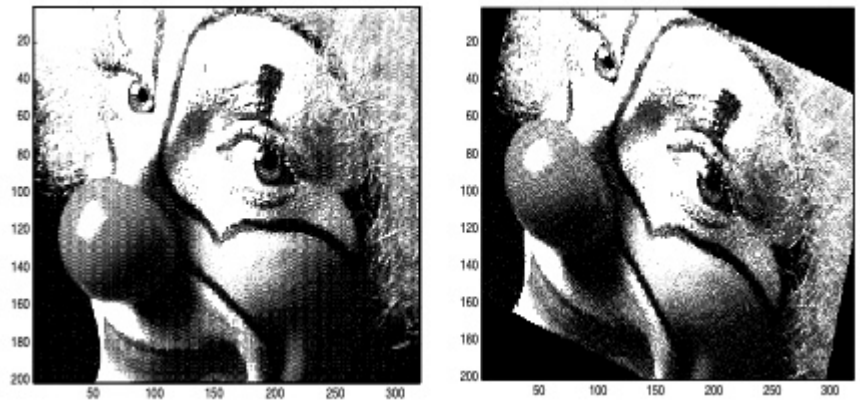
Lab 1 Log Book

Task 1 - Image Rotation

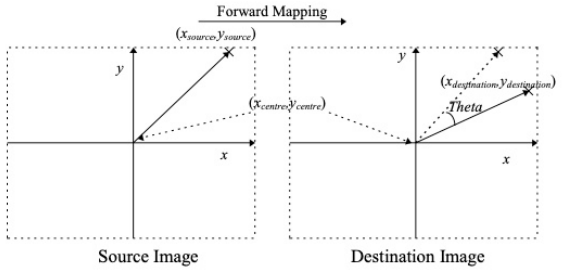
The Requirement

You are required to write a function which rotates a grey scale image by *theta* degrees radian, as shown below.



The function should have the following format:

```
function [Out] =  Rotate(In, Theta)
```

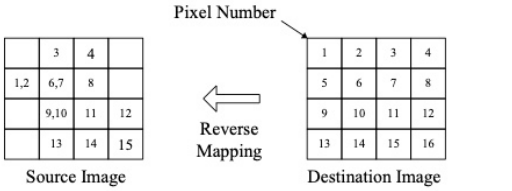


Equation 1 - Equation for rotation of theta about image centre

$$\begin{pmatrix} x_{destination} \\ y_{destination} \end{pmatrix} = \begin{pmatrix} \cos(theta) & \sin(theta) \\ -\sin(theta) & \cos(theta) \end{pmatrix} \left(\begin{pmatrix} x_{source} \\ y_{source} \end{pmatrix} - \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix} \right) + \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix}$$

Equation 2 - Reversing mapping of equation 1

$$\begin{pmatrix} x_{source} \\ y_{source} \end{pmatrix} = \begin{pmatrix} \cos(theta) & \sin(theta) \\ -\sin(theta) & \cos(theta) \end{pmatrix}^{-1} \left(\begin{pmatrix} x_{destination} \\ y_{destination} \end{pmatrix} - \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix} \right) + \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix}$$



So the way to use the reverse mapping would be as follows:

```
function [Out] = rotate(In, Theta)
    % Get the dimensions of the input image
    [height, width] = size(In);

    % Compute the center point of the image (rotation center)
    cp = [round(width / 2), round(height / 2)];

    % Compute the transformation matrix and its inverse (for backward mappin
    tm = [cos(Theta), -sin(Theta); sin(Theta), cos(Theta)];
    rtm = inv(tm);

    % Keep the original image size (crop exceeding parts)
    Out = zeros(height, width); % Initialize output image with a black backgrour

    % Loop through each pixel in the output image
    for y = 1:height
        for x = 1:width
            % Compute the coordinates relative to the center
            p = [x - cp(1); y - cp(2)];

            % Compute the nearest corresponding point in the source image
            tp = round(rtm * p + cp');

            % Limit the index range to prevent exceeding the array boundary
            xSource = max(1, min(width, tp(1)));
            ySource = max(1, min(height, tp(2)));

            % Assignment (ensuring the index does not exceed the boundary)
            Out(y, x) = In(ySource, xSource);
        end
    end
end
```

