جامـــــعـة الـمــــسـتـقـبـل
AL MUSTAQBAL UNIVERSITY

كلية العلوم

قســـم الانـظـمـة الـطـبية الـذكـيـة

Intelligent Medical Systems Department

**Subject: DSP**

**Class: Third Stage**

**Lecturer: Asst.Lec. Reyam Thair Ahmed**

# Lecture: ( 6 )

**Study Year: 2024-2025**

# Linear Algebra and Image Processing

**Study Year: 2024-2025**

1. **What is the connection between Digital Image Processing (DIP) and linear algebra?**
2. **Basic Examples**

**Study Year: 2024-2025**

# Linear Algebra and Image Processing

Linear algebra is considered one of the most important and popular subjects in the undergraduate mathematics curriculum.

## 1. What is the connection between Digital Image Processing (DIP) and linear algebra?

While linear algebra is the study of vector spaces and matrices that act on themas linear operators, DIP is the study of how to process images or digitalmovies. What are the main connections between the two disciplines?

An image may be defined as a two-dimensional function, $(x,)$, where $x$ and $y$ are spatial (plane) coordinates, and the amplitude of $f$ at any pair of coordinates $(x,y)$ is called the intensity or gray level of the image at that point. When $x$, $y$ and the amplitude values of $f$ are all finite, discrete quantities, we call the image a digital image.

When stored digitally, an $m\times n$ image can be considered as a matrix.

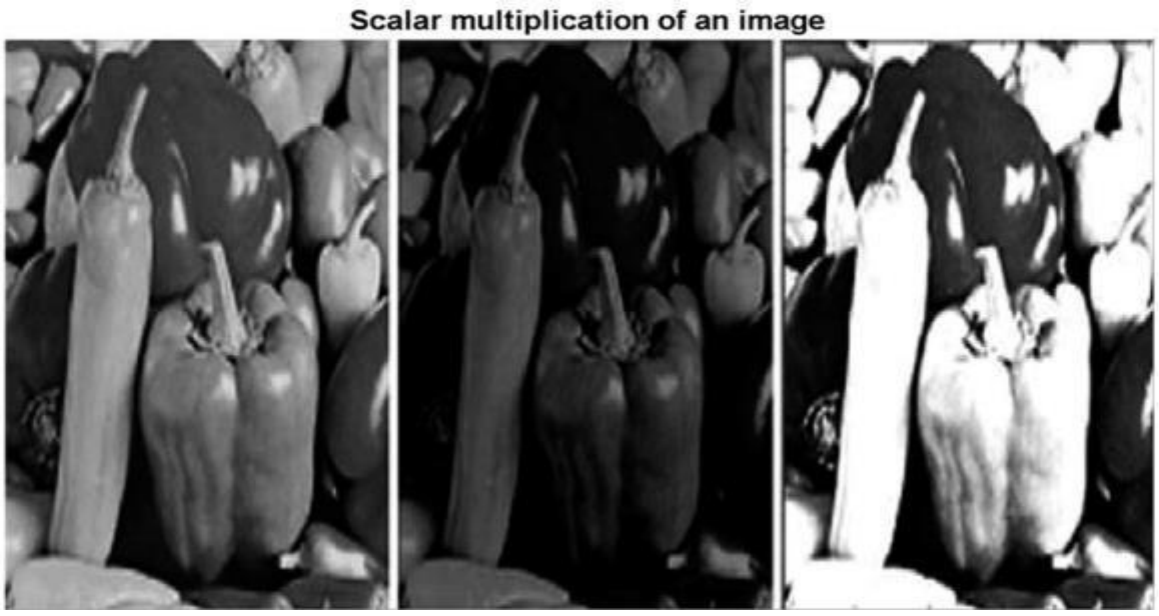
$$f(x,y)=\begin{pmatrix} f(0,0) & f(0,1) & \cdots & f(0,n-1) \\ f(1,0) & f(1,1) & \cdots & f(1,n-1) \\ \vdots & \vdots & & \vdots \\ f(m-1,0) & f(m-1,1) & \cdots & f(m-1,n-1) \end{pmatrix}$$

All of the standard ways to manipulate images (contrast enhancement, lightening, darkening, histogram analysis, fading, warping, etc.) can be performed by applying mathematical operations to the matrix associated with the image.

