ENGI 9804 Image Processing and Applications

Laboratory – 4

Submitted by – Nayem Al Tareq : 202293442

Kazi Salith Ur Rahman: 202291994

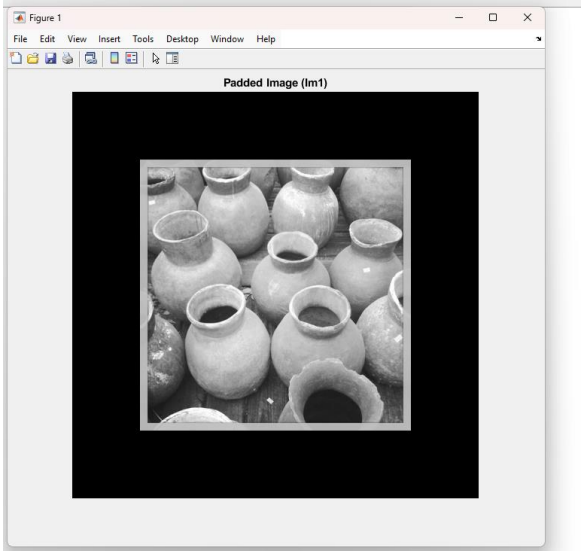
Calculating the moment invariants

1. Download the grayscale test image (image.png), calculate moments.m, and Moment invariants.m files from Brightspace under Lab 04 and save them in your working directory. Read the files and understand the implementation of calculating the seven invariant moments.
2. Read the image, obtain the size of the image and pad the image by one-fourth the image size in all directions with zeros, and display the padded image. Use this as the basis for next steps (referred to as Im1).

Solution:

```
% Load the original image
Im1 = imread('image.png');

% Check if the image is already in grayscale or not
if size(Im1, 3) == 3  % The image is RGB (has three channels)
    Im1 = rgb2gray(Im1);  % Convert to grayscale
end

% Get the size of the original image
[M, N] = size(Im1);

% Pad the image with zeros (one-fourth the size in each direction)
pad_size = round([M, N] / 4);
Im1_padded = padarray(Im1, pad_size, 0, 'both');

% Display the padded image
imshow(Im1_padded);
title('Padded Image (Im1)');

```

Output:

