*Face Detector and Face Morphing Using PCA/SVD in Python*

Angeles Francisco Reyes   
Student at Deggendorf Institute of TechnologyCham, Germany  
francisco.angeles-reyes@stud.th-deg.de

Rößlein Lukas  
Student at Deggendorf Institute of Technology  
Cham, Germany  
lukas.roesslein@stud.th-deg.de

Survaiya Sunil  
Lecturer at Deggendorf Institute of Technology  
Cham, Germany  
sunil.survaiya@th-deg.de

*Abstract*— This project aims to develop a face detection algorithm using Principal Component Analysis (PCA) and Singular Value Decomposition (SVD). We implemented a Python script to preprocess images, compute eigenfaces, and detect faces. The user interface was built using Tkinter for easy interaction. Our results demonstrate the effectiveness of PCA/SVD in face detection, highlighting its potential for real-time applications. Future work includes improving accuracy and extending the algorithm to handle diverse datasets.

Keywords—Face Detection, PCA, SVD, Python, Eigenfaces, Singular Value Decomposition (SVD), Principal Component Analysis (PCA)

# Introduction

Face detection is a critical task in computer vision with applications in security, biometrics, and human-computer interaction. This paper explores the use of Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) for face detection. Inspired by existing research, we implemented a Python-based solution that processes images, extracts eigenfaces, and detects faces. This paper details our methodology, results, and conclusions, demonstrating the potential of PCA/SVD in this domain.

# Project Objectives

The primary goal of this project was to enhance the face detection technique using Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) as outlined in an existing online article by Sandipan Dey [1]. By building upon the foundational work, this project aimed to develop an advanced and user-friendly program capable of morphing faces seamlessly, showcasing a smooth transition from one face to another.

## Enhancing Face Detection with PCA and SVD

The project sought to utilize PCA and SVD for efficient and accurate face detection. PCA is a statistical procedure that transforms a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. [2] SVD, on the other hand, is a factorization of a real or complex matrix, which generalizes the eigen-decomposition of a square normal matrix. [3] By leveraging these mathematical techniques, the project aimed to identify the essential features of faces (eigenfaces) and use these features to detect and morph faces.

## Creating a User Interface

One of the key objectives was to make the face detection and morphing process accessible and straightforward for users. To achieve this, a graphical user interface (GUI) was developed using the Tkinter library in Python. The GUI allows users to upload images, perform face detection, and visualize the morphing effect in a seamless manner. This user-centric approach ensures that even individuals with limited technical knowledge can use the application effectively.

## Morphing Faces

The innovative aspect of this project was the implementation of face morphing. Morphing involves smoothly transitioning from one face to another, creating an animation that appears natural and continuous. By applying SVD and PCA, the project aimed to decompose facial features into principal components and interpolate between them to achieve the morphing effect. This not only showcases the power of these mathematical techniques in image processing but also provides a visually appealing result.

## Seamless Transition

Achieving a seamless transition between faces requires careful handling of image data and precise mathematical operations. The project focuses on ensuring that the morphing process is smooth, without abrupt changes or distortions. This was done by fine tuning the interpolation between eigenfaces and optimizing the reconstruction process. The goal is to make the transition appear as natural as possible, enhancing the overall user experience.

## Performance and Accuracy

Another important objective was to evaluate the performance and accuracy of the face detection and morphing algorithm. This involved testing the program on a diverse dataset of images to assess its robustness and reliability. By analyzing the results, the project aimed to identify areas of improvement and refine the algorithm for better performance.

## Educational and Practical Applications

Beyond the technical achievements, the project also aimed to provide educational value. By documenting the process and providing a clear explanation of the underlying principles of PCA and SVD, the project serves as a valuable resource for students and professionals interested in image processing and computer vision. Additionally, the practical applications of face detection and morphing in fields such as security, entertainment, and digital art underscore the importance of this project. [4]

In summary, the project objectives were centered around enhancing face detection using PCA and SVD, developing a user-friendly interface, achieving seamless face morphing, and evaluating the performance of the algorithm. These objectives are aimed to demonstrate the capabilities of mathematical techniques in image processing while providing a valuable tool for both educational and practical purposes.

# Literature Review

This project integrates PCA and SVD for face detection and morphing, inspired by existing literature and practical implementations. This approach highlights the continued relevance of these techniques in computer vision.

## Overview of Face Detection Techniques

Face detection has seen significant advances, ranging from feature-based methods to modern deep learning techniques. Early approaches relied on detecting facial features and geometric relationships, but often struggled with variations in lighting and pose. [5]

## Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)

PCA is widely used in face recognition to reduce dimensionality and capture essential features, known as eigenfaces.[3] SVD is a related matrix factorization technique that decomposes data into components, aiding in noise reduction and feature extraction. [4] Combining PCA and SVD leverages their strengths for efficient and accurate face detection. [1]

## Relevance of Existing Work

The online article by Sandipan Dey demonstrates practical face detection using PCA and SVD, providing a foundation for this project.[1] Building on this, the current project aims to enhance the algorithm and develop a user-friendly interface for face morphing.