The natural connection between linear algebra and DIP, supported by tools of contemporary DIP technology, needs to be exploited more efficiently in an elementary linear algebra course. Stretching, rotating or flipping images, and proceed to the contrast enhancement, detecting images in data corrupted by noise, feature extraction, edge detection and techniques of data compression, all these topics are fruitfully studied using concepts from linear algebra.

## 2. Basic Examples:

**Example 1:** One of the first exposed concepts in linear algebra is scalar multiplication. By using matrix scalar multiplication we can change the intensity of the original image. We choose $c = 1/2$ and $c = 3/2$ to get a darker and brighter version, respectively.



Scalar multiplication applied to the original image

**Example 2:** To get familiar with the sigma notation used heavily in a linear algebra course, the 'average optical density' of an image is introduced. The average optical density (AOD) of an image A is defined as:

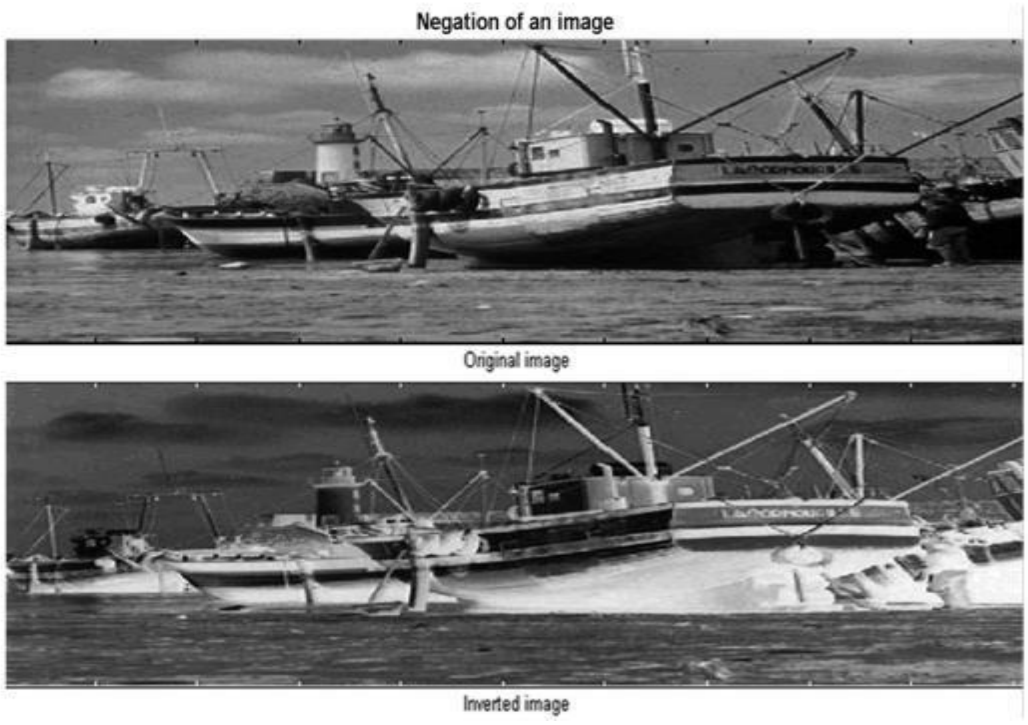
$$\mathrm{AOD}(A) = \left(\frac{1}{mn}\right) \sum_{i=1}^{m} \sum_{j=1}^{n} A(i, j),$$

which is the average intensity of the pixels in the $m \times n$ image A. The AOD is the basic measure of an image's overall average brightness or grey level.

