Exercise: Simulating spatial processing in the visual pathway with convolution

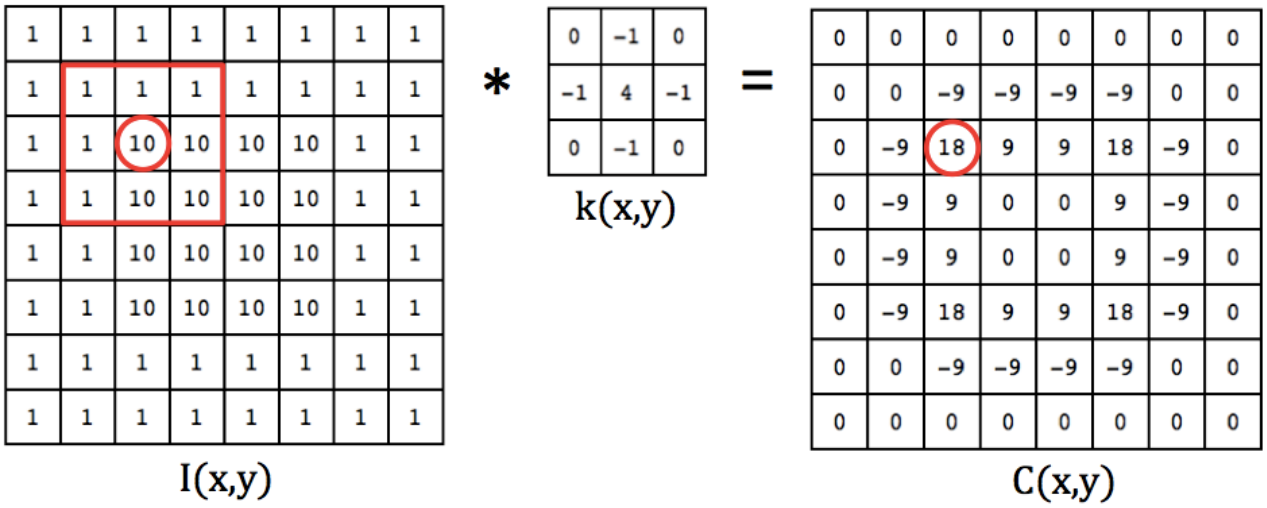
This problem uses *convolution* to simulate the spatial filtering performed by neurons in the early stages of the visual pathway, from the retina to cortical area V1. The solution code uses built-in MATLAB®functions, `meshgrid` and `conv2`, and functions for loading and displaying images from the MATLAB Image Processing Toolbox, `imread` and `imshow`.

Part 1: Two-dimensional convolution

Given an image $I(x,y)$ and kernel $k(x,y)$, the convolution operation is formally defined as

$$C(x,y) = I(x,y) * k(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} I(m,n)\, k(x-m, y-n)$$

In practice, convolution is performed over finite windows. Given the simple image and kernel shown below, the convolution operation involves centering the kernel on each image location and computing a weighted sum of the corresponding values in the image, with weights specified by the kernel (also referred to as the convolution *operator* or *mask*). Centering the kernel on the image value 10, circled in red, yields a convolution value of 18. In this example, the convolution is not computed at locations around the image border.



(a) Create a matrix containing the kernel below by directly entering the values, as in:

```
kernel = [1 2; 3 4];
```

0.1	0.3	0.5	0.3	0.1
0.3	0.7	0.9	0.7	0.3
0.5	0.9	1.0	0.9	0.5
0.3	0.7	0.9	0.7	0.3
0.1	0.3	0.5	0.3	0.1

(b) An image of coins comes with the Image Processing Toolbox. Load this image into MATLAB using `imread` and convert the values from type `unit8` (integers from 0 to 255) to floating point using the `double` function:

```
image = double(imread('coins.png'));
```

(c) Convolve the `image` with `kernel` using the `conv2` function, for example:

```
result = conv2(image, kernel, 'valid');
```

If the option `'valid'` is specified as the third input, MATLAB only computes convolution values for locations at which the entire kernel fits within the image. Compare the sizes of the `image` and `result` matrices.

