# EigenFaces
## Métodos Numéricos

Dan Heli Muñiz Sánchez — Lic. en Computación Matemática
Jafet Castañeda — 10 de octubre de 2023

---

**Resumen**

En este proyecto se describe un método para la clasificación y reconocimiento de caras, *Eigenfaces*. Se desarrolla una interfaz interactiva donde el usuario puede elegir imágenes a comprar, observar las *Eigenfaces* generadas y la reconstrucción de rostros con una cantidad $k$ de *Eigenfaces*.

# 1. Introducción

El reconocimiento de caras es un método biométrico para la identificación de individuos mediante características faciales. Existen diversas aplicaciones prácticas, abarcando numerosas disciplinas, donde puede utilizarse el reconocimiento de caras. Dichas aplicaciones pueden categorizarse en aplicaciones de seguridad o comerciales. El reconocimiento de caras es un problema de clasificación utilizando datos de alta dimensión, por lo que para permitir un reconocimiento rápido y robusto es necesario realizar tareas de reducción de dimensión (extracción de características) antes de la clasificación. Los métodos de reducción de dimensión puede clasificarse en dos categorías principales.

Para este proyecto decidimos trabajar sobre el tema de *EigenFaces* ("Caras Propias"). Una cara propia es el nombre que se le da a un conjunto de *Eigenvectores* ("vectores propios") el cual se utiliza comúnmente en el problema de visión por computadora del reconocimiento de rostros humanos. Los personajes Sirovich y Kirby desarrollaron el enfoque de usar caras propias para el reconocimiento y lo usaron quienes lo usaron para clasificación de caras fueron Matthew Turk y Alex Pentland.

Los vectores propios se derivan de la matriz de *covarianza* sobre el espacio vectorial de alta dimensión de imágenes de rostros. Las caras propias forman un conjunto base de todas las imágenes utilizadas para construir la matriz de *covarianza*. Esto produce una reducción de la dimensión al permitir que el conjunto más pequeño de imágenes base represente las imágenes de entrenamiento originales. La clasificación de imágenes se puede lograr comparando cómo se representan las caras por el conjunto base.

## 2. Descripción del Método

Es bien sabido que una fotografía se puede representar como una matriz de píxeles , si la foto es cuadrada entonces es una matriz $k \times k$ , esta la podemos representar como un vector $k^2 \times 1$ donde se conservara la información de la fotografía

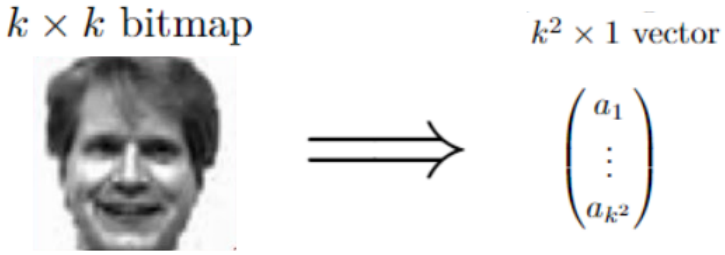



Figura 2.1: Representar una cara como un vector

El procedimiento anterior se repita para cada una de las $n$ caras que se tienen de muestra, y con los vectores resultantes que denotaremos como $\Gamma_i$ con $i = 1, 2, \ldots, n$. Ahora debemos definir la **cara promedio (mean face)** de la siguiente manera

$$\Psi = \frac{1}{n} \sum_{i=1}^{n} \Gamma_i$$

Gracias a esto, podemos construir la matriz $A$ como

$$A = [\Phi_1, \Phi_2, \cdots, \Phi_n]$$

Donde

$$\Phi_i = \Gamma_i - \Psi$$

Cada $\Phi_i$ representa que tanta diferencia hay entra cada una de las caras diferentes y la cara promedio.

