Computer Graphics Seminar

Depth of Field

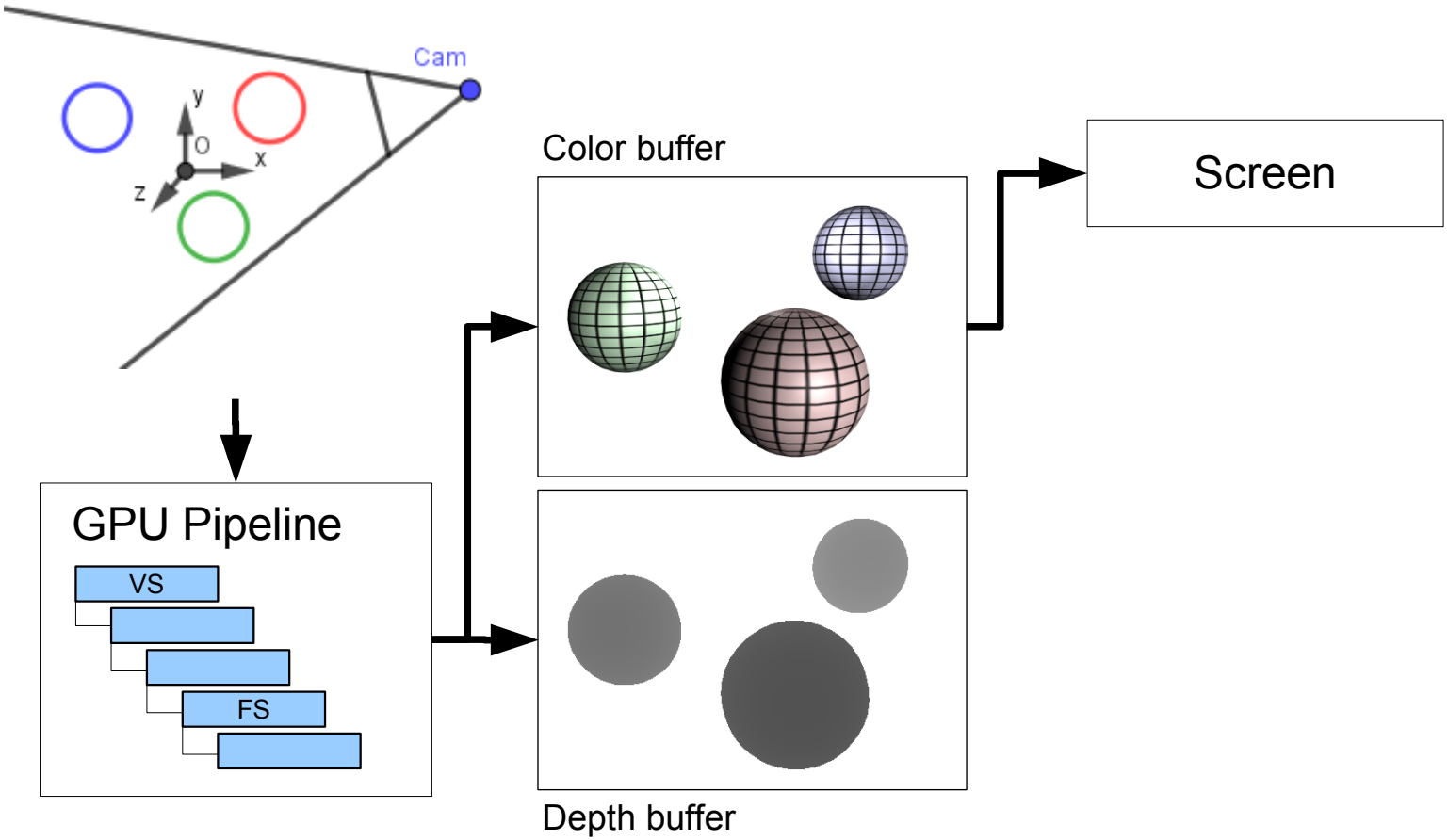
Raimond Tunnel

Post Processing Effects

- Applied to a frame **post** (after) rendering.

Post Processing Effects

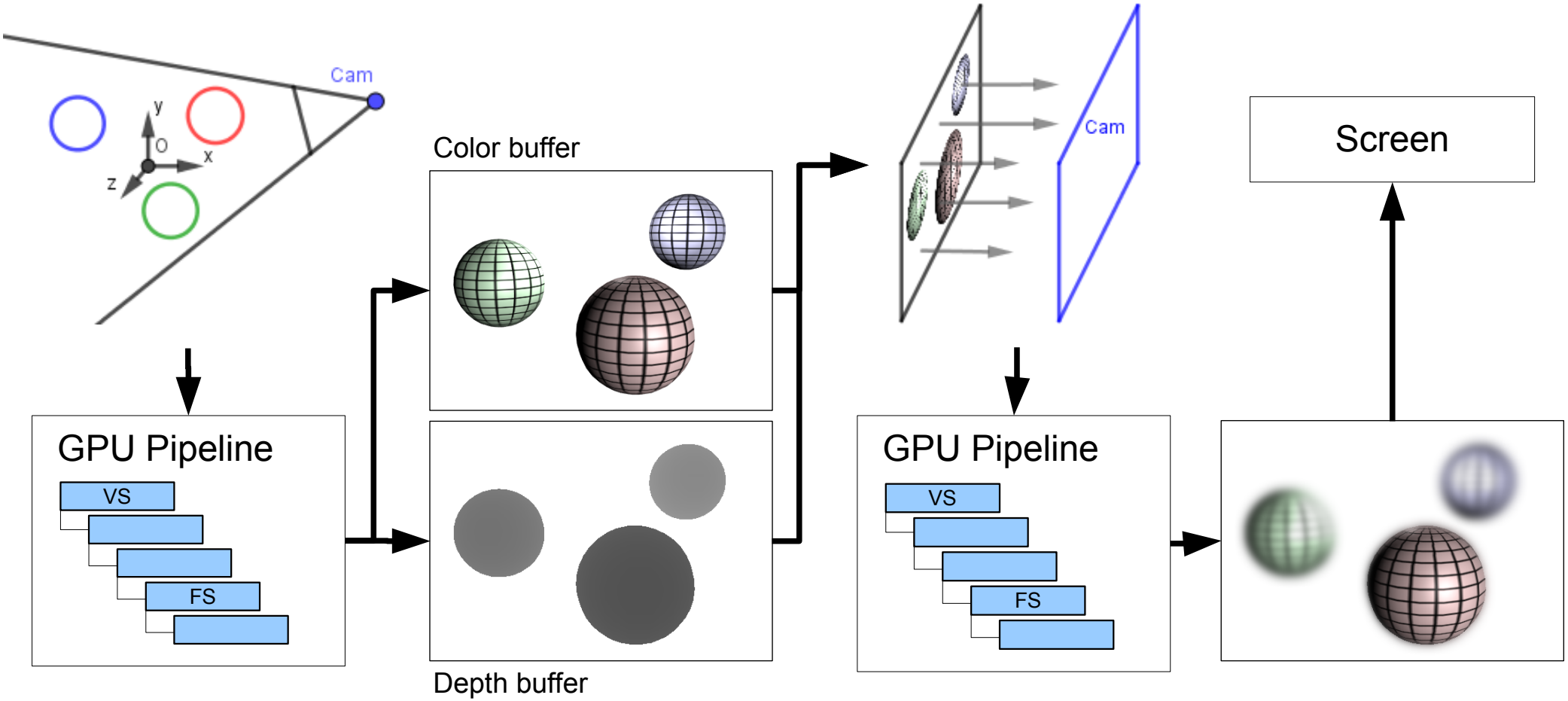
- Applied to a frame **post** (after) rendering.



Regular rendering pass

Post Processing Effects

- Applied to a frame **post** (after) rendering.

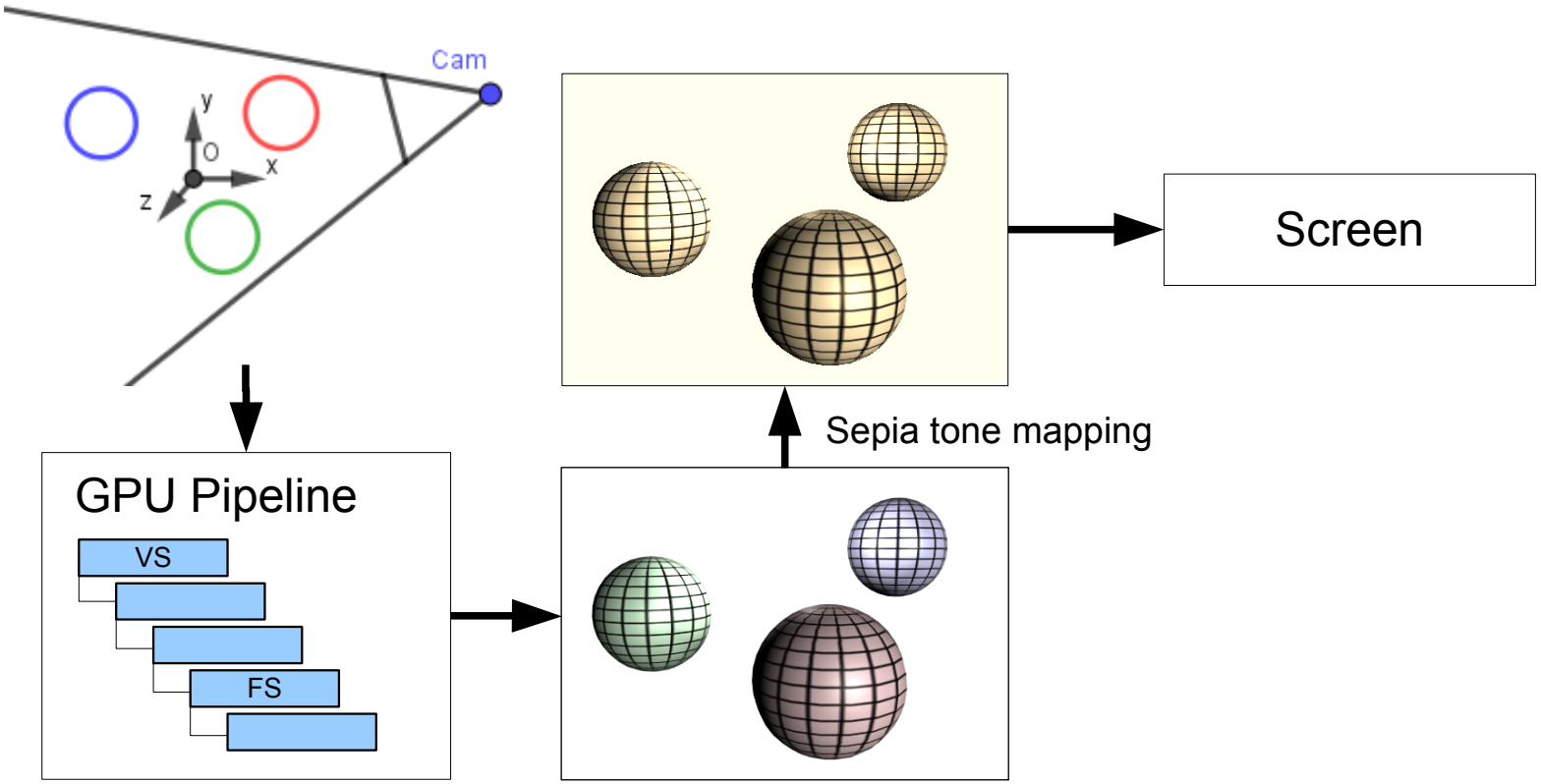


Regular rendering pass

Post effect rendering pass

Post Processing Effects

- Applied to a frame **post** (after) rendering.
- Some could be applied in the initial pass.



Regular rendering pass

Post Processing Effects

- Lots of different effects!

- Anti-aliasing
- Eye Adaption
- Blendables (lerp)
- Color Grading
- Filmic Tonemapper
- Depth of Field
- Panini Projection
- Chromatic Aberration
- Screen Space Reflections
- Vignette
- Bloom
- ...

https://docs.unrealengine.com/en-us/Engine/Rendering/PostProcessEffects

Post Processing Effects

- Lots of different effects!

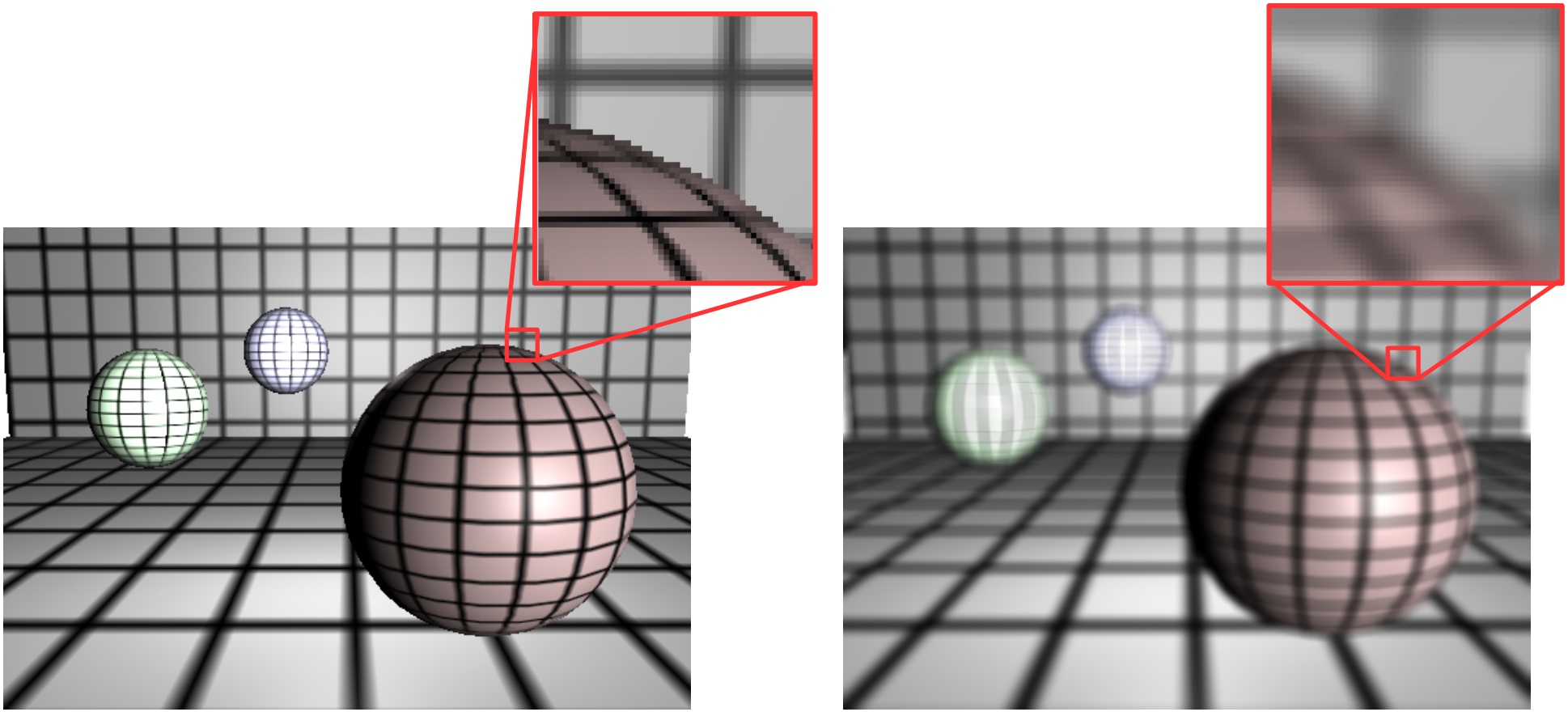
- Anti-aliasing
- Eye Adaption
- Fog
- Color Grading
- Grain
- Depth of Field
- Ambient Occlusion
- Chromatic Aberration
- Screen Space Reflections
- Vignette
- Bloom
- Dithering
- Motion Blur
- ...

https://docs.unity3d.com/Manual/PostProcessingOverview.html

Blurs and Convolution

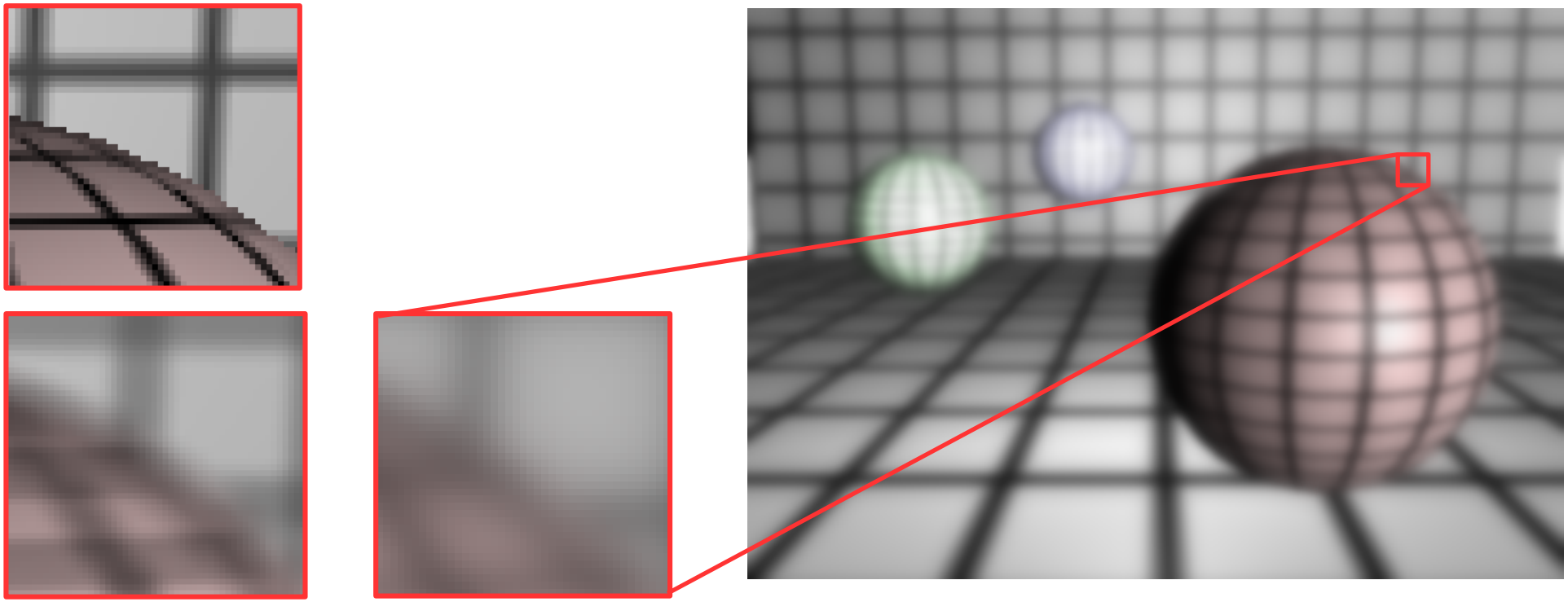
Blur

- Every pixel becomes a combination of pixels from a neighbourhood.



Blur

- Every pixel becomes a combination of pixels from a neighbourhood.
- Different possibilities for combinations.

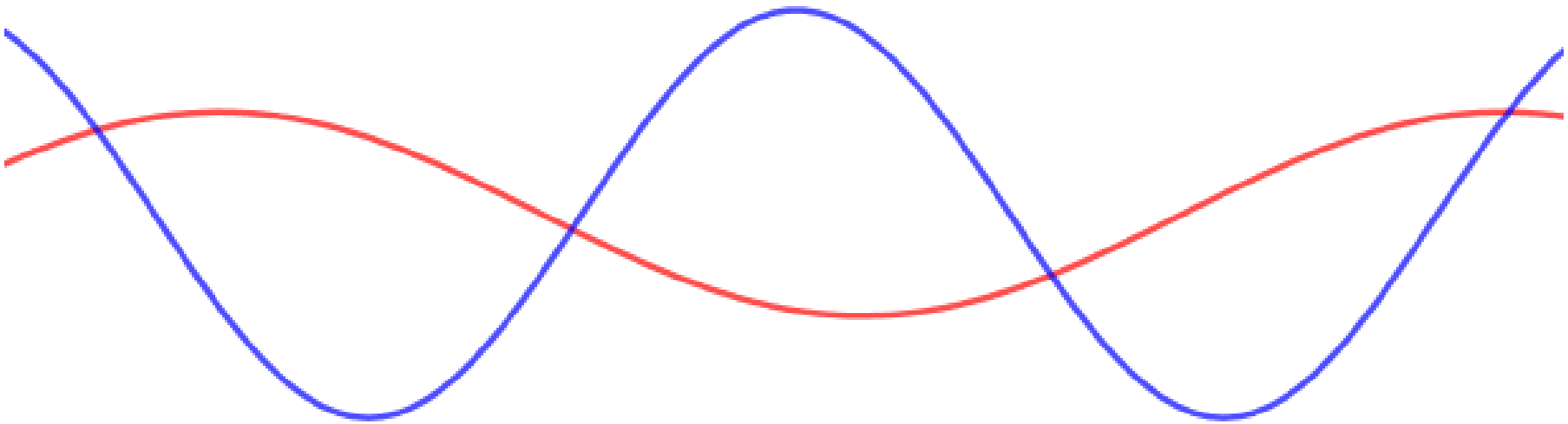


Box blur

Gaussian blur

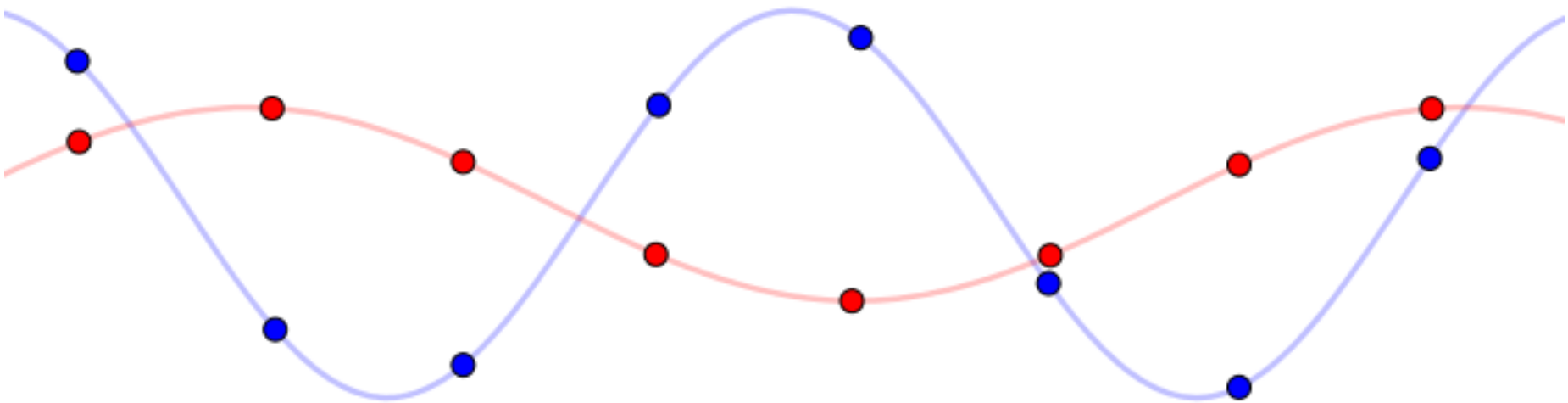
Convolution

- Operation between 2 signals.



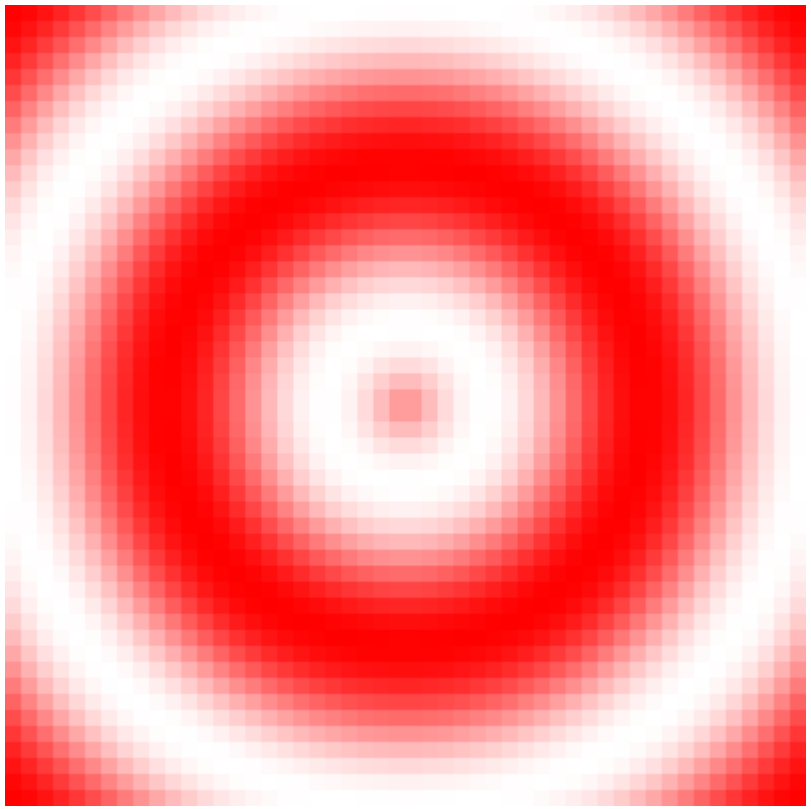
Convolution

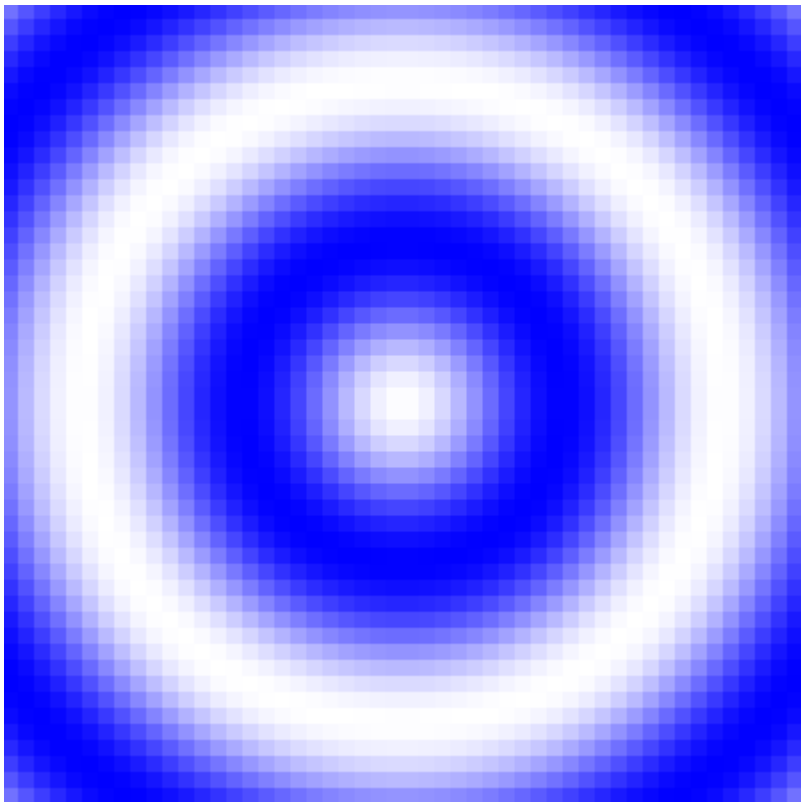
- Operation between 2 discrete signals.



Convolution

- Operation between 2 discrete 2D signals.





Convolution

- Operation between 2 matrices.

$$\begin{pmatrix} k_{0,0} & \dots & k_{p,0} \\ \dots & \dots & \dots \\ k_{0,s} & \dots & k_{p,s} \end{pmatrix} * \begin{pmatrix} p_{0,1} & p_{0,2} & \dots & p_{0,n} \\ p_{1,0} & p_{1,2} & \dots & p_{1,n} \\ \dots & \dots & \dots & \dots \\ p_{m,0} & p_{m,0} & \dots & p_{m,n} \end{pmatrix}$$

Convolution

- Operation between 2 matrices.
- In digital image processing:

$$\begin{pmatrix} k_{0,0} & \dots & k_{s,0} \\ \dots & \dots & \dots \\ k_{0,s} & \dots & k_{s,s} \end{pmatrix} * \begin{pmatrix} p_{0,1} & p_{0,2} & \dots & p_{0,n} \\ p_{1,0} & p_{1,2} & \dots & p_{1,n} \\ \dots & \dots & \dots & \dots \\ p_{m,0} & p_{m,0} & \dots & p_{m,n} \end{pmatrix}$$

Kernel
{odd}x{odd}

Image
(picture, any dim)

Convolution

- New element is a combination of kernel elements and the current neighbourhood.

$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(-\Delta x,-\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

$$K*P=Q$$

$$K\in Mat(s\times s)$$

$$P\in Mat(m\times n) \qquad r=\left\lfloor\frac{s}{2}\right\rfloor$$ (radius)

$$Q\in Mat((m+r)\ \times\ (n+r))\approx Mat(m\times n)$$

Convolution

- New element is a combination of kernel elements and the current neighbourhood.

$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r} K(\underline{-}\Delta x,\underline{-}\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

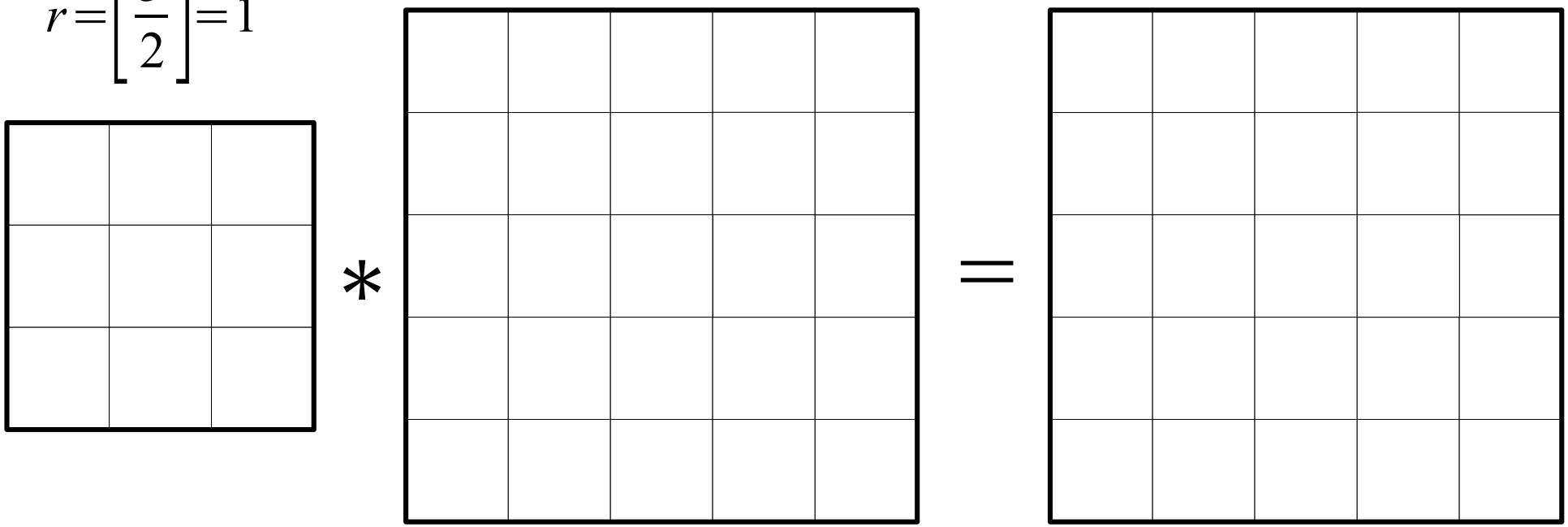
- For **symmetric kernels**:

$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r} K(\underline{+}\Delta x,\underline{+}\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

Convolution

$$(K * P)(x, y) = \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(\Delta x, \Delta y) \cdot P(x+\Delta x, y+\Delta y)$$

$$r = \left\lfloor \frac{3}{2} \right\rfloor = 1$$ (radius)



Convolution

$$(K * P)(x, y) = \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(\Delta x, \Delta y) \cdot P(x+\Delta x, y+\Delta y)$$

a	**b**	**a**
b	**c**	**b**
a	**b**	**a**

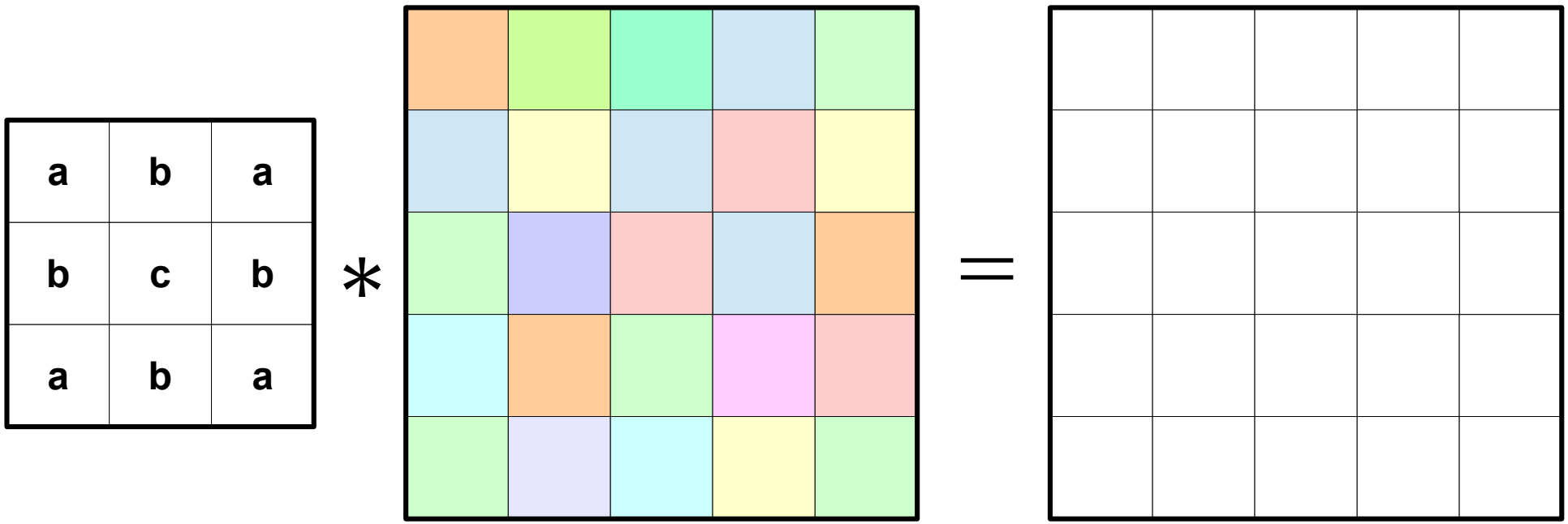
Symmetric kernel

$*$

$=$

Convolution

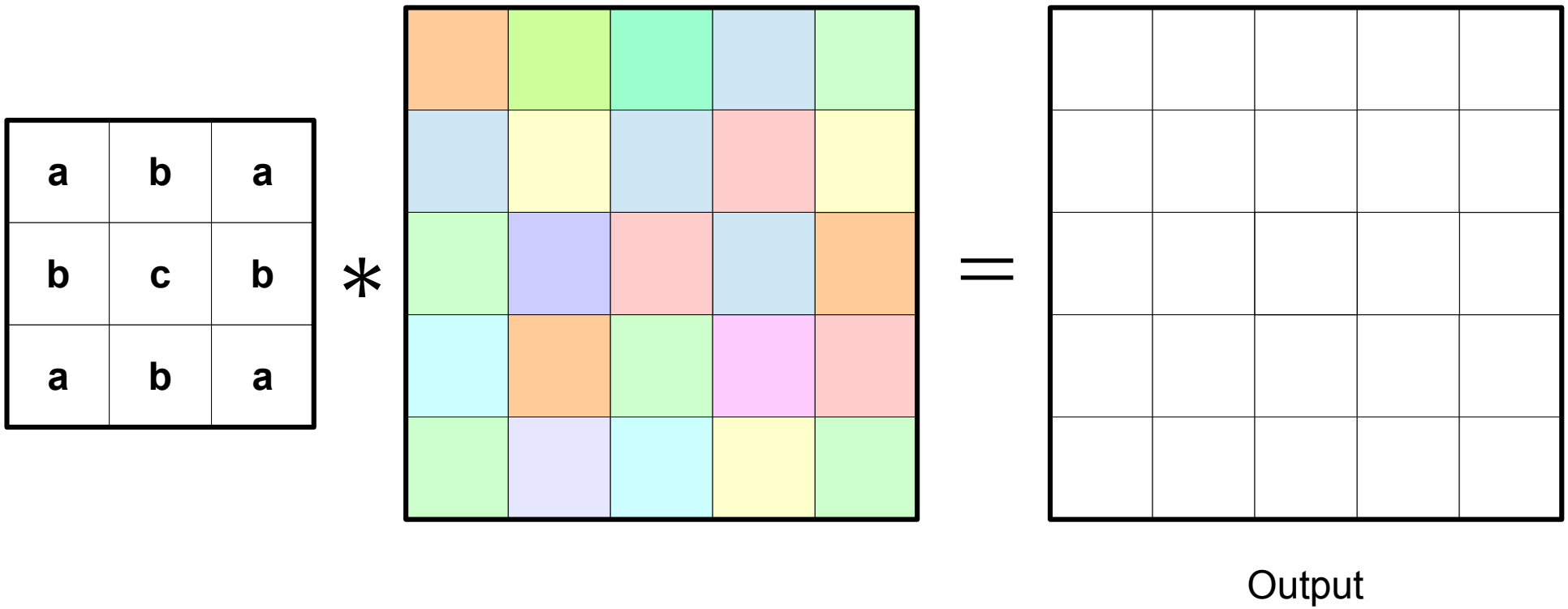
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$



