ASSIGNMENT 2

150001C
Abarajithan G.

1. Image Gradients

```
%Define a Kernel

sobX = [
    -1 -2 -1;
    0 0 0;
    1 2 1 ];

sobY = [
    -1 0 1;
    -2 0 2;
    -1 0 1 ];

edgeX = conv2(sobX, double(im));
edgeY = conv2(sobY, double(im));

edgeabs = (edgeX.^2 + edgeY.^2).^0.5;
edge45 = (edgeX + edgeY)/sqrt(2);
```

```
[ = conv2(sobX, double(im));
[ = conv2(sobY, double(im));
abs = (edgeX.^2 + edgeY.^2).^0.5;
45 = (edgeX + edgeY)/sqrt(2);

ivative of Gaussian

a = 0.5;

= fspecial('gaussian', ceil(sigma*3),sigma);
 = conv2(gau, sobX);
 = conv2(gau, sobY);

geX = conv2(gauX, im);
geY = conv2(gauX, im);

geabs = conv2(gau,edgeabs);
ge45 = conv2(edge45, gau);
```

	Sobel Filter 3 x 3	Gaussian Smoothed Gradient 3x3 with sigma = 0.5	Gaussian Smoothed Gradient 5x5 with sigma = 1.5
X			
Y			
45°			
Magnitude			

2. Canny Edge Detection

Matlab's edge function with 'canny' and threshold as argument gives a binary image. Threshold is between 0 and 1 where 0 finds all the edges and 1 finds none.

Threshold = 0	Threshold = 0.1	Threshold = 0.3
		

```
K = [
    0 0 0 1 0 0 0
    0 0 1 1 1 0 0
    0 1 1 1 1 1 0
    1 1 1 1 1 1 1
    0 1 1 1 1 1 0
    0 0 1 1 1 0 0
    0 0 0 1 0 0 0 ];
```

Structural Elements used

```
K2 = [
    1 1 1
    1 1 1
    1 1 1 ];
```

3. Morphological Operations

Image Erosion with a 7x7 diamond structural element

```
Ans = imerode(im,s);
```

* NOTE: Black blobs have expanded and white regions have shrunk

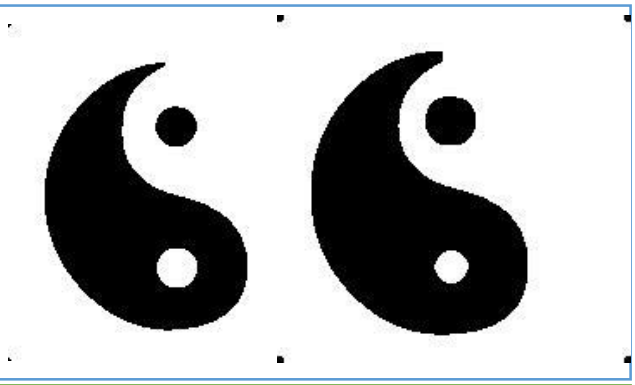


Image Dilation with a 7x7 diamond structural element

```
Ans = imdilate(im,s);
```

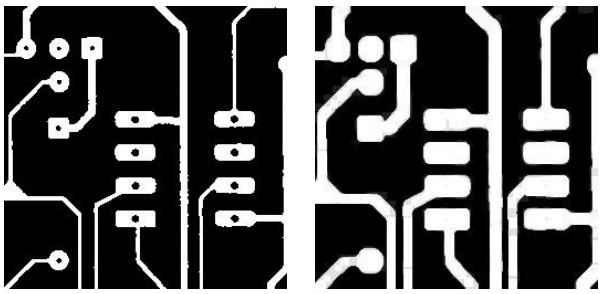


Image Opening with a 3x3 square structural element

```
Ans = imopen(im,s);
```

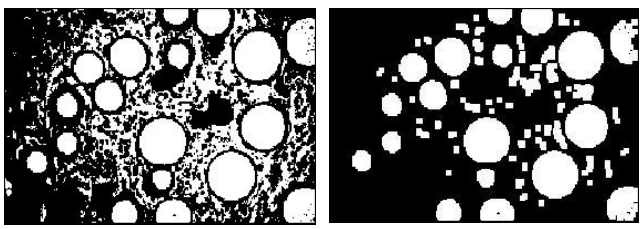
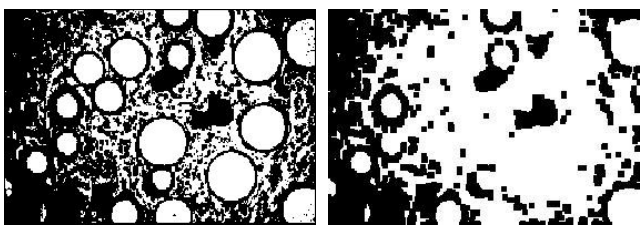


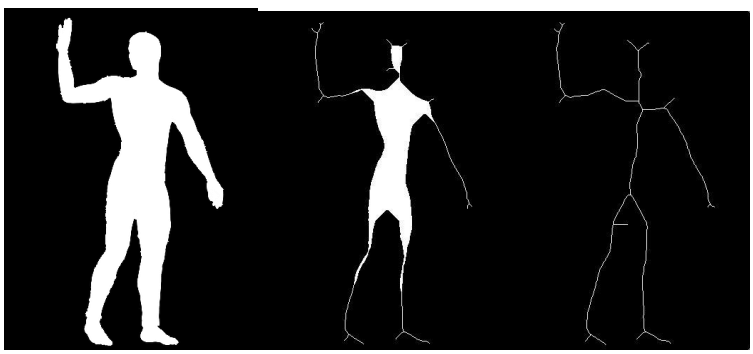
Image Closing with a 3x3 square structural element

```
Ans = imclose(im,s);
```



