Linear Image Filtering

Audiovisual Processing CMP-6026A

Dr. David Greenwood

david.greenwood@uea.ac.uk

SCI 2.16a University of East Anglia

November 22, 2021

Content

- 2D Convolutions
- Smoothing Filters
- Sharpening and Unsharp Masking
- Template Matching

What is Image Filtering?

Filtering replaces each pixel with a value based on some function performed on it's *local neighbourhood*.

What is Image Filtering?

Used for smoothing and sharpening. . .

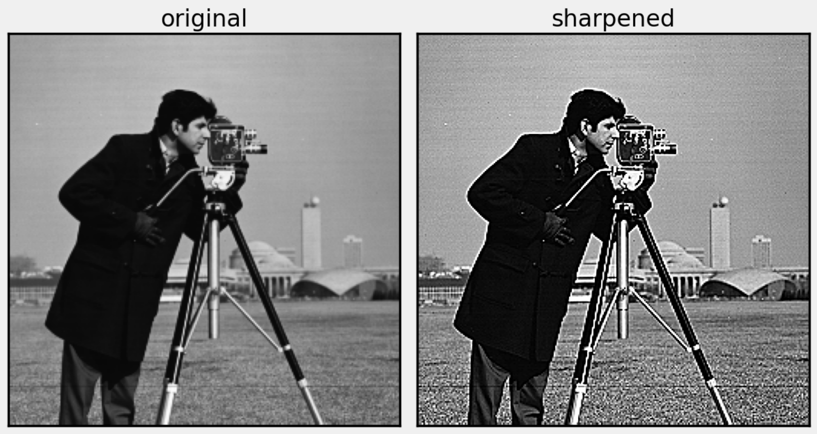


Figure 1: Sharpen Example

What is Image Filtering?

Estimating gradients. . .

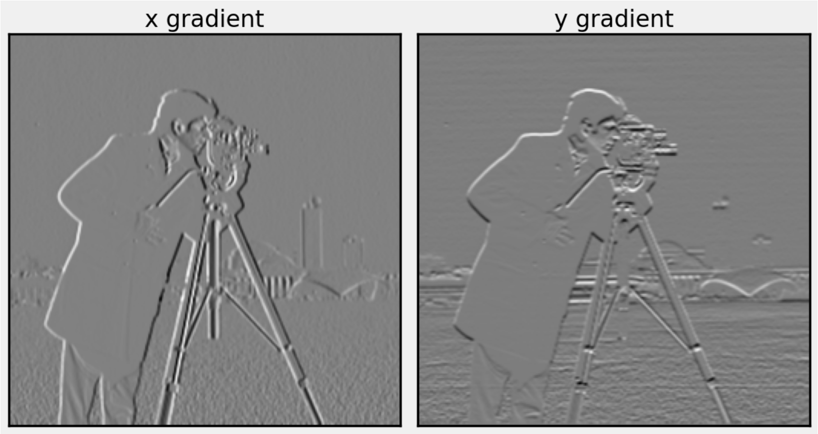


Figure 2: Gradient Example

What is Image Filtering?

Removing noise. . .

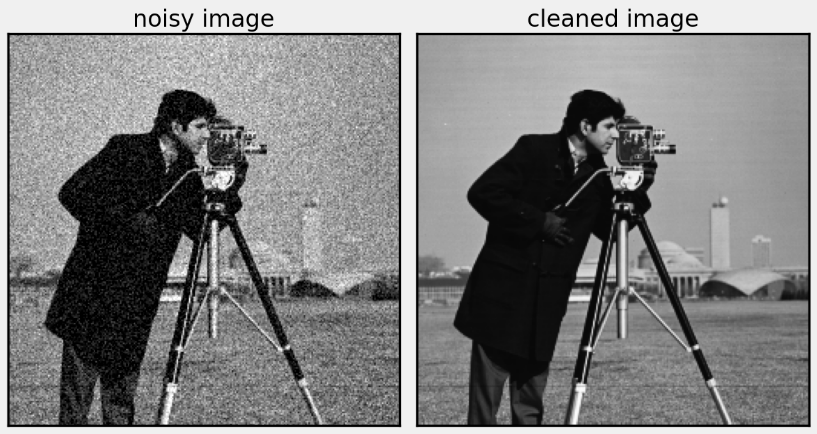


Figure 3: Noise Example

Linear Filtering

Linear Filtering is defined as a convolution.

This is a *sum of products* between an image region and a **kernel** matrix:

$$g(i,j) = \sum_{m=-a}^{a} \sum_{n=-b}^{b} f(i-m, j-n)h(m,n)$$

where g is the filtered image, f is the original image, h is the kernel, and i and j are the image coordinates.

Convolution

Typically:

$$a = \lfloor \frac{h_{rows}}{2} \rfloor, \;\; b = \lfloor \frac{h_{cols}}{2} \rfloor$$

So for a 3x3 kernel:

$$\text{both } m, n = -1, 0, 1$$

Kernel Matrix

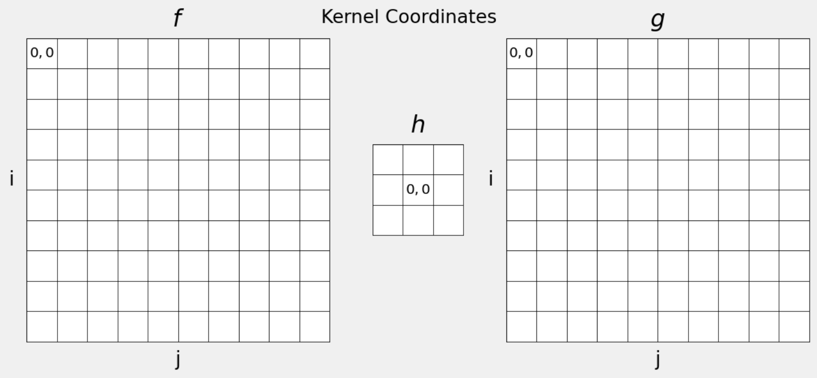


Figure 4: The kernel origin is in the *centre*.