Some raster image
(picture)

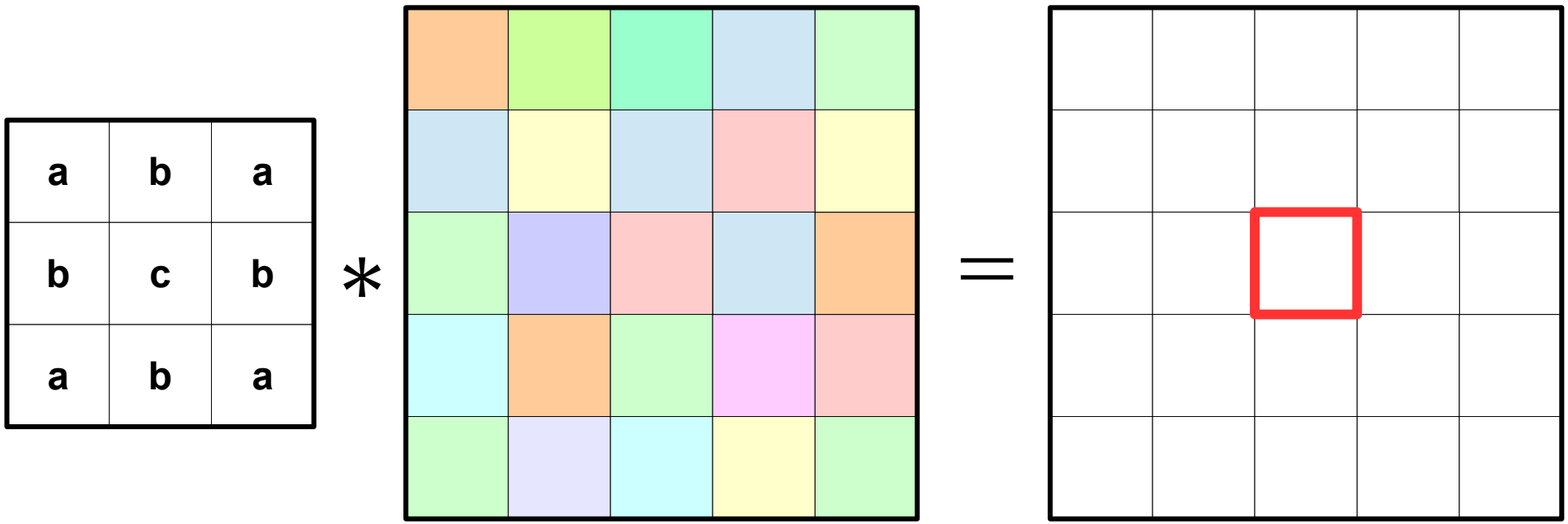
Convolution

$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$



Convolution

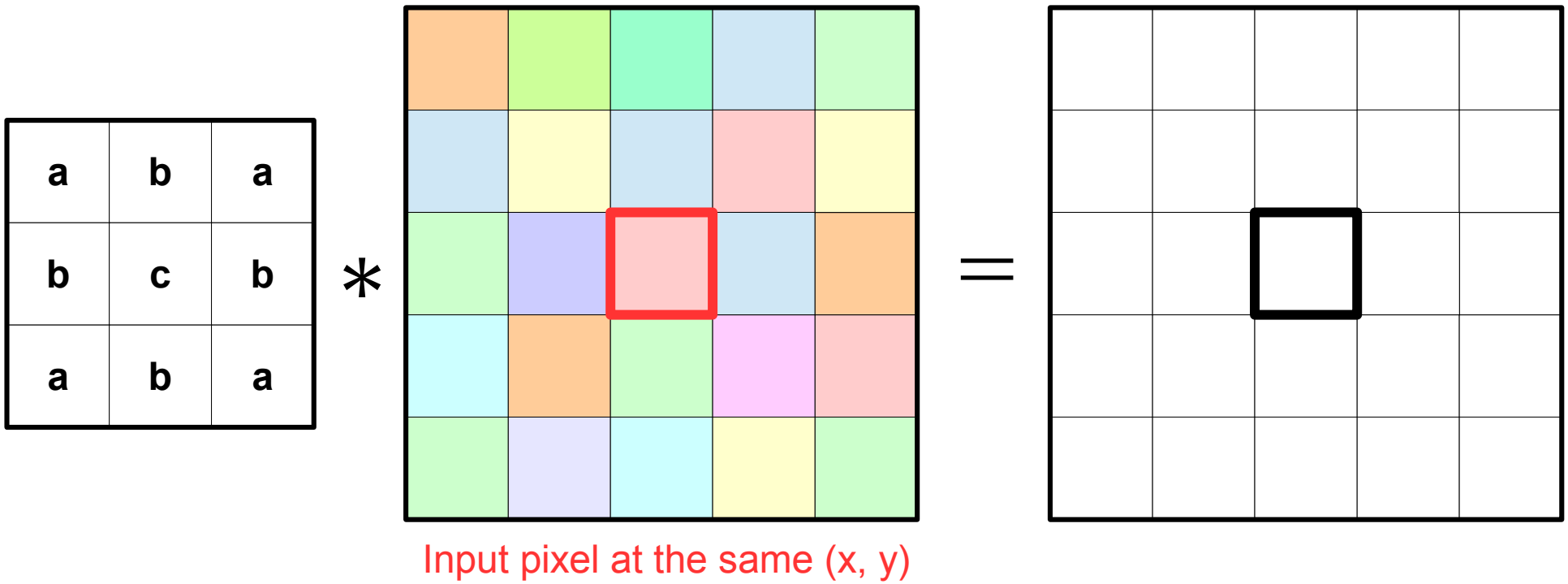
$$(K * P)(x, y) = \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(\Delta x, \Delta y) \cdot P(x+\Delta x, y+\Delta y)$$



Single output pixel at some (x, y)

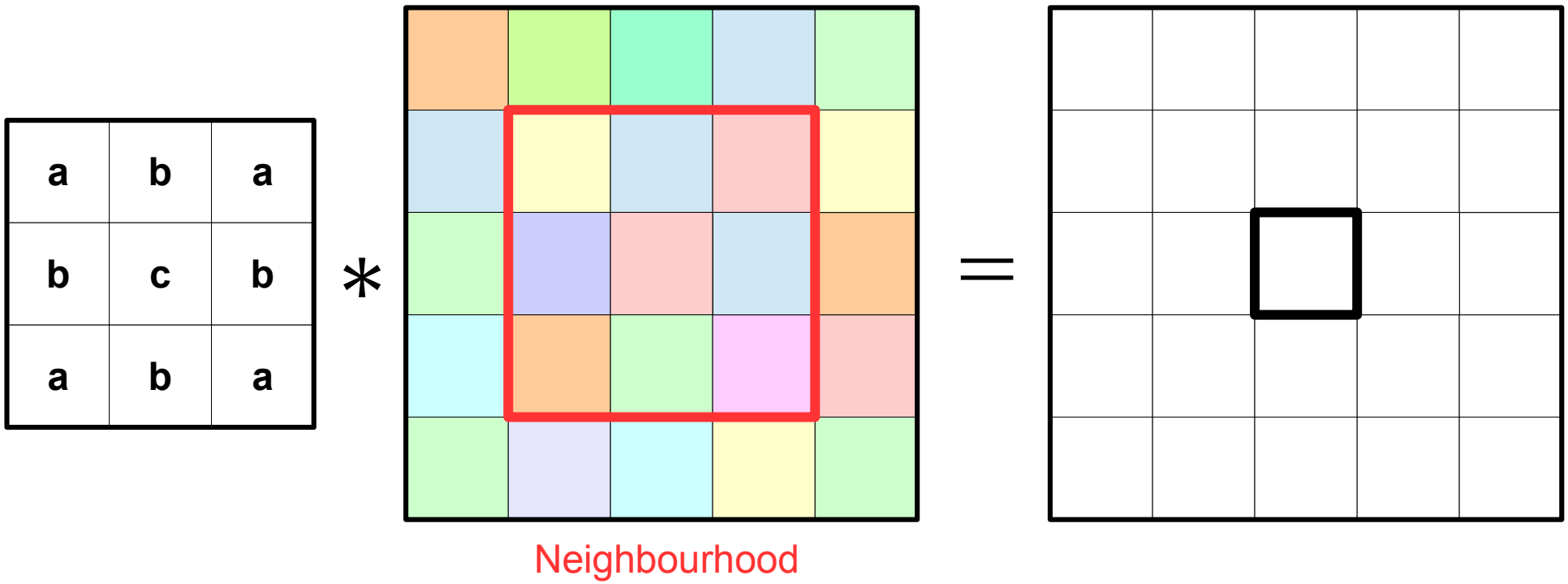
Convolution

$$(K * P)(x, y) = \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(\Delta x, \Delta y) \cdot P(x+\Delta x, y+\Delta y)$$



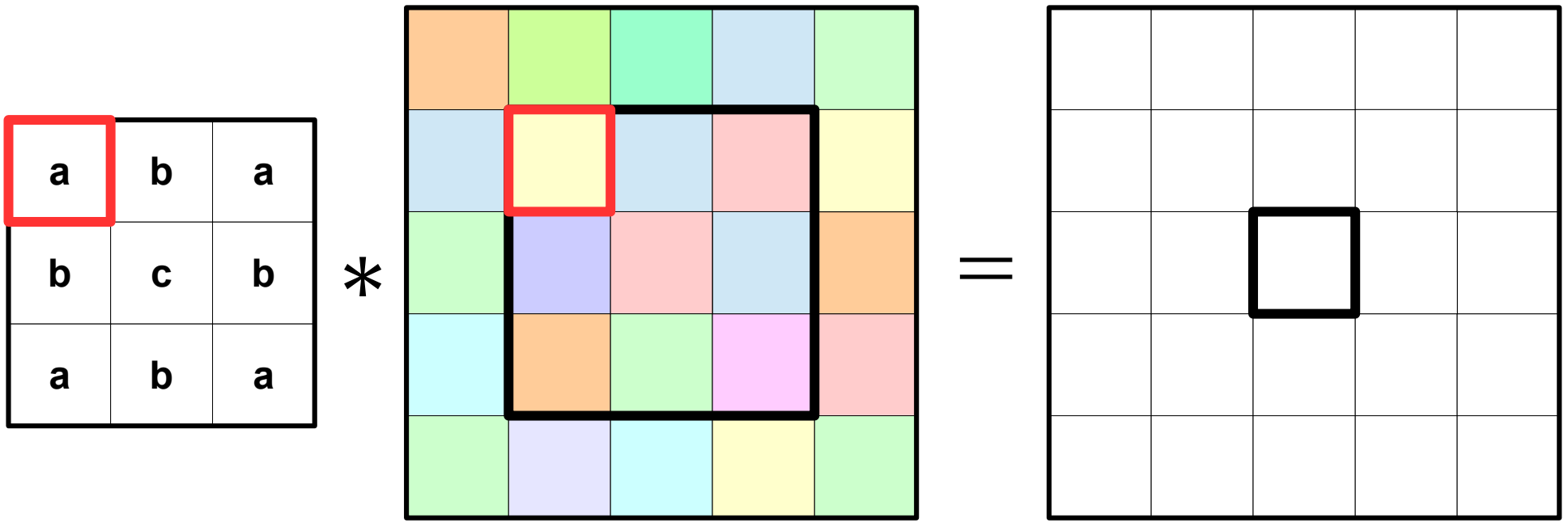
Convolution

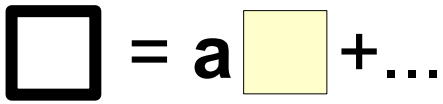
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$



Convolution

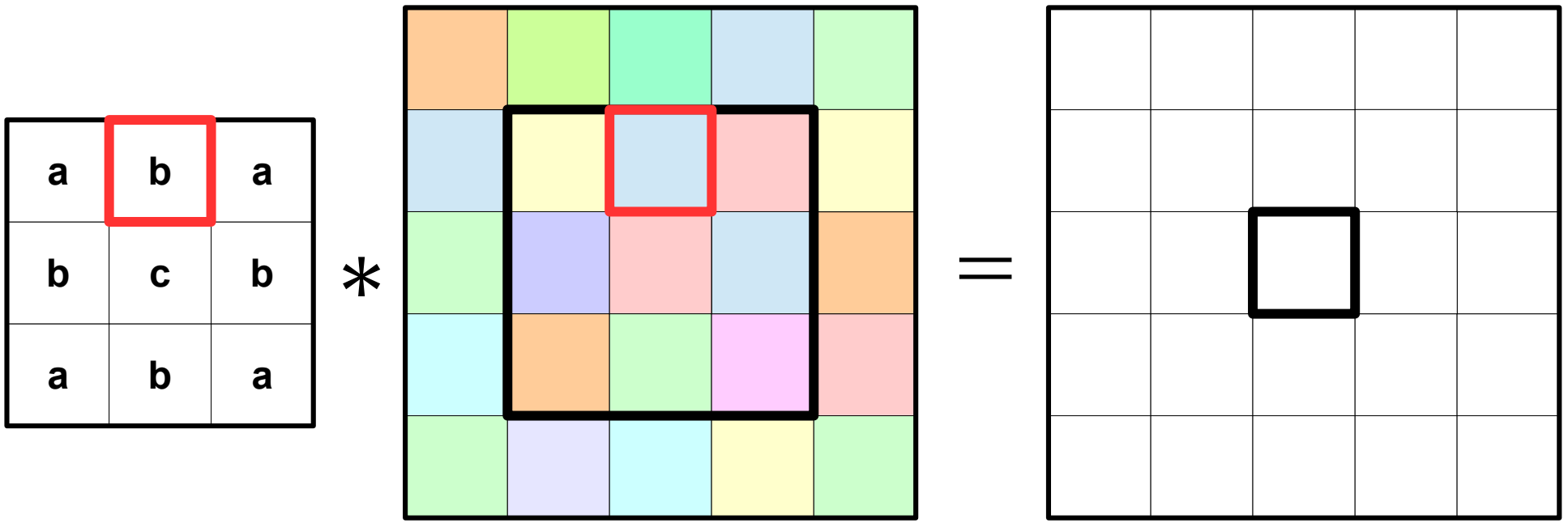
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

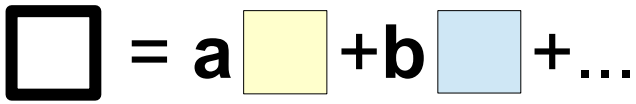




Convolution

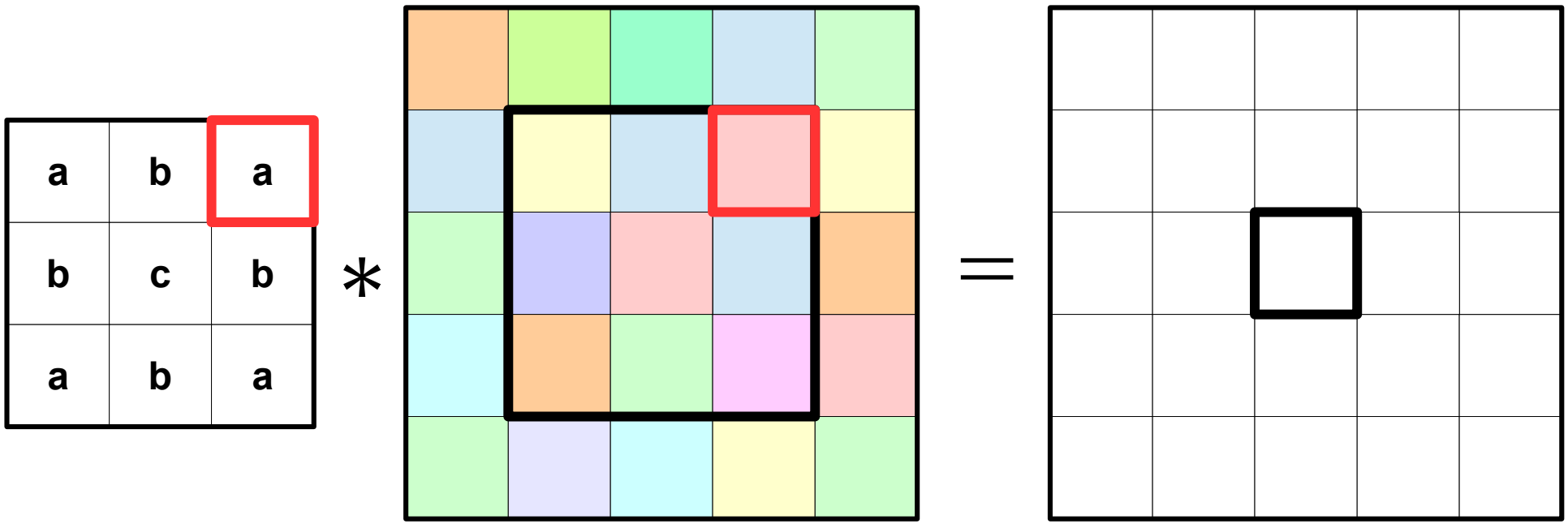
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

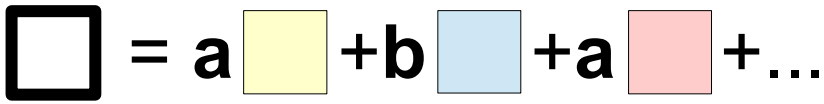




Convolution

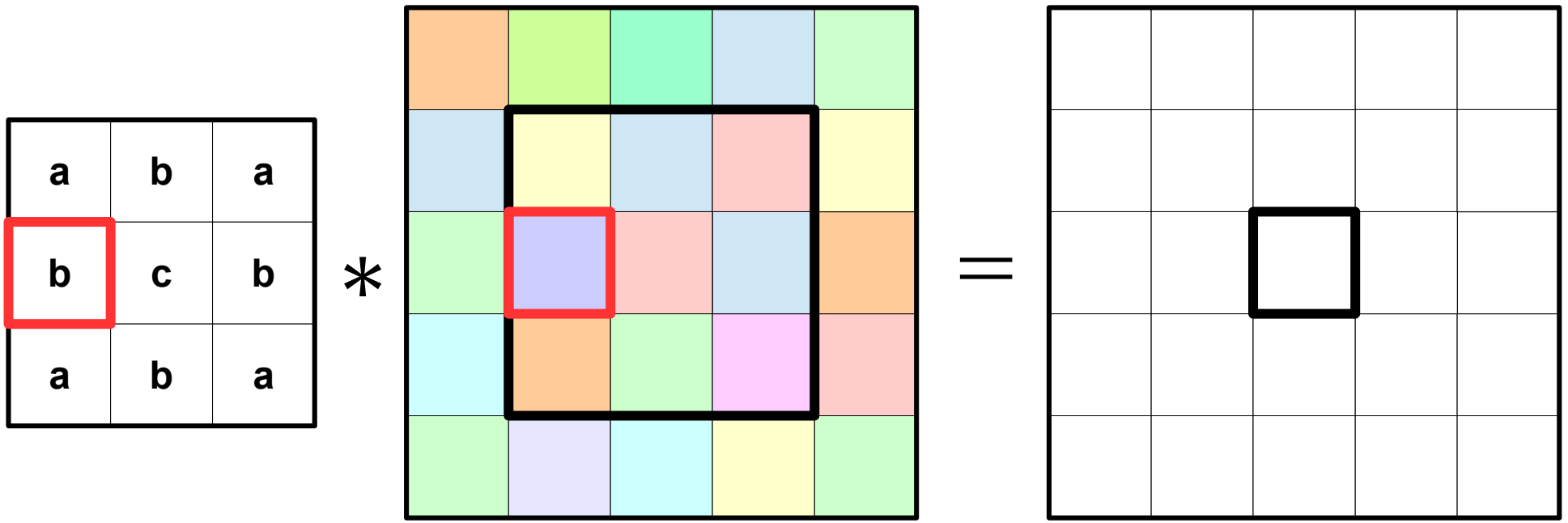
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

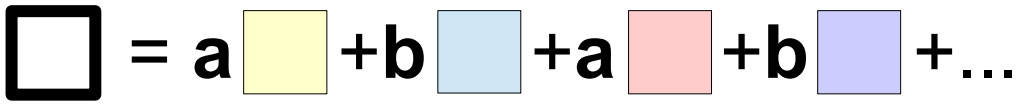




Convolution

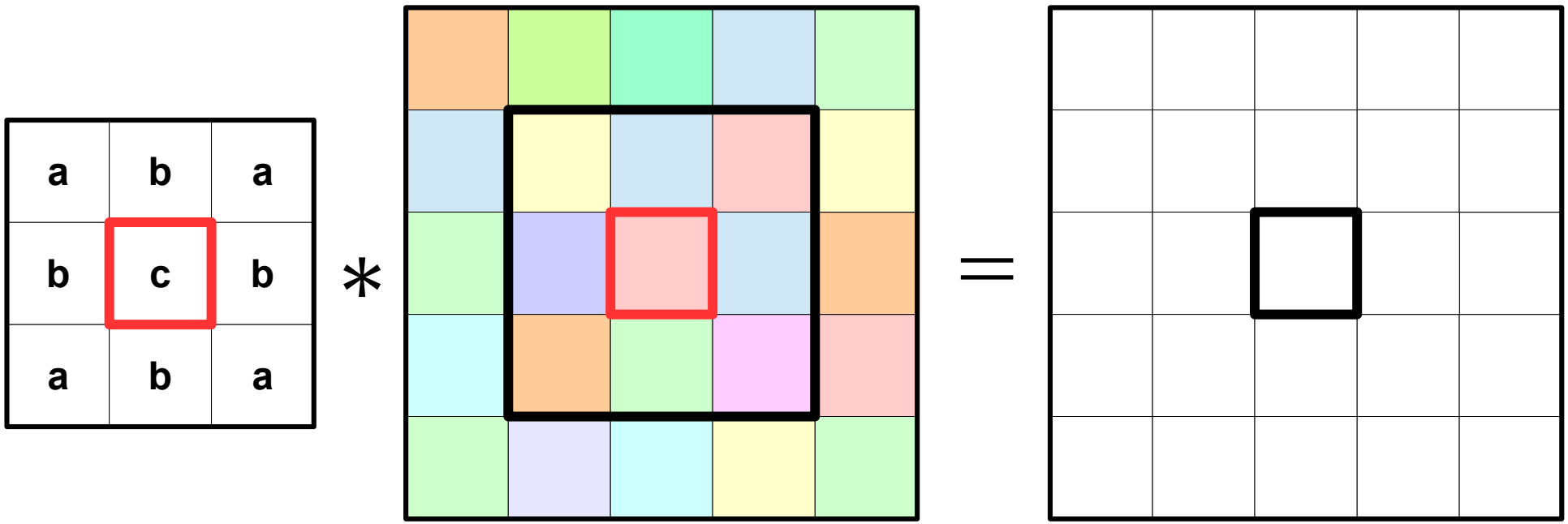
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

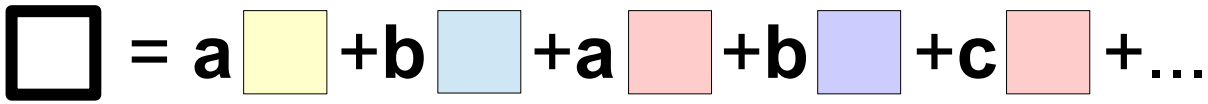




Convolution

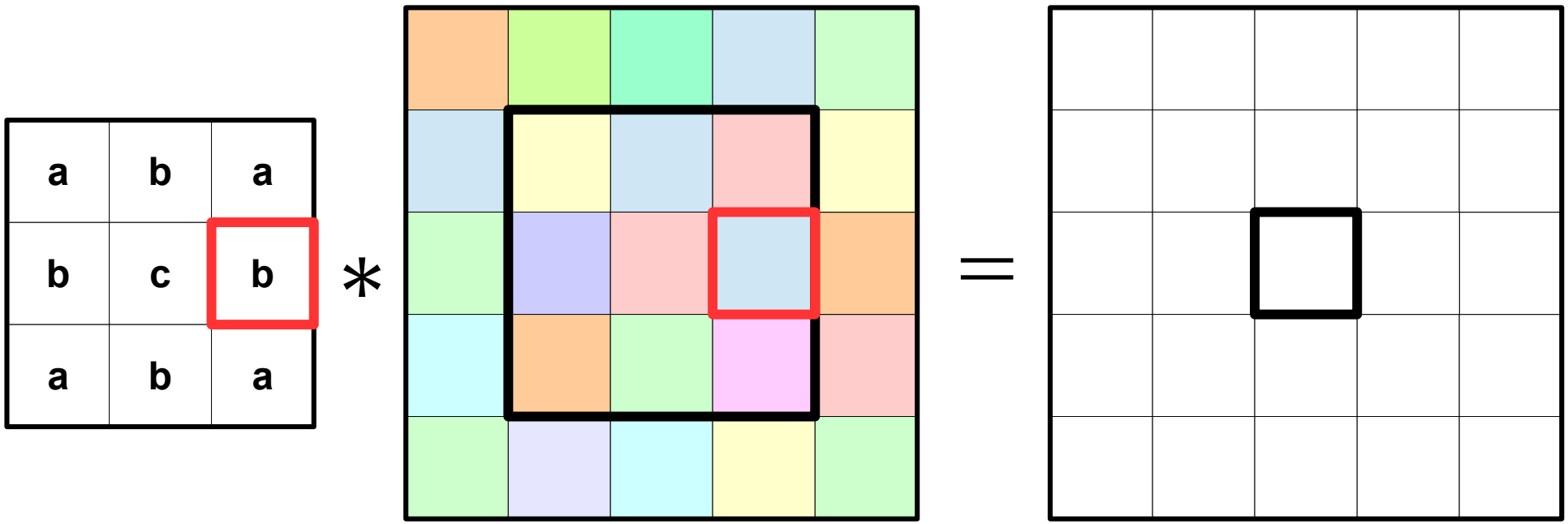
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

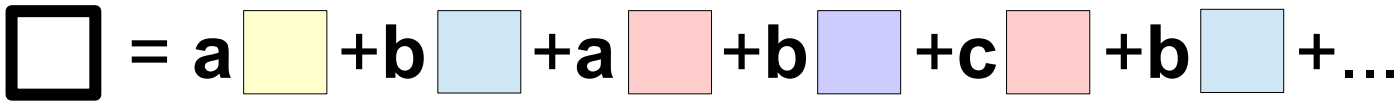




Convolution

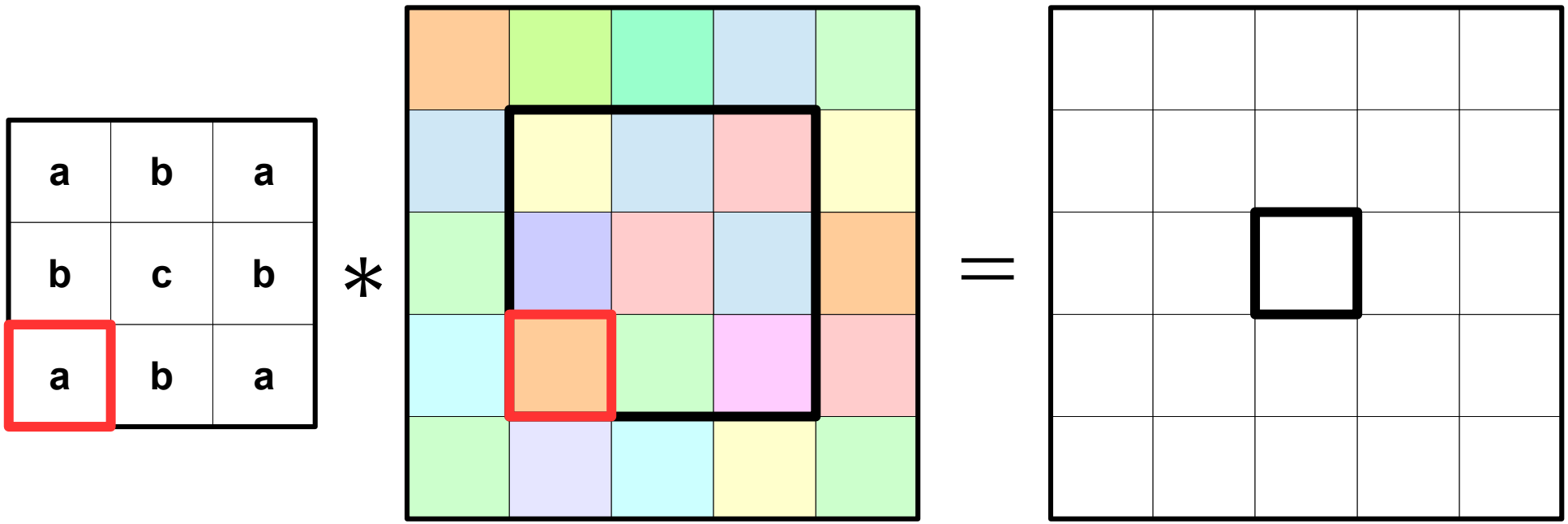
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

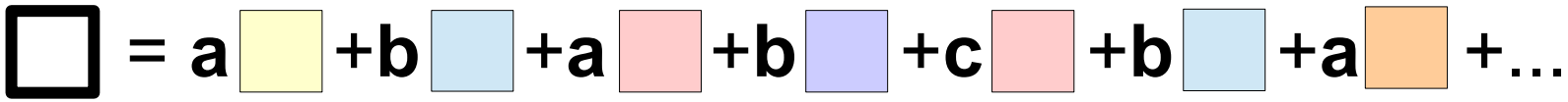




Convolution

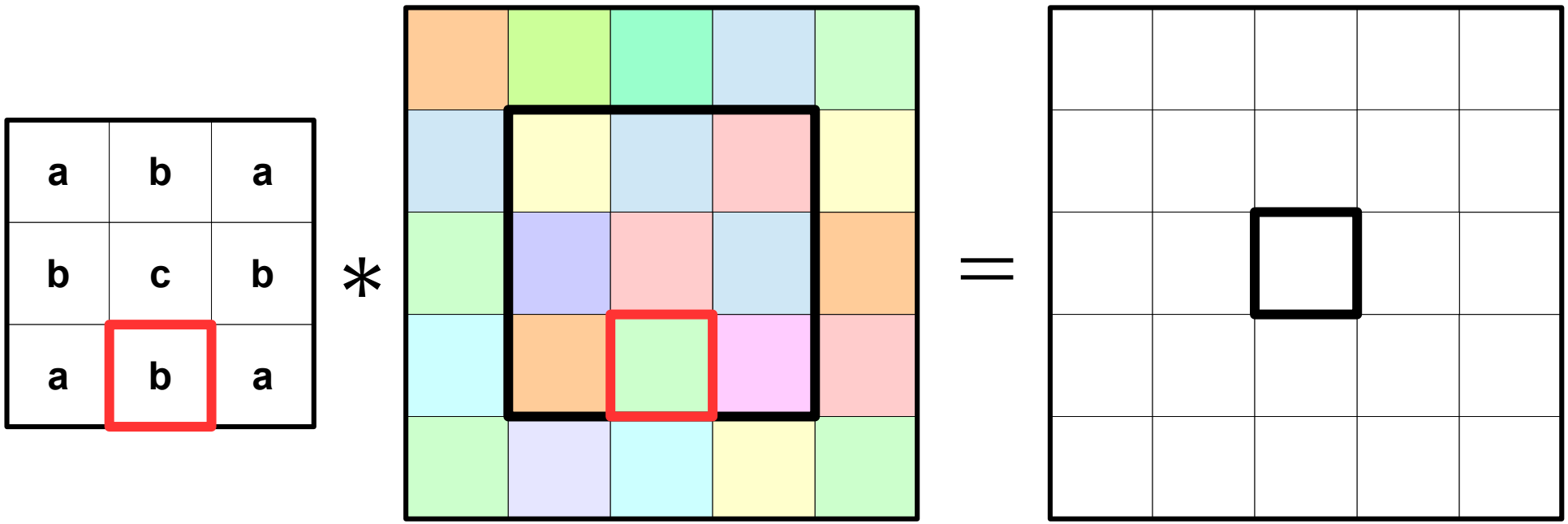
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

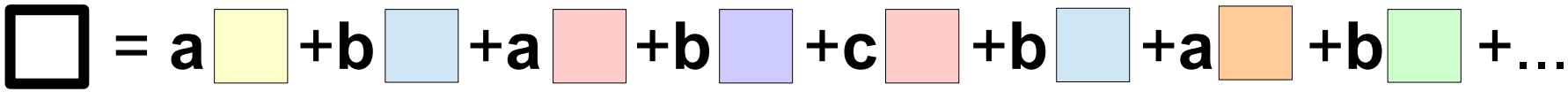




Convolution

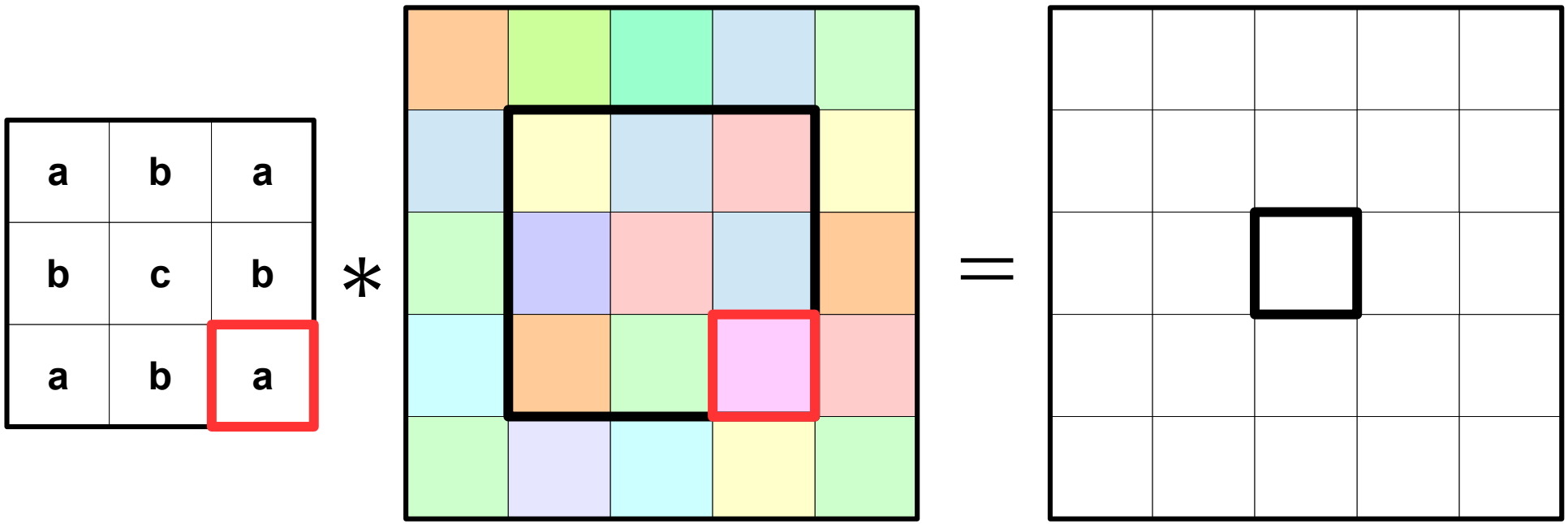
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$

