Introduction to Computer Vision: Filters

Navasardyan Shant

September 19, 2019

Overview

Overview

What is image?

What is image?

Definition

Let H, W and C be arbitrary positive integers. Then we call a $3-$ *dimentional* tensor $A = (a_{ijk})^{H \times W \times C}$ **image**, if $a_{ijk} \geq 0$ for any i, j, k. H, W and C are called image *height*, *width* and *number of channels* respectively.

What is image?

Definition

Let H, W and C be arbitrary positive integers. Then we call a $3-$*dimentional* tensor $A = (a_{ijk})^{H \times W \times C}$ **image**, if $a_{ijk} \geq 0$ for any i, j, k. H, W and C are called image *height*, *width* and *number of channels* respectively.

Definition

In majority of cases we will have $C = 1$ or $C = 3$. If for an image A the number of it's channels is $C = 1$, we call this image **grayscaled**.

What is image?

Definition

Let H, W and C be arbitrary positive integers. Then we call a $3-$*dimentional* tensor $A = (a_{ijk})^{H \times W \times C}$ **image**, if $a_{ijk} \geq 0$ for any i, j, k. H, W and C are called image *height*, *width* and *number of channels* respectively.

Definition

In majority of cases we will have $C = 1$ or $C = 3$. If for an image A the number of it's channels is $C = 1$, we call this image **grayscaled**.

Note

In majority of cases we will consider images with $a_{ijk} \in [0, 1]$ and images with $a_{ijk} \in \{0, 1, \ldots, 255\}$.

Example

Figure: 3-channeled image, displayed in RGB color space

Figure: grayscaled version of the first image

Example

Figure: 3-channeled image, displayed in RGB color space

Figure: grayscaled version of the first image

Note

If a 3-channeled image $A = (a_{ijk})^{H \times W \times C}$ will be displayed in RGB color space, we refer to the tensors $A_R = (a_{ij1})^{H \times W}$, $A_G = (a_{ij2})^{H \times W}$ and $A_B = (a_{ij3})^{H \times W}$ as $R(Red)-, G(Green)-, B(Blue) - channels$ of A respectively.

Image as Function

In some cases it is theoretically convenient to refer image as a function.

Image as Function

In some cases it is theoretically convenient to refer image as a function.

Definition

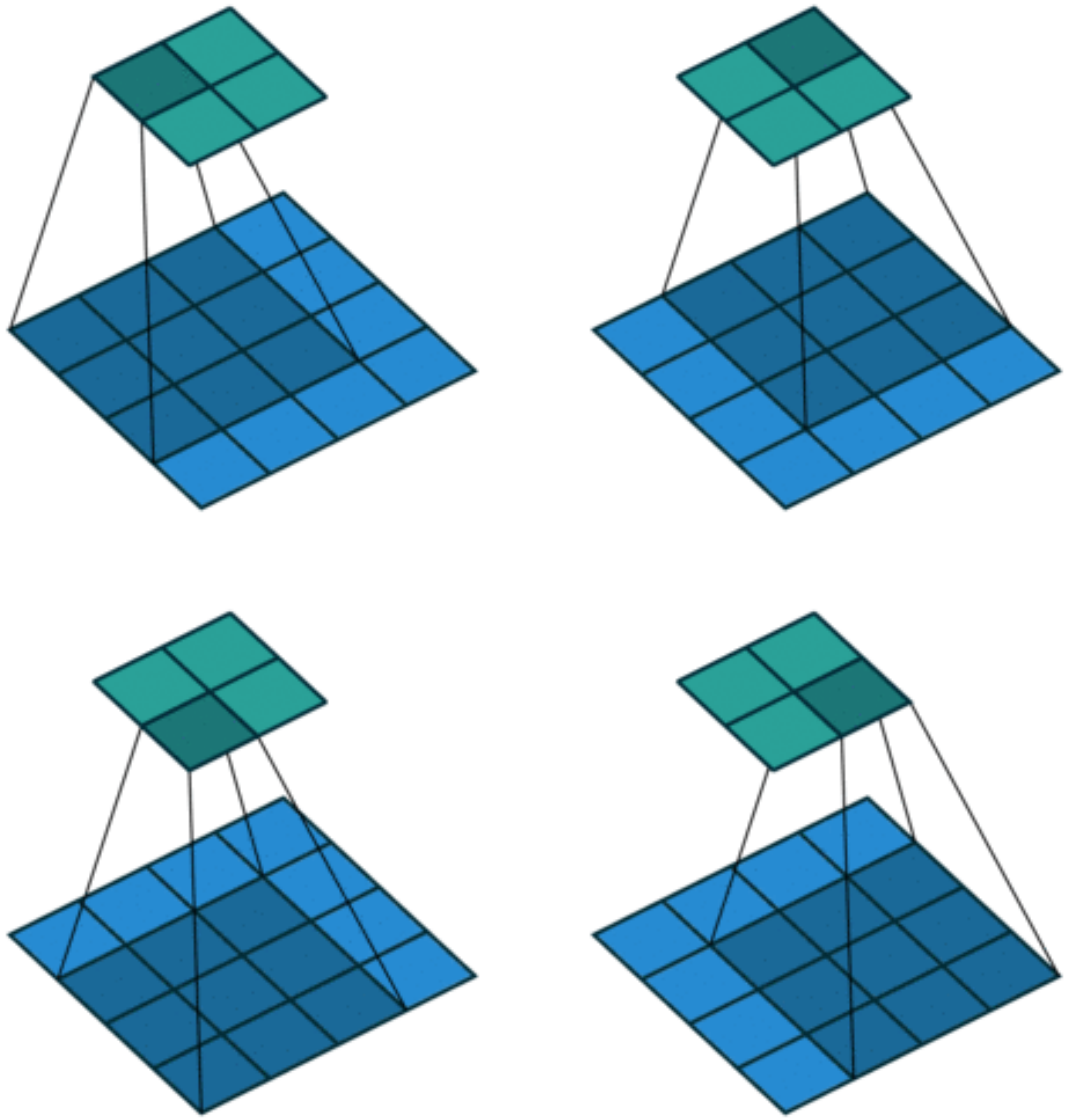
Let C be a positive integer. We call a function