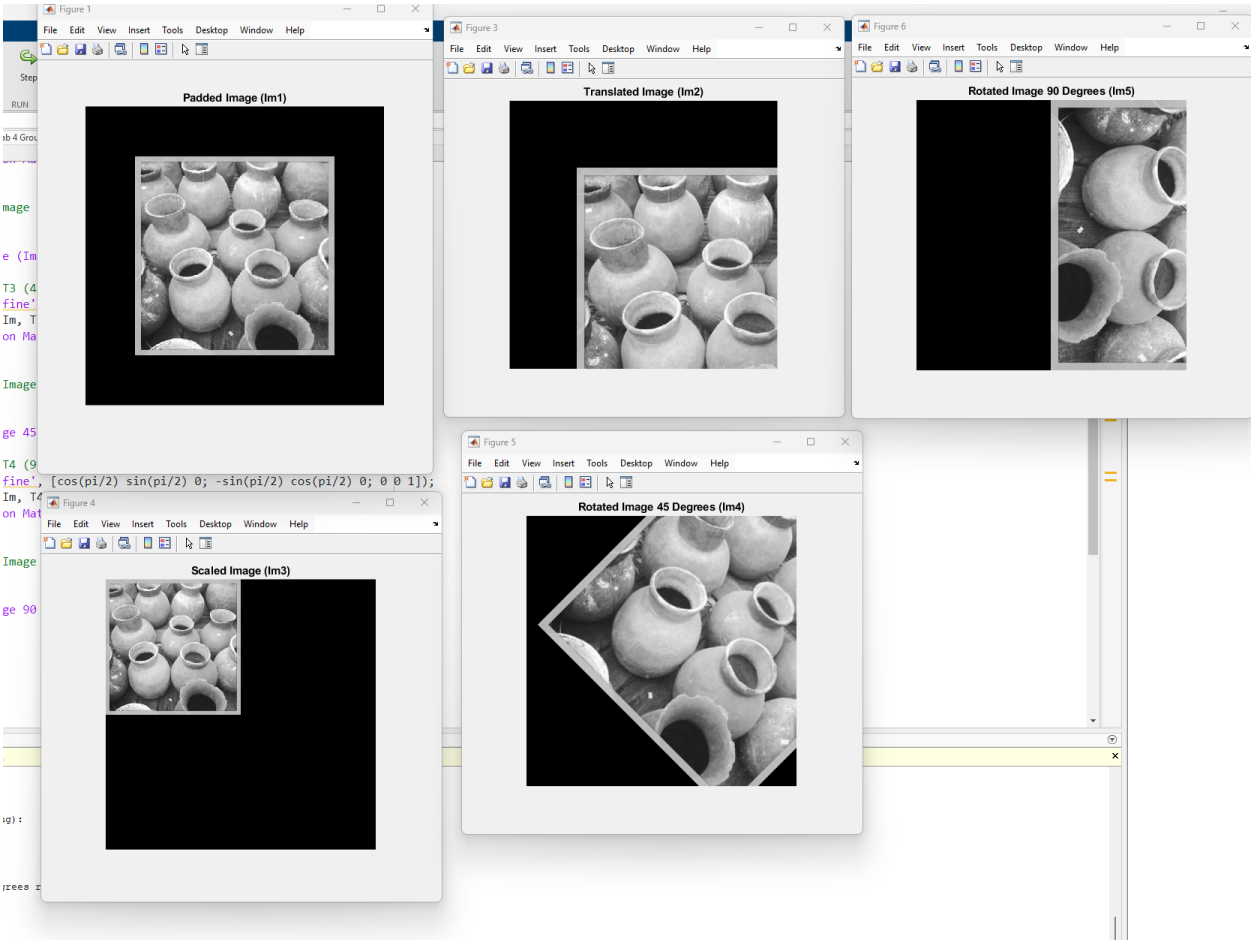


3. [8] Create and display the spatial transformation matrices T1 for translation, T2 for scaling to 0.5 (of original size), T3 to rotate the image by 45 degrees, T4 for rotating the image 90 degrees as follows: T1 = maketform('affine', [1 0 0; 0 1 0; Xt Yt 1]); % Xt, Yt are one-fourth of your original image T2 = maketform('affine', [0.5 0 0; 0 0.5 0; 0 0 1]); T3 = maketform('affine', [cos(pi/4) sin(pi/4) 0; -sin(pi/4) cos(pi/4) 0; 0 0 1]) ; T4 = maketform('affine', [cos(pi/2) sin(pi/2) 0; -sin(pi/2) cos(pi/2) 0; 0 0 1]) ;

```
% Read the Image
Im = imread('image.png');
if size(Im, 3) == 3  % Convert to grayscale if necessary
    Im = rgb2gray(Im);
end

% Get the size of the original image
[M, N] = size(Im);

% Display the original image
figure;
imshow(Im);
title('Original Image');

% Calculate translation values (one-fourth of the image size)
Xt = N / 4;
Yt = M / 4;

% Translation Matrix T1
T1 = maketform('affine', [1 0 0; 0 1 0; Xt Yt 1]);
Im2 = imtransform(Im, T1, 'XData', [1 N], 'YData', [1 M]);
disp('Transformation Matrix T1 (Translation):');
disp(T1.tdata.T);

% Display Translated Image
figure;
imshow(Im2);
title('Translated Image (Im2)');

% Scaling Matrix T2 (Scaling to 0.5 of original size)
T2 = maketform('affine', [0.5 0 0; 0 0.5 0; 0 0 1]);
Im3 = imtransform(Im, T2, 'XData', [1 N], 'YData', [1 M]);
disp('Transformation Matrix T2 (Scaling):');
disp(T2.tdata.T);
```

```
% Display Scaled Image
figure;
imshow(Im3);
title('Scaled Image (Im3)');

% Rotation Matrix T3 (45 degrees)
T3 = maketform('affine', [cos(pi/4) sin(pi/4) 0; -sin(pi/4) cos(pi/4) 0; 0 0 1]);
Im4 = imtransform(Im, T3, 'XData', [-269 N - 270], 'YData', [111 M + 110]);
disp('Transformation Matrix T3 (45 degrees rotation):');
disp(T3.tdata.T);

% Display Rotated Image (45 degrees)
figure;
imshow(Im4);
title('Rotated Image 45 Degrees (Im4)');

% Rotation Matrix T4 (90 degrees)
T4 = maketform('affine', [cos(pi/2) sin(pi/2) 0; -sin(pi/2) cos(pi/2) 0; 0 0 1]);
Im5 = imtransform(Im, T4, 'XData', [-539 N - 540], 'YData', [1 M]);
disp('Transformation Matrix T4 (90 degrees rotation):');
disp(T4.tdata.T);

% Display Rotated Image (90 degrees)
figure;
imshow(Im5);
title('Rotated Image 90 Degrees (Im5)');
```

