# Scilab Manual for Image Processing and Computer Vision Laboratory by Dr Majharoddin Others Dr. G. Y. Pathrikar College Of CS And IT, Aurangabad[1]

Solutions provided by
Dr Majharoddin
Others
Dr. G. Y. Pathrikar College Of CS And IT, Aurangabad

March 11, 2026

[1]Funded by a grant from the National Mission on Education through ICT, http://spoken-tutorial.org/NMEICT-Intro. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website http://scilab.in

# Contents


List of Scilab Solutions 3

1 Basic Statistical Analysis of Images 5

2 Different Filtering Operations on Images 8

3 Morphological Operations on Images 11

4 Basic Image Segmentation 16

5 Image Enhancement for Performance 20

6 Hough Transform Implementation on Images 25

7 SIFT Implementation for Images 28

8 Different Distance Measures implementations 31

9 Basic Shape Analysis Implementation 34

10 Feature Extraction from Images 42

11 Basics of OCR 45

12 Face Detection Algorithm Implementation 50

# List of Experiments


| | | |
|---|---|---|
| Solution 1.1 | Example 1 | 5 |
| Solution 2.1 | Example 1 | 8 |
| Solution 3.1 | Basic Morphological Operations on Image | 11 |
| Solution 4.1 | Example 1 | 16 |
| Solution 5.1 | Example 1 | 20 |
| Solution 6.1 | Example 1 | 25 |
| Solution 7.1 | Example 1 | 28 |
| Solution 8.1 | Example 1 | 31 |
| Solution 9.1 | Example 1 | 34 |
| Solution 10.1 | Example 1 | 42 |
| Solution 11.1 | Example 1 | 45 |
| Solution 12.1 | Example 1 | 50 |
| AP 1 | groupimage | 56 |
| AP 2 | kalpanachawla | 56 |
| AP 3 | randomimage | 57 |
| AP 4 | numberimage | 57 |
| AP 5 | regionimage | 58 |
| AP 6 | textimage | 58 |
| AP 7 | shape | 59 |
| AP 8 | lennaimage | 59 |
| AP 9 | circleimage | 60 |
| AP 10 | harewood | 61 |
| AP 11 | Text Image | 62 |
| AP 12 | inputimage | 63 |

# List of Figures


2.1 Example 1 . . . . . 10

3.1 Basic Morphological Operations on Image . . . . . 14
3.2 Basic Morphological Operations on Image . . . . . 15

4.1 Example 1 . . . . . 19

5.1 Example 1 . . . . . 22
5.2 Example 1 . . . . . 23
5.3 Example 1 . . . . . 24

6.1 Example 1 . . . . . 27

7.1 Example 1 . . . . . 30

9.1 Example 1 . . . . . 37
9.2 Example 1 . . . . . 38
9.3 Example 1 . . . . . 39
9.4 Example 1 . . . . . 40
9.5 Example 1 . . . . . 41

10.1 Example 1 . . . . . 44

11.1 Example 1 . . . . . 48
11.2 Example 1 . . . . . 49
11.3 Example 1 . . . . . 49

12.1 Example 1 . . . . . 53
12.2 Example 1 . . . . . 54
12.3 Example 1 . . . . . 54

# Experiment: 1

# Basic Statistical Analysis of Images

check Appendix AP 12 for dependency:

`cameraman.png`

**Scilab code Solution 1.1** Example 1

```
// This scilab code is to calculate basic image statistics
//such as sum, average, standard deviation, min and max

//The scilab environment for this is : Scilab 5.5.2

//Toolbox used: SIVP 0.5.3.2

//OS used :Windows 10 64 bit
//
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

```

```
clc //to clear command window.
clear all //to kill previously defined variables.
xdel(winsid())//to close all currently open figure(s).

//This code uses camaraman.tif file for processing

img = imread('cameraman.tif'); // Reads the cameraman image
imshow(img); // Show original image.
if(size(img,3)>1) then
    Gray_img= rgb2gray(img); //Converts color image to grayscale
else
    Gray_img=img;            // if image is 2D image
end
imshow(Gray_img);            // Show gray scale image.
imgsum=sum(Gray_img);        // Calculates sum of image pixels
imgmean=mean2(Gray_img);     // Calculates mean / average of image pixels
imgstd=std2(Gray_img);       // Calculates standard deviation of image
imgmin=min(Gray_img);        // Calculates minimum of image pixels
imgmax=max(Gray_img);        // Calculates maximum of image pixels.
disp("Size of Image =");
disp(size(Gray_img));        // Displays size of image
disp("Sum = ");
disp(imgsum);                // Displays Sum of image pixels
disp("Mean = ");
disp(imgmean);               // DIsplays mean / average of image pixels
disp("Standard Deviation = ");
```

```
disp(imgstd);                    // Displays standard deviation of image pixels
disp("Minimum of Image = ");
disp(imgmin);                    // Displays minimum of image
disp("Maximum of Image =");
disp(imgmax);                    // Displays maximum of image

////////// OUTPUT //////////
//
//Size of Image =
//
//     256.     256.
//
// Sum =
//
//   120
//
// Mean =
//
//     118.72449
//
// Standard Deviation =
//
//     62.341715
//
// Minimum of Image =
//
//   7
//
// Maximum of Image =
//
//   253
```

# Experiment: 2

# Different Filtering Operations on Images

check Appendix AP 12 for dependency:

`cameraman.png`

**Scilab code Solution 2.1** Example 1

```
// This scilab code is to calculate basic image statistics
//such as sum, average, standard deviation, min and max

//The scilab environment for this is : Scilab 5.5.2

//Toolbox used: SIVP 0.5.3.2

//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

```

```
clc //to clear command window.
clear all //to kill previously defined variables.
xdel(winsid())//to close all currently open figure(s).

//This code uses camaraman.tif file for processing
//SCI + '/contrib/SIVP_0.5.3.2/images/lena.png'
img = imread('cameraman.png');  // Reads the cameraman image
//title("Original Image");
imshow(img);                    // Show original image.
if(size(img,3)>1) then
    Gray_img= rgb2gray(img);    //Converts color image to grayscale
else
    Gray_img=img;               // if image is 2D image
end


filter = fspecial('sobel');     // Creates 2D special filter 'sobel'
imf = imfilter(Gray_img, filter); // Apply filter on 2D image
//pause;
//title("Filtered Image");
figure, imshow(imf);                    // Show Resultant Image
```

Figure 2.1: Example 1

# Experiment: 3

# Morphological Operations on Images

check Appendix AP 11 for dependency:

`Fig0419.png`

**Scilab code Solution 3.1** Basic Morphological Operations on Image

```

// This scilab code is to perform basic morphological operation on image
//such as erosion and dilation , opening and closing an image

//The scilab environment for this is : Scilab 5.5.2

//Toolbox used: Image Processing Desing (IPD) Toolbox ver. 8.3.3
//                SIVP 0.5.3.2

//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
```

```
//book author: Rafael C. Gonzalez and Richard E. Woods

clc //to clear command window.
clear all //to kill previously defined variables.
xdel(winsid())//to close all currently open figure(s).

//This code uses camaraman.tif file for processing
//SCI + '/contrib/SIVP_0.5.3.2/images/lena.png'
img = imread('Fig0419.png');  // Reads the text image (Image from References)
//title("Original Image");
imshow(img);                      // Show original image.
if(size(img,3)>1) then
    Gray_img= rgb2gray(img);      //Converts color image to grayscale
else
    Gray_img=img;                 // if image is 2D image
end
SE=CreateStructureElement('square',3); // create structuring element 3x3 square matrix

Eimg=ErodeImage(Gray_img,SE); // erosion of the image
pause;                            // type 'resume' command on command prompt
imshow(Eimg);                     // show Resultant image
//se=[1 1 1;1 1 1;1 1 1];
Dimg=DilateImage(Gray_img,SE);  // dilation of the image
pause;                            // type 'resume' command on command prompt
imshow(Dimg);                      // Show Resultant Image
```

```

Oimg=OpenImage(Gray_img,SE);  // opening operation on the image
pause;                          // type 'resume' command on command prompt
imshow(Oimg);                    // Show Resultant Image


Cimg=CloseImage(Gray_img,SE);  // closing operation on the image
pause;                          // type 'resume' command on command prompt
imshow(Cimg);                    // Show Resultant Image
```

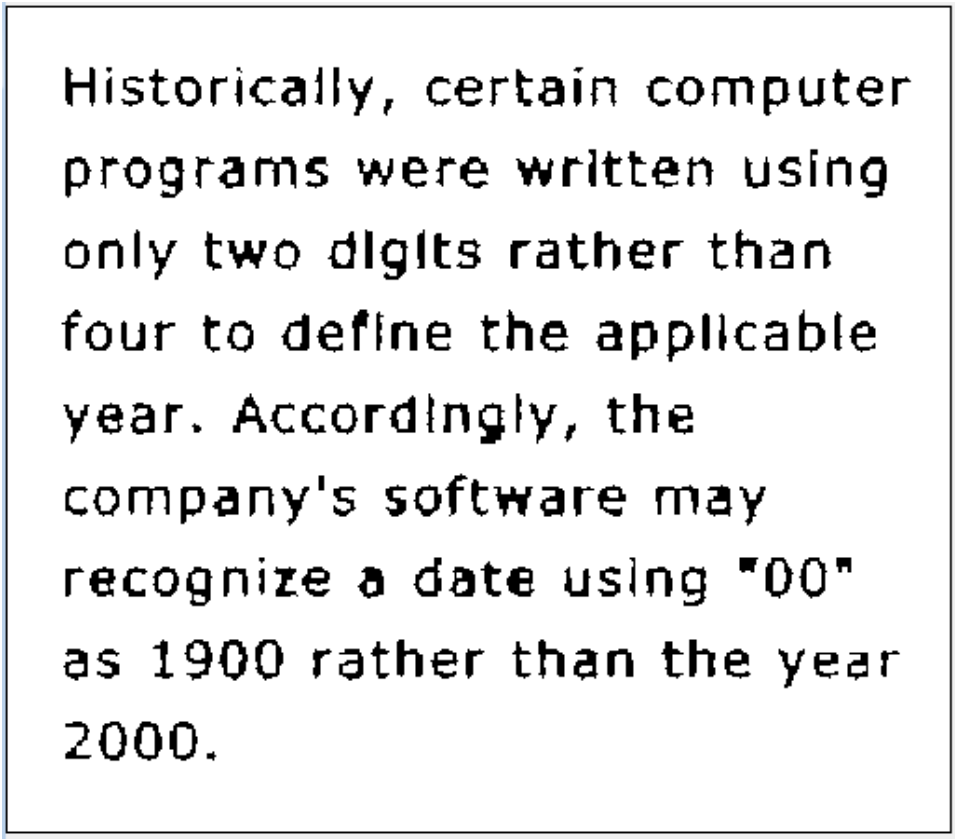



Figure 3.1: Basic Morphological Operations on Image

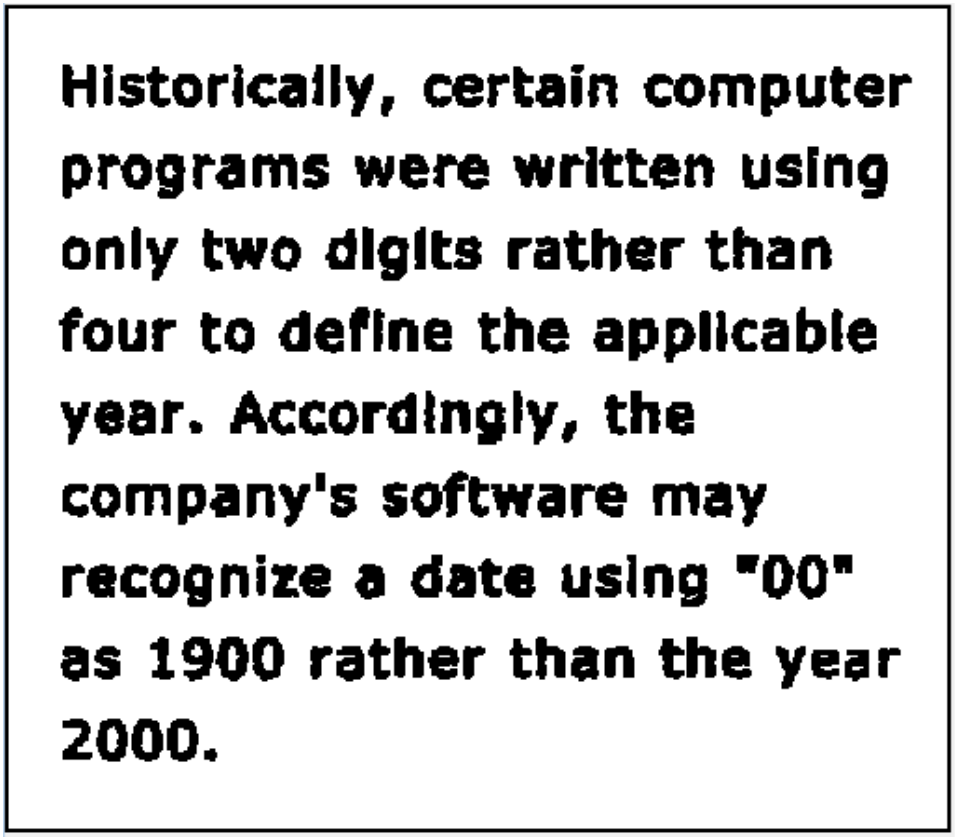



Figure 3.2: Basic Morphological Operations on Image

# Experiment: 4

# Basic Image Segmentation

check Appendix AP 10 for dependency:

harewood.jpg

**Scilab code Solution 4.1** Example 1

```

// This scilab code is used to perform image segmentation using Otsu Thresholding method

//The scilab environment for this is : Scilab 6.0.1

//Toolbox used: Image Processing and Computer Vision Toolbox ver. 1.1


//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

//clc //to clear command window.
```

```
//clear all //to kill previously defined variables.
//xdel(winsid())//to close all currently open figure(s).

//This code uses harewood.jpg image file for processing
//clc;
//clear all;
//I=imread('nutsBolts.jpg');
I=imread('harewood.jpg');                    // Reads input image
//I=rgb2gray(I);

Ihist=imhist(I);                             // Calculate histogram
Ihist=Ihist';                                // Transpose the histogram output
s=length(I);                                 // Calculates Total number of pixels
t=0:255;                                     // set counter t as 0 to 255
sumA=0.0;                                    // Initialize the sum variable
for t=1:256                                  // Calculate sum of pixels in histogram output
    sumA = sumA+ t*Ihist(t);
end
sumB = 0;                                    // Initialize another variable as summation
wB = 0;                                      // Initialize weight background variable
wF = 0;                                      // Initialize weight foreground variable
varMax = 0;                                  // Initialize maximum value for threshold
threshold = 0;                               // Initialize threshold to zero

```

```
// This for block calucates threshold using Otsu Thresholding method
for t=1:256
    wB = wB + Ihist(t);                    // Weight Background
    if (wB == 0)
        continue;
    end
    wF = s - wB;                      // Weight Foreground
    if (wF == 0)
        break;
    end
    sumB = sumB + (t * Ihist(t));
    mB = sumB / wB;              // Mean Background
    mF = (sumA - sumB) / wF;     // Mean Foreground
    // Calculate Between Class Variance
    varBetween = wB * wF * (mB - mF) * (mB - mF);
    // Check if new maximum found
    if (varBetween > varMax)
       varMax = varBetween;
       threshold = t;
    end
end
//n=(threshold - max(I))/ (max(I)-min(I));
R=im2bw(I,threshold/256);                          // Binarize image with given threshold
imshow(R);                                         // Show resultant binarized image
//
```

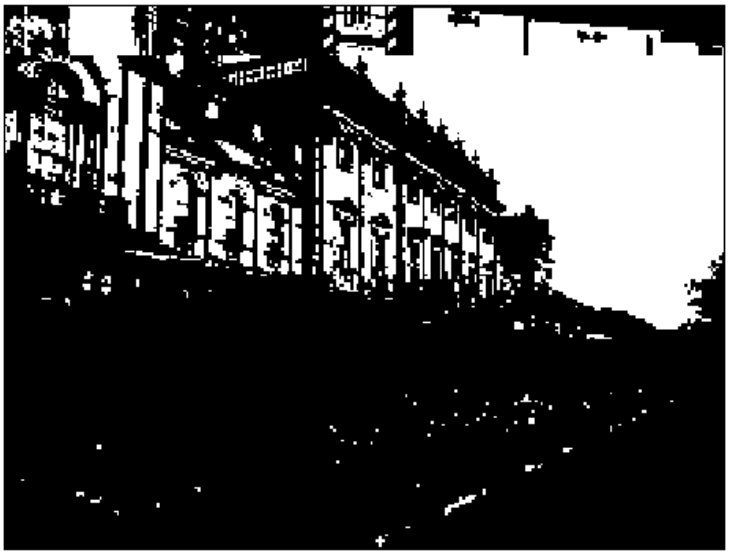


Figure 4.1: Example 1

# Experiment: 5

# Image Enhancement for Performance

check Appendix AP 8 for dependency:

`lenna1.png`

**Scilab code Solution 5.1** Example 1

```

// This scilab code is to perform image enhancement by adjusting the intensity values in the image
//which also called as histogram equalization.

//The scilab environment for this is : Scilab 6.0.1

//Toolbox used: Image Processing and Computer Vision Toolbox ver. 1.1


//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
```

```
//book author: Rafael C. Gonzalez and Richard E. Woods

clc //to clear command window.
clear all //to kill previously defined variables.
xdel(winsid())//to close all currently open figure(s).

//This code uses lenna.png image file for processing


img=imread('lenna.png');                    // Reads original lenna (color) image
grayImg=rgb2gray(img);                      // Converts color image to gray scale image
histequalImg=imhistequal(grayImg);          // Perform image enhancement using intensity adjustment i.e. histogram equalization
imshow(img);                                // Displays original lenna image
figure;imshow(grayImg);                     // Display resultant gray scale image
figure;imshow(histequalImg);                // Displyas resultant histogram equalized image
```

Figure 5.1: Example 1

Figure 5.2: Example 1

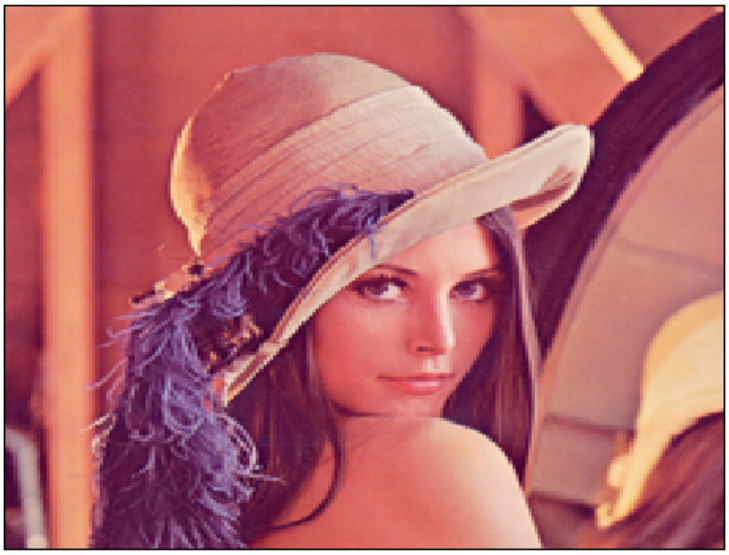


Figure 5.3: Example 1

# Experiment: 6

# Hough Transform Implementation on Images

check Appendix AP 9 for dependency:

pic26.png

**Scilab code Solution 6.1** Example 1

```

// This scilab code is used to perform Hough
   Transform operation on the image
//to extract the features from image.

//The scilab environment for this is : Scilab 6.0.1

//Toolbox used: Image Processing and Computer Vision
    Toolbox ver. 1.1


//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
```

```
//book author: Rafael C. Gonzalez and Richard E. Woods

//clc //to clear command window.
//clear all //to kill previously defined variables.
//xdel(winsid())//to close all currently open figure(s).

//This code uses an circle image pic26.png file for processing

S = imread('pic26.png');                    // Read input image using imread function
[houghmat, dist, theta] = imhough(S);       // Apply hough transformation
                                            // This returns hough matrix houghmat
                                            // distance from center to the point
                                            // and angle of point to the
```

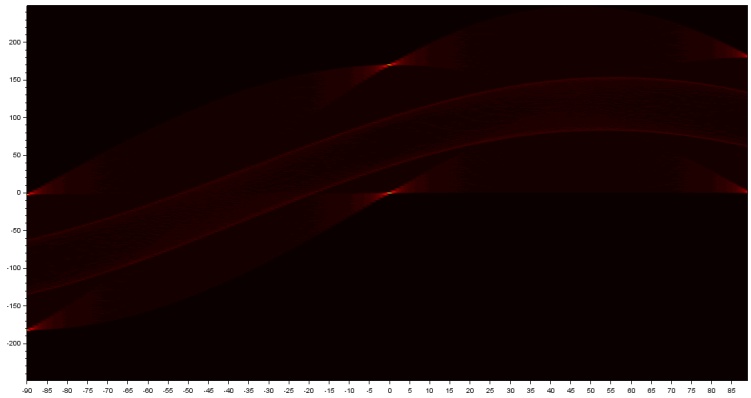



Figure 6.1: Example 1

center

```
f=scf();                                          // Assign current graphics window handle to f
f.color_map=hotcolormap(64);                      // Change colormap for current graphics window
Sgrayplot(theta,dist,houghmat',strf="021"); // Plots smooth 2D surface
```

# Experiment: 7

# SIFT Implementation for Images

check Appendix AP 8 for dependency:

`lenna1.png`

**Scilab code Solution 7.1** Example 1

```

// This scilab code is to perform SIFT Transfom on the image
//to extract the features from image.

//The scilab environment for this is : Scilab 6.0.1

//Toolbox used: Image Processing and Computer Vision Toolbox ver. 1.1


//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
```

```
//book author: Rafael C. Gonzalez and Richard E. Woods

//clc //to clear command window.
//clear all //to kill previously defined variables.
//xdel(winsid())//to close all currently open figure(s).

//This code uses lenna.png image file for processing


img=imread('lenna1.png');                    // Reads original lenna (color) image
SIFTfeatures=imdetect_SIFT(img);             // Extract features from image using SIFT algorithm
imshow(img);                                 // Displays original lenna image
plotfeature(SIFTfeatures);                   // Plot the features extracted on displayed image
```

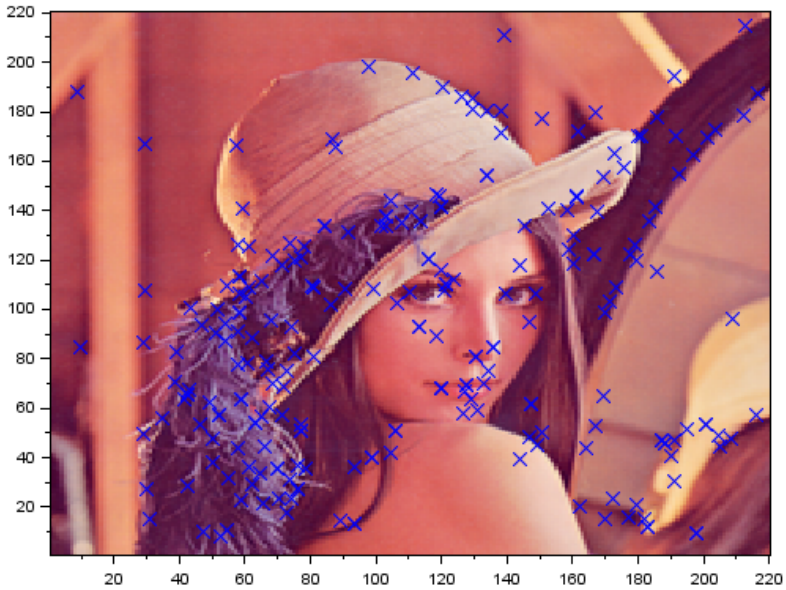



Figure 7.1: Example 1

# Experiment: 8

# Different Distance Measures implementations

**Scilab code Solution 8.1** Example 1

```

// This scilab code is used to calculate distance measure between different vectors
//such as euclidean distance, checkerboard distance and mahalanobis distance

//The scilab environment for this is : Scilab 6.0.1

//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

//clc //to clear command window.
clear all //to kill previously defined variables.
//xdel(winsid())//to close all currently open figure(s).

