Mathematics Project

Aim/Objective: To develop an in-depth knowledge on the concept of matrices by analysing the different operations on various matrices and studying its significant role in its applications in the field of computer graphics.

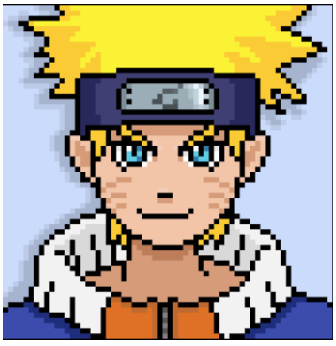
Introduction: The term matrix was introduced by the 19th century English mathematician James Sylvester along with his friend the mathematician Arthur Cayley who developed the algebraic aspect of matrices in two papers in the 1850s. A matrix can be defined as a set of data arranged in rows and columns to form a table or a rectangular array. This data can be any sort of elements or entries. Matrices have varied applications in the fields of physics, economics, statistics, engineering, computer science and last but not the least in mathematics. In this project, we will be talking about the application of matrices in the digital world by studying digital image processing (computer graphics).

Mathematical process: All the images that we see on the internet through our mobile phones, laptops and computers are nothing but digital images. These digital images can be represented using matrices. For example, let's consider the image of a famous Shōnen anime protagonist, Uzumaki Naruto, whose image can be represented by a matrix whose elements are the numbers 0 and 1. In computer graphics, these numbers specify the colour of each pixel where a pixel is the smallest graphical element of a matricial image, which can only access one colour at a time. The number 0 indicates black and the number 1 indicates white. These digital images can be referred to as binary images.

Uzumaki Naruto

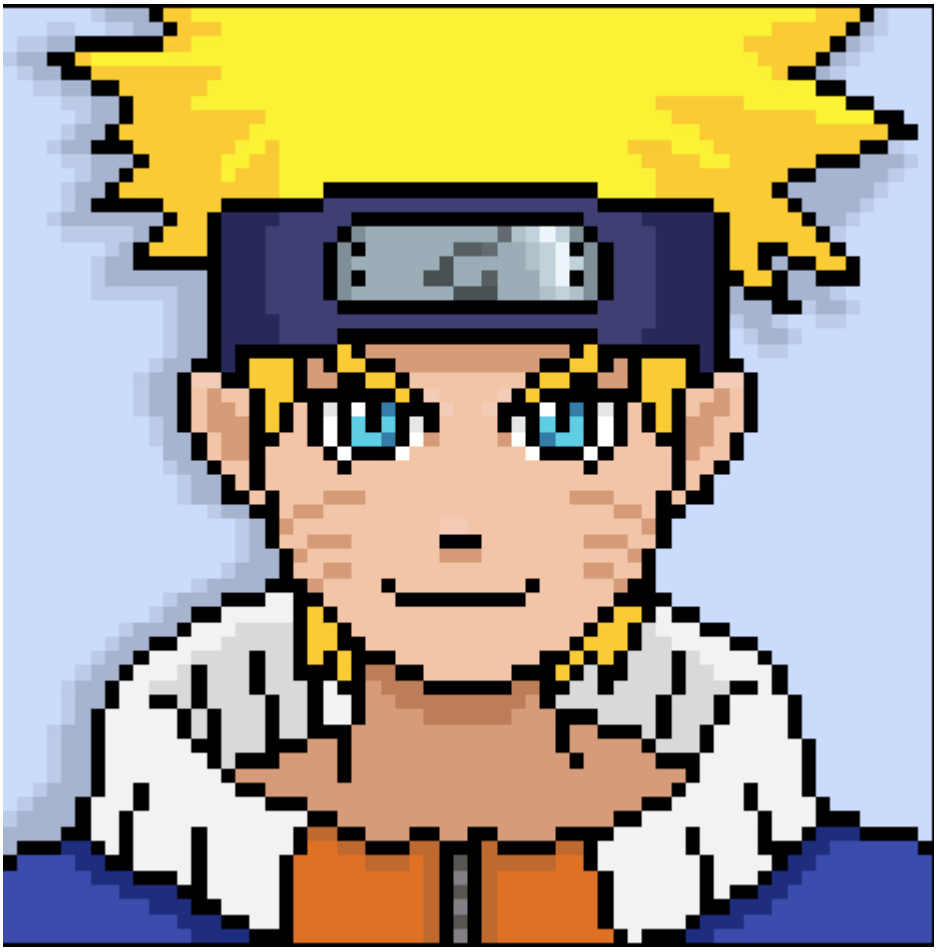
But these types of images are usually not too appealing and flashy to watch, so let us check out how matrices help us to fill in this void by adding colourful elements (pixels) to Naruto.

