfaff	1990s	1990s	0.8.4 / Septiembre 27, 2014	Free	GPL v.3	FOSS alternativa a IBM SPSS. Análisis de los datos incluidos en la muestra, las frecuencias, la comparación entre las pestañas de medios (pruebas t y ANOVA de una vía); regresión lineal, regresión logística, la fiabilidad (Alfa de Cronbach, no el fracaso o Weibull)
R	R Foundation	1997	1997	2.15.3 / Marzo 1, 2013	Free	GPL	Principalmente para las estadísticas, pero hay muchas interfaces de software de fuente abierta numérica
Sage	William Stein	N/A	2005	6.4.1	Free	GPL	Programable, incluye álgebra computacional, trazado 2D + 3D. Interfaces de software de código abierto demasiados. Basado en la Web HTTP o HTTPS interfaz
SAS	Anthony Barr y James Goodnight	1966	1972	9.4 / Julio 10, 2014	No libre	Privado	N/A
SCaViS	S.Chekanov (jwork.org)	2005	2005	1.0 / 2013	Free	Gratis para uso no comercial	Matemáticas, cálculos simbólicos, álgebra, análisis de datos, minería de datos, 2D interactivo / gráficos en 3D, Java scripts, que se utiliza para la ciencia y la ingeniería, entre plataformas

Herramienta	Creador	Desarrollado desde	Primer lanzamiento público	Última versión estable	Costo (USD)	Licencia	Descripción
S-Lang	John E. Davis	N/A	1992	2.2.0 / 2009	Free	GPL, Artistic License	Disponible como un independiente (slsh) y embebidos intérprete
Scilab	Scilab Enterprises	1990	1994	5.4.1 / Abril 2, 2013	Free	CeCILL license	Código abierto, soportado por la comunidad. Programable, el apoyo directo de trazado 2D + 3D. Interfaces con muchos otros paquetes de software. Las interfaces para módulos externos escritos en C, Java, Python y otros lenguajes. Sintaxis del lenguaje similar a MATLAB. Se utiliza para la computación numérica en la ingeniería y la física.
SPSS	Normal H. Nie, Dale H. Bent, and C. Hadlai Hull	N/A	1968	22.0 / Agosto 13, 2013	No libre	privado	N/A
Sysquake	Calerga	N/A	1998	5.0 / 2013	free / $2500 (Pro, commercial) / $1000 (Pro, academic)	privado	Gráficos interactivos
TK Solver	Universal Technical Systems, Inc	1970	1982	5.0.141 / 2011	$399 commercial / $49 (student)	privado	Cálculo numérico y desarrollo de aplicaciones basadas en reglas
VisSim	Visual Solutions	N/A	1989	7.0a / 2008	$495-$2800 (commercial)	privado	Lenguaje Visual para la simulación y el diseño basado en modelos. Se utiliza en los negocios, la ciencia y la ingeniería. Realiza escalar o matriz compleja ODE basado resolver con la optimización paramétrica. Tiene trazado 2D y 3D, animación 3D, y de transición de estado incorporado.
Yorick	n/a	n/a	n/a	2.1.06 / Abril 17, 2010	Free	GPL	Es de código abierto. Programable, exigible trazado 2D + 3D. Sintaxis del lenguaje similar al C. Interfaz con otros paquetes de software a través de llamadas C.

Herramientas para procesamiento de imágenes

Existen muchos lenguajes/paquetes para el procesamiento de imágenes, como Mathematica, Maple, Origin, R, SPSS, entre otros. A continuación se da un resumen sobre los lenguajes/paquetes más utilizados por ingenieros y científicos para proyectos e investigaciones a nivel mundial.

MATLAB

- Es una ambiente computacional numérico multi-paradigma y un lenguaje de cuarta generación. Es desarrollado por MathWorks, MATLAB permite la manipulación de matrices, funciones de gráficos y datos, implementación de algoritmos, creación de interfaces de usuario, y la interconexión con programas escritos en otros lenguajes incluyendo C, C++, Java, Fortran y Phyton.[xviii]

Ventajas y desventajas

- MATLAB es un lenguaje interpretado para computación numérica. Permite el manejo de cálculos numéricos y visualizar los resultados sin la necesidad de programación complicada y consumo de tiempo. MATLAB permite a sus usuarios resolver problemas precisos, producir gráficos fácilmente y producir código eficiente.
- La gran desventaja que posee MATLAB es que es un lenguaje interpretado que puede llegar a ser lento debido a malas prácticas de programación. Otra desventaja, no en si del lenguaje, es el costo de la herramienta, la versión comercial tiene un costo alrededor de los US$ 2150, y la versión estudiante tiene el costo de US$ 99. Existen otras herramientas similares a MATLAB que no tienen costo.[xix]

GNU OCTAVE

- Es un lenguaje de programación de alto nivel, principalmente enfocado para la computación numérica. Provee de una interface de línea comando para

solventar problemas numéricos lineales y no lineales. Puede ser usado también como un lenguaje orientado a procesamiento por lotes (batch processing).[xx]

Ventajas y desventajas

- Es una herramienta software libre que provee de un reemplazo a MATLAB al menos para su principal funcionalidad. Es compatible con MATLAB, comparte algunas características.[xxi] Es multi-plataforma disponible en 19 lenguajes con licencia GPL.
- Por otro lado, Octave tiene ciertas desventajas como la ejecución de programas pueden llegar a ser relativamente lentos comparado con programas pre-compilados escritos en C o Fortran. Octave no está diseñado para matemática analítica o simbólica.[xxii]

SCILAB

- Es un paquete software libre, multi-plataforma para computación numérica y un lenguaje de programación de alto nivel. Puede ser usado para procesamiento de señales, análisis estadístico, mejoramiento de imágenes, simulaciones dinámicas fluidas, optimización numérica y modelamiento, simulación de sistemas dinámicos explicita e implícita y manipulación simbólica.[xxiii]

Ventajas y desventajas

Tiene licencia CeCILL GPL, el cual puede ser utilizado por cualquier persona sin ningún costo. SCILAB contiene un paquete libre llamado Xcox (basado en Scicos) para modelamiento y simulación de sistemas dinámicos explícitos e implícitos, Xcos es open source, es equivalente a Simulink de MATLAB. SCILAB tiene muchos toolboxes contribuidos para diferentes tareas como las siguientes[xxiv]

- Scilab Image Processing Toolbox (SIP)
- Scilab Wavelet Toolbox
- Scilab Java and .NET Module
- Scilab Remote Access Module

- Scilab MySQL
- Equalis Communication Systems Module
- Equalis Signal Processing Module
- SoftCruncher Performance Accelerator

FREEMAT

- Es una ambiente libre software libre para computación numérica y es un lenguaje de programación similar a MATLAB y GNU Octave. Adicionalmente tiene soporte con muchas funciones de MATLAB. Permite trabajar con gráficos y visualizaciones en 3D. Es multi-plataforma.[xxv]

Ventajas y desventajas

- Un área del desarrollo de FREEMAT, es la programación en paralelo y desarrollo. El algoritmo en FREEMAT es cubierto por tres tecnologías potenciales. La simplificación de protocolos de mensajes MPI (Message passing interface), Multi-core, Alto nivel de abstracción.[xxvi]

Herramientas para Redes Neuronales Artificiales (RNA)

Hay una gran cantidad de software relacionado a RNA que permiten la simulación, investigación y desarrollo de las redes neuronales. Las RNA son adaptadas a conceptos de redes neuronales biológicas, y en algunos casos, una larga colección de sistemas adaptivos como inteligencia artificial y machine learning. A continuación, se da a conocer las diferentes herramientas para el manejo de RNA en donde se observa el tipo de licencias que poseen, disponibilidad en plataformas, el o los lenguajes de programación que permiten ejecución de programas, entre otros.

Software	Licencia	Plataforma	Lenguaje	Enfoque	Estilo de código	Ambiente Virtual	Backpropagation	Neuronas Soportadas	Cálculo en paralelo
Emergent Neural Network Simulation System	GNU GPL	Windows, OS X, Linux, Unix	C++	redes	Visual Scripts C++	Física de cuerpos rígidos gráficos 3D Archivo del modelo 3D de portaciones Apoyo robótica completa	Tranversal-Entropía Anticipativo (SRN) Continuo en tiempo real	Punto Biológico	MPI GPU
XNBC	GNU GPL	Windows, OS X, Linux, Unix	C++	redes neuronales	Visual	N/A	N/A	Punto Biológico	N/A
Wolfram Mathematica Neural Networks	Privado	N/A	Mathematica	redes	Scripts	Feedforward radial Basis Dinámico (recurrente)	N/A	N/A	N/A
VERTEX (Virtual Electrode Recording Tool for EXtracellular potentials)	Otro	Matlab (Windows, OSX, Linux)	Matlab	redes neuronales	Matlab Scripts	N/A	N/A	Punto 3d	MPI Distribuido
Torch5	GNU GPL	Windows, OS X, Linux, Unix	C	N/A	Lua Scripts	N/A	N/A	N/A	N/A

Software	Licencia	Plataforma	Lenguaje	Enfoque	Estilo de código	Ambiente Virtual	Backpropagation	Neuronas Soportadas	Cálculo en paralelo
Topographica Neural Map Simulator	BSD	Linux, Mac OS X, Windows	Phyton, C++	redes	Phyton Scripts, C++	Blender Interface	N/A	Punto Biológico	MPI Distribuido
Theano	N/A	N/A	C, Phyton	redes	Scripts	N/A	N/A	N/A	GPU
Stuttgart Neural Network Simulator	Otro FOSS	Windows	C++	redes	Visual	N/A	Quickprop	Punto	N/A
SpikeNET	GNU GPL	N/A	C++	redes	Configuración archivos	N/A	N/A	Biológico	N/A
SpikeFun	Otro	Windows	C/C++	redes	Configuración archivos	N/A	N/A	3d	N/A
Simbrain	GNU GPL	Java	Java	redes	Visual scripts	Simple 2D world	N/A	Punto Biológico	N/A
Siemens ECANSE	Privado	Windows	Visual C++		Visual scripts	N/A	N/A	N/A	N/A
Sharky Neural Network	Privado	Windows 2000/XP/Vista/7/8		redes	N/A	N/A	Si	Punto	N/A
SNNAP (Simulator for Neural Networks and Action Potentials)	Privado	Java	Java	redes neuronales	N/A	N/A	N/A	Punto Biológico	N/A
PyNeurGen	N/A	N/A	Phyton	redes	Script	N/A	Si	Punto	N/A
PyDSTool	BSD	Phyton	Phyton	neuronas	Phyton Scripts	N/A	N/A	Punto	N/A
Peltarion Synapse	Privado	Windows	.NET	N/A	Visual scripts .NET	N/A	Quickprop Levenberg-Marquardt Recurrent	Other FOSS	N/A
PSICS (Parallel Stochastic Ion Channel Simulator)	GNU GPL	Java	Java Fortran	N/A	Visual	N/A	N/A	Punto 3d Biológico	N/A
PDPTool	GNU GPL	Windows, OS X	MATLAB	redes	Visual MATLAB	N/A	Backprogragation recurrente	Punto	N/A

Software	Licencia	Plataforma	Lenguaje	Enfoque	Estilo de código	Ambiente Virtual	Backpropagation	Neuronas Soportadas	Cálculo en paralelo
PCSIM	GNU GPL	OS X, Linux, Unix	C++, phyton	neuronas	Scripts Java	N/A	posible	Punto Biológico	MPI
OpenNN	GNU LGPL	Multiplataforma	C++	redes	c++ main function	N/A	Si		N/A
Nodus	Privado	N/A	Fortran	redes neuronales	N/A	N/A	N/A	Biológico	N/A
Neuroph	Otro FOSS	Java	Java	redes	Java	N/A	Multicapa perceptron con Backpropagation	Punto	N/A
NeuronC	GNU GPL	OS X,Linux,Unix	C/C++	redes neuronales	Visual Scripts (interpretado) Scripts (compilado a DLL)	No	posible	Biológico	MPI
NeuroSolutions	Privado	Windows, Linux (w/o GUI) Unix (w/o GUI)	Visual C++	redes	Visual C (compilado DLL) MATLAB	N/A	Generalized Feedforward Levenberg-Marquardt	N/A	GPU
NeuroJet	Otro FOSS	Multiplataforma	C/C++	neuronas	MATLB DSL	N/A	Excitatory Synaptic Modification	Biológico	MPI
Neural Designer	Privado	N/A	C++	redes	GUI	N/A	Si	N/A	N/A
NetMaker	Privado	N/A	C, C#, ASM	redes	GUI	N/A	Quickprop Levenberg-Marquardt BPTT	N/A	N/A
Nengo	Otro FOSS	Multiplataforma	Java, Phyton	redes neuronales	GUI	No	N/A	Punto Biológico	N/A
NeMo		Linux, Windows, Mac OS	C++	redes	C++ Phyton C y Matlab	N/A	N/A	Punto Biológico	GPU
NEURON	GNU GPL	Multiplataforma	C, C++, Fortran	redes neuronales	Visual Scripts (interpretado) Scripts (compilado a DLL)	No	posible	Punto 3d Biológico	MPI
NEST (NEural Simulation Tool)	GNU GPL	OS X, Linux, Unix	C++, phyton	redes neuronales	Phyton scripts interpretador SLI	N/A	N/A	Punto 3d Biológico	MPI Distribuido

