Image guessing game using Image compression and singular value decomposition

What is SVD?

Singular value decomposition is the process of converting a matrix of any given size m x n into the form U\*S\*V^T (transpose) where U is a m x m orthogonal matrix, S is a m x n matrix and V is a n x n matrix. This decomposition allows understand the nature of a given linear transformation matrix since every linear transformation can be achieved through a rotation, scaling and another rotation which are described by the matrices U, S and V respectively.

SVD’s Application in image compression

The decomposition has various applications in the IT field. When examined closely, the 3 matrices allow us to exploit not only a better understanding of the linear transformation, but also **dimensionality reduction**. As seen in the figure below, we can obtain matrices of lower ranks (with linearly dependent rows), that are approximately close to the original matrix using the product of just a row matrix and a column matrix. This allows us to save space and still possess a good representation of the original image. The more detailed we require the image to be, the more matrices we need to store and vice versa.

A screenshot of a computer game

Description automatically generated

Aim of the project:

To demonstrate a solid understanding of the C programming language and present an application of Linear algebra through innovatively designing an engaging game that utilizes SVD as a tool for achieving image compression.

About the game:

Firstly, a BMP image of any size is chosen and assigned a name. The end program will prompt the user to guess the name of the image by displaying the image via a suitable graphical user interface. On the first run, the user will be shown an image that uses just 1 row matrix, and a column matrix generated from the decomposition, which provides a simplified version of the image. The user will be able to use certain number of hints in return for their score. Each ‘hint’ will mean that a few extra row matrices, column matrices and singular values will be extracted from the decomposition to add further detail to the image.

Does it really save space?

Storing the whole decomposition will obviously require as much space as the original matrix. In order to avoid this, we need to truncate the decomposition by only taking a certain number of singular values. There are C libraries that allow the retrieval of complete or partial decomposition data as desired by the programmer. This enables the reconstruction of the image without needing to compute the entire decomposition, hence saving space and time. But for this project I have opted to compute the whole decomposition to further develop my skills in the language.

Diary (8/11/2024):

Day 1: Did thorough research about SVD and Image compression, tried to write down what each of those meant in simple words.

Day 2: Started to describe each of the components of the project in more detail. Ran into a question: wouldn’t we have to store the decomposition as well, which defeats the whole purpose of storing the image with less space? Upon research, only a certain number of singular values need to be considered. So, the process would be finding eigenvalues, sorting them, finding singular values, finding eigenvectors for V, and using the formulae ui = (1/σ) \* A \* vi find the orthonormal vectors in U

Day 3: initialised git repository, added few files & started working on converting a given image to matrix form.

Day 4: Faced many challenges. OpenCV was required to convert an image of any format into matrix form. Downloading OpenCV for C was quite an impossible task. As a result, RGB value sin the BMP file were translated into greyscale values using a weighted sum. Greyscale value calculation is as follows: unsigned char gray = (unsigned char)(0.299 \* rgb[2] + 0.587 \* rgb[1] + 0.114 \* rgb[0]); This was done using Copilot. A module for the conversion was created and used in the Masterfile. The program now only accepts valid BMP images.