Convolution

1. Scan image with a sub-window centred at each pixel.
   - The sub-window is known as the kernel, or mask.
2. Replace the pixel with the sum of products between the kernel coefficients and all of the pixels beneath the kernel.
   - Sum of products only for linear filters
3. Slide the kernel so it's centred on the next pixel and repeat for all pixels in the image.

Convolution

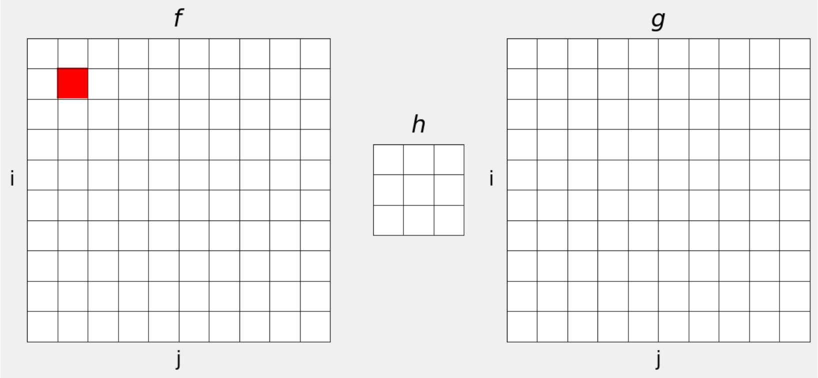


Figure 5: The kernel is positioned at (1,1) in input image.

Convolution

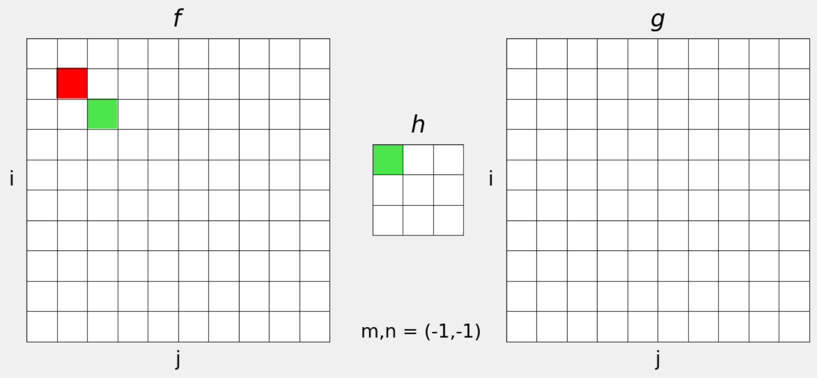


Figure 6: We iterate the values of *m* and *n*.

