IMAGE PROCESSING

(i) Translation

ALGORITHM:

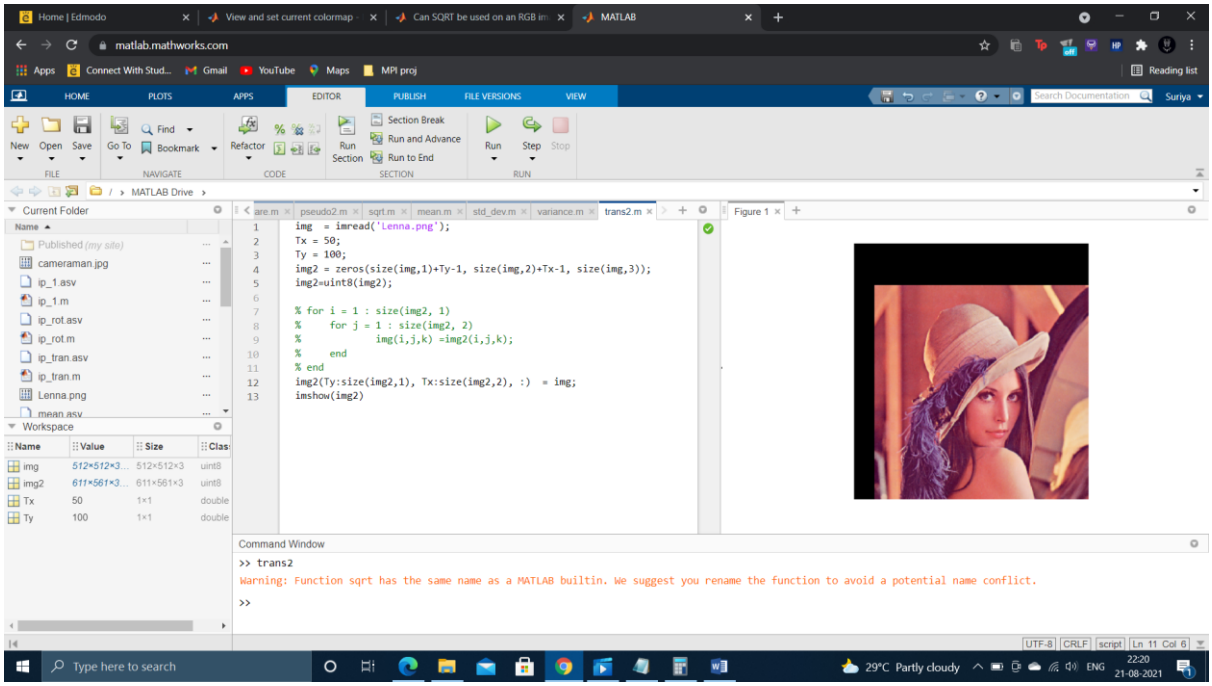
The centre of the image is repositioned based on the given Tx and Ty values

- A zero matrix with size of rows and columns equal to Tx(shift in x- coord) and Ty(shift in Y-cord) is declared
- The original image matrix is looped from Tx and Ty to the last pixel and the respective pixel values are stored to the new matrix, img2

CODE:

```
img  = imread('Lenna.png');
Tx = 50;
Ty = 100;
img2 = zeros(size(img,1)+Ty-1, size(img,2)+Tx-1, size(img,3));
img2=uint8(img2);

% for i = 1 : size(img2, 1)
%     for j = 1 : size(img2, 2)
%             img(i,j,:) =img2(i,j,:);
%     end
% end
img2(Ty:size(img2,1), Tx:size(img2,2), :)  = img;
imshow(img2)
```

OUTPUT:



(ii) Rotation

ALGORITHM:

The original image is rotated by an angle theta taking midpoint as reference point

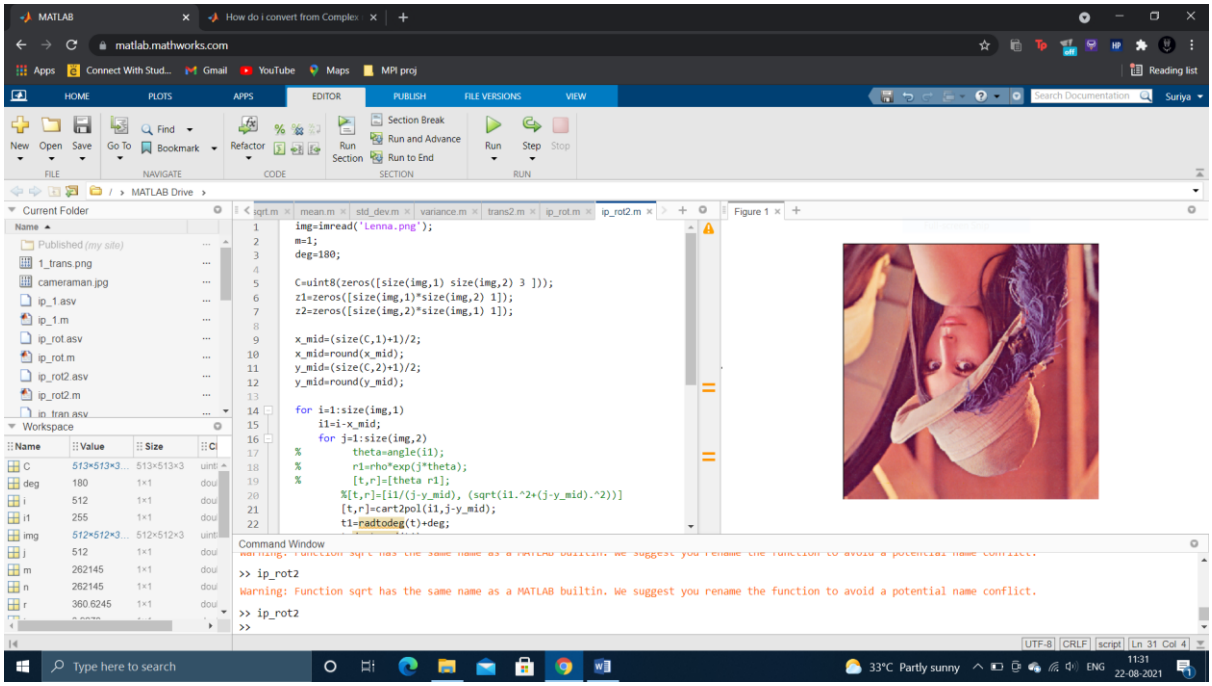
- Each pixel coordinate is converted into its corresponding polar coordinates to obtain the angle (in radians) and radius
- The angle theta is then converted into degrees and the rotation angle is added
- The angle is converted back to radians to obtain the polar coordinates of the resultant rotation matrix
- These matrices are checked to make sure than all their elements lie within the range of the image size