**Example 3:** Inverting the intensities of a given image, also known as taking its negative, is equivalent to subtracting the entries of the associated matrix from the highest intensity level which is 256 for an 8-bit image.

$$B = 256I - A,$$

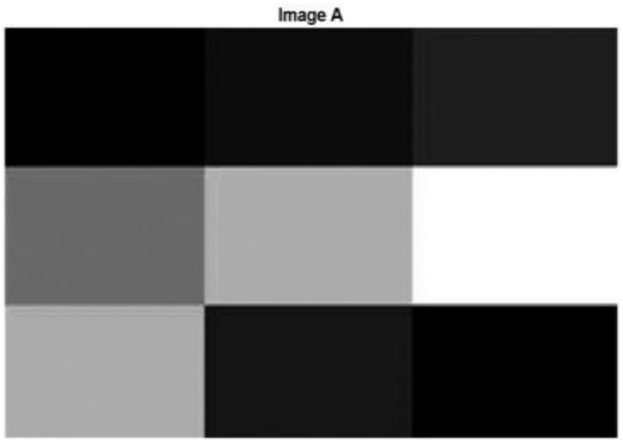
where $A$ is the original image, $B$ the inverted one and $I$ an $n \times n$ matrix of ones.



**Example 4:** In a linear algebra, elementary row operations for matrices are introduced. We present an elementary example to visually show the effect of multiplying matrices on the left by elementary matrices.
Let

$$A = \begin{pmatrix} 30 & 60 & 80 \\ 150 & 200 & 250 \\ 200 & 70 & 30 \end{pmatrix}$$

Using the basic Matlab, our matrix gets transformed to the following image:

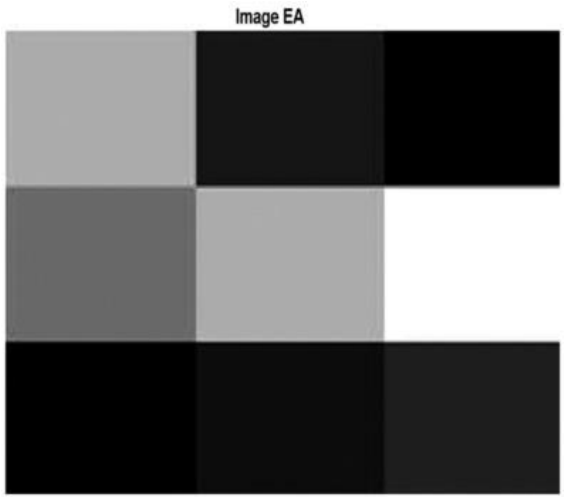


Original image

Let $E$ be the elementary matrix obtained from the identity matrix $I_3$ by a single elementary row operation, interchanging the first and the third row,

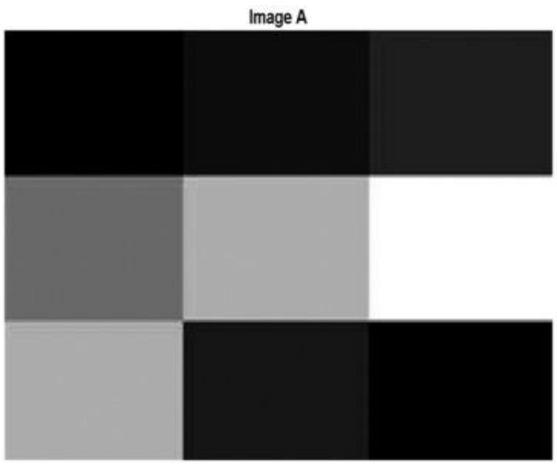
$$E = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

