

Scilab IPCV represents images in a few formats. The 3 basic types of images supported in Scilab are:

1. Binary images: Boolean  $\rightarrow$  %f, %t
2. Gray Scale images: uint8  $\rightarrow$  0 to 255, dounle  $\rightarrow$  0 to 1
3. RGB images: m-by-n-by-3 matrix, could be uint8 and double

To see how Scilab imports and handles images, let's see following tutorial.

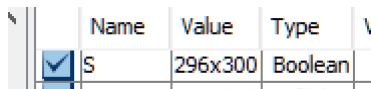
## Tutorial 1: Image Representation in Scilab

While Scilab importing binary image, it will be represented in 2 values,%f for black and %t for white.

### Binary Image

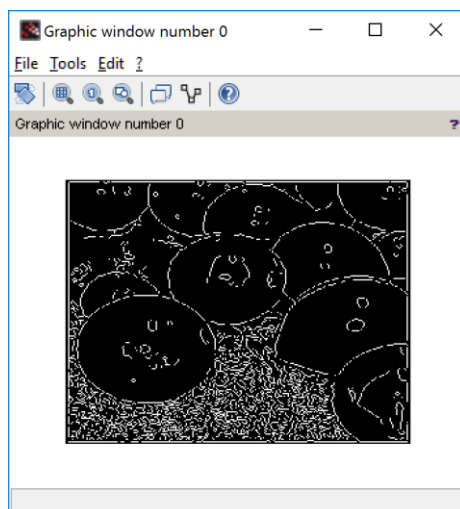
1. Change to the working directory where the images located, for e.g.:  
**--> `cd 'pathto/PCVfiles';`**
2. Read the image using "imread" command  
**--> `S1 = imread('balloons_binary.png');`**

From the variable browser, you should see the variable S1 created as an 296x300 boolean type. Double click the variable to launch the variable editor to see the values of the pixel in the image matrix.



	Name	Value	Type	V
<input checked="" type="checkbox"/>	S	296x300	Boolean	

3. Use "imshow" command to visualize the imported image  
**--> `imshow(S1);`**

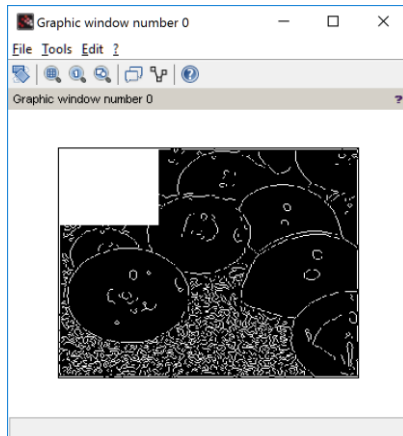


4. Try to manipulate the pixels value, for example, use the following command to change the pixels value to 255, which is correspond to white color.

```
--> S1(1:100,1:100) = %t;
```

5. Use the “imshow” function again to show S

```
--> imshow(S1);
```



We are actually adding a white patch on the upper left corner of the image.

### Gray Scale Image

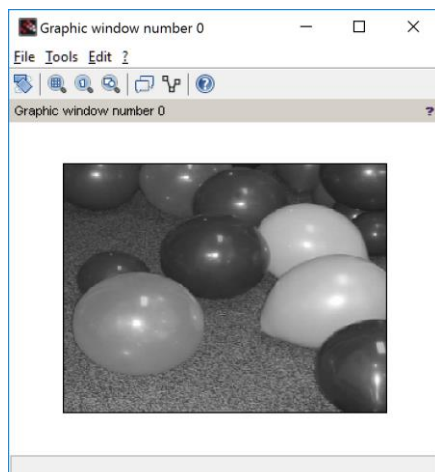
1. Now let's explore the gray scale image. Using the same command, read in a gray scale image from the image file.

```
--> S2 = imread('balloons_gray.png');
```

2. Double click on the variable S2 from the variable browser, you would see the values between 0 to 255, which represent the intensity of the image.