Software	Licencia	Plataforma	Lenguaje	Enfoque	Estilo de código	Ambiente Virtual	Backpropagation	Neuronas Soportadas	Cálculo en paralelo
Mvaspike	Otro	N/A	N/A	redes neuronales	N/A	N/A	N/A	Punto Biológico	N/A
MOOSE (Multiscale Object-Oriented Simulation Environment)	GNU LGPL	Windows, OS X, Linux, Unix	C++, phyton	redes neuronales	Scripting Phyton GUI	N/A	N/A	Punto 3d Biológico	MPI
MATLAB Neural Network Toolbox	Privado	Windows, OS X Linux, Solaris 64-bit	N/A	redes	Visual Matlab	N/A	N/A	N/A	N/A
LENS (The light, efficient neural network simulator)	Otro FOSS	Windows, OS X,Linux,Unix	C	redes	Visual	N/A	Quickprop Feed-forward recurrent RBPTT/CRBPTT	Punto	N/A
KInNeSS (KDE Integrated NeuroSimulation Software)	GNU GPL	Linux	C++	N/A	N/A	N/A	N/A	N/A	N/A
JavaNNS	Privado	Java	Java	redes	Visual	N/A	Quickprop Feed-forward recurrent RBPTT/CRBPTT	Punto	N/A
HHsim	GNU GPL	Linux, OS X,Windows	C++ y CUDA	redes	Codificación o interfaces de terceros	Si	Si	Punto Biológico	GPU
GENESIS (the GEneral NEural SImulation System)	GNU GPL	Linux, OS X,Windows	C	neuronas	Visual Scripts (interpretado) Scripts (compilado a DLL)	No	posible	Punto Biológico	N/A
FANN (Fast Artificial Neural Network Library)	GNU LGPL	Windows, OS X, Linux, Unix	C	redes neuronales	visual Decenas de enlaces de lenguaje	No	RPROP iRPROP Quickprop	Punto	N/A

Software	Licencia	Plataforma	Lenguaje	Enfoque	Estilo de código	Ambiente Virtual	Backpropagation	Neuronas Soportadas	Cálculo en paralelo
Encog	Otro FOSS	Java, .NET	Java, C#	redes	Java C# otros lenguajes .NET	N/A	Counterpropagation Elman Recurrent	Punto	GPU
Catacomb2 (Components And Tools for Accessible COmputer Modeling in Biology	GNU GPL	Java	Java	redes neuronales	Visual	N/A	N/A	Biológico	N/A
CX3D	GNU GPL	Java	Java	redes	Java	N/A	N/A	Biológico	N/A
CNS (Cortical Network Simulator)	GNU GPL	Windows, OS X, Linux,Solaris 64-bit	C	redes neuronales	Matlab C (via mex)	N/A	3D convolutional networks	Biológico	GPU
Brian	GNU GPL	Python	Python	redes neuronales	Python scripts	N/A	N/A	Punto 3d Biológico	GPU Distribuído
Basic Prop	Otro	Multiplataforma	Java	redes	Sin experiencia en programación necesaria. Adecuado para la enseñanza.	JVM	Si	N/A	N/A

Capítulo 3
Propuesta de reconocimiento de rostros

Para el desarrollo de este proyecto, no se hará uso de redes neuronales artificales, en cambio, lo que se propone, es el uso de JJIL[xxviii] (Jon's Java Imaging Library), el cual es una librería open source bajo la licencia LGPL, este paquete esta enfocado principalmente a plataformas Java ME/Android, pero ha sido modificado para el uso en Java SE.

JJIL es una librería de procesamiento de imágenes, la cual incluye una arquitectura de procesamiento y cuenta con más de 60 rutinas sobre tareas enfocadas a procesamiento de imágenes. Junto con esta librería se encuentra también el uso de Cascada Haar, el cual ha sido generado por OpenCV[xxix] (Open Source Computer Vision); las Cascadas Haar son imágenes digitales para la detección de características usados en reconocimiento de objetos. Tienen este nombre debido a la intuitiva similaridad con los Wavelets Haar y que fueron usados en el primer detector facial en tiempo real. Una Cascada Haar codifica una serie de operaciones de detección de características en una estructura de árbol.

Estado del arte

La detección automática de la cara es un problema complejo en el procesamiento de imágenes. Existen muchos métodos para resolver este problema, como la comparación de plantillas, la discriminante lineal de Fisher, Redes Neuronales, Máquinas de vectores de soporte (SVM). Se ha logrado éxito con cada método en diversos grados y complejidades.

La misión que nos fue asignada a desarrollar es de un algoritmo capaz de localizar cada cara de una imagen en color. Nos dieron siete imágenes de entrenamiento correspondientes para desarrollar y capacitar a nuestros algoritmos. El resultado final fue de un algoritmo capaz de encontrar el 95% de las caras excepto una en una imagen en aproximadamente 30 segundos.

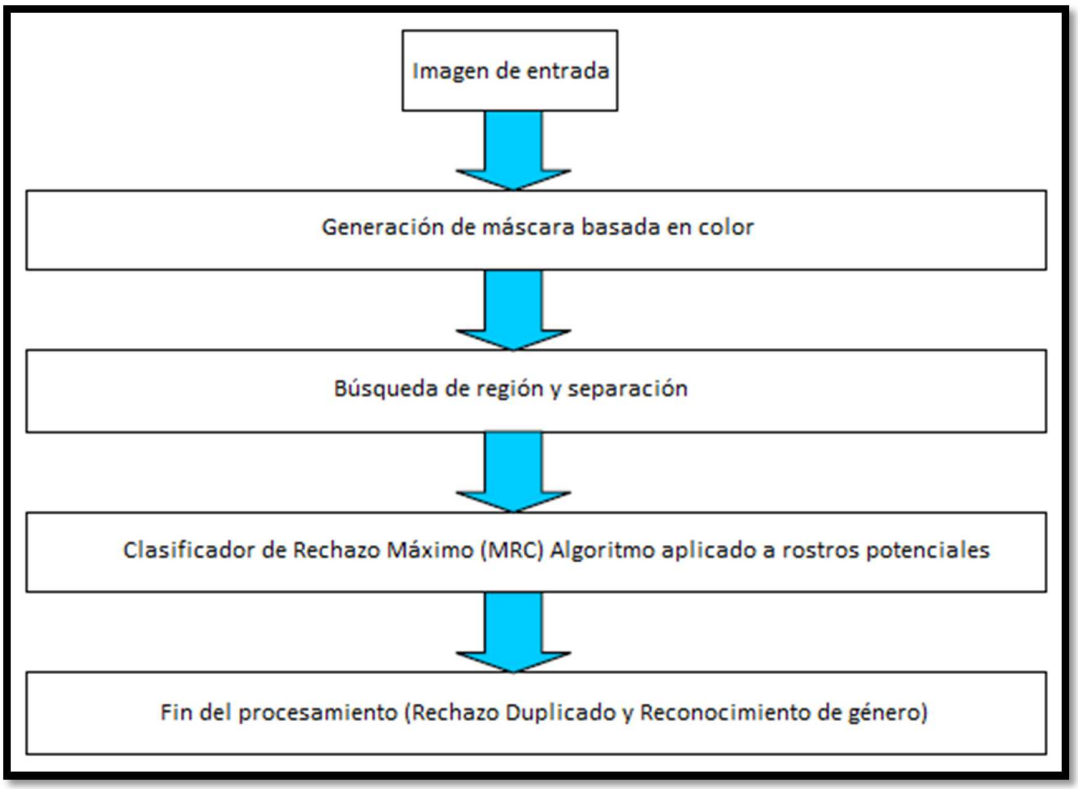


Ilustración 3.1 Diagrama de bloque para el algoritmo de detección de rostros

Basado de Color – generación de máscara

Se enfoca en reducir el número de lugares en la imagen donde se tiene que ubicar las caras. El basado de color – generación de máscara selecciona los pixeles que son lo más acertados a ser caras, los cuales pueden ser buscados usando técnicas más avanzadas.

El primer paso en el algoritmo es asignar la probabilidad de ser una cara a cada pixel de la imagen. Se empieza usando el conjunto de entrenamiento (training set) para determinar la distribución en espacio-RGB de los pixeles de caras y los pixeles que están en segundo plano (background). La Ilustración 3.2 muestra los límites de las regiones de los pixeles de caras y los pixeles que están en segundo plano basado en imágenes entrenadas.

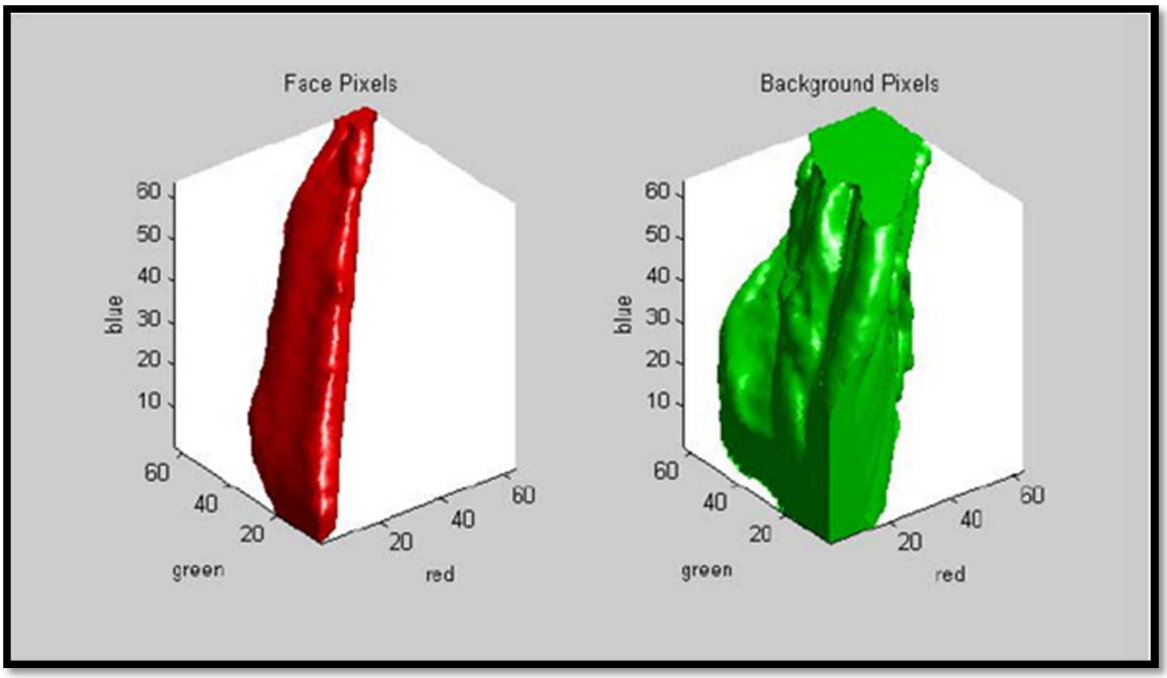


Ilustración 3.2 Espacio 3D abarcado por un Rostro y un No-Rostro de vectores de color RGB

La opción más simple para la detección de caras sería encontrar solo esos pixeles los cuales están contenidos en la región de bordes para los pixeles de cara. Sin embargo, existe un solapamiento notable entre la región de la píxeles-cara y la región de píxeles de fondo. Si tomamos cortes transversales de la trama 3d mostrado arriba, obtenemos la Ilustración 3.3, que muestra la distribución de los píxeles de la cara (rojo), los píxeles de fondo (verde), y donde se superponen (amarillo). Sería ideal asignar una alta probabilidad de que sea un píxel-cara a los píxeles que se encuentran en la región roja, una probabilidad media a los que residen en la región amarilla, y una baja probabilidad a aquellos que se encuentran en las regiones verde / negro.