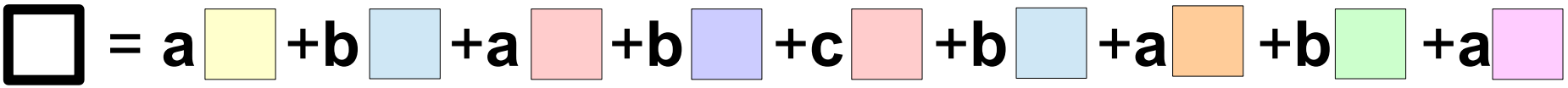




Convolution

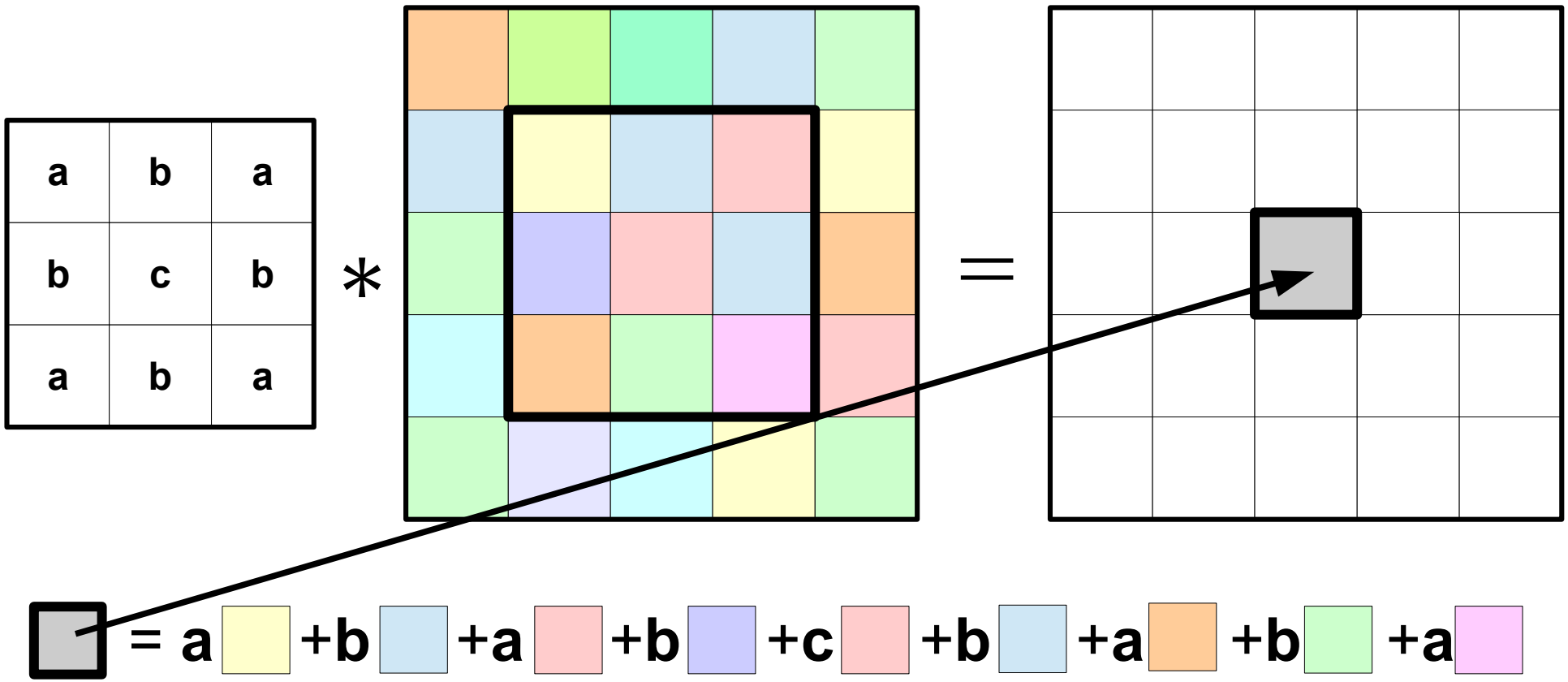
$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$





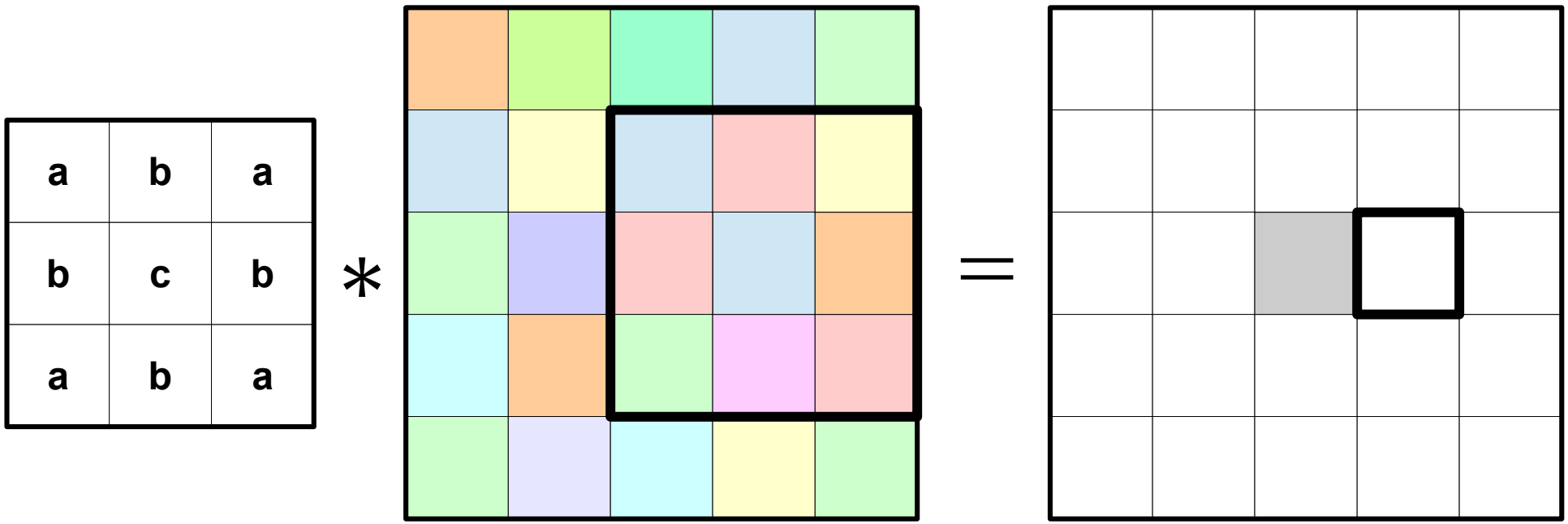
Convolution

$$(K*P)(x,y)=\sum_{\Delta x=-r}^{\Delta x=r}\sum_{\Delta y=-r}^{\Delta y=r}K(\Delta x,\Delta y)\cdot P(x+\Delta x,y+\Delta y)$$



Convolution

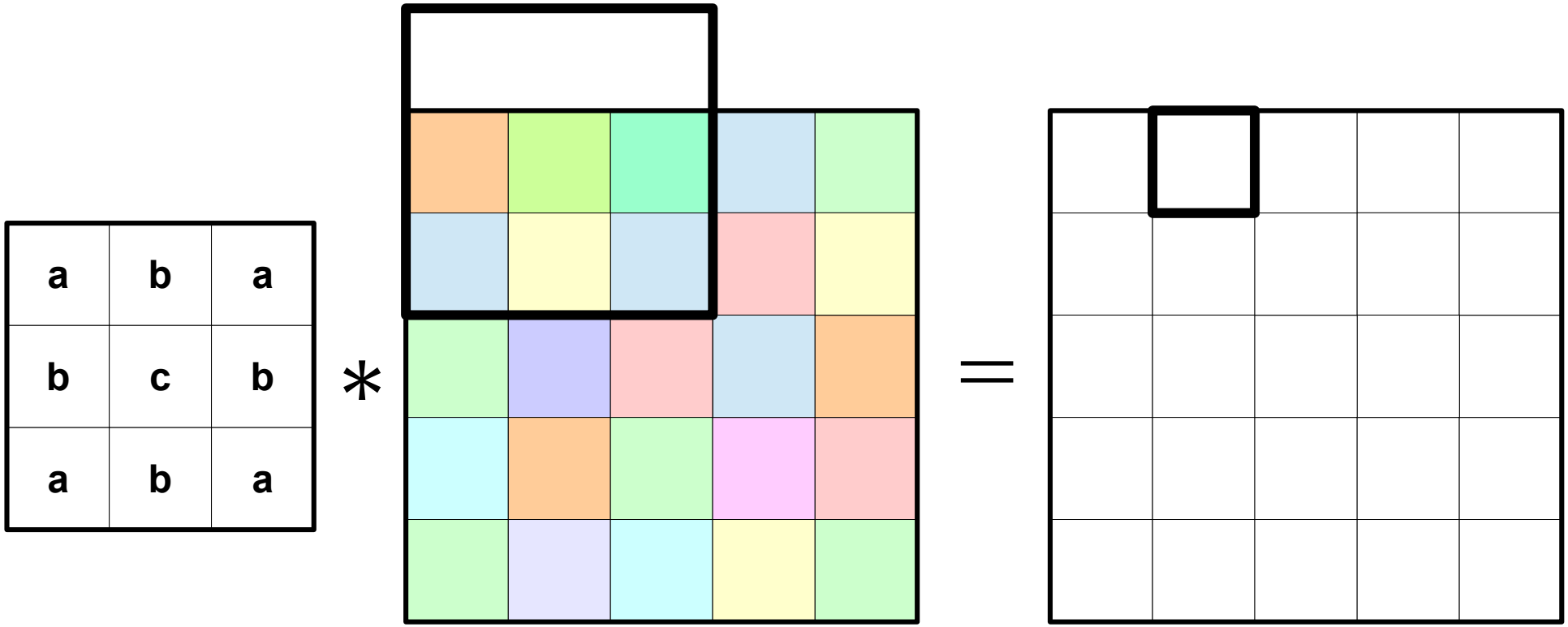
$$(K * P)(x, y) = \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(\Delta x, \Delta y) \cdot P(x+\Delta x, y+\Delta y)$$



- All output elements can be calculated like that.

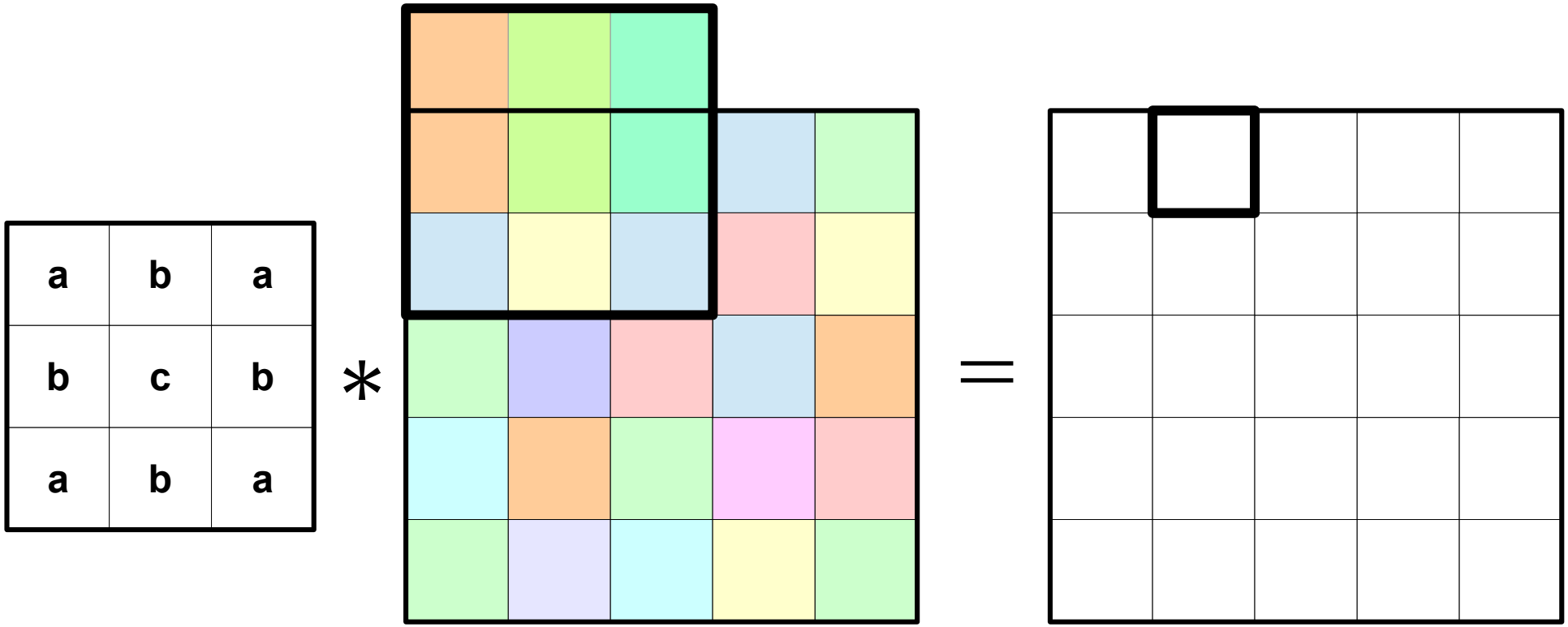
Convolution

- When the neighbourhood is outside the input:



Convolution

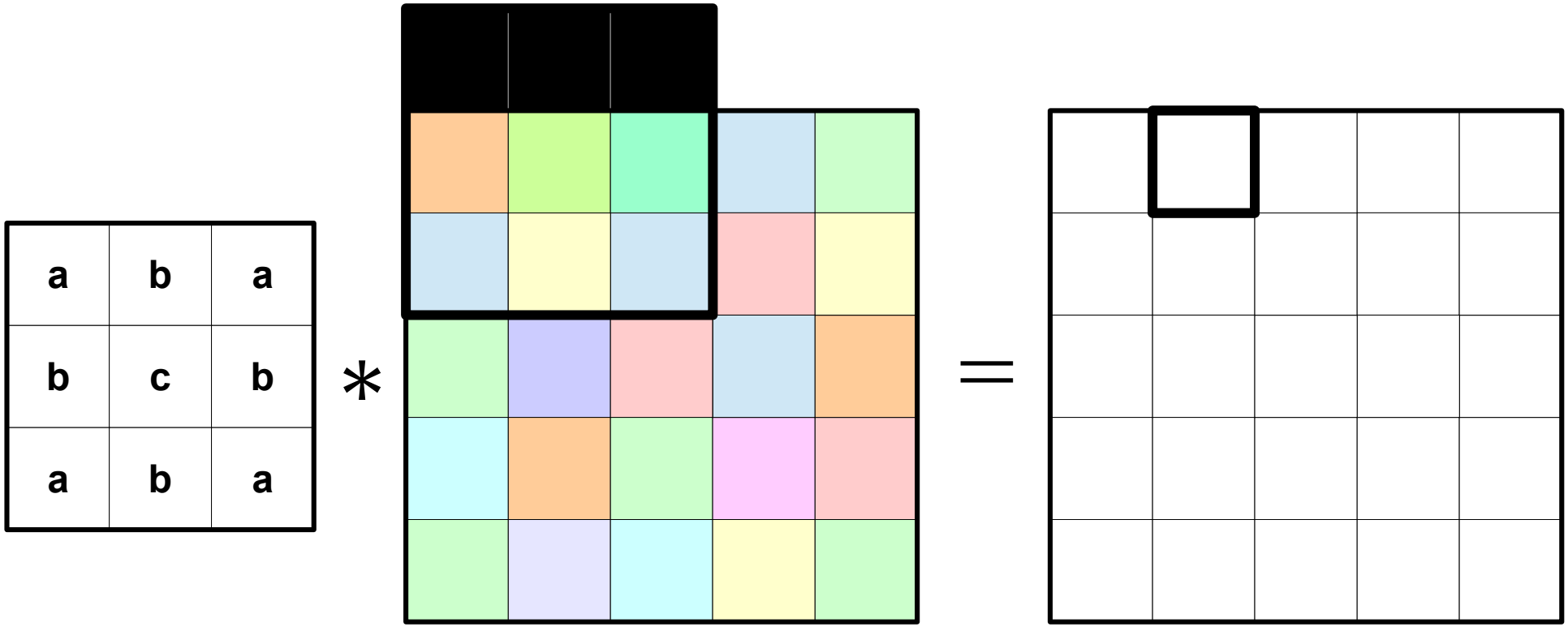
- When the neighbourhood is outside the input:



- The values could be repeated edge values

Convolution

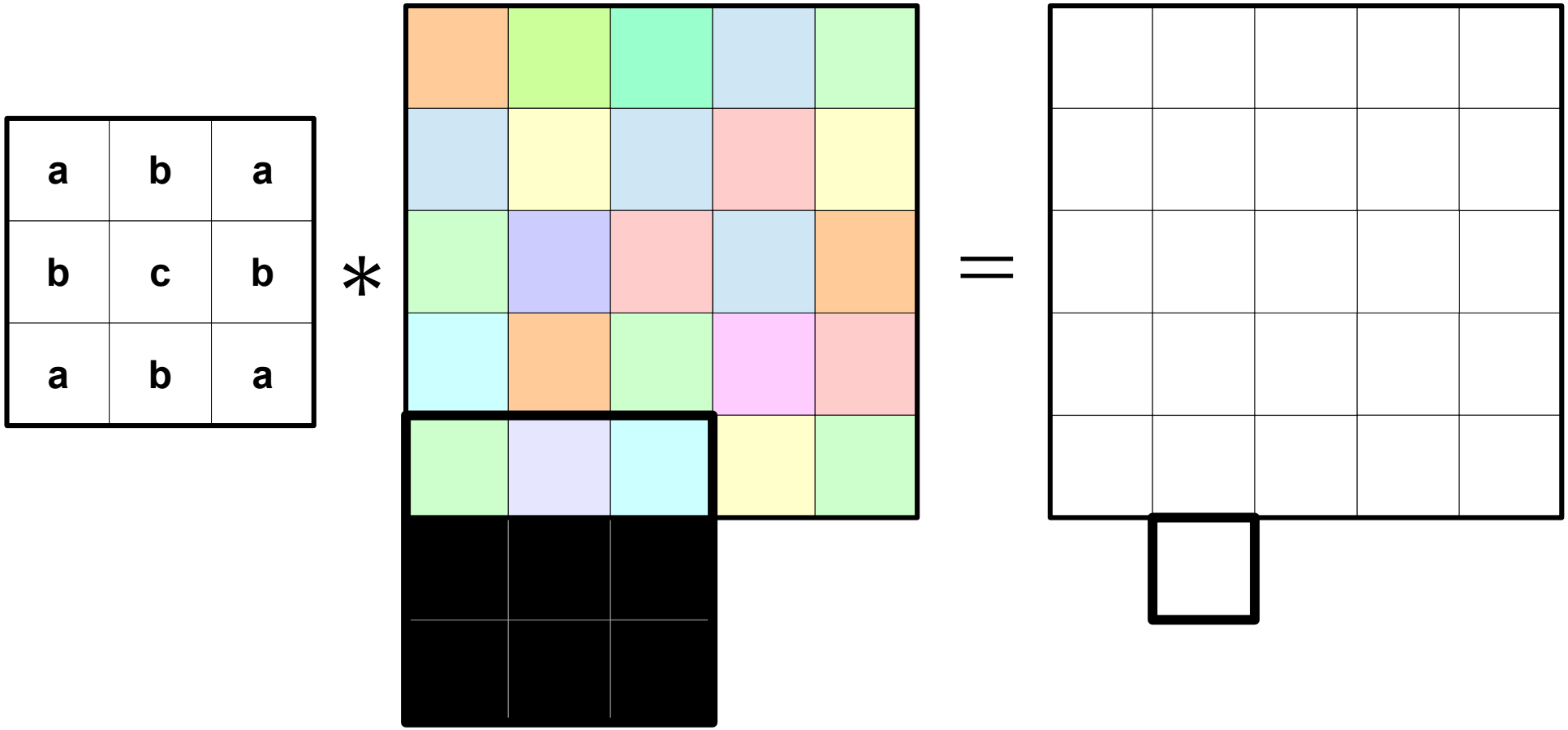
- When the neighbourhood is outside the input:



- The values could be zero values (*eg* with alpha)

Convolution

- We could also calculate some outside values:



Convolution

- Naive implementation complexity:

$$(K * P) = O(s^2 \cdot m \cdot n) = O(n^4)$$

$$K * P = Q$$

$$K \in Mat(s \times s)$$

$$P \in Mat(m \times n)$$

Convolution

- Naive implementation complexity:

$$(K * P) = O(\underline{s}^2 \cdot m \cdot n) = O(n^4)$$

Kernel size

$$K * P = Q$$

$$K \in Mat(s \times s)$$

$$P \in Mat(m \times n)$$

Convolution

- Naive implementation complexity:

$$(K * P) = O(s^2 \cdot \underline{m \cdot n}) = O(n^4)$$

Image size

$$K * P = Q$$

$$K \in Mat(s \times s)$$

$$P \in Mat(m \times n)$$

Convolution

- Naive implementation complexity:

$$(K * P) = O(s^2 \cdot m \cdot n) = O(\underline{n^4})$$

Longest dimension

$$K * P = Q$$

$$K \in Mat(s \times s)$$

$$P \in Mat(m \times n)$$

Convolution

- Naive implementation complexity:

$$(K * P) = O(s^2 \cdot m \cdot n) = O(n^4)$$

- If our fragment shader runs parallel on every output pixel, we are left with:

$$(K * P) = O(s^2)$$ (for a single pixel)

Convolution

- Naive implementation complexity:

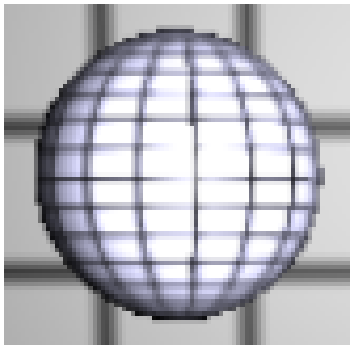
$$(K * P) = O(s^2 \cdot m \cdot n) = O(n^4)$$

- If our fragment shader runs parallel on every output pixel, we are left with:

$$(K * P) = O(s^2)$$ (for a single pixel)

- That is still not very good if our kernel is larger.

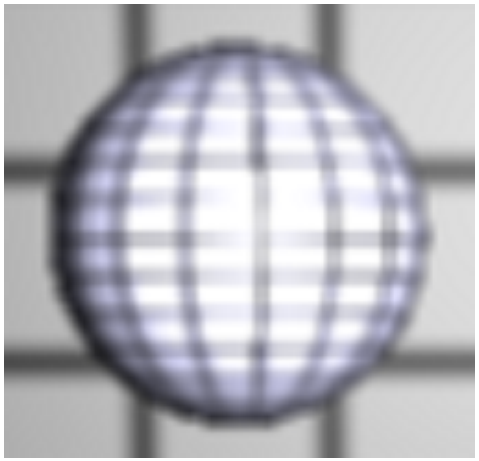
Box Blur



- Finds the average value.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

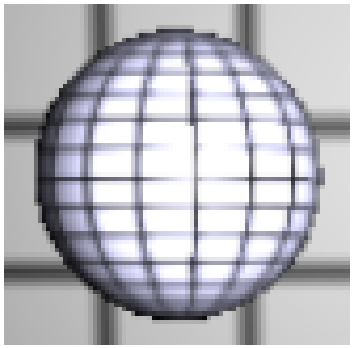
Kernel



Output

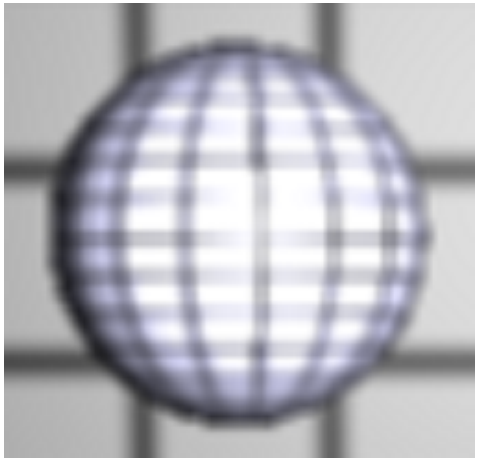
Also called the median filter.

Box Blur

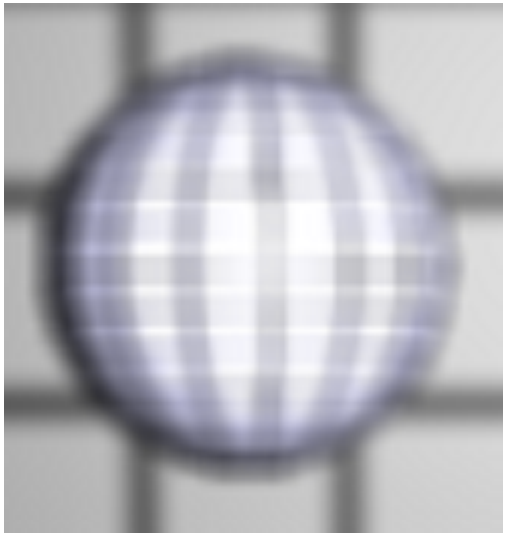


- Finds the average value.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



Aliasing

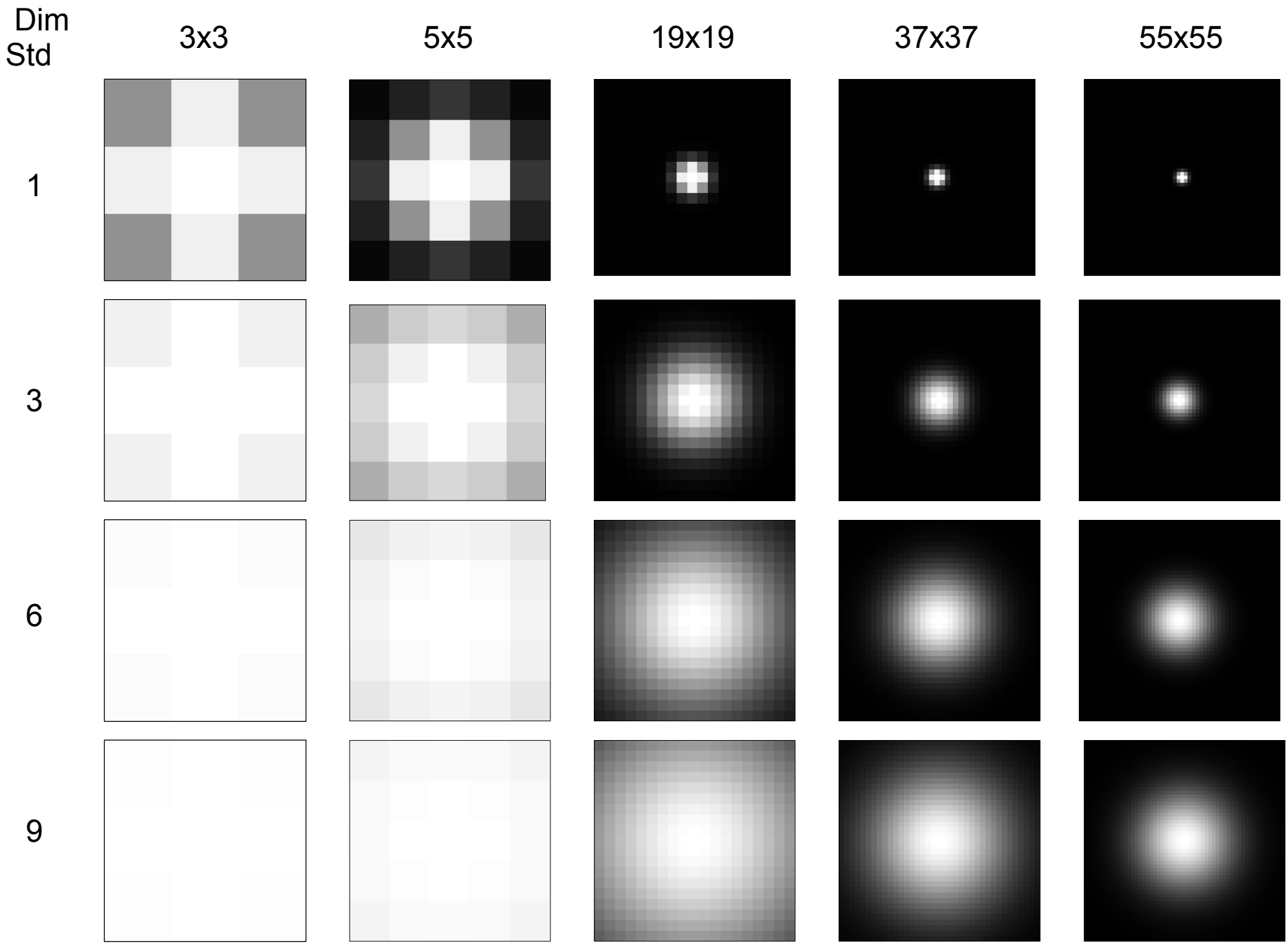
Gaussian Blur

- Kernel is a sampling of a 2D Gaussian function

$g(-1, 1)$	$g(0, 1)$	$g(1, 1)$
$g(-1, 0)$	$g(0, 0)$	$g(1, 0)$
$g(-1, -1)$	$g(0, -1)$	$g(1, -1)$

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

σ – standard deviation



Kernels normalized to have 1 in the middle

What is the best combination and why?

Gaussian Blur

dim=72

std=1

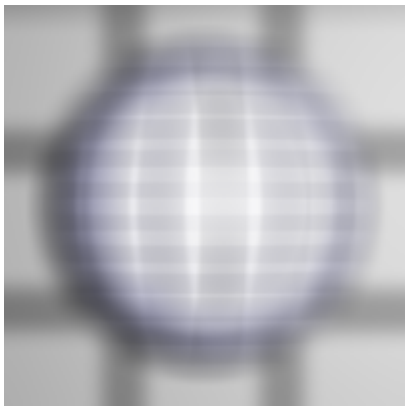
std=3

std=6

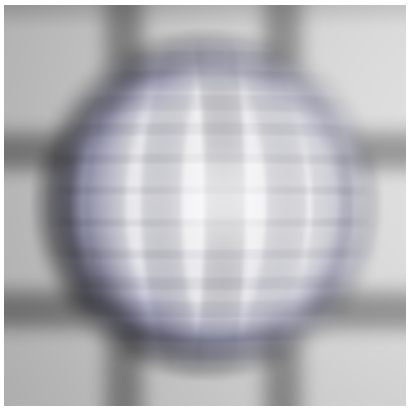
std=9

std=12

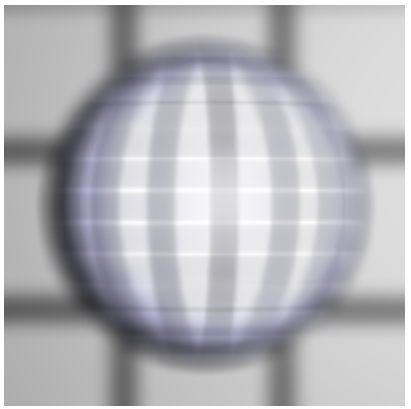
std=12, dim=28



std=12, dim=22



std=12, dim=18



std=12, dim=12

Aliasing

Separable Kernels

- If the columns of K are linearly dependent*, then:

$$\underline{K} = \underline{v}_0 \cdot \underline{v}_1$$

$$Mat(s \times s) \qquad Mat(s \times 1) \quad Mat(1 \times s)$$

* Matrix K is of rank 1 – the vector space spanned by the columns is 1D.
All the columns are linear multiples of one column.

Separable Kernels

- If the columns of K are linearly dependent, then:

$$\underline{K} = \underline{v}_0 \cdot \underline{v}_1$$

$$Mat(s \times s) \qquad Mat(s \times 1) \qquad Mat(1 \times s)$$

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix} \cdot \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}$$

Separable Kernels

- If the columns of K are linearly dependent, then:

$$\underline{K} = \underline{v_0} \cdot \underline{v_1}$$

$$Mat(s \times s) \qquad Mat(s \times 1) \qquad Mat(1 \times s)$$

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

Separable Kernels

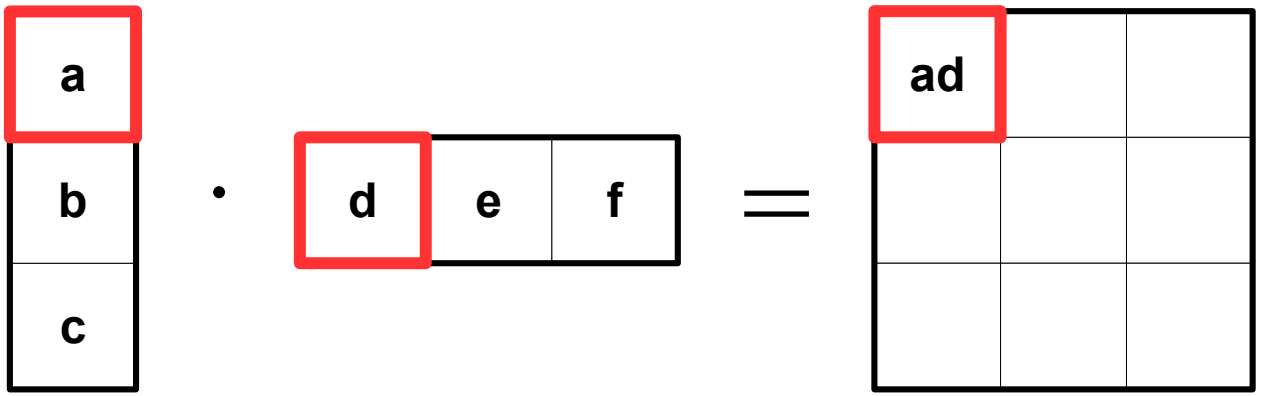
- If the columns of K are linearly dependent, then:

$$\underline{K} = \underline{v_0} \cdot \underline{v_1}$$

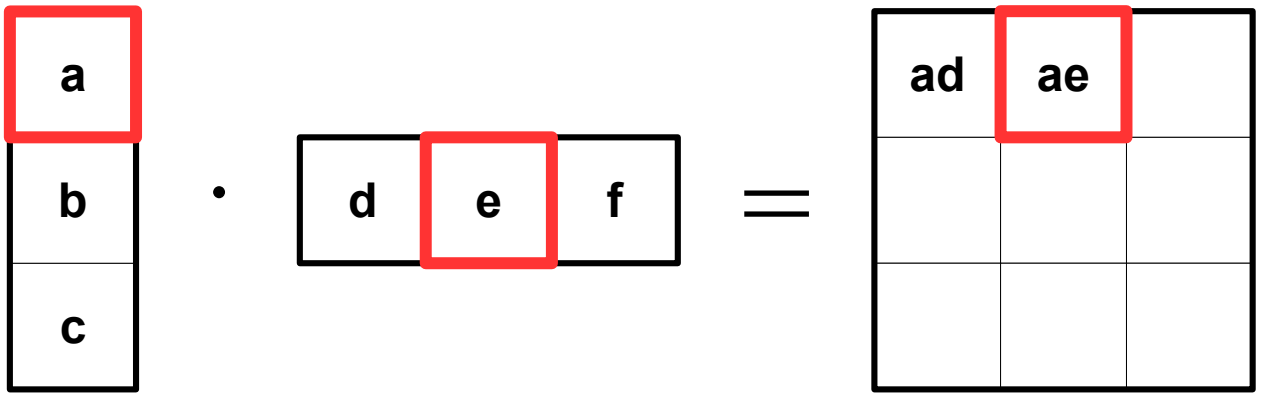
$$Mat(s \times s) \qquad Mat(s \times 1) \quad Mat(1 \times s)$$