Convolution

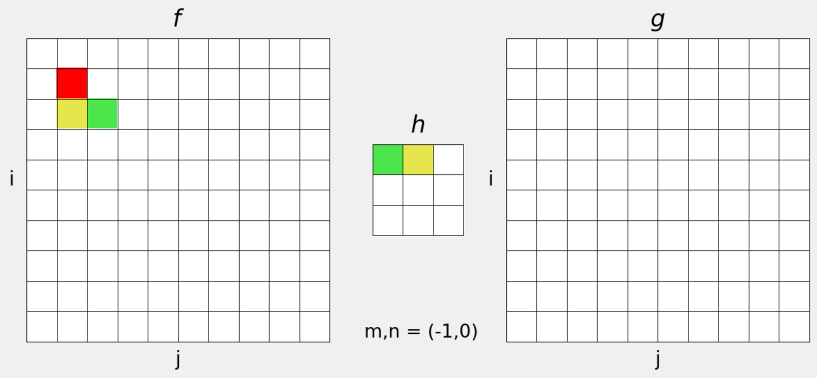


Figure 7: $m = -1, \; n = 0$

Convolution

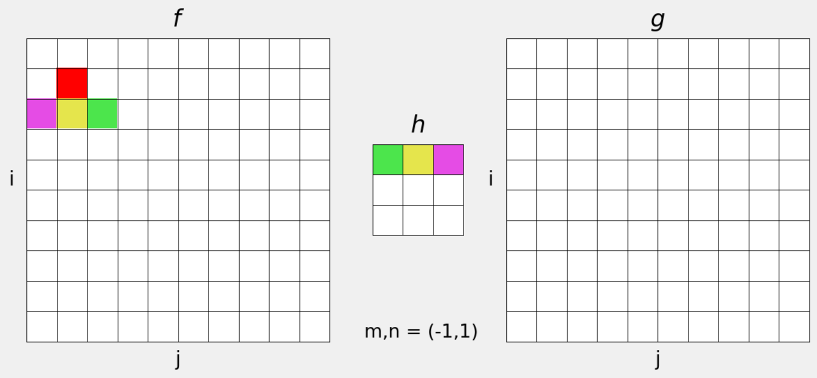


Figure 8: $m = -1, \; n = 1$

Convolution

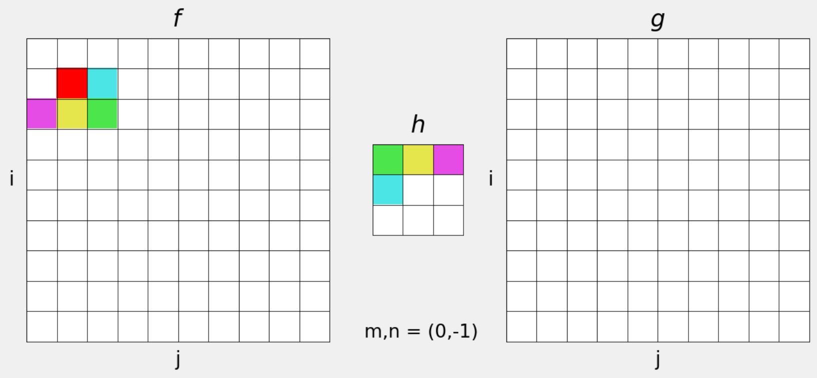


Figure 9: $m = 0, \ n = 1$

Convolution

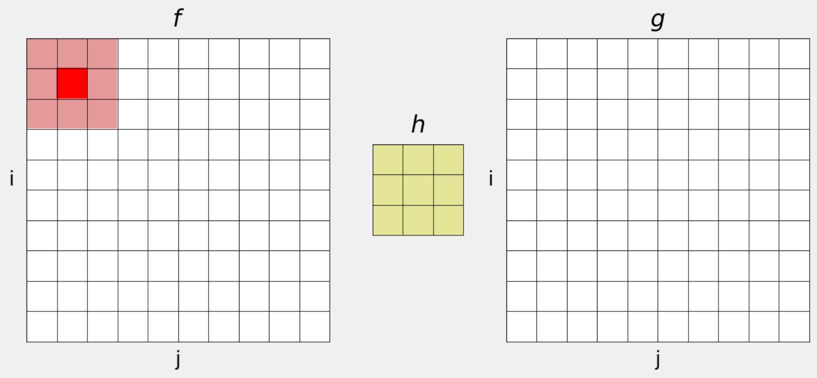


Figure 10: Iteration is complete.

Convolution

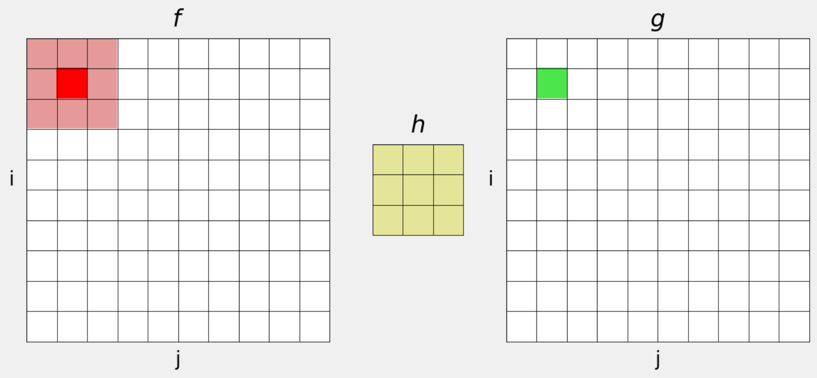


Figure 11: The product sum is assigned to the output image.

Convolution

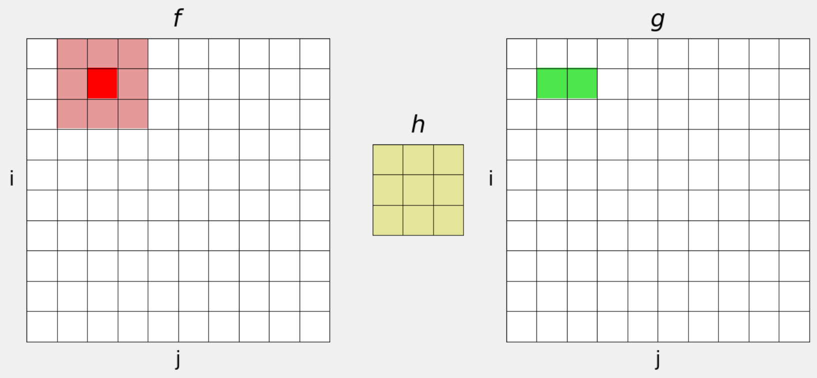


Figure 12: Slide the kernel along the row.

Convolution

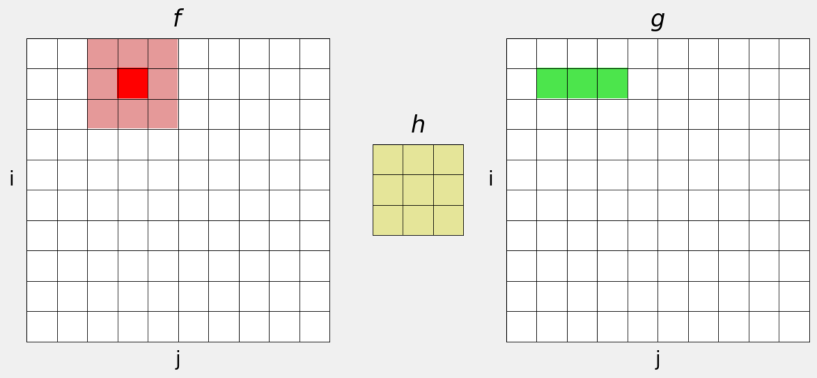


Figure 13: Slide the kernel along the row.

Convolution

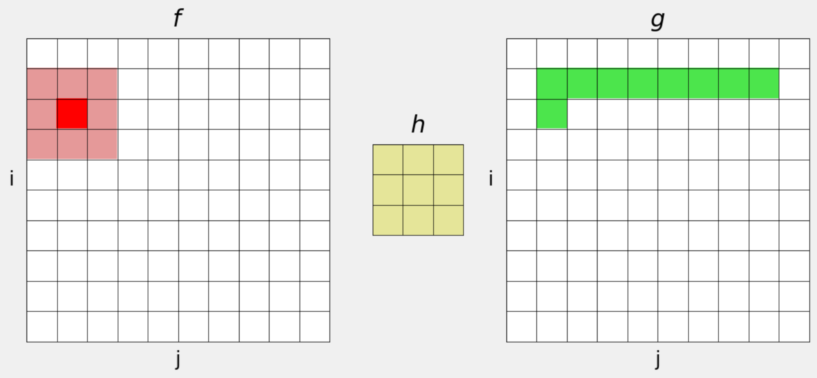


Figure 14: Move the kernel to the next row.