$$f : [0,1]^2 \to [0,1]^C$$

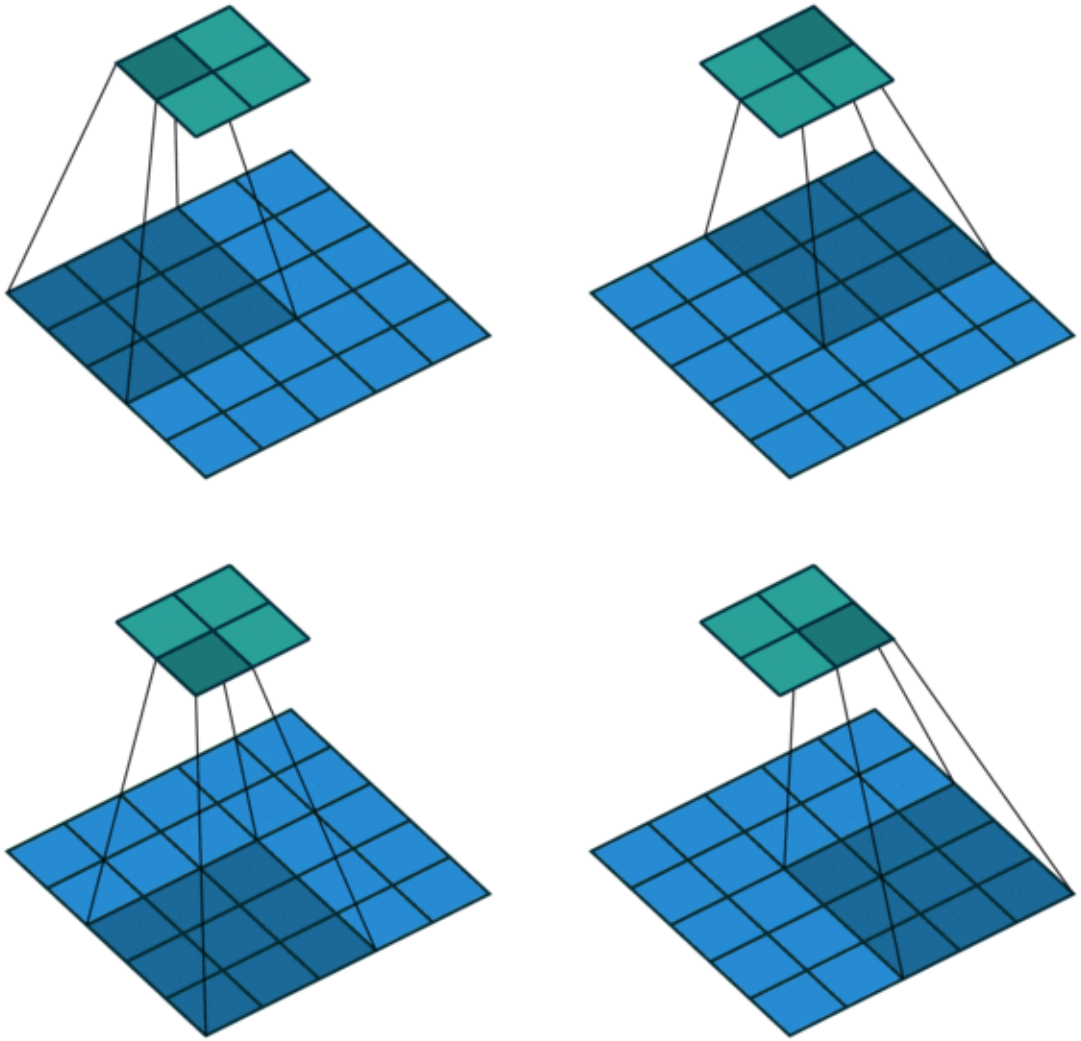
image, and C the number of its channels. If not mentioned otherwise, we will assume the function f is continuous.