```

```
//A  = rand(1, 100);                                  // Create a vector of random values
//B = rand(1, 100);                                   // Creates another vector of random values

A = [5.76 43 34;6.7 32 5;3 3 5;34 12 6];     // Create a vector A
B = [1.55 5 32;4.5 9 8;2 4 3;3.2 6 7];                                  // Create a vector B

// Euclidean Distance
D  = sqrt(sum((A - B) .^ 2))                  // Calculates euclidean distance between A and B
Da  = sqrt(sum((A - A) .^ 2))                 // Calculates euclidean distance between A itself
Db  = sqrt(sum((B - B) .^ 2))                 // Calculates euclidean distance between B itself


// Checkerboard Distance
Dc1 = sum(sum((A-B).^2,2))                          // Calculates Checkerboard distance between A and B
Dc2 = sum(sum((A-A).^2,2))                          // Calculates Checkerboard distance between A and B
Dc3 = sum(sum((B-B).^2,2))                          // Calculates Checkerboard distance between A itself


// Mahalanobis Distance
A = [5.76 43 34;6.7 32 5;3 3 5;34 12 6];     // Create a vector A
B = [1.55 5 32];                              // Create a vector B
cv = cov(A);                                  // Calculate covariance matrix of A
mu = mean(A,1);                               // Calucated mean value of A
Dm = (B-mu)*inv(cv)*(B-mu)';                   //
```

```
    Calculates Mahalanobis distance between A and B
//out=[D Da Db Dc1 Dc2 Dc3 Dm]
disp([D Da Db Dc1 Dc2 Dc3 Dm]);                    //
    Displays output on Scilab console

////////////////////////////// OUTPUT
    ////////////////////////////////////////////////

//54.774119    0.    0.    3000.2041    0.    0.
    11.17056
```

# Experiment: 9

# Basic Shape Analysis Implementation

check Appendix AP 7 for dependency:

`shapes.jpg`

**Scilab code Solution 9.1** Example 1

```

// This scilab code is used to perform different edge detection methods on images
//which can be used in shape analysis.

//The scilab environment for this is : Scilab 6.0.1

//Toolbox used: Image Processing and Computer Vision Toolbox ver. 1.1


//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
```

```
//book author: Rafael C. Gonzalez and Richard E. Woods

//clc //to clear command window.
//clear all //to kill previously defined variables.
//xdel(winsid())//to close all currently open figure(s).

//This code uses shapes.jpg image file for processing


img = imread('shapes.jpg'); // Reads input image shapes.jpg
img = rgb2gray(img); // Converts input image to gray scale
clf // Clears figure handle
imshow(img); // Show gray scale image (Result file exp_9_1_result_1.png)

e = edge(img); // This performs edge detection operation with sobel, thresh = 0.5
figure(2); // Opens new figure
imshow(e) // Show result image (Result file exp_9_1_result_2.png)

e = edge(img, 'prewitt'); // thresh = 0.5 // Applied prewitt edge detection method
figure(3);
imshow(e) // Show result image (Result file exp_9_1_result_3.png)

```

```
e = edge(img, 'canny', [0.06  0.2]);
    // Applies canny edge detection method
figure(4);
imshow(e)
    // Show result image (Result file
    exp_9_1_result_4.png)

e = edge(img, 'fftderiv', 0.4);
    // Applies FFT gradient method; 0.4 threshold
figure(5);
imshow(e)
    // Show result image (Result file
    exp_9_1_result_5.png)
```