3. Use the “imshow” command to visualize the image.

```
--> imshow(S2);
```



### Color Image

1. Again, use “imread” command to import color image.

```
--> S3 = imread('balloons.png');
```

2. In Scilab 6, the hypermatrices are loaded natively. The variable browser will show the width and height of the image but not the depth.

	ans	296x300	Integer
int	S3	296x300	Integer
ans		1x1	String

3. To inspect the size of the image matrix, we could use “size” command

```
-->size(S3)
```

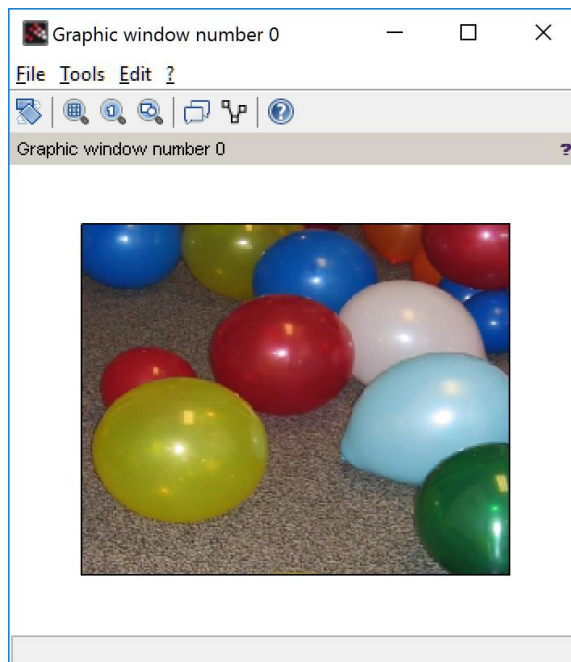
```
ans =
```

```
296. 300. 3.
```

Now we could see the matrix is now 3 layers, which representing R, G and B layers (Red, Green and Blue).

4. Use “imshow” to visualize the color image as well.

```
--> imshow(S3);
```



### Tutorial 2: Converting from Color to Gray and Binary image

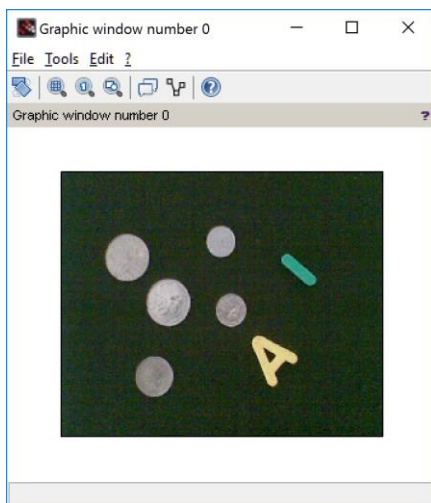
Most of the images acquired are in color or RGB format, which would need more processing power, memory and time to process. In a lot of cases, images could be converted to gray scale or binary for processing, which require less computing power.

In this tutorial, we are going to learn how to convert the color image to grayscale and binary image prior to the processing.

#### Converting from Color to Grayscale image

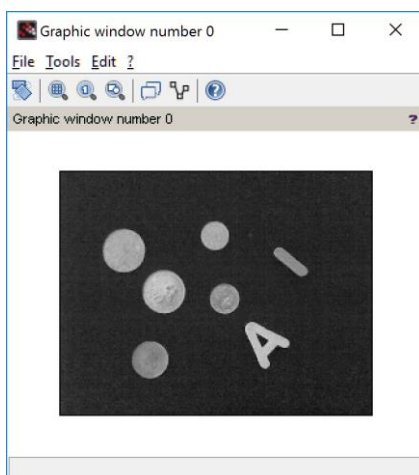
1. Now let's use the command we've learn to import and visualize another color image taken by handphone camera.

```
-->S = imread('measure.jpg');  
-->imshow(S);
```



2. Convert the color image to grayscale by using "rgb2gray" command.

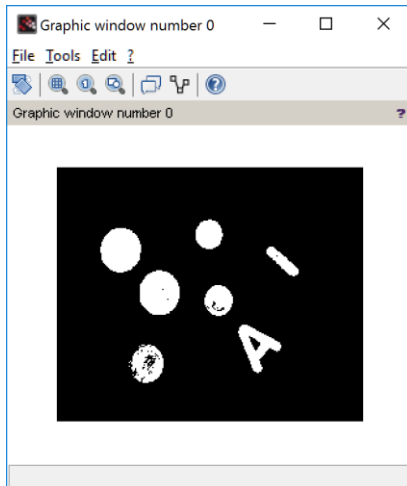
```
-->Sgray = rgb2gray(S);  
-->imshow(Sgray);
```