Overview

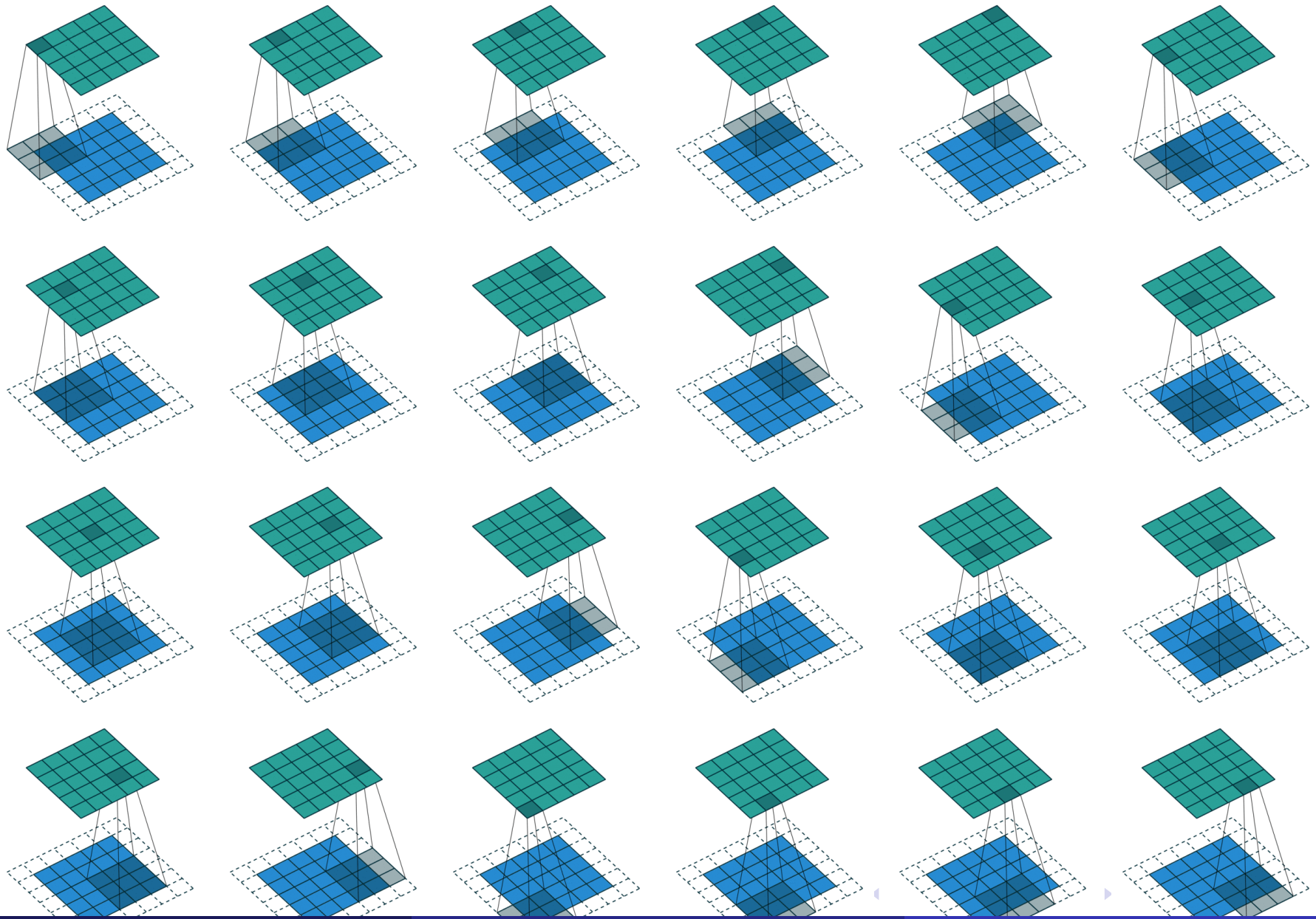
Convolution with no padding, no strides



Convolution with no padding, strides=2



Convolution with same padding, no strides



Convolution: Example

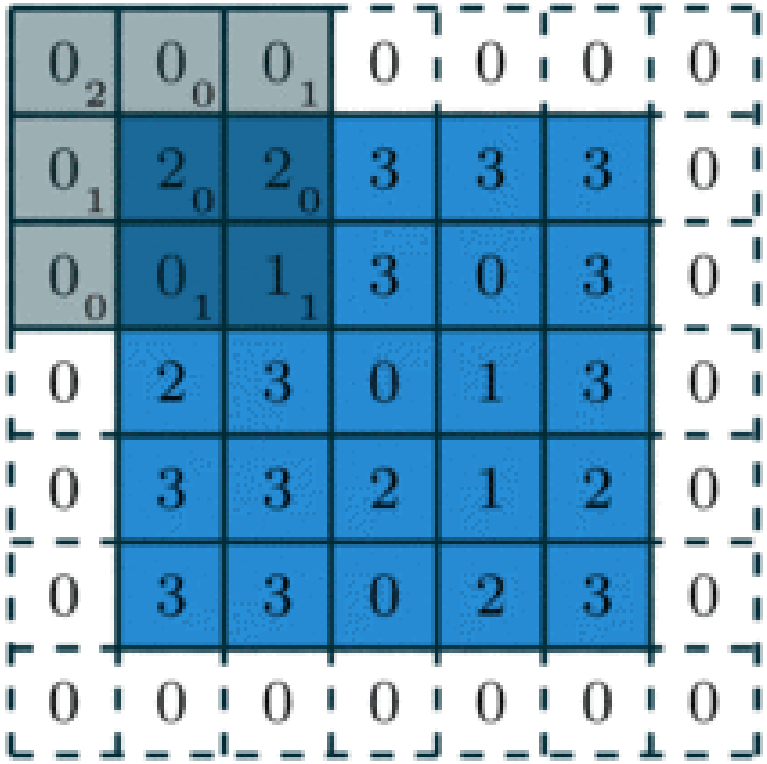
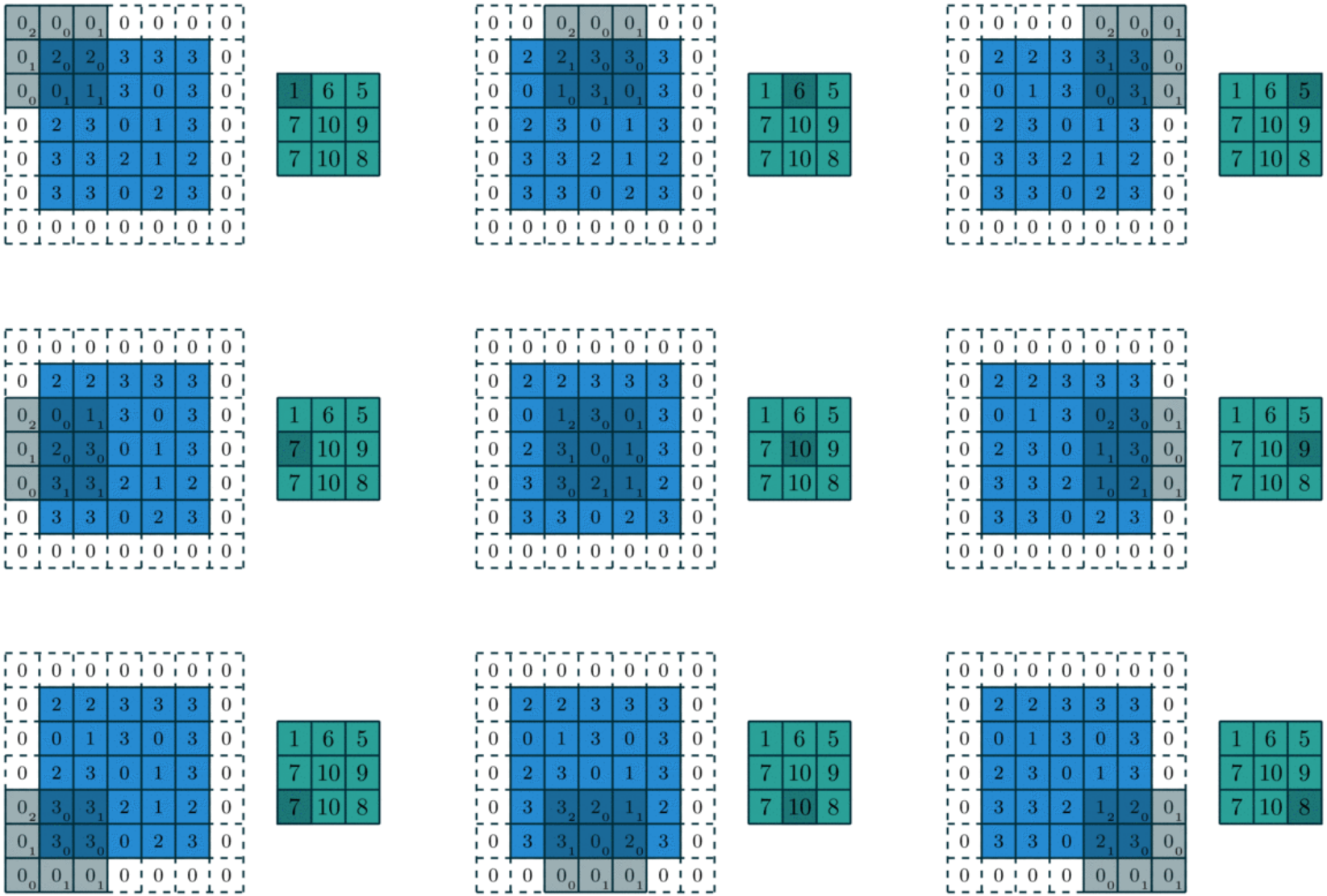


Figure: Find the result after convolution with padding $= 1$, strides $= 2$. The initial image is in the blue box, the kernel is written in gray box with its values in right-bottom corners.

Convolution: Example



Convolution: Formal Definition

Let $A = (a_{ijk})^{H \times W \times C}$ be an image, $\Omega = (\omega_{pqr})^{h \times w \times C}$ be a tensor and $h \leq H, w \leq W$. Let s_h, s_w be positive integers.

Convolution with filter-size $= 1$

Let us consider the function

$$f_\Omega : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H_o \times W_o}, \quad B = (b_{i',j'})^{H_o \times W_o} := f_\Omega(A).$$

$$b_{i'j'} = \sum_{p,q,r} \omega_{pqr} a_{i' \cdot s_h + p, j' \cdot s_w + q, r},$$

where $H_o = \left\lceil \frac{H-h+1}{s_h} \right\rceil, W_o = \left\lceil \frac{W-w+1}{s_w} \right\rceil$

Convolution: Formal Definition

Let $\Omega_0, \Omega_1, \ldots, \Omega_{D-1}$ be tensors of shape $h \times w \times C$. Consider the tensor $\Omega = (\omega_{pqrd})^{h \times w \times C \times D}$, $\omega_{pqrd} = (\Omega_d)_{pqr}$ and the function

$$f_\Omega : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H_o \times W_o \times D} \quad f_\Omega(A) = (f_{\Omega_0}(A), \ldots, f_{\Omega_{D-1}}(A))$$

Convolution: Formal Definition

Let $\Omega_0, \Omega_1, \ldots, \Omega_{D-1}$ be tensors of shape $h \times w \times C$. Consider the tensor $\Omega = (\omega_{pqrd})^{h \times w \times C \times D}$, $\omega_{pqrd} = (\Omega_d)_{pqr}$ and the function

$$f_\Omega : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H_o \times W_o \times D} \quad f_\Omega(A) = (f_{\Omega_0}(A), \ldots, f_{\Omega_{D-1}}(A))$$

Definition (the milestone concept in CV)

The function f_Ω is called a convolution with kernel Ω. s_w, s_h are called its horizontal and vertical strides respectively. The slices $\Omega_0, \ldots \Omega_{D-1}$ of Ω are called filters of convolution f_Ω and D is called the filter-size of f_Ω. We call the slices $(f_\Omega(A)_{i'j'd'})^{H_o \times W_o}$ feature maps of the convolution f_Ω.

Convolution: Formal Definition

Let $\Omega_0, \Omega_1, \ldots, \Omega_{D-1}$ be tensors of shape $h \times w \times C$. Consider the tensor $\Omega = (\omega_{pqrd})^{h \times w \times C \times D}$, $\omega_{pqrd} = (\Omega_d)_{pqr}$ and the function

$$f_\Omega : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^{H_o \times W_o \times D} \quad f_\Omega(A) = (f_{\Omega_0}(A), \ldots, f_{\Omega_{D-1}}(A))$$

Definition (the milestone concept in CV)

The function f_Ω is called a convolution with kernel Ω. s_w, s_h are called its horizontal and vertical strides respectively. The slices $\Omega_0, \ldots \Omega_{D-1}$ of Ω are called filters of convolution f_Ω and D is called the filter-size of f_Ω. We call the slices $(f_\Omega(A)_{i'j'd'})^{H_o \times W_o}$ feature maps of the convolution f_Ω.

Exercise

Formalize the concept of padding !!!

Overview

Bluring

Images taken by camera are sometimes defocused. In some cases we need to defocus image or maybe a part of it.

Bluring

Images taken by camera are sometimes defocused. In some cases we need to defocus image or maybe a part of it.

Figure: Image is taken in "portrait" mode, the background is defocused.

Figure: Sticker is added with edge blur

Figure: Patch inside of red frame.

Figure: Sticker is added with edge blur

Figure: Patch inside of red frame.

Figure: Sticker is added without edge blur

Figure: Patch inside of red frame.

Average Blur

Exercise

Let A be an one-channeled image with a white (1-valued pixels) disk on the black (0-valued pixels) background. Obtain its "blured" version.