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

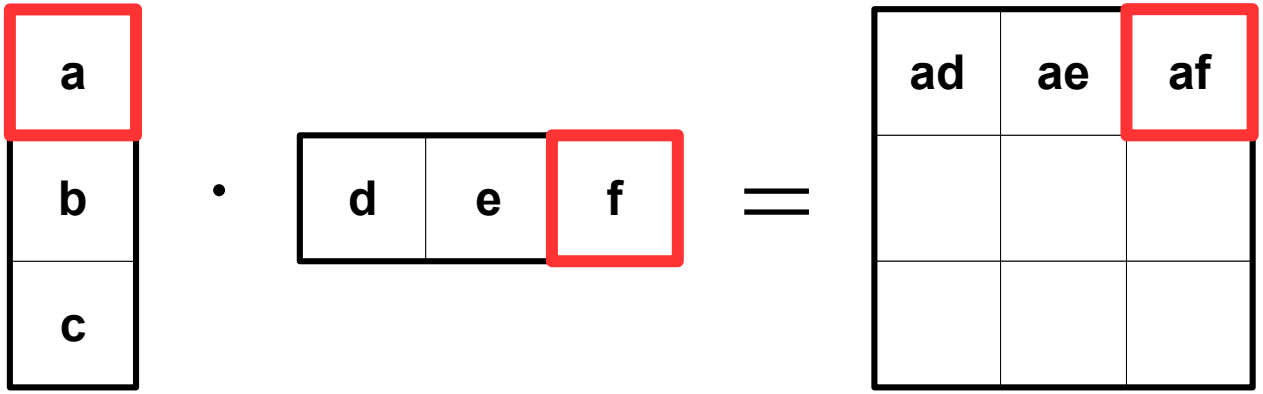
Separable Kernels



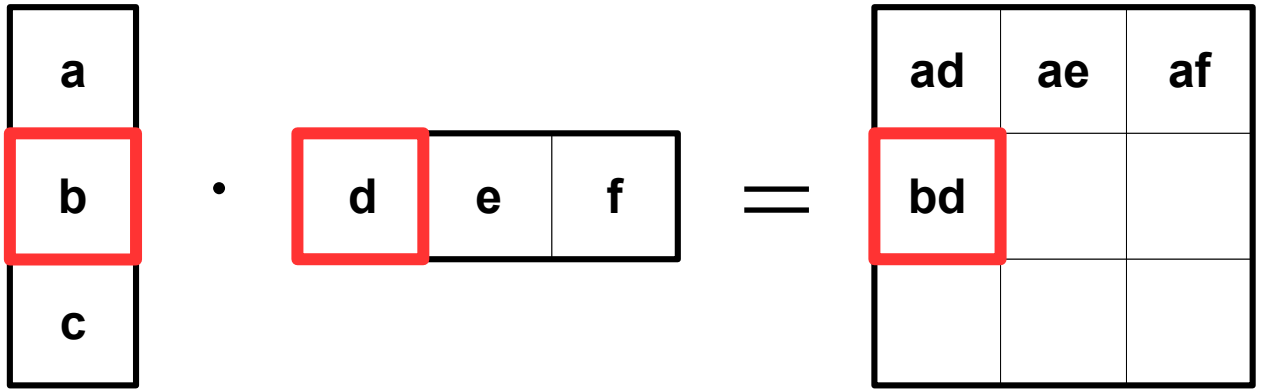
Separable Kernels



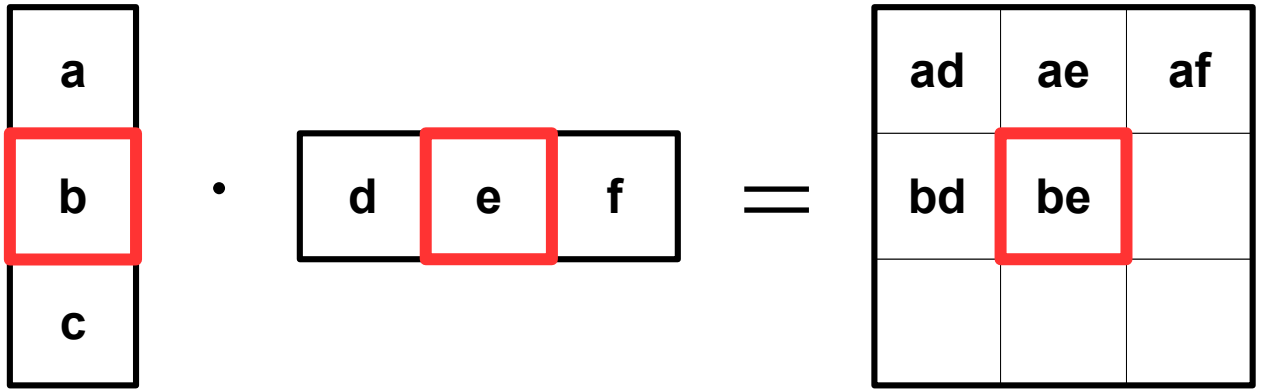
Separable Kernels



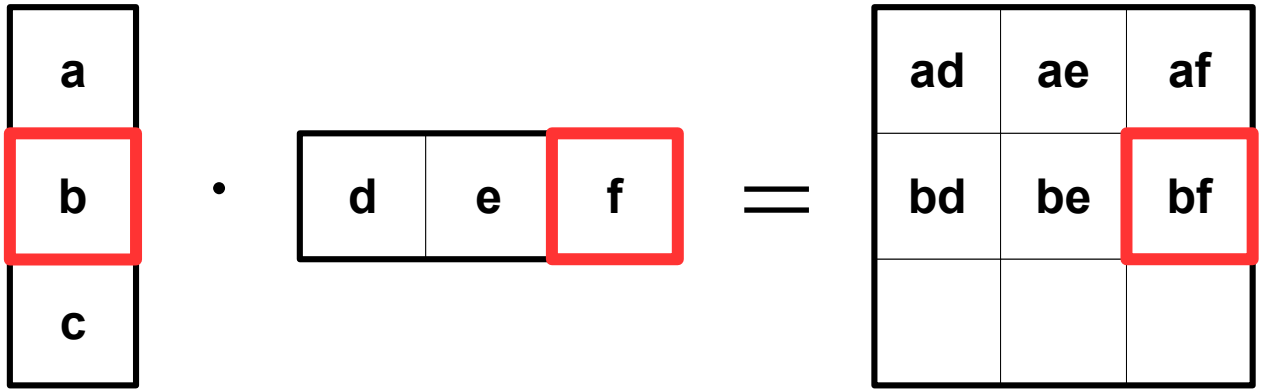
Separable Kernels



Separable Kernels



Separable Kernels



Separable Kernels

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot \begin{bmatrix} d & e & f \end{bmatrix} = \begin{bmatrix} ad & ae & af \\ bd & be & bf \\ bd & & \end{bmatrix}$$

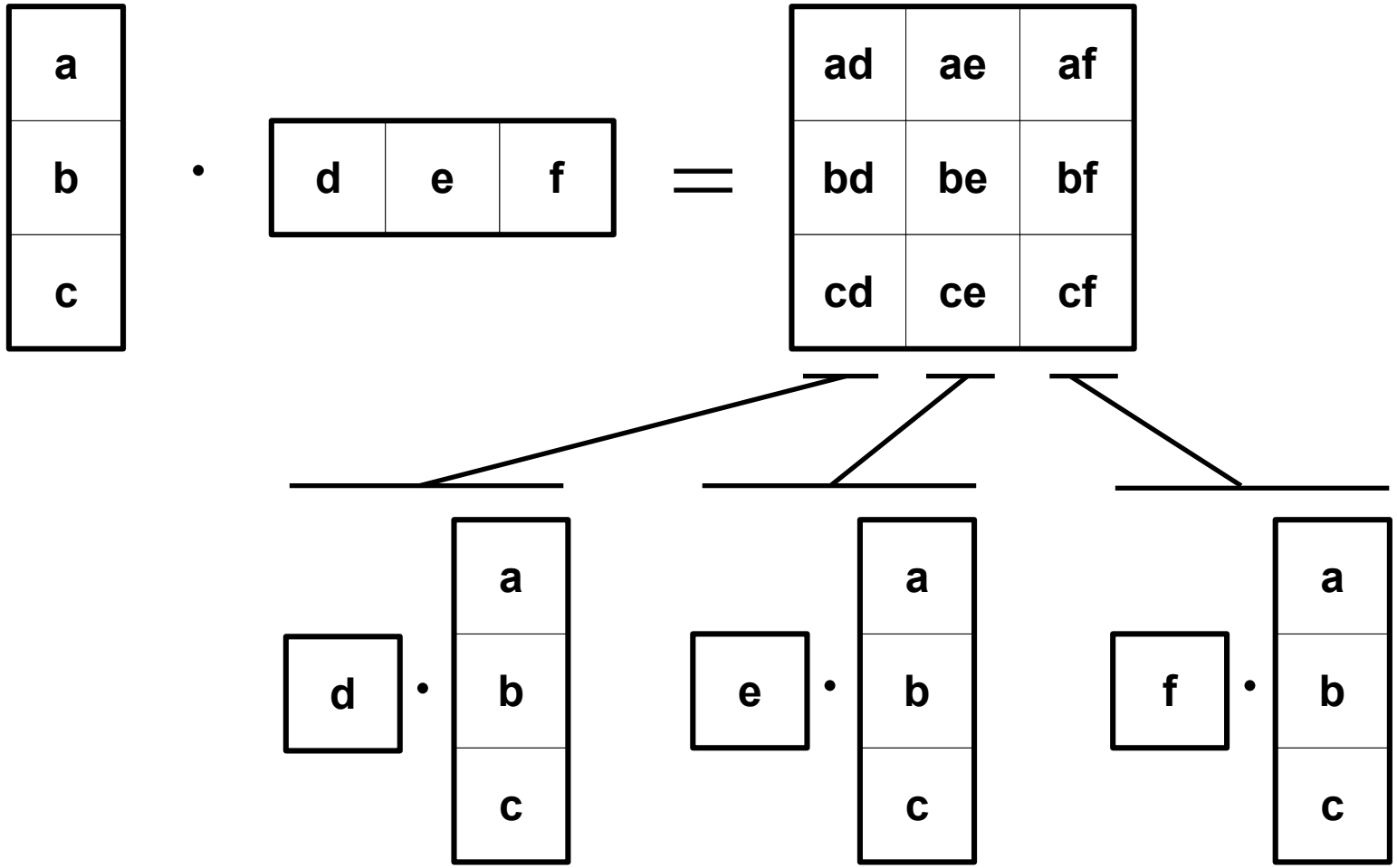
Separable Kernels

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot \begin{bmatrix} d & e & f \end{bmatrix} = \begin{bmatrix} ad & ae & af \\ bd & be & bf \\ cd & ce & \end{bmatrix}$$

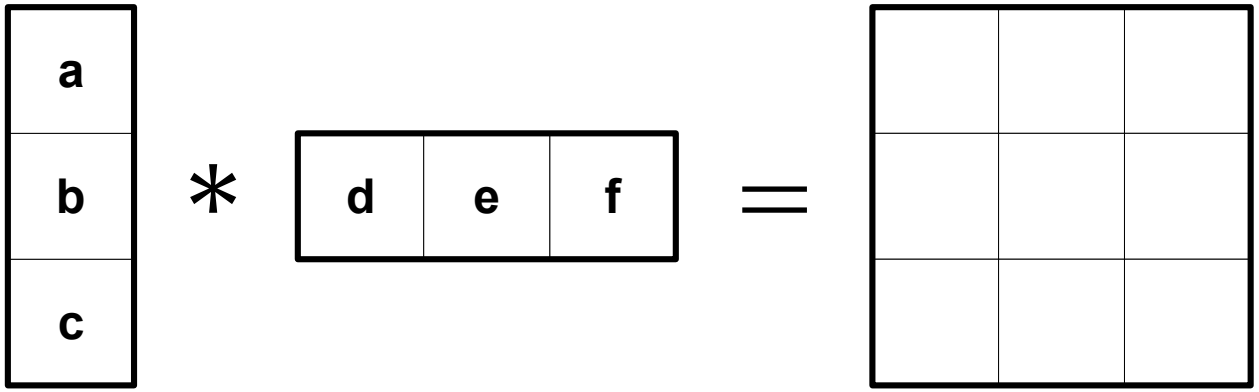
Separable Kernels

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot \begin{bmatrix} d & e & f \end{bmatrix} = \begin{bmatrix} ad & ae & af \\ bd & be & bf \\ cd & ce & cf \end{bmatrix}$$

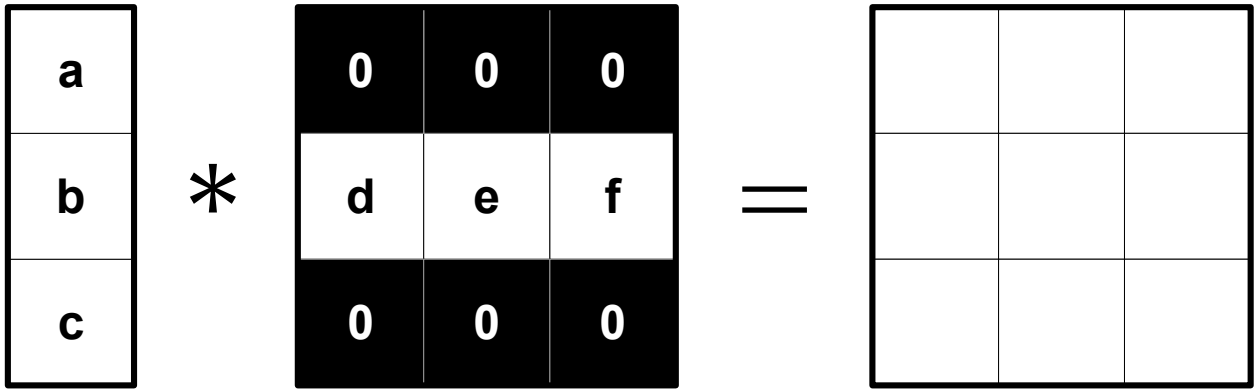
Separable Kernels



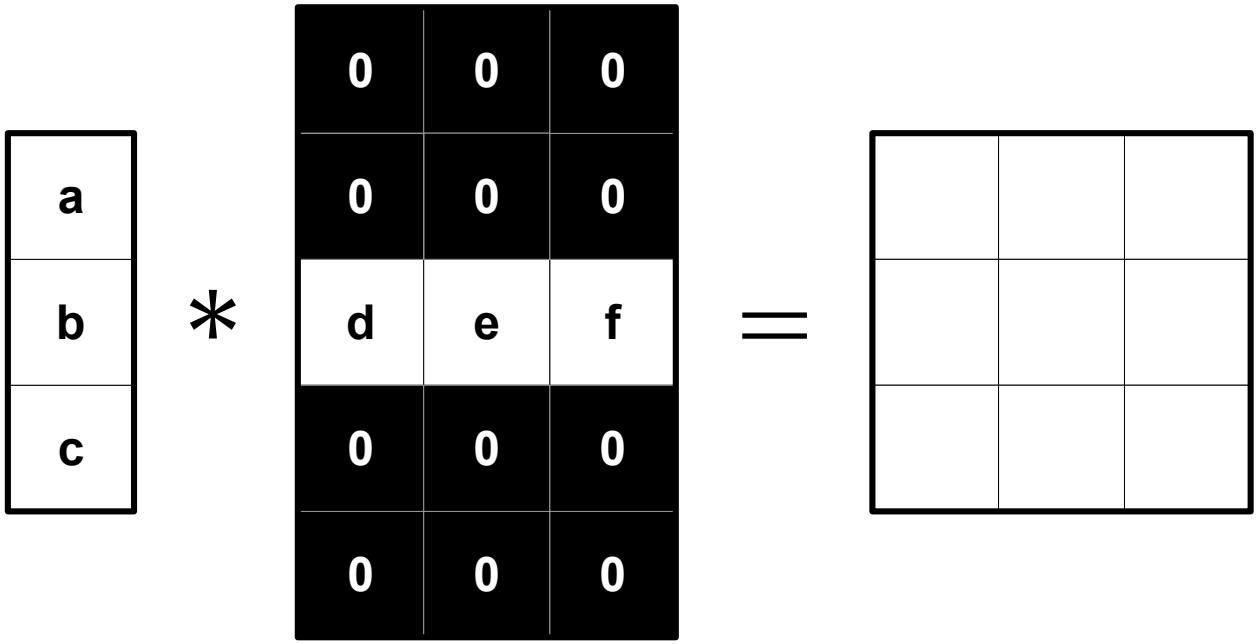
Separable Kernels



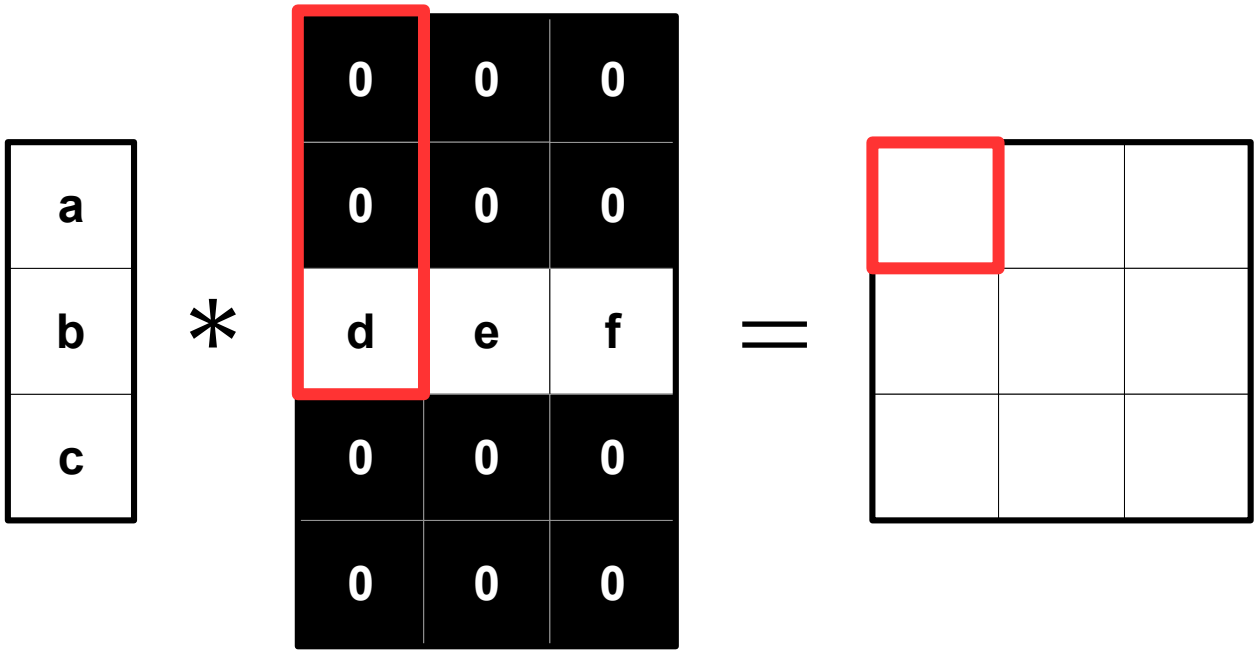
Separable Kernels



Separable Kernels

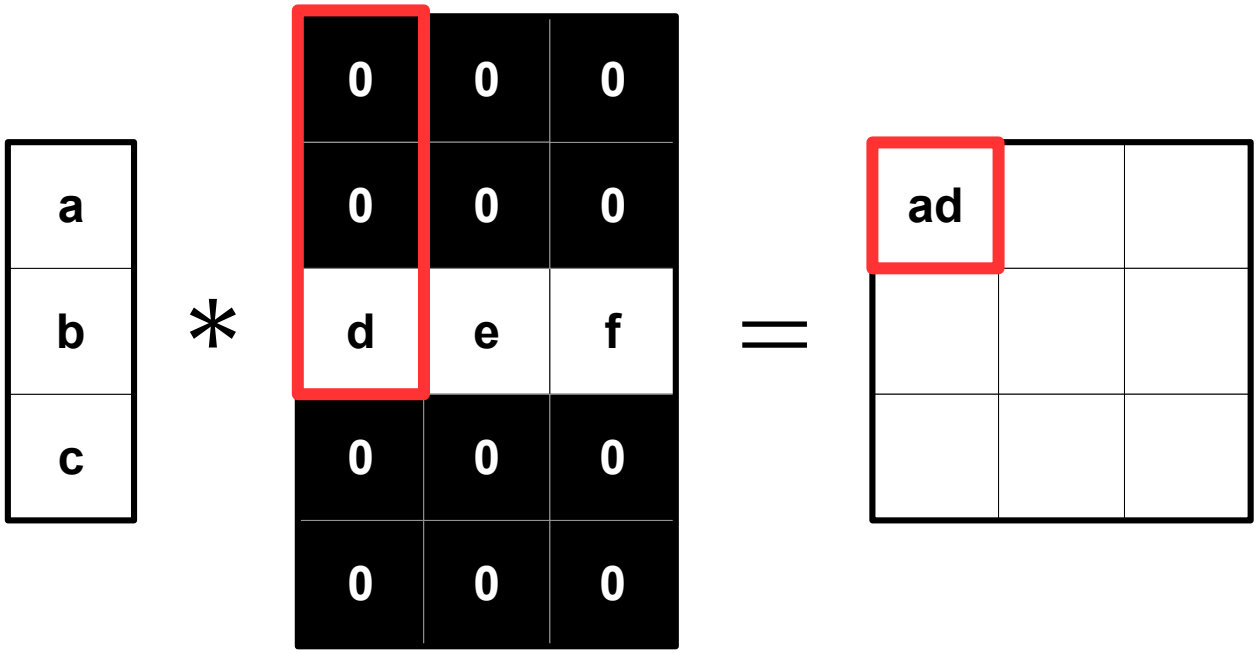


Separable Kernels



Kernel is not symmetric!
So we must use values opposite of the current pixel.

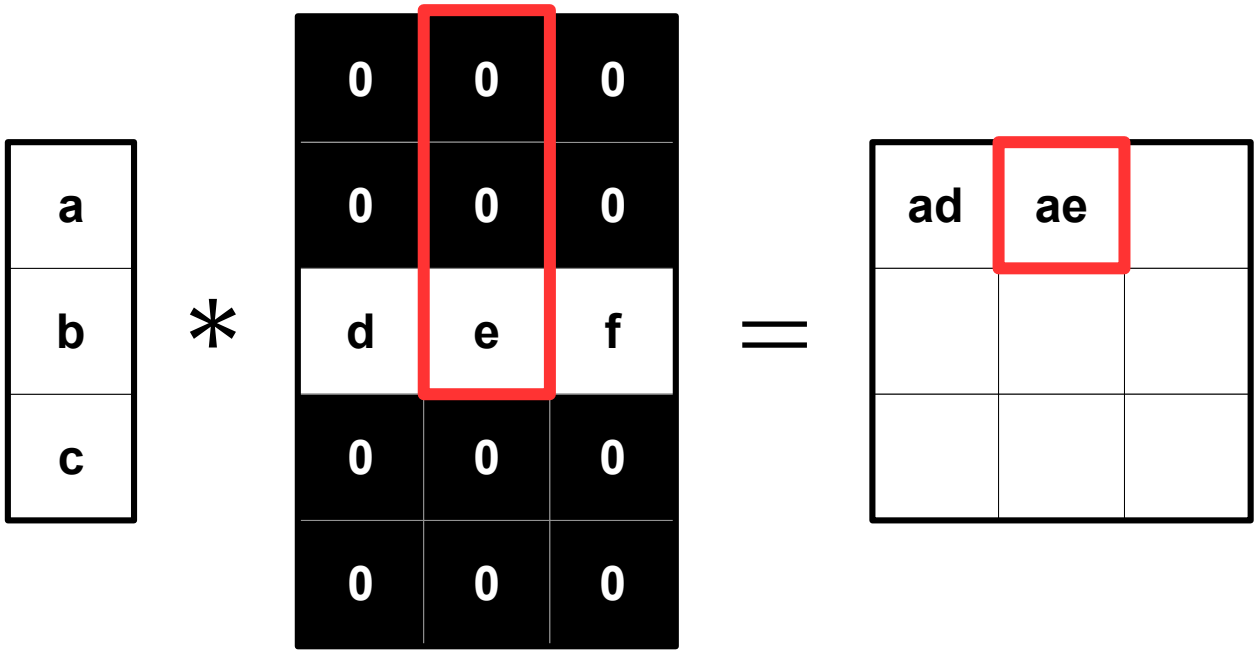
Separable Kernels



$$ad + 0\,b + 0\,c = ad$$

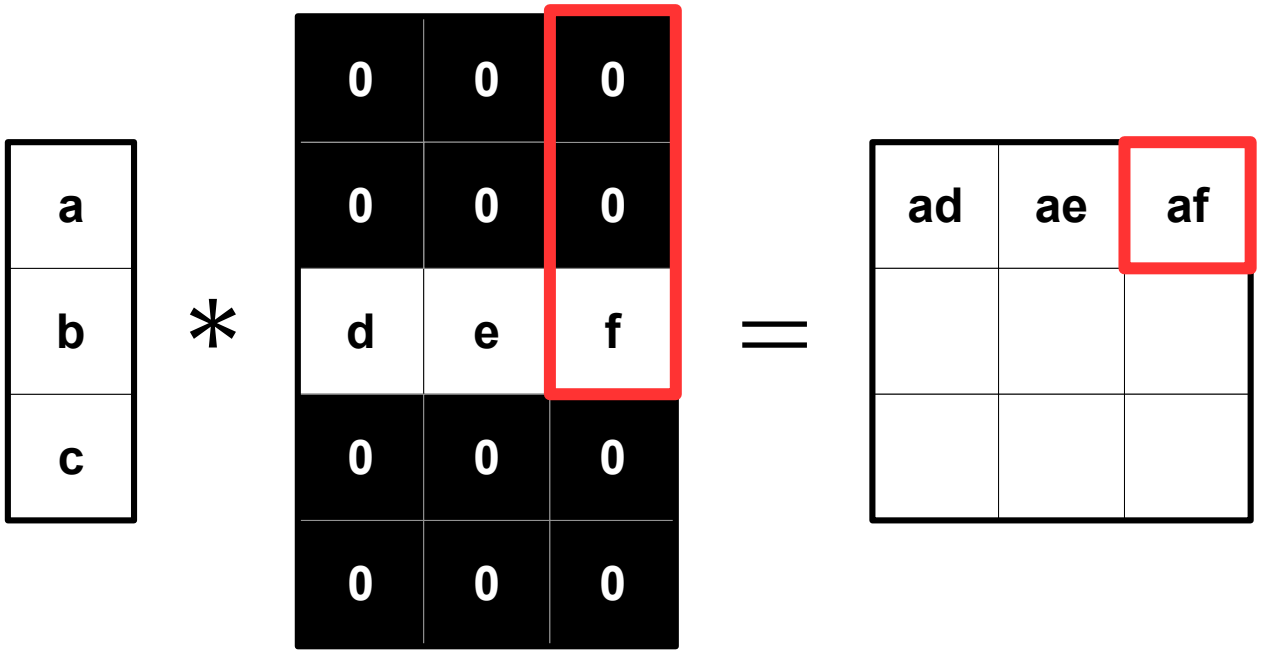
Kernel is not symmetric!
So we must use values opposite of the current pixel.

Separable Kernels



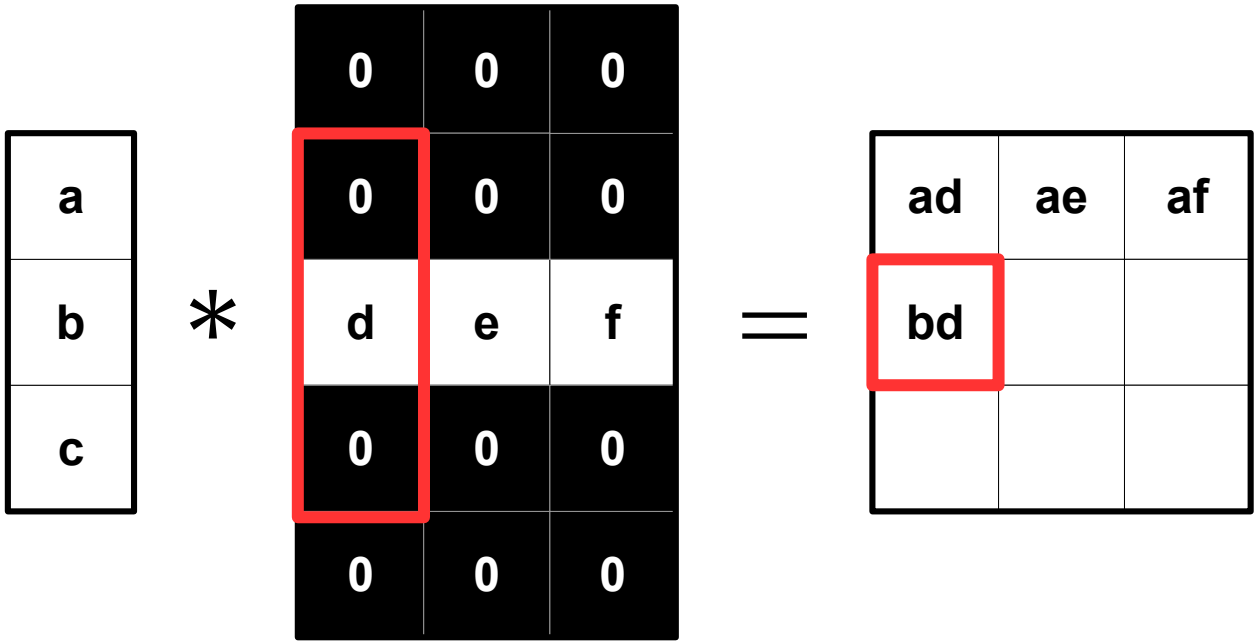
$$ae + 0\,b + 0\,c = ae$$

Separable Kernels



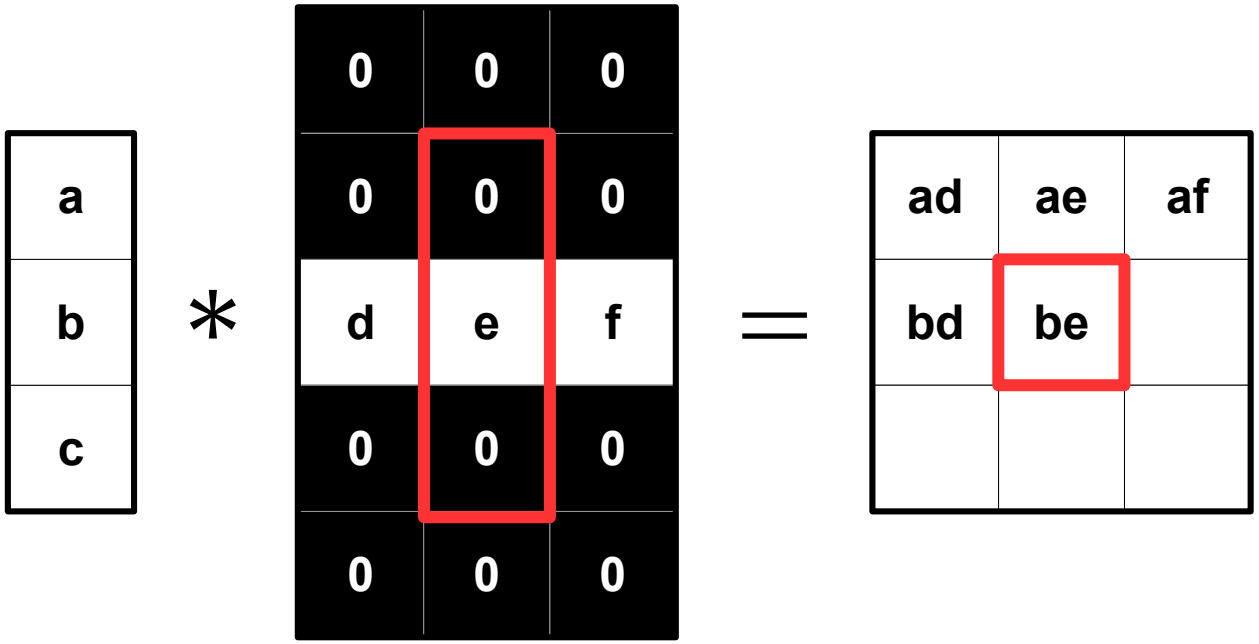
$$af + 0\,b + 0\,c = af$$

Separable Kernels



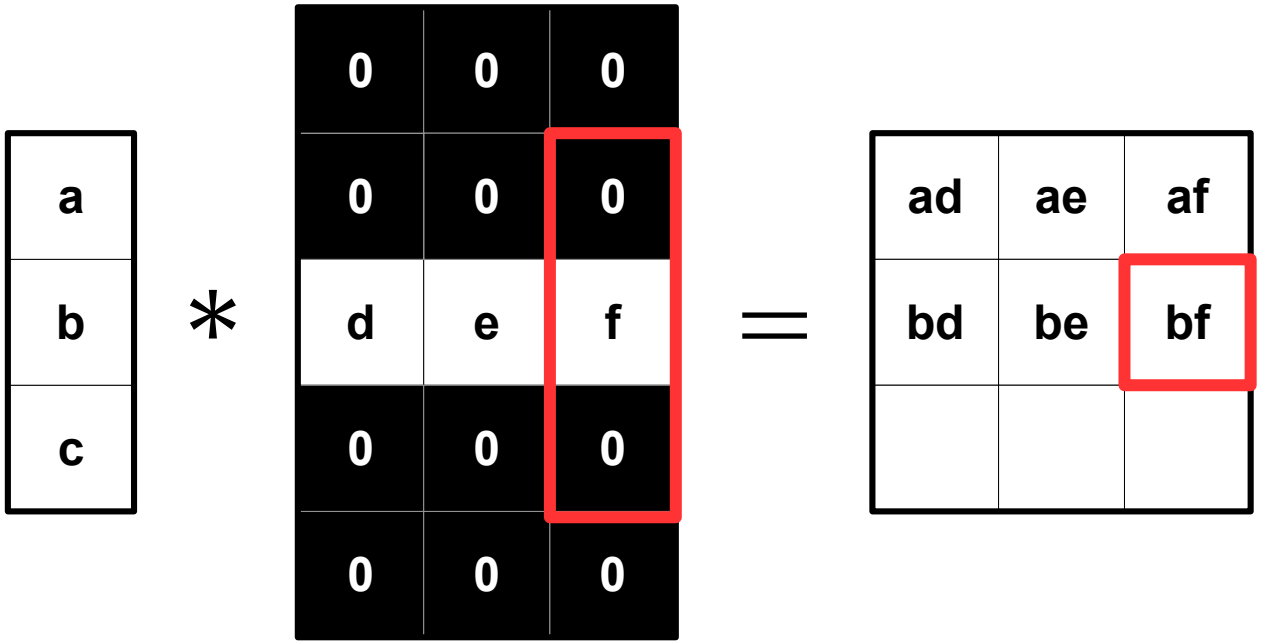
$$0\,\boldsymbol{a} + \boldsymbol{bd} + 0\,\boldsymbol{c} = \boldsymbol{bd}$$

Separable Kernels



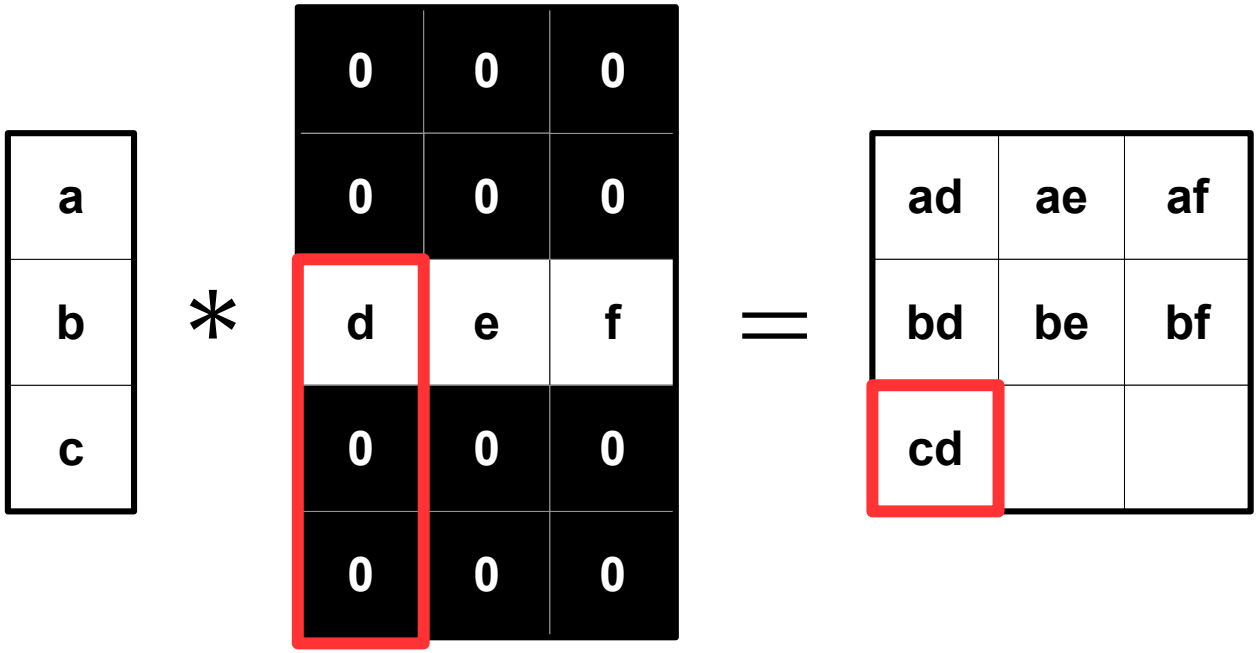
$$0\,\boldsymbol{a} + \boldsymbol{be} + 0\,\boldsymbol{c} = \boldsymbol{be}$$