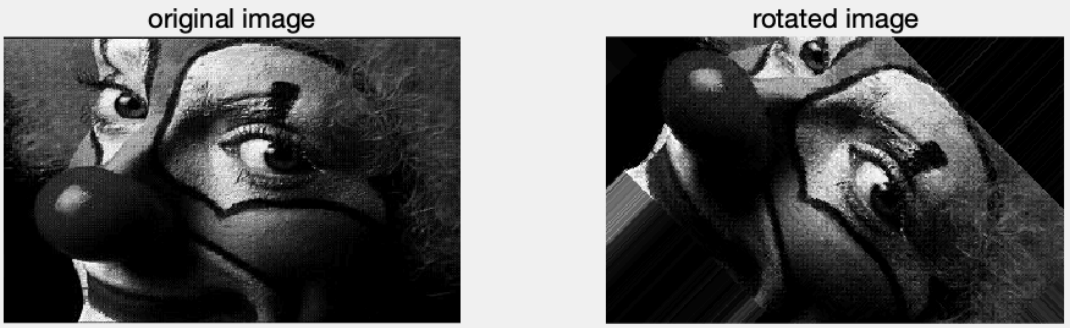
1. First step, I obtained the size of the input image, then calculated the center point of the image, which is the rotation center, and the image rotation is carried out around this point.

2. In the second step, I defined the rotation matrix tm and its inverse rtm. According to the information given in the problem, since the forward operation can cause multiple coordinates to be compressed into a new coordinate, it is difficult to determine how many pixel values should be selected. The inverse operation is very convenient, as it can find the original coordinates of the image before the rotation through your operation, and use interpolation methods (round function) to select the pixel value of the nearest integer coordinate, ensuring that each coordinate is used once.
3. I first preprocess the images to turn them all black, because rotation can cause some parts of the image to have no pixel values. Then I start using two for loops to traverse each position. By using the formula tp = round(rtm * p + cp'), I calculate the pixel values of the original image and insert them into the new coordinates of the output image.

```
load clown.jpg
In = imread('clown.jpg');
Theta = pi/4;
Out = Rotate(In, Theta);
figure;
subplot(1, 2, 1);
imshow(In);
title('original image');

subplot(1, 2, 2);
imshow(uint8(Out));
title('rotated image');
```

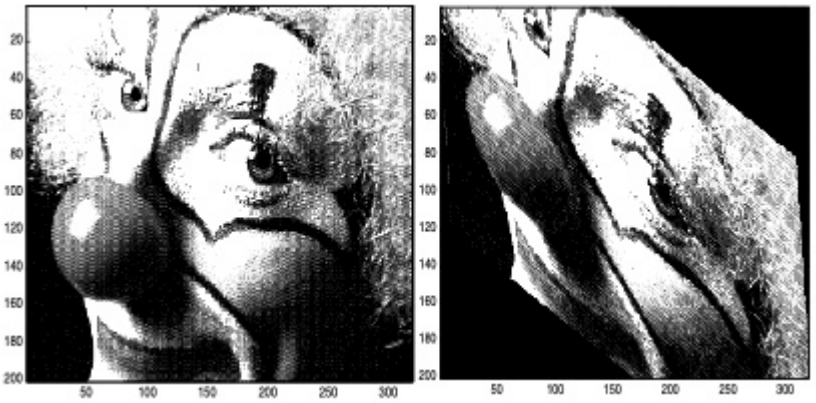
Output:



Task 2 - Image Shearing

The Requirement

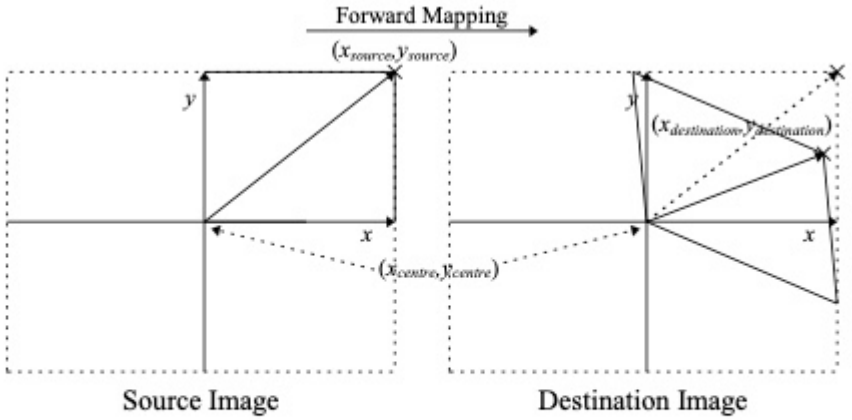
You are required to write a function which shears the input image in both the x and y direction and centres the result, as shown in the figure below (using Xshear=0.1, Yshear=0.5).