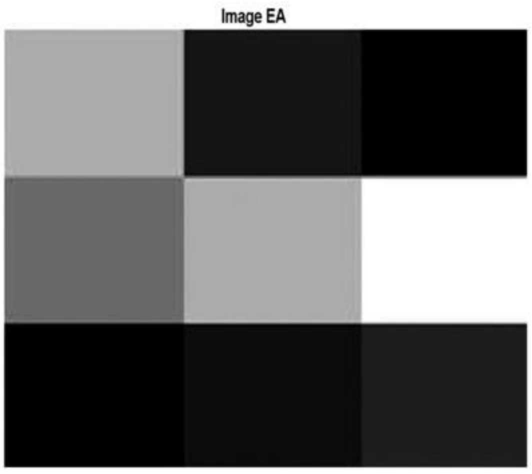
Multiplying matrix $A$ on the left by $E$ flips the image $A$ horizontally, which is shown in the following:



Flipped image

Finally, we get





Before After

Note that multiplying matrix $A$ on the right by $E$ flips the image $A$ vertically.

**Example 5:** The cameraman image is a 256 × 256 greyscale image with 8 bits per pixel. It had been corrupted by salt-and-pepper noise which is also called impulse noise or binary noise. Its appearance is randomly scattered white or black (or both) pixels over the image.

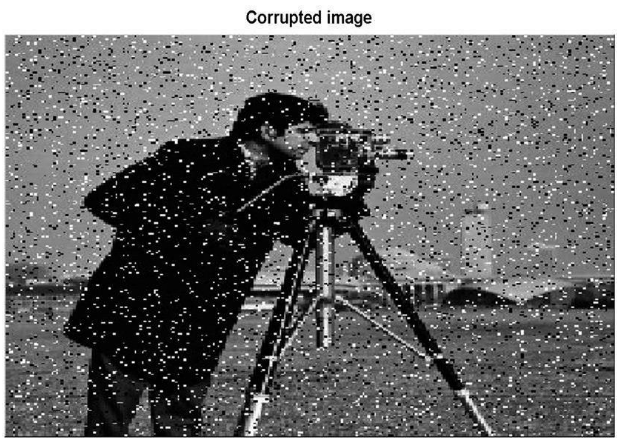
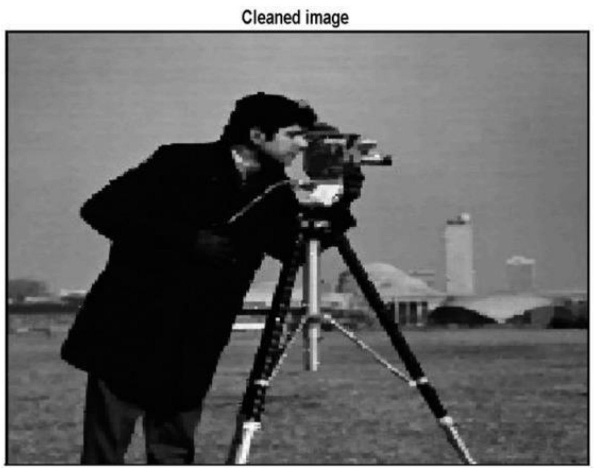


Image corrupted by salt-and-pepper noise

Define the median filtered image as:
For every pixel of the image (entry of the matrix) except pixels on the boundary, the median filter takes the value of the nine pixels around andincluding the pixel itself and finds the median value. Then, it replaces the central pixel value with the median of the structuring element. Applying this method to the corrupted image, we get the cleaned version of the cameraman image.