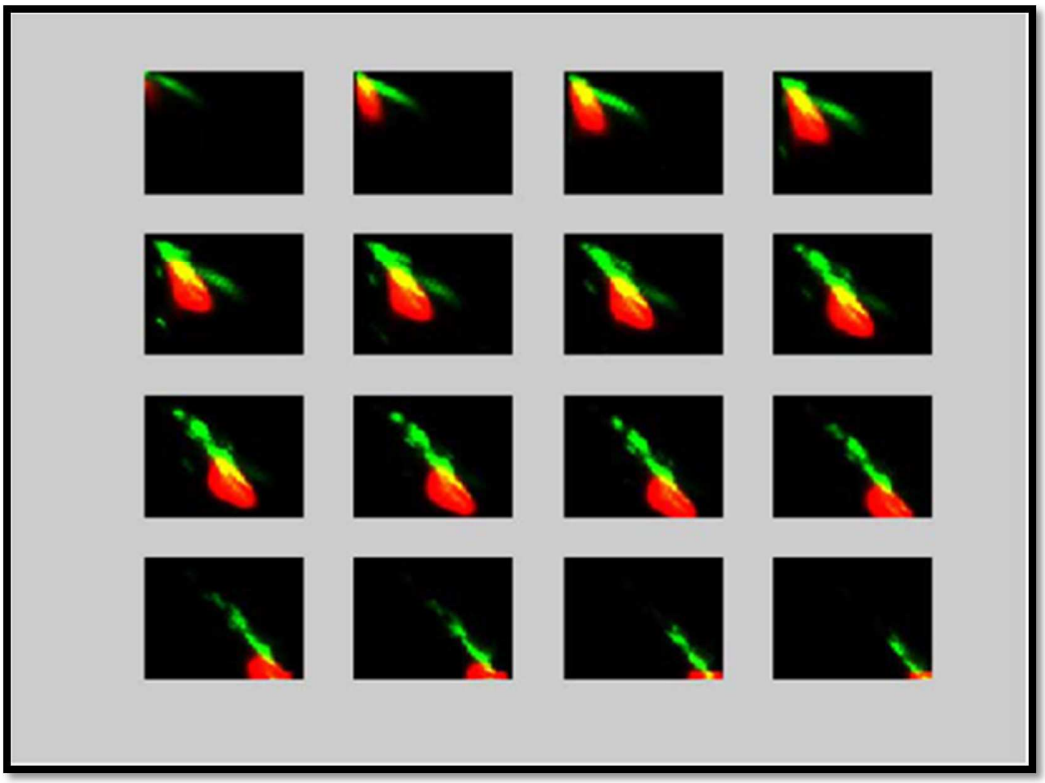


Ilustración 3.3 Cortes transversos a través de un Espacio 3D en Color RGB mostrado en figura 3.2

Para cada ubicación en el espacio RGB, utilizamos la siguiente fórmula para calcular la probabilidad asociada de ser una cara:

$$Probabilidad = (\#\ de\ píxelesCara) / (\#\ de\ píxelesCara + (número\ de\ píxeles\ de\ fondo)\ ^\wedge\ peso)$$

El peso se usa para compensar el hecho de que hay muchos más píxeles de fondo que la píxeles-cara en las imágenes de entrenamiento. También proporciona una manera simple de sesgo hacia la búsqueda de la función o el rechazo de los píxeles del fondo. Por ejemplo, un bajo peso asignará una mayor probabilidad de hacer frente a píxeles también en la región del fondo, pero también hará que los píxeles en el fondo tienen una probabilidad más alta. Se encontraron valores de peso de 0,6 a 0,8 que funciona bastante bien.

La ilustración 3.4 muestra cortes transversales a través de la función de probabilidad 3D resultante. Para cada valor RGB, podemos asignar una probabilidad entre 0 y 1. Este

trabajo es entonces filtrado por un cuadro kernel 3D con el fin de reducir la sensibilidad a las condiciones de iluminación específicas. Afortunadamente, este trabajo se puede calcular una vez y después acaba de cargar desde un archivo.

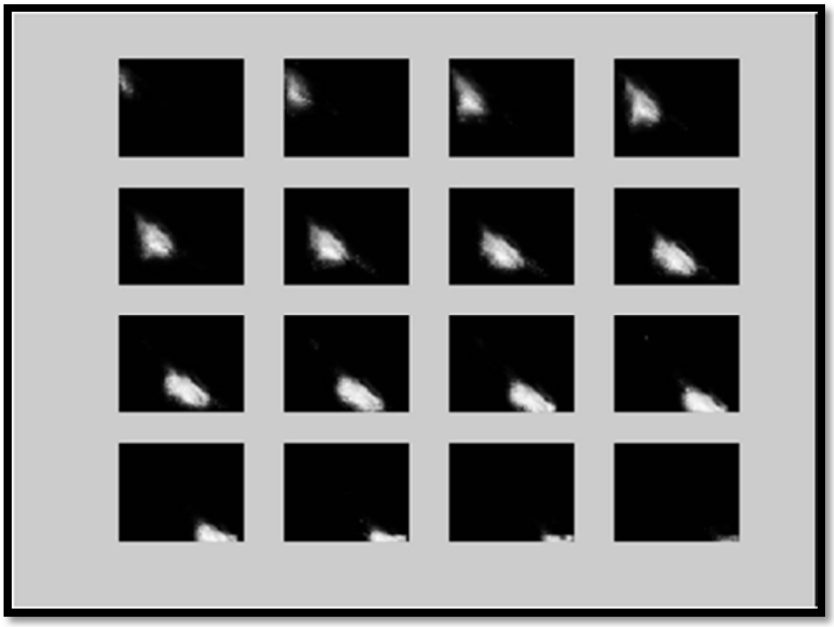


Ilustración 3.4 Cortes transversos a través de una función de probabilidad 3D dado una imagen en especifico

Usando esta función de probabilidad, vemos el valor de probabilidad para cada pixel dado la imagen. La Ilustración 3.6 muestra los resultados de esta operación.

Ilustración 3.5 Imagen mostrando la probabilidad que dado un pixel, es un Pixel-Rostro. El color blanco significa una probabilidad más alta.

Claramente, las caras son asignadas probabilidades altas, mientras que el fondo es asignado probabilidades bajas. Ahora podemos aprovechar la alta coherencia espacial de las caras, mediante la convolución con óvalos aproximadamente el mismo tamaño de una cara. Convolución con un óvalo seguido por los resultados de umbral en la máscara mostrada en la Ilustración 3.6.[xxx]

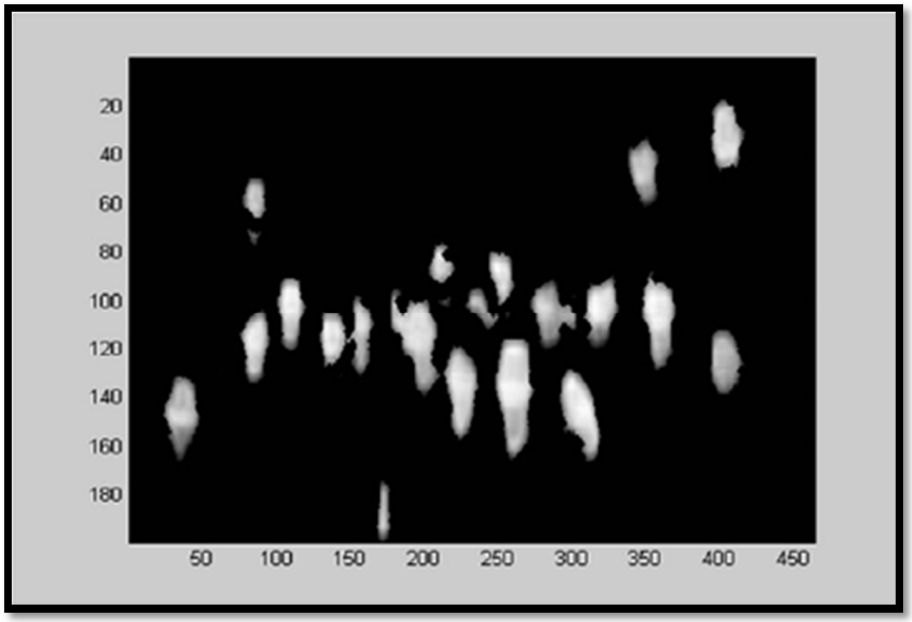


Ilustración 3.6 Máscara que resulta la probabilidad de filtrado en la imagen 3.5 con un ovalo. El resultado es entonces limitado para conseguir la máscara de arriba

Un Sistema de aplicación de reconocimiento de rostros usando procesamiento de imágenes y redes neuronales

Introducción

A menudo es útil tener una máquina que realice el reconocimiento de patrones. En particular, las máquinas que pueden leer las imágenes faciales son muy rentables. Una máquina que lee pasaportes pueden procesar muchas más pasaportes que un ser humano en el mismo tiempo. Este tipo de aplicación permite ahorrar tiempo y dinero, y elimina el requisito de que realice un ser humano semejante tarea repetitiva. Se muestra cómo el reconocimiento de rostros se puede hacer con una red neuronal artificial a través de la propagación hacia atrás. Sistema de reconocimiento (FRS) se puede subdividir en dos partes principales. La primera parte es el procesamiento de imágenes y la segunda parte son técnicas de reconocimiento. La parte de procesamiento de imágenes consiste en la adquisición del rostro en la imagen a través de la exploración, el realce de imagen, recorte de imagen, filtrado, detección de bordes y extracción de características. La segunda parte consiste en Inteligencia Artificial, el cual es compuesto por algoritmos genéticos y existen muchos enfoques para el reconocimiento de rostros.

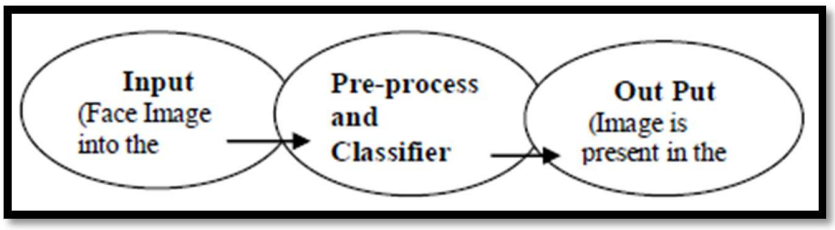


Ilustración 3.7 Representación genérica de un sistema de reconocimiento facial

Enfoque geométrico para el reconocimiento facial

La primera forma histórica para reconocer a las personas se basa en la geometría de la cara. Hay un montón de características geométricas basadas en los puntos. Nosotros experimentalmente seleccionamos 37 características geométricas de puntos que pueden ser generados por segmentos, perímetros y áreas de algunas figuras formadas por los puntos.

Para comparar los resultados del reconocimiento se estudió el conjunto de características. Incluye 15 segmentos entre los puntos y los valores medios de 15 pares de segmentos simétricos. Se prueba diferentes subconjuntos para buscar las características más importantes. Se tiene 70 imágenes de 12 personas. Imágenes de dos personas fueron añadidas a partir de la base de datos de imágenes. Se hicieron con una diferencia de tiempo enorme (de 1 a 30 años). Se selecciona 28 largometrajes. A pesar de las pequeñas variancias de rotaciones, orientación y de iluminación, el algoritmo funciona. Cada imagen se puso a prueba como una consulta y se comparó con los demás. Sólo en un caso de 70 pruebas, allí no resultó ninguna imagen de la persona en la consulta a través de las 5 más cercanas, es decir, la tasa de reconocimiento fue de 98,5%. El enfoque es robusto, pero principal problema es la ubicación del punto automático. Algunos problemas surgen si la imagen es de mala calidad o varios puntos están cubiertos por el cabello.