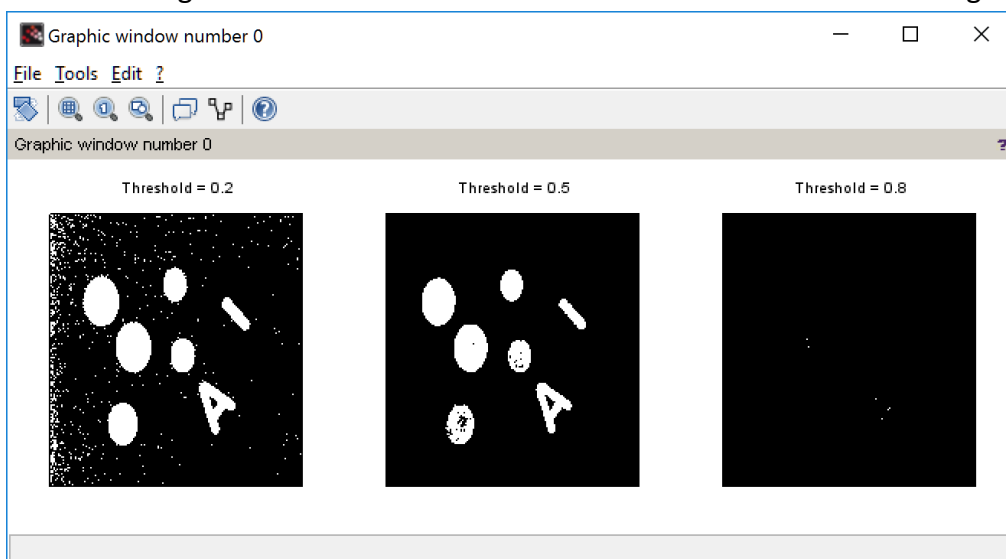
- Convert the grayscale image to binary image by using "im2bw".

```
-->Sbin = im2bw(Sgray,0.5);
```

```
-->imshow(Sbin);
```



The second argument of the "im2bw" command is the threshold value which is used to decide the level of intensity value to be converted to white and black. Different values would give different results. Some results are shown in the following figures:

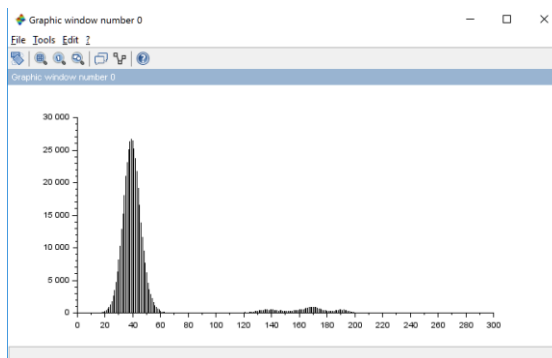


## Choosing Threshold Value

As seen in previous example, choosing a right threshold value is important to get a good binary image for further processing. First we try a manual way to select the threshold value by looking into the image histogram.

1. Use “imhist” command to compute and show the image histogram.

```
-->imhist(Sgray,[],1);
```



Since the background of the image is in darker color, the pixel value should be lower than the objects (coins and alphabets). From the histogram, we could roughly select the threshold value to differentiate background from objects.

2. Choose the value 90, as the threshold, normalisation is required to get the threshold value for the “im2bw” function.

```
-->90/255
```

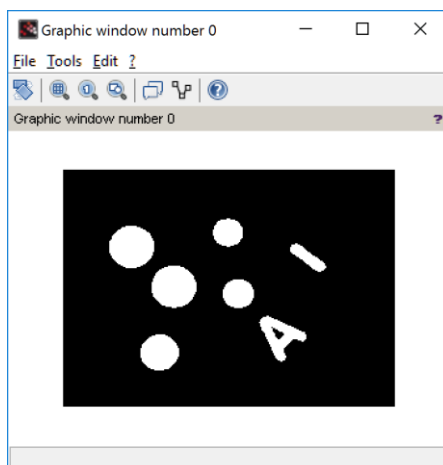
```
ans =
```

```
0.3529412
```

3. Use the new threshold value for the conversion to binary image.

```
-->Sbin = im2bw(Sgray,0.353);
```

```
-->imshow(Sbin);
```