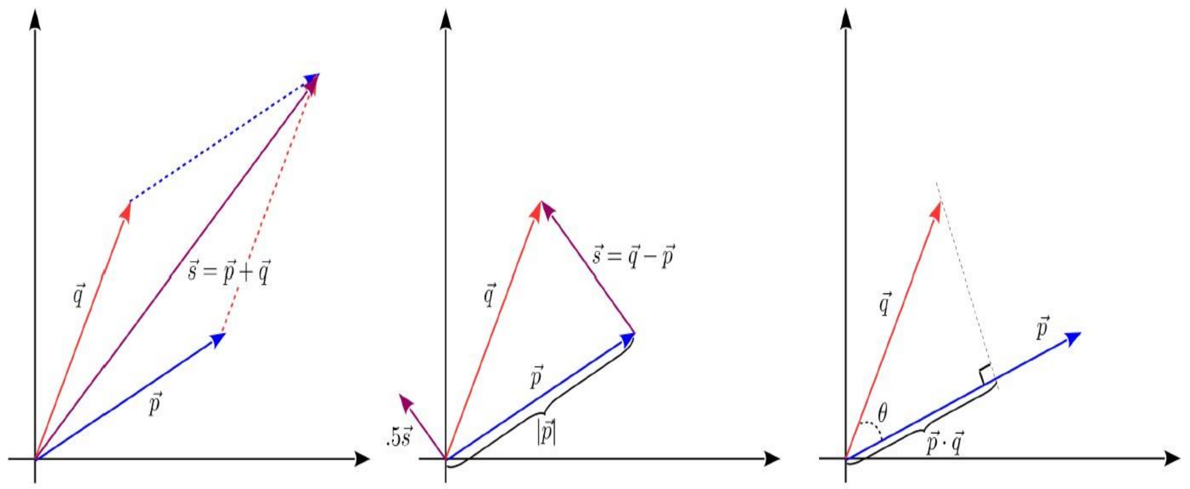


Cleaned image by a median filter

Note that the median filter gives a poor quality in terms of sharp edges. One way of sharpening the edges is by detecting them first and then adding them to the image to create a sharper image.

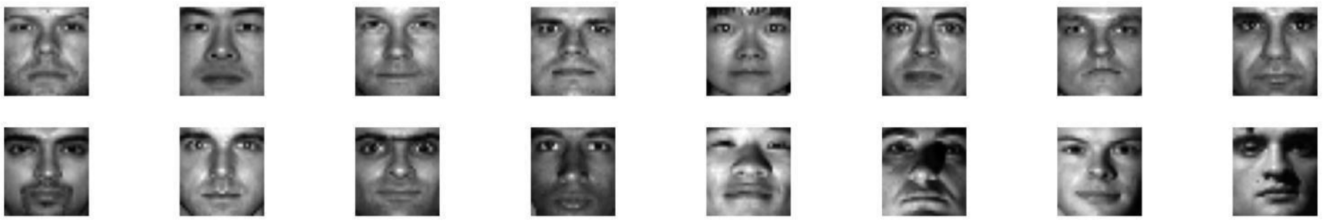
**Example 6:** To remind you, we repeat the Vector operations.
The basic vector operations we are concerned with are:

- **Addition**: $\vec{s} = \vec{p} + \vec{q} = [p_1 + q_1 \;\; p_2 + q_2 \;\; \ldots \;\; p_n + q_n]$
- **Subtraction**: $\vec{s} = \vec{q} - \vec{p} = [q_1 - p_1 \;\; q_2 - p_2 \;\; \ldots \;\; q_n - p_n]$ (note the order of subtraction)
- **Magnitude**: $|\vec{v}| = \sqrt{\sum_i v_i^2}$
- **Scaling**: $\alpha\vec{p} = [\alpha p_1 \;\; \alpha p_2 \;\; \ldots \;\; \alpha p_n]$
- **Dot product**: $\vec{p} \cdot \vec{q} = |\vec{p}||\vec{q}| \cos\theta = \sum_i p_i q_i = pq^T$



Basic vector operations in 2D

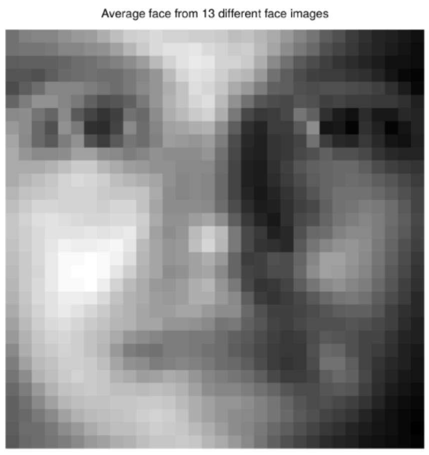
The operations above seem simple enough, but they are already sufficient to let us have some fun with data. Consider a set of pictures of faces. For simplicity, let’s assume they are all small (e.g. 32×32 pixels) and contain only grayscale information.