CODE:

```
img=imread('Lenna.png');
```

```
m=1;
deg=180;
slices=3;
C=uint8(zeros([size(img,1) size(img,2) slices ]));
z1=zeros([size(img,1)*size(img,2) 1]);
z2=zeros([size(img,2)*size(img,1) 1]);

x_mid=(size(C,1)+1)/2;
x_mid=round(x_mid);
y_mid=(size(C,2)+1)/2;
y_mid=round(y_mid);

for i=1:size(img,1)
    i1=i-x_mid;
    for j=1:size(img,2)
%         theta=angle(i1);
%         r1=rho*exp(j*theta);
%         [t,r]=[theta r1];
        %[t,r]=[i1/(j-y_mid), (sqrt(i1.^2+(j-y_mid).^2))]
        [t,r]=cart2pol(i1,j-y_mid);
        t1=radtodeg(t)+deg;
        t=degtorad(t1);
        [x,y]=pol2cart(t,r);
        z1(m)=round(x+x_mid);
        z2(m)=round(y+y_mid);
        m=m+1;

    end

end

z1(find(z1 < 1))=1;
z2(find(z2 < 1))=1;

end
n=1;
for i=1:size(img,1)
    for j=1:size(img,2)
        C(z1(n),z2(n),:)=img(i,j,:);
        n=n+1;
    end

end
imshow(C);
```

OUTPUT:



(iii) Scaling

ALGORITHM:

Scaling involves enlarging or shrinking the original image.

Here the pixels of the original image are looped through and every alternate pixel is skipped (i.e, skip count of 2) and the image produced from the resultant matrix is displayed

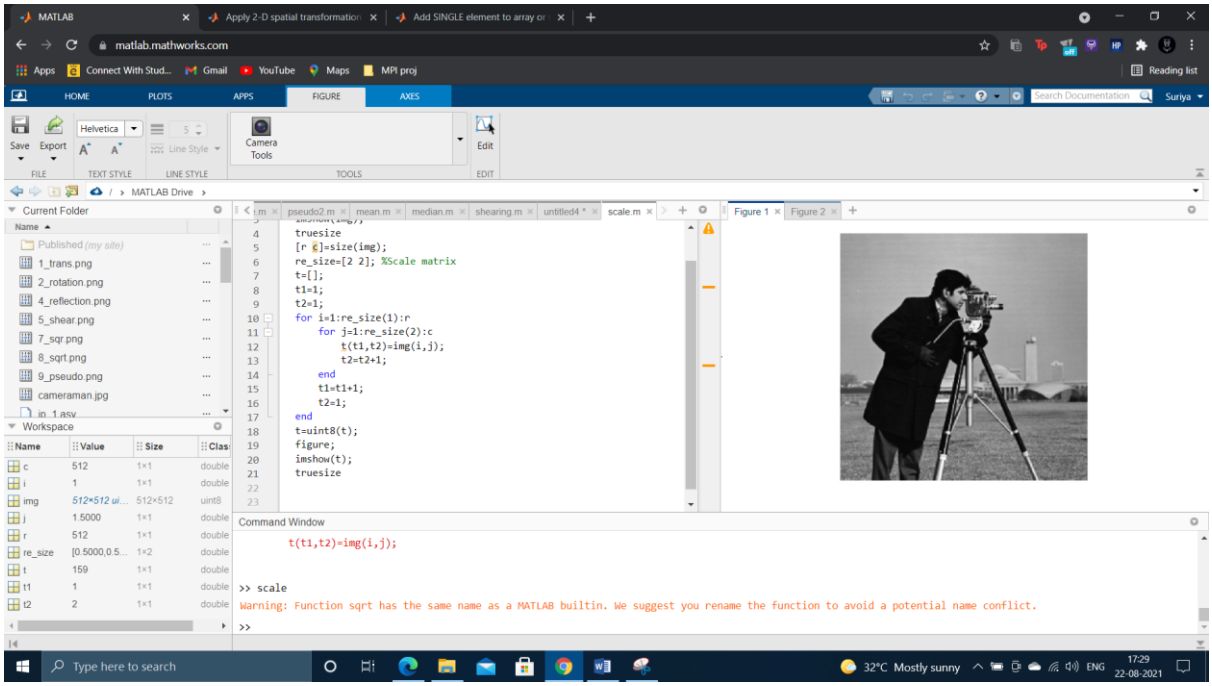
CODE:

```
img=imread('cameraman.jpg');
figure;
imshow(img);
truesize
[r c]=size(img);
re_size=[2 2]; %Scale matrix
t=[];
t1=1;
t2=1;
for i=1:re_size(1):r
    for j=1:re_size(2):c
        t(t1,t2)=img(i,j);
        t2=t2+1;
    end
    t1=t1+1;
    t2=1;
end
```