Colour images can be represented by three matrices. Each matrix specifies the amount of red, green and blue that makes up the image. This colour system is known as RGB. The elements of these matrices are integers between 0 and 255 (1 - when dealing with binary images) and they determine the intensity of the pixel with respect to the colour of the matrix. The main reason why these RGB matrices are significant is because of the possibility to represent $256^3 = 2^{24} = 16777216$ different colours.

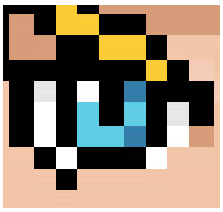


Coloured Uzumaki Naruto

In Coloured Uzumaki Naruto, we can see that the black and white image got replaced by different colours. If we zoom into the image, we can see each pixel representing an element of the matrix.



In the image above, we can now clearly see each rectangular or square box representing a matrix. Now to study a matrix more carefully, we will zoom into Naruto's right eye.



In the image above, we can see that each colour(pixel) has its own matrix combining the 3 RGB matrices (so we can say that a pixel is a matrix made up of 3 RGB matrices/panels). These RGB matrices also known as RGB matrix panels have their own set of values(elements) obtained from tri-colour LED chips, but we will be speaking about working with pixels and their RGB values in this project. If we zoom into the eye a bit more, we can see the different colours in Naruto's pupil.

In the image above, if we check the RGB values for each element or colour of the matrix, we get:

Assuming matrix to be A,

- A [1,1]: Red = 255, Green = 255, Blue = 255
- A [1,2]: Red = 0, Green = 0, Blue = 0
- A [1,3]: Red = 55, Green = 124, Blue = 170
- A [2,1]: Red = 96, Green = 207, Blue = 227
- A [2,2]: Red = 0, Green = 0, Blue = 0
- A [2,3]: Red = 95, Green = 205, Blue = 228
- A [3,1]: Red = 102, Green = 221, Blue = 246
- A [3,2]: Red = 98, Green = 212, Blue = 234
- A [3,3]: Red = 56, Green = 136, Blue = 181

We can represent matrix A as:

$$\begin{bmatrix} (255,255,255) & (0,0,0) & (55,124,170) \\ (96,207,227) & (0,0,0) & (95,205,228) \\ (102,221,246) & (98,212,234) & (56,136,181) \end{bmatrix}$$

ROTATING AN IMAGE:

1. Now if we want to rotate the image 90^{o} clockwise, we will have to apply certain operations to get there:
   First, let's look at what our image should look like after turning it clockwise:

To solve this, we have to divide it into 2 steps.

- Transposing matrix A:

$$\begin{bmatrix} (255,255,255) & (96,207,227) & (102,221,246) \\ (0,0,0) & (0,0,0) & (98,212,234) \\ (55,124,170) & (95,205,228) & (56,136,181) \end{bmatrix}$$

- Interchanging columns
  COLUMN 1 ⇔ COLUMN 3

$$\begin{bmatrix} (102,221,246) & (96,207,227) & (255,255,255) \\ (98,212,234) & (0,0,0) & (0,0,0) \\ (56,136,181) & (95,205,228) & (55,124,170) \end{bmatrix}$$

On viewing this digital image, we will now get the 90° clockwise rotated image required.

