

Programación 1: Lenguaje C

Primer obligatorio

Análisis de gestos faciales

Docentes:

Marcelo Bondarenco: mbondarenco@gmail.com,
Mariano Parodi: mparodi@cup.edu.uy

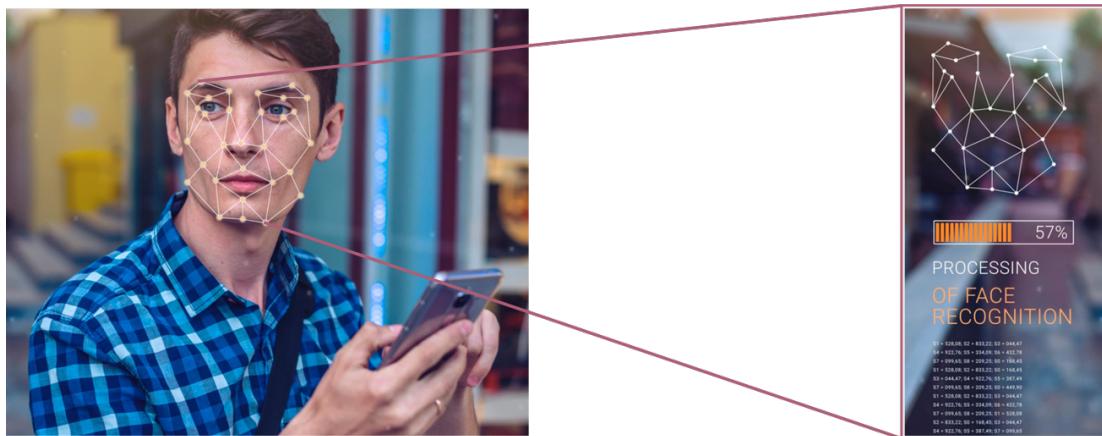
1. Introducción al problema

1.1. Reconocimiento de gestos

El reconocimiento de gestos forma parte de las ciencias de la computación y la tecnología de lenguaje teniendo como objetivo poder interpretar gestos humanos a través de algoritmos matemáticos. Los gestos pueden ser cualquier movimiento corporal pero comúnmente se origina a partir de la cara o la mano. En este obligatorio nos centraremos en los gestos faciales. El reconocimiento de gestos puede verse como una manera de que las computadoras empiecen a entender el lenguaje corporal humano tal de entablar una comunicación.

1.1.1. Aplicaciones

En la actualidad el reconocimiento facial y la detección de gestos, es utilizado en diversas áreas de la comunidad, tanto como en el área de la salud como en aplicaciones para juegos, redes sociales (Facebook, Instagram, Snapchat, etc..), seguridad entre otras. La detección de gestos, se puede utilizar para la identificación de emociones, o simplemente traducirlas a órdenes asociadas.



1.2. Objetivos

Se pretende que el estudiante, utilizando técnicas de programación y álgebra lineal sea capaz de identificar los siguientes gestos faciales:

- Ojos abiertos o cerrados.
- Boca abierta o cerrada.
- Cara rotada o derecha.

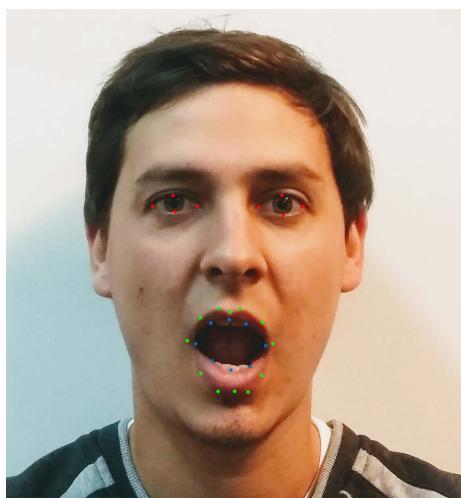
Para esto se entregara al estudiante un conjunto de archivos en formato de texto (.txt) con las coordenadas de los puntos faciales específicos del ojo y la boca tal que a partir de estos pueda realizar los cálculos correspondientes e identificar los gestos que vemos en las siguientes figuras:



(b) Boca cerrada ojos abiertos



(c) Ojo izquierdo cerrado



(d) Boca abierta



(e) Cara rotada

Figura 1

2. Metodología de trabajo

Algunas recomendaciones generales sobre cómo trabajar con proyectos:

- Prolijidad. Documentar bien lo que se hace es fundamental; es muy fácil olvidarse lo que uno mismo hizo. Esto incluye comentarios y el uso de variables con nombres auto explicativos, si es posible.
- Incrementalidad. Implementar y probar de a pequeños pasos. No escribir todo el código de entrada. Es muy difícil encontrar las causas de un problema si se prueba todo simultáneamente.

3. Formato del archivo a entregar

Se deberá entregar un archivo comprimido en formato **zip** de nombre: *apellido-nombre.zip*. El mismo debe contener lo siguiente archivos generados durante el laboratorio:

- Cuerpo del programa (**main.c**).
- Archivos de coordenadas con las que realizo los cálculos (**01_abierto.txt**, **02_guinio.txt**, etc...).
- Archivo de texto plano (**resultados.txt**). El mismo deberá contener los siguientes resultados:
 - **Nombre del archivo**
 - **Ojos**
 - Distancia vertical ojo izquierdo.
 - Distancia vertical ojo derecho.
 - Ojo izquierdo abierto o cerrado.
 - Ojo derecho abierto o cerrado.
 - **Boca**
 - Distancia vertical de los 3 puntos de la boca interior.
 - Promedio de distancias.
 - Boca abierta o cerrada.
 - **Rotación**
 - Angulo de rotación.
 - Cara rotada o derecha.

Por ejemplo, si su nombre es Pablo Rodriguez, entonces debe subir un archivo de nombre **rodriguez_pablo.zip** con el siguiente contenido:

- main.c
- 01_abierto.txt, 02_guinio.txt, 03_boca.txt, 04_rotado.txt.
- resultados.txt

- **Ejemplo de resultados.txt**

01_archivo.txt

Ojos

distIzq: 60 px

distDer: 62 px

Estado: Ojos abiertos

Boca

dist $P1$: 9 px

dist $P2$: 8 px

dist $P3$: 8 px

Promedio ($P1, P2, P3$): 8.33 px

Estado: Boca cerrada

Rotación

Angulo: 2

Estado: Cabeza sin rotar

...

...

...

El resultado de todos los archivos deben de estar dentro del mismo archivo (resultado.txt) como se muestra en el ejemplo anterior.

4. Consigna

Al estudiante se le entregara para la realización del laboratorio, un archivo .zip con el siguiente contenido:

- Un archivo .c (*main.c*) el cual contendrá las funciones necesarias para utilizar y en el cual deberá escribir su programa.
- Cuatro archivos .txt (*01_abierto.txt*, *02_guinio.txt*, *03_boca.txt*, *04_rotado.txt*). Estos contendrán las coordenadas ordenadas por regiones, con las que deberán trabajar. Dichas regiones son las siguientes:
 - Coordenadas del ojo izquierdo: #ojolIzq.
 - Coordenadas del ojo derecho: #ojolDer.
 - Coordenadas de la boca interior: #bocaInt.

Estos identificadores (#región) se utilizarán a la hora de cargar los datos.

- Cuatro imágenes de un rostro con los identificadores de los puntos faciales de los archivos anteriores (no deberán utilizarla mas que para corroborar, cada una corresponde a un .txt).
- **Imagen de referencia con indice de coordenadas.**

Las coordenadas en los archivos, de los puntos faciales tanto de los ojos como de la boca, están ordenadas de igual forma que se enumeran en la Figura 2.