Separable Kernels



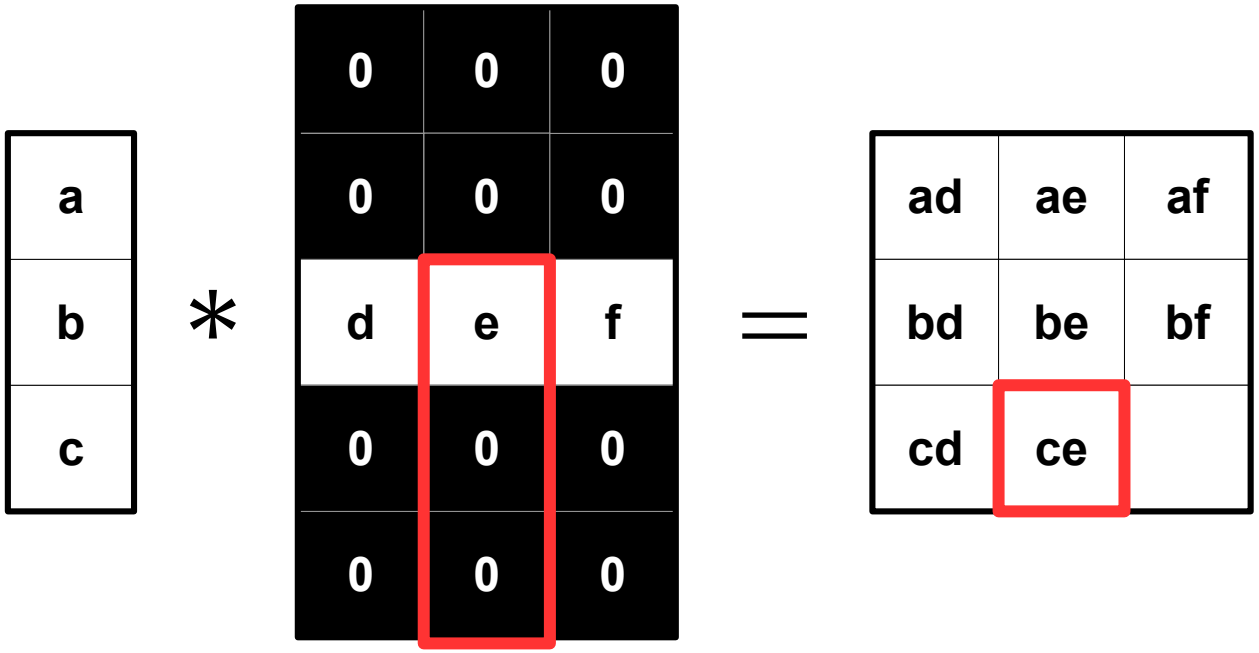
$$0\,\boldsymbol{a} + \boldsymbol{bf} + 0\,\boldsymbol{c} = \boldsymbol{bf}$$

Separable Kernels



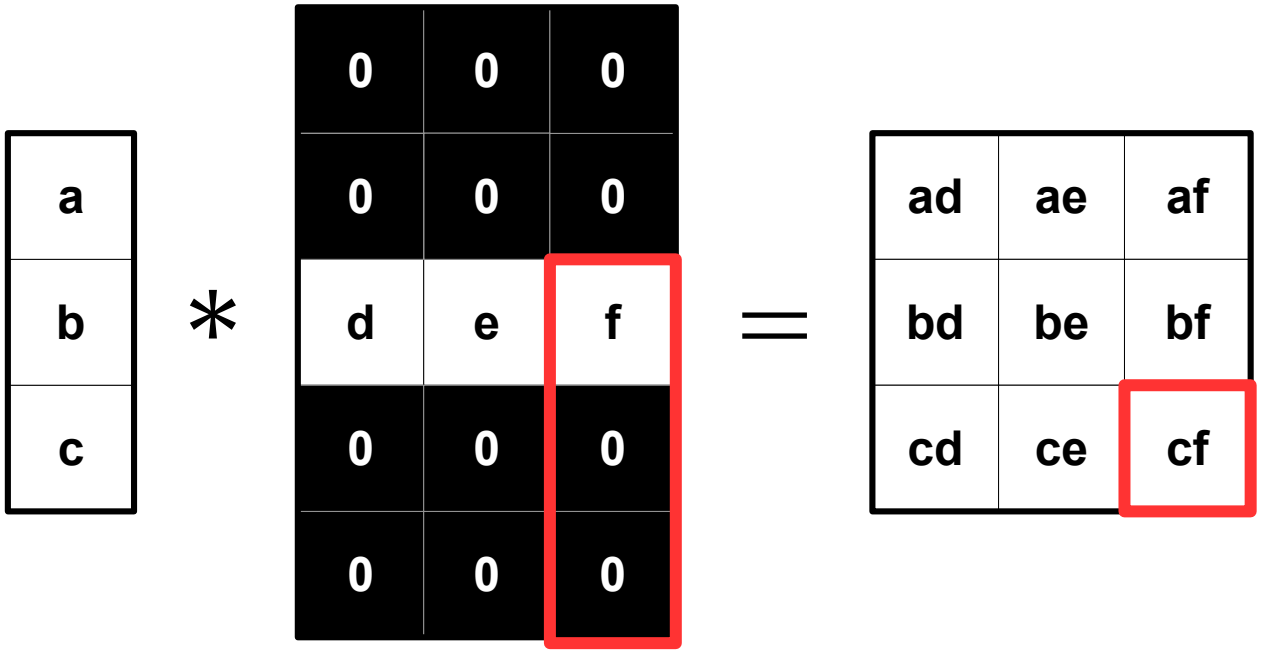
$$0\,\boldsymbol{a} + 0\,\boldsymbol{b} + 0\,\boldsymbol{cd} = \boldsymbol{cd}$$

Separable Kernels



$$0\,\boldsymbol{a}+0\,\boldsymbol{b}+0\,\boldsymbol{ce}=\boldsymbol{ce}$$

Separable Kernels



$$0\,\boldsymbol{a} + 0\,\boldsymbol{b} + 0\,\boldsymbol{cf} = \boldsymbol{cf}$$

Separable Kernels

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ d & e & f \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} ad & ae & af \\ bd & be & bf \\ cd & ce & cf \end{bmatrix}$$

Convolution

Multiplication

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot \begin{bmatrix} d & e & f \end{bmatrix} = \begin{bmatrix} ad & ae & af \\ bd & be & bf \\ cd & ce & cf \end{bmatrix}$$

Separable Kernels

- Turns out that in our current case:

$$K = \underbrace{v_0 \cdot v_1}_{\text{Matrix multiplication}} = \underbrace{v_0 * v_1}_{\text{Convolution}}$$

Separable Kernels

- Turns out that in our current case:

$$K = v_0 \cdot v_1 = v_0 * v_1$$

$$K * P = \left(v_0 \cdot v_1\right) * P = \left(v_0 * v_1\right) * P$$

Separable Kernels

- Turns out that in our current case:

$$K = v_0 \cdot v_1 = v_0 * v_1$$

$$K * P = (v_0 \cdot v_1) * P = (v_0 * v_1) * P$$

- Convolution is commutative*:

$$K * P = (v_0 \cdot v_1) * P = v_0 * (v_1 * P)$$

*Proof is out of the scope here.

Separable Kernels

- If our kernel is **separable**:

$$K = v_0 \cdot v_1$$

Separable Kernels

- If our kernel is **separable**:

$$K = v_0 \cdot v_1$$

- Then a 1×2D convolution is the same as 2×1D:

$$K * P = v_0 * \left(v_1 * P \right)$$

Separable Kernels

- If our kernel is **separable**:

$$K = v_0 \cdot v_1$$

- Then a 1×2D convolution is the same as 2×1D:

$$\underbrace{K * P}_{O(s^2)} = \underbrace{v_0 * (v_1 * P)}_{O(s)+O(s)=O(s)}$$

Ignoring the complexity of parsing all the image pixels.

Separable Kernels

- The Gaussian blur kernel is **separable**:

2D Gaussian function

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

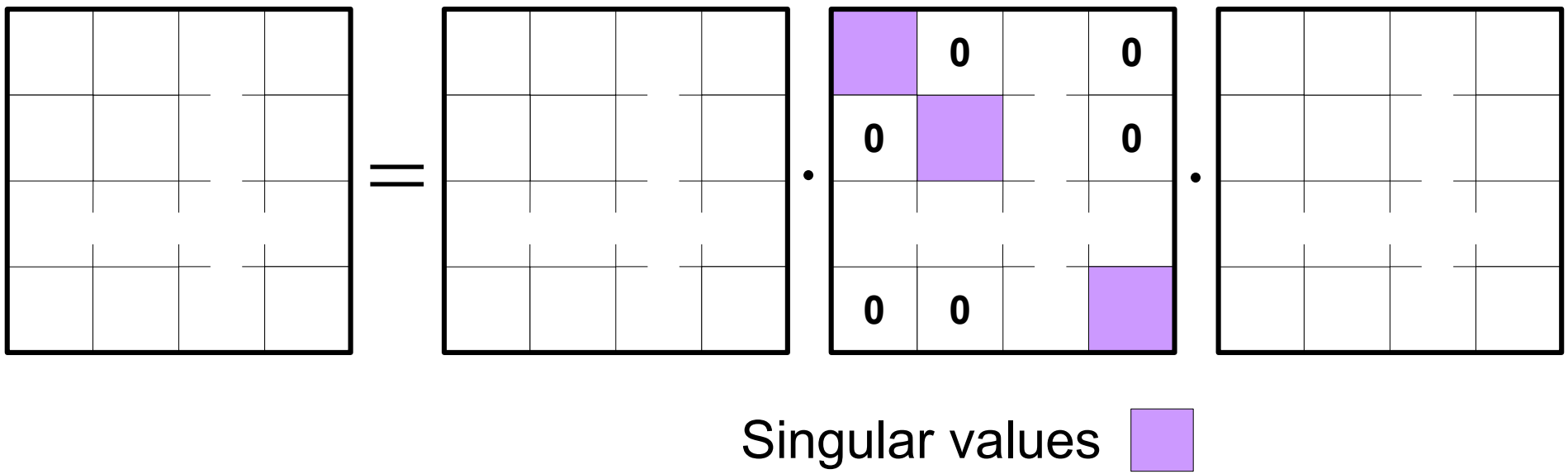
$$=\frac{1}{\sqrt{2\pi\sigma^2}}\cdot e^{-\frac{x^2}{2\sigma^2}}\cdot\frac{1}{\sqrt{2\pi\sigma^2}}\cdot e^{-\frac{y^2}{2\sigma^2}}$$

1D Gaussian functions

Singular Value Decomposition

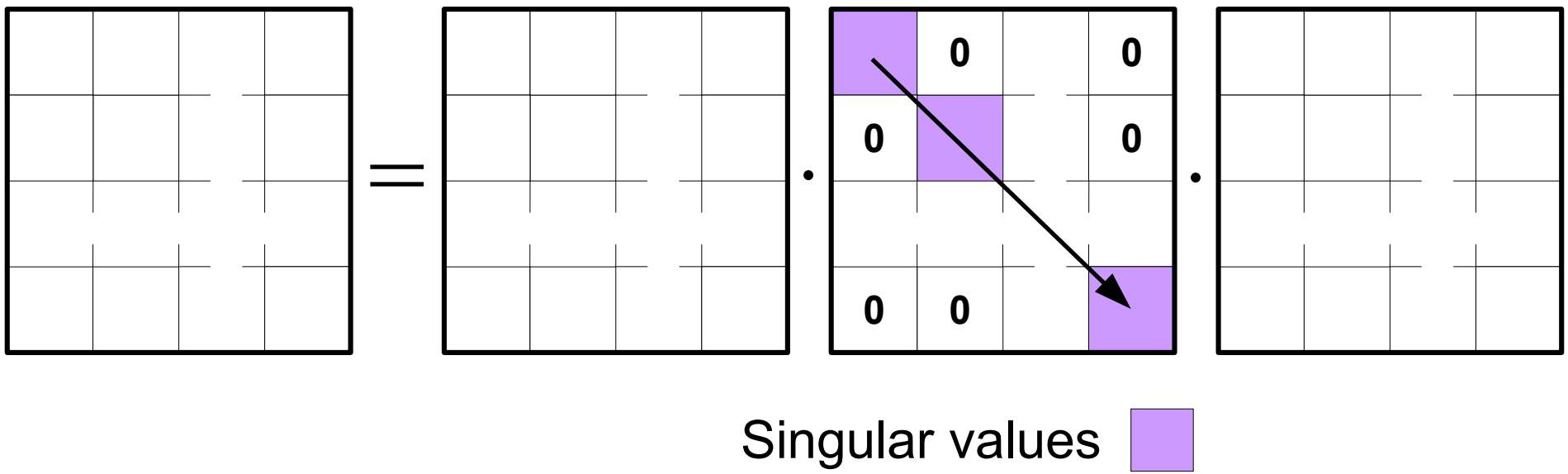
- Decomposes matrix K into 3 other matrices.

$$K = U \Sigma V$$



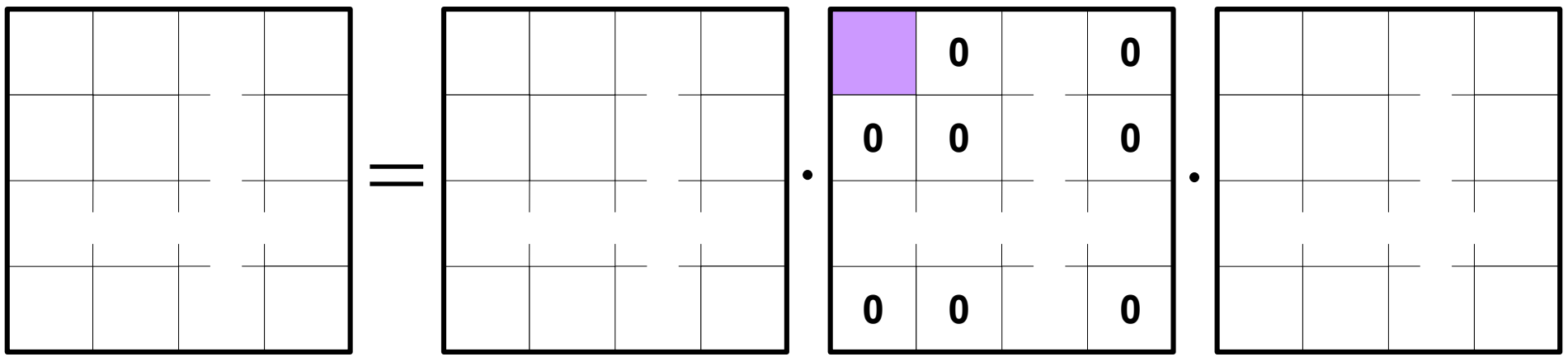
Singular Value Decomposition

- Decomposes matrix K into 3 other matrices.
- Singular values are ordered.

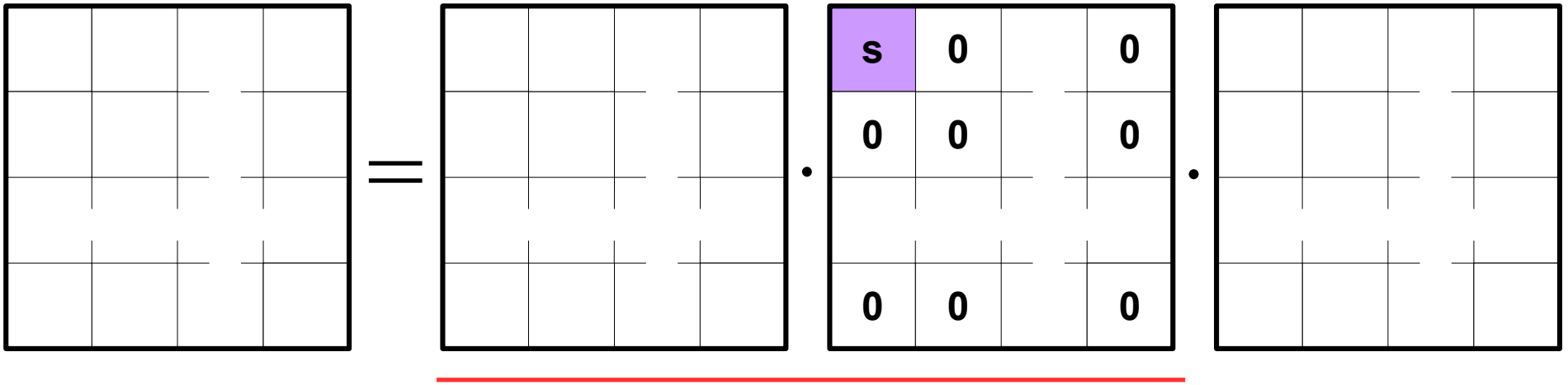


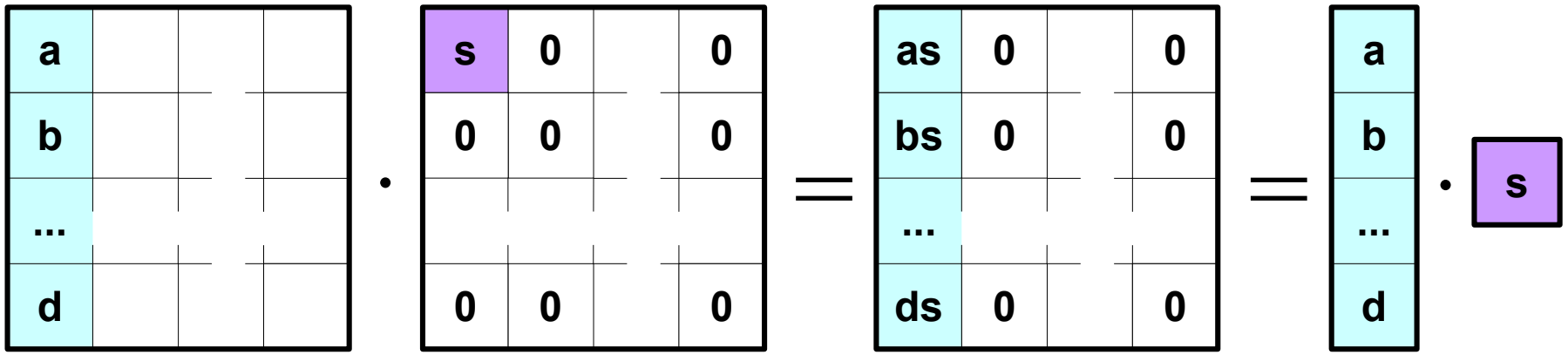
Singular Value Decomposition

- The number of singular values = rank of K.
- If K is separable, rank of K = 1.

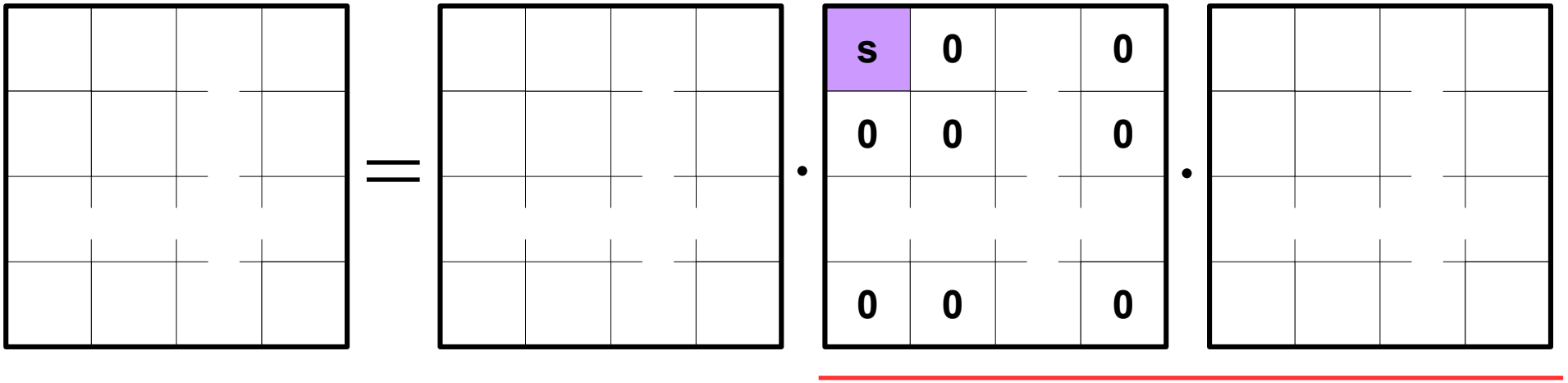


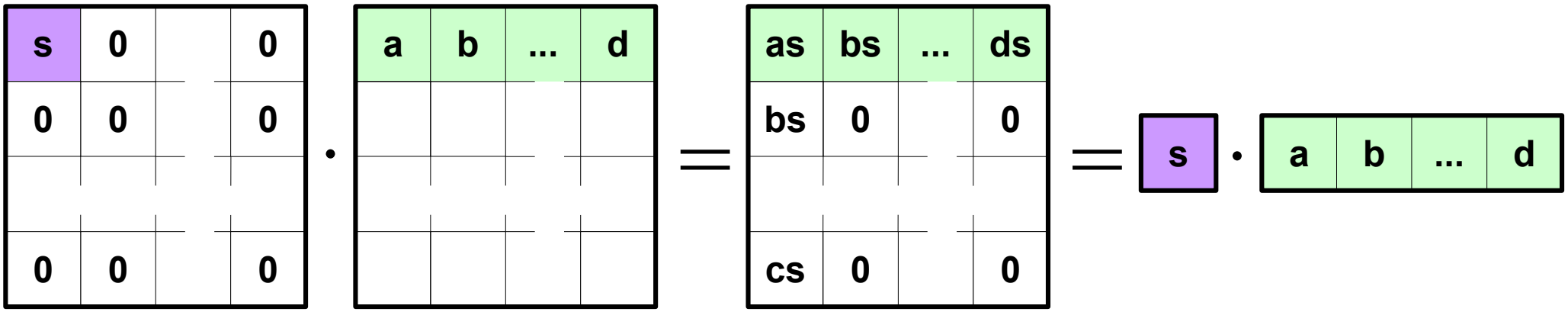
Singular Value Decomposition



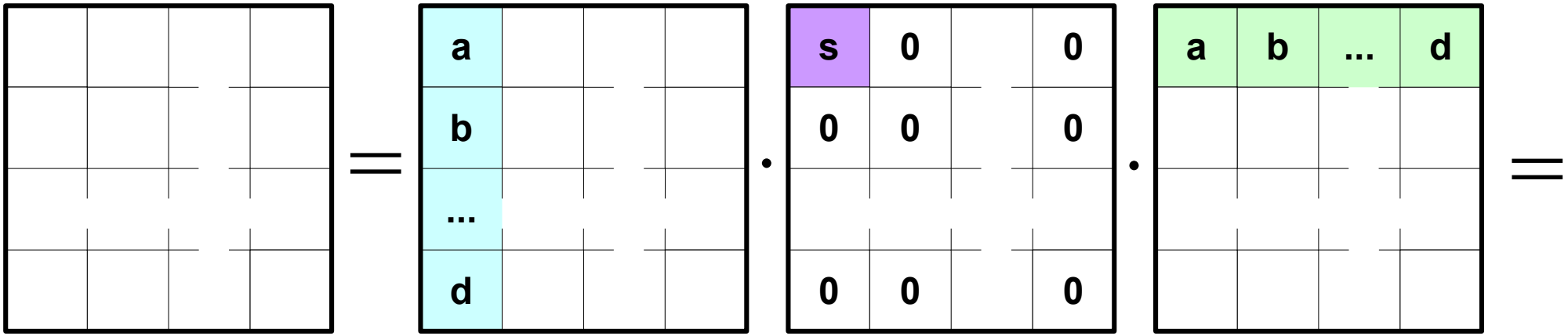


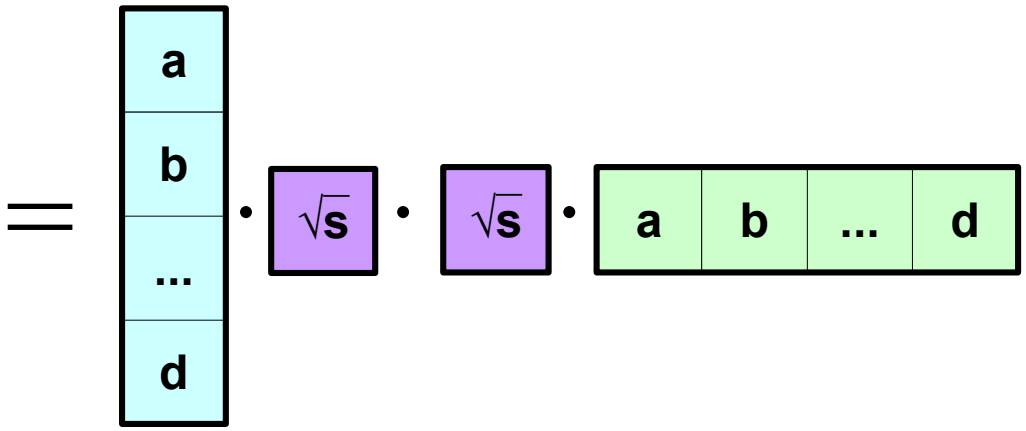
Singular Value Decomposition



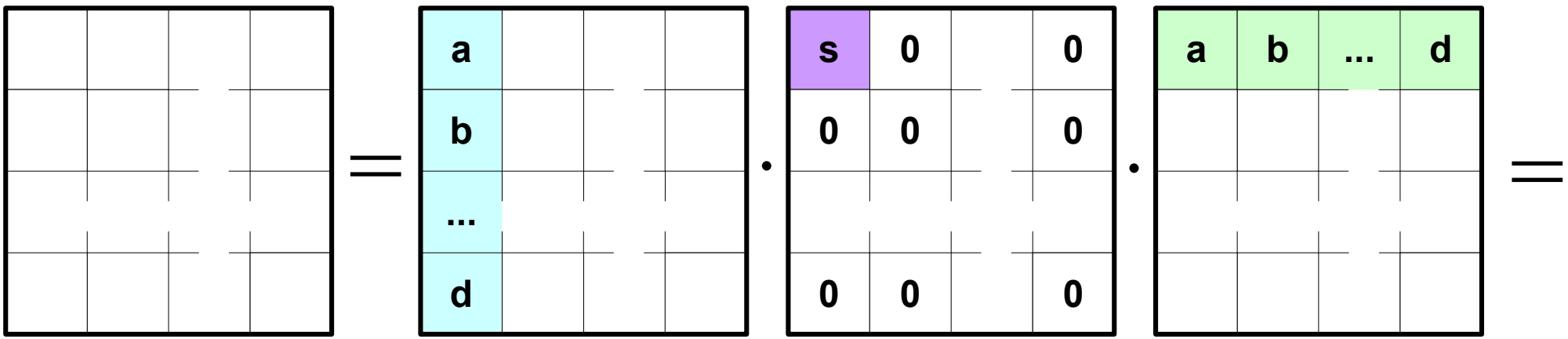


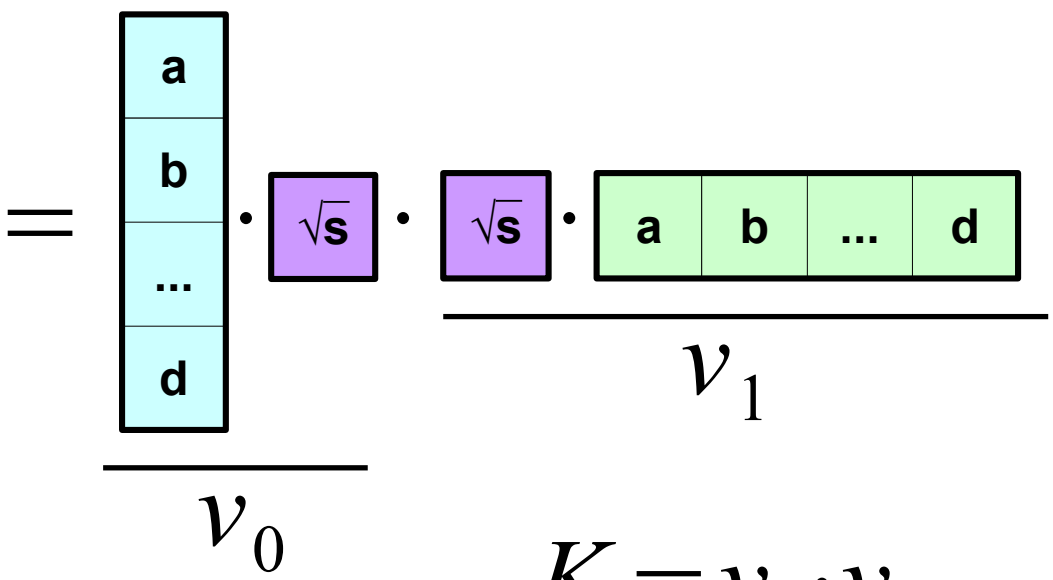
Singular Value Decomposition





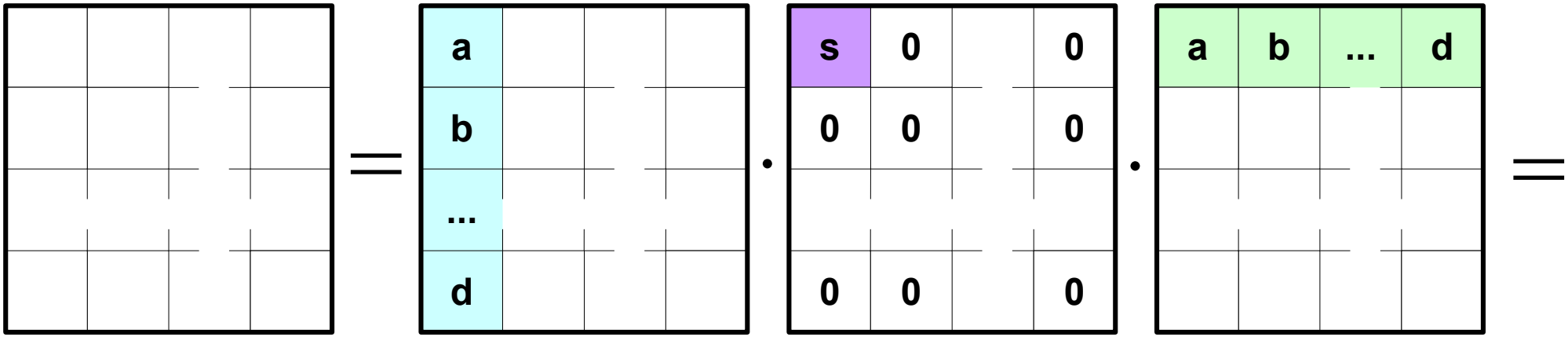
Singular Value Decomposition

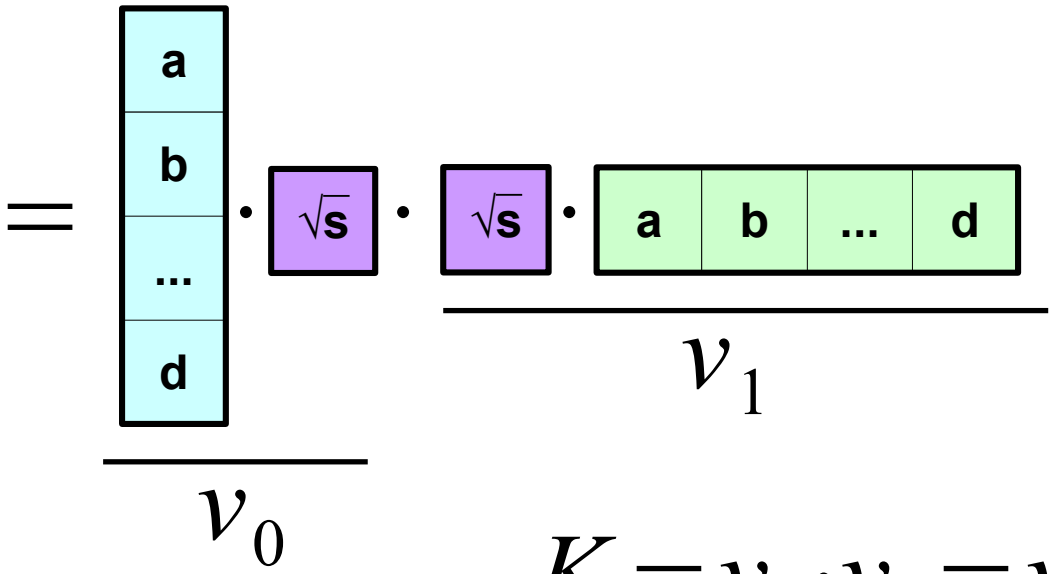




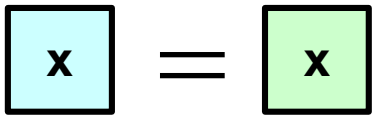
$$K = v_0 \cdot v_1$$

Singular Value Decomposition



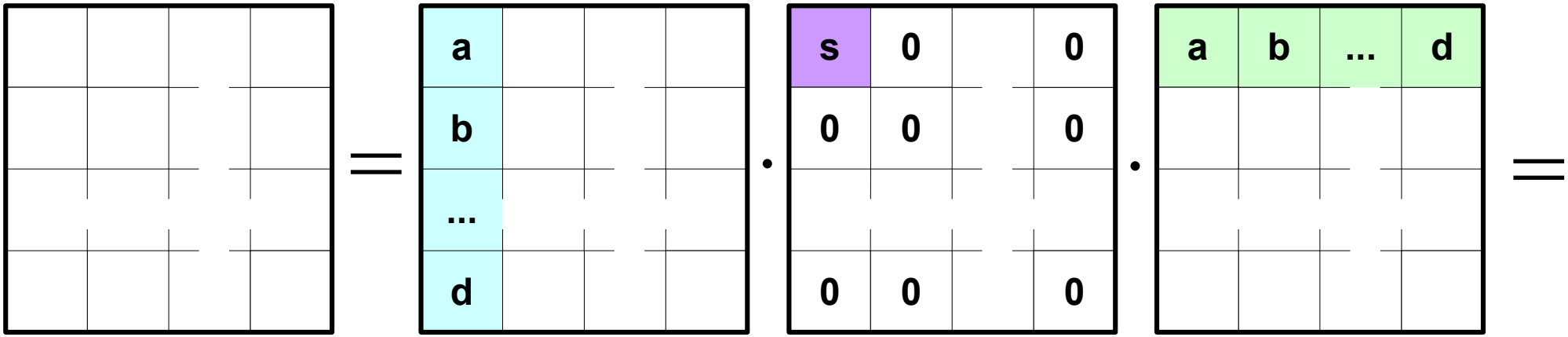


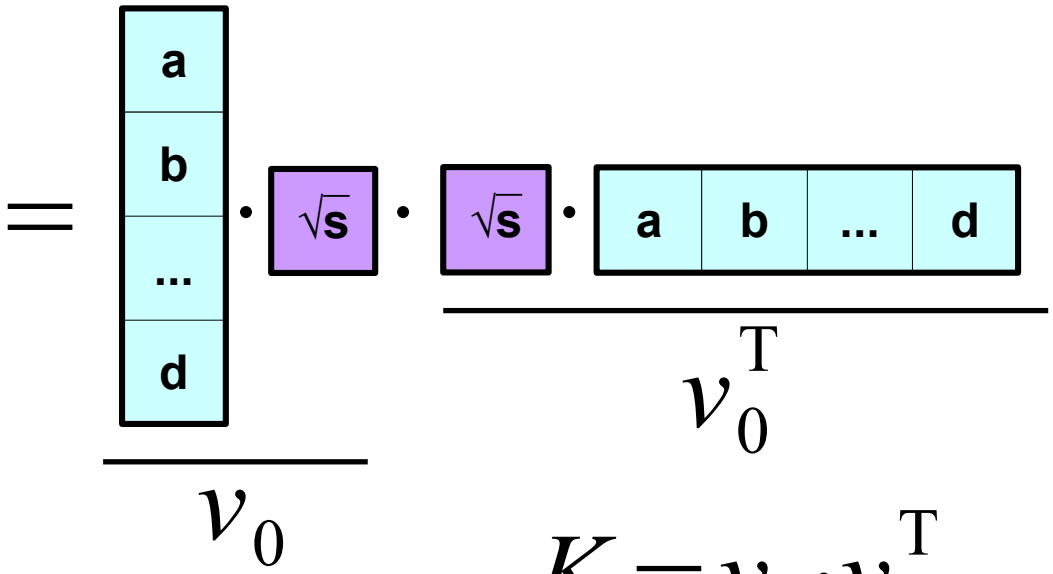
When the kernel is **circularly symmetric**:



$$K = v_0 \cdot v_1 = v_0 \cdot v_0^{\mathrm{T}}$$

Singular Value Decomposition





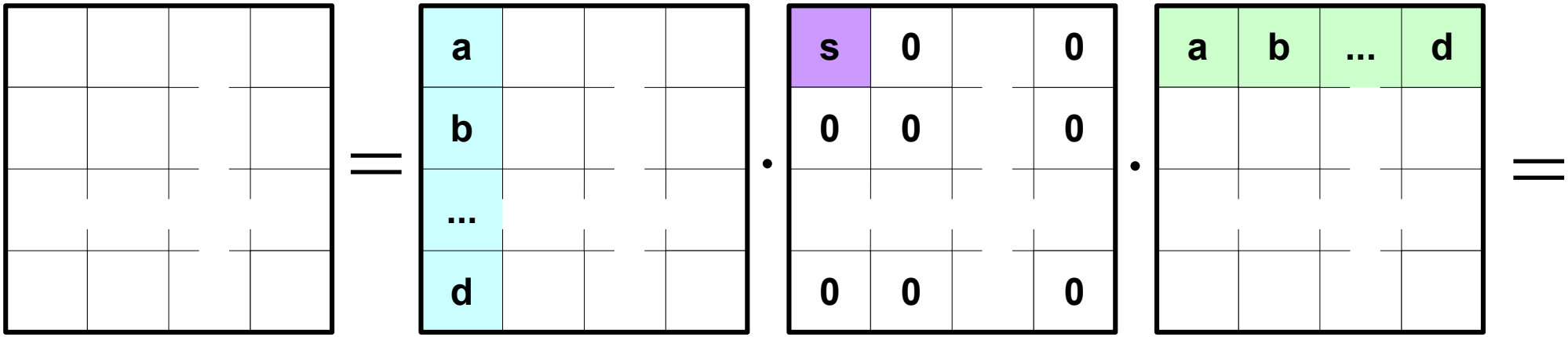
$$K = v_0 \cdot v_0^{\mathrm{T}}$$

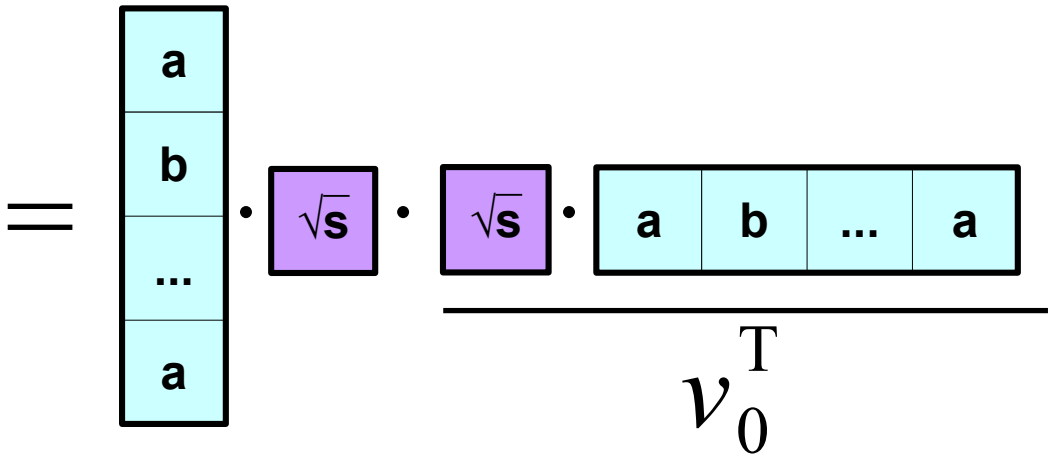
When the kernel is **circularly symmetric**:

$$\boxed{x} = \boxed{x}$$

$$\boxed{v_0(+\Delta x)} = \boxed{v_0(-\Delta x)}$$

Singular Value Decomposition





$$v_0 \qquad v_0^{\mathrm{T}}$$

$$K = v_0 \cdot v_0^{\mathrm{T}}$$

When the kernel is **circularly symmetric**:

$$\boxed{x} = \boxed{x}$$

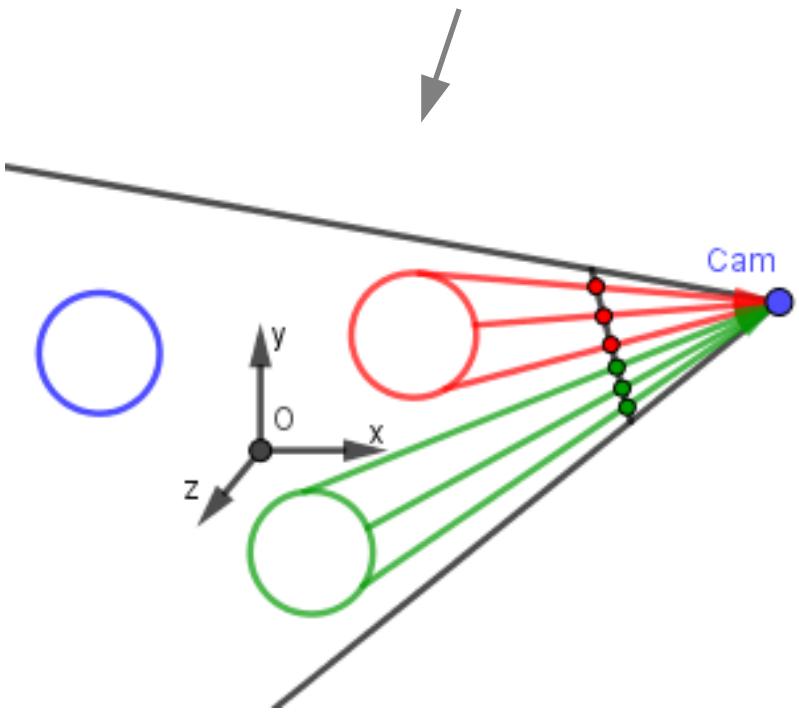
$$\boxed{v_0(+\Delta x)} = \boxed{v_0(-\Delta x)}$$