The function should have the following format:

```
function [Out] =  Shear(In, Xshear, Yshear)
```

Here's the basics of shearing transformations:



Equation 3 - Equation for shearing the image

$$\begin{pmatrix} x_{destination} \\ y_{destination} \end{pmatrix} = \begin{pmatrix} 1 & Xshear \\ Yshear & 1 \end{pmatrix} \left(\begin{pmatrix} x_{source} \\ y_{source} \end{pmatrix} - \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix} \right) + \begin{pmatrix} x_{centre} \\ y_{centre} \end{pmatrix}$$

```
function [Out] =  Shear(In, Xshear, Yshear)
    % Get the dimensions of the input image.
    [rows, cols] = size(In);

    % Compute the center of the image
    centerX = (cols + 1) / 2;
    centerY = (rows + 1) / 2;

    % define the output image (filled with zeros/black)
    Out = zeros(rows, cols);
    % Loop over every pixel in the output image
    for x_out = 1:cols
        for y_out = 1:rows
            % Compute the coordinates relative to the center
            xRel = x_out - centerX;
            yRel = y_out - centerY;

            % Perform inverse shear transformation (backward mapping)
            x_in = xRel - Xshear * yRel + centerX;
            y_in = yRel - Yshear * xRel + centerY;

            % Use nearest neighbor interpolation
            x_in_round = round(x_in);
            y_in_round = round(y_in);
```

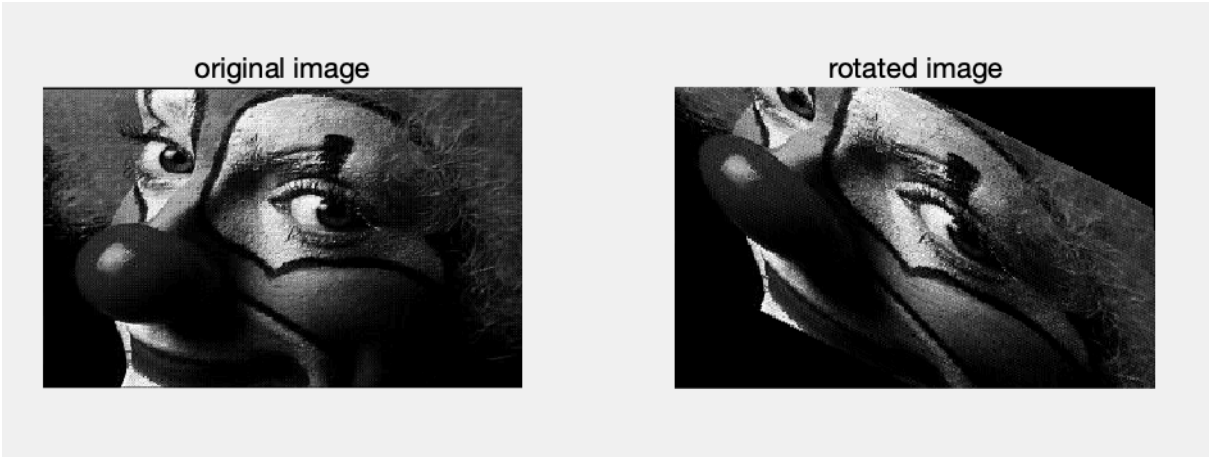
```
            % Check if the source pixel is within bounds of the input image
            if x_in_round >= 1 && x_in_round <= cols && y_in_round >= 1 && y_in_ro
                % Assign the pixel value from the source image
                Out(y_out, x_out) = In(y_in_round, x_in_round);
            end
        end
    end
end
```

1. I adopted an idea similar to rotation for this shear task. First, I get the dimensions of the input images, then calculate the center point of the image, because the shear operation is also performed around this point.
2. In the loop, I still use the inverse transformation and the round function to obtain the pixel values of the original image and assign them to the new coordinates in the new image. Additionally, I check whether this point is within the image; only when it is within the image will I assign a value.

```
load clown.jpg
In = imread('clown.jpg');
Xshear = 0.1;
Yshear = 0.5;
Out = Shear(In, Xshear, Yshear);
figure;
subplot(1, 2, 1);
imshow(In);
title('original image');

subplot(1, 2, 2);
imshow(uint8(Out));
title('rotated image');
```

Output：


original image
rotated image