Gracias al resultado anterior, podemos construir la matriz de **covarianza** C como

$$C = \frac{1}{n} A^T A$$

Esta matriz no necesariamente es cuadrada, pero gracias a un resultado en álgebra lineal, sabemos que $A^T A$ y $AA^T$ tienen los mismo *eigenvalores* y *eigenvectores* , por lo tanto consideramos el segundo producto matricial, y de este encontraremos los *eigenvectores* ($v_i$) y los *eigenvalores* ($\lambda_i$), así obtendremos vectores de dimensión $k^2 \times 1$ des de normalizarlos como

$$u_i = \frac{v_i}{||v_i||}$$

estos serán nuestras *eigencaras*. Esto nos permite la reconstrucción aproximada de cara $\Gamma_i$ mediante la siguiente ecuación matricial

$$\Gamma_i = \Psi + \begin{bmatrix} u_1 & \cdots & u_m \end{bmatrix} \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

o también se puede ver como una simple combinación lineal

$$\Gamma_i = \Psi + \sum_{i=1}^{m} w_i u_i$$

Una meta de interés es el calculo de los pesos $w_j$ , y estos se obtienen de la siguiente manera

$$w_j = u_j^T \Phi_i,$$

donde $j$ es la $j$-ésima posición del vector de pesos y $i$ es de la $\Gamma_i$ imagen que se quiere reconstruir.

El vector de pesos para cada cara $\Gamma_i$ lo definiremos como $\Omega_i = [w_1, \cdots, w_m]$ que puede ser la representación de cada cara bajo el sistema coordenado generado por las *Eigencaras*, el vector $\Omega_i$ es particularmente útil para empezar con la tarea de clasificación , puesto que podemos usar la *distancia euclideana* para ver que tan similares son las caras entre si y encontrar emparejamientos.

# 3. Experimentos y Resultados

En esta sección se describen los experimentos realizados para llegar a implementar el método de clasificación de caras. Para esto, se ha utilizado las bases de imágenes: INSERTAR NOMBRE. Dicha base contiene 25 imágenes de tamaño $425 \times 425$ de diferentes individuos.


Figura 3.1: Ejemplo de imágenes de la base de caras

Notemos que todas las imágenes utilizadas están en escala de grises. Entonces, dada la base de imagenes, calculamos la **cara promedio** (**mean face**), así obteniendo la siguiente imagen

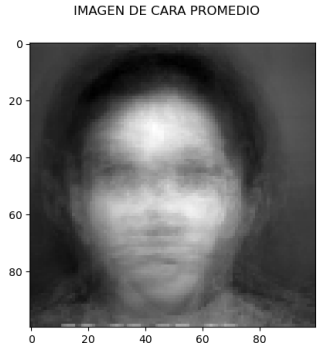



Figura 3.2: Cara promedio de la base de imágenes

Dada la **cara promedio**, calculando la matriz de **covarianza** y sus correspondientes *eigenvectores* y *eigenvalores* obtenemos la siguiente base de *EigenFaces*


Figura 3.3: EigenCaras

Notemos que dependiendo del color de piel, la iluminación y diferentes factores, cada *eigencara* varía. Lo interesante de este experimento es que ahora es posible reconstruir imágenes usando la base de *EigenFaces* y los pesos correspondientes. A continuación se presenta la reconstrucción de las imágenes de la base, usando una cantidad $k$ de *EigenFaces*


(a) (b)

Figura 3.4: Reconstrucción

Al haber hecho el proceso de reconstrucción de imágenes, también se experimentó con el proceso de clasificar o reconocer imágenes. Como se comentó en la **Descripción del Método**, se implementó la distancia *euclideana* para comparar y clasificar una imagen. Veamos los siguientes ejemplos

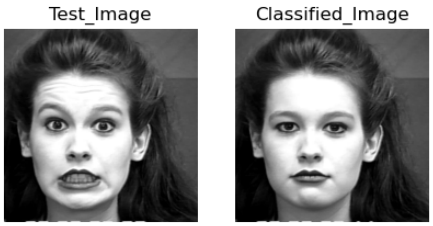



Figura 3.5: Ejemplo de Clasificación

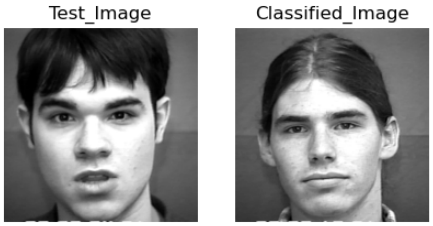



Figura 3.6: Ejemplo de Clasificación


Figura 3.7: Ejemplo de Clasificación


Figura 3.8: Ejemplo de Clasificación

Notemos que las imagenes de prueba no pertenecen a la base, aunque incluimos las mismas personas solo que con expresiones diferentes o simplemente son personas diferentes que son asociadas a la persona más cercana en cuanto a la distancia *euclideana*.