4. We could also use the function from the Scilab Image Processing module to find the threshold value automatically.

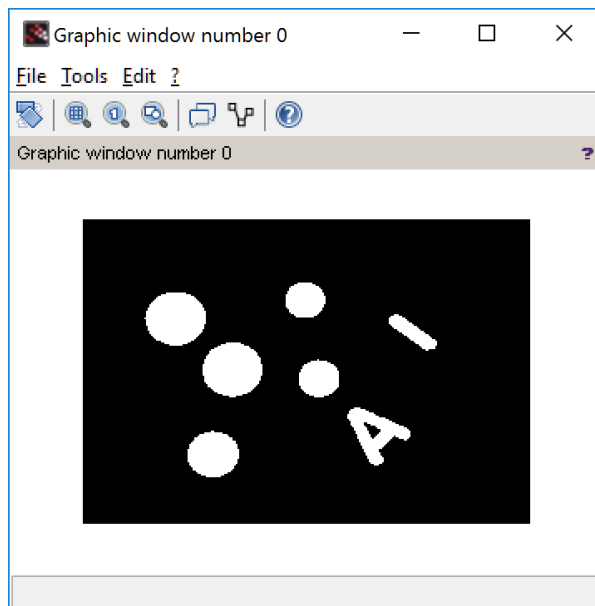
```
-->th = imgraythresh(Sgray)
```

```
th =
```

```
0.3984375
```

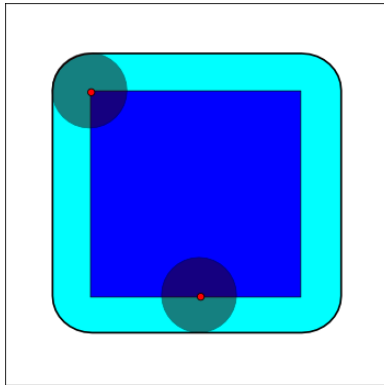
```
-->Sbin = im2bw(Sgray,th);
```

```
-->imshow(Sbin);
```



### Tutorial 3: Morphological Operation on Binary Image

The two principal morphological operations are dilation and erosion. Dilation allows objects to expand, thus potentially filling in small holes and connecting disjoint objects. Erosion shrinks objects by etching away (eroding) their boundaries. These operations can be customized for an application by the proper selection of the structuring element, which determines exactly how the objects will be dilated or eroded.



#### Dilation and Erosion

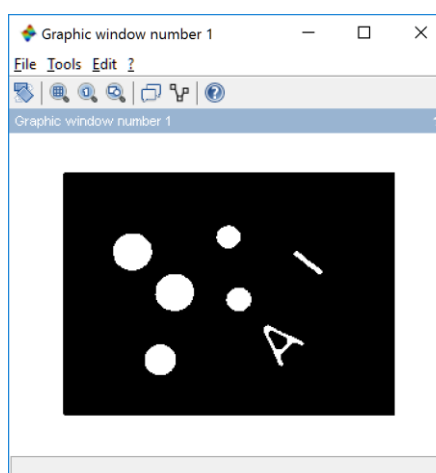
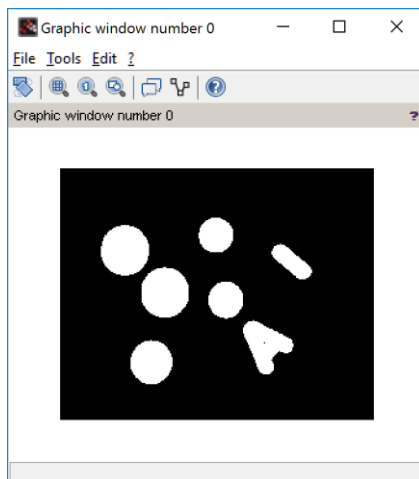
1. Using the output from previous tutorial, *Sbin*, we are going to try the morphological operator.
2. First, we create a structure element using “*imcreatese*” command.  
`-->se = imcreatese('ellipse',15,15);`
3. Using the “*imdilate*” and “*imerode*” command, we perform dilation and erosion on the binary image. Show and compare the results with the original binary image.

```
-->Sd = imdilate(Sbin,se);
```

```
-->imshow(Sd);
```

```
-->Se = imerode(Sbin,se);
```

```
-->scf();imshow(Se);
```





### Opening and Closing

Dilation and Erosion would increase and reduce the components size. In order to preserve the size of the components, these 2 operations are always used together to preserve the size.