Average Blur

Exercise

Let A be an one-channeled image with a white (1-valued pixels) disk on the black (0-valued pixels) background. Obtain its "blured" version.

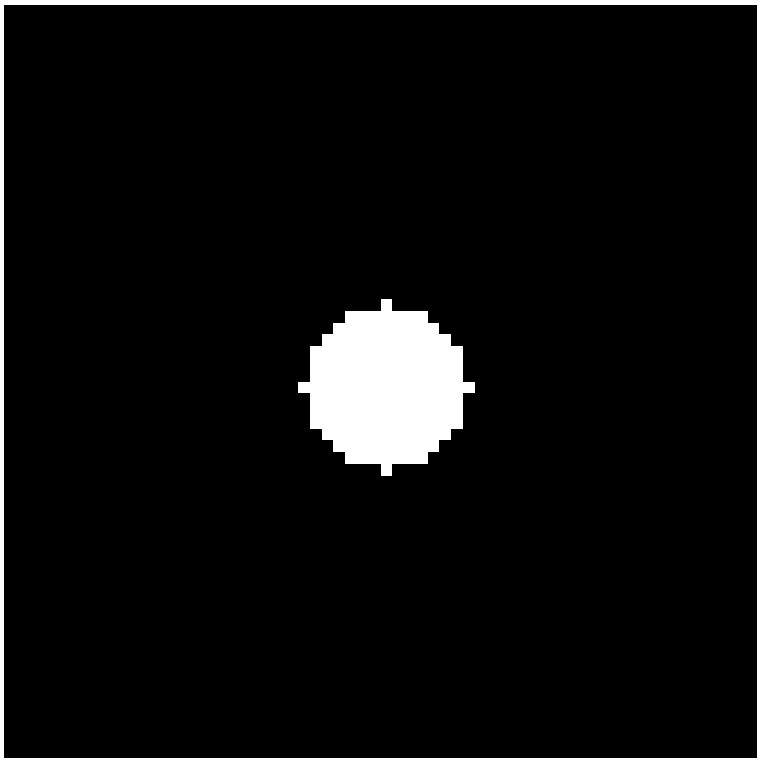


Figure: image of size 64×64

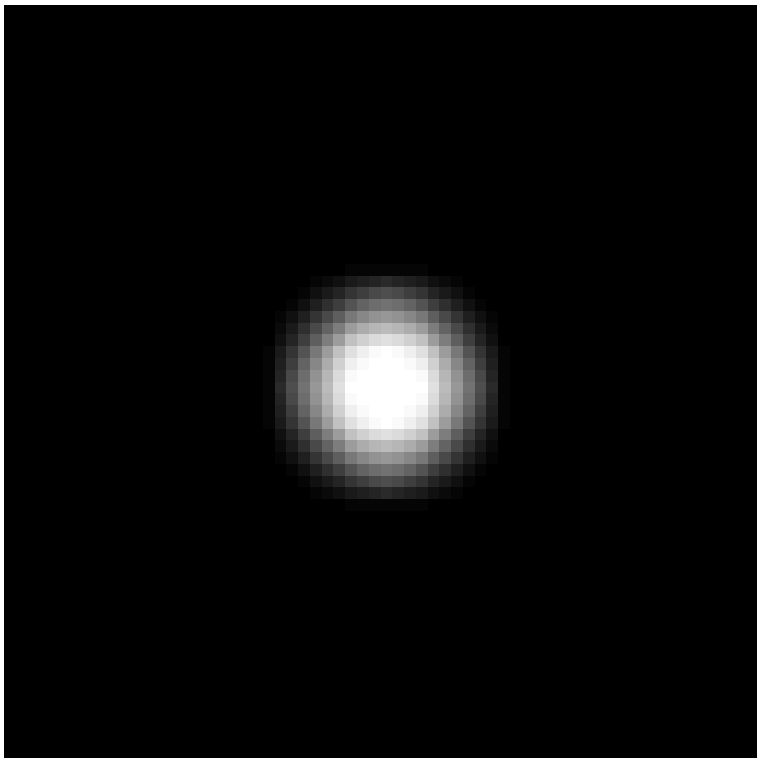


Figure: Average blur with k=7

Average Blur

For bluring an image, we can use convolution operation. Let $A \in \mathbb{R}^{H \times W \times 1}$ be an image, k is a positive integer. Then the convolution of A with a kernel $\frac{1}{k^2}\mathbf{1}_{H \times W \times 1}$ will result to a blured version of the image A (example below). In case of 3-channeled image, one can use the kernel $\frac{1}{k^2 * 3}\mathbf{1}_{H \times W \times 3}$ or apply 1-filter-sized convolution on each channel then combine them. Here $\mathbf{1}_{H \times W \times C}$ is a tensor of shape $H \times W \times C$ with all ones.

Average Blur: Defocusing Example

Note

In spite of Average Blur is something similar to the defocusing effect of a camera, the following example illustrates that it can't be used in order to get defocus effect.

Average Blur: Defocusing Example

Note

In spite of Average Blur is something similar to the defocusing effect of a camera, the following example illustrates that it can't be used in order to get defocus effect.

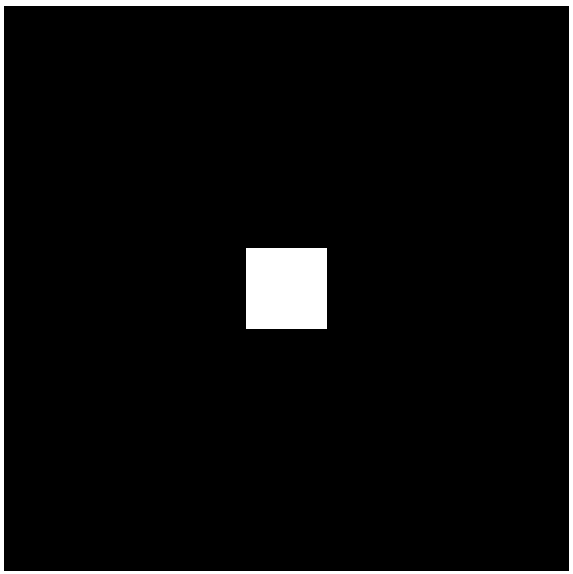


Figure: image of size 7×7 with a unique 1-value in the center

Figure: Average blur of the image, with kernel-size $= 3$

Gaussian Blur

Note

We want blur to gain a defocus effect of the camera, so the center must be more bright than the any other pixel.

Gaussian Blur

Note

We want blur to gain a defocus effect of the camera, so the center must be more bright than the any other pixel.

This can be obtained by convolving with **Gaussian Kernel**:

Definition (Gaussian Kernel)

The following function is called (2-d) **Gaussian Kernel**:

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

For a positive integer k we can take a filter $\Omega = (\omega_{ij})^{k\times k}$ with $\omega_{ij} = \frac{1}{2\pi\sigma^2} e^{-\frac{(m-i)^2+(m-j)^2}{2\sigma^2}}$, where $m = [k/2]$. The convolution with the kernel Ω we call **Gaussian Blur**. Sometimes we call the kernel Ω also **Gaussian Kernel**.

