Table of Contents


```
%NAME: Vatsalya Chaubey
%INST: IIT, Bhubaneswar
%DATE: 3.11.2020
%CATEGORY: Btech
%BRANCH: Electronics and Communication
%Roll Number: 17EC01044
```

Image and Video Processing Assignment - 6

```
clc;
clear all;
close all;
```

Affine Transformation

Affine transformation are linear matrix operations on an image through which an image can be scaled, rotated or sheared. These transformations are linear and so does not affect the shapes in a image. It is usually represented as:

$$[u, v] = T([x, y])$$

(u,v) -> Transformed Coordinates

(x, y) -> Original Coordinates

Code for Affine Transformation

```
function [out] = affine_transformation(img,T)
    % This function performs an affine transformation on an image
    % img: Input image
    % T: Tranformation matrix

    [row, col] = size(img);

    % To calculate the dimension of the output image we can find the
    % maximum and minimum values of the transformed coordinates
    x_min = 0; x_max = 0;
    y_min = 0; y_max= 0;
```

```
    for i=1:row
        for j=1:col
            transformed_coords = round(T * [i; j; 1]);
            x_min = min(transformed_coords(1), x_min);
            x_max = max(transformed_coords(1), x_max);
            y_min = min(transformed_coords(2), y_min);
            y_max = max(transformed_coords(2), y_max);
        end
    end

    out = zeros(x_max-x_min, y_max-y_min);

    % If the minimum value of any coordinate turns out to be zero we
 need
    % to shift it to bring it to 1
    x_shift = (1 + abs(x_min))*(x_min<0);
    y_shift = (1 + abs(y_min))*(y_min<0);

    for i=1:row
        for j=1:col
            transformed_coords = round(T * [i; j; 1]);
            out(transformed_coords(1) + x_shift, transformed_coords(2)
 + y_shift) = img(i, j);
        end
    end

    out = mat2gray(out);
end
```

Example: Scaling

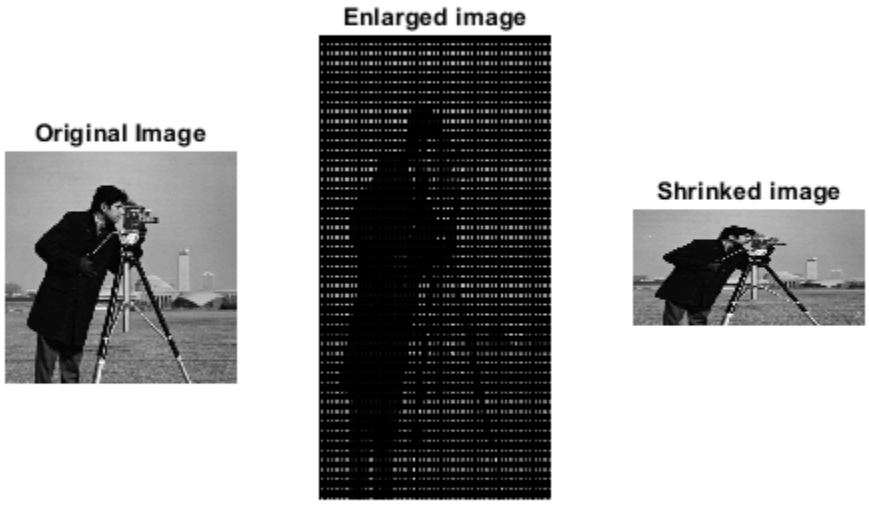
Scaling means to enlarge or compress an image. The transformation matrix for scaling is defined as:

$$T = \begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

a and b are scaling parameters in x and y direction. If a and b are greater than 1 image is enlarged otherwise the image is shrunk.

```
% Read an image
orig_img = imread('cameraman.tif');
img = double(orig_img);

% Create transformation matrix to enlarge
T = [4, 0, 0; 0, 2, 0; 0, 0, 1];
enlarged = affine_transformation(img, T);

% Create transformation matrix to shrink
T = [0.5, 0, 0; 0, 1, 0; 0, 0, 1];
```

```
shrinked = affine_transformation(img, T);

figure('Name', 'Affine Transformation: Scaling');
subplot(131)
imshow(orig_img);
title('Original Image');

subplot(132)
imshow(enlarged);
title('Enlarged image');

subplot(133)
imshow(shrinked);
title('Shrinked image');
```