A small set of pictures of human faces

From the text above, we know we can take each of these faces and transform it into a vector with 1024 entries. What can we learn from these sample faces using a few simple vector operations? Suppose we compute the following:

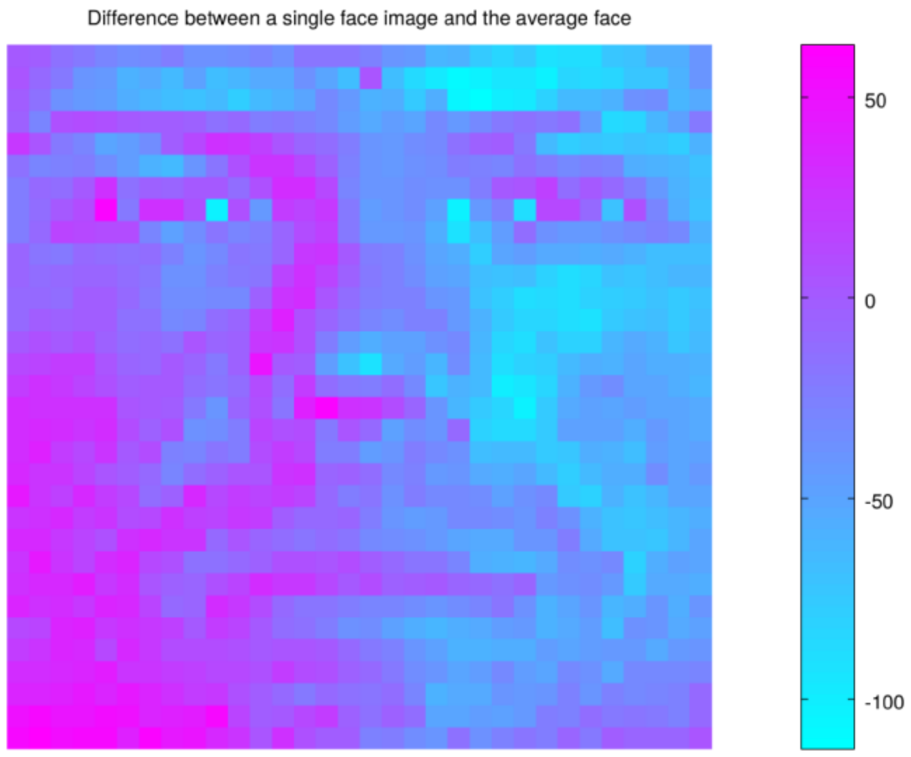
$$\vec{\mu} = \frac{1}{n}\sum_{i=1}^{n}\vec{f}_i,$$

where $f\vec{}i$ is the $i$th face vector. The vector $\mu\rightarrow$ is the mean or average vector for the dataset. Since it comes from a set of face images, its values should correspond to what an average face looks like at least in our very small data set. Let’s reshape the $\mu\rightarrow$ vector back into a 32×32 image and see what that is:



Average face for our small dataset

**Study Year: 2024-2025**



Difference between a sample face and the mean face for the dataset

At first glance, it may not look like we have learned much. The image looks just like a face with a funny color. However, if you pay attention to the color bar at the right of the image, you may notice that the values in the difference image go from somewhere close to -110 to about 60 or so. The range of values is about 170. The range of the original images is 255.