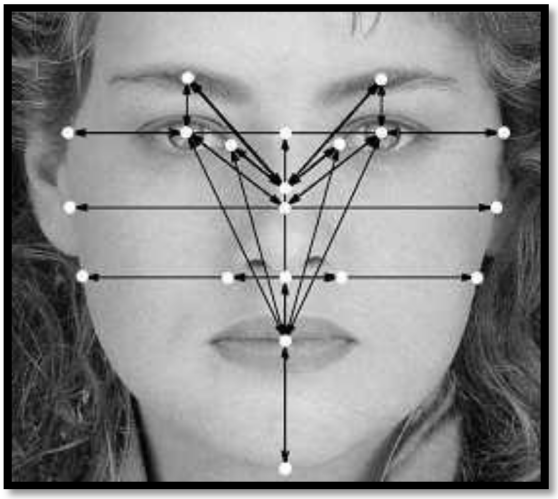


Ilustración 3.8 Algunos puntos faciales y distancias entre ellos que son usados en el reconocimiento facial

Control de acceso

Reconocimiento de Rostro es una tecnología generalizada utilizado para control de acceso. La tarea se indica cómo sigue. Hay un grupo de personas autorizadas, que un sistema de reconocimiento debe aceptar. Todas las otras personas son los "extranjeros" no autorizados y deben ser rechazados. Podemos entrenar un sistema para reconocer el pequeño

grupo de personas es por eso que se estudió la aplicación de un perceptrón multicapa (MLP) de Redes Neuronales (RN) para esta tarea. La configuración del MLP fue elegido por los experimentos hechos anteriormente. Contiene tres capas. La entrada para la red neuronal es una imagen de escala de grises. Número de unidades de entrada es igual al número de píxeles en la imagen. Número de unidades ocultas tenía 30 números de unidades de salida, es igual al número de personas a ser reconocidos. Cada unidad de salida está asociado con una persona. La red neuronal está capacitado para responder "1" en la unidad de salida, correspondiente a persona reconocida y "-1" en otras salidas. Llamamos a esta salida perfecta. Después de la formación de salida más alta de la RN indica la imagen de prueba de una persona reconocida. La mayoría de estos experimentos se aprobaron en la base de datos. Cualquier imagen de entrada se normalizó previamente por ángulo, tamaño y posición. También se estudia otras representaciones de imágenes: un conjunto de coeficientes de transformada de coseno discreta y un mapa de degradado. Un mapa de degradado permite lograr invariancia parcial a condiciones de iluminación. En los experimentos con las RN se estudia varios temas. Se exploró las reglas de umbral que permite aceptar o rechazar las decisiones de la RN. Se introdujo una regla de umbral, que permite mejorar el rendimiento del reconocimiento, considerando todas las salidas de la RN. Se llama a esta regla "SQR". Se calcula la distancia euclidiana entre el producto “real” y “perfecto” para la persona reconocida. Cuando esta distancia es mayor que el umbral, se rechaza a esta persona. De lo contrario se acepta la persona. El mejor umbral se elige experimentalmente.

Retos en el reconocimiento de rostros

1. Verificación de rostro: es un proceso de empate uno a uno que compara una imagen contra una plantilla en donde la identidad está siendo verificada. Para comprobar el desempeño de la verificación, la tasa de verificación (la tasa a la que los usuarios legítimos tienen acceso) versus la tasa de aceptaciones falsas (la tasa a la que los impostores se les conceden acceso) es dibujada, llamada curva ROC. Un buen sistema de verificación debe balancear estas dos tasas basadas en necesidades operacionales.

2. Identificación de rostros: es un proceso de empate uno a muchos que compara imágenes contra todas las plantillas de imágenes de rostros en la base de datos para determinar la identificación del rostro. La identificación de la imagen de prueba es hecha ubicando la imagen en la base de datos que tiene la similaridad más alta.[xxxi]

Posibles soluciones

Eigenfaces

Los eigenfaces se refieren a un enfoque basado en apariencia al reconocimiento de rostros en donde busca capturar la variación en una colección de imágenes y utilizar esa información para codificar y comparar imágenes de rostros individuales de una forma holística.

Específicamente, los eigenfaces son los componentes principales de una distribución de rostros, o equivalentemente, los eigenvectores de la matriz de covarianza del conjunto de imágenes de rostros, donde una imagen con N pixeles se considera un punto (o vector) en el espacio N-dimensional. La idea de utilizar componentes principales para representar rostros humanos fue desarrollada por Sirovich y Kirby (Kirby y Sirovich 1987) y utilizada por Turk y Pentland (Turk y Pentland 1991) para la detección y el reconocimiento de caras. El enfoque Eigenfaces es considerado por muchos como la tecnología de reconocimiento facial, y que sirvió como base para uno de los mejores productos comerciales. Desde su desarrollo inicial y su publicación, se han realizado varias extensiones del método original y muchos nuevos avances en los sistemas de reconocimiento facial automático. Eigenfaces está todavía considerado a menudo como un método de comparación de línea de base para demostrar el rendimiento mínimo esperado de un sistema de este tipo.

Antes de generar los eigenfaces, las imágenes de rostros son normalizadas para alinear los ojos y bocas, luego todo es retomado con la misma resolución de pixeles. Luego son extraídos los Eigenfaces de los datos de la imagen por medio de PCA (Análisis principal de componente) de la siguiente manera:

1. Dado imágenes de rostros M con un tamaño de h x w, cada imagen es transformada en un vector de tamaño $D(hw)$ y colocado en un conjunto :

$$\{\Gamma_1, \Gamma_2, \cdots, \Gamma_M\}$$

Las imágenes de rostros deben ser escaladas y alineadas apropiadamente, y los fondos (áreas que no pertenezcan al rostro como el cabello y el cuello).

2. Cada rostro se diferencia de la media por vector $\Phi_i = \Gamma_i - \Psi$, donde el promedio del rostro es definido por $\Psi = \frac{1}{M}\sum_{i=1}^{M} \Gamma_i$.

3. La covarianza de la matriz $\boldsymbol{C} \in \mathbb{R}^{DxD}$ está definida como:

$$C = \frac{1}{M}\sum_{i=1}^{M} \Phi_i \Phi_i^{\intercal} = \boldsymbol{A}\boldsymbol{A}^{\intercal}$$

Donde $\boldsymbol{A} = \{\Phi_1, \Phi_2, \ldots, \Phi_M\} \in \mathbb{R}^{DxM}$

4. La determinación de los eigenvectores de $\boldsymbol{C}$ es una tarea insuperable para tamaños de imagen típicas cuando $D \gg M$. Sin embargo, para calcular eficientemente los eigenvectores de $\boldsymbol{C}$, uno podría primeramente calcular los eigenvectores la matriz $\boldsymbol{A}\boldsymbol{A}^{\intercal}$ mucho más pequeña M x M. Los eigenvectores y los eigenvalores de las matrices de $\boldsymbol{A}\boldsymbol{A}^{\intercal}$ están definidas como:

$$\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots \mathbf{v}_r\}$$

Y $\Lambda = diag\{\lambda 1, \lambda 2, \cdots \lambda r\}$, $\lambda 1 \geq \lambda 2 \geq \cdots \lambda r > 0$, donde r es el rango o categoría de $\boldsymbol{A}$.

Los eigenvalores y eigenvectores de la matriz $\boldsymbol{C}$ son Λ y $\boldsymbol{U} = \boldsymbol{A}\boldsymbol{V}\Lambda^{-1/2}$, donde $\boldsymbol{U} = \{\boldsymbol{u_i}\}$ es la colección de los eigenfaces.

Ilustración 3.9 Ejemplos de imágenes de rostros CMU PIE

Ilustración 3.10 El extremo izquierdo en la primera fila es el promedio de rostros, los otros son dos mejores eigenfaces; la segunda fila muestra eigenfaces con lo menos de tres eigenvalores.

Usando Eigenfaces en procesamiento facial

Los eigenfaces abarcan un sub-espacio m-dimensional del espacio de la imagen original seleccionando el subconjunto de vectores propios $\widehat{\boldsymbol{U}} = \{u_1, \dots, u_m\}$ asociado con el eigenvalor m más largo. Esto resulta en el espacio-rostro, en donde su origen es el promedio facial, y sus ejes son los eigenfaces. Para llevar a cabo la detección o el reconocimiento de rostros, se puede calcular la distancia dentro o desde el espacio de la cara.

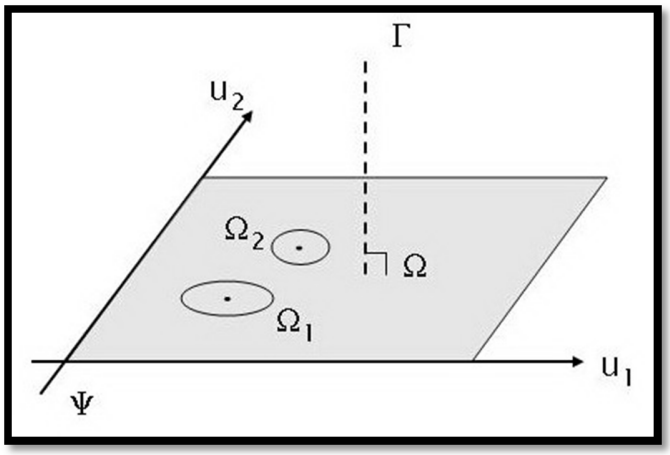


Ilustración 3.11 Visualización de un espacio facial 2D con los ejes representando dos Eigenfaces

Detección del rostro

Debido a que el espacio del rostro (el sub-espacio abarcado por los eigenfaces) define el espacio de imágenes de la cara, la detección de la cara puede ser considerada como detección de manchas de imagen que se encuentran cerca de la cara-espacio. En otras palabras, la distancia de proyección δ debe estar dentro de cierta θ_δ. La distancia δ punto-espacio es la distancia entre la imagen de la cara y su proyección sobre el espacio de la cara y se la puede calcular como:

$$\delta = \|(\mathbf{I} - \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^{\top})(\Gamma - \Psi)\|$$

Donde $\mathbf{I}$ es la matriz identidad. La distancia entre una imagen y su proyección espacio facial, es mucho más pequeño para un rostro que para una imagen que no sea un rostro.

Ilustración 3.12 Imágenes originales (fila 1) y sus proyecciones en el espacio facial (fila 2)

Detección facial

Un nuevo rostro Γ es proyectado en el espacio facial por $\Omega = \widehat{\boldsymbol{U}}(\Gamma - \Psi)$. Donde $\widehat{\boldsymbol{U}}$ es el conjunto más característico de los eigenvectores. Notar que el peso del vector Ω es la representación de un nuevo rostro en el espacio facial. Una manera más simple de determinar cuál clase de rostro Γ pertenece, es minimizando la distancia Euclidiana

$$\epsilon_k = \|\Omega - \Omega_k\|$$

Donde Ω_k es el peso del vector representando la k-ava clase rostro. El rostro Γ se considera como perteneciente a la clase k si el mínimo ϵ_k es mucho más pequeño que algún otro umbral θ_ϵ predefinido, de lo contrario, es clasificado como desconocido. [xxxii]

Mapa bordes de líneas (LEM)

Este algoritmo describe una nueva técnica basada en líneas borde de mapa (LEM) para lograr el reconocimiento de rostros. Adicionalmente, propone una técnica de búsqueda de línea para hacer posible esta tarea. En oposición con otros algoritmos, LEM utiliza características fisiológicas de rostros humanos para resolver el problema; que utiliza principalmente la boca, la nariz y los ojos como los más característicos. Para medir la

similitud de caras humanas Las imágenes de caras se convierten en primer lugar en imágenes de nivel de gris. Las imágenes se codifican en mapas binarios de bordes a través del uso del algoritmo de detección de bordes. Este sistema es muy similar a la forma en que los seres humanos perciben otras caras de personas. La principal ventaja de LEM es la baja sensibilidad a los cambios de iluminación, porque es una representación de una imagen de nivel intermedio derivado de una representación de mapa de nivel bajo. El algoritmo también tiene otra importante mejora, y es el uso de memoria el cual es baja debido al tipo de datos usados. En la siguiente ilustración se aprecia un mapa de bordes de líneas faciales, se puede notar que mantiene características de la cara pero en un nivel muy simplificado.

Ilustración 3.13 Ejemplo de un rostro LEM

Una de las partes más importantes del algoritmo, es la distancia del segmento de línea Hausdorff (LHD) descrito para lograr una búsqueda precisa en imágenes de rostros. Este método no es orientado para un cálculo exacto. Su principal característica es la flexibilidad del tamaño, posición y orientación.