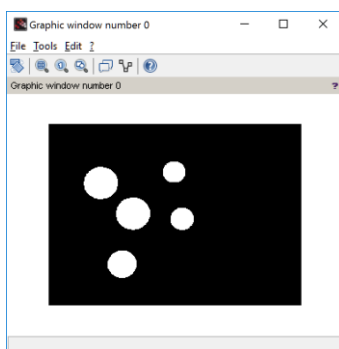
Opening operation is the process of erosion followed by dilation. The usage of this operation is to remove smaller components or components which are not the desired shape.

1. Create a structure element.
2. Perform opening operation and show the result.

```
-->se = imcreate('ellipse',35,35);
```

```
-->So = imopen(Sbin,se);
```

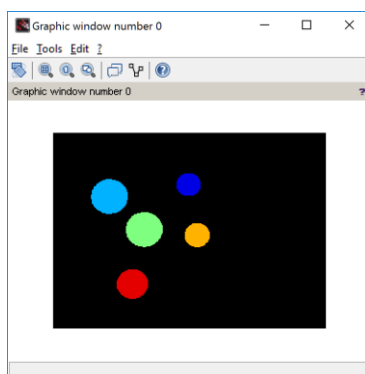
```
-->imshow(So);
```



### Find Number of Objects in An Image

1. Numbers of objects could be found by using “imlabel” function.
2. S\_labeled is the labelled components, for example, first component pixels value are all marked as ‘1’, second marked as ‘2’, etc. n is the numbers of labelled objects in the image. We could visualize them using following extra parameter in imshow command.

```
-->imshow(S_labeled,jetcolormap(5));
```



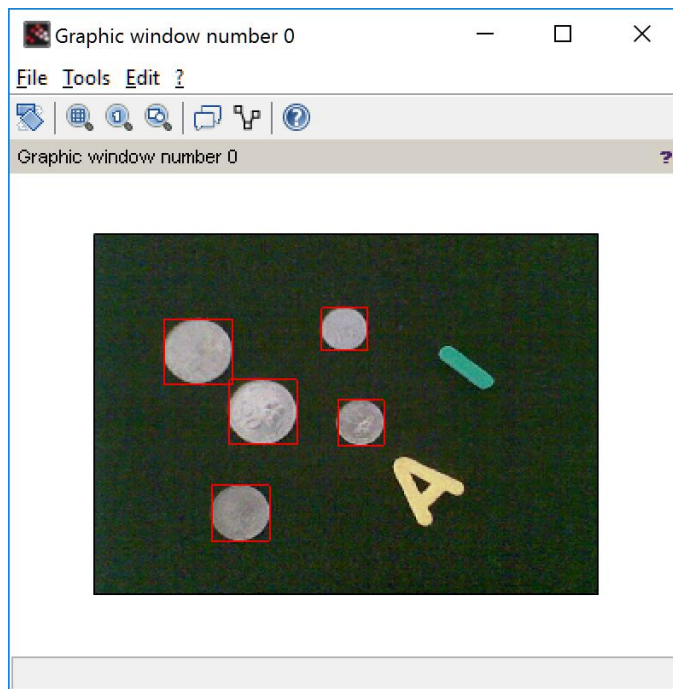
3. Function “`imblobprop`” is used to find the area and bounding box for all components in labelled image.

```
-->[A, BB] = imblobprop(S_labeled);
```

4. Use the bounding boxes to show the detected components on the original image.

```
-->imshow(S);
```

```
-->imrects(BB,[255 0 0]);
```



## Tutorial 4: Convolution and Correlation --> Image filtering and Template matching

In image processing, most of the time the used of convolution and correlation for filtering is more to personal preferences, as they perform almost the same operation. They are identical if the kernel is symmetrical.

In general, we use convolution for image smoothing, while correlation for template matching.