Here is how Naruto will look like after applying operations to each and every pixel

2. If we want to rotate it 90° anti-clockwise, we will follow the same steps but in the reverse order.

- Interchanging columns
  COLUMN 1 ⇔ COLUMN 3

$$\begin{bmatrix} (55,124,170) & (0,0,0) & (255,255,255) \\ (95,205,228) & (0,0,0) & (96,207,227) \\ (56,136,181) & (98,212,234) & (102,221,246) \end{bmatrix}$$

- Transposing the above matrix

$$\begin{bmatrix} (55,124,170) & (95,205,228) & (56,136,181) \\ (0,0,0) & (0,0,0) & (98,212,234) \\ (255,255,255) & (96,207,227) & (102,221,246) \end{bmatrix}$$

After these operations we get a 90° anti-clockwise image of Naruto,

3. Lastly, we have Naruto rotated 180°.
   To achieve this, we will continue from the anti-clockwise image.

- Transposing the above image.

$$\begin{bmatrix} (55,124,170) & (0,0,0) & (255,255,255) \\ (95,205,228) & (0,0,0) & (96,207,227) \\ (56,136,181) & (98,212,234) & (102,221,246) \end{bmatrix}$$

- Interchanging rows
  Row 1 ⇔ Row 3

$$\begin{bmatrix} (56,136,181) & (98,212,234) & (102,221,246) \\ (95,205,228) & (0,0,0) & (96,207,227) \\ (55,124,170) & (0,0,0) & (255,255,255) \end{bmatrix}$$

Required 180° rotated image:

Another method of rotating the image by 180° is to transpose the clockwise image, followed by interchanging Column 1 and Column 3.

To achieve a black and white filter (grayscale image) again, each individual pixel has the average of its red, green, blue values taken; then the new pixel for the black and white image has the red, green, and blue values set to that average.

MIRRORING AN IMAGE:

Images are stored in matrices, in which each element of the matrix corresponds to a single discrete pixel of the image. We can get the mirror image of the given image if we reverse the order of the pixels (elements of the matrix) in each row i.e., the rows remain the same but the columns interchange.

The image we will be mirroring this time is given below:

Although we cannot evaluate each and every pixel of the pixelated image of Naruto above, we will look at an example of COLUMNS interchanging with each other from both the sides.

127	255	220	112	255
110	90	0	190	110
0	127	100	255	120
27	45	91	127	0
255	200	190	185	140

Original Image

255	112	220	255	127
110	190	0	90	110
120	255	100	127	0
0	127	91	45	27
140	185	190	200	255

Mirror Image

The steps carried out in the above image are pretty easy. As we can see, only the values in each column interchange and not the rows. Therefore,

- INTERCHANGING COLUMNS
  1. COLUMN 1 ⇔ COLUMN 5

$$\begin{bmatrix} 255 & 255 & 220 & 112 & 127 \\ 110 & 90 & 0 & 190 & 110 \\ 120 & 127 & 100 & 255 & 0 \\ 0 & 45 & 91 & 127 & 27 \\ 140 & 200 & 190 & 185 & 255 \end{bmatrix}$$

  2. COLUMN 2 ⇔ COLUMN 4

$$\begin{bmatrix} 255 & 112 & 220 & 255 & 127 \\ 110 & 190 & 0 & 90 & 110 \\ 120 & 255 & 100 & 127 & 0 \\ 0 & 127 & 91 & 47 & 27 \\ 140 & 185 & 190 & 200 & 255 \end{bmatrix}$$

Although this example was completed in two steps, interchanging columns in Naruto's pixelated image would have a 1000 or more interchanges between its columns. Here is what Naruto's Mirrored image will look like:

The values of these pixels followed by the operations can be extracted and carried out on them respectively, with help of a digital image manipulator, which can be built using a programming language such as Java.

Once a digital image can be represented through matrices, we can play around with it by applying various operations on their elements and affect the corresponding image. One such way is by using kernels:

KERNELS:

Before looking at the role of kernels in image processing, we will look at certain terms involved in it.

