Laboratory 04

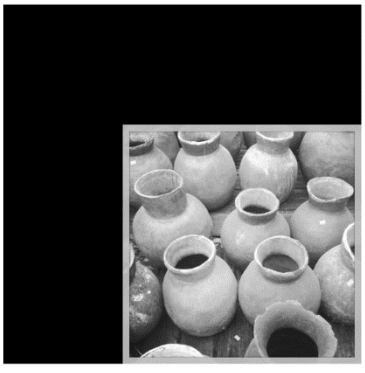
Mohammad Sorkhian

This lab is about using Hu's seven moment invariants for image description. The procedure of calculating this in this lab is as following.

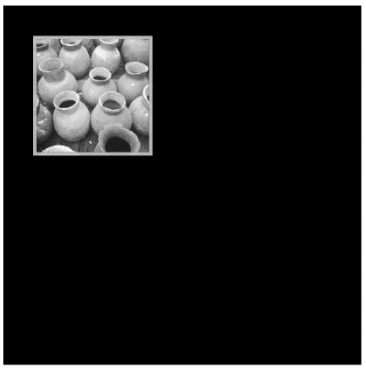
1. I Downloaded and *image.png, calculate moments.m,* and *Moment invariants.m*

2. I padded the original Image by one-fourth of its dimension with zeroes in all directions and called it Im1.

3. I created the T1 spatial transformation matrix and applied it on the Im1 image and named it Im2.



4. I created the T2 spatial transformation matrix (scale 0.5) and applied it on the Im1 image and named it Im3.



5. I created the T3 spatial transformation matrix (rotation 45 degrees) and applied it on the Im1 image and named it Im4.

6. I created the T4 spatial transformation matrix (rotation 90 degrees) and applied it on the Im1 image and named it Im5.

7. I filliped the original Image (Im1).

8. I used *Moment_invariant.m* and *calculate_moments.m* and ran *Moment_invariants()* function for Im1 to Im6.

moment invariants Im1(Original):

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

moment invariants Im2 (Translate):

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

moment invariants Im3 (Scale):

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

moment invariants Im4 (Rotate 45 degrees):

-0.5738 -4.2396 -4.4615 -4.5267 -9.1165 6.6465 9.2446

moment invariants Im5 (Rotate 90 degrees):

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

moment invariants Im6 (Flip):

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

In image recognition detecting similar objects after even a slight translation is really challenging for the computer. Interestingly, regarding output values for moments, we can see almost all these transformations have similar amounts, but in Im3(scale), there is a little difference that can be neglected. Just in the case of flipping, we have a negative sign, while numerically, it is very similar to the others. Therefore, we can conclude that moment is insensitive to transformation.