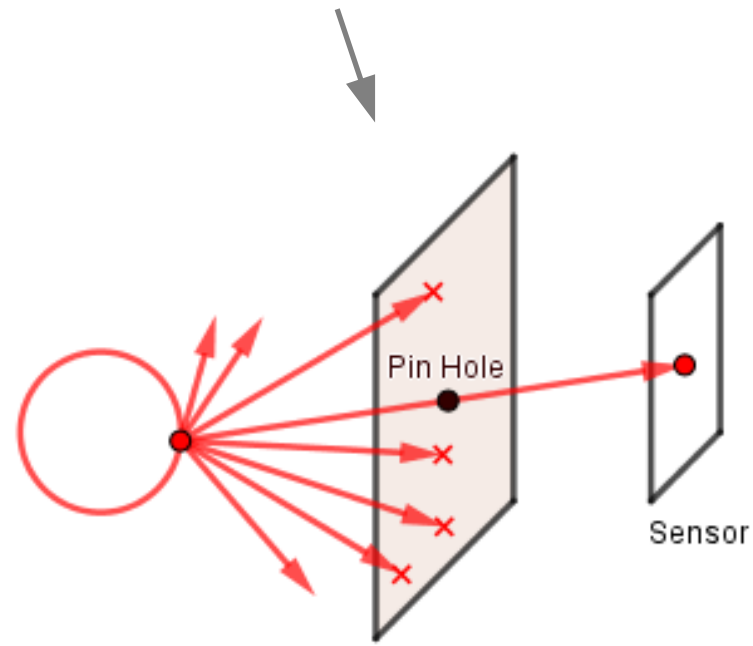
$$K = K^{\mathrm{T}}$$

Depth of Field

Depth of Field

- Standard rendering models a *pinhole camera*.

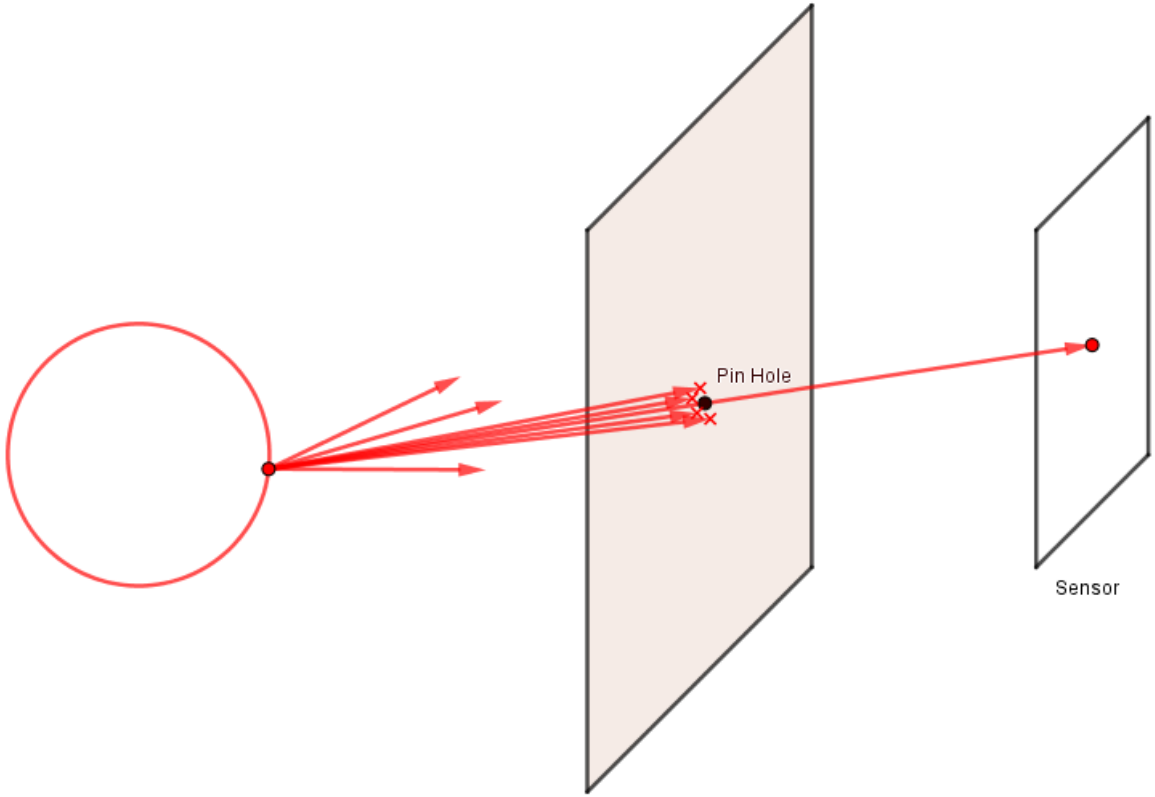




- Every surface point has 1 light ray in the projection.

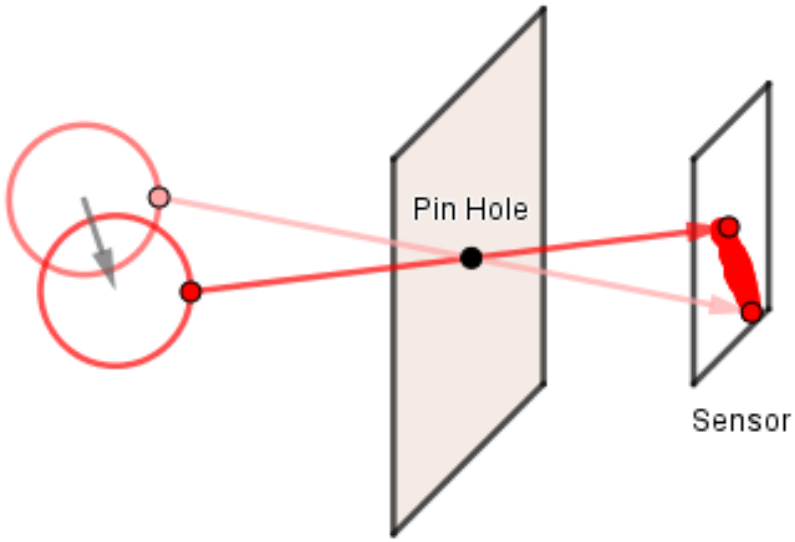
Depth of Field

- Problems with a pinhole camera:
  - Smaller hole means less light passes through
  - Less light means a darker image



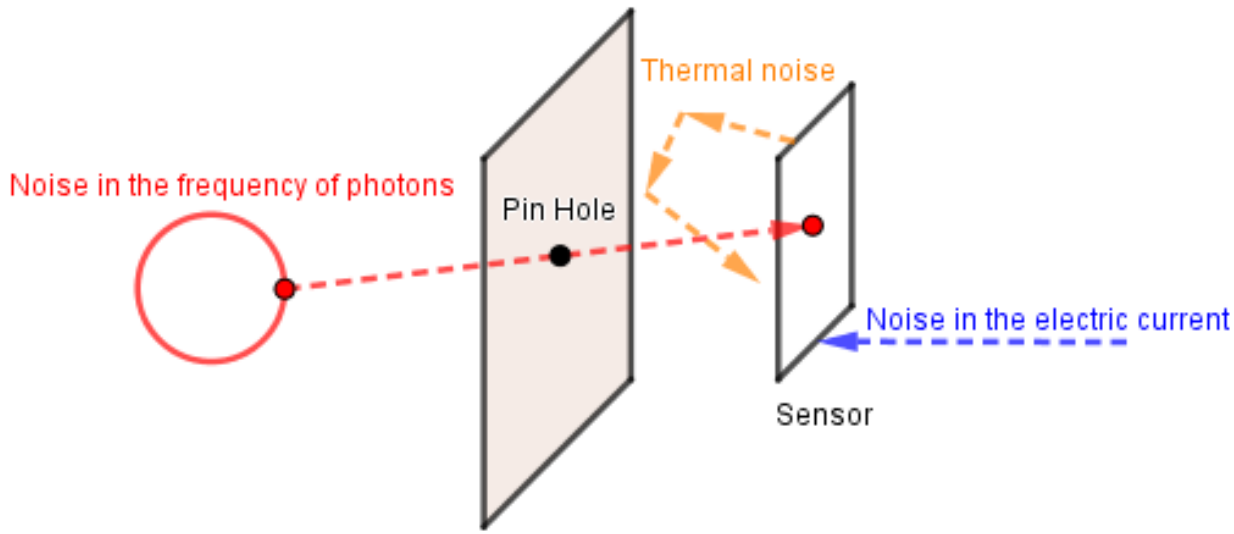
Depth of Field

- Problems with a pinhole camera:
  - Smaller hole means less light passes through
  - Less light means a darker image
  - Longer exposure could compensate
  - Longer exposure creates motion blur



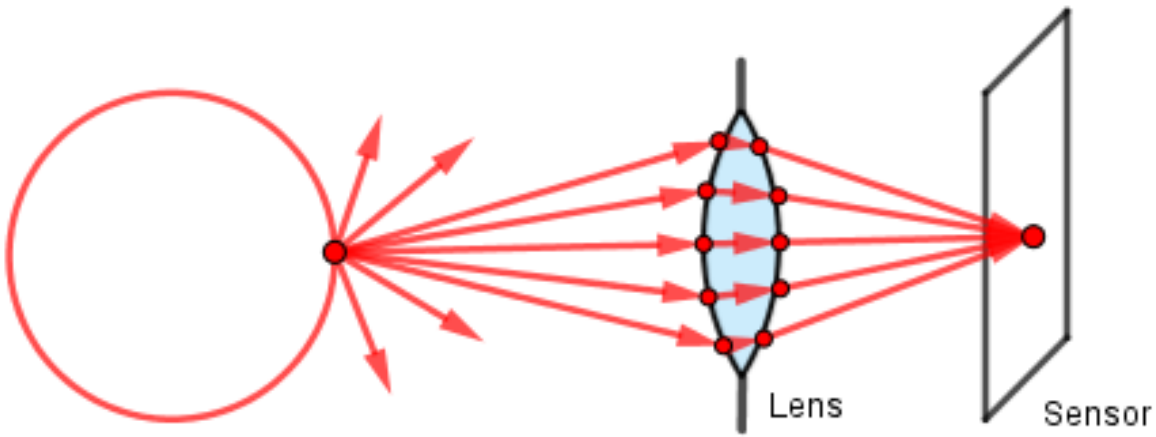
Depth of Field

- Problems wi
  - Smaller h
  - Less light
  - Longer ex
  - Longer ex
  - More sensitive sensor could compensate
  - More sensitive sensor means more noise



Depth of Field

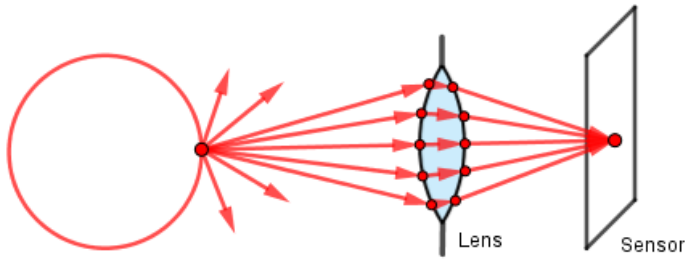
- Cameras with lenses*



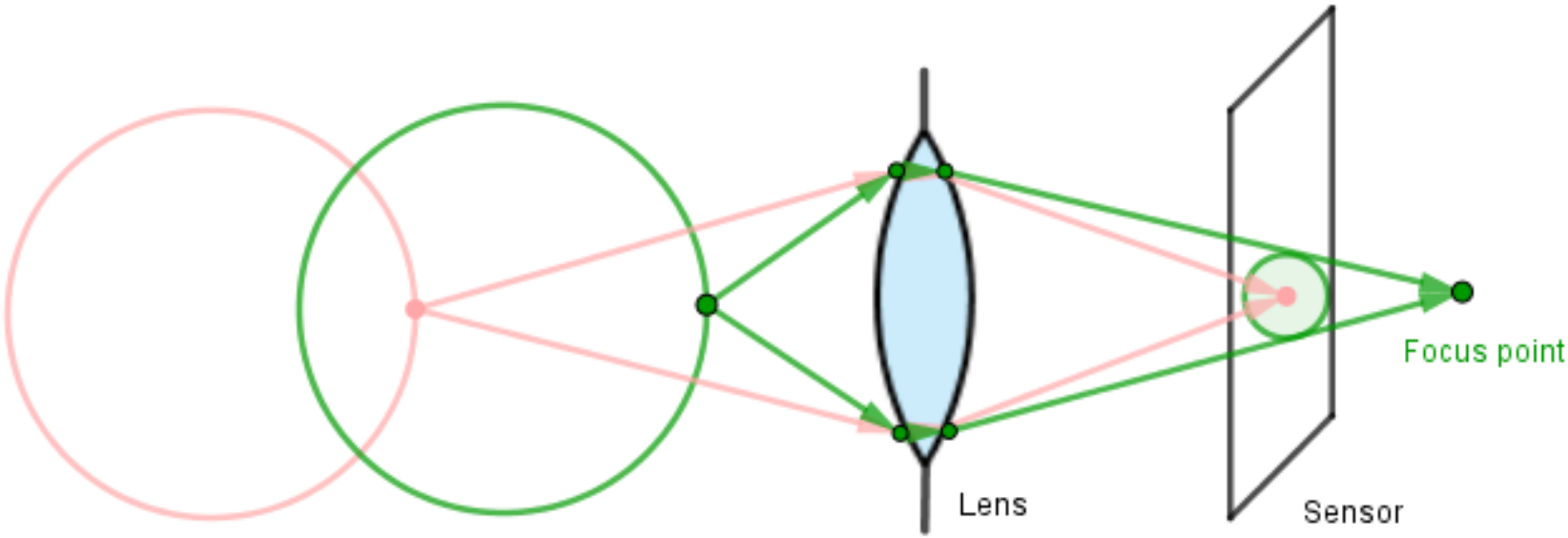
* There are actually differently shaped lenses that each refract light differently for specific goals.

Depth of Field

- Cameras with lenses

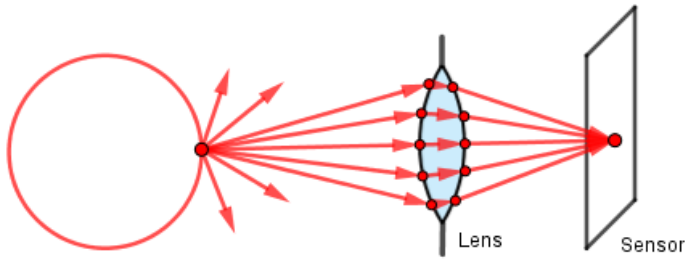


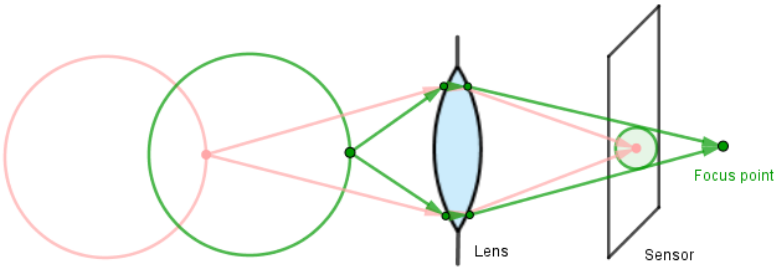
- Close objects are *out of focus* on the sensor



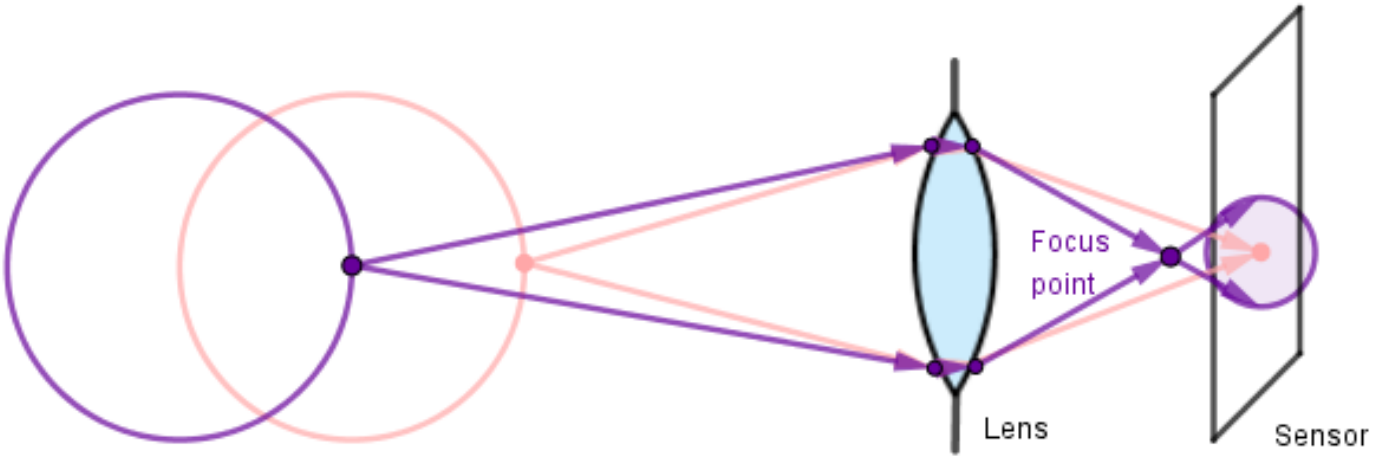
Depth of Field

- Cameras with lenses



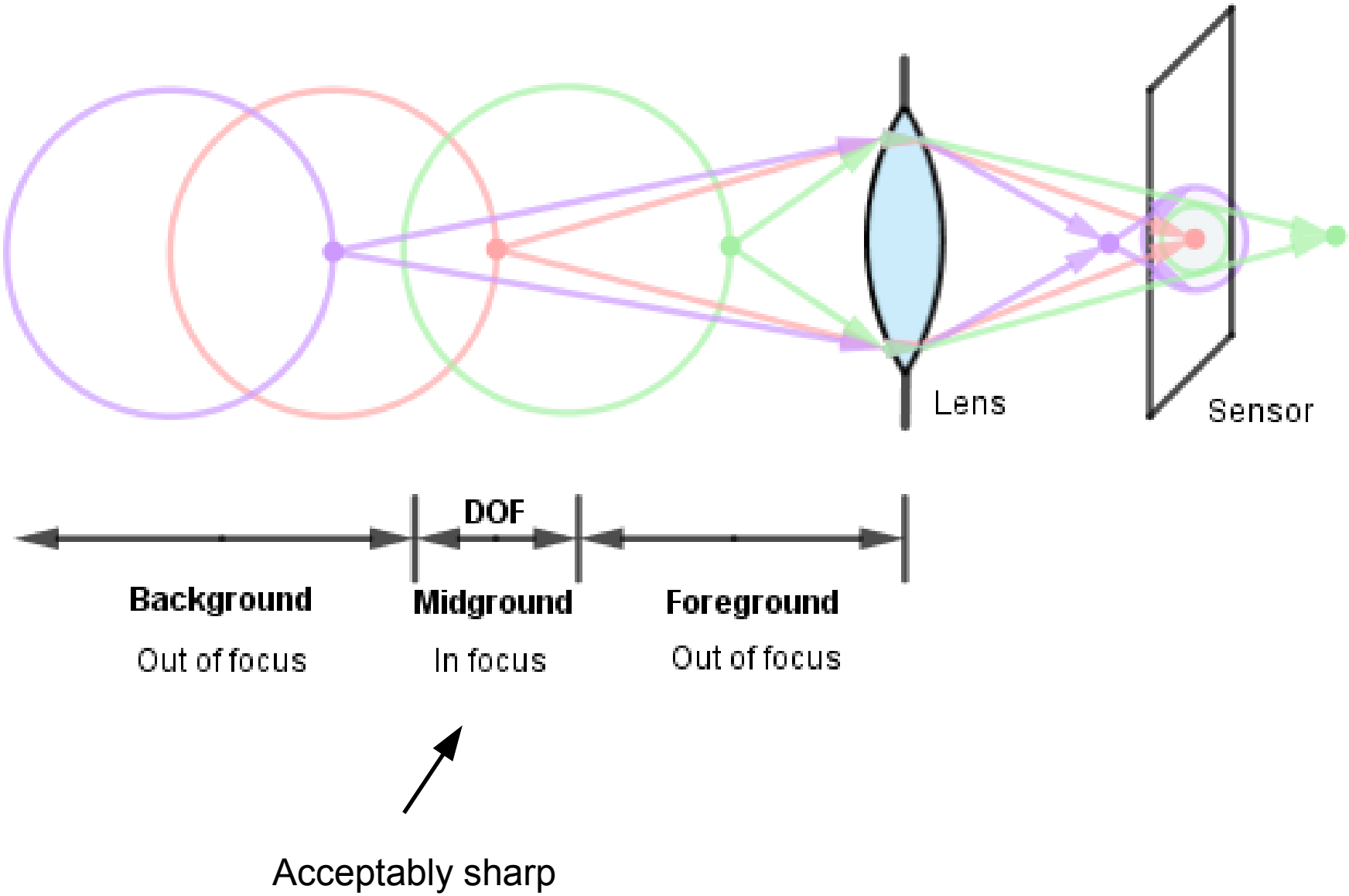


- Close objects are out of focus on the sensor
- Far away objects are out of focus on the sensor

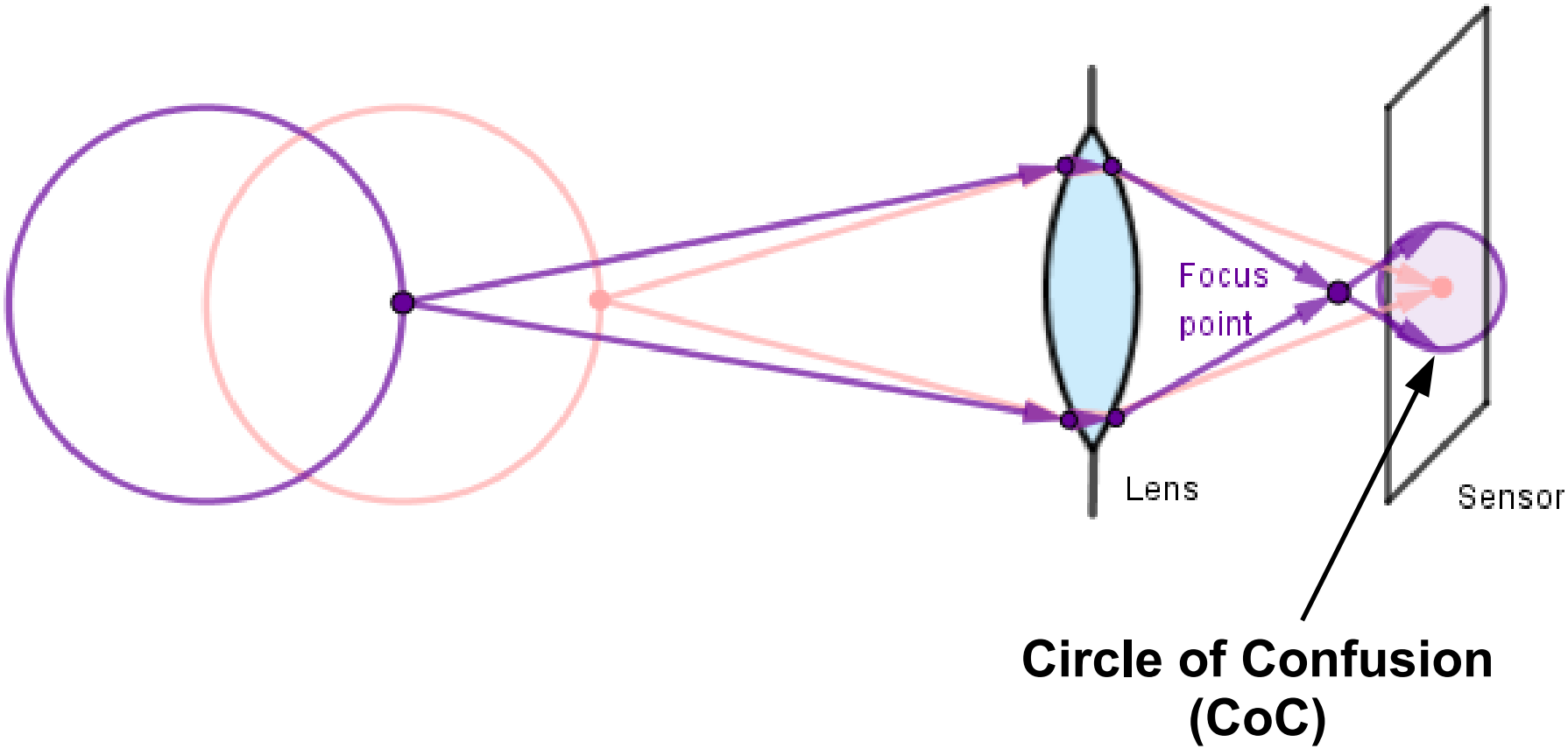


More info: https://photographylife.com/sensor-size-perspective-and-depth-of-field

Depth of Field



Depth of Field



- Area on the sensor where the ouf of focus surface point is spread out in.

Depth of Field

Implementation

Object Space

Image Space

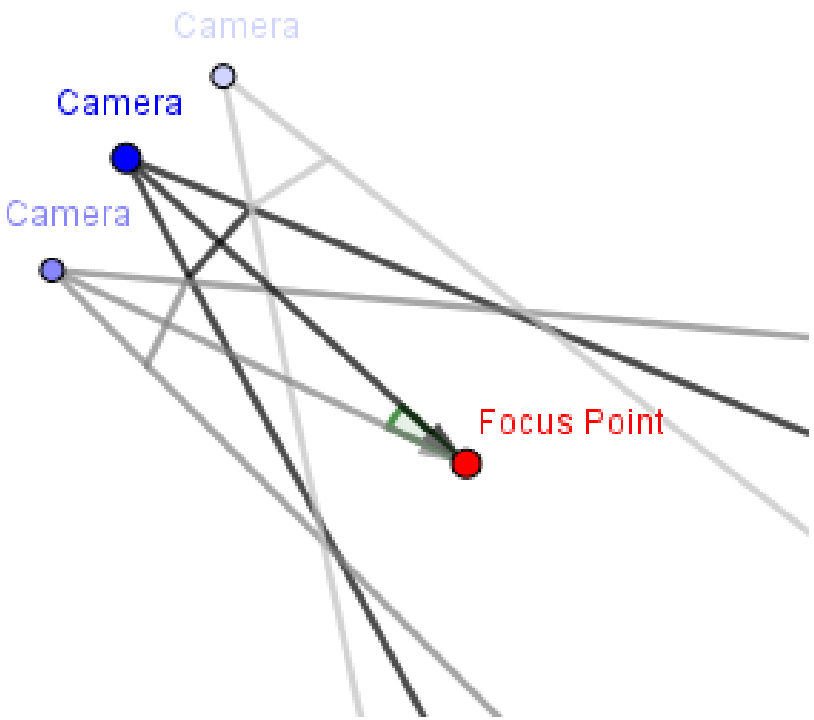
Depth of Field

Implementation

Object Space

Image Space

eg



Depth of Field

Implementation

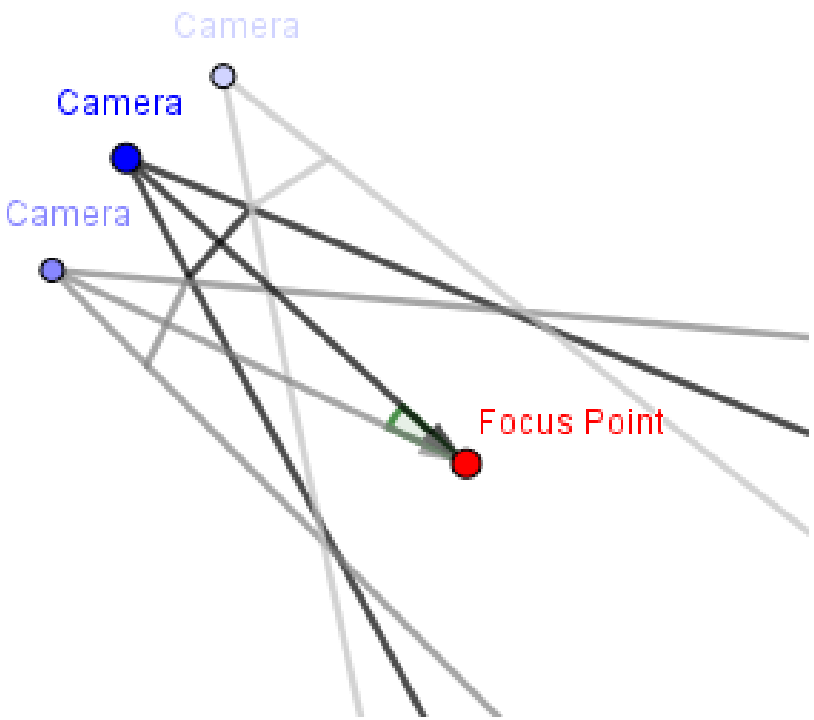
Object Space

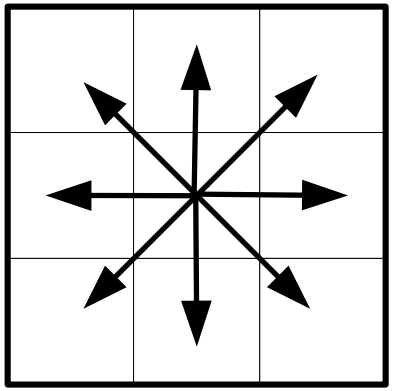
Image Space

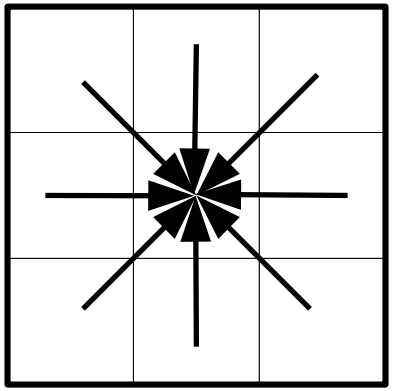
Scatter

Gather

eg







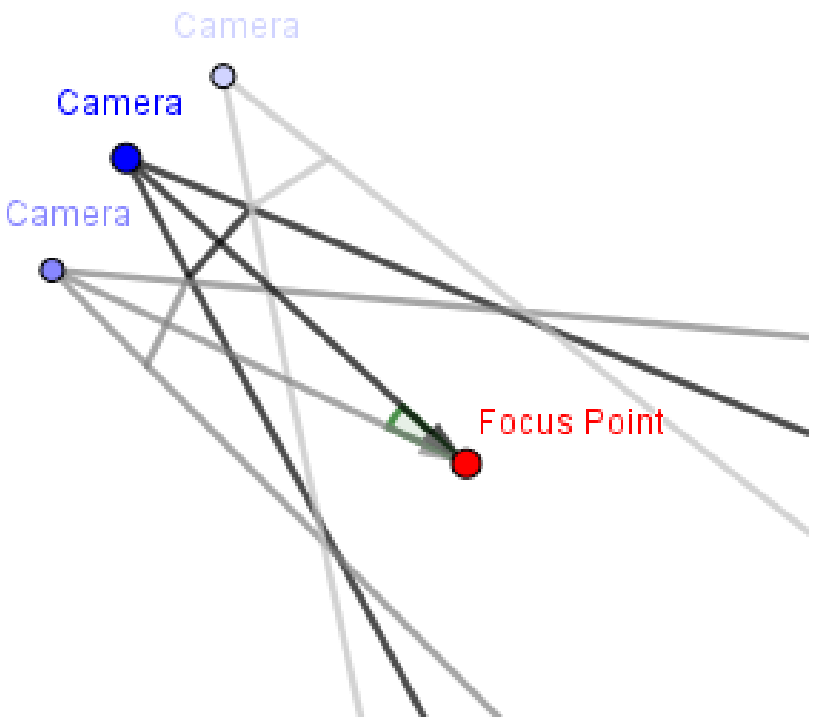
Depth of Field

Implementation

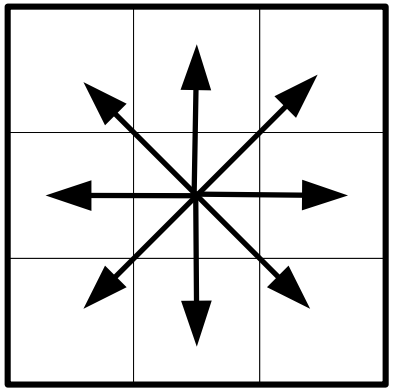
Object Space

Image Space

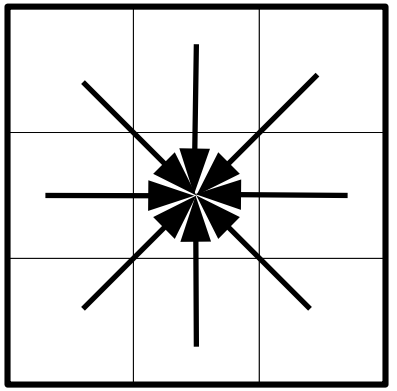
eg



Scatter



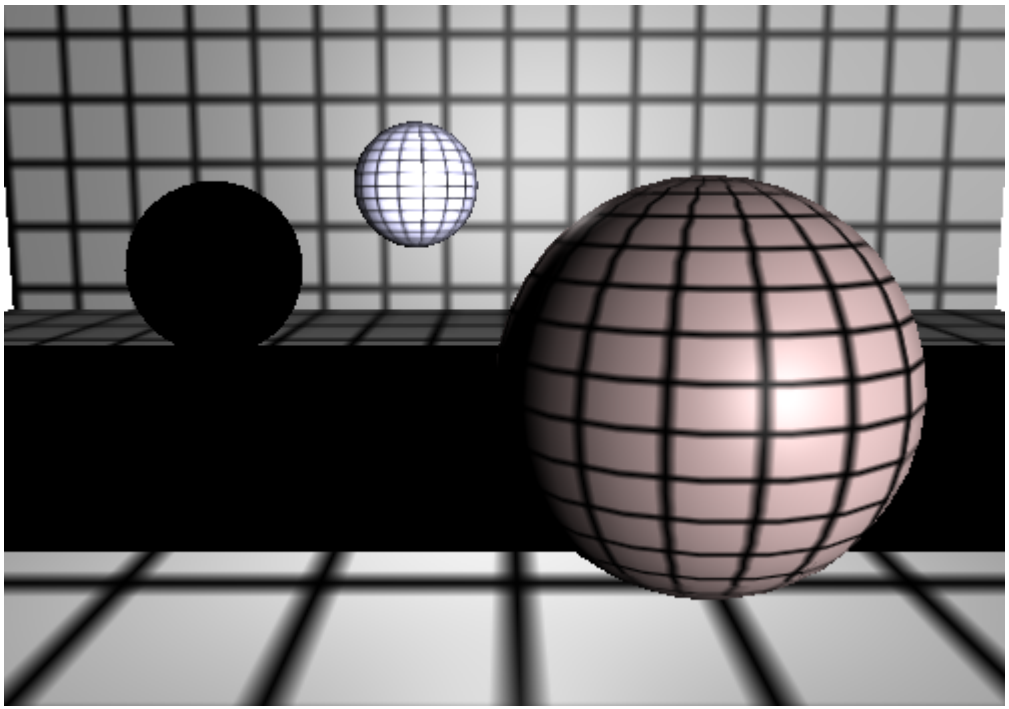
Gather



(convolution)