1. **Kernel**: In image processing, a kernel, convolution matrix, or mask is a small matrix. It is used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image.
2. **Convolution**: It is the process of adding each element of the image to its local neighbours, weighted by the kernel. This is related to a form of mathematical convolution. The matrix operation being performed—convolution—is not traditional matrix multiplication, despite being similarly denoted by *.
3. **Origin**: In kernels, the origin is the position of the kernel which is above (conceptually) the current output pixel. This could be outside of the actual kernel, though usually it corresponds to one of the kernel elements. For a symmetric kernel, the origin is usually the centre element.

When we are convoluting an image, we require two matrices: an image matrix and a kernel matrix. Here is what a convolution operation looks like:

$$\begin{bmatrix}111 & 100 & 163\\121 & 253 & 142\\173 & 222 & 345\end{bmatrix}*\begin{bmatrix}0 & 1 & 0\\1 & 0 & 1\\0 & 1 & 0\end{bmatrix}=\begin{bmatrix}111*0 & 100*1 & 163*0\\121*1 & 253*0 & 142*1\\173*0 & 222*1 & 345*0\end{bmatrix}=\begin{bmatrix}0 & 100 & 0\\121 & 0 & 142\\0 & 222 & 0\end{bmatrix}$$

Image matrix *Kernel*

To find the final value of the required centre pixel, we will add all the elements.

Therefore, Sum = 0 + 100 + 0 + 121 + 0 + 142 + 0 + 222 + 0 = 585

In the process of convolution above, we have multiplied the kernel values times the corresponding pixel values of the image matrix, and added the result - this final value is the new value of the current pixel. This pixel value is our centre pixel.

To better understand the concept of convolution between kernels and images, we will look at the example below:

$$\begin{bmatrix}111 & 222 & 163 & 230\\225 & 32 & 110 & 215\\41 & 92 & 184 & 97\\194 & 233 & 127 & 202\end{bmatrix}*\begin{bmatrix}0 & 1 & 0\\1 & 0 & 1\\0 & 1 & 0\end{bmatrix}=\begin{bmatrix}a & b\\c & d\end{bmatrix}$$

1. To convolute the above matrix, we will first consider the top left 3x3 matrix of the 4x4 matrix and find the value of its centre pixel.

$$\begin{bmatrix}111 & 222 & 163\\225 & 32 & 110\\41 & 92 & 184\end{bmatrix}*\begin{bmatrix}0 & 1 & 0\\1 & 0 & 1\\0 & 1 & 0\end{bmatrix}=\begin{bmatrix}110*0 & 222*1 & 163*0\\225*1 & 32*0 & 110*1\\41*0 & 92*1 & 184*0\end{bmatrix}=\begin{bmatrix}0 & 222 & 0\\225 & 0 & 110\\0 & 92 & 0\end{bmatrix}$$

The sum of the elements = 0+222+0+225+0+110+0+92+0 = 649

$$\begin{bmatrix}649 & b\\c & d\end{bmatrix}$$

2. We will now shift one column towards the right to calculate the next centre value of the 3x3 matrix.

$$\begin{bmatrix}222 & 163 & 230\\32 & 110 & 215\\92 & 184 & 97\end{bmatrix}*\begin{bmatrix}0 & 1 & 0\\1 & 0 & 1\\0 & 1 & 0\end{bmatrix}=\begin{bmatrix}222*0 & 163*1 & 230*0\\32*1 & 110*0 & 215*1\\92*0 & 184*1 & 97*0\end{bmatrix}=\begin{bmatrix}0 & 163 & 0\\32 & 0 & 215\\0 & 184 & 0\end{bmatrix}$$

The sum of the elements = 163+32+215+184+0+0+0+0 = 594

$$\begin{bmatrix}649 & 594\\c & d\end{bmatrix}$$

3. We will now shift back to the position in step 1 and shift one row down to calculate the next centre value of the 3x3 matrix.