Ahora, para visualizar lo anterior, implementamos una interfaz interactiva usando la librería de tkinter. A continuación se explica su funcionamiento. Realizando el siguiente comando en la terminal obtenemos de manera visual la herramienta.

```
python EigenFaces.py
```

Con lo anterior, obtenemos en primera estancia la siguiente interfaz

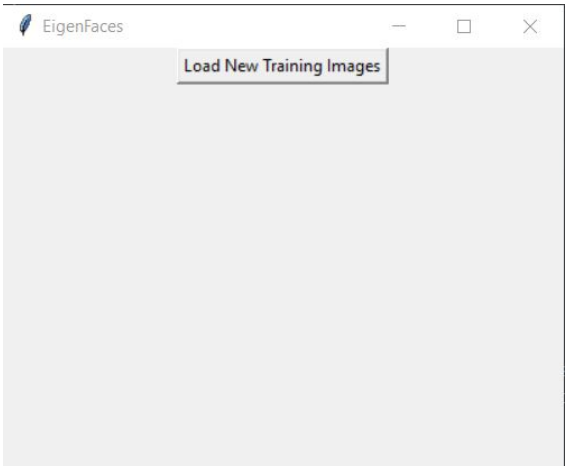



Figura 3.9: Primer Instancia de Interfaz

Notemos que en la Figura 3.9 tenemos un botón para cargar la carpeta de las imágenes base de entrenamiento. Al elegir la opción, el usuario deberá de elegir la carpeta y abrirla ahí mismo. A continuación se presenta un ejemplo.