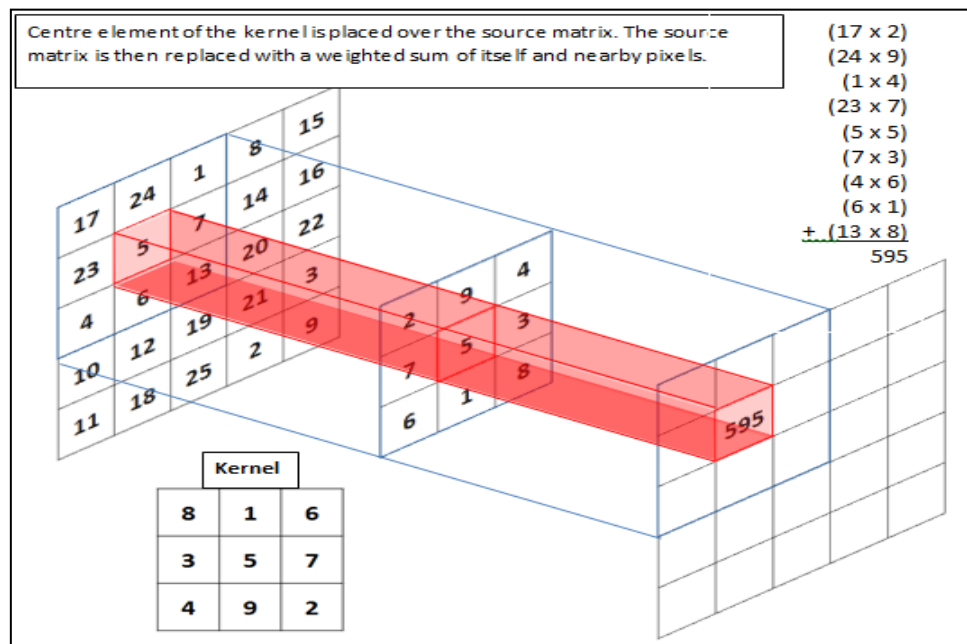


Figure 1: Convolution Operation for an Image and Kernel

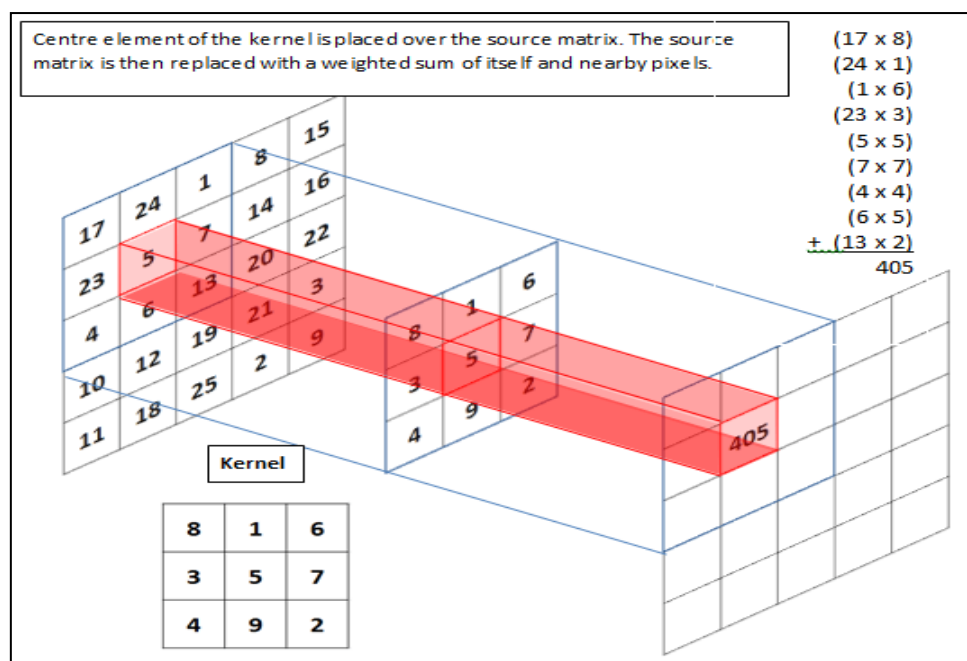


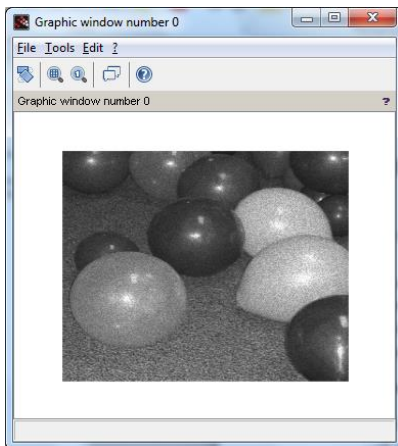
Figure 2: Correlation Operation for an Image and Kernel

There are couples of functions in Scilab could be used for convolution or correlation, such as function “conv2” and “filter2”. However, these 2 functions from Scilab core functions only support image of double data type. So in our example, we are going to use a function from Image Processing module, which is “imfilter”. By default, imfilter performs correlation.