$$\begin{bmatrix}225 & 32 & 110\\ 41 & 92 & 184\\ 194 & 233 & 127\end{bmatrix} * \begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix} = \begin{bmatrix}225*0 & 32*1 & 110*0\\ 41*1 & 92*0 & 184*1\\ 194*0 & 233*1 & 127*0\end{bmatrix} = \begin{bmatrix}0 & 32 & 0\\ 41 & 0 & 184\\ 0 & 233 & 0\end{bmatrix}$$

The sum of the elements = 0+0+0+0+32+41+233+184 = 490

$$\begin{bmatrix}649 & 594\\ 490 & d\end{bmatrix}$$

4. Finally, we will shift one column to the right (the bottom right 3x3) matrix to calculate its centre pixel value.

$$\begin{bmatrix}32 & 110 & 215\\ 92 & 184 & 97\\ 233 & 127 & 202\end{bmatrix} * \begin{bmatrix}0 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 0\end{bmatrix} = \begin{bmatrix}32*0 & 110*1 & 215*0\\ 92*1 & 184*0 & 97*1\\ 233*0 & 127*1 & 202*0\end{bmatrix} = \begin{bmatrix}0 & 110 & 0\\ 92 & 0 & 97\\ 0 & 127 & 0\end{bmatrix}$$

The sum of all the elements = 0+0+0+0+110+92+97+127 = 426

$$\begin{bmatrix}649 & 594\\ 490 & 426\end{bmatrix}$$

Finally, we have got our convoluted matrix above. Although the values of pixels lie between 0 and 255, we can scale it down to that range before displaying the output. This is done through the process of normalization (Though this topic is out of the project's scope, it is an important concept in programming). Now these convolutions can be applied to different types of kernels which affect our image. Blurring, edge detection, and sharpening all rely on kernels - small matrices of numbers - to be applied across an image in order to process the image as a whole. A kernel works by operating on these pixel values using straightforward mathematics to construct a new image. These kernels can also be called "filters" and form the basis of convolution operations which affect our image in different ways. The following are different types of kernels that we can use to enhance the image:

1. **BLUR**

$$* \begin{bmatrix}0.111 & 0{,}111 & 0.111\\ 0.111 & 0.111 & 0.111\\ 0.111 & 0{,}111 & 0.111\end{bmatrix} =$$

In the above image, the convolution starts from the top right corner of the image i.e., the 3x3 matrix and shifts according to give us the required blurred image.

2. **SHARPEN**

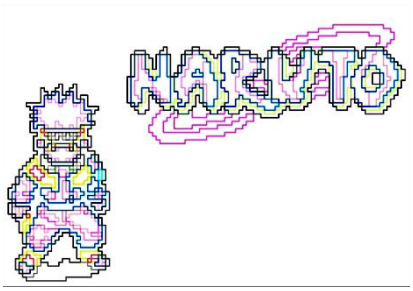
$$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} =$$

3. **SEPIA EFFECT**

$$* \begin{bmatrix} 0.272 & 0.534 & 0.131 \\ 0.349 & 0.686 & 0.168 \\ 0.393 & 0.769 & 0.189 \end{bmatrix} =$$

4. **SOBEL-FELDMAN EDGE DETECTION**

$$* \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} =$$



In the above image, the first kernel represents the Sobel detection filter along x-axis of the image and the second kernel represents the same along the y-axis. The image is individually convoluted with each matrix. Thus, the combined effect (sum of both X and Y outputs) of both the kernels gives us the resultant image.

IMAGE ENCRYPTION(CRYPTOGRAPHY):

Although we can digitally process images, we need a safe means to store them. In order to do so we use matrices. One of the most famous ciphering(encryption) techniques is the Hill Cypher which dates back to 1929. Lester Hill invented the Hill Cypher in 1929. At the time, the cypher was one of the first to be able to operate on more than three letters or symbols during a single encryption or decryption operation. However, as the number of symbols increases, the arithmetic required to perform the encryption becomes more and more difficult. Let us understand it using an example:

Let's say we want to encrypt a word called "NARUTO" in a message. To do the following, we will look at how each letter is numbered in the Hill Cypher.

Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Number	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

In the table above, each letter is represented by a number modulo 26.

Let us write NARUTO in the matrix form:

But to do this, we will divide NARUTO into two parts since the Hill Cypher can only manage a 3x3 invertible matrix easily. The divided parts will be NAR and UTO. They can be represented through matrices in the following way:

$$\begin{bmatrix} N \\ A \\ R \end{bmatrix} = \begin{bmatrix} 13 \\ 0 \\ 17 \end{bmatrix} \qquad \begin{bmatrix} U \\ T \\ O \end{bmatrix} = \begin{bmatrix} 20 \\ 19 \\ 14 \end{bmatrix}$$

If we want to encrypt a message, each block of N letters is multiplied with an invertible matrix (nxn) or also known as a cipher key, against modulus 26.

Although ciphering already has ready made invertible matrices, we will see how to check if a matrix is invertible or not. Let us say we have a matrix which is going to be our cipher key by which we are going to multiply NAR and UTO. To check if the matrix is invertible or not, we will follow certain steps listed below:

A. FOR ENCRYPTION

1. **WE WILL CHECK WHETHER A MATRIX IS INVERTIBLE OR NOT.**
2. **FINDING THE DETERMINANT. IF THE DETERMINANT ≠ 0 [NON-SINGULAR MATRIX], THEN OUR MATRIX IS INVERTIBLE.**

Let us consider the matrix A given below:

$$\begin{bmatrix} 5 & 0 & 4 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$|A| = \begin{vmatrix} 5 & 0 & 4 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{vmatrix}$$

= 5(3-4) + 0(2-2) + 4(4-3)

= 5(-1) + 0 + 4

= -1 **≠ 0 [NON-SINGULAR MATRIX]**

Therefore, A^{-1} exists.

Now we will encrypt our message by multiplying our cipher key (Matrix A) with NAR and UTO.

$$\begin{bmatrix}5 & 0 & 4\\ 2 & 3 & 2\\ 1 & 2 & 1\end{bmatrix} * \begin{bmatrix}13\\ 0\\ 17\end{bmatrix} = \begin{bmatrix}133\\ 60\\ 30\end{bmatrix} = \begin{bmatrix}3\\ 8\\ 4\end{bmatrix} \text{(mod 26)} \qquad \begin{bmatrix}5 & 0 & 4\\ 2 & 3 & 2\\ 1 & 2 & 1\end{bmatrix} * \begin{bmatrix}20\\ 19\\ 14\end{bmatrix} = \begin{bmatrix}156\\ 125\\ 72\end{bmatrix} = \begin{bmatrix}0\\ 21\\ 20\end{bmatrix} \text{(mod 26)}$$

The two cipher texts that we get are DIG and AVQ.

When the decoder will be decrypting this message, we will use the inverse of matrix A (A^{-1}) and simply multiply it to the two cipher texts DIG and AVQ.

But first we will find the inverse of matrix A:

We know that the matrix is invertible, so we will:

1. Find the Cofactor of all elements to get the Cofactor matrix.
2. Transpose the Cofactor matrix to get the Adjoint matrix.
3. Multiply the reciprocal of determinant A with the Adjoint matrix.

Matrix A: $\begin{bmatrix}5 & 0 & 4\\ 2 & 3 & 2\\ 1 & 2 & 1\end{bmatrix}$

Cofactor of all elements:

A_{11}= -1 A_{21}= 0 A_{31}= +1

A_{12}= +8 A_{22}= +1 A_{32}= -10

A_{13}= -12 A_{23}= -2 A_{33}= +15

Therefore, Cofactor matrix = $\begin{bmatrix}-1 & 0 & 1\\ 8 & 1 & -10\\ -12 & -2 & 15\end{bmatrix}$