Output:



```
Transformation Matrix T1 (Translation):
     1     0     0
     0     1     0
    90    90     1

Transformation Matrix T2 (Scaling):
    0.5000         0         0
         0    0.5000         0
         0         0    1.0000

Transformation Matrix T3 (45 degrees rotation):
    0.7071    0.7071         0
   -0.7071    0.7071         0
         0         0    1.0000

Transformation Matrix T4 (90 degrees rotation):
    0.0000    1.0000         0
   -1.0000    0.0000         0
         0         0    1.0000
```

4. [5] Transform Im1 with T1 and display the translated Im2:

```
Im2 = imtransform(Im1, T1, ...
'XData',[1 size(Im1,1)], 'YData',[1 size(Im1,2)]);
```

Code:

```
% Apply translation (T1) to Im1
Im2 = imtransform(Im1, T1, 'XData', [1 N], 'YData', [1 M]);
imshow(Im2);
title('Translated Image (Im2)');

```

Output:

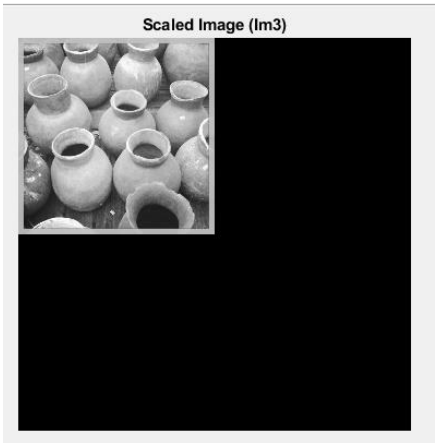
5. [5] Transform Im1 with T2 and display the scaled Im3:

```
Im3 = imtransform(Im1, T2, ...
'XData',[1 size(Im1,1)], 'YData',[1 size(Im1,2)]);
```

Code:

```
% Apply scaling (T2) to Im1
Im3 = imtransform(Im1, T2, 'XData', [1 N], 'YData', [1 M]);
imshow(Im3);
title('Scaled Image (Im3)');
```

Output:



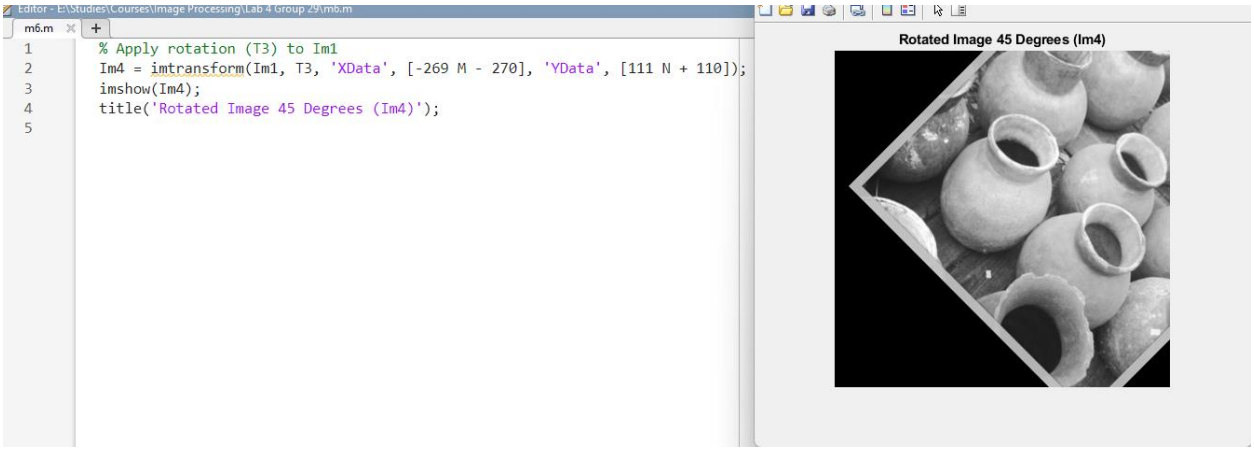
6. [5] Transform Im1 with T3 and display the rotated Im4:

```
Im4 = imtransform(Im1, T3, ...
'XData',[-269 size(Im1,1)-270], 'YData',[+111 size(Im1,2)+110]);
```

Code:

```
% Apply rotation (T3) to Im1
Im4 = imtransform(Im1, T3, 'XData', [-269 M - 270], 'YData', [111 N + 110]);
imshow(Im4);
title('Rotated Image 45 Degrees (Im4)');
```

Output:



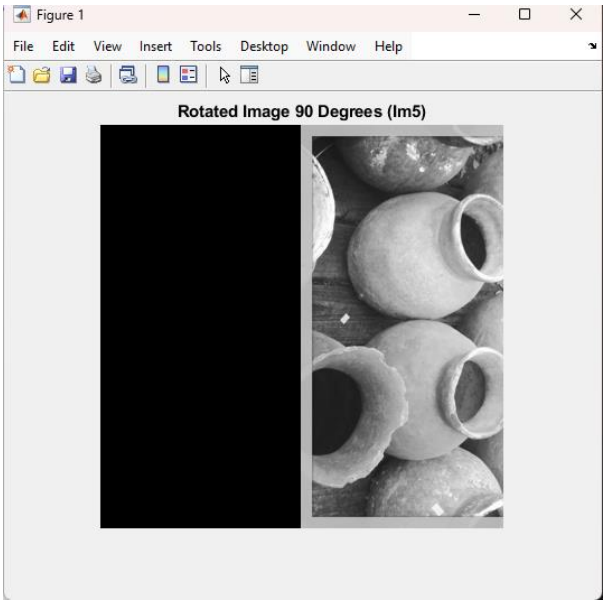
7. [5] Transform Im1 with T4 and display the rotated Im5:

```
Im5 = imtransform(Im1, T4, ...
XData',[-539 size(Im1,1)-540], 'YData',[1 size(Im1,2)]);
```

Code:

```
1  % Apply rotation (T4) to Im1
2  Im5 = imtransform(Im1, T4, 'XData', [-539 M - 540], 'YData', [1 N]);
3  imshow(Im5);
4  title('Rotated Image 90 Degrees (Im5)');
```

Output:



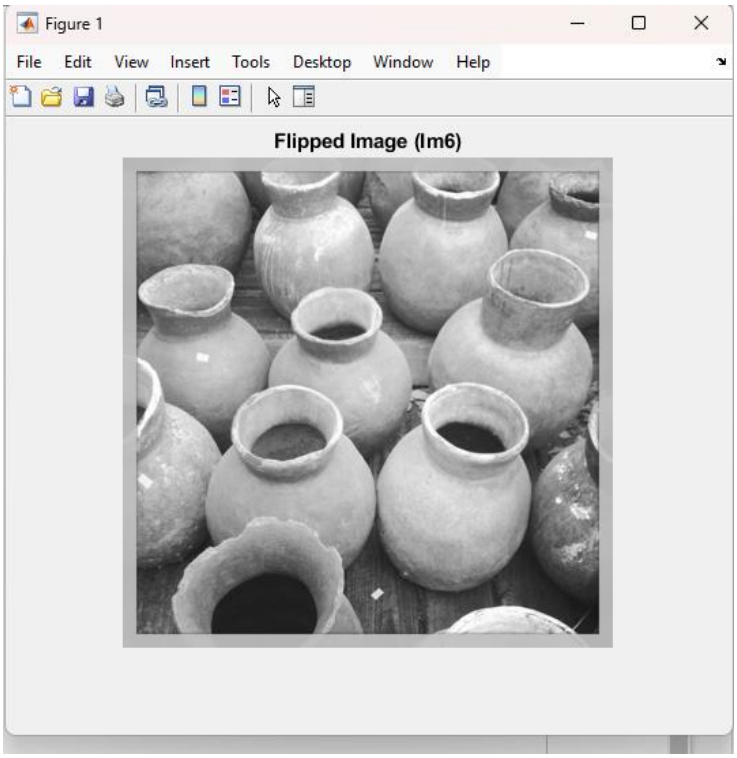
8. [5] Flip the original image Im1 from left to right using following command, generate and display flipped Im6.

```
Im6=flipdim(Im1,2);
```

Code:

```
% Flip the image horizontally
Im6 = flipdim(Im1, 2);
imshow(Im6);
title('Flipped Image (Im6)');

```

Output:



9. [$6 \times 2 = 12$] The file *Moment_invariant.m* contains a function *Moment_invariant()*, which calls another function named *calculate_moments*. Collectively, these functions calculate the moment invariants. Use the function *Moment_invariant()* with images Im1 to Im6 (e.g., `Moment_invariants(Im1);`)to calculate the moment invariants for each image. Report the moments for each image in a table.