Convolution

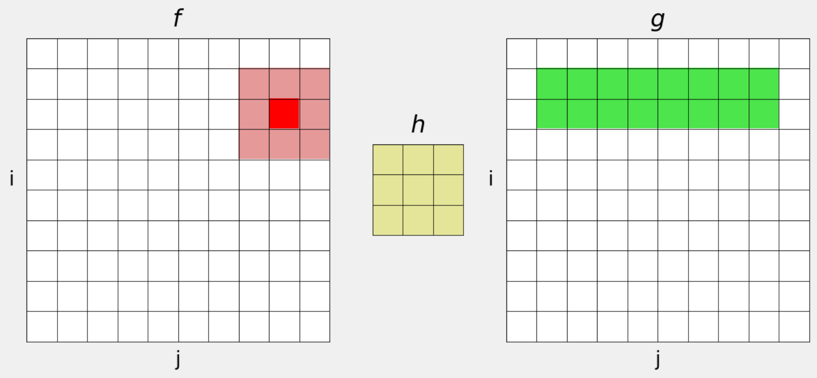


Figure 15: Continue sliding.

Convolution

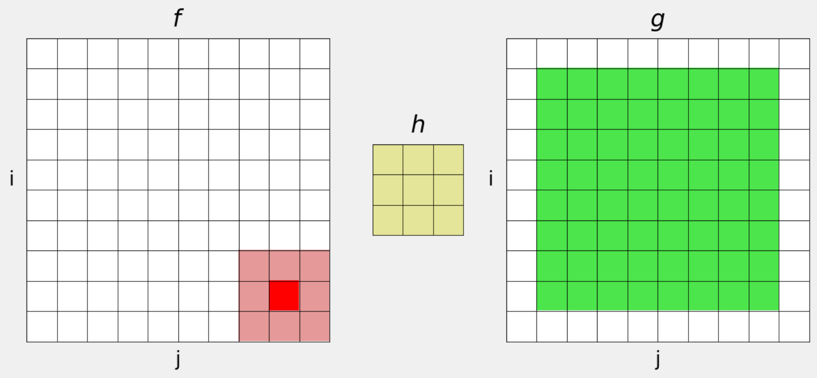
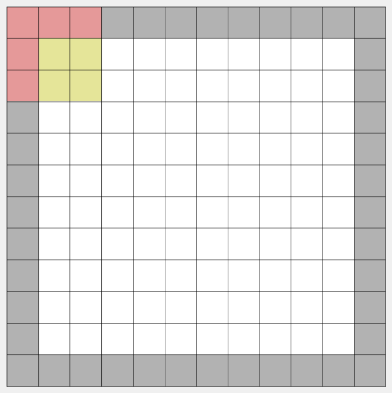


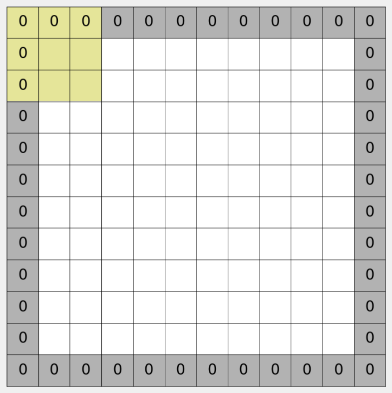
Figure 16: The image is completely covered.

What about the edges?



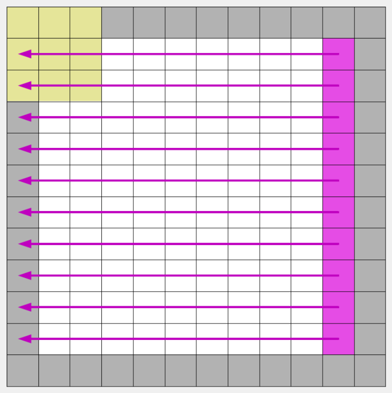
The filter window falls off the edge of the image.

What about the edges?



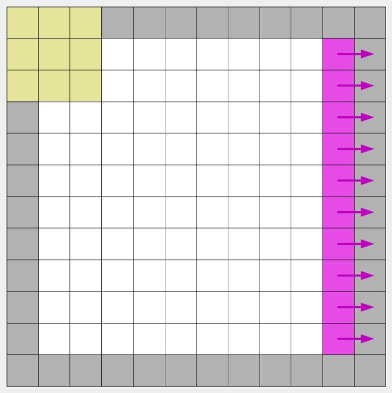
A common strategy is to pad with zeros.
The image is effectively larger than the original.

What about the edges?



We could *wrap* the pixels, from each edge to the opposite. Again, the image is effectively larger.

What about the edges?



Alternatively, we could *repeat* the pixels, extending each edge outward.

What would the filtered image look like?