Dado dos LEM $\boldsymbol{M}^l = \{m_1^l, m_2^l, \dots m_p^l\}$ (rostros desde la base de datos) y $\boldsymbol{T}^l = \{t_1^l, t_2^l, \dots t_q^l\}$ (imagen de entrada a ser detectada) el LHD es representado por el vector $\bar{d}(m_i^l, t_j^l)$. Los elementos de este vector representan así mismos tres medidas de distancias diferentes: distancia orientación, distancia paralela, distancia perpendicular respectivamente.

$$\bar{d}(m_i^l, t_j^l) = \begin{bmatrix} d_\theta(m_i^l, t_j^l) \\ d_\parallel(m_i^l, t_j^l) \\ d_\perp(m_i^l, t_j^l) \end{bmatrix}$$

$$d_\theta(m_i^l, t_j^l) = f(\theta(m_i^l, t_j^l)) \qquad d_\parallel(m_i^l, t_j^l) = \min(l_{\parallel 1}, l_{\parallel 2}) \qquad d_\perp(m_i^l, t_j^l) = l_\perp$$

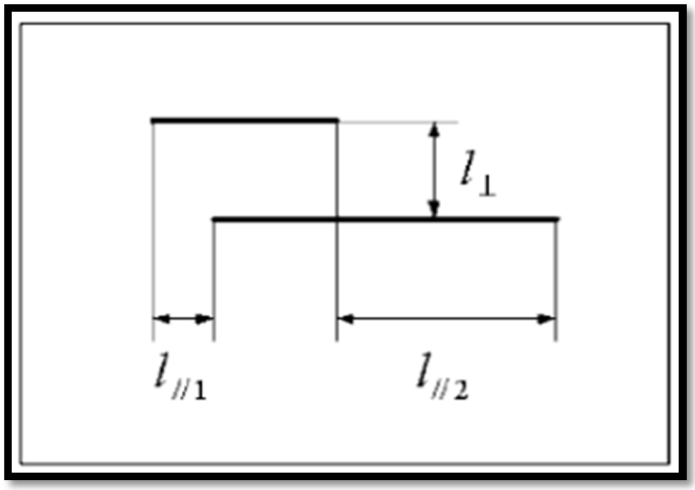


Ilustración 3.14 Ejemplo práctico para calcular distancias paralelas y perpendiculares

La función $\theta(m_i^l, t_j^l)$ representa el ángulo de la intersección más pequeña entre las líneas m_i^l y t_j^l. La función f es una función no lineal de penalización que ignora pequeños ángulos y penaliza a los grandes. Puede ser usado como $f(x) = {x^2}/{W}$ como una función de penalización. Finalmente la distancia entre los dos segmentos puede ser calculada con la siguiente ecuación:

$$d(m_i^l, t_j^l) = \sqrt{d_\theta^2(m_i^l, t_j^l) + d_\parallel^2(m_i^l, t_j^l) + d_\perp^2(m_i^l, t_j^l)}$$

Después de haber definido la distancia entre las dos líneas, la distancia del segmento línea Hausdorff (pLHD) es definida como:

$$h(M^l, T^l) = \max(h(M^l, T^l), h(T^l, M^l))$$

Donde

$$h(M^l, T^l) = \frac{1}{\sum_{m_i^l \in M^l} l_{m_i^l}} \sum_{m_i^l \in M^l} l_{m_i^l} \min_{t_k^l \in T^l} d(m_i^l, t_j^l)$$

Y $l_{m_i^l}$ es la longitud del segmento m_i^l.

La principal fortaleza de esta medición de distancia es que la medición de la distancia paralela, elegimos la distancia mínima entre los bordes. Ayuda cuando la línea borde es fuertemente detectada y la otra no. Evita el cambio de puntos característicos. Sin embargo, también tiene una debilidad; brevemente, puede confundir a las líneas y no detectar similitudes que se deben detectar. Se puede añadir un nuevo parámetro para la distancia de Hausdorff, comparando el número de líneas en las imágenes, es una buena forma para excluir imágenes.[xxxiii]

Propuesta a implementar

El algoritmo funciona con imágenes de entrada de cualquier tipo (formato, tamaño, colores) para luego transformarlas a imágenes de 8 niveles de grises para su posterior uso, junto con la implementación de la cascada Haar dentro del paquete JJIL, permite determinar las áreas en la imagen en donde fija si existen rostros faciales o no. La cascada Haar consiste en un conjunto de operaciones de detección de características estructurado en forma de árbol, en donde en cada punto de decisión en la imagen falla o aprueba la detección. La palabra cascada da a entender que en cada etapa del proceso de detección es aplicado subsecuentemente a una región de interés en la imagen hasta que en algún punto la zona candidata se aprueba o se la rechaza. De la forma en la que está implementado en JJIL, en cada paso, aplica un conjunto de operaciones de detección a la imagen y luego hace una sumatoria de los resultados en donde verifica si esta suma es mayor que el límite, para luego continuar con los siguientes pasos del algoritmo, caso contrario la imagen es desaprobada, que sería en donde no logre detectar ningún rostro. La forma en la que se realiza el proceso de detección es a través del escalamiento o ajuste de la imagen en cada etapa junto con otra imagen que viene a ser una máscara; en un inicio, esta máscara tiene una resolución de 1x1 pixeles para luego escalarla al tamaño de la imagen transformada al principio del algoritmo, esta máscara es completamente de color negra para que posteriormente se coloree de blanco

la zona en donde se detecte un rostro, es por esta razón del escalamiento que se puede llegar a detectar rostros de cualquier tamaño y/o posicionamiento en la imagen empezando por la escala más gruesa para luego terminar con la más pequeña. Para cada escala, si un rostro no ha sido detectado previamente en la posición actual, se aplica el clasificador Haar a la imagen reducida en esta posición, si un rostro es encontrado, se marca la imagen (máscara) para continuar con las siguientes escalas, en el nivel más bajo, la cascada Haar consiste en detectores de características rectangulares los cuales son sumadas, multiplicadas por un coeficiente y luego añadidas conjuntamente. Estos detectores de características siempre tienen dos o tres regiones en donde se da las coordenadas de la parte superior izquierda del rectángulo, ancho, alto y el coeficiente respectivamente, recorriendo por completo a la imagen de entrada obteniendo como resultado final una imagen de salida que posee la localización de el o los rostros detectados.

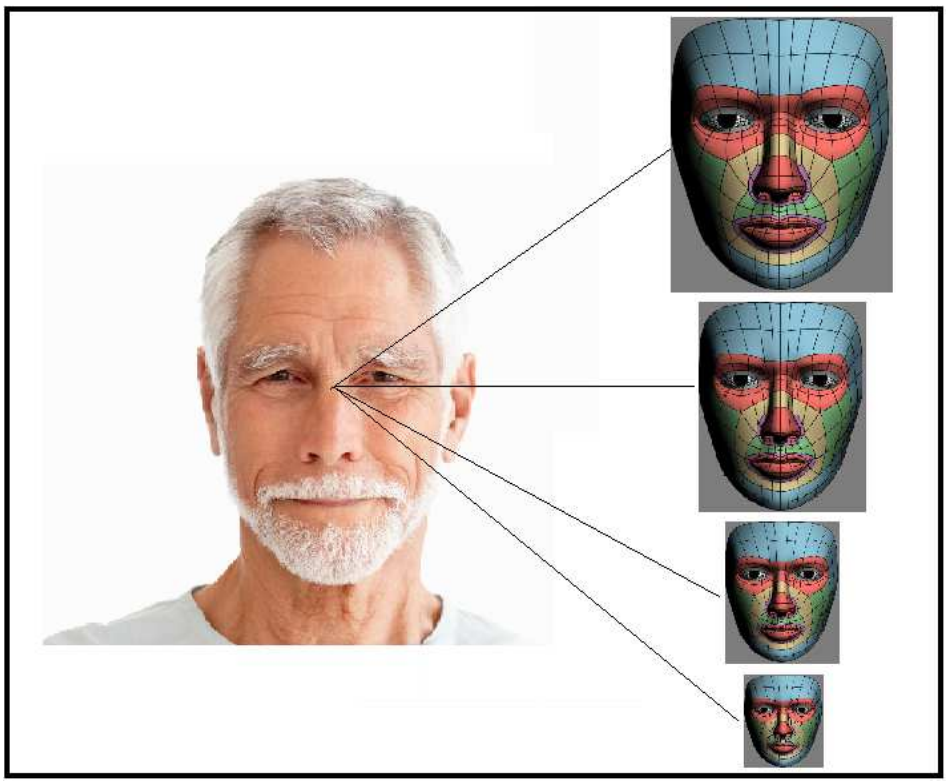


Ilustración 3.15 Visualización general de cómo trabaja la detección de rostros sobre imágenes.

Diagrama de flujo

A través de un diagrama de flujo, se muestra cómo trabaja el algoritmo para hacer posible la detección de rostros en imágenes.