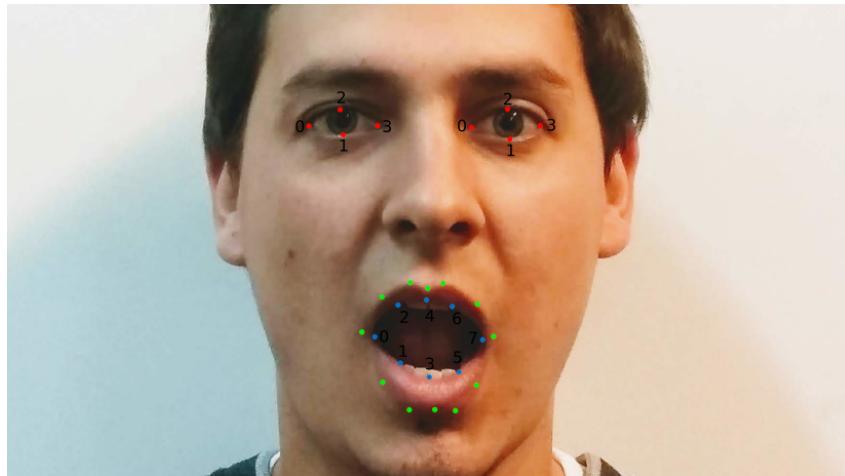


Figura 2: Ejemplo de imagen con los puntos identificados.

4.1. Introducción

El estudiante deberá escribir su código en el main (sin modificar las funciones dadas por el grupo de docentes), y deberá realizar los ejercicios en el orden dado. Durante la etapa de implementación se recomienda que se trabaje con un solo archivo y al finalizar y corroborar que funcione correctamente, se probara con el resto de los archivos.

Al finalizar se deberá ejecutar el código para cada archivo de coordenadas dado (01_abierto.txt, 02_guinio.txt, 03_boca.txt, 04_rotado.txt.) y copiar los resultados impreso en pantalla al archivo *resultados.txt* como se describe al inicio de la letra.

Se utilizará el conjunto de datos del archivo **01_abierto.txt** como ejemplo para la letra de la consigna.

4.2. Ejercicios

1. Ejercicio 1 (Cargar los datos).

En este ejercicio se pretende que el estudiante cargue los datos n (tamaño de array) y las coordenadas en x e y de los archivos.

- a) Cargar en dos variables enteras la dimensión (n) de los vectores coordenadas de los ojos, y de la boca (interior). Para esto deberá utilizar la función dada *loadRango()*.

Ejemplo:

```
1 int Nojo = loadRango("#ojolIzq");
```

- b) Definir 6 array de tipo entero y tamaño N (cargado en la parte (a)) y cargar las coordenadas x e y de:

- Ojo izquierdo.
- Ojo derecho.
- Boca (interior).

Para esto deberá utilizar la función *loadCoord()*.

Ejemplo:

```

1 //Defino array para las coord (x,y) del ojo izquierdo
2 int xOjoIz [Nojo] , yOjoIz [Nojo];
3 //Cargo las coordenadas en los array definidos
4 loadCoord (Nojo , xOjoIz , yOjoIz , "#ojoIzq");

```

2. Ejercicio 2 (Implementación de funciones)

En este ejercicio se pretende que el estudiante implemente 3 funciones que utilizará mas adelante para el cálculo de los descriptores.

- a) Implementar una función (*distancia()*) que reciba como parámetro dos puntos $P_1 = (x_1, y_1)$ y $P_2 = (x_2, y_2)$ (4 valores enteros). Dicha función deberá calcular y retornar la distancia (d) entre ambos puntos.

La distancia entre dos puntos ($d(P_1, P_2)$) se calcula de la siguiente forma:

$$\begin{aligned}
d(P_1, P_2) &= \|P_2 - P_1\| \\
&= \|(x_2, y_2) - (x_1, y_1)\| \\
&= \|(x_2 - x_1, y_2 - y_1)\| \\
&= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}
\end{aligned} \tag{1}$$

RECOMENDACIÓN: Utilizar la función `sqrt()` de la librería `<math.h>` para el cálculo de la raíz cuadrada y la función `pow()` para el cálculo de las potencias.

- b) Implementar una función (*centroide()* tipo int) que reciba como parámetros:

- Array `<int>` para almacenar las coordenadas (x, y) del centroide a calcular.
- Array `<int>` con las coordenadas x de los puntos del ojo.
- Array `<int>` con las coordenadas y de los puntos del ojo.
- Variable `<int>` con el tamaño de los vectores del ojo.

dicha función deberá calcular el centroide de los puntos del ojo.