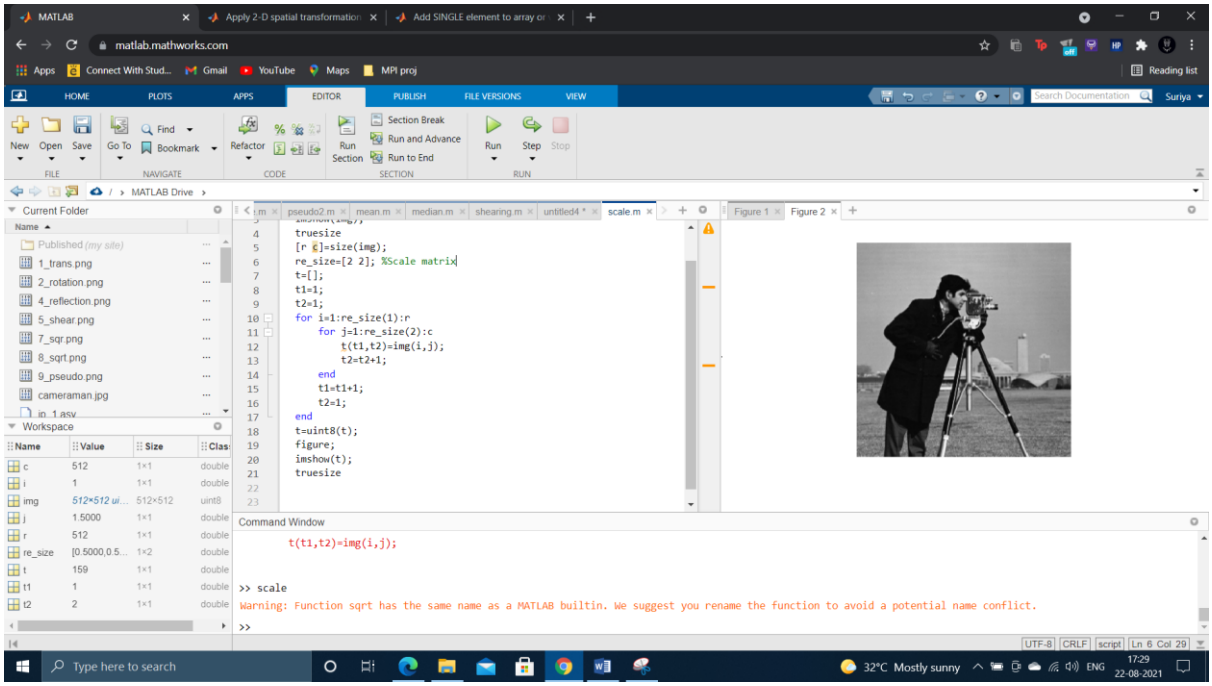
```
t=uint8(t);
figure;
imshow(t);
truesize
```

OUTPUT:

Unscaled



Scaled down



(iv) Reflection

ALGORITHM:

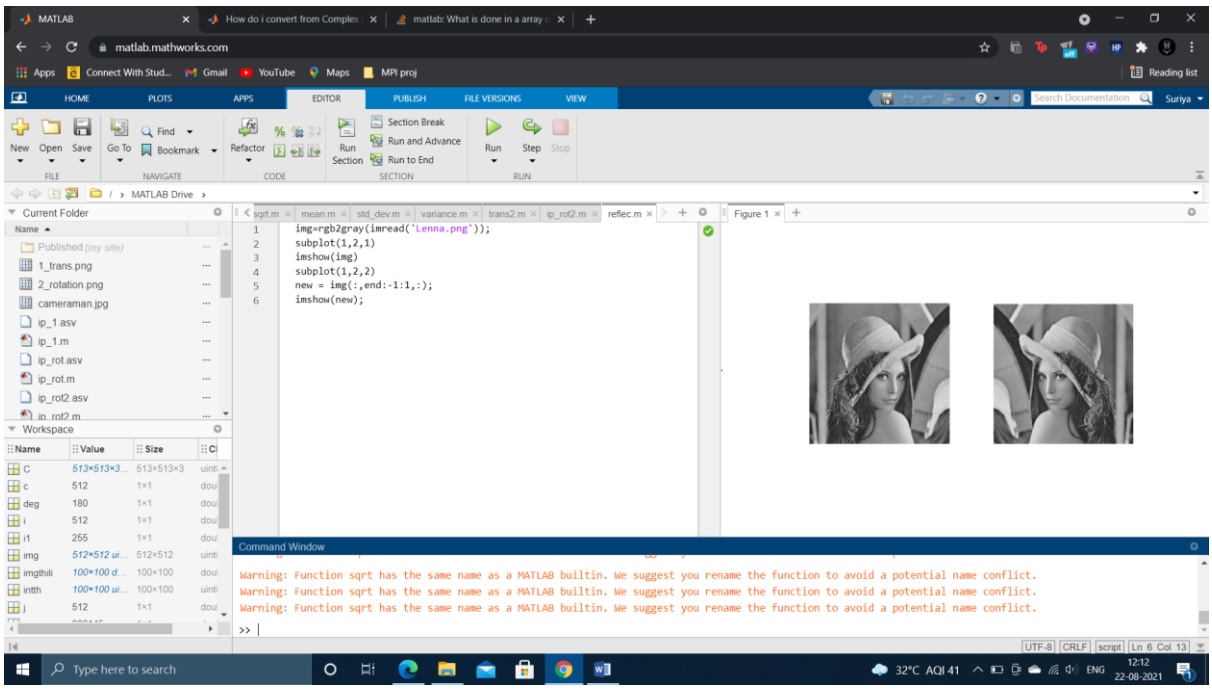
Reflection of an image involves flipping an image with respect to the Y-axis

- The rows and columns of submatrix of the original image is assigned to new, reversing only the column's order by -1 steps

CODE:

```
img=rgb2gray(imread('Lenna.png'));
subplot(1,2,1)
imshow(img)
subplot(1,2,2)
new = img(:,end:-1:1,:);
imshow(new);
```

OUTPUT:



(v) Shearing

ALGORITHM:

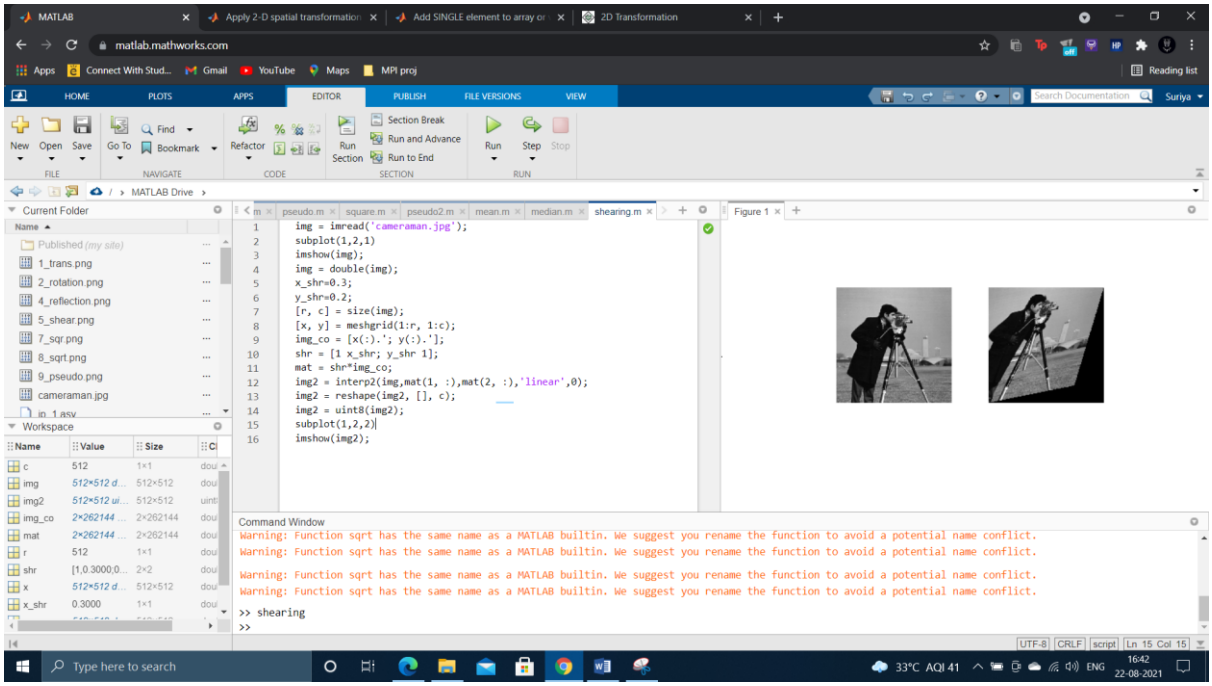
Shearing alters the shape of the image wrt the x or y axes, showing a drag-like appearance

- A shear matrix with the desired xshear and yshear is created. The image matrix is multiplied to the shear matrix
- Linear interpolation is done to obtain the image values for the resultant x and y coordinates

CODE:

```
img = imread('cameraman.jpg');
subplot(1,2,1)
imshow(img);
img = double(img);
x_shr=0.3;
y_shr=0.2;
[r, c] = size(img);
[x, y] = meshgrid(1:r, 1:c);
img_co = [x(:).'; y(:).'];
shr = [1 x_shr; y_shr 1];
mat = shr*img_co;
img2 = interp2(img,mat(1, :),mat(2, :),'linear',0);
img2 = reshape(img2, [], c);
img2 = uint8(img2);
subplot(1,2,2)
imshow(img2);
```

OUTPUT:



(vi) Eigen Value

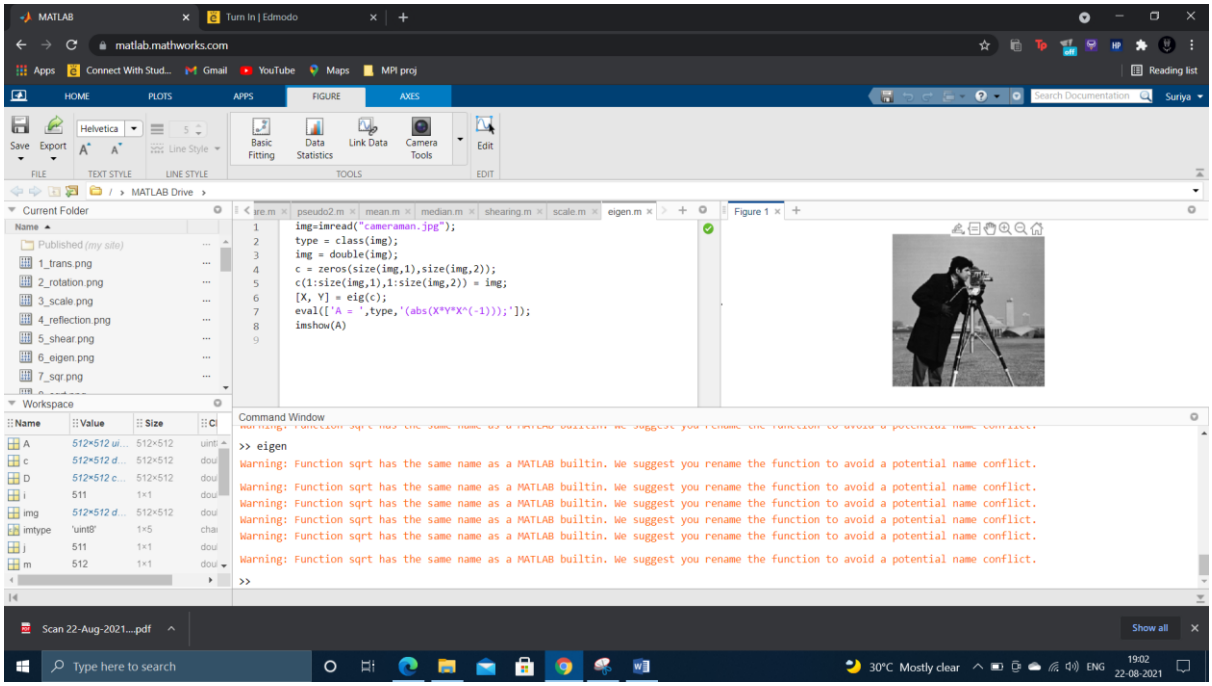
ALGORITHM:

Eigen value of an image determines the correlation between different features of the same image

CODE:

```
img=imread("cameraman.jpg");
type = class(img);
img = double(img);
c = zeros(size(img,1),size(img,2));
c(1:size(img,1),1:size(img,2)) = img;
[X, Y] = eig(c);
eval(['A = ',type,'(abs(X*Y*X^(-1)));']);
imshow(A)
```

OUTPUT:



(vii) Square of an Image

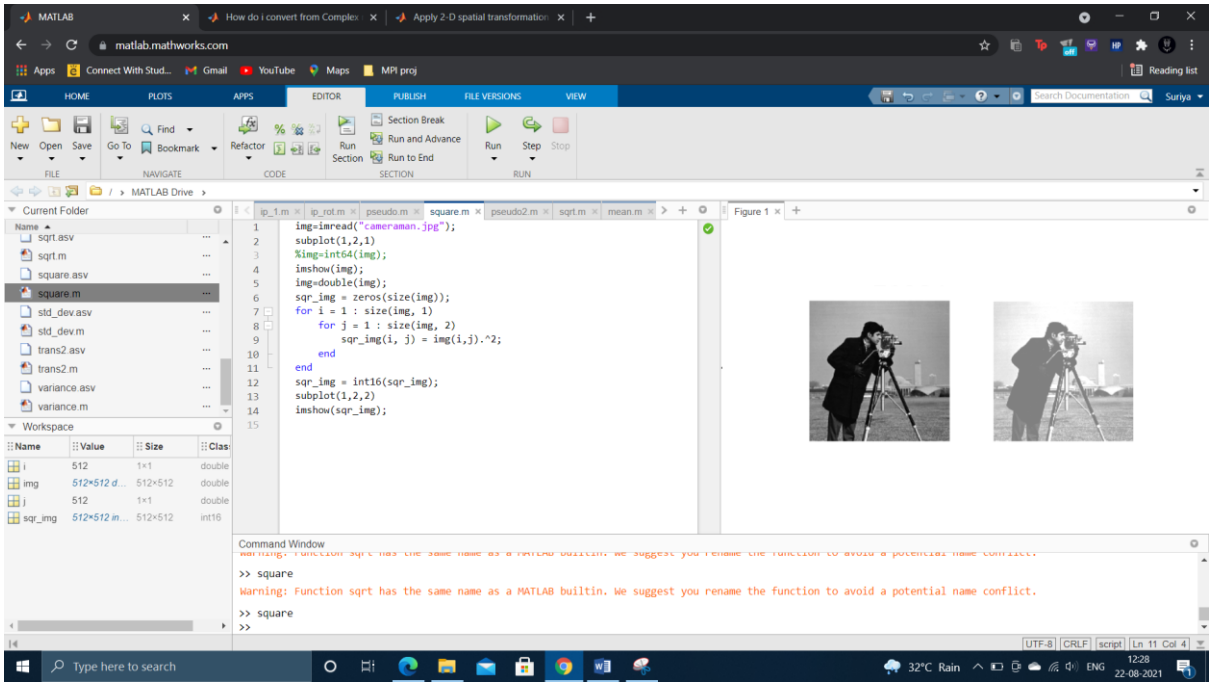
ALGORITHM:

The original image is looped through, changing each pixel value to its corresponding square

CODE:

```
img=imread("cameraman.jpg");
subplot(1,2,1)
%img=int64(img);
imshow(img);
img=double(img);
sqr_img = zeros(size(img));
for i = 1 : size(img, 1)
    for j = 1 : size(img, 2)
        sqr_img(i, j) = img(i,j).^2;
    end
end
sqr_img = int16(sqr_img);
subplot(1,2,2)
imshow(sqr_img);
```

OUTPUT:



(viii) Square root of an Image

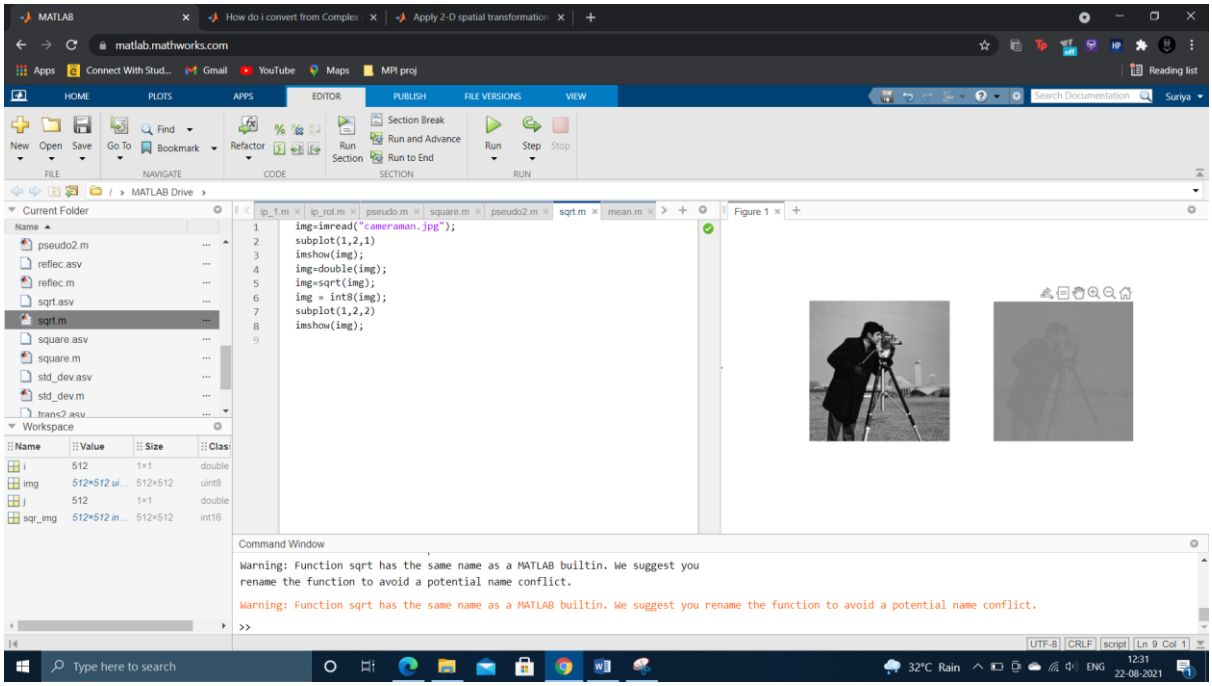
ALGORITHM:

The original image is looped through, changing each pixel value to its corresponding square root

CODE:

```
img=imread("cameraman.jpg");
subplot(1,2,1)
imshow(img);
img=double(img);
img=sqrt(img);
img = int8(img);
subplot(1,2,2)
imshow(img);
```

OUTPUT:



(ix) Grayscale to Color (Pseudocolor)