Depth of Field (1)

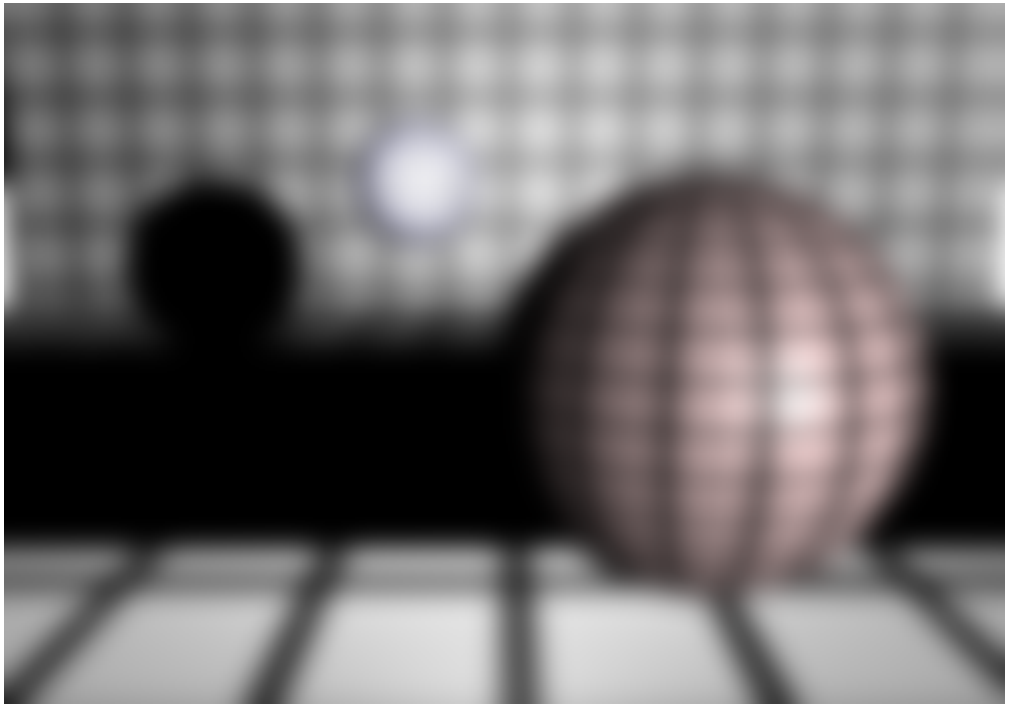
- When rendering:
  - Separate pixels in the fore- or background



Typical beginning when someone explains DoF.

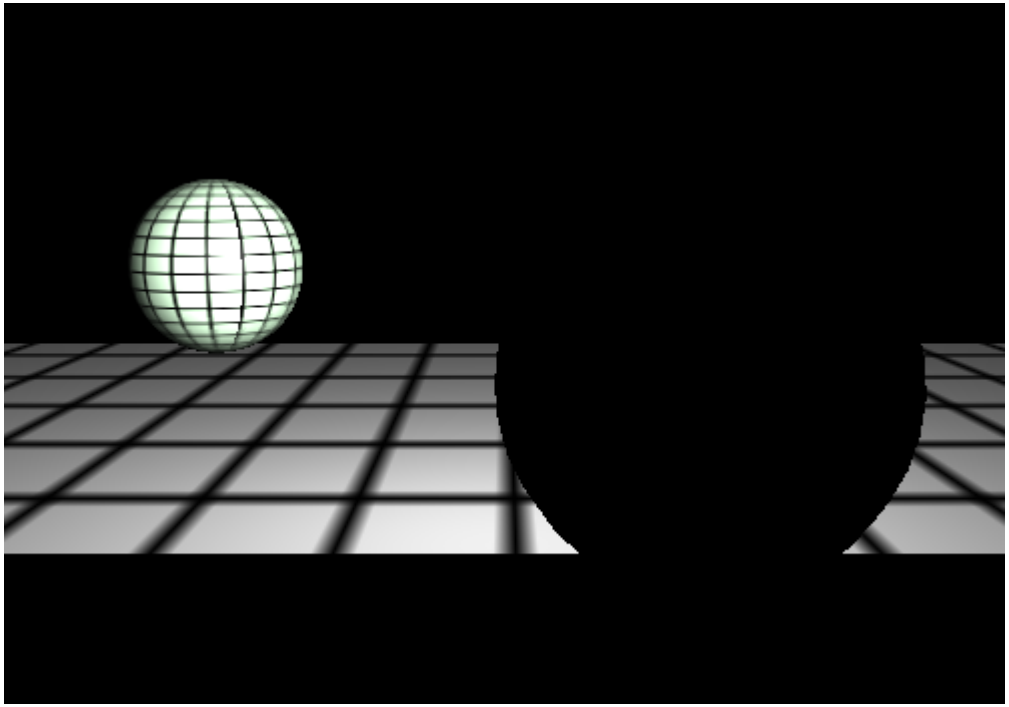
Depth of Field (1)

- When rendering:
  - Separate pixels in the fore- or background
  - Blur those pixels



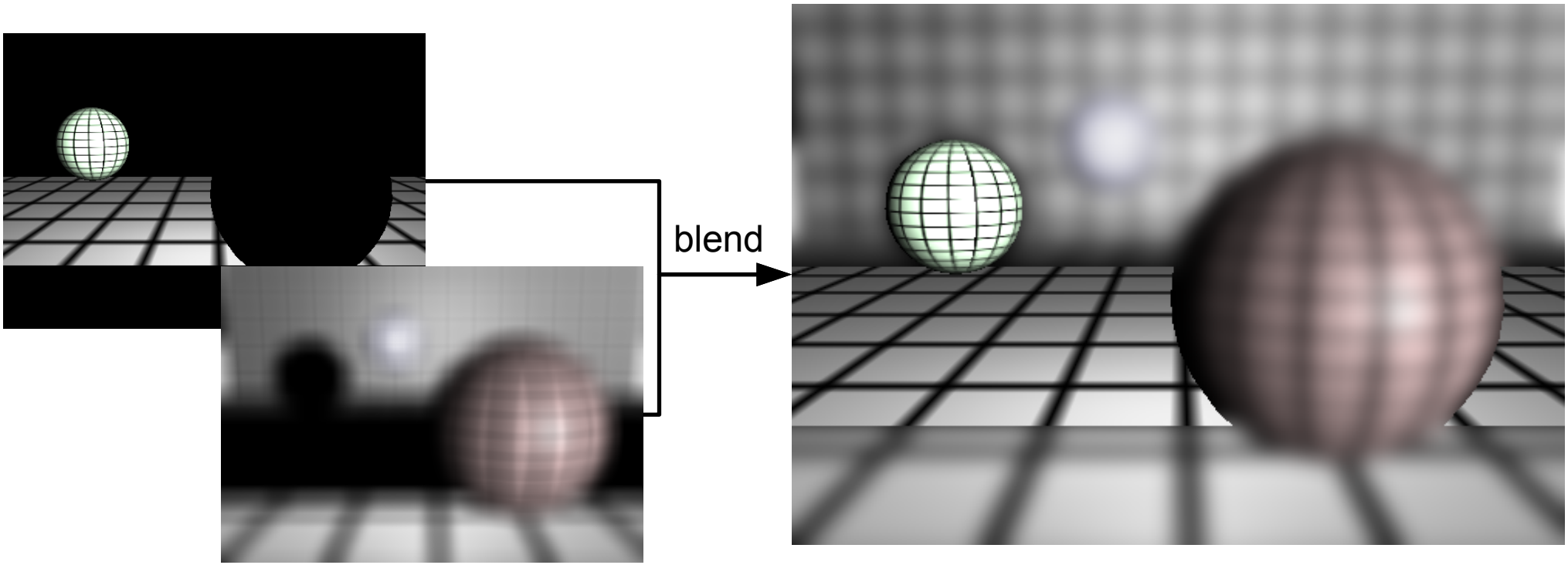
Depth of Field (1)

- When rendering:
  - Separate pixels in the fore- or background
  - Blur those pixels
  - Render the midground



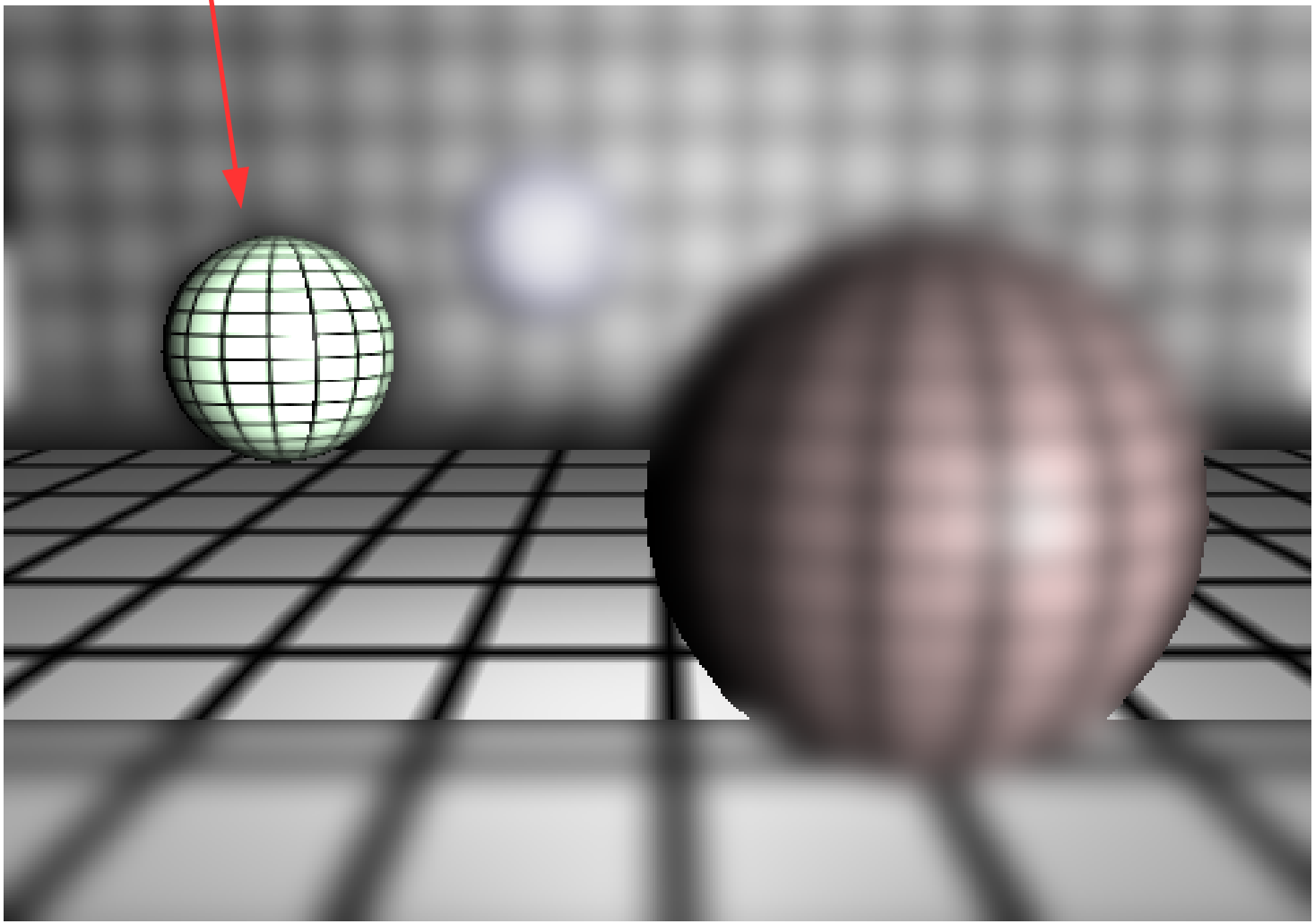
Depth of Field (1)

- When rendering:
  - Separate pixels in the fore- or background
  - Blur those pixels
  - Render the midground and blend the results



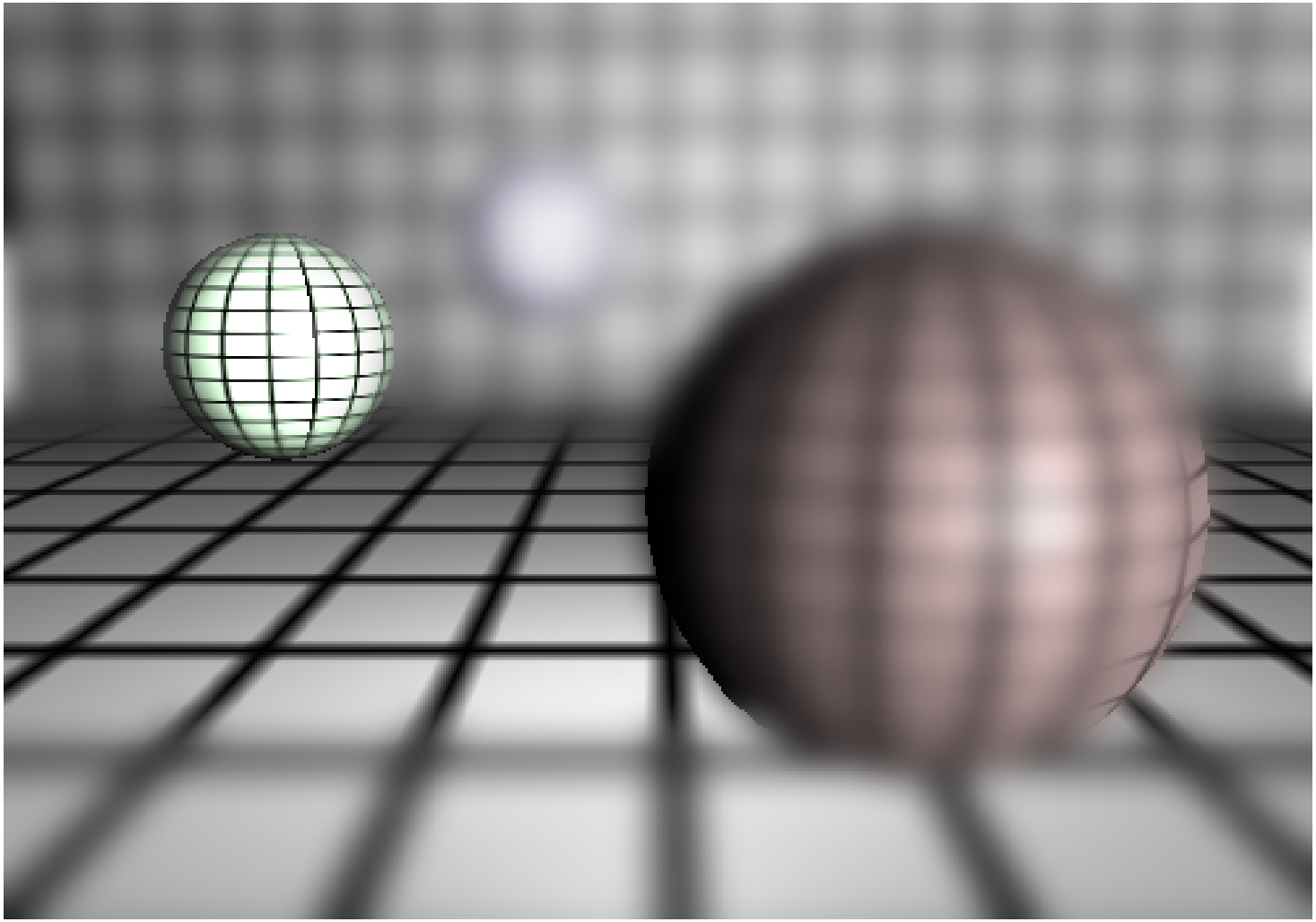
Depth of Field (1)

Transparent pixels on the screen



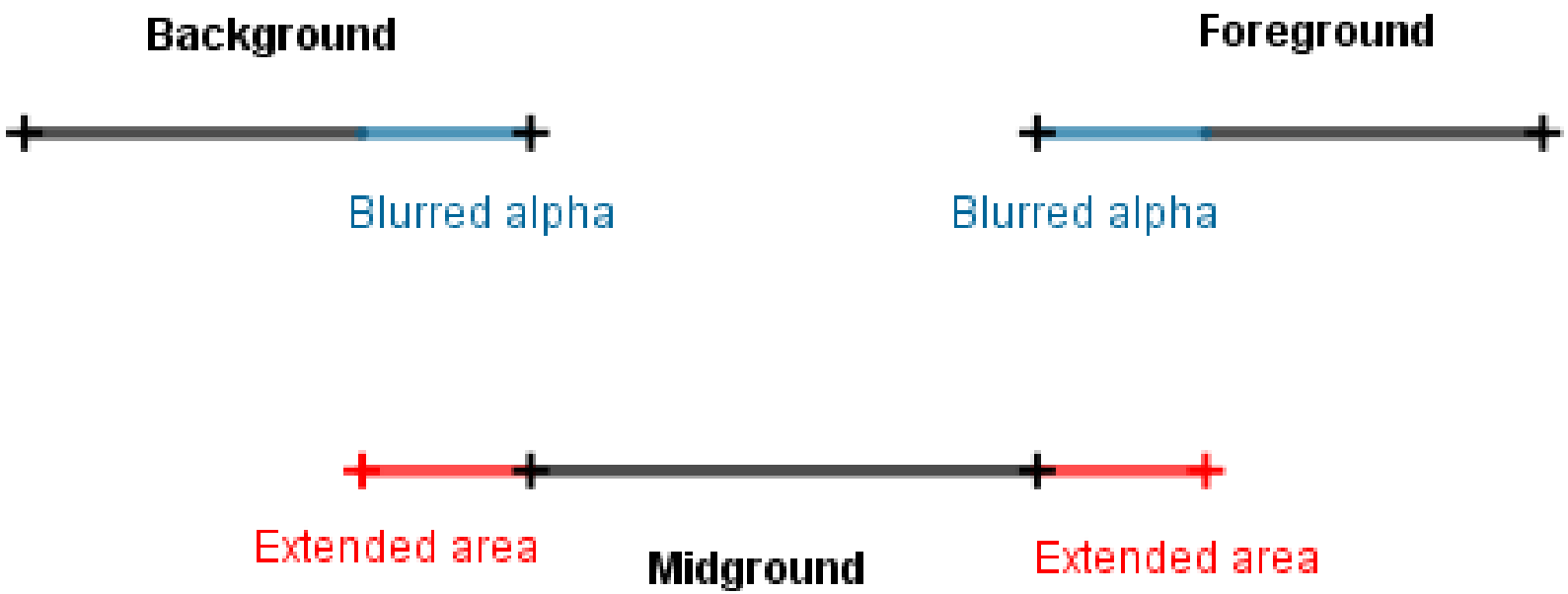
Depth of Field (2)

Extend the midground to cover them.



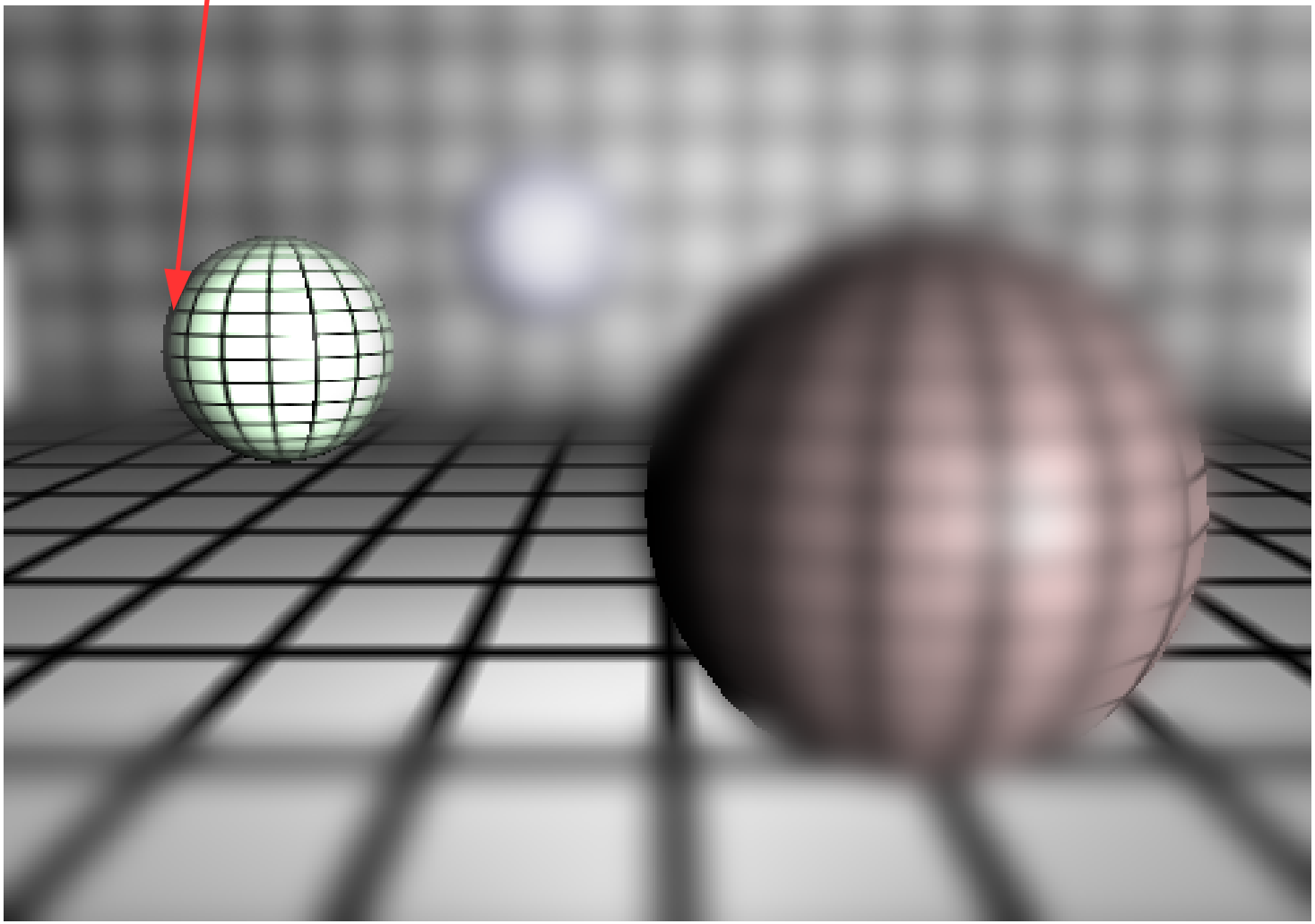
Depth of Field (2)

- Current layers:



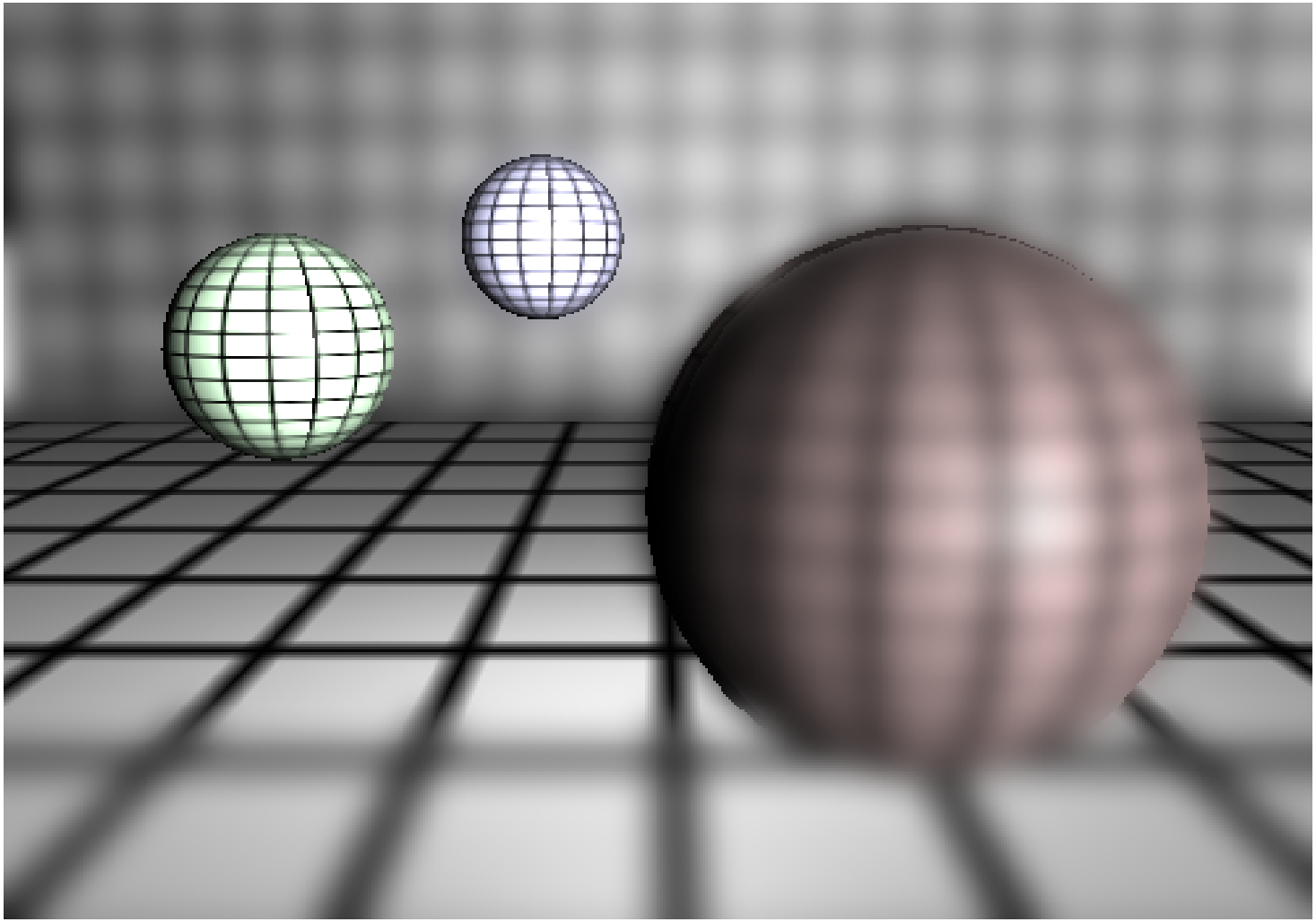
Depth of Field (2)

Background blur should not cover objects in the front



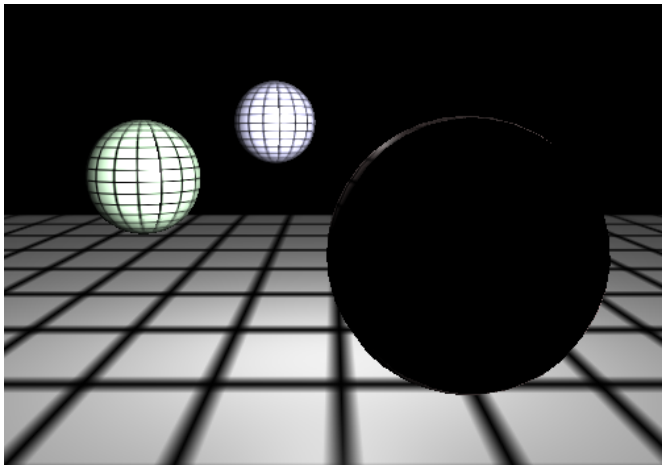
Depth of Field (3)

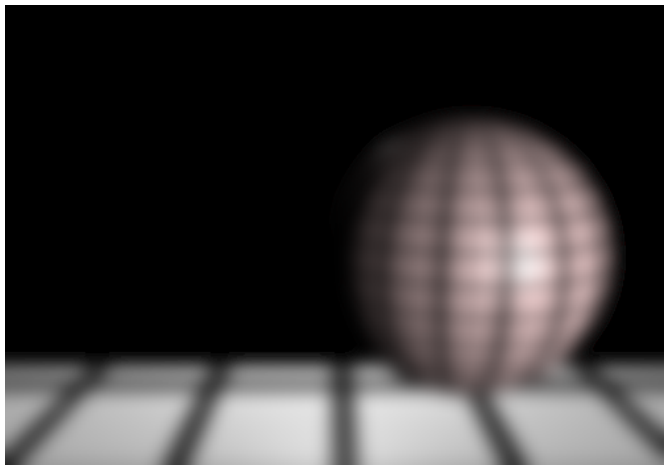
Render background first, then blend midground, then foreground.

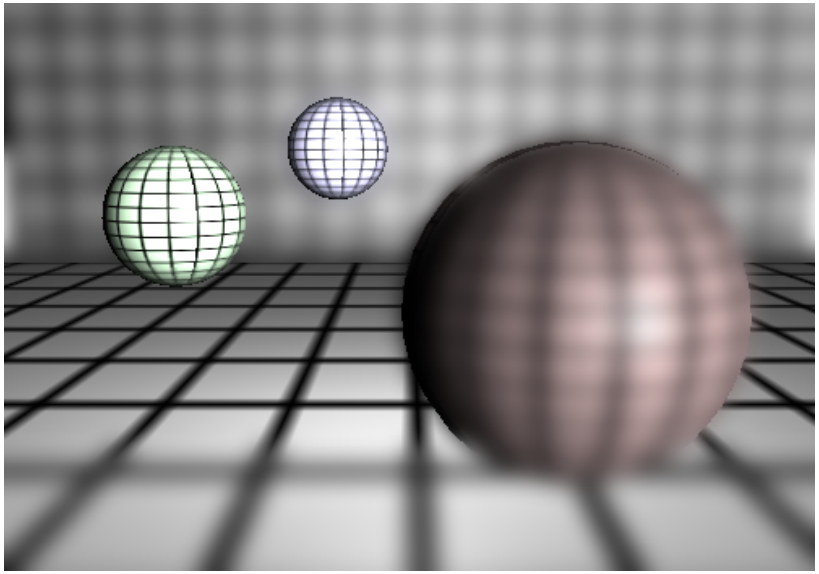


Depth of Field (3)

Render background first, then blend midground, then foreground.