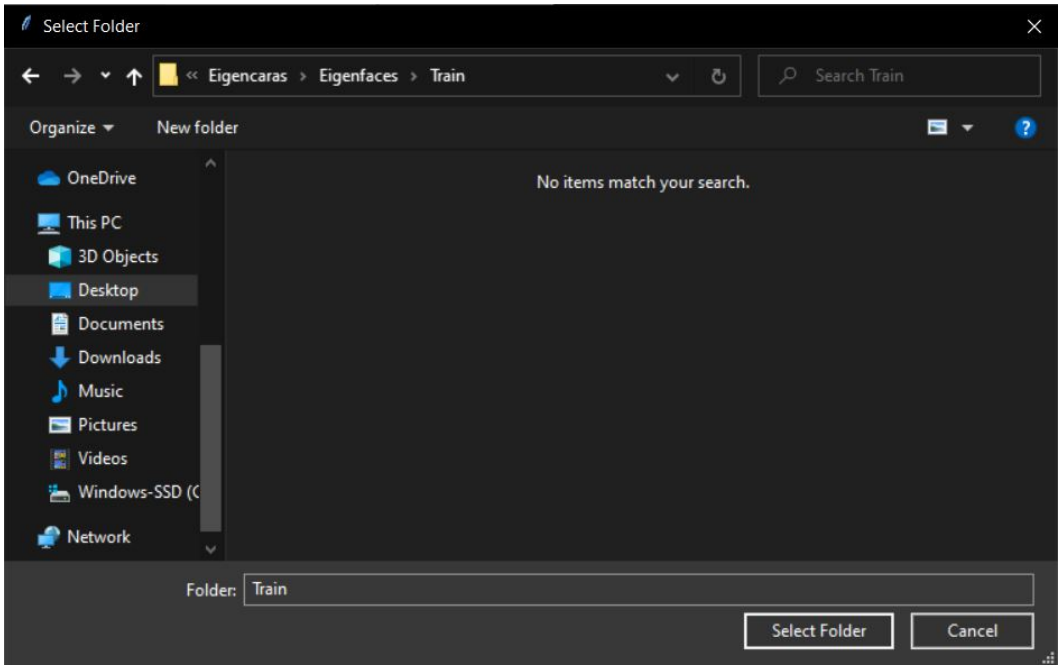



Figura 3.10: Seleccionamos el Folder y lo abrimos

Una vez que seleccionamos la carpeta de imágenes obtendremos el siguiente menú

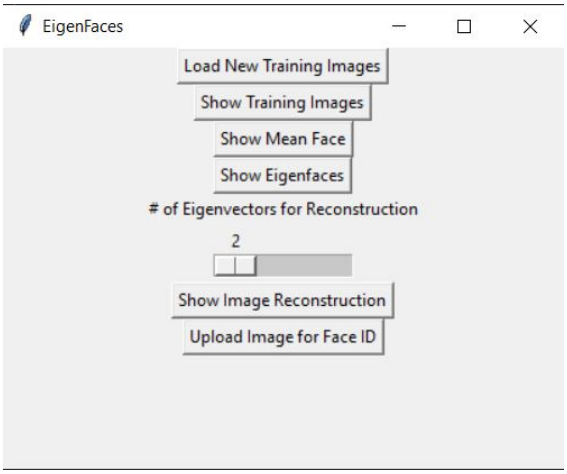
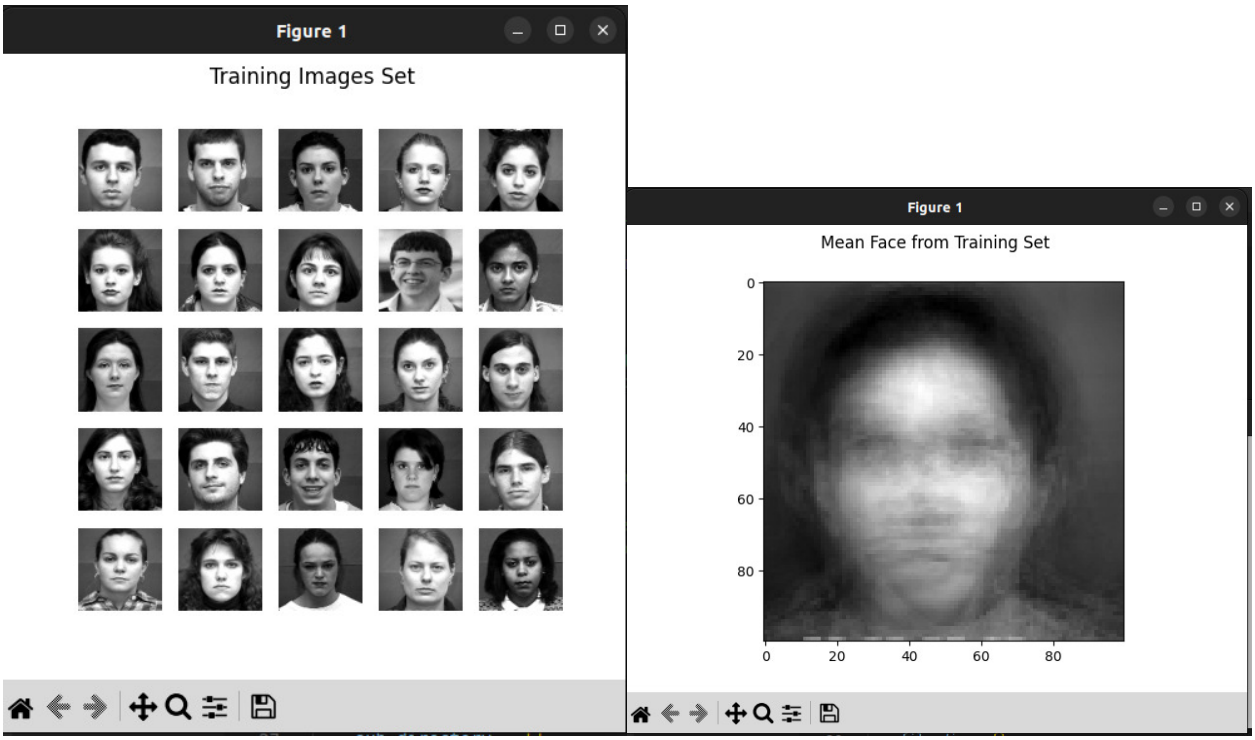