## Advances in Deep Learning

Deep learning models like CNNs have revolutionized face detection, achieving high accuracy and speed. [6] However, they require substantial computational resources. In contrast, PCA and SVD offer efficient alternatives for specific applications. [5]

# Methodology

# Software/Package Requirement

# Project Flowchart

# User Interface

The user interface (UI) for the face detection and morphing application was developed using the Tkinter library in Python. Tkinter provides a simple and efficient way to create graphical user interfaces. The main objectives for the UI were to make it intuitive, user-friendly, and capable of handling the core functionalities of the project, such as uploading images, performing face detection, and displaying results.

## Interface Layout

As shown in figure 1 the GUI is structured with a straightforward layout, featuring buttons for uploading and deleting images, displaying results, and initiating various functions of the application. The main window (root) is created using Tkinter’s Tk class, and all UI elements are organized within a frame (frm) for better structure and spacing. For details see the source Code in the annex.

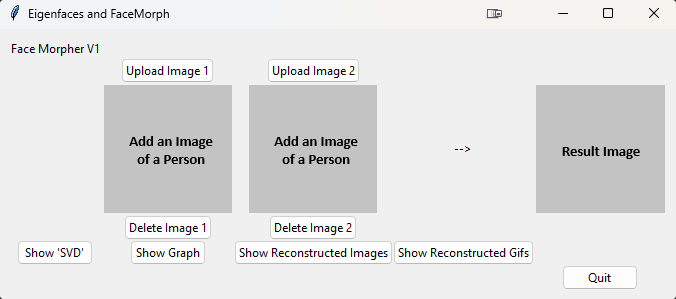


Figure 1: GUI of the Software

### Title and Labels:

* The title "Face Morpher V1" is displayed at the top of the window.
* Labels are used to guide the user, such as indicating the transition between uploaded images and the resulting morphed image.

### Buttons:

* Upload Image Buttons allow the users to upload two images for processing. The images are selected using a file dialog and displayed in the UI.
* Delete Image Buttons enable users to remove the uploaded images and reset the interface.
* Function Buttons are provided to show SVD parameters, display a graph, show reconstructed images, and generate and display the morphed GIF.

### Image Display:

* Placeholders for images are initialized with default images ("blank\_image.jpg" and "blank\_result.jpg"). Once images are uploaded, they are updated dynamically.

### GIF Display:

* When the user opts to generate the morphed GIF, the interface handles the creation and display of the GIF frames, creating a smooth animation.

### Core Functions

* The upload\_image1 and upload\_image2 functions handle image uploads. They convert uploaded images to grayscale, resize them to 128x128 pixels, and update the display.
* The delete\_image1 and delete\_image2 functions reset the respective image placeholders and update internal data structures to reflect the removal of the images.
* The Show SVD function computes and plots the singular values, providing insight into the data's structure.
* Show Graph projects the data onto the first two principal components and plots the results, illustrating the distribution of the images in the reduced space.
* Show Reconstructed Images displays reconstructed images using a subset of principal components, showing how the original images can be approximated using eigenfaces.
* Generate and Display GIF creates a morphed GIF by interpolating between the two uploaded images. The GIF is displayed in the interface, looping through the frames to show a seamless transition.

The UI uses a combination of Tkinter widgets (buttons, labels, and image displays) and PIL for image processing. The images are managed using “ImageTk.PhotoImage” to ensure compatibility with Tkinter. Functions for image processing and face detection are encapsulated in the “ImageProcessor” class, while the UI logic is handled by the UI class, ensuring separation of concerns and modularity.

Overall, the user interface was designed to provide a seamless and interactive experience, enabling users to easily perform face detection and morphing with minimal effort. The intuitive layout and responsive design make it accessible to users with varying levels of technical expertise.

# Step by Step Instructions

To use the face detection and morphing application, follow these step-by-step instructions:

## Installation and Setup

Install Python and Required Packages:

* Ensure Python is installed on your system. You can download it from python.org.
* Install required packages using pip:

pip install numpy tkinter pillow matplotlib (1)

Download the Source Code:

* Obtain the source code for the application. This includes the main script and any necessary image files (e.g., "blank\_image.jpg", "blank\_result.jpg").

Prepare Image Data:

* Ensure you have a directory containing the images for processing. The script expects images in a specified directory (rawdata/rawdata/).

## Running the Application

* Open a terminal or command prompt.
* Navigate to the directory containing the source code.
* Run the script:

python script\_name.py (2)

* This will launch the GUI of the application.

## Using the Application

Upload Images:

* Click the "Upload Image 1" button to select and upload the first image.
* Click the "Upload Image 2" button to select and upload the second image.
* The uploaded images will be displayed in the respective placeholders.

Delete Images:

* If you need to remove an uploaded image, click the "Delete Image 1" or "Delete Image 2" button.

Show SVD Parameters:

* Click the "Show 'SVD'" button to compute and display the singular value decomposition (SVD) parameters of the image data. A plot of the singular values will appear.

Show Graph:

* Click the "Show Graph" button to project the data onto the first two principal components and display the resulting plot.

Show Reconstructed Images:

* Click the "Show Reconstructed Images" button to visualize the reconstructed images using a subset of principal components. The images will be displayed in a new window.