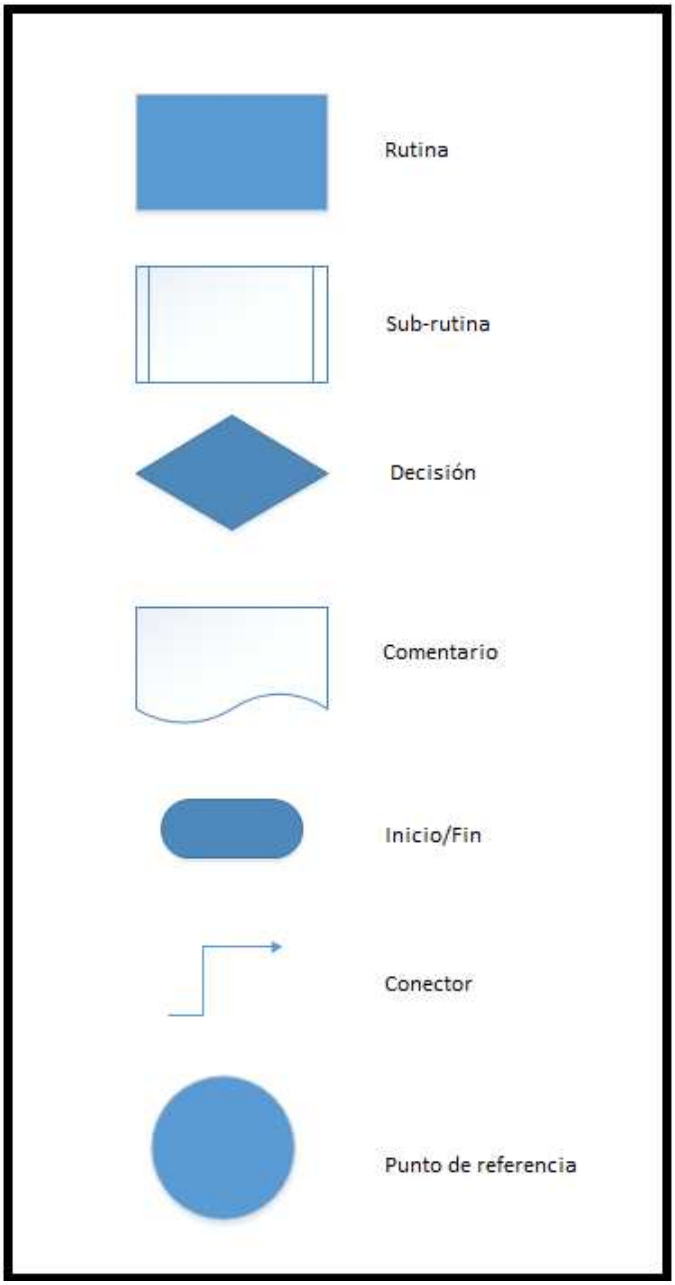
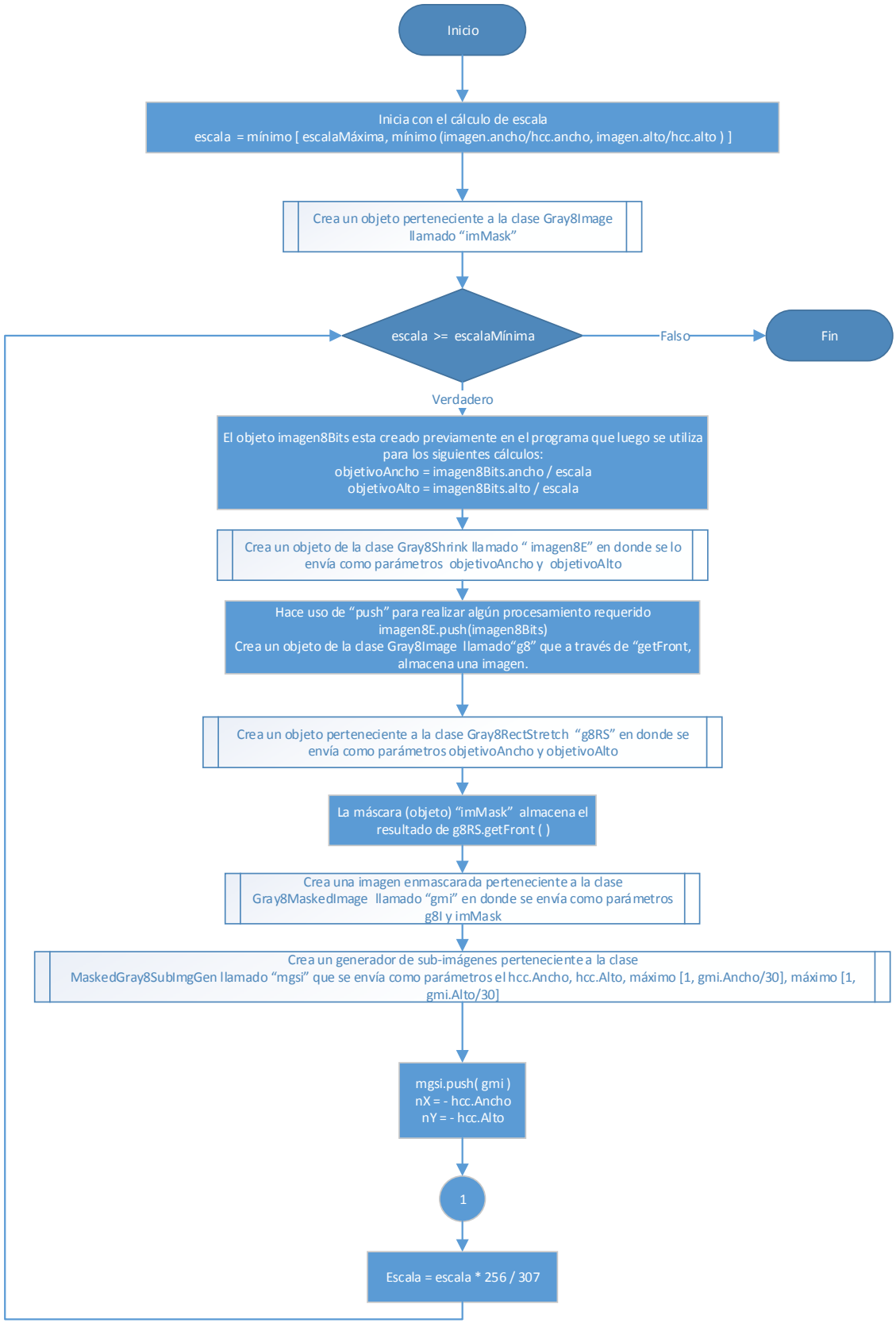
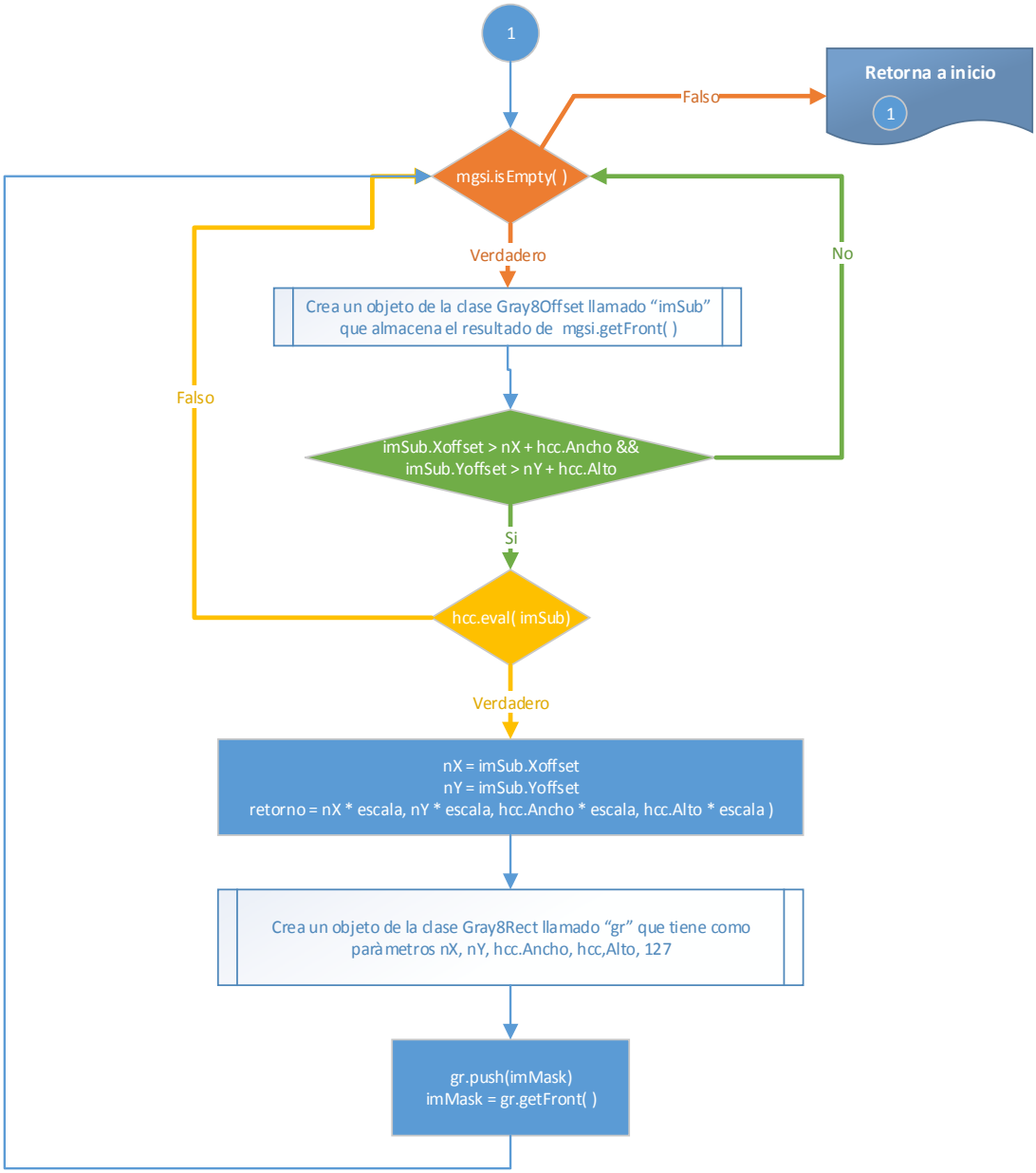


Ilustración 3.16 Componentes utilizados en el diagrama de flujo.


Inicio
Inicia con el cálculo de escala
escala = mínimo [escalaMáxima, mínimo (imagen.ancho/hcc.ancho, imagen.alto/hcc.alto)]
Crea un objeto perteneciente a la clase Gray8Image llamado "imMask"
escala >= escalaMínima
Falso
Fin
Verdadero
El objeto imagen8Bits esta creado previamente en el programa que luego se utiliza para los siguientes cálculos:
objetivoAncho = imagen8Bits.ancho / escala
objetivoAlto = imagen8Bits.alto / escala
Crea un objeto de la clase Gray8Shrink llamado " imagen8E" en donde se lo envía como parámetros objetivoAncho y objetivoAlto
Hace uso de "push" para realizar algún procesamiento requerido
imagen8E.push(imagen8Bits)
Crea un objeto de la clase Gray8Image llamado"g8" que a través de "getFront, almacena una imagen.
Crea un objeto perteneciente a la clase Gray8RectStretch "g8RS" en donde se envía como parámetros objetivoAncho y objetivoAlto
La máscara (objeto) "imMask" almacena el resultado de g8RS.getFront ( )
Crea una imagen enmascarada perteneciente a la clase Gray8MaskedImage llamado "gmi" en donde se envía como parámetros g8I y imMask
Crea un generador de sub-imágenes perteneciente a la clase MaskedGray8SubImgGen llamado "mgsi" que se envía como parámetros el hcc.Ancho, hcc.Alto, máximo [1, gmi.Ancho/30], máximo [1, gmi.Alto/30]
mgsi.push(gmi)
nX = - hcc.Ancho
nY = - hcc.Alto
1
Escala = escala * 256 / 307


1
Retorna a inicio
1
Falso
mgsi.isEmpty( )
Verdadero
No
Crea un objeto de la clase Gray8Offset llamado “imSub”
que almacena el resultado de mgsi.getFront( )
Falso
imSub.Xoffset > nX + hcc.Ancho &&
imSub.Yoffset > nY + hcc.Alto
Si
hcc.eval(imSub)
Verdadero
nX = imSub.Xoffset
nY = imSub.Yoffset
retorno = nX * escala, nY * escala, hcc.Ancho * escala, hcc.Alto * escala)
Crea un objeto de la clase Gray8Rect llamado “gr” que tiene como
parámetros nX, nY, hcc.Ancho, hcc,Alto, 127
gr.push(imMask)
imMask = gr.getFront( )

Dentro de la librería JJIL se hace uso de las siguientes clases enlistadas a continuación, que están organizadas de la siguiente manera:

- Pipeline Stage
- Gray8DetectHaarMultiScale
- Error
- Gray8Image
- Gray8MaskedImage
- Gray8OffsetImage
- Image
- Rect

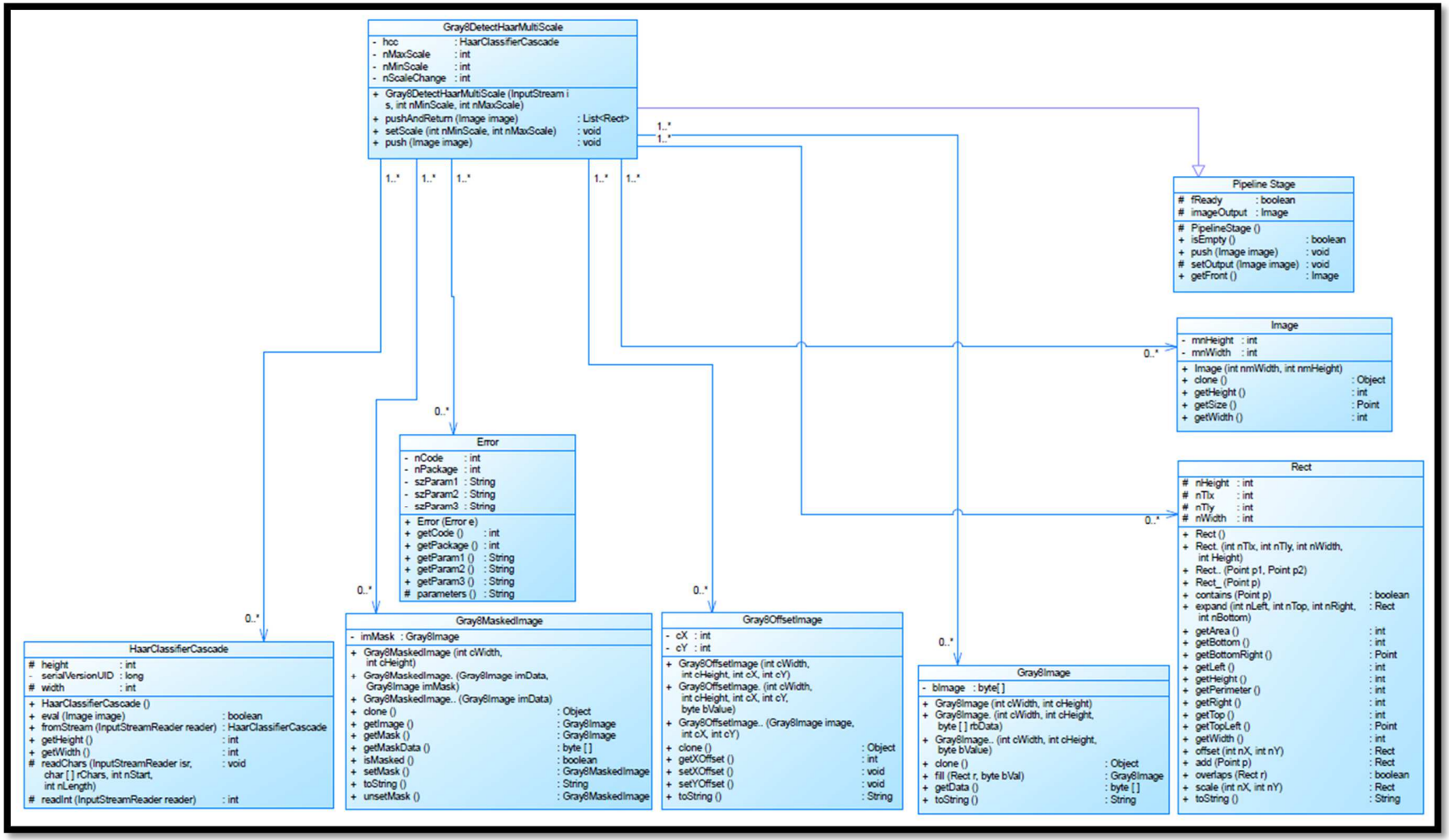


Ilustración 3.17 Diagrama de clases a detalle

Capítulo 4
Análisis de resultados

En este apartado, se presenta de forma gráfica los resultados obtenidos por la aplicación. Como salida, se obtiene una imagen de las mismas características que la imagen de entrada, en donde tiene ubicado el o los lugares de la detección de rostros. La imagen de salida, por defecto, es completamente de color negro (no detección), en el caso de que se detecte uno o más características, la imagen de salida tendrá uno o más recuadros de color blanco indicando el lugar donde se hizo la detección. Para la obtención de los resultados, se realizó pruebas con imágenes de diferentes resoluciones y tipos.