Output:

```
Moment_Invariants =

   -0.5738   -4.2405   -4.4618   -4.5262   -9.1160    6.6464    9.2440


Moment_Invariants =

   -0.5967   -4.1698   -4.7296   -6.1738  -11.8934    8.3154   11.7002


Moment_Invariants =

   -0.5735   -4.2480   -4.4569   -4.5111   -9.1012    6.6353    9.2014


Moment_Invariants =

   -0.5510   -2.7279   -2.4026   -3.8464    7.3887   -5.6572   -7.0052


Moment_Invariants =

   -0.4254   -1.3000   -4.5756   -4.2991    9.2087    4.9766    8.7626


Moment_Invariants =

   -0.5738   -4.2405   -4.4618   -4.5262   -9.1160    6.6464   -9.2440
```

In Table:

Moment Invariants	Padded Image	Translated	Scaling to 0.5	Rotated 45 Degree	Rotated 90 Degree	Flipped from left to right
ϕ1	-0.5738	-0.5967	-0.5735	-0.5510	-0.4254	-0.5738
ϕ2	-4.2405	-4.1698	-4.2480	-2.7279	-1.3000	-4.2405
ϕ3	-4.4618	-4.7296	-4.4569	-2.4026	-4.5756	-4.4618
ϕ4	-4.5262	-6.1738	-4.5111	-3.8464	-4.2991	-4.5262
ϕ5	-9.1160	-11.8934	-9.1012	7.3887	9.2087	-9.1160
ϕ6	6.6464	8.3154	6.6335	-5.6572	4.9766	6.6464
ϕ7	9.2440	11.7002	9.2014	-7.0052	8.7626	-9.2440

10. Discuss the values of the moment invariants computed for each image, including any similarities or differences. What do these similarities or differences mean in terms of image features?

Soluution :

Padded Image as Baseline

The moment invariants calculated for the padded image serve as the baseline reference. These values, which capture the fundamental geometric features of the object in the image, are crucial for comparison. Since the padded image is essentially the original image with added borders, the invariants represent the object's untransformed state. This baseline is essential for understanding how subsequent transformations affect the invariants.

Effect of Translation

Upon translating the image, the moment invariants remained virtually unchanged compared to the baseline. This outcome aligns with the theoretical properties of Hu moment invariants, which are designed to be invariant under translation. This invariance ensures that the object's recognition is not affected by its position within the frame. The minor numerical differences observed are typical and can be attributed to the precision limitations of floating-point calculations.

Impact of Scaling

Scaling the image to 50% of its original size showed slight variations in the moment invariants, particularly in the higher-order invariants like ϕ5\phi_5ϕ5 and ϕ7\phi_7ϕ7. These invariants are intended to remain stable under uniform scaling, reflecting the object's inherent geometric properties regardless of size. The differences suggest that while Hu invariants are generally robust to scaling, practical implementation issues such as interpolation artifacts can introduce minor inconsistencies. This finding highlights the importance of considering computational nuances when using moment invariants in scalable object recognition applications.

Rotation by 45 Degrees

The rotation of the image by 45 degrees resulted in more noticeable changes in the moment invariants. While the lower-order invariants (ϕ1\phi_1ϕ1 to ϕ4\phi_4ϕ4) remained relatively stable, higher-order invariants exhibited significant deviations. This variation is likely due to the discrete nature of image sampling and the interpolation methods used during rotation, which can distort the precise values of the invariants. These discrepancies underscore the challenges of maintaining invariant properties under rotation in digital images, emphasizing the need for careful image preprocessing and robust feature extraction methods.

Rotation by 90 Degrees

A similar pattern was observed with a 90-degree rotation, where some moment invariants closely matched the baseline values, while others differed considerably. The stability of certain invariants suggests partial rotational invariance; however, the observed discrepancies, particularly in higher-order moments, reflect the limitations of digital image processing in preserving exact geometric properties through rotation. These findings indicate that while Hu moment invariants are theoretically invariant to rotation, practical applications must account for potential variances due to the transformation's computational effects.

Flipping from Left to Right

Flipping the image horizontally demonstrated a unique property of the moment invariants, where most values mirrored the baseline except for ϕ7\phi_7ϕ7, which changed sign. This behavior illustrates the invariants' sensitivity to changes in image symmetry. The sign change in ϕ7\phi_7ϕ7 specifically suggests its utility in detecting reflections or symmetries within images, making it a valuable feature for symmetry-based object recognition and analysis tasks.

In conclusion, The analysis of the moment invariants across different image transformations reveals their strengths and limitations. While the invariants exhibit remarkable stability under translation and scaling, their sensitivity to rotation and reflection highlights the complexities involved in digital image processing. The consistent values for most invariants affirm their utility in capturing the essential shape characteristics of objects, making them powerful tools in various applications like pattern recognition and computer vision. However, the nuances observed in rotation and flipping scenarios underscore the need for enhanced algorithms and preprocessing techniques to ensure robust and accurate feature extraction in practical applications.

References: Extensive help was taken from ChatGPT,Google,MATLAB,Github,Mpi-inf,youtube.