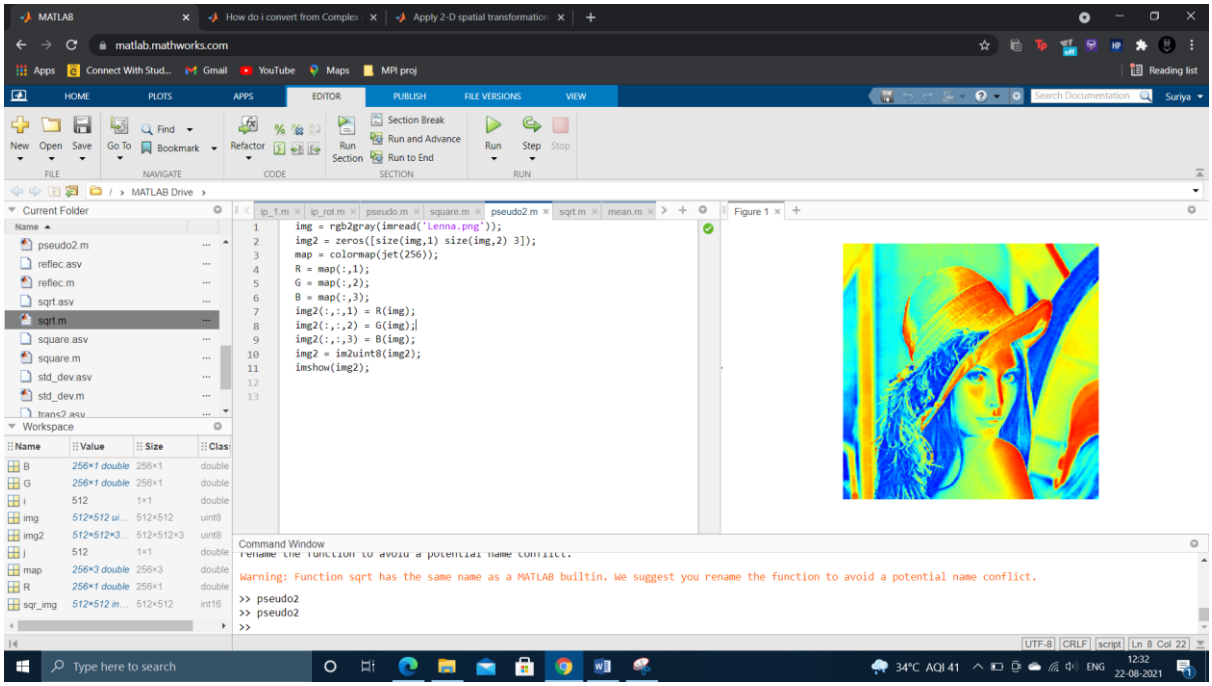
ALGORITHM:

In this transformation, the pixels of the grayscale image are mapped to the respective R,G,B values on the color map depending on their intensities

CODE:

```
img = rgb2gray(imread('Lenna.png'));
img2 = zeros([size(img,1) size(img,2) 3]);
map = colormap(jet(256));
R = map(:,1);
G = map(:,2);
B = map(:,3);
img2(:,:,1) = R(img);
img2(:,:,2) = G(img);
img2(:,:,3) = B(img);
img2 = im2uint8(img2);
imshow(img2);
```

OUTPUT:



(x) Bi-cubic Interpolation

ALGORITHM:

CODE:

OUTPUT:

(xi) Mean of an Image

ALGORITHM:

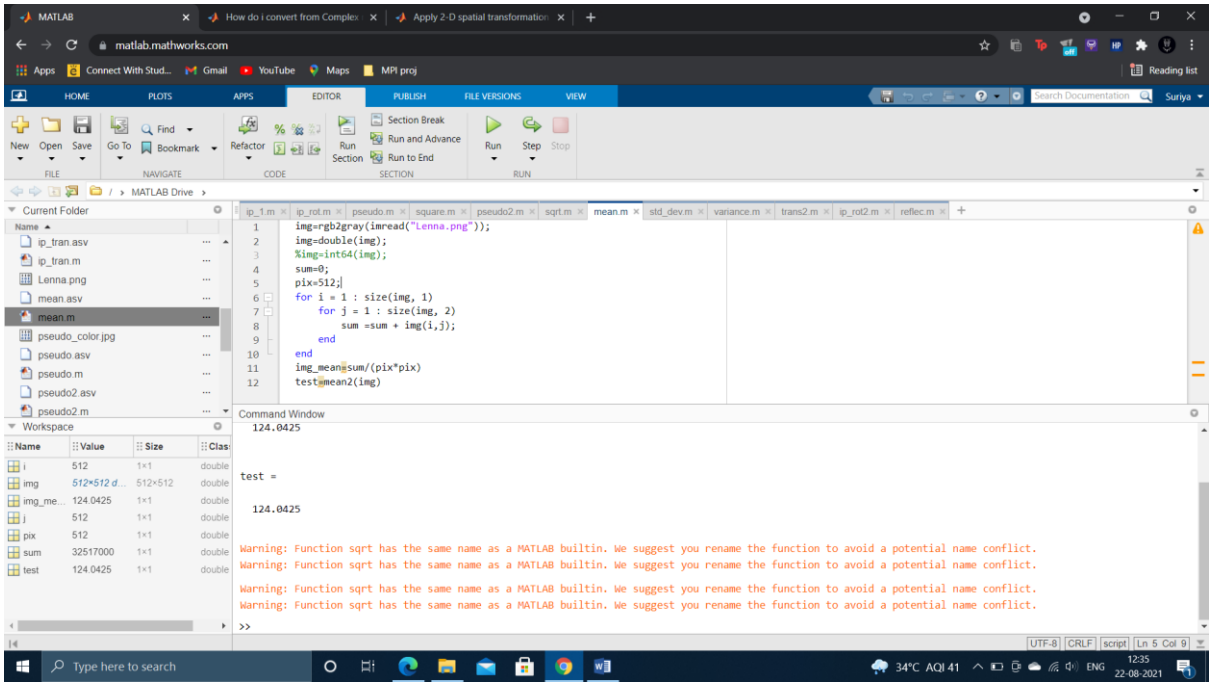
$$m = \frac{\text{sum of the terms}}{\text{number of terms}}$$

- The original image is looped through to find the sum of all elements.
- The sum is then divided by the size of the standard image (512*512)

CODE:

```
img=rgb2gray(imread("Lenna.png"));
img=double(img);
%img=int64(img);
sum=0;
pix=512;
for i = 1 : size(img, 1)
    for j = 1 : size(img, 2)
        sum =sum + img(i,j);
    end
end
img_mean=sum/(pix*pix)
test=mean2(img)
```

