Mind, Brain, and Models 2023

CNCR

Computational vision

In this workshop, you will learn to blur images, perform a furrier transform of the image, and enhance edges.
Grading: A correct and full answer to each question (preceded by an asterisk) is worth one point.

Task 1: Gaussian blurring filter
Read the notes below and try the commands in Matlab:

- Remember that the equation for a 2d Gaussian is: $G=exp(-(x^2+y^2)/2\sigma^2)/2\pi\sigma^2$
- Load the image face.png using `imread()` and then convert it to a floating point image using `mat2gray()`. E.g. `A=mat2gray (imread('filename.type','type'));`
- You can view images using `imshow()` but generally you also needs to use `mat2gray()` to make the image more presentable: `imshow(mat2gray(A));`
- When using `conv2()` put the image first and the filter second and use the 'same' option. This crops the final image to the same size as the original image. eg `C=conv2(A,G,'same');`
- Meshgrid: `[X,Y]=meshgrid(x,y)` produces two arrays that together specify the x & y co-ordinates of an image relative to the specified axes. If x and y are n-vectors that specify the image axes then X contains the x co-ordinates of a notional n by n image and Y contains the y co-ordinates of the same notional image. Hence if x=y=[-1,0,1] then `[X,Y]=meshgrid(x,y)` will produce the following arrays:

```
X=    -1  0  1          Y=    -1  -1  -1
      -1  0  1                 0   0   0
      -1  0  1                 1   1   1
```

  `X` and `Y` can then be used in future array operations to supply values for x and y thus avoiding the use of multiple nested for loops to specify `x` and `y` values. Suitable ranges for the axis vectors `x` and `y` can be constructed as follows `x=-(s-1)/2:(s-1)/2` where `s` is the side length (`size`) of the filter in pixels, similarly for `y`. This places zero in the middle of the kernel.

Once you have read these notes, write a <u>function</u> `gaussianfilter()` using `meshgrid()` and `exp()` to make a 2D Gaussian function with some specified sigma and apply it to the `face.png` image using `conv2()`. It should be possible to specify the filter `sigma` and the overall size of the filter separately although generally size should be an integer `6 x sigma`.

Filter the `face.png` image with the following Gaussian filters

```
sigma =  5, size =  30
sigma = 15, size =  90
sigma = 30, size = 180
```

Save the `sigma=30` filtered image using `imwrite()` as follows:
`imwrite(uint8(mat2gray(image).*255),'filename.type','type');`

* Report the results, describe the effects of larger `sigma`s and determine at which step they correspond in biological processing.
* What do you notice about the time taken for each filtering operation and what impact might this have on a complex filter-based model: especially one where filter parameters are optimised by a cost function?
* Looking at the <u>filters</u> using `imshow()` describe how do they compare to retinal receptive fields.

Task 2: Working in Fourier domain
Make a new filter with `sigma = 30` as before but with size equal to the size of the image to be filtered (i.e., 2048). **<u>DO NOT USE THIS FILTER IN A CONV2() OPERATION</u>**.

Use `fft2()`, `iff2()` and `ifftshift()` to filter the image using this filter. You need to transform both the filter and the image into the Fourier domain using `fft2()`, then multiply the two Fourier images together using the `.*` form for multiply, then transform back to the space domain using `ifft2()` and finally use `ifftshift()` to correct the image.

* Compare the image produced this way with one produced using `conv2()`. What differences are there between the filtered images?
* What are the implications of this for modelling visual processes?
* What is the possible drawback of the Fourier method?

Task 3: Difference of Gaussian (DoG) filters: smoothed edge enhancement
Make a 2D filter that is the difference of two Gaussians one with a larger `sigma` than the other. It is important that the positive and negative parts of the filter be balanced (such that applying the filter does not lighten or darken the image overall). This will be achieved by the division of the Gaussian by `sigma^2` instead of the `2*pi*sigma^2` term of the formula at the beginning of the assignment.
Make filters with the following `sigma`s and apply them to the image through the convolution.

```
sigmaPositive = 1, sigmaNegative = 3
sigmaPositive = 3, sigmaNegative = 9
sigmaPositive = 10, sigmaNegative = 30
```

*What features are picked up by the three filters and what is not picked up? How could you simply improve the situation?
* What is a biologically plausible method to implement filters at different scales?
* In which way a DoG filter mimics retinal processing and what would the sigmas correspond to?
* Referring to images from this workshop, describe the utility of DoG and Gabor filters as models of human vision.