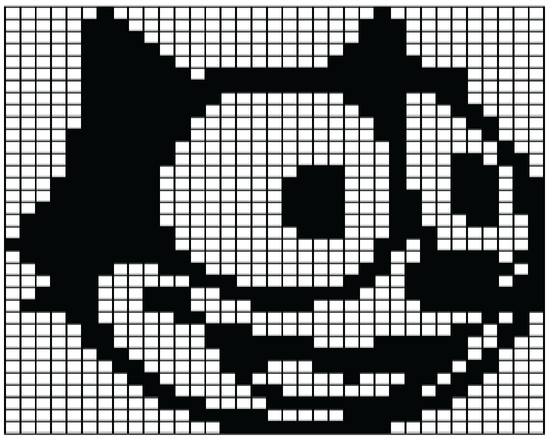
This is meaningful because it suggests we can represent the original images by storing their difference with respect to the mean, along with the mean vector. Why would that be useful? Well, the difference images have a smaller range and a smaller amount of internal variation compared to the originals. They contain less information than the originals. That means they can be compressedmuch more effectively.

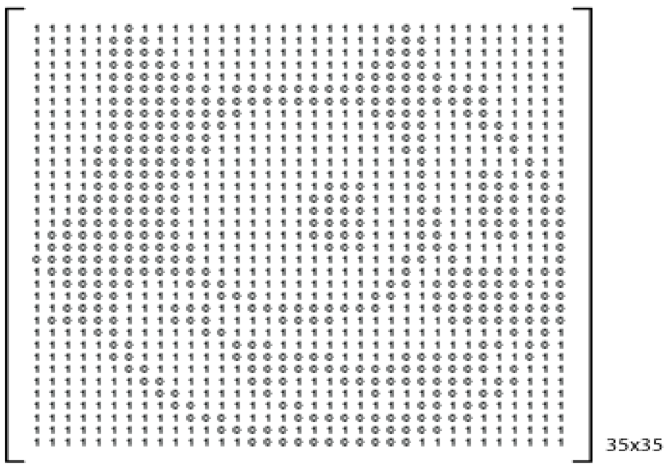
For such a tiny dataset it makes little difference, but imagines having millions of different face images: We could save a lot of bandwidth by compressing and transferring the difference images instead of the originals. Of course, we also need to transfer the mean face image, but there is only one of those.

This is a very crude way to compress information. As we shall see, there are much smarter ways to achieve even greater compression power while also obtaining much more meaningful information about the structure of our data:

The Discrete Cosine Transform (DCT) used in JPEG image compression, and the Discrete Fourier Transform (DFT and its efficient version the FFT) used for sound analysis are examples of simple vector operations that are nevertheless very powerful tools for data analysis.
Example 7: The small image of Felix the Cat

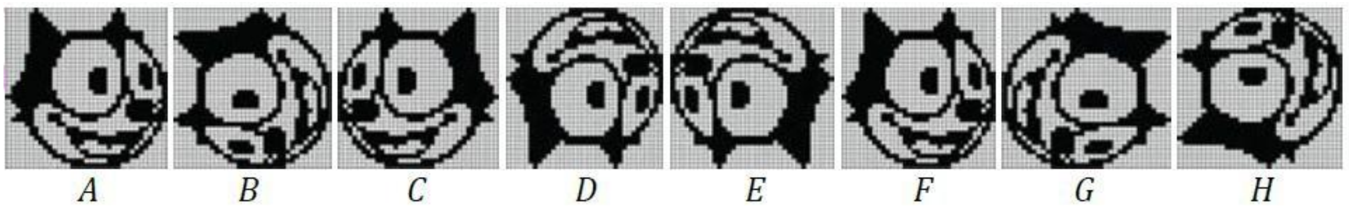
can be represented by a 35 × 35 matrix whose elements are the numbers 0 and 1. These numbers specify the color of each pixel: the number 0 indicates black, and the number 1 indicates white.