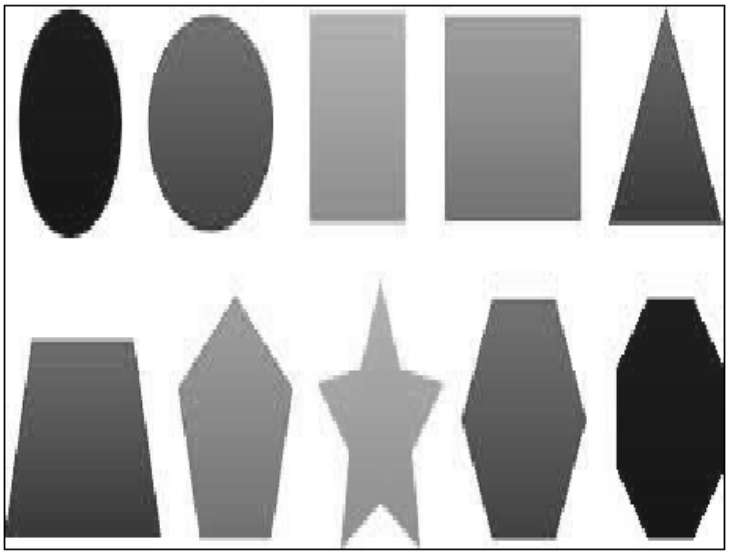


Figure 9.1: Example 1

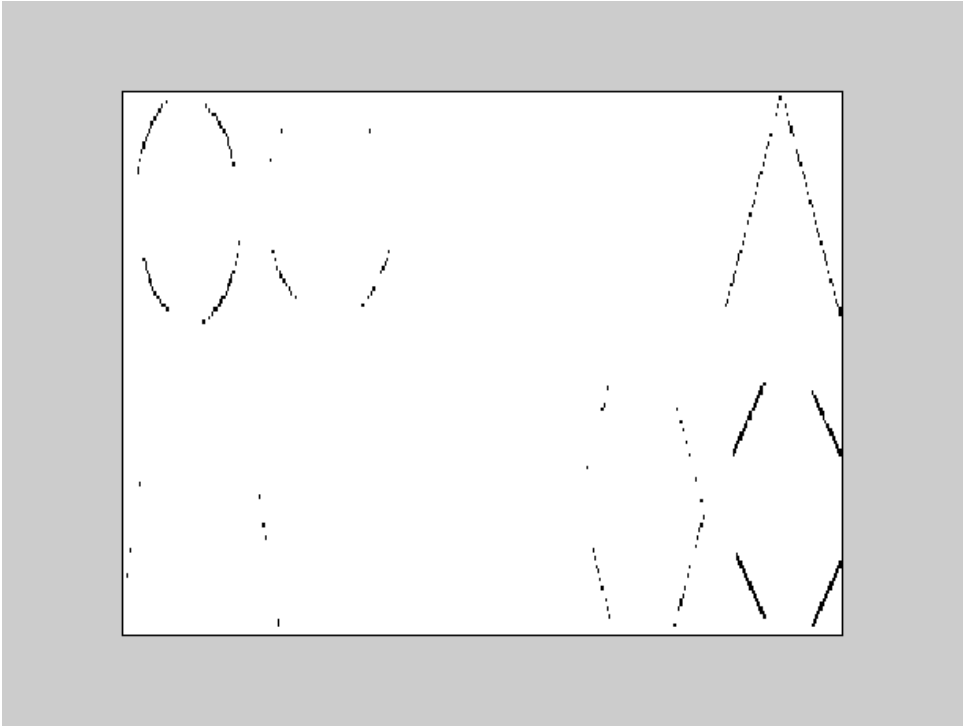


Figure 9.2: Example 1

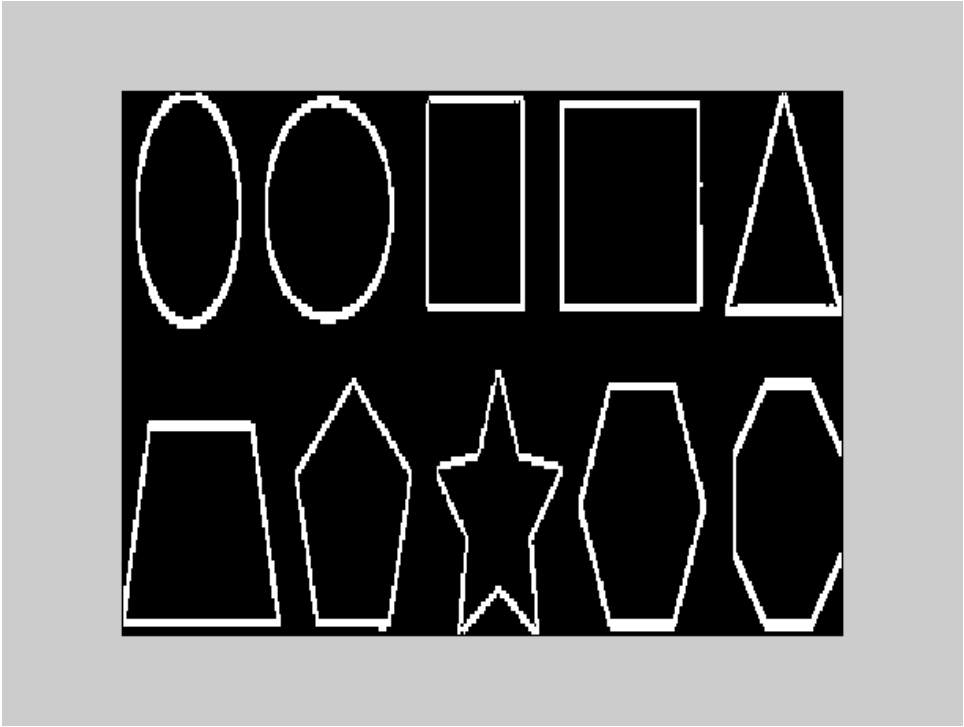


Figure 9.3: Example 1

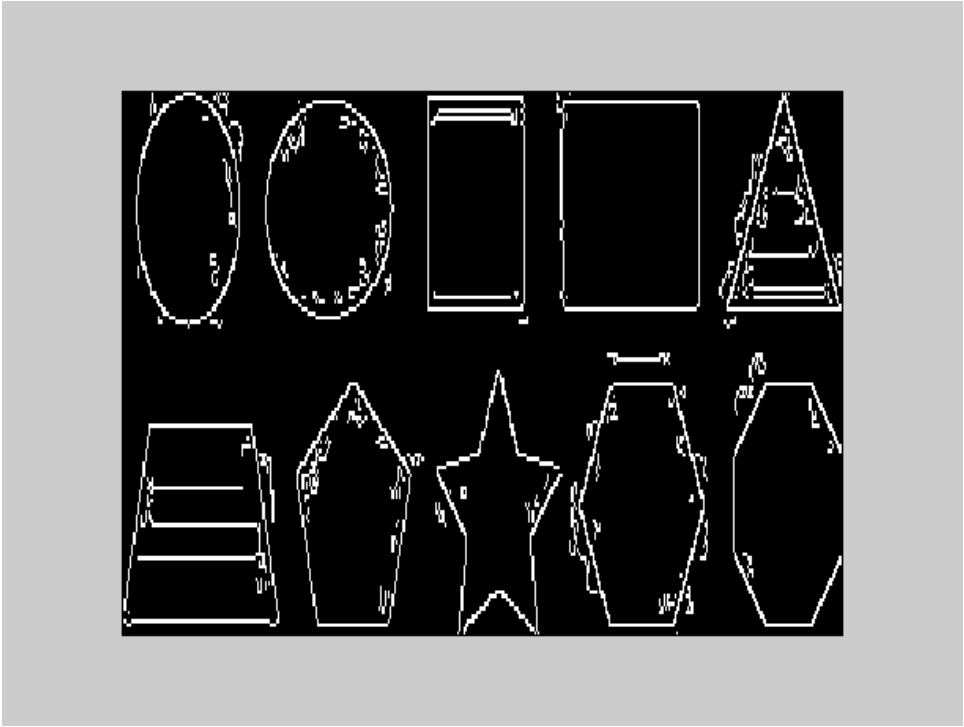


Figure 9.4: Example 1

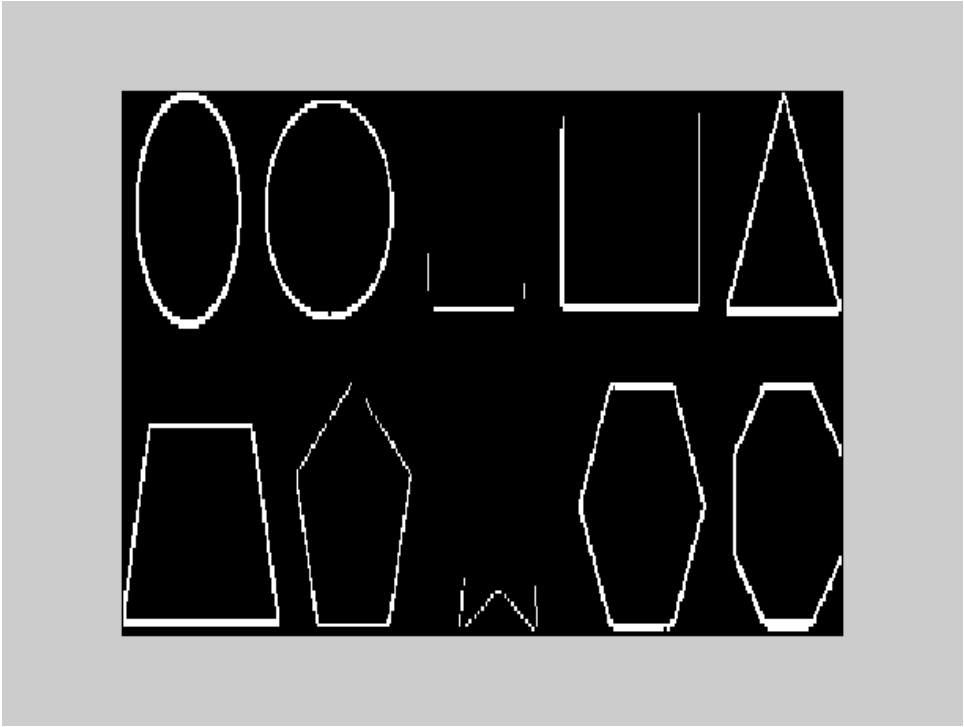


Figure 9.5: Example 1

# Experiment: 10

# Feature Extraction from Images

check Appendix AP 7 for dependency:

`shapes.jpg`

**Scilab code Solution 10.1** Example 1

```

// This scilab code is used to extract features from images
//using FAST algorithm (this algorithm is best suitable for corner detection)

//The scilab environment for this is : Scilab 6.0.1

//Toolbox used: Image Processing and Computer Vision Toolbox ver. 1.1


//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods
```

```
// Rosten. Machine Learning for High-speed Corner Detection, 2006.

//clc //to clear command window.
//clear all //to kill previously defined variables.
//xdel(winsid())//to close all currently open figure(s).

//This code uses shapes.jpg image file for processing


//S = imcreatechecker(8,8,[1 0.5]);
S=imread('shapes.jpg');                                    // Reads an input image
fobj = imdetect_FAST(S);                                   // Extract features from image using FAST algorithm
imshow(S);                                                 // Displays original image
plotfeature(fobj);                                         // Plots the extracted features on the image shown in figure
```

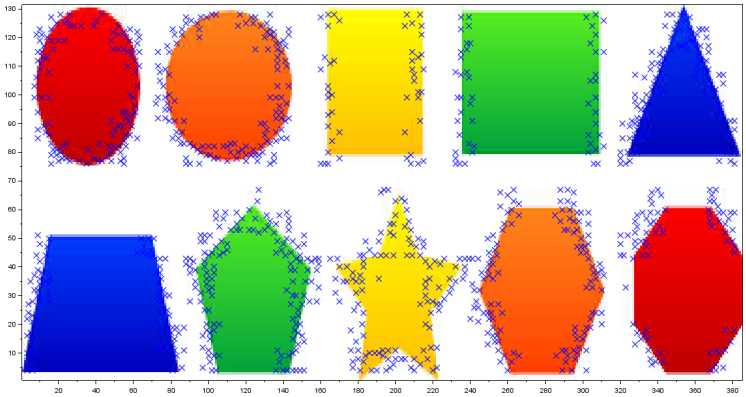



Figure 10.1: Example 1

# Experiment: 11

# Basics of OCR

check Appendix AP 4 for dependency:

`number.png`

check Appendix AP 5 for dependency:

`region.png`

check Appendix AP 6 for dependency:

`text.png`

**Scilab code Solution 11.1** Example 1

```

// This scilab code is used to perform basic operation such as binarization of images
//that contains text with random value of threshold

//The scilab environment for this is : Scilab 6.0.1

//Toolbox used: Image Processing and Computer Vision Toolbox ver. 1.1

```

```

//OS used : Windows 10 64 bit
//
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

//clc //to clear command window.
clear all //to kill previously defined variables.
//xdel(winsid())//to close all currently open figure(s).

//This code uses text.png, region.png and number.png image files for processing

img = imread('text.png');                    // Reads input image
threshold=0.93;                              // set a random threshold value
imagen =~im2bw(img,threshold);                 // Convert input image to binarized image with given threshold
                                             // and also negate the output
imshow(imagen);                              // Shows the resultant image (exp_11_1_result_1.png)

img = imread('number.png');                    // Reads input image
threshold=0.93;                              // set a random threshold value
imagen =~im2bw(img,threshold);                 // Convert input image to binarized image with given threshold
                                             // and also negate
```

```
                                                the
                                                output
imshow(imagen);                            // Shows the
    resultant image (exp_11_1_result_2.png)

img = imread('region.png');                // Reads
   input image
threshold=0.93;                            // set a
   random threshold value
imagen =~im2bw(img,threshold);             // Convert
    input image to binarized image with given
   threshold
                                           // and also
                                                negate
                                                the
                                                output
imshow(imagen);                            // Shows the
    resultant image (exp_11_1_result_3.png)
```

Figure 11.1: Example 1

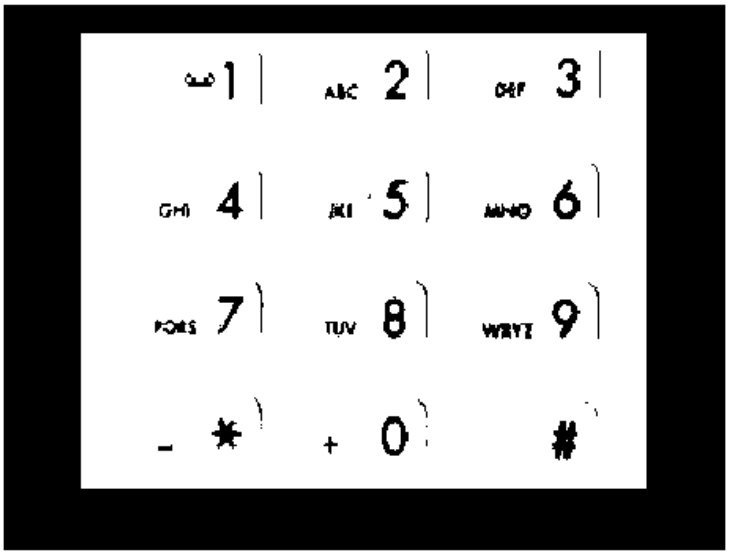



Figure 11.2: Example 1

Figure 11.3: Example 1

# Experiment: 12

# Face Detection Algorithm Implementation

check Appendix AP 1 for dependency:

`group.png`

check Appendix AP 2 for dependency:

`kalpana.jpg`

check Appendix AP 3 for dependency:

`random.jpg`

**Scilab code Solution 12.1** Example 1

```

// This scilab code is used to detect the human face
    in the image

//The scilab environment for this is : Scilab 5.5.2

//Toolbox used: SIVP 0.5.3.2