Adjoint matrix = $\begin{bmatrix}-1 & 0 & 1\\ 8 & 1 & -10\\ -12 & -2 & 15\end{bmatrix}^T = \begin{bmatrix}-1 & 8 & -12\\ 0 & 1 & -2\\ 1 & -10 & 15\end{bmatrix}$

$A^{-1} = \frac{1}{|A|} *$ Adjoint matrix

$= \frac{1}{-1} * \begin{bmatrix}-1 & 8 & -12\\ 0 & 1 & -2\\ 1 & -10 & 15\end{bmatrix}$

$= -1 * \begin{bmatrix}-1 & 8 & -12\\ 0 & 1 & -2\\ 1 & -10 & 15\end{bmatrix} = \begin{bmatrix}1 & -8 & 12\\ 0 & -1 & 2\\ -1 & 10 & -15\end{bmatrix}$

$$A^{-1} = \begin{bmatrix} 1 & -8 & 12 \\ 0 & -1 & 2 \\ -1 & 10 & -15 \end{bmatrix} = \begin{bmatrix} 1 & 18 & 12 \\ 0 & 25 & 2 \\ 25 & 10 & 11 \end{bmatrix} \text{(mod 26)}$$

Now that we have got inverse of A, we can easily decipher our text by multiplying inverse of A with the two cipher texts DIG and AVQ.

B. DECIPHERING

1. MULTIPLYING DIG WITH A^{-1} AND AVQ WITH A^{-1}.

$$\begin{bmatrix} 1 & 18 & 12 \\ 0 & 25 & 2 \\ 25 & 10 & 11 \end{bmatrix} * \begin{bmatrix} 3 \\ 8 \\ 4 \end{bmatrix} = \begin{bmatrix} 195 \\ 208 \\ 199 \end{bmatrix} = \begin{bmatrix} 13 \\ 0 \\ 17 \end{bmatrix} \text{(mod 26) = NAR}$$

$$\begin{bmatrix} 1 & 18 & 12 \\ 0 & 25 & 2 \\ 25 & 10 & 11 \end{bmatrix} * \begin{bmatrix} 0 \\ 21 \\ 20 \end{bmatrix} = \begin{bmatrix} 618 \\ 565 \\ 430 \end{bmatrix} = \begin{bmatrix} 20 \\ 19 \\ 14 \end{bmatrix} (\text{mod } 26) = \text{UTO}$$

Finally, we have decrypted the message to give us NAR + UTO = NARUTO.

In the same way, we can use pixel values of an image to encrypt the pixel values using an invertible matrix and decrypt it using the inverse of the invertible matrix. While matrix multiplication alone does not result in a secure cipher it is still a useful step when combined with other non-linear operations, because matrix multiplication can provide diffusion.

OTHER APPLICATIONS/USES

1. Camera app on mobile phones
2. Instagram filters, Snapchat masks
3. Face recognition
4. Object recognition
5. Computer vision
6. Quantum computing with image processing using matrices
7. Solving the Rubik's Cube.

CONCLUSION

The representation of images using matrices has created a huge world of possibilities in digital image processing. The ones listed here are just a few important examples to demonstrate this. There are multitude more of kernels that can be applied, as well as applications of constants and changing of pixel positions (skewing,3D-tranformations etc.). Matrix manipulation has made a world of difference in image processing, making it a really interesting topic to explore.

BIBLIOGRAPHY

1. http://blog.kleinproject.org/?p=588#:~:text=Once%20a%20digital%20image%20can,turn%2C%20corresponds%20to%20the%20matrix%20.
2. https://www.naturefocused.com/articles/photography-image-processing-kernel.html
3. https://developer.apple.com/documentation/accelerate/blurring_an_image
4. https://en.wikipedia.org/wiki/Hill_cipher#:~:text=In%20classical%20cryptography%2C%20the%20Hill,than%20three%20symbols%20at%20once.
5. http://www.math.utah.edu/~gustafso/s2017/2270/projects-2017/joePuglianoBrandonSehestedt/LinearAlgebra_Project.pdf