

Kernel (image processing)

From Wikipedia, the free encyclopedia

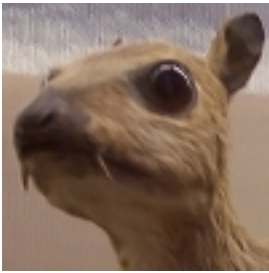


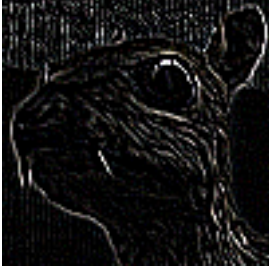
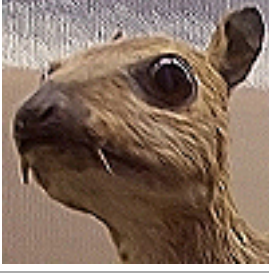
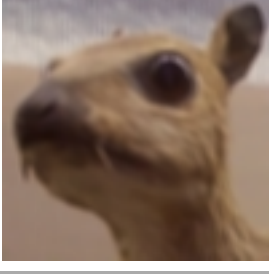
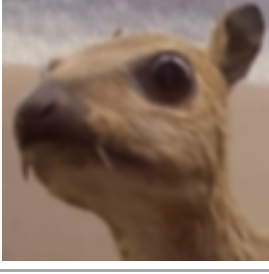

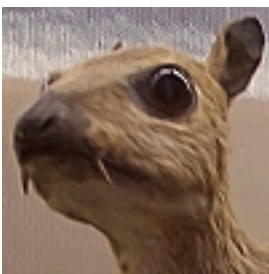
In image processing, a **kernel**, **convolution matrix**, or **mask** is a small matrix. It is used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between a kernel and an image.

Contents

- 1 Details
 - 1.1 Origin
- 2 Convolution
 - 2.1 Edge Handling
 - 2.2 Normalization
- 3 References
- 4 External links

Details

Depending on the element values, a kernel can cause a wide range of effects.

Operation	Kernel	Image result
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3×3 (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5×5 (approximation)	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5×5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

The above are just a few examples of effects achievable by convolving kernels and images.

Origin

The origin is the position of the kernel which is above (conceptually) the current output pixel. This could be outside of the actual kernel, though usually it corresponds to one of the kernel elements. For a symmetric kernel, the origin is usually the center element.

Convolution

Convolution is the process of adding each element of the image to its local neighbors, weighted by the kernel. This is related to a form of mathematical convolution. It should be noted that the matrix operation being performed - convolution - is not traditional matrix multiplication, despite being similarly denoted by $*$.

For example, if we have two three-by-three matrices, the first a kernel, and the second an image piece, convolution is the process of flipping both the rows and columns of the kernel and then multiplying locally similar entries and summing. The element at coordinates $[2, 2]$ (that is, the central element) of the resulting image would be a weighted combination of all the entries of the image matrix, with weights given by the kernel:

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \right) [2, 2] = (i * 1) + (h * 2) + (g * 3) + (f * 4) + (e * 5) + (d * 6) + (c * 7) + (b * 8) + (a * 9)$$

The other entries would be similarly weighted, where we position the center of the kernel on each of the boundary points of the image, and compute a weighted sum.

The values of a given pixel in the output image are calculated by multiplying each kernel value by the corresponding input image pixel values. This can be described algorithmically with the following pseudo-code:

```
for each image row in input image:
  for each pixel in image row:

    set accumulator to zero

    for each kernel row in kernel:
      for each element in kernel row:

        if element position corresponding* to pixel position then
          multiply element value corresponding* to pixel value
          add result to accumulator
        endif

    set output image pixel to accumulator
```

*corresponding input image pixels are found relative to the kernel's origin.

If kernel is symmetric then place the centre(origin) of kernel on the current pixel. Then kernel will be overlapped with neighboring pixels too. Now multiply each kernel element with the pixel value it overlapped with and add all the obtained values. Resultant value will be the value for the current pixel that is overlapped with the center of the kernel.

If the kernel is not symmetric, it has to be flipped both around its horizontal and vertical axis before calculating the convolution as above.^[1]

Edge Handling

Kernel convolution usually requires values from pixels outside of the image boundaries. There are a variety of methods for handling image edges.

Extend

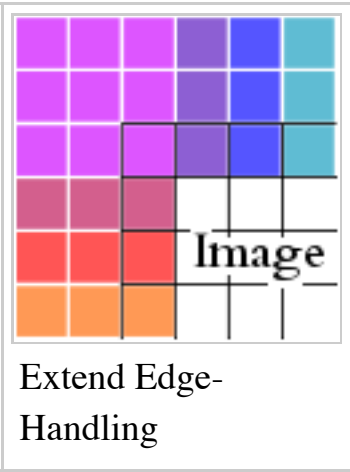
The nearest border pixels are conceptually extended as far as necessary to provide values for the convolution. Corner pixels are extended in 90° wedges. Other edge pixels are extended in lines.

Wrap

The image is conceptually wrapped (or tiled) and values are taken from the opposite edge or corner.

Crop

Any pixel in the output image which would require values from beyond the edge is skipped. This method can result in the output image being slightly smaller, with the edges having been cropped.



Normalization

Normalization is defined as the division of each element in the kernel by the sum.

References

1. http://www.songho.ca/dsp/convolution/convolution2d_example.html
 - Ludwig, Jamie (n.d.). *Image Convolution* (http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Ludwig_ImageCo)

- nvolution.pdf) (pdf). *Portland State University*.
- Lecarme, Olivier; Delvare, Karine (January 2013). *The Book of GIMP: A Complete Guide to Nearly Everything*. No Starch Press. p. 429. ISBN 978-1593273835.
- Gumster, Jason van; Shimonski, Robert (March 2012). *GIMP Bible*. Wiley. pp. 438–442. ISBN 978-0470523971.
- Stockman, George C.; Shapiro, Linda G. (February 2001). *Computer Vision*. Prentice Hall. pp. 53–54. ISBN 978-0130307965.

External links

- Implementing 2d convolution on FPGA (<https://www.youtube.com/watch?v=38lj0VQci7E>)
- vImage Programming Guide: Performing Convolution Operations (<http://developer.apple.com/library/IOs/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html>)
- Image Processing using 2D-Convolution (<http://williamson-labs.com/convolution-2d.htm>)
- GNU Image Manipulation Program - User Manual - 8.2. Convolution Matrix (<http://docs.gimp.org/en/plugin-in-convmatrix.html>)
- Interactive Demonstration of 3x3 Convolution Kernels (<http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo#demo>)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Kernel_(image_processing)&oldid=778412481"

Categories: Image processing | Feature detection (computer vision)

-
- This page was last edited on 2 May 2017, at 23:51.
 - Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.