(d) Use `subplot` to display `image` and `result` in a 1x2 arrangement. Any 2D matrix can be displayed with the `imshow` function:

```
imshow(image, [])
```

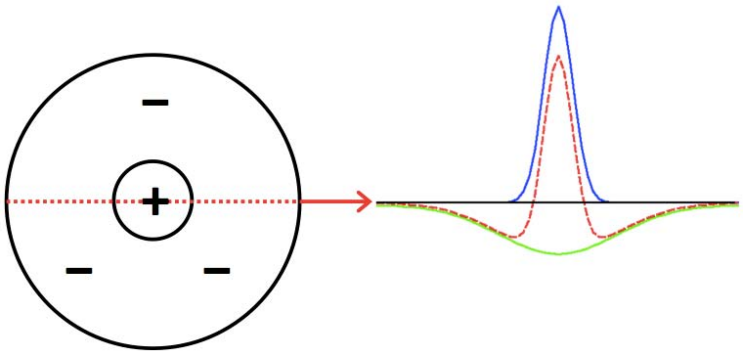
If an empty vector `[]` is given as the second input, MATLAB displays the smallest image value as black, the largest as white, and intermediate values spread over shades of gray. Explicit minimum and maximum values can be specified with a two-element vector, for example:

```
imshow(image, [150 200])
```

Note that the convolution result is just a slightly blurry version of the original image.

Part 2: Gaussian kernel

The optics of the eye blurs the incoming image slightly before light is captured by photoreceptors. The next stage of retinal processing is performed by *bipolar cells*, whose receptive fields have a spatial structure that can be described as the difference between two circularly symmetric Gaussian functions. Bipolar cells effectively compute a convolution of the photoreceptor array with a kernel that combines a narrow central Gaussian and broader Gaussian of opposite sign, creating a *center-surround* receptive field, as shown in the figure below. A neuron with this receptive field structure is excited by flashing light in the center of its receptive field and inhibited by flashing light in the surround (*on-center cell*). *Off-center cells* have the opposite structure. The magnitude of the response to a light stimulus at different locations in the receptive field is conveyed by the cross-section shown in red:



This spatial receptive field is preserved in the output of retinal processing carried by the retinal ganglion cells, and subsequent processing by the lateral geniculate nucleus.

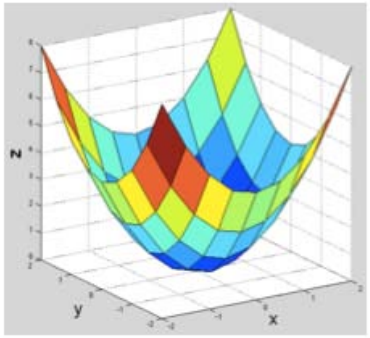
In this part of the exercise, you will define a function that creates a Gaussian kernel, and observe the results of smoothing an image with Gaussians of different size. We begin with the following definition of a 2D Gaussian function:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ specifies the spread of the Gaussian.

(a) Define a function `gauss2D` that has a single input corresponding to σ and returns a matrix containing a Gaussian kernel given by the above expression. Use `meshgrid` to generate two matrices that contain the x and y coordinates, respectively, of regularly spaced locations on a 2D grid. The following code illustrates the use of `meshgrid`:

```
values = -2:0.5:2;
[x y] = meshgrid(values);
z = x.^2 + y.^2;
surf(x, y, z)
```



Run the above example and view the contents of the matrices `x` and `y`. In your `gauss2D` function, use `meshgrid` to generate coordinates in the range from -2σ to $+2\sigma$ in increments of 0.5. The function `exp` can be used for the exponential.

(b) Use `gauss2D` to create Gaussian kernels with $\sigma = 1$ and $\sigma = 2$. Use `conv2` to convolve the coins image with each kernel and display the image and convolution results using `subplot` and `imshow`. How do the results differ for the two kernel sizes?

Part 3: Laplacian of Gaussian kernel

The difference-of-Gaussians structure of the receptive fields of neurons in the early visual pathway can be approximated by the *Laplacian* of a 2D Gaussian function. The Laplacian of a function $f(x, y)$ is defined as the sum of the second partial derivatives with respect to x and y. Applied to the 2D Gaussian function, this operation yields the following (∇^2 denotes the Laplacian):

$$\nabla^2 G = \left(\frac{1}{\sigma^2}\right)\left(\frac{x^2 + y^2}{\sigma^2} - 2\right)e^{-\frac{x^2+y^2}{2\sigma^2}}$$

(a) Define a function `lapGauss` that has a single input corresponding to σ and returns a matrix containing the Laplacian-of-Gaussian kernel given by the above expression. Use `meshgrid` to generate coordinates in the range from -3σ to $+3\sigma$, in increments of 0.5.