El centroide entre N puntos ($Cen(P_1, \dots, P_N)$) se calcula de la siguiente forma:

$$\begin{aligned}
Cen(P_1, \dots, P_N) &= \vec{C}_n = (x_c, y_c) \\
\Rightarrow (x_c, y_c) &= \left(\frac{1}{N} \sum_{i=1}^N P_{i(x)} , \quad \frac{1}{N} \sum_{i=1}^N P_{i(y)} \right) \\
&= \left(\frac{1}{N} [P_{1(x)} + P_{2(x)} + \dots + P_{N(x)}] , \quad \frac{1}{N} [P_{1(y)} + P_{2(y)} + \dots + P_{N(y)}] \right) \\
\Rightarrow \vec{C}_n &= \frac{1}{N} ([P_{1(x)} + P_{2(x)} + \dots + P_{N(x)}] , \quad [P_{1(y)} + P_{2(y)} + \dots + P_{N(y)}]) \tag{2}
\end{aligned}$$

- c) Implementar una función (*calcularAngulo()*) que reciba como parámetros dos puntos $C_{n1} = (x_{c1}, y_{c1})$ y $C_{n2} = (x_{c2}, y_{c2})$ donde C_{n1} son las coordenadas del centroide del ojo izquierdo y C_{n2} las del ojo derecho.

Dicha función deberá generar el punto C_{n3} que vemos en la Figura 3 y luego deberá calcular y retornar el ángulo θ .

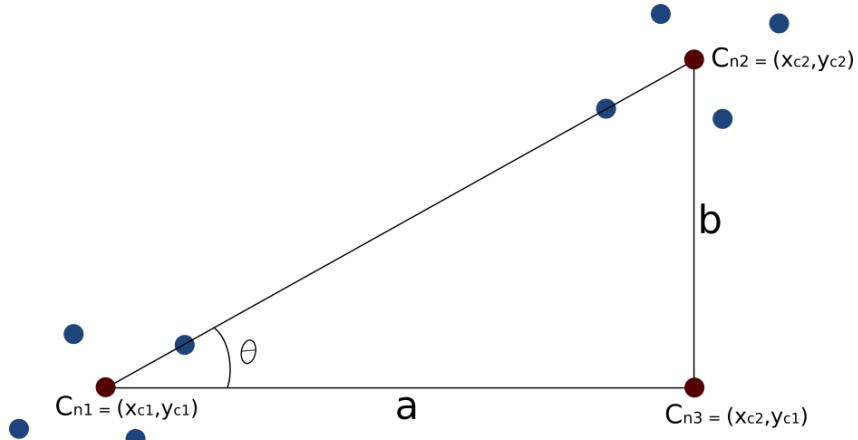


Figura 3

Tener en cuenta que $x_{c3} = x_{c2}$ y que $y_{c3} = y_{c1}$. Para calcular el ángulo θ deberán utilizar razones trigonométricas:

$$\begin{aligned}\tan(\theta) &= \frac{\text{Cateto opuesto}}{\text{Cateto adyacente}} \Rightarrow \tan(\theta) = \frac{b}{a}, \quad a = \|C_{n3}C_{n1}\| \quad y \quad b = \|C_{n2}C_{n3}\| \\ \Rightarrow \tan(\theta) &= \frac{\|C_{n2}C_{n3}\|}{\|C_{n3}C_{n1}\|} = \frac{\|(x_{c2} - x_{c3}, y_{c2} - y_{c3})\|}{\|(x_{c3} - x_{c1}, y_{c3} - y_{c1})\|} = \frac{\sqrt{(x_{c2} - x_{c3})^2 + (y_{c2} - y_{c3})^2}}{\sqrt{(x_{c3} - x_{c1})^2 + (y_{c3} - y_{c1})^2}} \\ \Rightarrow \tan(\theta) &= \frac{d(C_{n2}, C_{n3})}{d(C_{n3}, C_{n1})} \Rightarrow \theta = \arctan\left(\frac{d(C_{n2}, C_{n3})}{d(C_{n3}, C_{n1})}\right)\end{aligned}$$