```

```
//OS used : Windows 7 32 bit
//
//Reference book name : Digital Image Processing
//book author: Rafael C. Gonzalez and Richard E. Woods

//clc //to clear command window.
//clear all //to kill previously defined variables.
//xdel(winsid())//to close all currently open figure(s).

//This code uses different images having human faces for processing
// Images used:
//    kalpana.jpg
//    random.jpg
//    group.png

img = imread('kalpana.jpg');                                // Reads input image

faces = detectfaces(img);    // Uses function in the toolbox to detect faces
[m,n] = size(faces);    // Calculates number of faces in image

for i=1:m,    // This block draws rectangles for each face images detected
    img = rectangle(img, faces(i,:), [0,255,0]);
end;

imshow(img);    // Shows resultant images with the rectangle around face images

pause;
```

```
    // Pause till user enters resume command at
    command window
img = imread('random.jpg');
    // Reads input image

faces = detectfaces(img);
    // Uses function in the toolbox to detect faces
[m,n] = size(faces);
    // Calculates number of faces in image

for i=1:m,
    // This block draws rectangles for each face
    images detected
     img = rectangle(img, faces(i,:), [0,255,0]);
end;

imshow(img);
    // Shows resultant images with the rectangle
    around face images
pause;
    // Pause till user enters resume command at
    command window
img = imread('group.png');
    // Reads input image

faces = detectfaces(img);
    // Uses function in the toolbox to detect faces
[m,n] = size(faces);
    // Calculates number of faces in image

for i=1:m,
    // This block draws rectangles for each face
    images detected
     img = rectangle(img, faces(i,:), [0,255,0]);
end;

imshow(img);
    // Shows resultant images with the rectangle
```

Figure 12.1: Example 1

around face images

Figure 12.2: Example 1

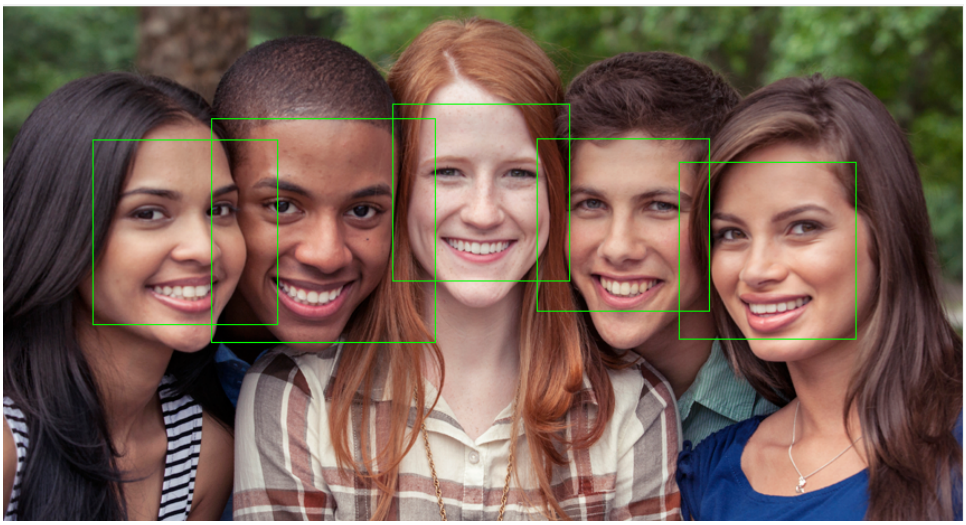


Figure 12.3: Example 1

# Appendix

groupimage

kalpanachawla

randomimage

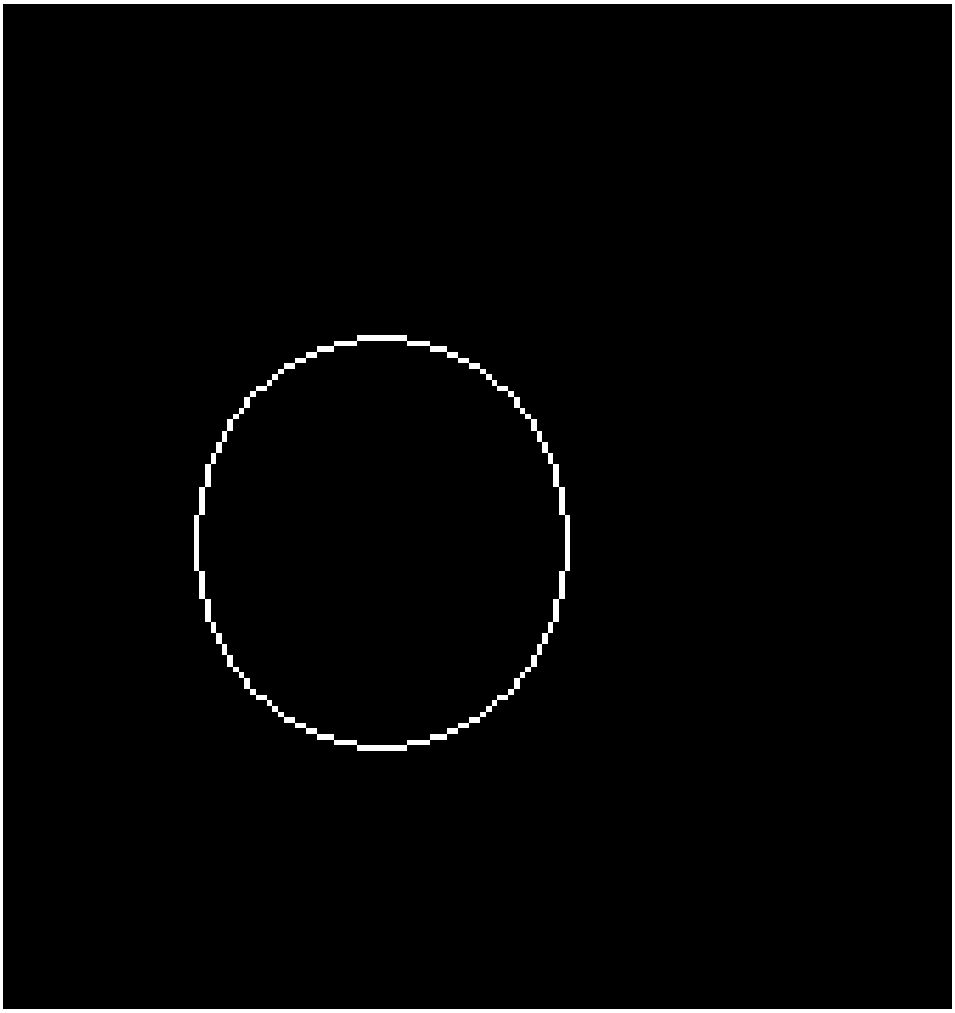
numberimage

regionimage

YES, it's true M Ivan WINKLER! R
guaranteed chances to win a shar
Lump-Sum, TAX-FREE CASH. Plu
Australian Lotto Multi-Millionaire,
granted you WIN-PLUS PRIORITY. textimage

shape

lennaimage



circleimage

harewood

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Text

Image

inputimage