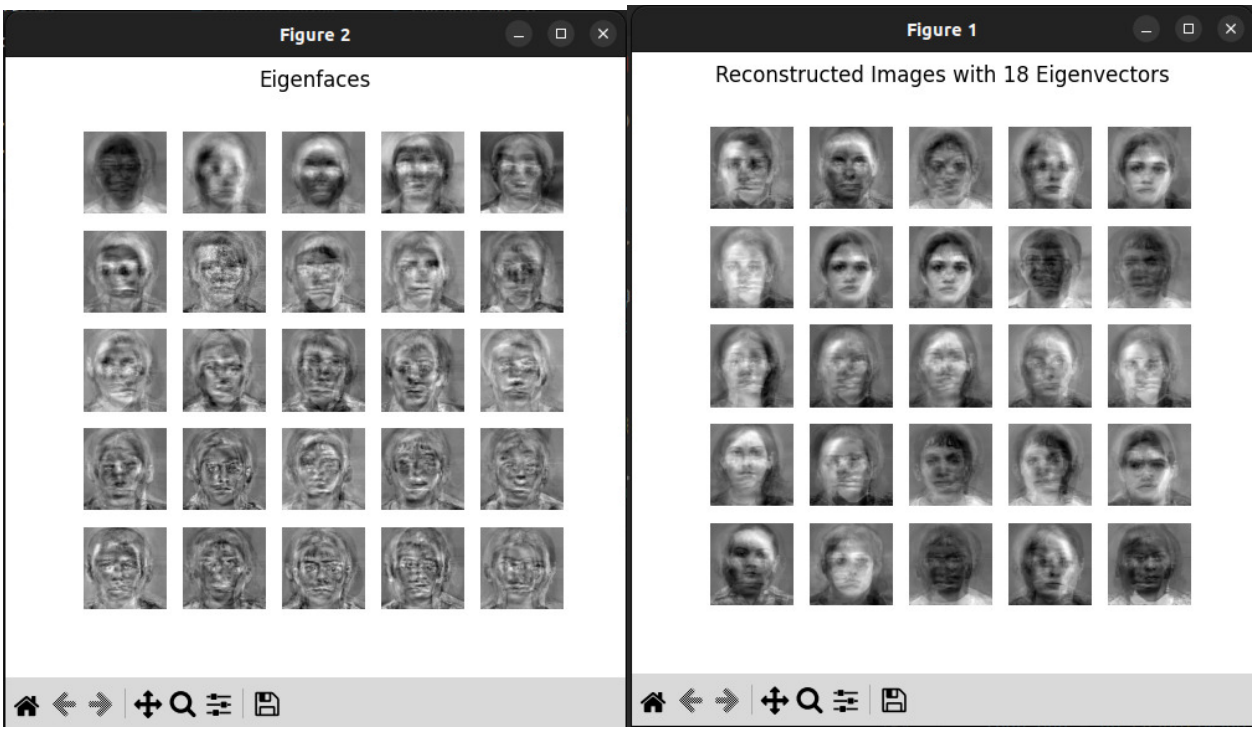

Figura 3.11: Menú de operaciones

Correspondiendo al botón seleccionado obtienes una de las siguientes imágenes.




(a) (b)

Figura 3.12: Resultados




(a) (b)

Figura 3.13: Resultados

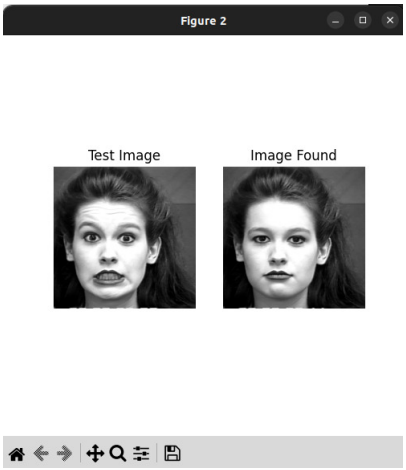



Figura 3.14: Clasificación de Imágenes

# 4. Conclusiones

Es claro que el uso de *eigenfaces* es la punta del *iceberg* cuando de reconocimiento facial se trata, pero este método ofrece muchas ventajas, entre ellas el que se pueda usar todo lo que ya sabemos de álgebra lineal a un nivel intermedio, poder imaginar una noción geométrica al momento de concebir las caras resultantes, algunas desventajas podría ser la gran manipulación de datos que necesita hacer, principalmente al momento de computar la matriz de covarianza, pero siempre se puede optar por métodos alternativos mas avanzados, un ejemplo seria la descomposición de valores singulares (**SVD**) , que por lo que se ha investigado para este proyecto suele ser el método favorito cuando se trabaja con fotografías que representan grandes matrices y cuando se tiene una muy basta base de datos.

# Bibliografía


- Burden, Richard; Douglas Faires. Análisis numérico. Editorial Iberoamérica, México, 1985.
- M. Turk and A. Pentland. Eigenfaces for recognition. J. Cognitive Neuroscience, 1991.
- Papers with code - CK+ dataset. CK+ Dataset — Papers With Code. (n.d.). Recuperado el 4 de diciembre, 2022, de https://paperswithcode.com/dataset/ck
- Wikimedia Foundation. (2022, November 10). Eigenface. Wikipedia. Recuperado el 4 de diciembre, 2022, de https://en.wikipedia.org/wiki/Eigenface


$$f(z) = z^4 \quad f(z) = \frac{1}{z^4}$$