Gaussian Kernel

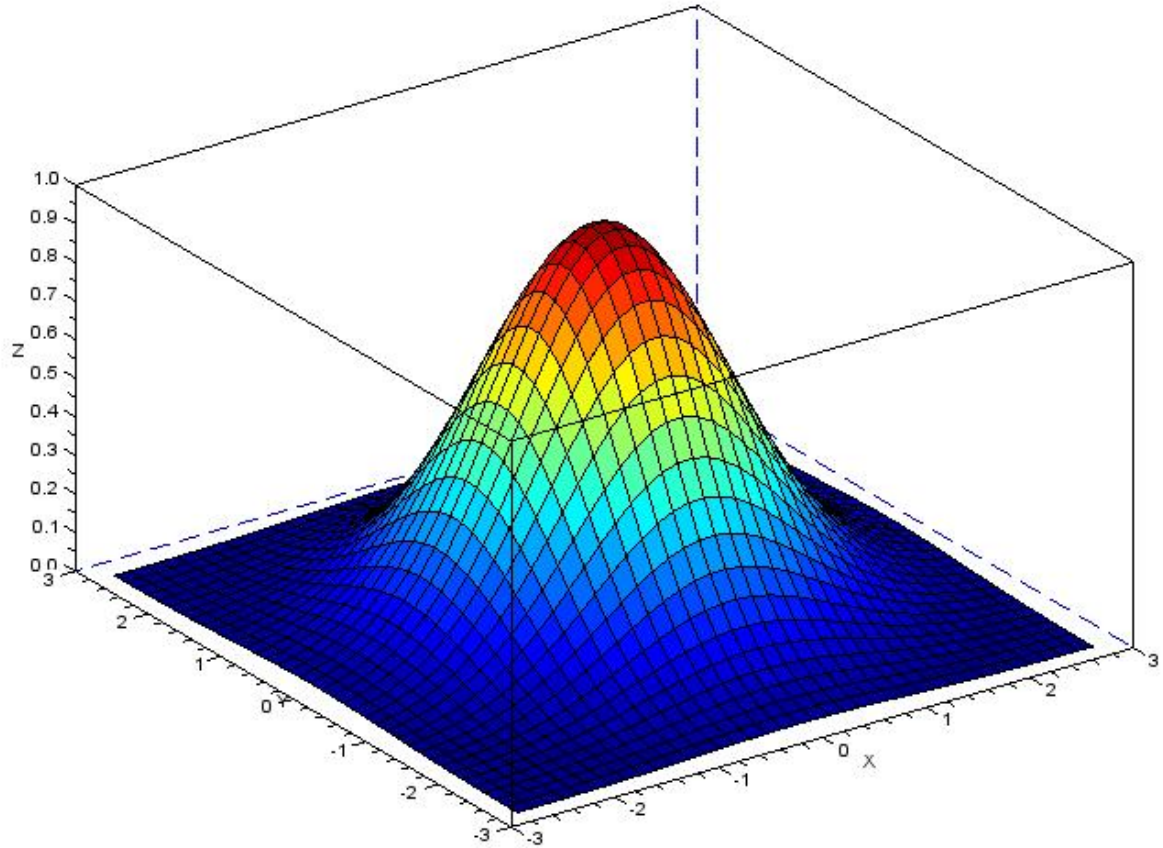


Figure: Gaussian Kernel

Center-Point Example Revisited

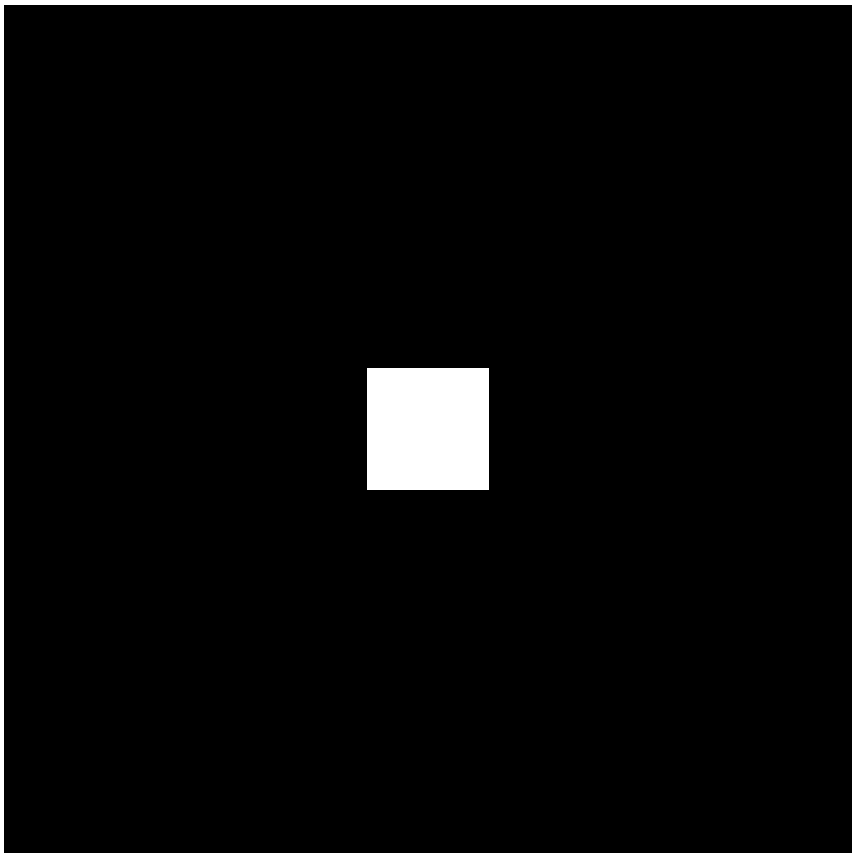


Figure: image of size 7×7 with a unique 1-value in the center

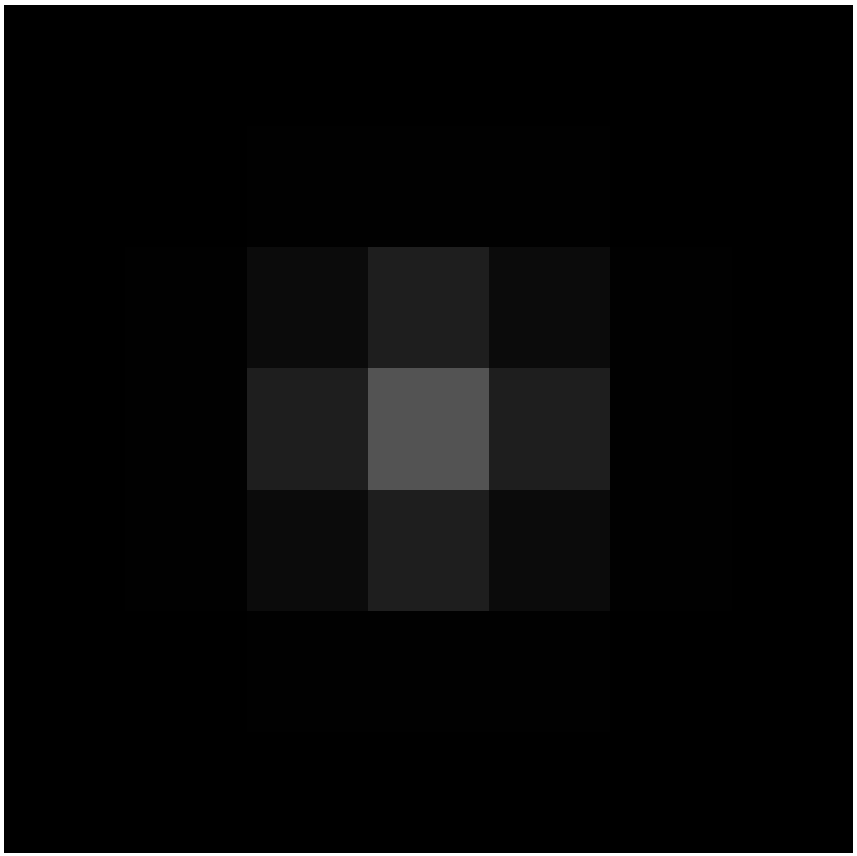


Figure: Gaussian blur of the image, with kernel-size $= 5$, sigma=0.7

Overview

Edge Detection Toy Example

Exercise

Let $A \in \mathbb{R}^{H \times W}$ be an image and S be an arbitrary set of its indices. Let $A(i,j) = 1$ iff $(i,j) \in S$.

- How can we define the **border** of S? This border we will denote by ∂S.
- Find the ∂S.

Edge Detection Toy Example

Exercise

Let $A \in \mathbb{R}^{H \times W}$ be an image and S be an arbitrary set of its indices. Let $A(i,j) = 1$ iff $(i,j) \in S$.

- How can we define the **border** of S? This border we will denote by ∂S.
- Find the ∂S.