### Convolution as Image Filtering

1. Load and visualise an image which is distorted by noise.

```
-->S = imread('balloons_noise.png');  
-->imshow(S);
```

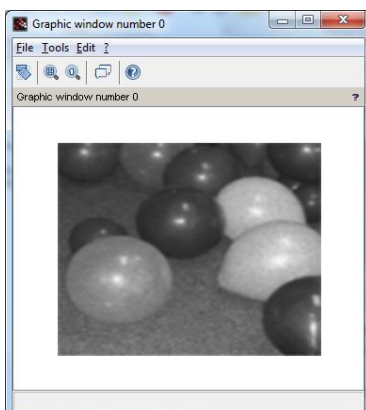


2. Create an filter kernel, which will perform averaging operation on the image.

```
-->h = 1/25.*ones(5,5)  
h =  
0.04 0.04 0.04 0.04 0.04  
0.04 0.04 0.04 0.04 0.04  
0.04 0.04 0.04 0.04 0.04  
0.04 0.04 0.04 0.04 0.04  
0.04 0.04 0.04 0.04 0.04
```

3. Apply the filter kernel to the image by using “imfilter” function.

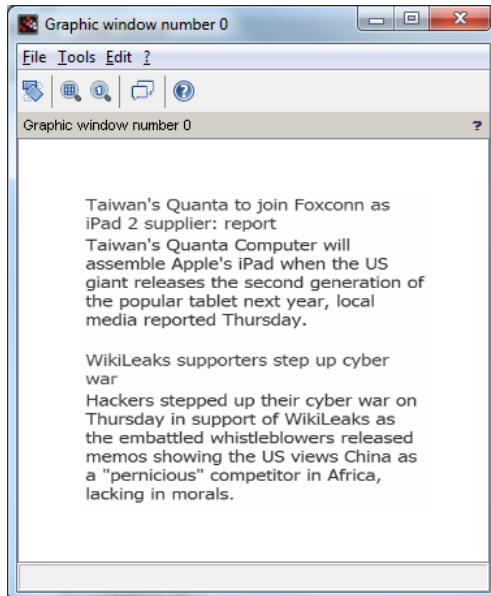
```
-->S2 = imfilter(S,h);  
-->imshow(S2);
```



### Correlation as Template Matching

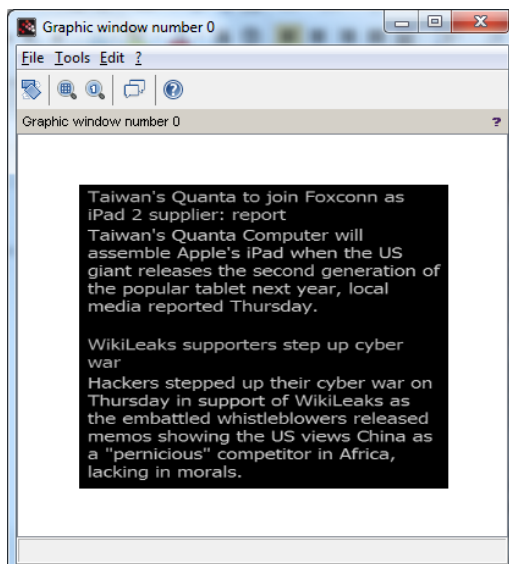
1. Load the image of an article and visualize it.

```
-->S = imread('text.png');  
-->imshow(S);
```



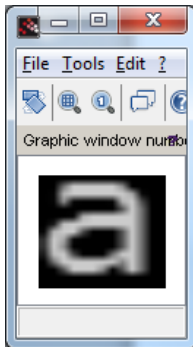
2. In image processing, we used to treat the object as white and background as black, hence, we convert the image to negative image by using "imcomplement" function.

```
-->S2 = imcomplement(S);  
-->imshow(S2);
```



- Now, load a template of letter a which is cropped from the original text as the kernel, and convert to its complement image.

```
-->h = imread('a.png');  
-->h2 = imcomplement(h);  
-->imshow(h2);
```