OUTPUT:



(xii) Median of an Image

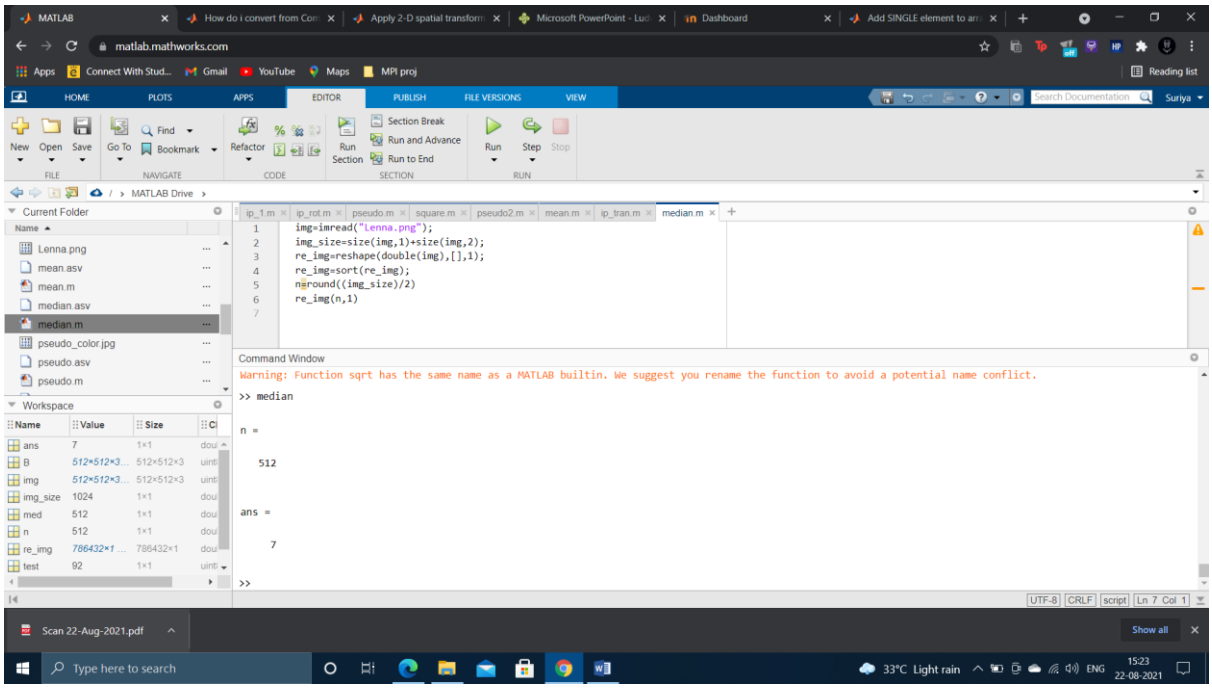
ALGORITHM:

- Median is obtained by reshaping the original 512x512 image matrix into (512*512)x1 matrix
- The row of the resultant matrix is then sorted
- The middle most pixel value is displayed (median)

CODE:

```
img=imread("Lenna.png");
img_size=size(img,1)+size(img,2);
re_img=reshape(double(img),[],1);
re_img=sort(re_img);
n=round((img_size)/2)
re_img(n,1)
```

OUTPUT:



(xiii) Standard Deviation

ALGORITHM:

$$\sigma = \sqrt{\frac{\sum(x_i - \mu)^2}{N}}$$

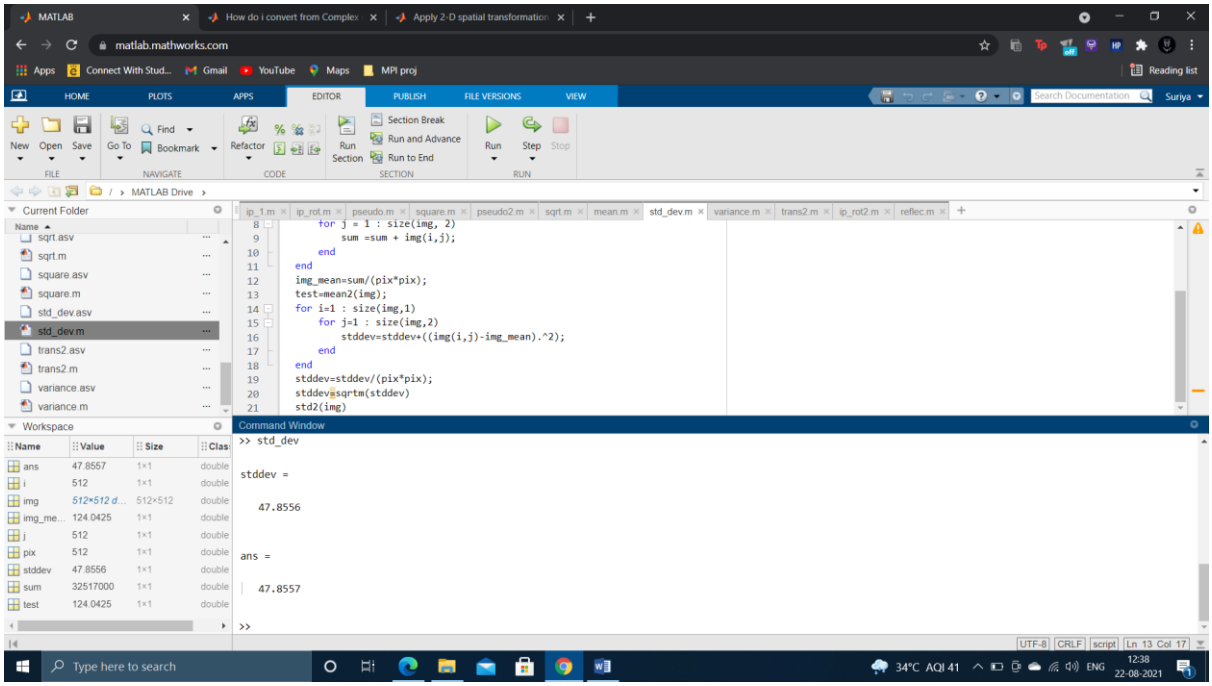
- The mean value previously obtained is subtracted from each individual element of the matrix and squared