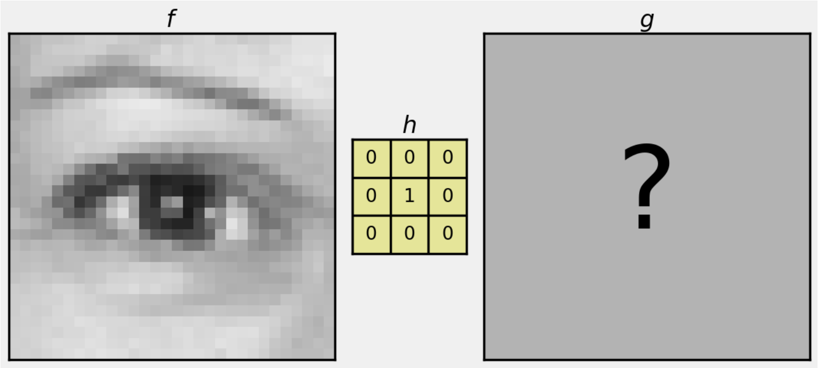


Figure 17: kernel 1

No change!

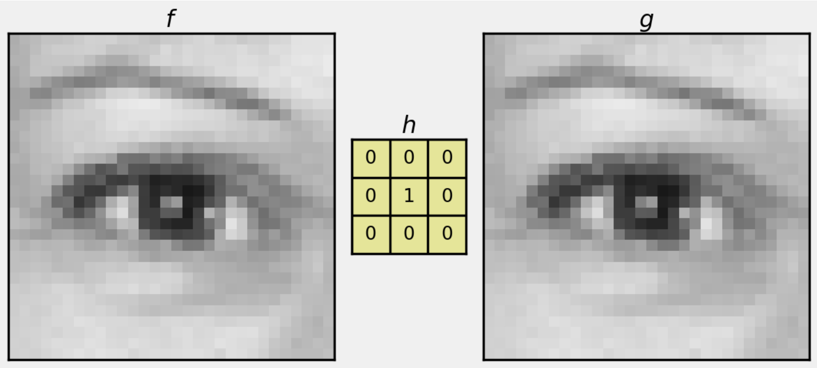


Figure 18: kernel 1

What would the filtered image look like?

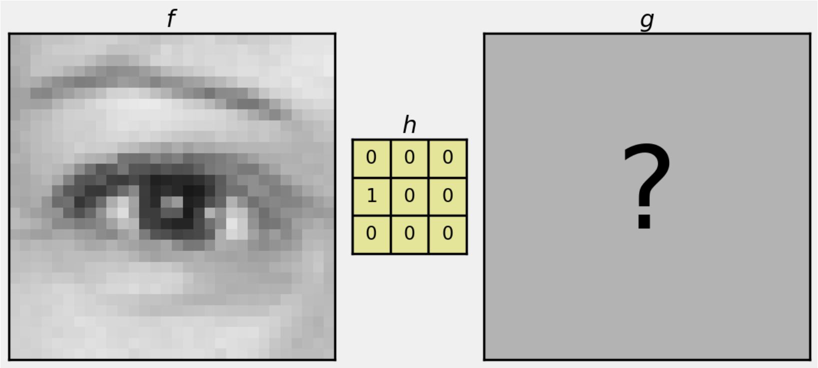


Figure 19: kernel 2

Shifted left by 1 pixel.

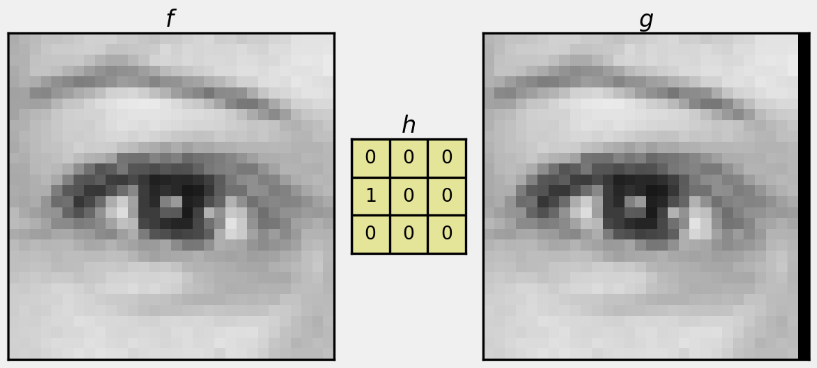


Figure 20: kernel 2

What would the filtered image look like?

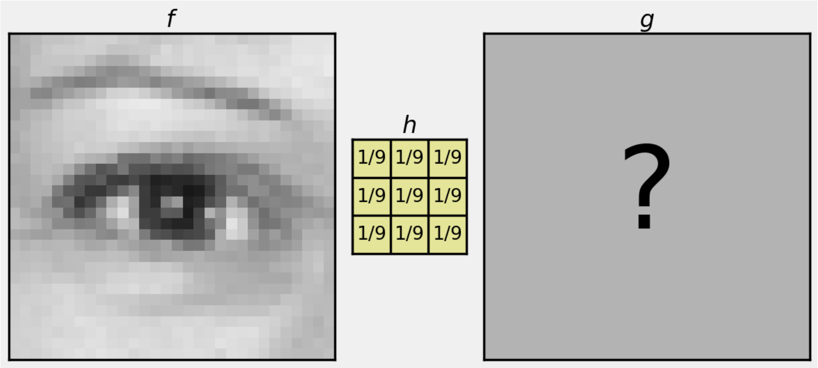


Figure 21: kernel 3

Blurred. . .