Answer

- Here we say that a pixel $(i,j) \in S$ is a **border pixel** of S if its value differs from one of its 4-neighbour values.
- The code is below:

Code For Example

```python
import numpy as np
def get_border(img):
    B = []
    H, W = img.shape[0], img.shape[1]
    img_pad = np.zeros(shape = (H+2, W+2))
    img_pad[1:H+1,1:W+1] = img.copy()
    for i in range(H):
        for j in range(W):
            if img_pad[i+1,j+1] == 1:
                dist_1 = np.abs(img_pad[i+1,j+1] - img_pad[i,j+1])
                dist_2 = np.abs(img_pad[i+1,j+1] - img_pad[i+1,j+2])
                dist_3 = np.abs(img_pad[i+1,j+1] - img_pad[i+2,j+1])
                dist_4 = np.abs(img_pad[i+1,j+1] - img_pad[i+1,j])
                if dist_1+dist_2+dist_3+dist_4 >0:
                    B.append((i,j))
    return B
```

Image Gradient

Note

Instead of computing with explicit for-loop as it was done in the code above (essentially in lines 10-14), we can use point-wise L1-distances between the image A and its 4 shifted versions (shifting by one pixel top, bottom, left and right).

Image Gradient

Note

Instead of computing with explicit for-loop as it was done in the code above (essentially in lines 10-14), we can use point-wise L1-distances between the image A and its 4 shifted versions (shifting by one pixel top, bottom, left and right).

Note

This distances can be obtained by taking absolute values of convolutions of the image A by the kernels

$$(-1, 1), \quad (1, -1), \quad \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Image Gradient

Note

Instead of computing with explicit for-loop as it was done in the code above (essentially in lines 10-14), we can use point-wise L1-distances between the image A and its 4 shifted versions (shifting by one pixel top, bottom, left and right).

Note

This distances can be obtained by taking absolute values of convolutions of the image A by the kernels

$$(-1, 1), \quad (1, -1), \quad \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

Exercise

Solve the same problem by using convolutions with the kernels above.

Image Gradient

Let $f : [0, 1]^2 \to [0, 1]$ be an image, and $A \in \mathbb{R}^{H \times W}$ is its sampled discrete version.

Image Gradient

Let $f : [0,1]^2 \to [0,1]$ be an image, and $A \in \mathbb{R}^{H \times W}$ is its sampled discrete version.

We have that the partial derivative w.r.t. the variable x of $f(x,y)$ is

$$\frac{\partial f}{\partial x}(x,y) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}.$$

Image Gradient

Let $f : [0,1]^2 \to [0,1]$ be an image, and $A \in \mathbb{R}^{H \times W}$ is its sampled discrete version.

We have that the partial derivative w.r.t. the variable x of $f(x,y)$ is

$$\frac{\partial f}{\partial x}(x,y) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}.$$

As in the sampled array the smallest index difference is 1, we can approximate this partial derivative by taking $|\varepsilon| = 1$. So we get

Image Gradient

Let $f : [0,1]^2 \to [0,1]$ be an image, and $A \in \mathbb{R}^{H \times W}$ is its sampled discrete version.

We have that the partial derivative w.r.t. the variable x of $f(x,y)$ is

$$\frac{\partial f}{\partial x}(x,y) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}.$$

As in the sampled array the smallest index difference is 1, we can approximate this partial derivative by taking $|\varepsilon| = 1$. So we get

$$\frac{\partial f}{\partial x}(x,y) \approx A_{i+1,j} - A_{i,j} \quad (\varepsilon = 1),$$

$$\frac{\partial f}{\partial x}(x,y) \approx A_{i,j} - A_{i-1,j} \quad (\varepsilon = -1),$$

where $i = [x \cdot H], \quad j = [y \cdot W]$ for $0 \le x, y < 1$ (and $i = H-1, j = W-1$ for $x = 1, y = 1$).

Image Gradient

This approximations are called finite differences.

Image Gradient

This approximations are called finite differences. We might also approximate the partial derivative with *symmetric* finite differences:

$$\frac{\partial f}{\partial x}(x, y) \approx A_{i+1,j} - A_{i-1,j}.$$

Image Gradient

This approximations are called finite differences. We might also approximate the partial derivative with *symmetric* finite differences:

$$\frac{\partial f}{\partial x}(x, y) \approx A_{i+1,j} - A_{i-1,j}.$$

This can be done with a convolution with the following kernel (and appropriate padding=1):

$$\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Image Gradient

This approximations are called finite differences. We might also approximate the partial derivative with *symmetric* finite differences:

$$\frac{\partial f}{\partial x}(x, y) \approx A_{i+1,j} - A_{i-1,j}.$$

This can be done with a convolution with the following kernel (and appropriate padding=1):

$$\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

By the same way we can use the convolution with the kernel $\begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$ for approximating $\frac{\partial f}{\partial y}(x, y)$.

Image Gradient

Definition (Image Gradient)

Any approximation of the gradient $\nabla f(x, y)$ in terms of elements of A we call the **gradient** of the image A. We will mostly use the approximation with symmetric finite differences, so if not mentioned otherwise, we call the tensor $G \in \mathbb{R}^{H \times W \times 2}$ the gradient of the image A, if $G_x = (G_{ij0})^{H \times W}$, $\quad G_y = (G_{ij1})^{H \times W}$ are the results of convolutions on A with kernels

$$\begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

respectively.

Image Gradient: Magnitude and Orientation

In some applications of image gradients (e.g. edge detection, neighbourhood description), the concept of **magnitude** and **orientation** of gradient is used.

Image Gradient: Magnitude and Orientation

In some applications of image gradients (e.g. edge detection, neighbourhood description), the concept of **magnitude** and **orientation** of gradient is used.

Definition (Magnitude, Orientation)

Let A be an image and G be its gradient. Let $G_x = (G_{ij0}), G_y = (G_{ij1})$ are partial derivatives of A. Then the tensor of lengths of image gradient vectors is called the **magnitude** of image gradient and the tensor of orientation angles of image gradient is called the **orientations** of image gradient:

$$Mag = \sqrt{G_x^2 + G_y^2}, \qquad Ori = arcsin(G_y/Mag)).$$

Image Gradient: Example

Note

If the image A is noisy, we might get high-valued responses in pixels which are not on the edges of A. To avoid this "fake" edge responses, we sometimes apply (mostly gaussian) blur, then compute image gradients.

Figure: noisy image of zebra pattern

Figure: left: magnitude of the gradient of the image of zebra pattern, right: magnitude of the gradient of the image of zebra pattern after gaussian blur

Overview

An Optimization Problem

Exercise

Let $x_0 \in \mathbb{R}^n$, $x_0 \neq \mathbf{0}$, solve the optimization problem

$$< x, x_0 > \rightarrow \max_{\|x\|=1} .$$

An Optimization Problem

Exercise

Let $x_0 \in \mathbb{R}^n$, $x_0 \neq \mathbf{0}$, solve the optimization problem

$$< x, x_0 > \rightarrow \max_{\|x\|=1} .$$

Solution

By Cauchy-Shwartz inequality, we have $| < x, x_0 > | \leq \|x\| \cdot \|x_0\|$ and the equality holds iff $x = \lambda x_0$ for some $\lambda \in \mathbb{R}$. So for any $\|x\| = 1$ we have $< x, x_0 > \leq | < x, x_0 > | \leq \|x\| \cdot \|x_0\| = \|x_0\|$, and the equality holds iff $< x, x_0 > \geq 0$ and $x = \lambda x_0$ for some $\lambda \in \mathbb{R}$. Since $\|x\| = 1$, we get $|\lambda| = \frac{1}{\|x_0\|}$, on the other hand, from $< x, x_0 > \geq 0$ we obtain $\lambda = \frac{1}{\|x_0\|}$. So the vector $x = \frac{x_0}{\|x_0\|}$ is a solution of the problem and it is the unique solution.