RECOMENDACIÓN: Utilizar la función `atan()` de la librería `<math.h>` para el cálculo del arctan. Dado que θ esta en radianes deberán convertirlo a grados:

$$\theta^\circ = \theta^{rad} \cdot \frac{180}{\pi}$$

3. Ejercicio 3 (Cálculo de descriptores)

En este ejercicio se pretende que el estudiante utilice las funciones implementadas en el Ejercicio 2 para el cálculo de los descriptores los cuales deberán utilizar para identificar los gestos faciales, anteriormente definidos.

- Calcular la distancia entre los puntos (1,2) de los ojos izquierdo y derecho. Si la distancia entre estos puntos de cada ojo es menor a 30 px entonces decimos que el ojo se encuentra cerrado. Imprimir en pantalla las distancias de ambos ojos y si se encuentran abiertos o cerrados.
- Calcular la distancia entre los puntos (1,2), (3,4) y (5,6) de la boca. Imprimir en pantalla las 3 distancias. Evaluar el promedio de las distancias y si es menor a 30 px entonces se considera que la boca esta cerrada (imprimo "Boca cerrada"), en caso que no la boca esta abierta (imprimo "Boca abierta").
- Calcular los centroides de los 4 puntos de cada ojo. Luego a partir de esto calcular el anguló (θ) que se forma entre el segmento formado por ambos centroides ($\overrightarrow{C_{n1}C_{n2}}$) con el segmento proyectado ($\overrightarrow{C_{n1}C_{n3}}$). Si el ángulo resultante es mayor a 10° entonces decimos que la cara esta rotada. Imprimir en pantalla el anguló y si la cara se encuentra rotada o no.

- d) Generar el archivo *resultados.txt* copiando y pegando los resultados impresos en pantalla para cada ejecución.

5. Anexo

En el siguiente cuadro se remarcán las funciones que deberán utilizar para cargar los datos solicitados. Cada una de estas tiene una breve descripción de lo que hacen y los parámetros que reciben. A tener en cuenta:

- No se deberán modificar ni eliminar ninguna de las funciones dadas.
- Se deberá escribir las funciones en el espacio remarcado al igual que el programa.

```

1  '',
2  En nameFile deben colocar el nombre del archivo con el cual van a ejecutar.
3  Ej: "01_abierto.txt"
4  ''
5  char nameFile[30] = {"01_abierto.txt"};
6
7  ''
8  Esta funcion parsea el rango de los array a cargar
9  Parametros:
10   - region []: De que region cargar el n
11   - Ej: "#ojoIzq"
12  ''
13 int loadRango(char region[])
14  ''
15  Esta funcion parsea los datos de las coordenadas
16  Parametros:
17   - cant: n cargado anteriormente (dimension del array)
18   - coordX []: array donde almacenar las coordenadas en x
19   - coordY []: array donde almacenar las coordenadas en y
20   - region []: region de la cual se pretende cargar las coordenadas.
21   - Ej: loadCoord(Nojo,xOjoIz,yOjoIz,"#ojoIzq");
22  ''
23 void loadCoord(int cant, int coordX[], int coordY[], char region[])
24
25 ///////////////////////////////////////////////////////////////////
26 /////////////////////////////////////////////////////////////////// AQUI VAN LAS FUNCIONES A IMPLEMENTAR ///////////////////////////////////////////////////////////////////
27 ///////////////////////////////////////////////////////////////////
28 /////////////////////////////////////////////////////////////////// AQUI VA EL CUERPO DEL PROGRAMA COMPLETO ///////////////////////////////////////////////////////////////////
29
30 int main()
31 {
32 ///////////////////////////////////////////////////////////////////
33 /////////////////////////////////////////////////////////////////// AQUI VA EL CUERPO DEL PROGRAMA COMPLETO ///////////////////////////////////////////////////////////////////
34 ///////////////////////////////////////////////////////////////////
35 }
36
37

```