PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

MAY 2020: IN SEMESTER ASSESSMENT (ISA) B.TECH. IV SEMESTER

UE18MA251- LINEAR ALGEBRA

MINI PROJECT REPORT

ON

IMAGE PROCESSING

Submitted by

1. Name: Ajitesh Nair SRN: PES2201800681
2. Name: Ishan Padhy SRN: PES2201800158

Branch & Section : CSE , B SECTION

PROJECT EVALUATION

(For Official Use Only)

Sl.No.	Parameter	Max Marks	Marks Awarded
1	Background & Framing of the problem	4	
2	Approach and Solution	4	
3	References	4	
4	Clarity of the concepts & Creativity	4	
5	Choice of examples and understanding of the topic	4	
6	Presentation of the work	5	
	Total	25	

Name of the Course Instructor :

Signature of the Course Instructor :

INTRODUCTION

Image processing is an interesting application of linear algebra. Every image can be represented in the form of a matrix. This matrix when multiplied with transformation matrices can help to transform an image .Each transformation has its own transformation matrix which when multiplied with the original matrix transforms it into a new transformed matrix. There are various types of transformations like rotation,reflection (or flipping),shearing,stretching,flipping etc. With the help of computers and algorithms an image can be converted into a matrix and then transformed by multiplying with the corresponding transformation matrix. We have implemented a program using C++ to perform these transformations.

REVIEW OF LITERATURE

Most common geometric transformations that keep the origin fixed are linear, including rotation, scaling, shearing, reflection, and orthogonal projection; if an affine transformation is not a pure translation it keeps some point fixed, and that point can be chosen as origin to make the transformation linear. In two dimensions, linear transformations can be represented using a 2×2 transformation matrix.

Stretching

A stretch in the xy-plane is a linear transformation which enlarges all distances in a particular direction by a constant factor but does not affect distances in the perpendicular direction. We only consider stretches along the x-axis and y-axis. A stretch along the x-axis has the form x' = kx; y' = y for some positive constant k.

The matrix associated with a stretch by a factor k along the x-axis is given by:

$$\begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix}$$

Similarly, a stretch by a factor k along the y-axis has the form x' = x; y' = ky, so the matrix associated with this transformation is

$$\begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}$$

Squeezing

If the two stretches above are combined with reciprocal values, then the transformation matrix represents a squeeze mapping:

$$\begin{bmatrix} k & 0 \\ 0 & 1/k \end{bmatrix}.$$

Rotation

For rotation by an angle θ **clockwise** about the origin, the matrix form is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shearing

For shear mapping (visually similar to slanting), there are two possibilities.
A shear parallel to the *x* axis has the matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

A shear parallel to the *y* axis has the matrix form:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Flipping

For flipping a matrix about x axis the transformation matrix is

1 0

0 -1

About y axis :

-1 0

0 1

About the line y=x:

0 1

1 0

REPORT

A C++ program has been written for the implementation of the project.The main function of the program is as follows

```
int main()

{

    //size for matrix transformation matrix

    const int m = 2;
```

```
//matrix for transofrmations
typedef double MATRIX3X3[m][m];
//choice for operation
int choice;
//choose whether x axis stretch or y axis stretch
int XorYStretch;


cv::Mat img = cv::imread("file.jpg");
int Pixels[10000][10000];
int PixelsTransformed[10000][10000];


//size of image matrix
int ImageRow = img.rows;
int ImageColumn = img.cols;
for(int i=0;i<ImageColumn;i++)
  {
    for(int j=0;j<ImageRow;j++)
      {
        Pixels[i][j]=img.at<int>(i,j);
      }
  }




// Call functions to be performed
cout << "Please choose a number to perform the operation desired"<< endl;
cout << "1 for Stretching. 2 for Rotating. 3 for Squeezing. 4 for Shearing. 5 for flipping"
<< endl;
cin >> choice;
```

```
    switch (choice)

    {

        case 1:

            cout << "Please enter whether it is a Stretch along x axis(Press 1) or y axis (Press 2)"
<< endl;

            cin >> XorYStretch;

            void Stretch(&m, &p, XorYStretch);

            break;

        case 2:

            cout << "Please enter degrees for rotation" << endl;

            cin >> degrees;

            void Rotate(&m, &p, degrees);

            break;

        case 3:

            cout << "Please enter Squeeze Factor" << endl;

            cin >> SqFactor;

            void Squeeze(&m,&p, SqFactor);

            break;

        case 4:

            cout << "Please enter whether it is a Shear along x axis(Press 1) or y axis (Press 2)"
<< endl;

            cin >> XorYShear;



            void Shearing(&m, &p, XorYShear);

            break;

        case 5:

            cout << "Please enter whether it is a Stretch along x axis(Press 1) or y axis (Press 2)
or along y = x (press 3)" << endl;

            cin >> XorYorLflip;

            void Flip(&m, &p, XorYorLflip)

            break;
```

```
        default:
            // choice is doesn't match any case
            cout << "Error! Number is not correct";
            break;
    }

    // OUTPUT FINAL PICTURE
    return 0;
}
void CopyMatrix(Pixels& src, PixelsTransformed& dst, int ImageRow, int ImageCol)
{
  for (int i=0; i<ImageRow; ++i)
   for (int j=0; j<ImageCol; ++j)
     dst[i][j] = src[i][j];
}

//using multiplication from swift library
void MultMatrix(MATRIX3X3& product,MATRIX3X3& matrix1, Pixels& matrix2)
{
  for (int x=0; x<3; ++x)
    for (int y=0; y<3; ++y)
    {
      double sum = 0;
      for (int z=0; z<3; ++z)
        sum += matrix1[x][z] * matrix2[z][y];
      product[x][y] = sum;
    }
}
```

These are the main function,the copy matrix function and multiply matrix function.