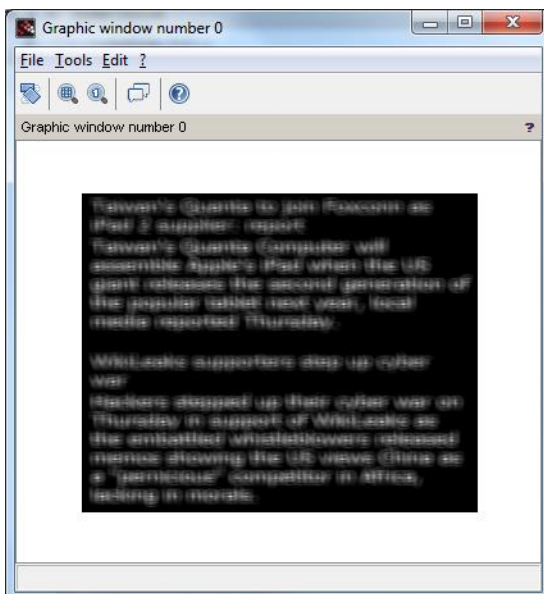


- Convert both image and template (kernel) to double data type.

```
-->S3 = im2double(S2);  
-->h3 = im2double(h2);
```

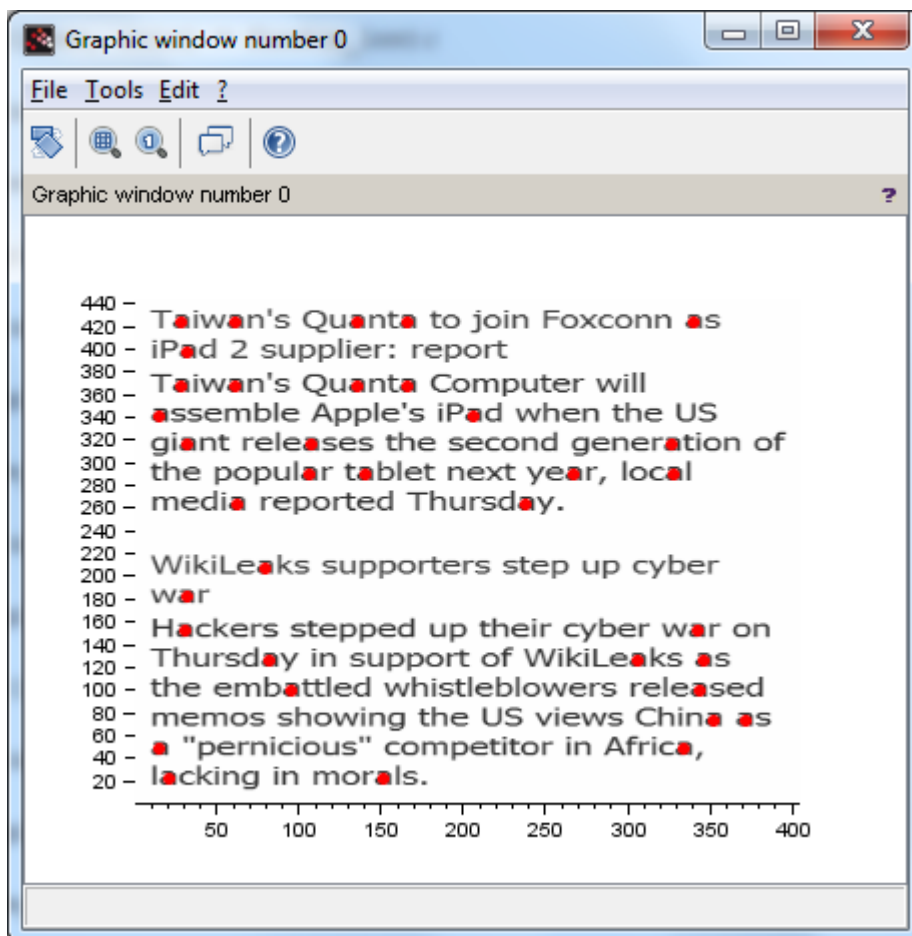
- Use the “imfilter” function to perform correlation, then normalizes and shows the image.

```
-->S4 = imfilter(S3,h3);  
-->S5 = imnorm(S4);  
-->imshow(S5);
```



6. Do note that the letter which is the most “correlated” to the template would have highest intensity. In ideal condition, the location should return a value of ‘1’ in normalized image. In practical, we set a threshold of 0.9 to identify the location of letter ‘a’ (template).

```
-->a_loc = S5 > 0.9;
-->[rows, cols] = find(a_loc);
-->imshow(S);
-->sz = size(S);
-->plot(cols,sz(1)-rows,'r.');
```



More tutorial coming up at : <http://scilabipcv.tritytech.com>