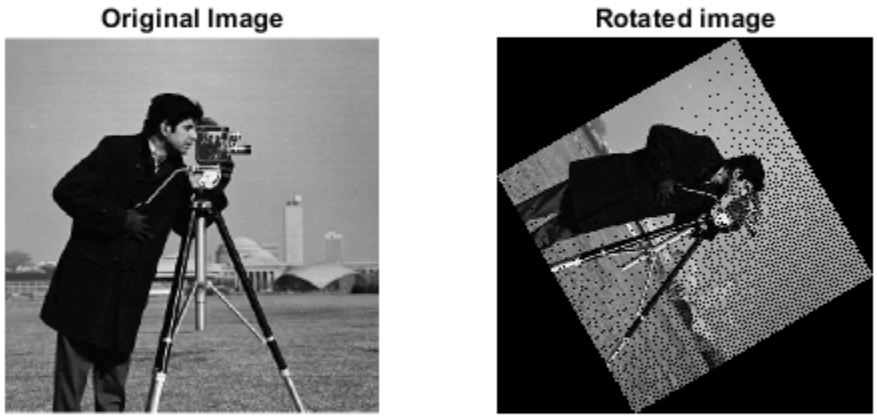
Example: Rotation

To rotate an image through a particular angle we need to define the transformation matrix in the following manner:

$$T = \begin{pmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
% Create transformation matrix to rotate through angle theta
```

```
theta = 60;
T = [ cosd(theta), sind(theta), 0;
     -sind(theta), cosd(theta), 0;
                0,          0, 1];

rotated = affine_transformation(img, T);


figure('Name', 'Affine Transformation: Rotation');
subplot(121)
imshow(orig_img);
title('Original Image');

subplot(122)
imshow(rotated);
title('Rotated image');
```

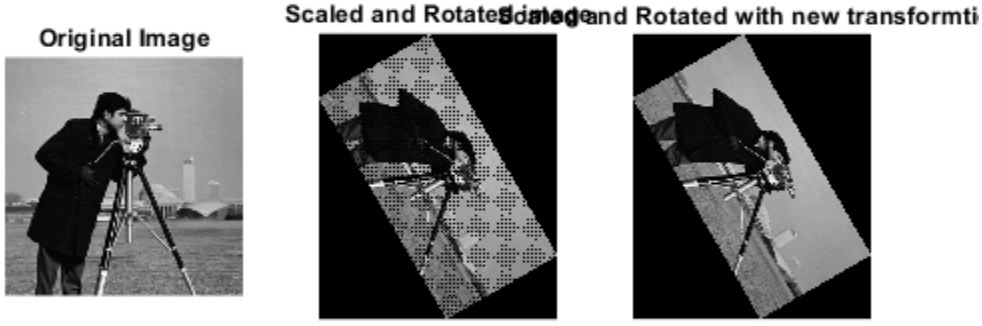


Example: Rotation and Scaling

In this example we will show how can we use succesive transformations

```
% Create transformation matrix to shrink
T1 = [0.5, 0, 0; 0, 1, 0; 0, 0, 1];
enlarged = affine_transformation(img, T1);

% Now, we will rotate the enlarged image
```

```
rot_and_scaled = affine_transformation(enlarged, T);

% We can show that this entire two step process can be represented by
 just
% one transformation matrix which is the product of the two matrices

T_new = T * T1;
transformed = affine_transformation(img, T_new);

figure('Name', 'Scaling and Rotation');
subplot(131)
imshow(orig_img);
title('Original Image');

subplot(132)
imshow(rot_and_scaled);
title('Scaled and Rotated image');

subplot(133)
imshow(transformed);
title('Scaled and Rotated with new transformtion');
```



Conclusion

Through this experiment we learnt about affine transformation. These operations are linear in nature and act on the coordinates of the pixels so they only affect the geometry of the image rather than the image

quality. Affine transformations are very simple to implement and have widespread use in resizing images, comparing rotated images, fixing orientations etc. We also noticed that any complex affine transformation can be broken down into simpler transformations. One more important point to notice is when we enlarge an image through affine transformation we need to use some interpolation technique to fill in the gaps between the pixels.

Published with MATLAB® R2020b