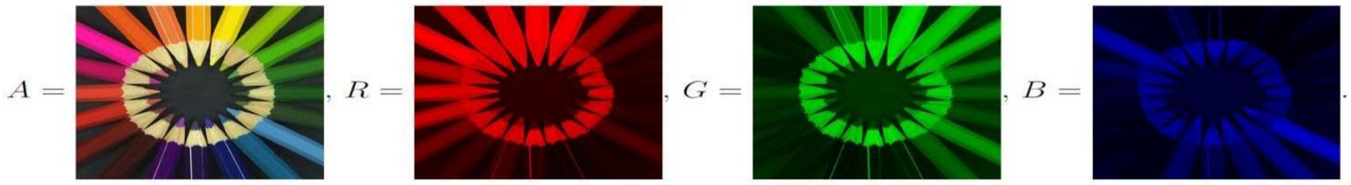
Now, if we consider the binary image $A$ below as a matrix, say $A = (a_{i,})$, then the image $B$ corresponds to the **transposed matrix** of $A$, that is, $B = (b_{i,j}) = (a_{j,i}) = A^T$ . The image $H$, by its turn, corresponds to the matrix $(a_{j,35-i+1})$.

Try to discover the metrical relationships between the image $A$ and the other images!

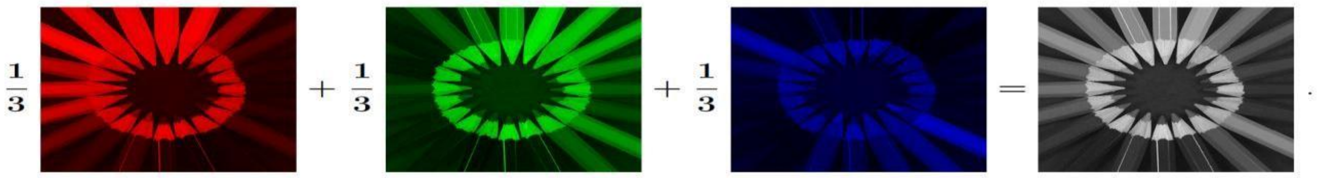


*A* *B* *C* *D* *E* *F* *G* *H*

**Example 8:** If we take the standard arithmetic mean of the component matrices
$R$, $G$ and $B$ from a color image $A$,



we will get a grayscale version of the image (non-integer values are rounded tothe nearest integer):

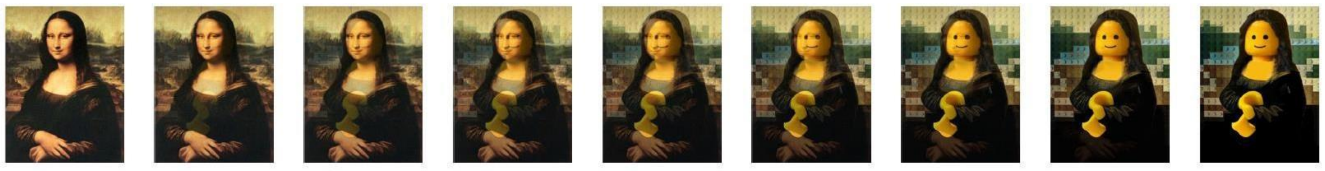


**Example 9:** In this example, we use the operations of multiplication by a scalarand sum of matrices, it is possible to create an image transition effect commonly used, for instance, in PowerPoint presentations and slide shows. More precisely, consider two grayscale images of the same size, represented bythe matrices $A$ and $Z$. For each scalar (real number) $t$ in the interval [0,1], define the matrix

$$M(t) = (1-t)A+tZ.$$

Notice that $(0) = A$, $(1) = Z$ and, for each $t$ between 0 and 1, the elements of the matrix $M(t)$ are between the elements of the matrices $A$ and $Z$. Therefore, when
$t$ varies from 0 to 1, the matrix $(t)$ varies from $A$ to $Z$. For the case of

color images, the transformation above must be applied to the matrices $R$, $G$ and $B$ that compose each image.



$M(0) = A$ $M(0.13)$ $M(0.25)$ $M(0.38)$ $M(0.50)$ $M(0.63)$ $M(0.75)$ $M(0.88)$ $M(1) = Z$