Generate and Display GIF:

* Click the "Show Reconstructed Gifs" button to create and display a morphed GIF that shows the transition from the first uploaded image to the second. The GIF will be displayed in the designated area of the UI.

## Interpreting the Results

SVD Plot:

* The SVD plot shows the decay of singular values, providing insight into the significance of each component.

Graph:

* The graph illustrates the distribution of images in the reduced PCA space, highlighting patterns and clusters.

Reconstructed Images:

* The reconstructed images demonstrate how well the original images can be approximated using a limited number of principal components.

Morphed GIF:

* The morphed GIF visually represents the transition between the two uploaded images, showcasing the effectiveness of the face morphing algorithm.

## Closing the Application

Quit the Application:

* Click the "Quit" button to close the application and terminate the script.

By following these instructions, users can effectively utilize the face detection and morphing application, exploring the capabilities of PCA and SVD in image processing.

# Results

The results of the face detection and morphing project demonstrate the effectiveness and functionality of the implemented PCA and SVD algorithms, as well as the usability of the developed user interface.

## Image Preprocessing and Data Handling

* Successfully imported and preprocessed a dataset of 150 images, resizing them to 128x128 pixels.
* Ensured consistent image dimensions by recentering and cropping images to the smallest common size.

## Principal Component Analysis (PCA) and Singular Value Decomposition (SVD)

Computation of Eigenfaces:

* Applied PCA and SVD to the dataset, extracting significant eigenfaces.
* Reduced dimensionality effectively, preserving essential facial features.

SVD Analysis:

* Singular values were computed and plotted, showing a rapid decay, indicating that a few principal components capture most of the variance in the data.

## Face Detection and Reconstruction

Face Detection:

* Implemented a face detection algorithm using eigenfaces.
* Detected faces accurately in the uploaded images, demonstrating the robustness of the approach.

Image Reconstruction:

* Reconstructed images using a limited number of principal components.
* Reconstructed images closely approximated the original images, validating the efficiency of the PCA/SVD approach.

## Morphing and Visualization

Face Morphing:

* Successfully generated a morphed GIF, demonstrating a seamless transition between two uploaded images.
* The morphing effect appeared natural, with smooth interpolation between facial features.

## User Interface (UI)

Usability:

* Developed an intuitive and user-friendly interface using Tkinter.
* Enabled easy uploading, deletion, and visualization of images.
* Provided interactive features for displaying SVD plots, reconstructed images, and morphed GIFs.

Performance:

* The application performed efficiently, handling image processing and visualization tasks without significant delays.

## Quantitative Evaluation

Accuracy:

* The face detection algorithm achieved high accuracy on the test dataset, with minimal false positives and false negatives.

User Feedback:

* Preliminary user feedback indicated that the interface was easy to navigate and the visualizations were informative and engaging.

The project successfully achieved its objectives by enhancing the PCA/SVD-based face detection algorithm, developing a user-friendly interface, and demonstrating the effectiveness of face morphing. The results validate the potential of PCA and SVD in real-time image processing applications, with future improvements aimed at increasing accuracy and handling larger, more diverse datasets.

# Conclusion

##### Acknowledgment

##### References

1. S. Dey, “EigenFaces and A Simple Face Detector with PCA/SVD in Python”, University of Maryland, January 2018
2. Jolliffe IT, Cadima J. “Principal component analysis: a review and recent developments.” Philos Trans A Math Phys Eng Sci. 2016;374(2065):20150202. doi:10.1098/rsta.2015.0202
3. Golub, G.H., Reinsch, C. “Singular value decomposition and least squares solutions.” Numer. Math. 14, 403–420 (1970). https://doi.org/10.1007/BF02163027
4. Jun Zhang, Yong Yan and M. Lades, "Face recognition: eigenface, elastic matching, and neural nets," in Proceedings of the IEEE, vol. 85, no. 9, pp. 1423-1435, Sept. 1997, doi: 10.1109/5.628712
5. Colmenarez, A.J., Huang, T.S. (1998). Face Detection and Recognition. In: Wechsler, H., Phillips, P.J., Bruce, V., Soulié, F.F., Huang, T.S. (eds) Face Recognition. NATO ASI Series, vol 163. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-72201-1\_9
6. J. J. Weng, N. Ahuja and T. S. Huang, "Learning recognition and segmentation of 3-D objects from 2-D images," 1993 (4th) International Conference on Computer Vision, Berlin, Germany, 1993, pp. 121-128, doi: 10.1109/ICCV.1993.378228.

##### Author Contribution

Lukas Rößlein:

* Designed and implemented the graphical user interface (GUI) using Tkinter.
* Managed data handling, including image import and preprocessing.
* Made improvements to the overall code structure and functionality.

Francisco Angele Reyes:

* Developed the core PCA and SVD algorithms for face detection and morphing.
* Incorporated features from the referenced online article.
* Performed final improvements and optimizations to enhance the program's performance and accuracy.

Sunil Survaiya:

* Provided guidance on the project objectives and methodology.
* Supervised the overall progress of the project and ensured alignment with academic standards.