The functions implemented for each of the transformations are as follows:

Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

```
void Rotate(MATRIX3X3& m, Pixels& p, int degrees)
{
  MATRIX3X3 m1;
  Pixels p1;
  if (degrees == 0) return;
  double radians = 6.283185308 / (360.0 / degrees);
  double c = cos(radians);
  double s = sin(radians);
    m1[0][0]=c; m1[0][1]=s;
  m1[1][0]=-s; m1[1][1]=c;
 MultMatrix(p1, m1, p);
  CopyMatrix(p, p1);
}
```

Stretching

$$\begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}$$

```
void Stretch(MATRIX3X3& m1, Pixels& p, int XorYStretch)
{
  MATRIX3X3 m1;
  Pixels p1;
  int xStretch = 100;
  int yStretch = 100;

  if (XorYStretch == 1)
  {
     m1[0][0]=xStretch; m1[0][1]=0;
```

```
    m1[1][0]=0;     m1[1][1]=1;
  }
   else if (XorYStretch == 2)
   {
      m1[0][0]=1;    m1[0][1]=0;
      m1[1][0]=0;    m1[1][1]=yStretch;
   }

  MultMatrix(p1, m1, p);
  CopyMatrix(p, p1);
}
```

Squeezing

$$\begin{bmatrix} k & 0 \\ 0 & 1/k \end{bmatrix}.$$

```
void Squeeze(MATRIX3X3& m, Pixels& p, int SqFactor)
{
  MATRIX3X3 m1;
  Pixels p1;

  m1[0][0]=SqFactor;    m1[0][1]=0;
  m1[1][0]=0;           m1[1][1]=(1/SqFactor);

  MultMatrix(p1, m1, p);
  CopyMatrix(p, p1);
}
```

Shearing

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

```
void Shearing(MATRIX3X3& m, Pixels& p, int XorYShear)
{
  MATRIX3X3 m1;
  Pixels p1;
  int ShearFactor = 100;
```

```
  // parallel to x axis
  if (XorYShear == 1)
  {
     m1[0][0]=1;     m1[0][1]=ShearFactor;
     m1[1][0]=0;     m1[1][1]=1;
  }
   // parallel to y axis
    else if (XorYShear == 2)
    {
       m1[0][0]=1;            m1[0][1]=0;
       m1[1][0]=ShearFactor;    m1[1][1]=1;
    }

  MultMatrix(p1, m1, p);
  CopyMatrix(p, p1);
}
```

Flipping

```
void Flip(MATRIX3X3& m, Pixels& p, int XorYorLflip)
{
  MATRIX3X3 m1;
  Pixels p1;
  // x axis
  if (XorYorLflip == 1)
  {
     m1[0][0]=1;     m1[0][1]= 0;
     m1[1][0]=0;     m1[1][1]= -1;
  }
   // y axis
    else if (XorYorLflip == 2)
    {
       m1[0][0]= -1;    m1[0][1]=0;
       m1[1][0]= 0 ;     m1[1][1]=1;
    }
     // along y  = x
      else if (XorYorLflip == 3)
      {
```

```
        m1[0][0]=0;     m1[0][1]= 1;
        m1[1][0]= 1;    m1[1][1]= 0;
      }

  MultMatrix(p1, m1, p);
  CopyMatrix(p, p1);
}
```

RESULTS AND SUMMARY

The image was fed into the program and the following images were obtained as output.

We were able to successfully perform the above transformations on the image by multiplying with the corresponding transformation matrices.

BIBLIOGRAPHY

1.Wikipedia(https://en.wikipedia.org/wiki/Transformation_matrix#Examples_in_2_dimensions)

2.Gentle, James E. (2007). "Matrix Transformations and Factorizations". *Matrix Algebra: Theory, Computations, and Applications in Statistics*. Springer. ISBN 9780387708737.

3.Lang, Serge (1987), Linear Algebra (Third ed.), New York: Springer-Verlag, ISBN 0-387-96412-6

4.Research Paper “Transformation of an Object in Computer Graphics: A Case Study of Mathematical Matrix Theory” by Manoj Kumar Srivastav (https://www.researchgate.net/publication/309741717_Transformation_of_an_Object_in_Computer_Graphics_A_Case_Study_of_Mathematical_Matrix_Theory)

5.Research Paper “IMAGE AND ITS MATRIX, MATRIX AND ITS IMAGE” by Vesna Vučković

(Faculty of Mathematics, Belgrade)

(http://elib.mi.sanu.ac.rs/files/journals/ncd/12/ncd12017.pdf)