Matlab Code:

```
clear all;
%%%%%    Section 1&2    %%%%%
img = imread("image.png");
[h, w] = size(img);    % Fetch image size
h_pad = h/4;
w_pad = w/4;
h_n = h + 2*h_pad;
w_n = w + 2*w_pad;
Im1 = zeros(h_n, w_n);
for j = 1:h            % Add original image to padded image
    for i = 1:w
        Im1(h_pad+j, w_pad+i) = img(j, i);
    end
end
Im1 = Im1/255;
% imshow(Im1);

%%%%%    Section 3    %%%%%
T1 = maketform('affine', [1 0 0; 0 1 0; w_pad h_pad 1]);
Im2 = imtransform(Im1, T1,'XData',[1 size(Im1,1)],
'YData',[1 size(Im1,2)]);
% imshow(Im2);

%%%%%    Section 4    %%%%%
T2 = maketform('affine', [0.5 0 0; 0 0.5 0; 0 0 1]);
Im3 = imtransform(Im1, T2, 'XData',[1 size(Im1,1)],
'YData',[1 size(Im1,2)]);
% imshow(Im3);
```

```
%%%%%    Section 5    %%%%%
T3 = maketform('affine', [cos(pi/4) sin(pi/4) 0; -sin(pi/4)
cos(pi/4) 0; 0 0 1]);
Im4 = imtransform(Im1, T3,'XData',[-269 size(Im1,1)-270],
'YData',[+111 size(Im1,2)+110]);
% imshow(Im4);

%%%%%    Section 6    %%%%%
T4 = maketform('affine', [cos(pi/2) sin(pi/2) 0; -sin(pi/2)
cos(pi/2) 0; 0 0 1]);
Im5 = imtransform(Im1, T4,'XData',[-539 size(Im1,1)-540],
'YData',[1 size(Im1,2)]);
% imshow(Im5);

%%%%%    Section 7    %%%%%
Im6=flipdim(Im1,2);
% imshow(Im6);

%%%%%    Section 8    %%%%%
% imshow(Moment_invariants(Im1));
Moment_invariants(Im1);
Moment_invariants(Im2);
Moment_invariants(Im3);
Moment_invariants(Im4);
Moment_invariants(Im5);
Moment_invariants(Im6);
```

```
function [Mpq,MUpq,NUpq]=calculate_moments(f,p,q)
% Calculate Hu's moment invariants
m=size(f,1); n=size(f,2); % get the size of the image
M00=0;M01=0;M10=0;

for i=1:m,
    for j=1:n,
        M00=M00 +double( f(i,j));
        M10=M10 +double( i*f(i,j));
        M01=M01 + double(j*f(i,j));
    end
end
x_bar=M10/M00; y_bar=M01/M00; % coordinate of the centroid
% Use them to calculate the central moments
MU00=M00;
MU11=0;MU12=0;MU20=0;MU02=0; MU21=0;MU30=0;MU03=0;
Mpq=0;MUpq=0;NUpq=0;

for i=1:m,
    for j=1:n,
       MUpq=MUpq+double((i-x_bar)^(p)*(j-
y_bar)^(q)*(f(i,j)));
       Mpq=Mpq+double((i)^p*(j)^q*f(i,j));
    end
end
gamma=(p+q)/2 +1;

NUpq=MUpq/double((MU00)^gamma);
```

```
function Moment_invariants(I)

f=im2double(I);
 % pass the image to a function to calculate the moments
[M20,MU20,NU20]=calculate_moments(f,2,0);
[M02,MU02,NU02]=calculate_moments(f,0,2);
[M11,MU11,NU11]=calculate_moments(f,1,1);
[M21,MU21,NU21]=calculate_moments(f,2,1);
[M12,MU12,NU12]=calculate_moments(f,1,2);
[M03,MU03,NU03]=calculate_moments(f,0,3);
[M30,MU30,NU30]=calculate_moments(f,3,0);

phi1= NU20+NU02;
phi2=(NU20-NU02)^2+(2*NU11)^2;
phi3=(NU30-3*NU12)^2+(3*NU21-NU03)^2;
phi4=(NU30+NU12)^2+(NU21+NU03)^2;
phi5=(NU30-3*NU12)*(NU30+NU12)*((NU30+NU12)^2 -
3*(NU21+NU03)^2)+(3*NU21-
NU03)*(NU21+NU03)*(3*(NU30+NU12)^2-(NU21+NU03)^2);
phi6=(NU20-NU02)*((NU30+NU12)^2-(NU21+NU03)^2) +
4*NU11*(NU30+NU12)*(NU21+NU03);
phi7= (3*NU21-NU03)*(NU30+NU12)*( (NU30+NU12)^2-
3*(NU21+NU03)^2) + (3*NU12-
NU30)*(NU21+NU03)*(3*(NU30+NU12)^2-(NU21+NU03)^2);

Moment_Invariants=[sign(phi1)*log10(abs(phi1))
sign(phi2)*log10(abs(phi2)) sign(phi3)*log10(abs(phi3))
sign(phi4)*log10(abs(phi4)) sign(phi5)*log10(abs(phi5))
sign(phi6)*log10(abs(phi6)) sign(phi7)*log10(abs(phi7))]
```