Primer grupo de imágenes

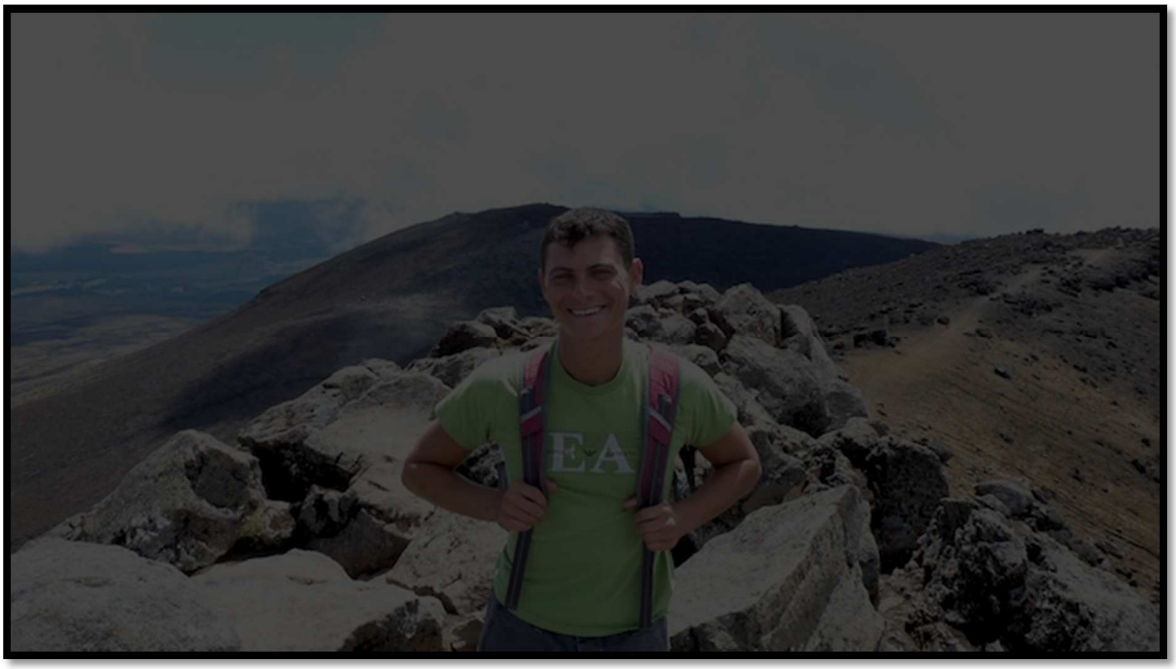
Este primer grupo de imágenes se enfoca en imágenes que contienen un solo rostro para luego observar qué resultados se obtienen a través de la ejecución de la aplicación.

Resultados

Imagen # 1 - 849 x 545px Imagen de salida

Superposición

Imagen # 2: – 675 x 375px Imagen de salida



Superposición

Imagen # 3 – 390 x 388px Imagen de salida

Superposición

Imagen # 4 – 598 x 397px Imagen de salida

Superposición

Imagen # 5 – 700 x 466px Imagen de salida

Superposición

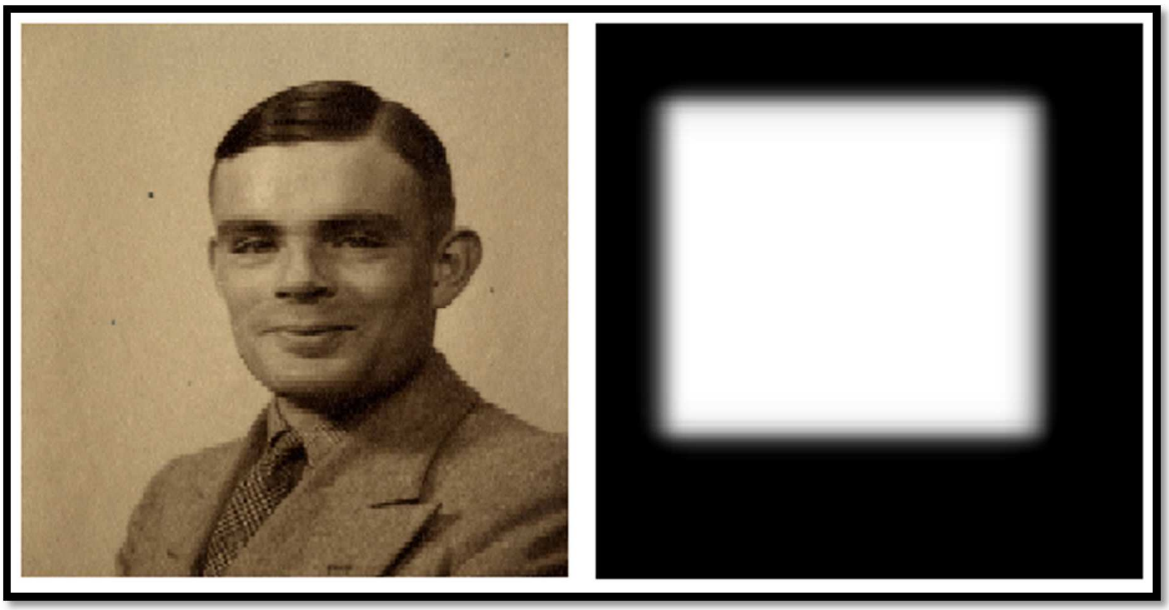


Imagen # 6 – 848 x 974px Imagen de salida

Superposición

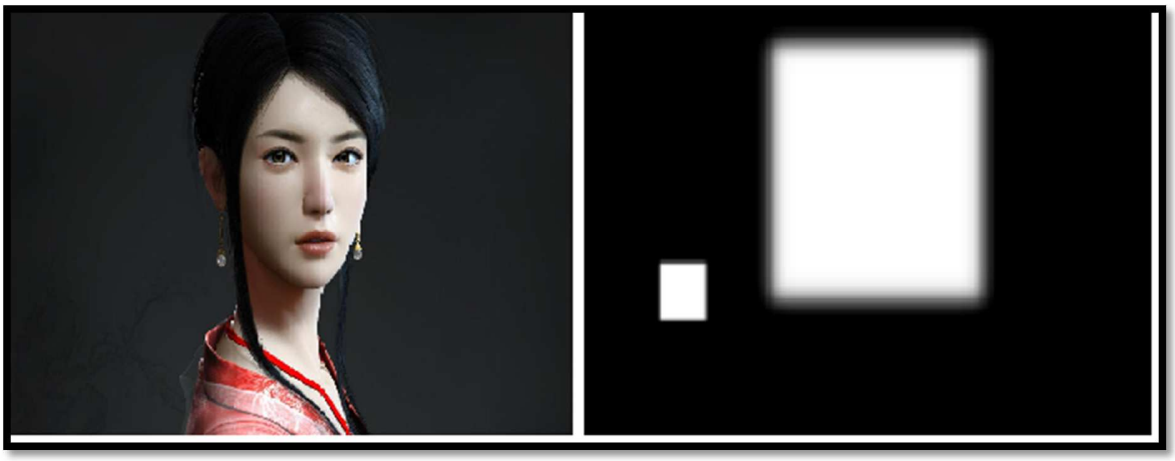
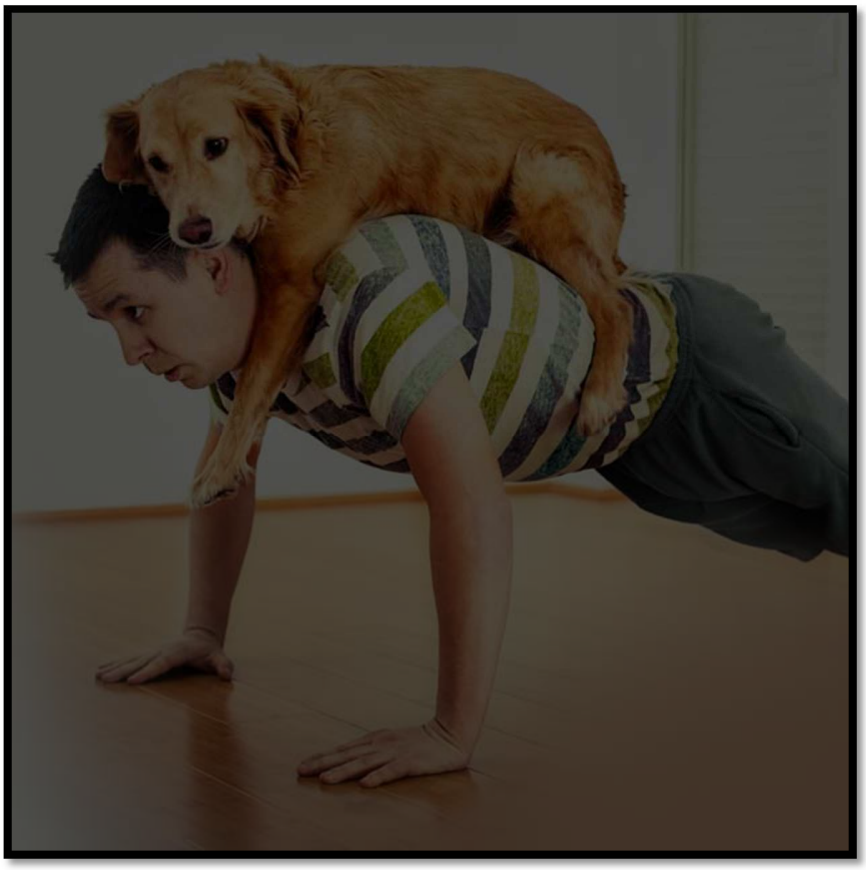


Imagen # 7 – 1920 x 1080px Imagen de salida

Superposición

En este resultado se observa que existe una detección falsa (el recuadro pequeño), esto es normal debido a cómo trabaja el algoritmo Haar que lo hace cambiando de tamaño a diferentes escalas a la imagen y ejecutando una matriz de tamaño configurado o arreglado sobre las imágenes de entrada.

Imagen # 8 – 623 x 629px Imagen de salida



Superposición

En el resultado mostrado arriba, el algoritmo, no ha logrado detectar un rostro y se debe a que en la imagen, el rostro de la persona no está posicionado frontalmente.

Imagen # 9 – 667 x 500px Imagen de salida

Superposición

Segundo grupo de imágenes

Este segundo grupo de imágenes se enfoca en imágenes que contienen varios rostros para luego observar qué resultados se obtienen a través de la ejecución de la aplicación.

Resultados

Imagen # 1 - 480 x 349px Imagen de salida

Superposición

Imagen # 2 – 545 x 254px Imagen de salida

Superposición

Imagen # 3 – 552 x 368px Imagen de salida

Superposición

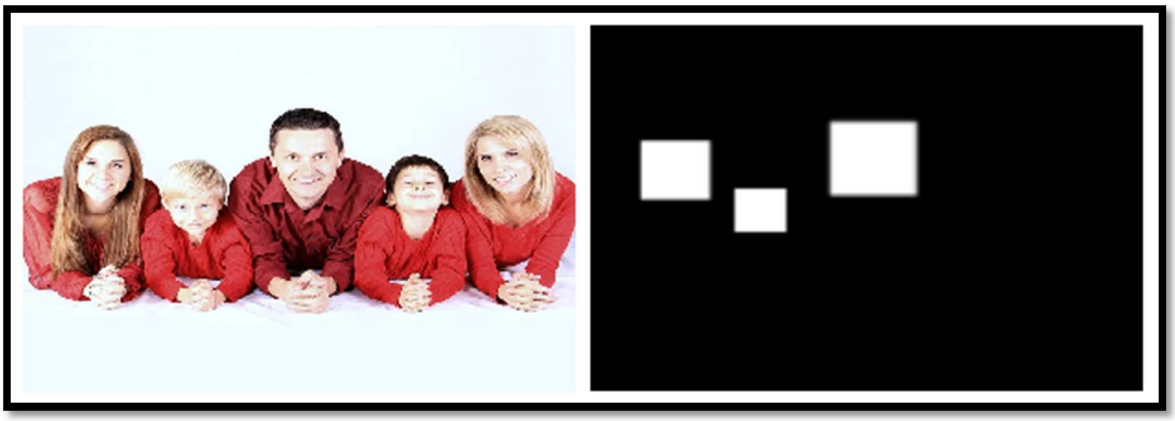


Imagen # 4 – 640 x 502px Imagen de salida

Superposición

En este resultado se muestra claramente que los tres rostros detectados están posicionados correctamente (de forma frontal), mientras que los dos últimos rostros no, en el rostro no detectado de la mujer su cabello bloquea parte de su cara.