Convolution Responses: Toy Example Exercise

Exercise

Let $A \in \mathbb{R}^{H \times W}$ be an image which contains 1-valued squares of side size l. Let the elements of A which are not belong to this squares be 0. Find the centers of this squares.

Convolution Responses: Toy Example Exercise

Exercise

Let $A \in \mathbb{R}^{H \times W}$ be an image which contains 1-valued squares of side size l. Let the elements of A which are not belong to this squares be 0. Find the centers of this squares.

Solution

For simplicity let's assume l is an odd number. Then let B be the convolution of the $\frac{l+1}{2}$-padded A image with the kernel $K \in \{0,1\}^{l+2 \times l+2}$, where K is 1-valued square of length l followed by 1-padding with values 0. Let C be the convolution of the $\frac{l+1}{2}$-padded A^2 image (pointwise square) with the kernel $K \in \{0,1\}^{l+2 \times l+2}$ with all values 1. Then the all we need is to take points from $argmax_{(i,j)}(\frac{B}{\sqrt{C}})_{ij}$.

Convolution Responses: Real image Example

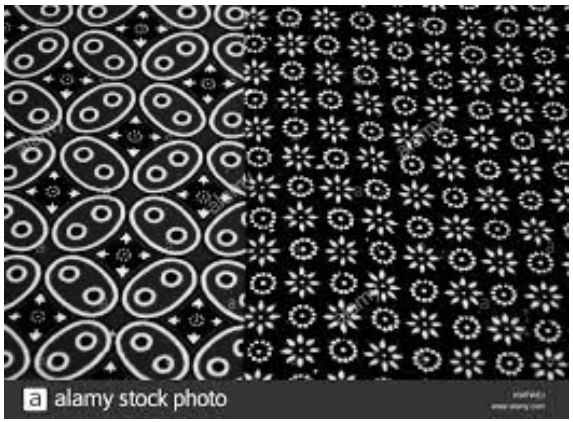


Figure: grayscaled image of a pattern, need to find circle centers

Convolution Responses: Real image Example

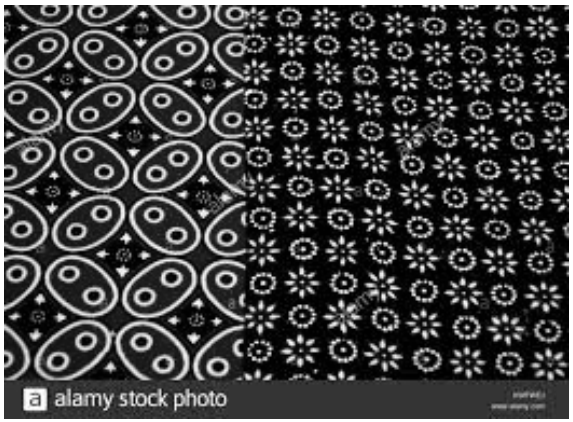


Figure: grayscaled image of a pattern, need to find circle centers

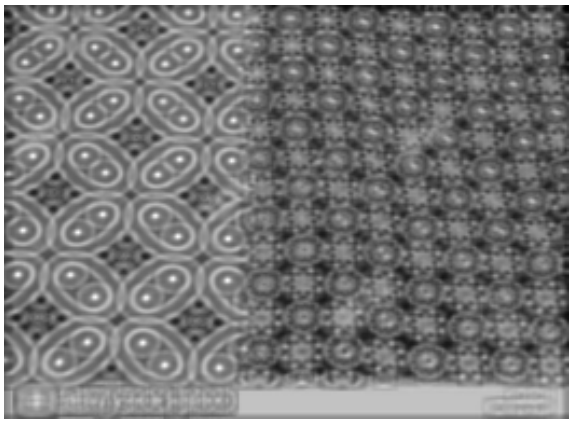


Figure: after convolution we get high responses in the centers of that circles

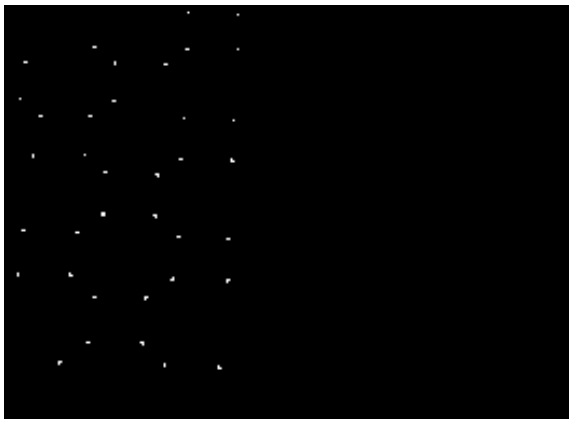


Figure: thresholded by the value 4.5 to show only centers of circles

Convolution Responses: Real image Example

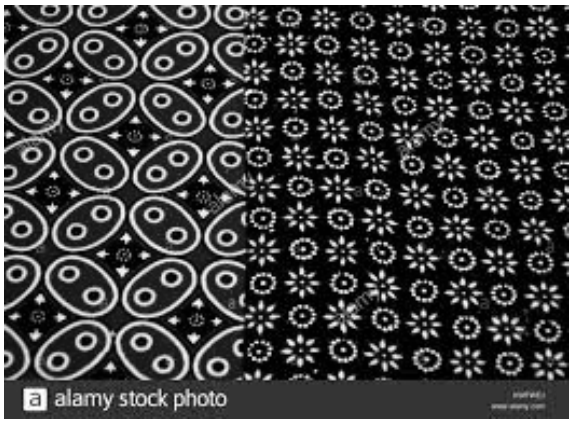


Figure: grayscaled image of a pattern, need to find circle centers

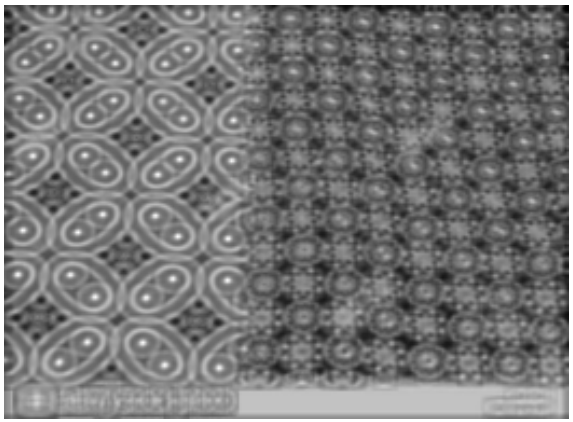


Figure: after convolution we get high responses in the centers of that circles

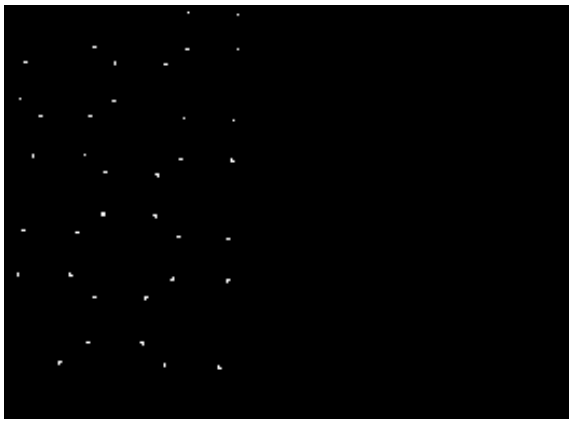


Figure: thresholded by the value 4.5 to show only centers of circles

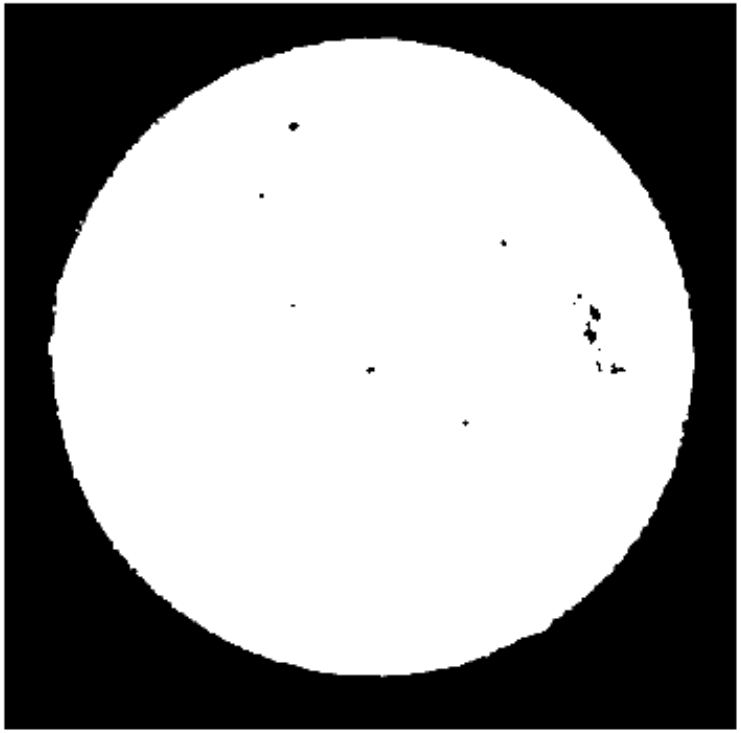
Question

How to activate not the centers of these circles (patches) but themselves ?

Overview

Non-Linear Filters: Exercise

Given the image of sun-disk mask which contains black spots



Exercise

Fill the black spots inside of the white disk by white pixels (here white pixels are 1-valued, and black pixels are 0-valued).

Erosion

Let we have an image which is "thicker" than we desire. How can we make it *thinner*? Example is below:

Erosion

Let we have an image which is "thicker" than we desire. How can we make it *thinner*? Example is below:

The corresponding operation is called **erosion**. Erosion can be done with "convloving" the image with a window of size $k \times k$, but instead of taking the weighted sum of image elements, take the *minimal* value of the image in the current window.

Erosion

Let we have an image which is "thicker" than we desire. How can we make it *thinner*? Example is below:

The corresponding operation is called **erosion**. Erosion can be done with "convloving" the image with a window of size $k \times k$, but instead of taking the weighted sum of image elements, take the *minimal* value of the image in the current window.

Exercise

Formalize the definition of the erosion operation.

Dilation

The opposite operation of the erosion is **dilation**, when one wants to make the object in the image "thicker". Example is below:

Dilation

The opposite operation of the erosion is **dilation**, when one wants to make the object in the image "thicker". Example is below:

Dilation can be done with "convloving" the image with a window of size $k \times k$, but instead of taking the weighted sum of image elements, take the *maximal* value of the image in the current window.

Dilation

The opposite operation of the erosion is **dilation**, when one wants to make the object in the image "thicker". Example is below:

Dilation can be done with "convloving" the image with a window of size $k \times k$, but instead of taking the weighted sum of image elements, take the *maximal* value of the image in the current window.

Exercise

Formalize the definition of the dilation operation.

Sun-Disk Mask Example Revisited

Now let us return to the problem of filling spots on the mask of the sun-disk image. We can take a window size 3×3 and dilate the image several times, until the spots disappear. Then the all we need to do is to erode the resulting image by the same window size and number of iterations:

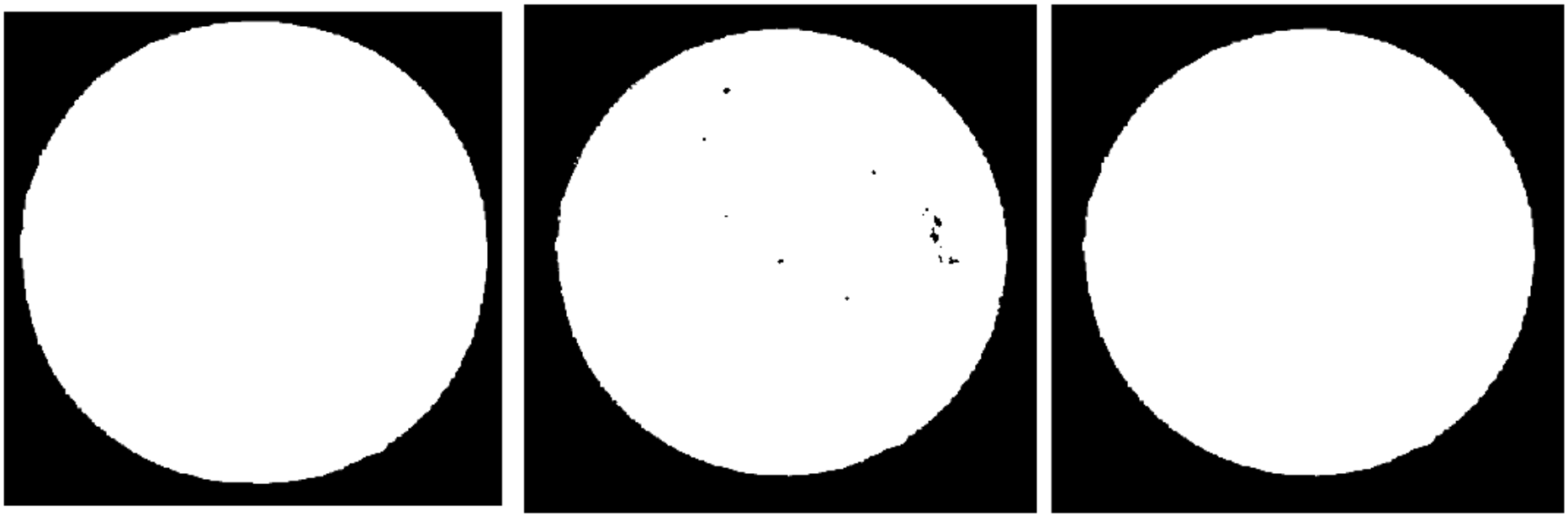


Figure: Left: dilation with window size 3×3 for 5 iterations, Middle: sun-disk mask, Right: erosion of dilated image with the same window size 3×3 and for 5 iterations.

Implementation of Erosion-Dilation Operations

Exercise

Implement the dilation (erosion) operation for window-size 1×2 by using convolutions and non-linear point-wise functions.

Implementation of Erosion-Dilation Operations

Exercise

Implement the dilation (erosion) operation for window-size 1×2 by using convolutions and non-linear point-wise functions.

Hint!

$$max\{a, b\} = |a - b| + (a + b)$$

Implementation of Erosion-Dilation Operations

Exercise

Implement the dilation (erosion) operation for window-size 1×2 by using convolutions and non-linear point-wise functions.

Hint!

$$max\{a, b\} = |a - b| + (a + b)$$

Solution

Let $A \in \mathbb{R}^{H \times W}$ be the image we want to dilate. At first let's pad it by 0-values from right by only one column. Let's denote the resulting image by $A' \in \mathbb{R}^{H \times (W+1)}$. Now let S be the result of convolution of A' by the kernel $(1, 1)$, and B be the result of convolution of A' by the kernel $(1, -1)$. Then the dilation of the image A with the window-size 1×2 will be the image $D = abs(B) + S$, where abs is the point-wise operation of taking the absolute value.

Implementation of Erosion-Dilation Operations

Exercise

By the same way implement the erosion operation for window-size 1×2.

Implementation of Erosion-Dilation Operations

Exercise

By the same way implement the erosion operation for window-size 1×2.

Problem

Implement the operation of dilation (erosion) for arbitrary window-size $k \times k$ by using convolutions and non-linear point-wise functions.

Implementation of Erosion-Dilation Operations

Exercise

By the same way implement the erosion operation for window-size 1×2.

Problem

Implement the operation of dilation (erosion) for arbitrary window-size $k \times k$ by using convolutions and non-linear point-wise functions.

Question

In the implementation of the problem above can both the number of convolutions and the number of non-linearity layers be constant?

Introduction to Computer Vision: Filters

Navasardyan Shant

September 19, 2019