Skeletonizing

```
Ans = bwmorph(im, 'skel', 15);
```



Original N=15 N=Inf

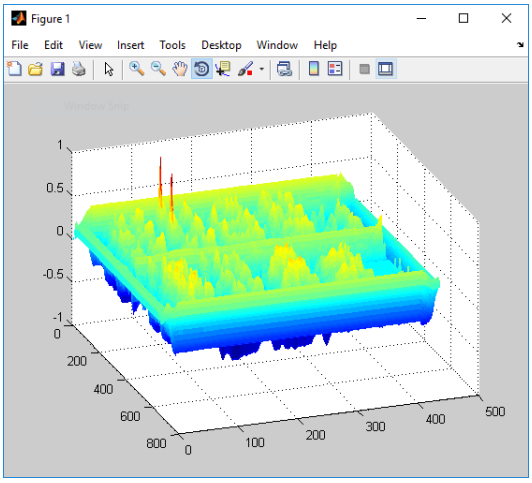
4. Template Matching

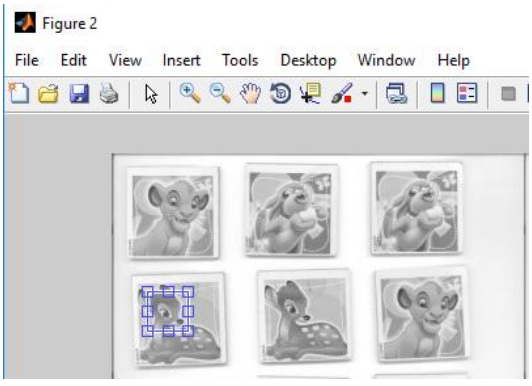
Template matching was carried out with normx2corr2 function which returns the cross correlation Between two matrices within [0,1]

Max value of the cross correlation was found by find and max functions

imrect was used to show the location

```
im = imread('a3.jpg');
im = rgb2gray(im);

q = imread('q3.ppm');
q = rgb2gray(q);


c = normxcorr2(q,im);           %Shows the 3d map
figure, surf(c), shading flat

[ypeak, xpeak ] = find( c == max(c(:)));     % Find the peak point

yoffSet = ypeak-size(q,1);                   % Matching is done orderly, so returns the
xoffSet = xpeak-size(q,2);                   % bottom right corner

hFig = figure;
hAx  = axes;                                 % To show a rectangle
imshow(im,'Parent', hAx);
imrect(hAx, [xoffSet+1, yoffSet+1, size(q,2), size(q,1)]);
```



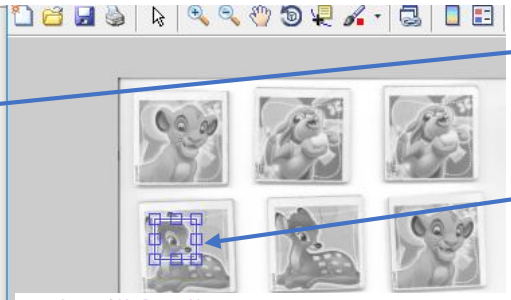


4. Scale Invariant Template Matching

A loop was used to evaluate different maps for each scale (0.5 increment). At each iteration, small sample image was scaled and cross correlation was calculated. These were built into an **image pyramid** after removing the padding caused by cross correlation. Maximum value in this 3D array was found and from its coordinates, the x,y locations and the right scale was found. Rectangle had to be scaled to fit.

Images on next page

```
>> TempMatchScale

rightscale =

    1.5000

>> TempMatchScale

rightscale =

     1

>> TempMatchScale

rightscale =

    2.5000

>> TempMatchScale

rightscale =

    0.5000

>> TempMatchScale

rightscale =

    1.5000
```



Scale Invariant Matching
Correctly identifies the right scale during testing

Correctly identifies the location

```
q = imread('q3.ppm');
q = rgb2gray(q);

c = zeros(size(im));

scales = minscale: increment :maxscale;

for scaleIndex = 2: size(scales,2)

    scale = scales(scaleIndex);
    qscaled = imresize(q,1/scale);

    plane = normxcorr2(qscaled,im);
    plane = plane(size(qscaled,1): size(plane,1)  , size(qscaled,2): size(plane,2) );

    c(:,:,scaleIndex) = plane;


end

% Finding coordinates of maximum
[max_val, position] = max(c(:));

[ypeak ,xpeak ,rightscaleIndex] = ind2sub(size(c),position);
rightscale = scales(rightscaleIndex);
```

5. Rotation Invariant Template Matching

Ciratefi is one of the algorithms (others are: Forapro..etc) used for RST (Rotation-Scale-Translation) invariant template matching. This algorithm follows three important steps:

1. Circular Sampling Filter (Cifi)
   - This filter is used to find the optimum scale matching.
   - It samples the query image (Q) and the template image (A) in concentric circles around each pixel and calculates the circular sampling correlation.
   - Based on a given threshold, the pixel is marked as first grade candidate
   - Also assigns a best-matching-scale to each pixel
2. Radial Sampling Filter (Rafi)
   - This filter checks the first grade candidates and updates them to second grade based on a threshold.
   - Both images are sampled along radial lines of circle centered in the pixel and correlation is checked.
   - Also assigned the best matching rotation angle to each pixel
3. Template Matching Filter (Tefi)
   - First, query image is resized and rotated based on the best matching scales and angles.
   - Then cross correlation is calculated around those pixels as in the previous section.
   - Based on a threshold, filter concludes the matching.

Reading:
Ciratefi: An RST-Invariant Template Matching with Extension to Color Images, Sidnei Alves de Araújoa,b and Hae Yong Kima

Image from:
lps.usp.br/hae/software/cirateg/index.html