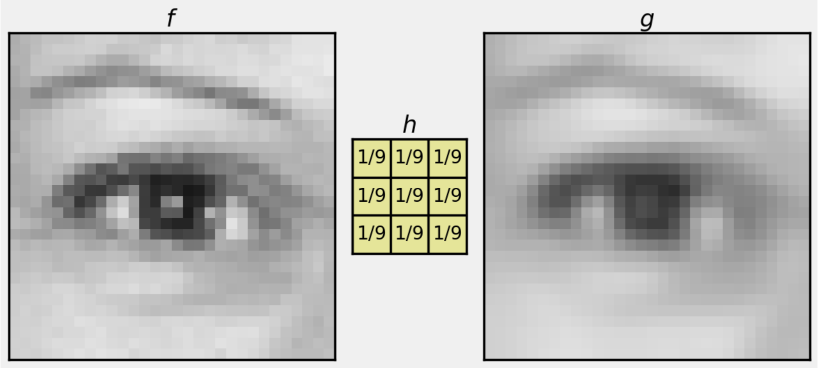


Figure 22: kernel 3

Mean Filter

replace each pixel with the mean of local neighbours:

$$h = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Mean Filter

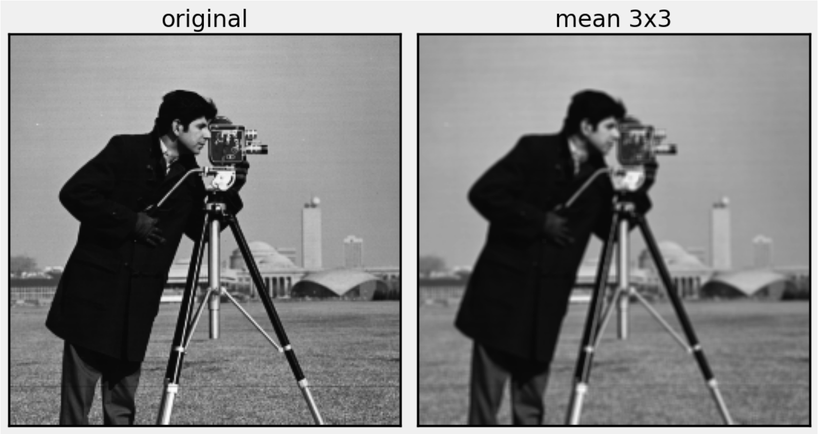


Figure 23: Mean filtered with 3 x 3 kernel

Mean Filter

we can increase the size of the kernel to get a smoother image:

$$h = \frac{1}{25} \times \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Mean Filter

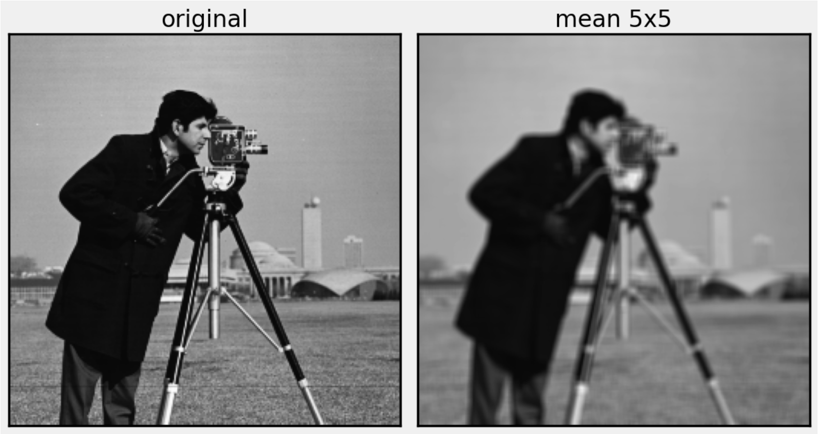


Figure 24: Mean filtered with 5 x 5 kernel

Mean Filter

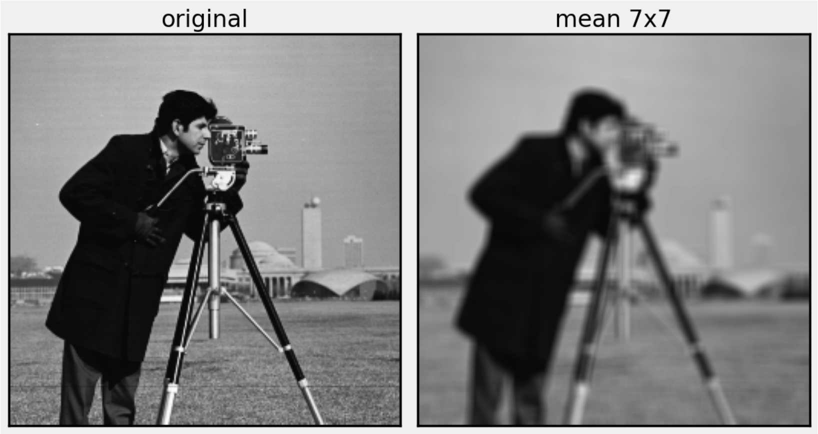


Figure 25: Mean filtered with 7 x 7 kernel

Gaussian blur

Similar to mean filter:

- Replace intensities with a weighted average of neighbours.
- Pixels closer to the centre of the kernel have more influence.

Gaussian blur

$$g(x, y) = \frac{1}{2\pi\sigma^2} \, \mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian blur