(b) Use `lapGauss` to create kernels with $\sigma = 1$ and $\sigma = 2$. Use `conv2` to convolve the coins image with each kernel and display the image and convolution results using `subplot` and `imshow`. How do the results differ for the two kernel sizes?

(c) The Laplacian embodies a second derivative operation. The second derivative changes sign (crosses zero) at the locations of intensity changes in the original image. The process of finding the locations of these intensity changes is often referred to as *edge detection*. To get a quick idea of where the zero-crossings of the convolution occur, display the results from Part b with `imshow`, specifying a minimum value of 0 and maximum value close to 0:

```
imshow(result, [0 0.001])
```

Part 4: Gabor kernel

The spatial receptive field of simple cells in cortical area V1 resembles the *Gabor* function, which is defined as the product of a 2D Gaussian and an oriented sine-wave function. The Gabor function can be expressed mathematically as follows:

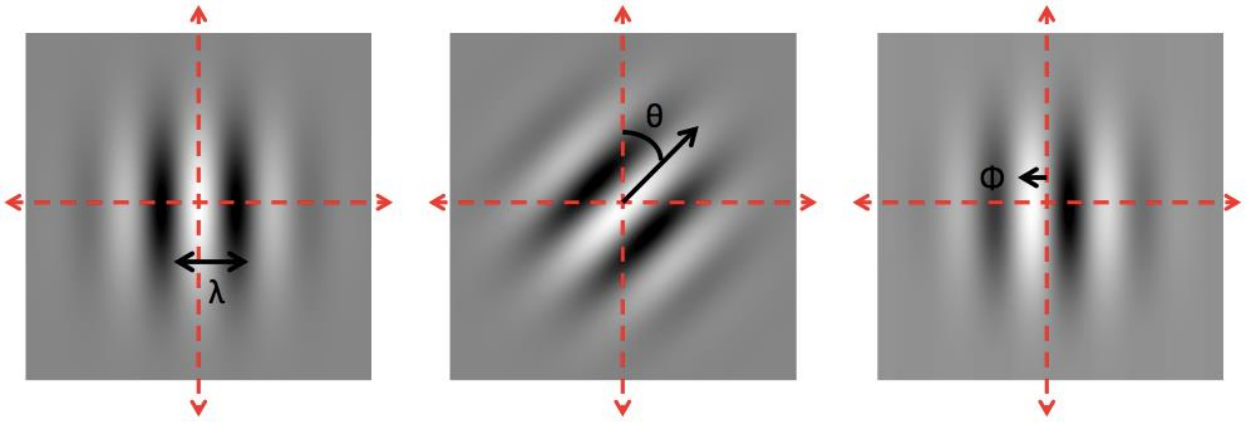
$$Gabor(x,y) = e^{-\frac{x_r{}^2 + y_r{}^2}{2\sigma^2}} \cos\left(2\pi\left(\frac{x_r}{\lambda}\right) + \psi\right)$$

where

$$x_r = x\cos(\theta) + y\sin(\theta)$$

$$y_r = -x\sin(\theta) + y\cos(\theta)$$

θ specifies the orientation of the Gabor, and λ and ψ are the wavelength and phase (shift) of the sine wave, respectively, as shown in the following diagram:



(a) Define a function `Gabor` that has four inputs corresponding to $\sigma, \theta, \lambda,$ and $\psi,$ and returns a matrix containing the Gabor kernel specified by the above expression. Use `meshgrid` to generate coordinates in the range from -3σ to $+3\sigma$, in increments of 0.5. For the sine and cosine functions, either use `sin` and `cos` (input angle is specified in radians), or `sind` and `cosd` (input angle is specified in degrees).

(b) Use `Gabor` to create kernels with four different orientations $(0°, 45°, 90°, 135°)$ and with $\sigma = 2$, $\lambda = 4$, and $\Psi = 0°$. Use `conv2` to convolve the coins image with each kernel and display the four convolution results using `subplot` and `imshow` (note that a loop can be used to implement these steps compactly). How do the results differ for the four orientations?

Resource: Brains, Minds and Machines Summer Course

Tomaso Poggio and Gabriel Kreiman