- The sum of the square values obtained is divided by the size of the standard image
- The resultant is then square rooted to find the standard deviation

CODE:

```
img=rgb2gray(imread("Lenna.png"));
%img=int64(img);
img=double(img);
sum=0;
stddev=0;
pix=512;
for i = 1 : size(img, 1)
    for j = 1 : size(img, 2)
        sum =sum + img(i,j);
    end
end
img_mean=sum/(pix*pix);
test=mean2(img);
for i=1 : size(img,1)
    for j=1 : size(img,2)
        stddev=stddev+((img(i,j)-img_mean).^2);
    end
end
stddev=stddev/(pix*pix);
stddev=sqrtm(stddev)
std2(img)
```

OUTPUT:



(xiv) Variance of an image

ALGORITHM:

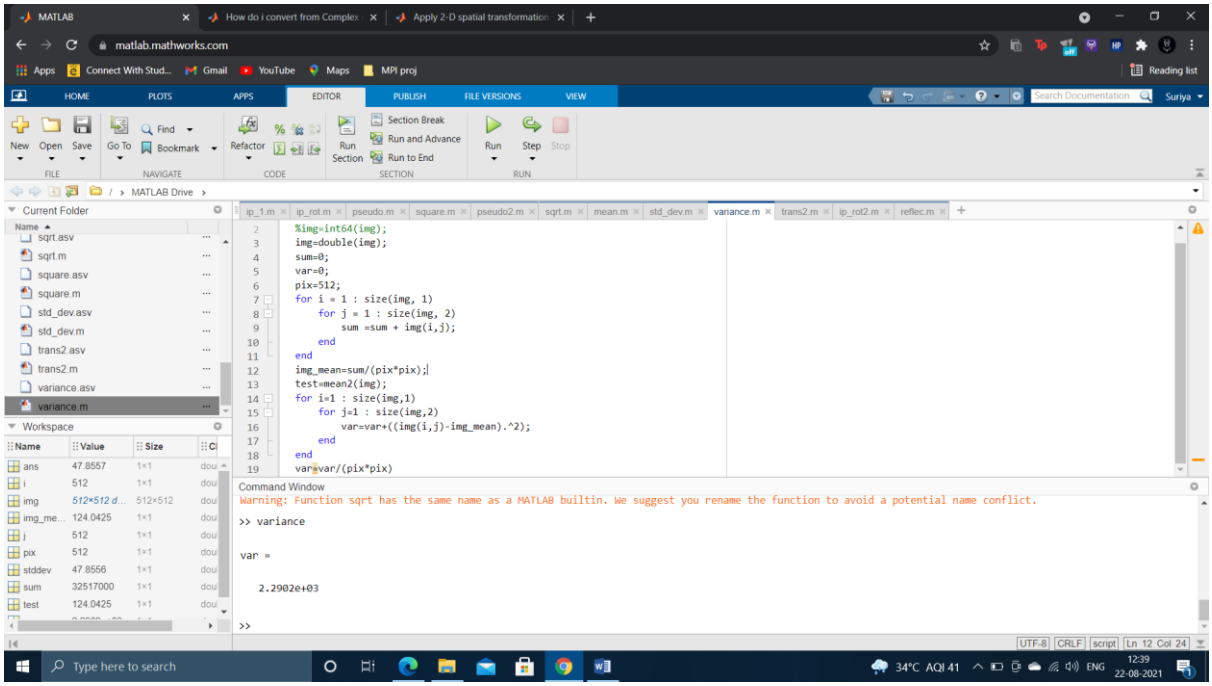
$$S^2 = \frac{\sum (x_i - \bar{x})^2}{n-1}$$

- The mean value previously obtained is subtracted from each individual element of the matrix and squared
- The sum of the square values obtained is divided by the size of the standard image to find the variance

CODE:

```
img=rgb2gray(imread("Lenna.png"));
%img=int64(img);
img=double(img);
sum=0;
var=0;
pix=512;
for i = 1 : size(img, 1)
    for j = 1 : size(img, 2)
        sum =sum + img(i,j);
    end
end
img_mean=sum/(pix*pix);
test=mean2(img);
for i=1 : size(img,1)
    for j=1 : size(img,2)
        var=var+((img(i,j)-img_mean).^2);
    end
end
var=var/(pix*pix)
```

OUTPUT:


MATLAB
How do i convert from Complex
Apply 2-D spatial transformation
matlab.mathworks.com
Apps
Connect With Stud...
Gmail
YouTube
Maps
MPI proj
Reading list
HOME
PLOTS
APPS
EDITOR
PUBLISH
FILE VERSIONS
VIEW
Search Documentation
Suriya
New
Open
Save
FILE
Go To
Find
Bookmark
NAVIGATE
Refactor
CODE
Run Section
Section Break
Run and Advance
Run to End
SECTION
Run
Step
Stop
RUN
MATLAB Drive
Current Folder
Name
sqrt.asv
sqrt.m
square.asv
square.m
std_dev.asv
std_dev.m
trans2.asv
trans2.m
variance.asv
variance.m
Workspace
Name
Value
Size
ans 47.8557 1×1
i 512 1×1
img 512×512 d... 512×512
img_me... 124.0425 1×1
j 512 1×1
pix 512 1×1
stddev 47.8556 1×1
sum 32517000 1×1
test 124.0425 1×1
ip_1.m
ip_rot.m
pseudo.m
square.m
pseudo2.m
sqrt.m
mean.m
std_dev.m
variance.m
trans2.m
ip_rot2.m
reflec.m
%img=int64(img);
img=double(img);
sum=0;
var=0;
pix=512;
for i = 1 : size(img, 1)
for j = 1 : size(img, 2)
sum =sum + img(i,j);
end
end
img_mean=sum/(pix*pix);
test=mean2(img);
for i=1 : size(img,1)
for j=1 : size(img,2)
var=var+((img(i,j)-img_mean).^2);
end
end
var=var/(pix*pix)
Command Window
Warning: Function sqrt has the same name as a MATLAB builtin. We suggest you rename the function to avoid a potential name conflict.
>> variance
var =
2.2902e+03
>>
UTF-8
CRLF
script
Ln 12 Col 24
Type here to search
34°C AQI 41
ENG
12:39
22-08-2021