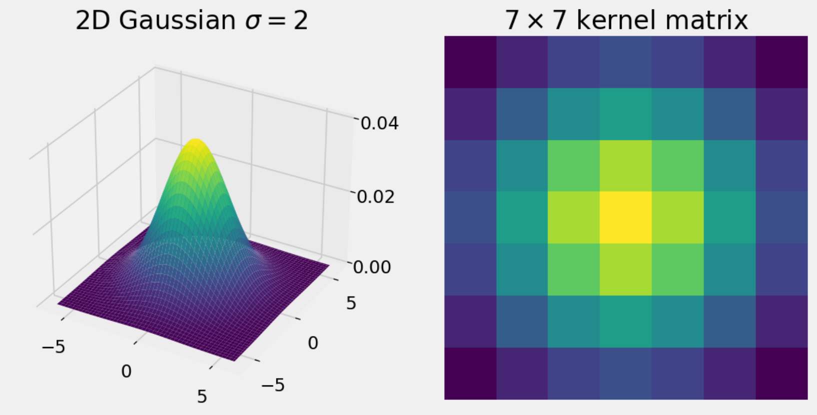


Figure 26: Gaussian kernel

Gaussian blur

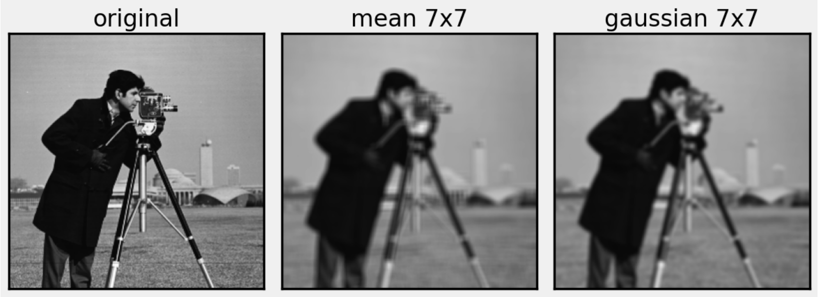


Figure 27: Mean and Gaussian blur

Image Smoothing

Smoothing effectively *low pass* filters the image.

- Only really practical for small kernels
- Blurring also destroys image information
- Difference between the mean and Gaussian filter is subtle, but Gaussian is usually preferred

Image Smoothing

If we have many images of the same scene:

- Use idea of averaging to reduce noise.
- Average pixel intensities across images rather than across the spatial neighbour.

Image Smoothing

- Effectively increases the signal-to-noise ratio.
- Useful in applications where image signal is low.
    - E.g., imaging astronomical objects.

What would the filtered image look like?

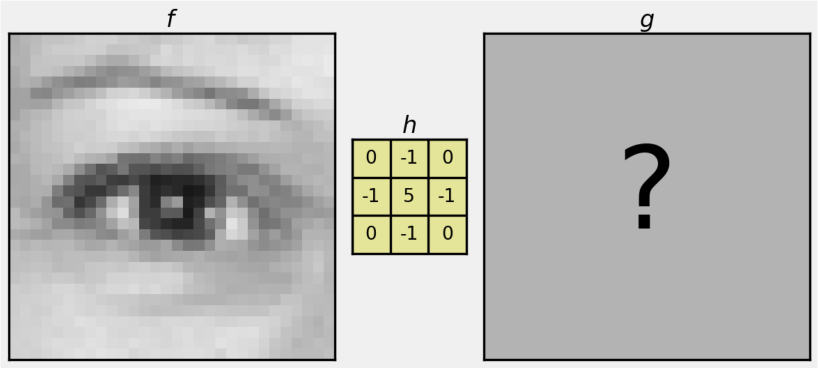


Figure 28: kernel 4

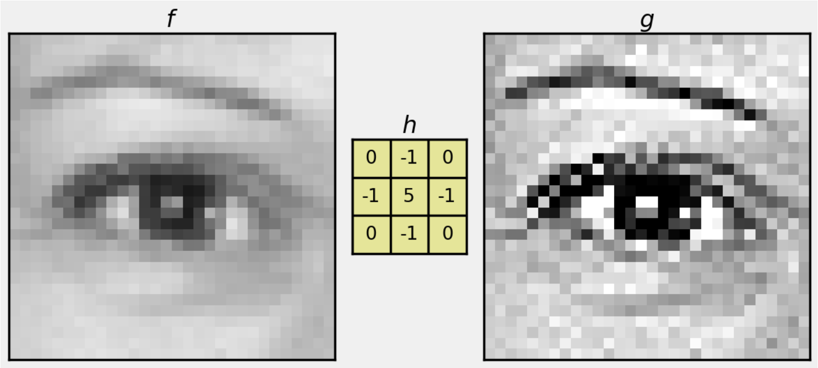


Figure 29: kernel 4

Image Sharpening

We can control the *amount* of sharpening:

$$h_{sharp} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} * amount$$

More Kernel Examples

There is a nice interactive tool to view kernel operations here:
https://setosa.io/ev/image-kernels/

The ImageMagick documentation has a nice list of kernels:
https://legacy.imagemagick.org/Usage/convolve/

Unsharp Masking

A *high pass* filter formed from a *low pass* filtered image.

- Usually preferred over kernel sharpening filter.
- A legacy of the pre-digital period.

Unsharp Masking

Low pass filter removes high-frequency detail.

- Difference between original and filtered images is what the filter removed.
    - high frequency information.
- Add difference to original image to enhance edges, etc.

Unsharp Masking

Figure 30: Unsharp masking 7x7 Gaussian kernel

Unsharp Masking

The **sharpened** image is the original image plus the *unsharp mask* multiplied by some factor.

- The difference image is the unsharp mask!