Imagen # 5 – 500 x 333px Imagen de salida

Superposición

En el resultado de arriba, no logra detectar ningún rostro debido a que alrededor de ellos, existe una obstrucción de tal forma que el algoritmo no funciona.

Imagen # 6 – 1024 x 740px Imagen de salida

Superposición

En este resultado se observa que existe una detección falsa (el recuadro pequeño), esto es normal debido a cómo trabaja el algoritmo Haar que lo hace cambiando de tamaño a

diferentes escalas a la imagen y ejecutando una matriz de tamaño configurado o arreglado sobre las imágenes de entrada.

Imagen # 7: – 750 x 500px Imagen de salida

Superposición

Imagen # 8 - 920 x 414px Imagen de salida

Superposición

En el resultado anterior se logra observar que la gran mayoría de rostros se ven bloqueados y no se logra una detección total de la imagen

Imagen # 9 – 636 x 380px Imagen de salida

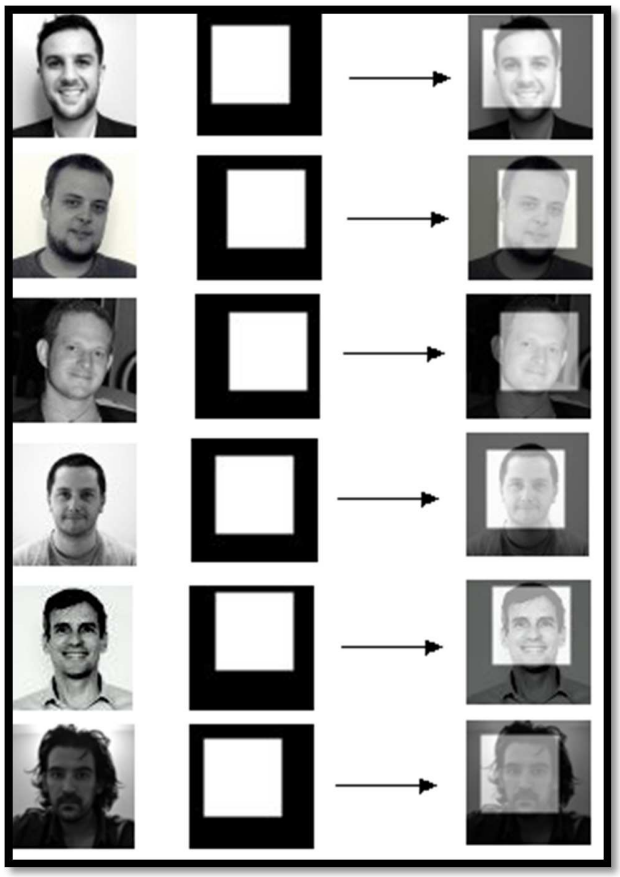
Superposición

En este resultado, el único rostro detectado está ubicado de manera frontal permitiendo así la detección, mientras que los otros rostros no lo están.

Tercer grupo de imágenes

Para este grupo, todas las imágenes tienen una resolución de 64 x 64 pixeles, B/N.

Resultados



Nota

Los resultados mostrados en el primer y segundo grupo de imágenes, se observa que la aplicación, en la mayoría de entradas, ha logrado la detección de rostros, y en otras no; además se nota que en las imágenes que contienen más de un rostro, no se ha logrado detectar todas ellas, esto puede ser debido a que algunos rostros no están posicionados de manera conveniente para la aplicación, se da los casos en que los rostros se encuentran de perfil u obstruidos por algún objeto como es el caso de un rostro. Por otra parte, en ciertas imágenes, se muestra “detecciones” de rostros en donde no existe, esto es normal debido a cómo trabaja el algoritmo Haar que lo hace cambiando de tamaño a la imagen de entrada a diferentes escalas y ejecutando una matriz de tamaño configurado o arreglado sobre las imágenes de entrada. Es posible que para un mismo rostro, pueda ser detectado en diferentes escalas y se termine obteniendo recuadros blancos dentro de otros. Es una cuestión relativamente fácil remover los recuadros “extras” mediante la verificación si están contenidos por otro recuadro anterior e ignorándolos apropiadamente. Por último, en el tercer grupo de imágenes, se observa que la aplicación, en todas las imágenes de prueba, detectó de manera satisfactoria los rostros encontrados.

El CD de instalación junto con el manual de usuario viene adjuntados con el proyecto.

Capítulo 5

Conclusiones

- En términos generales, la aplicación que ha sido planteada a desarrollar en este proyecto ha logrado la detección de rostros en imágenes de forma satisfactoria, sin embargo existen casos particulares en donde no se ha logrado buenos resultados debido a la ubicación y posicionamiento de los rostros en las imágenes ya que para esta aplicación necesita que los rostros estén ubicados totalmente de frente sin ningún tipo de obstrucción.

- La detección de rostros en imágenes en esta aplicación se da gracias al uso del detector Haar dentro del paquete JJIL por lo que ha permitido a nivel general, obtener resultados satisfactorios.

- Gracias al uso de la librería JJIL, se logró analizar y entender cómo funciona la detección de rostros en imágenes ya que JJIL es una herramienta de código abierto.

- Cuando se plantea desarrollar aplicaciones enfocadas a temas de reconocimiento, detección, procesamiento de imágenes, se requiere de conocimientos en varias disciplinas como matemáticas, desarrollo de software, complejidad de algoritmos, entre otros, ya que trata de temas complejos.

- Hay como obtener mejores resultados en la ejecución del programa si es que se logra modificar ciertas partes claves en el código fuente de la aplicación.

- La aplicación construida en esta disertación ha sido enfocada como una aplicación de escritorio, sin embargo, se la puede desarrollar para dispositivos móviles ya que las herramientas usadas consumen poco recursos de hardware.

Recomendaciones

- Dentro del análisis de resultados, las imágenes con mejores respuestas fueron aquellas que contienen un solo rostro, por lo tanto, si se quiere obtener resultados de igual manera para imágenes conteniendo varios rostros, se recomienda trabajar en la modificación del detector Haar.

- Es recomendable usar herramientas de software libre o de código abierto (dependiendo de las necesidades del tema a construir) para el desarrollo de software en temas relacionados como el de este proyecto ya que permiten enriquecer el conocimiento y de esa manera poder realizar modificaciones según sea necesario.
- Para mejores resultados, es recomendable trabajar con varias personas especializadas en diferentes disciplinas para facilitar el desarrollo del software.

- Existen varios formatos de imágenes pero para este proyecto se recomienda trabajar con las imágenes más comunes como JPG, JPEG, PNG.

Referencias bibliográficas

[i] Wikipedia. Digital image processing. http://en.wikipedia.org/wiki/Digital_image_processing. 03/02/2015
[ii] Guido E. Ochoa Moreno. Procesamiento de imágenes. Revisión 6 2013. Página 27, 28.
[iii] Guido E. Ochoa Moreno. Procesamiento de imágenes. Revisión 6 2013. Páginas 37, 38
[iv] Guido E. Ochoa Moreno. Procesamiento de imágenes. Revisión 6 2013. Páginas 38, 39
[v] Guido E. Ochoa Moreno. Procesamiento de imágenes. Revisión 6 2013. Páginas 39, 40
[vi] Guido E. Ochoa Moreno. Procesamiento de imágenes. Revisión 6 2013. Páginas 41
[vii] Guido E. Ochoa Moreno. Procesamiento de imágenes. Revisión 6 2013. Páginas 41
[viii] John C. Russ. The Image Processing Handbook. Sexta Edición.2011. Páginas 391.
[ix] Guido E. Ochoa Moreno. Procesamiento de imágenes. Revisión 6 2013. Páginas 50.
[x] Wikipedia. Image noise. http://en.wikipedia.org/wiki/Image_noise. 03/02/2015
[xi] Enrique Cabello Pardos. Técnicas de reconocimiento facial mediante redes neuronales. http://oa.upm.es/215/1/10200404.pdf. 12/02/15
[xii] Enrique Cabello Pardos. Técnicas de reconocimiento facial mediante redes neuronales. http://oa.upm.es/215/1/10200404.pdf. 12/02/15
[xiii] Marco Antonio Valencia Reyes, Cornelio Yáñez Márquez, Luis Pastor Sánchez Fernández3Algoritmo Backpropagation para redes neuronales: conceptos y aplicaciones. http://www.repositoriodigital.ipn.mx/bitstream/handle/123456789/8628/Archivo%20que%20incluye%20portada,%20%C3%ADndice%20y%20texto.pdf?sequence=1. 12/02/15.
[xiv] Marco Antonio Valencia Reyes, Cornelio Yáñez Márquez, Luis Pastor Sánchez Fernández3Algoritmo Backpropagation para redes neuronales: conceptos y aplicaciones. http://www.repositoriodigital.ipn.mx/bitstream/handle/123456789/8628/Archivo%20que%20incluye%20portada,%20%C3%ADndice%20y%20texto.pdf?sequence=1. 12/02/15.
[xv] http://cpsc.ualr.edu/milanova/image_processing/week1/introduction_07_06.pdf. 13/02/15
[xvi] Wikipedia. Digital image processing.http://en.wikipedia.org/wiki/Digital_image_processing#History. 13/02/15
[xvii] Wikipedia. Comparison of numerical analysis software. http://en.wikipedia.org/wiki/Comparison_of_numerical_analysis_software. 25/02/15
[xviii] Wikipedia. MATLAB. http://en.wikipedia.org/wiki/MATLAB. 23/02/15.
[xix] Math Utha. MATLAB Basics and a little beyond. http://www.math.utah.edu/~eyre/computing/matlab-intro/. 23/02/15.
[xx] Wikipedia. GNU Octave. http://en.wikipedia.org/wiki/GNU_Octave. 23/02/15.
[xxi] Telos. Matlab vesus Octave. http://www.telos.de/matlab/matlab-versus-octave/. 23/02/15.
[xxii] Google Books. GNU Octave. https://books.google.com.ec/books?id=_2dOUBnCzpwC&pg=PT28&lpg=PT28&dq=drawback+octave&source=bl&ots=oioDKGWXNn&sig=avjhT1y943aemZShtSx0hGR7bD4&hl=es&sa=X&ei=35rrVNDoKbaPsQTQ3YHADA&ved=0CEYQ6AEwBQ#v=onepage&q=drawback%20octave&f=false. 23/02/15.
[xxiii] Wikipedia. Scilab. http://en.wikipedia.org/wiki/Scilab. 24/02/15.
[xxiv] Wikipedia. Scilab. http://en.wikipedia.org/wiki/Scilab. 24/02/15.
[xxv] Wikipedia. FreeMat. http://en.wikipedia.org/wiki/FreeMat. 24/02/15.
[xxvi]Google code. FreeMat. https://code.google.com/p/freemat/wiki/ParallelizationPlans. 24/02/15.
[xxvii] Grey Colorado. Comparison of Neural Network Simulators. https://grey.colorado.edu/emergent/index.php/Comparison_of_Neural_Network_Simulators. 28/02/15.
[xxviii] Jon's Java Imaging Library, for mobile image processing. https://code.google.com/p/jjil/. 12/06/15.
[xxix] OpenCV. http://opencv.org/.12/06/15.
[xxx]Gary Chern, Paul Gurney and Jared Starman. Face Detection. https://web.stanford.edu/class/ee368/Project_03/Project/reports/ee368group01.pdf. 01/04/15
[xxxi] An Application of Face Recognition System using Image Processing and Neural Networks. Rakesh Rathi (Ph.D.*), Manish Choudhary (M.Tech.*), Bhuwan Chandra (Ph.D.*) Departamento de Ingeniería de Computación & TI., Colegio Ingeniería del Gobierno, Ajmer, India Departamento de Ingeniería de Computación & TI., Escuela Investigación, Universidad Singhania, Pacheri Bari, Jhunjhunu. http://www.ijcta.com/documents/volumes/vol3issue1/ijcta2012030108.pdf. 02/04/15
[xxxii] Scholarpedia. Eigenfaces. http://www.scholarpedia.org/article/Eigenfaces.08/04/15
[xxxiii]ECE533 – Image Processing Project. Face Recognition Techniques. https://homepages.cae.wisc.edu/~ece533/project/f06/orts_rpt.pdf. 09/04/15