Unsharp Masking

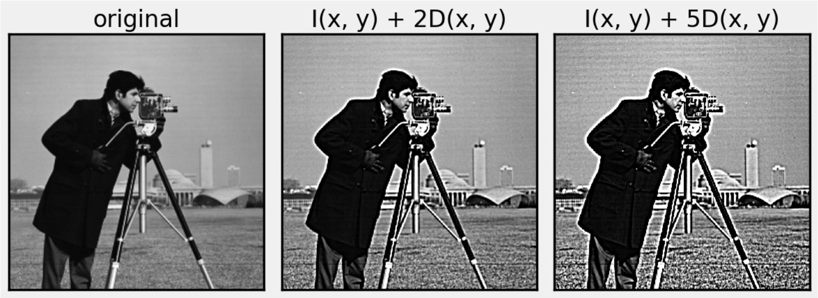


Figure 31: Unsharp masking 2D and 5D

Unsharp Masking

Figure 32: Unsharp masking 11x11 Gaussian kernel

Unsharp Masking

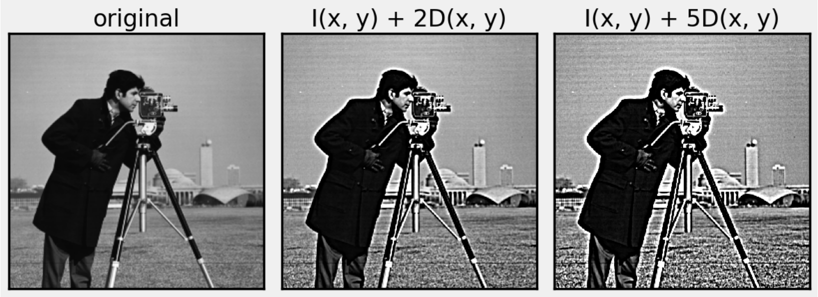


Figure 33: Unsharp masking 2D and 5D

Unsharp Masking

Generally don't want to boost all fine detail as noise would also be enhanced.

- Adjust the Gaussian parameters.
- Threshold the difference image.
- Care is required to avoid artefacts (e.g. halos).

Image Filters as Templates

2D Convolution can be thought of as comparing a little picture (the filter kernel) against all local regions in the image.

Image Filters as Templates

If the filter kernel contains a picture of something you want to locate inside the image (a template), the filter response should be maximised at the local region that most closely matches it.

- We can use image filtering for object location
- Known as Template Matching.

Image Filters as Templates

Algorithm:

- subtract the mean from the image and template
- convolve the template with the image
- find the location of the maximum response

Image Filters as Templates

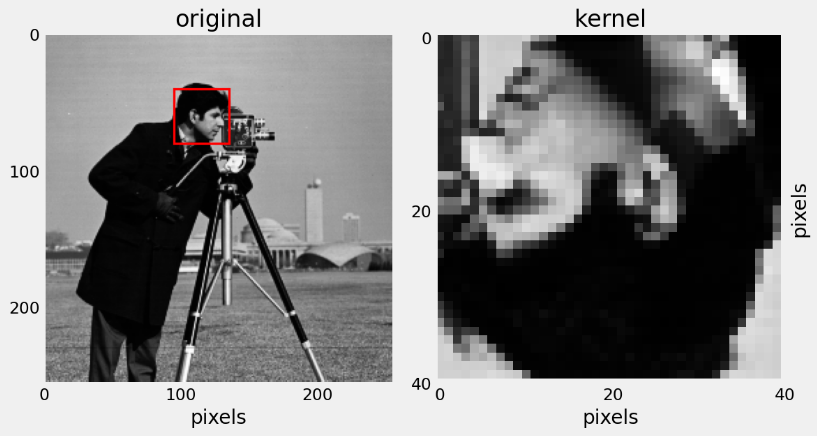


Figure 34: Select a region to form a template.

Image Filters as Templates

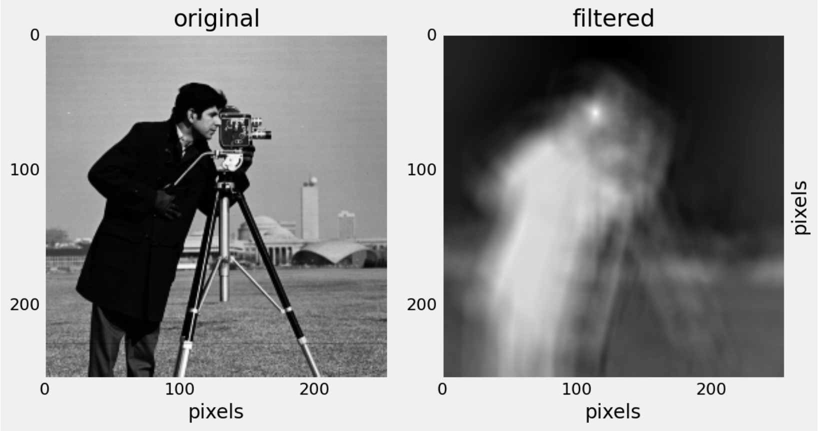


Figure 35: Perform the convolution operation.

Image Filters as Templates

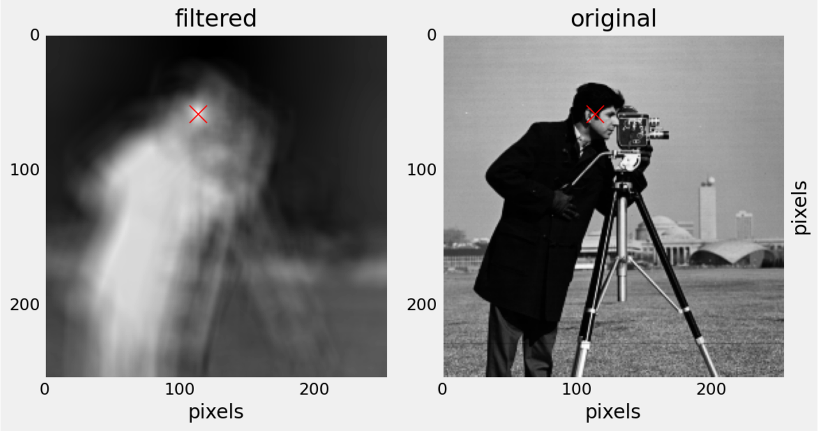


Figure 36: Locate the maximum filter response.

Summary

- 2D Convolutions
- Smoothing Filters
- Sharpening and Unsharp masking
- Template matching