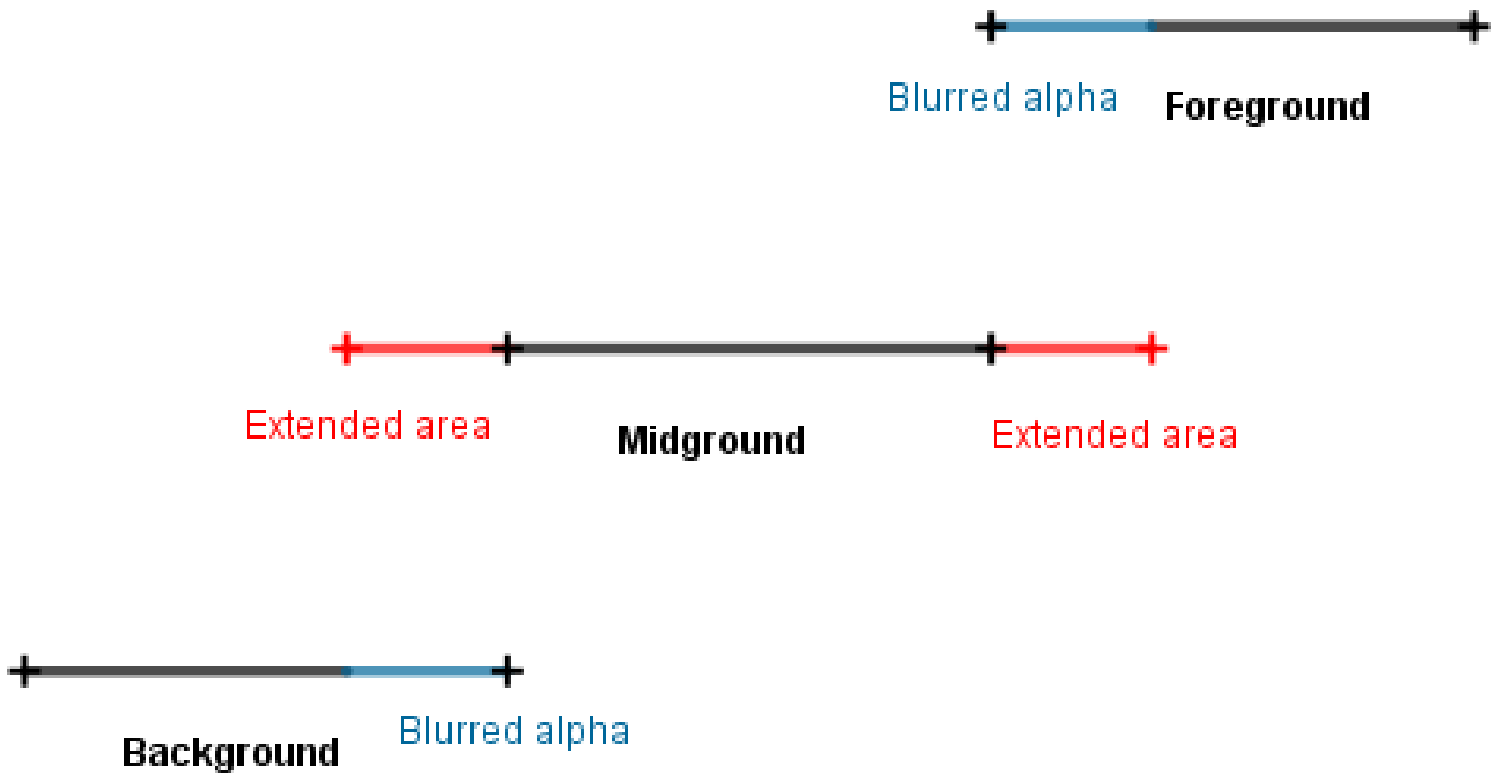






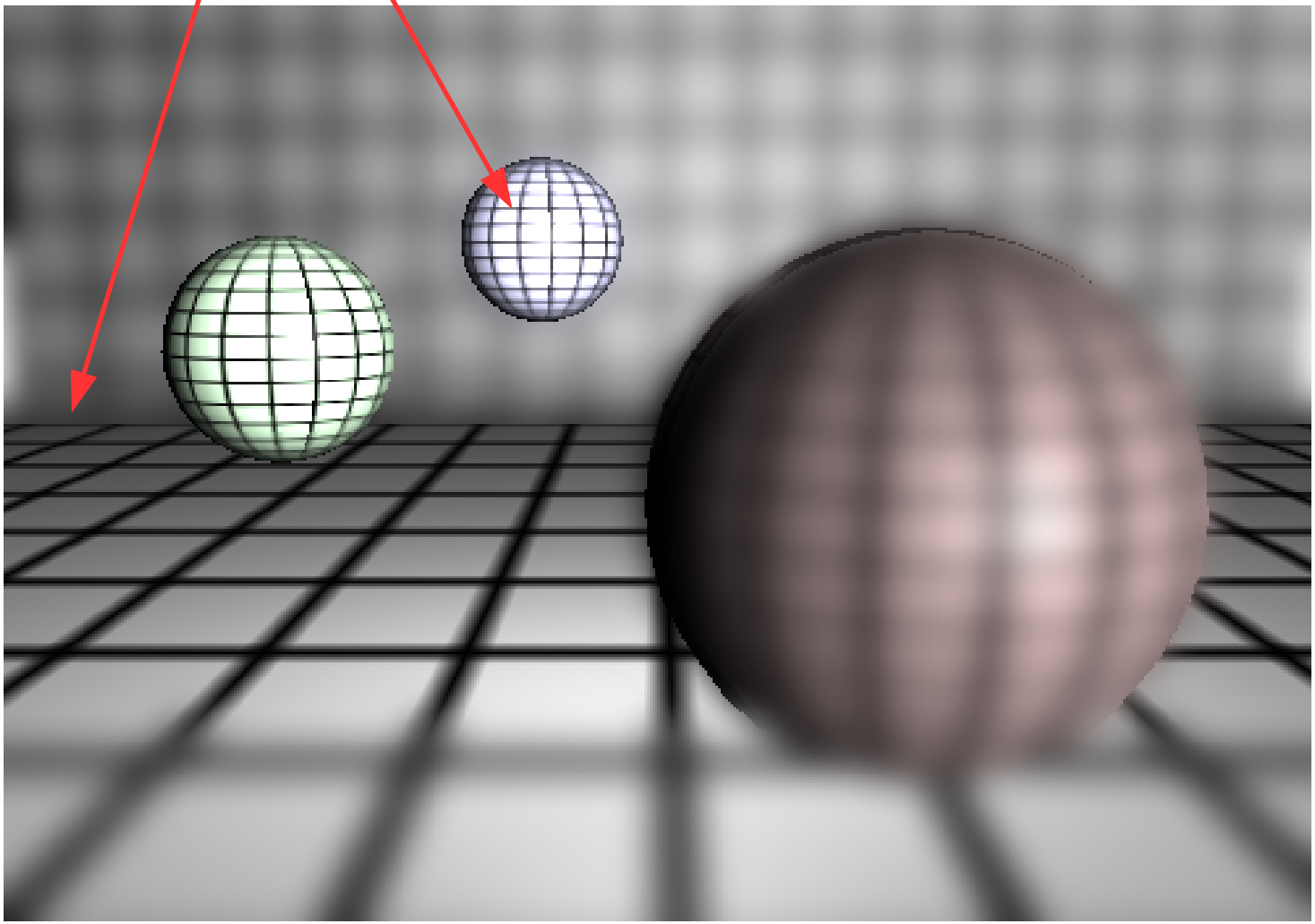
Depth of Field (3)

- Current layers:



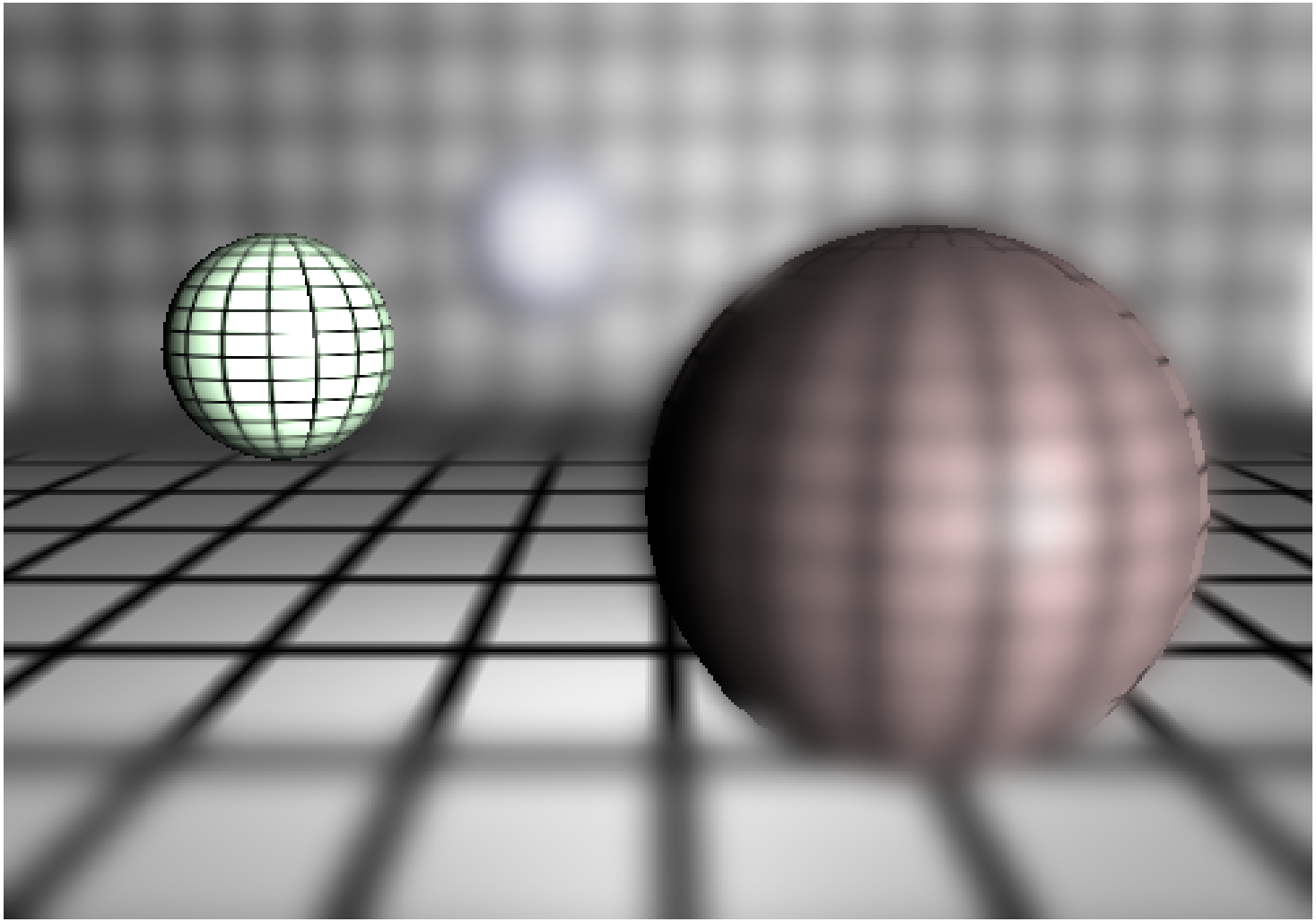
Depth of Field (3)

Too sharp border between the midground and background



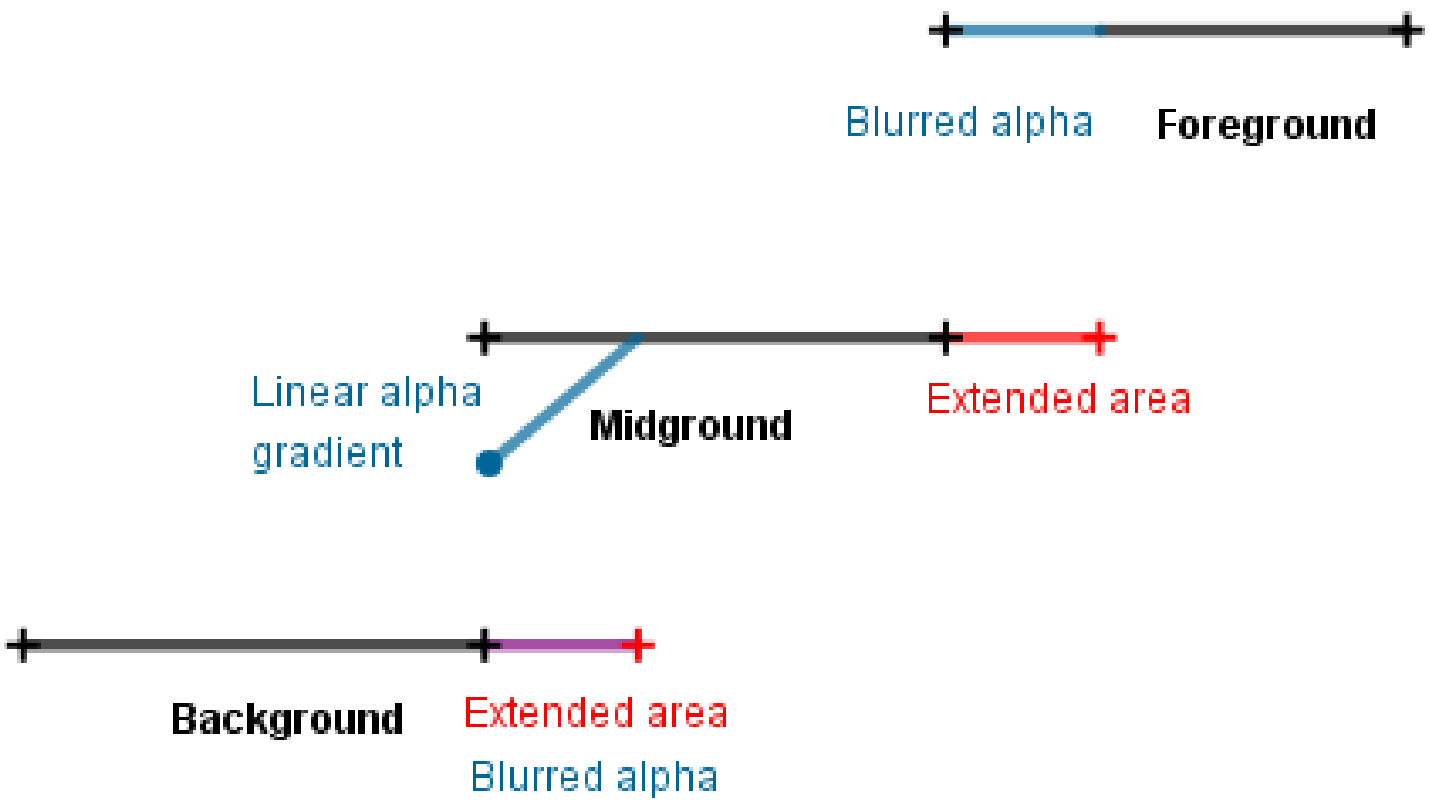
Depth of Field (4)

Extend the background instead of the midground.
Create a linear gradient between the midground and background



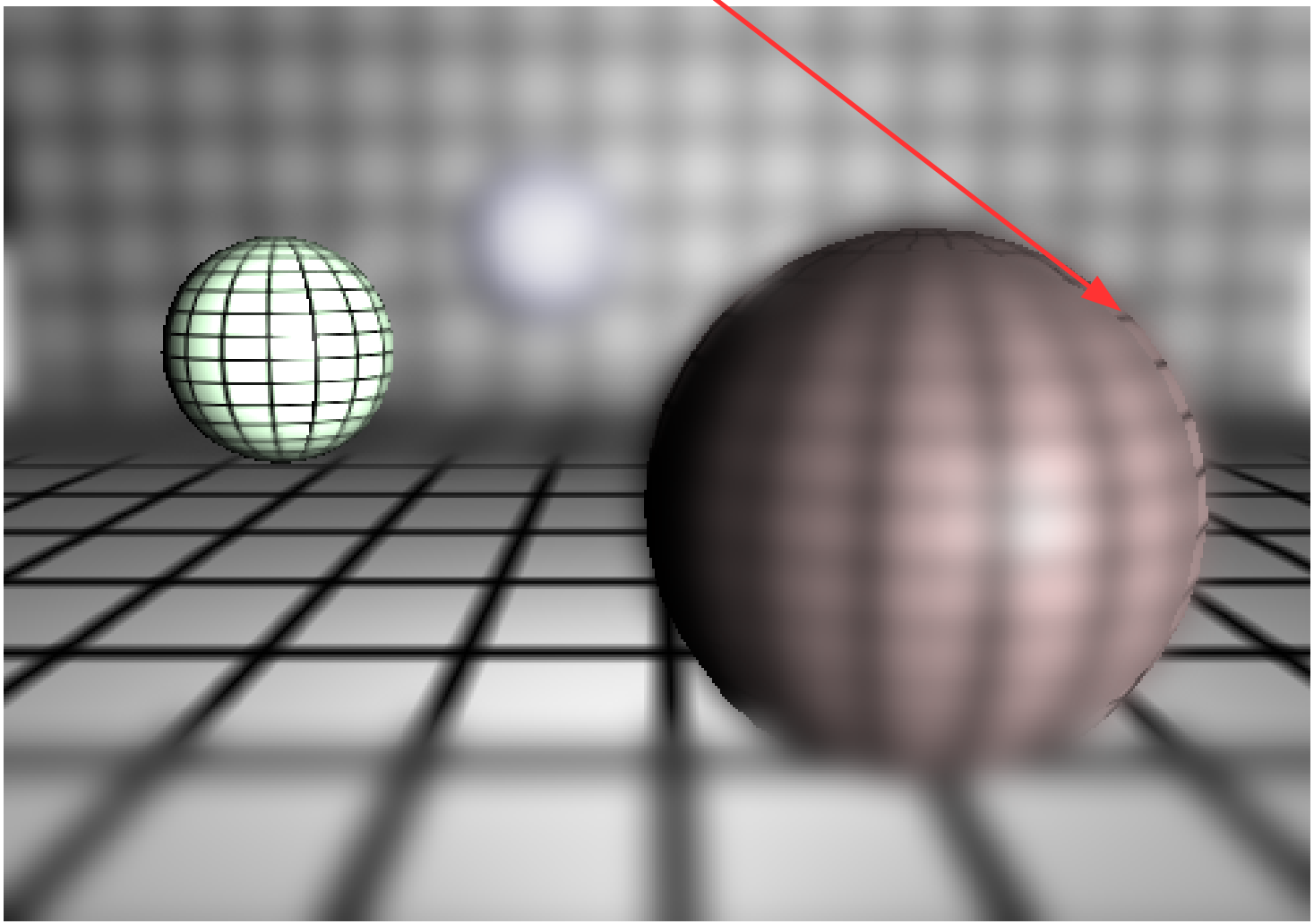
Depth of Field (4)

- Current layers:



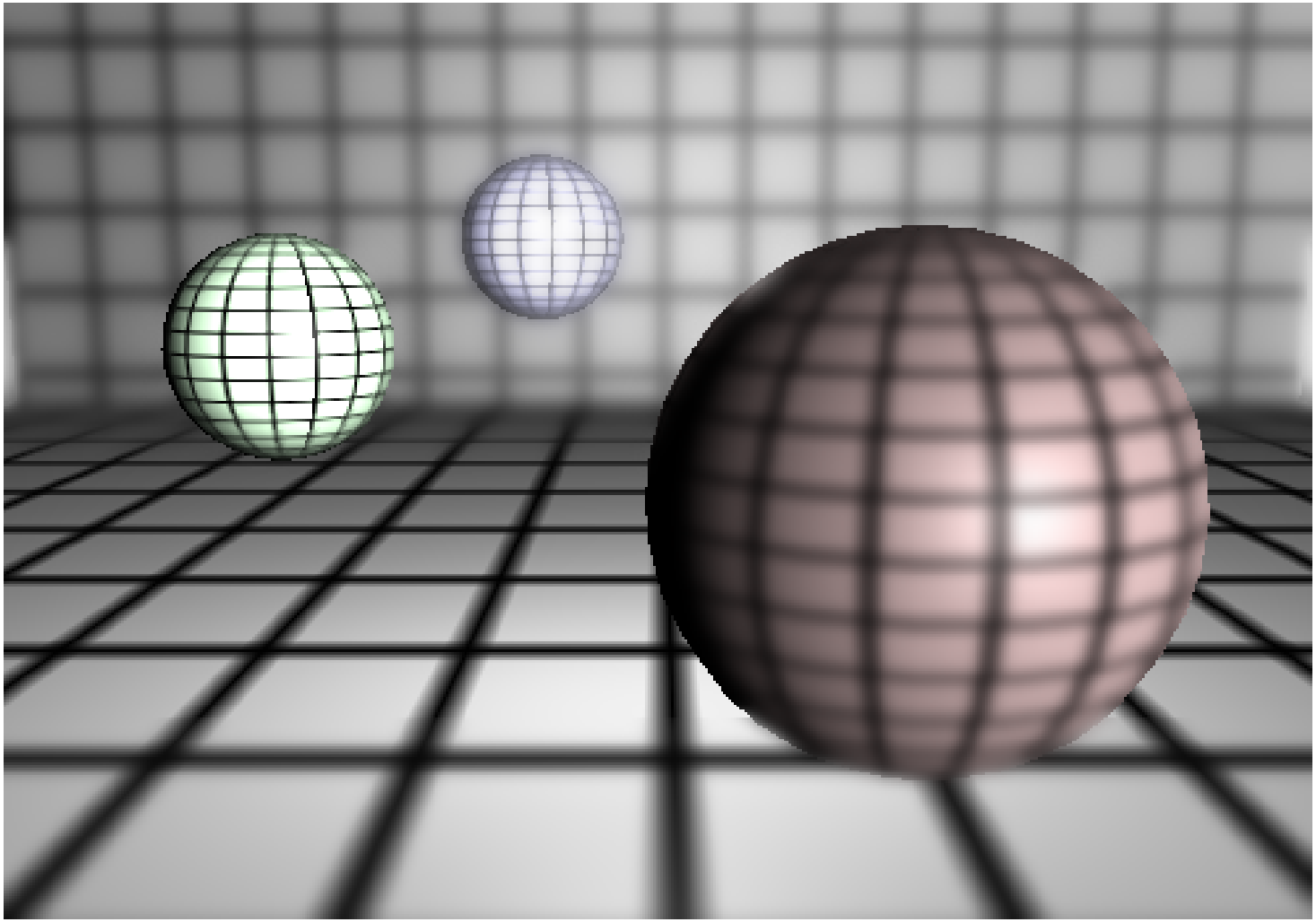
Depth of Field (4)

Sharp border between the foreground and midground.



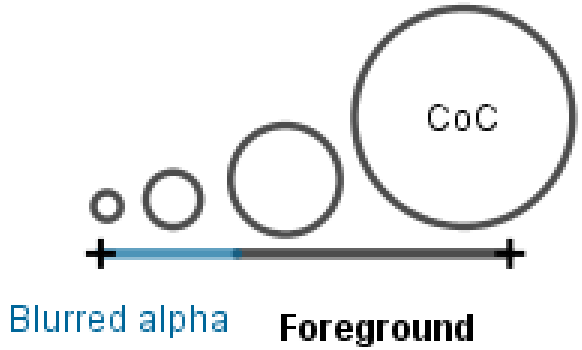
Depth of Field (5)

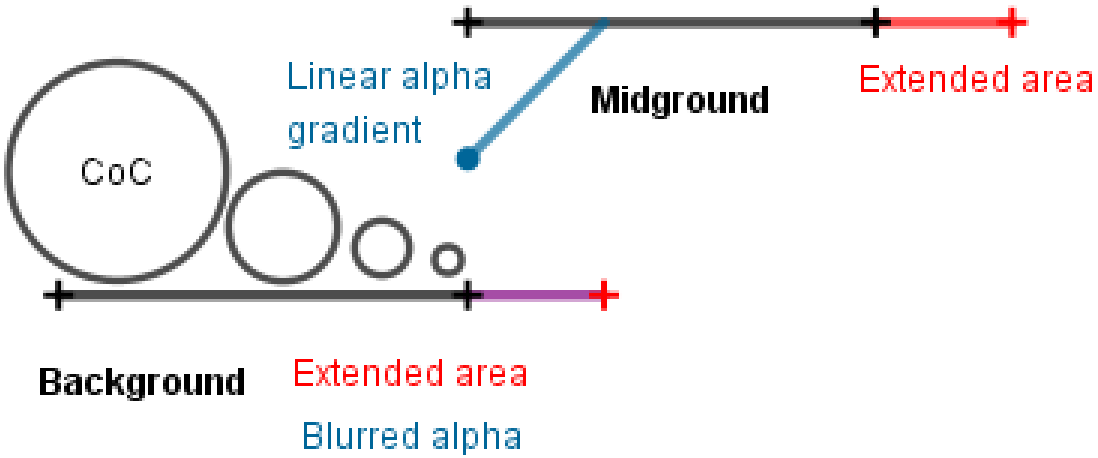
Use a varying size blur.



Depth of Field (5)

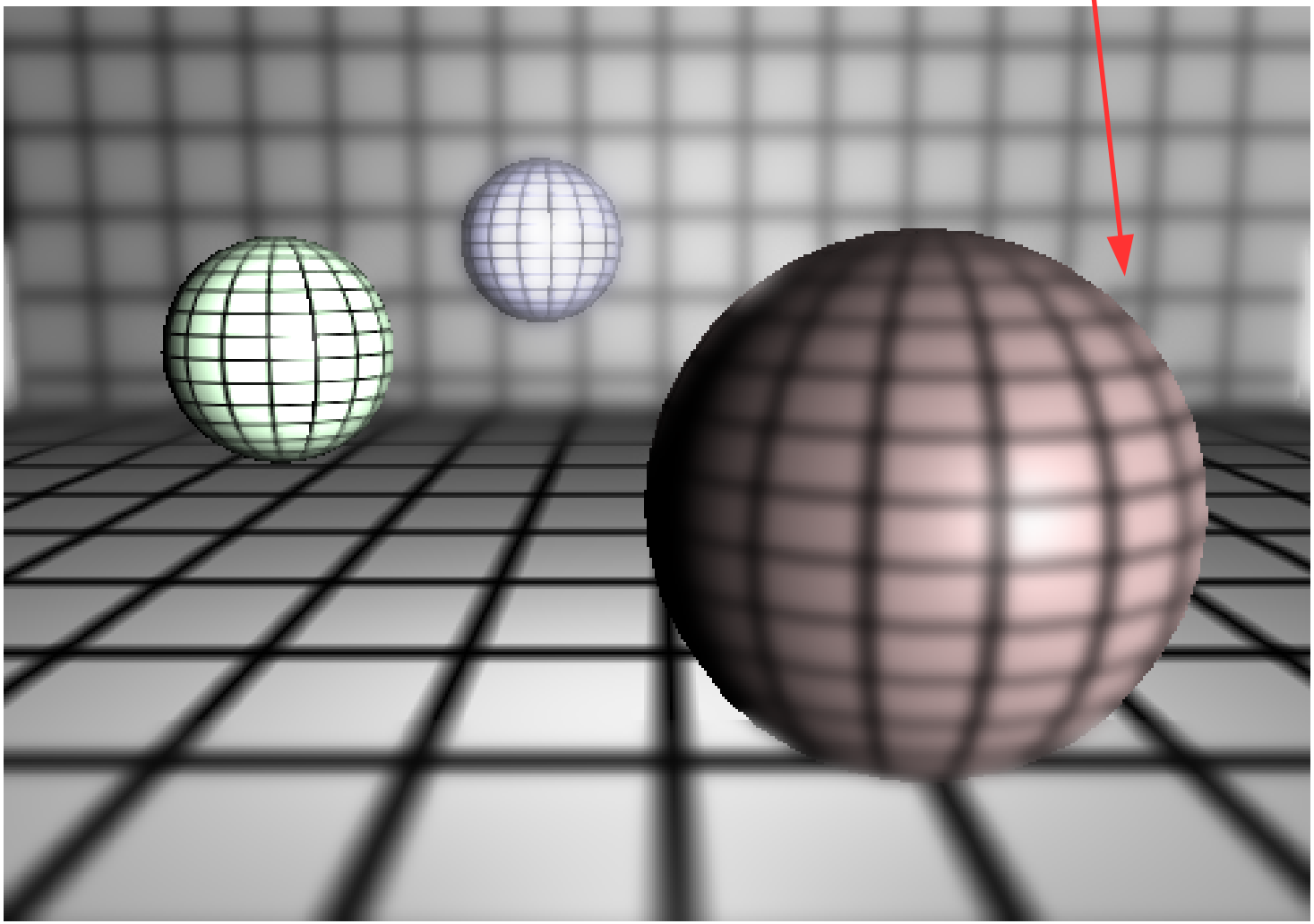
- Current layers:





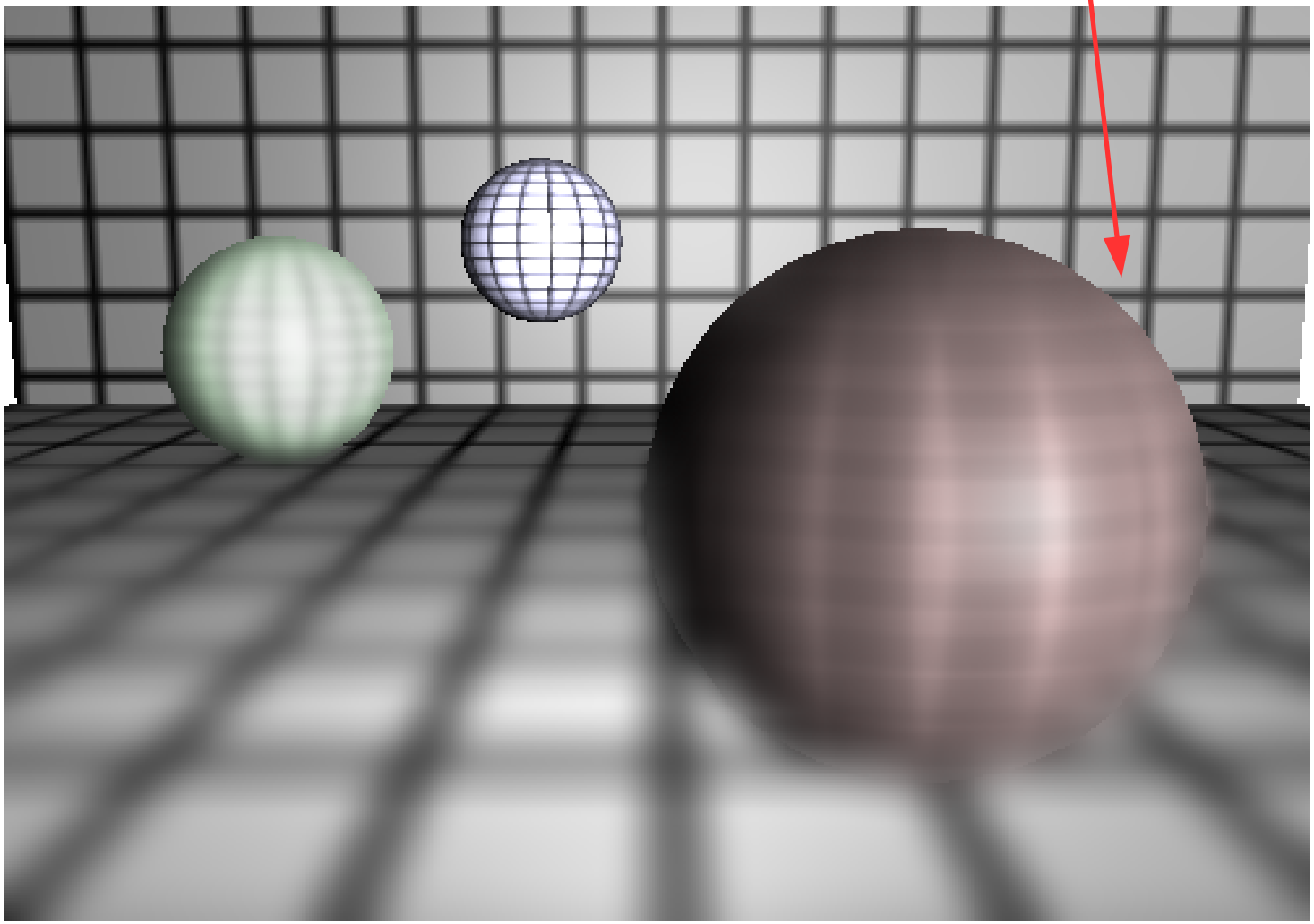
Depth of Field (5)

When CoC is read only from the current pixel, there will be no outside blur (halo).



Depth of Field (5)

When CoC is read only from the current pixel, there will be no outside blur (halo).



Depth of Field (5)

Transparent pixels in front of midground.

CoC = 0
No blur applied

0	0	0	0	0
1	0	0	0	0
1	1	0	0	0
1	1	1	0	0
1	1	1	1	0

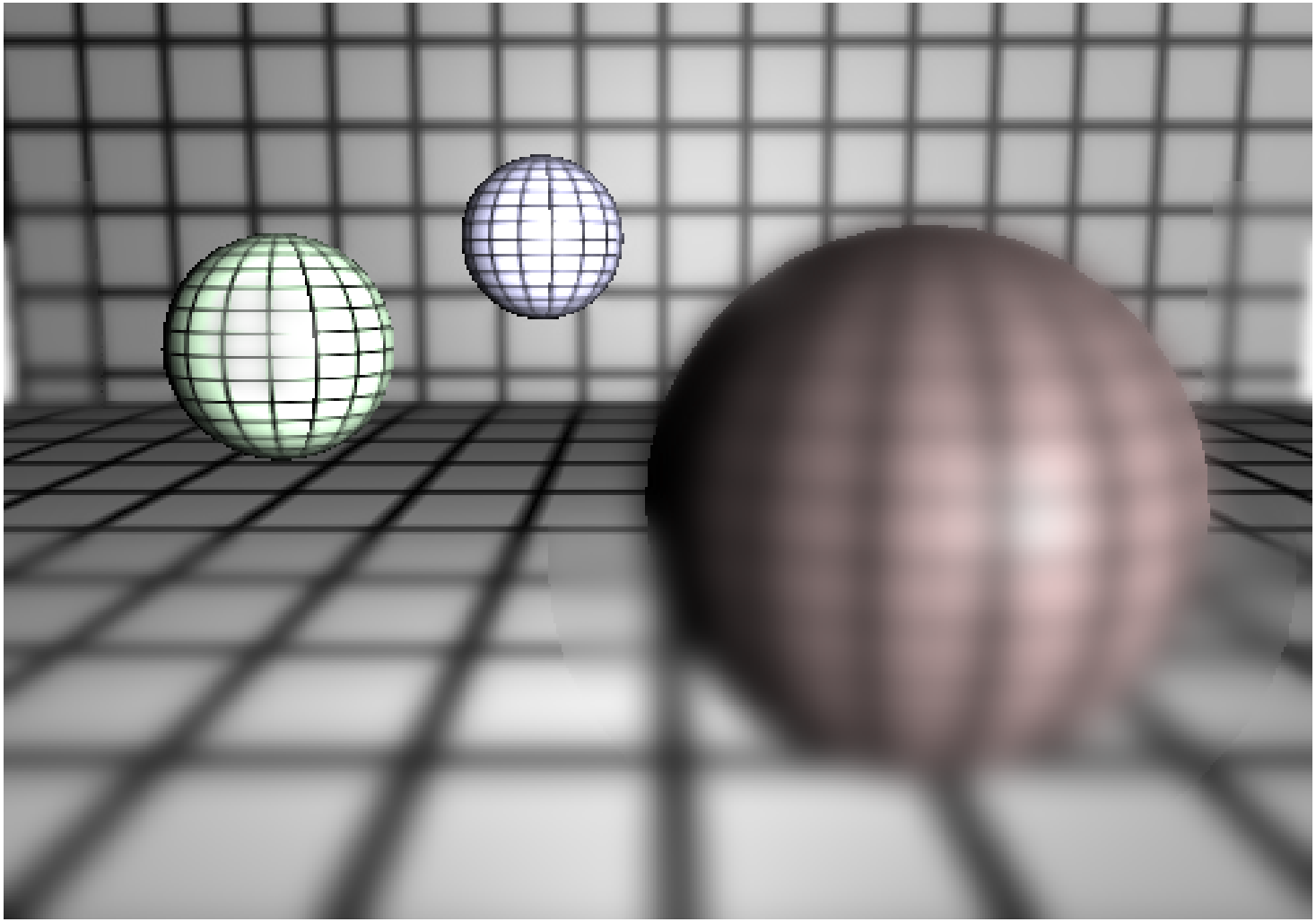
Foreground

CoC = 1
Full blur applied

Difference between scatter and gather techniques.

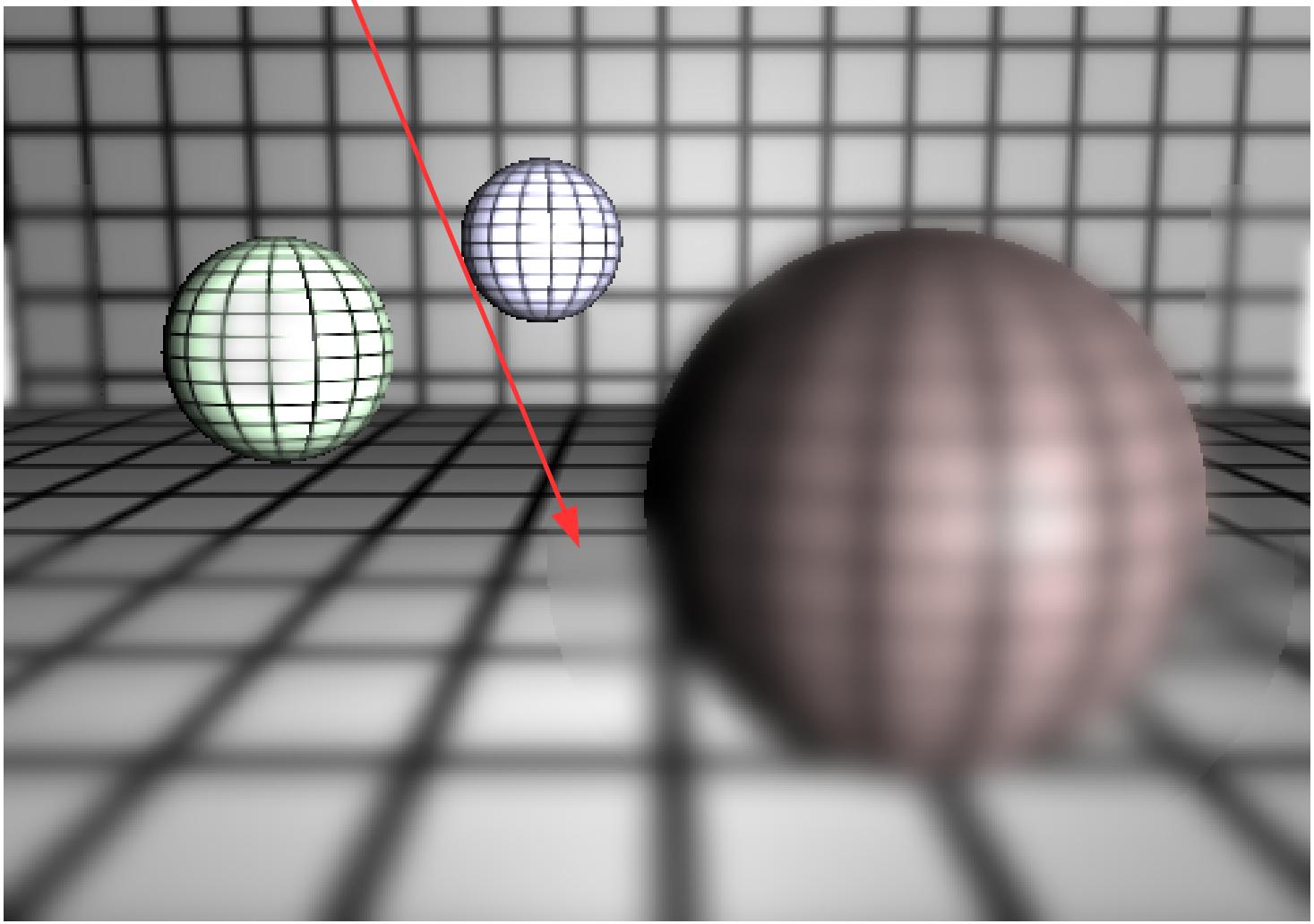
Depth of Field (6)

Take CoC to be MAX CoC from the neighbourhood.



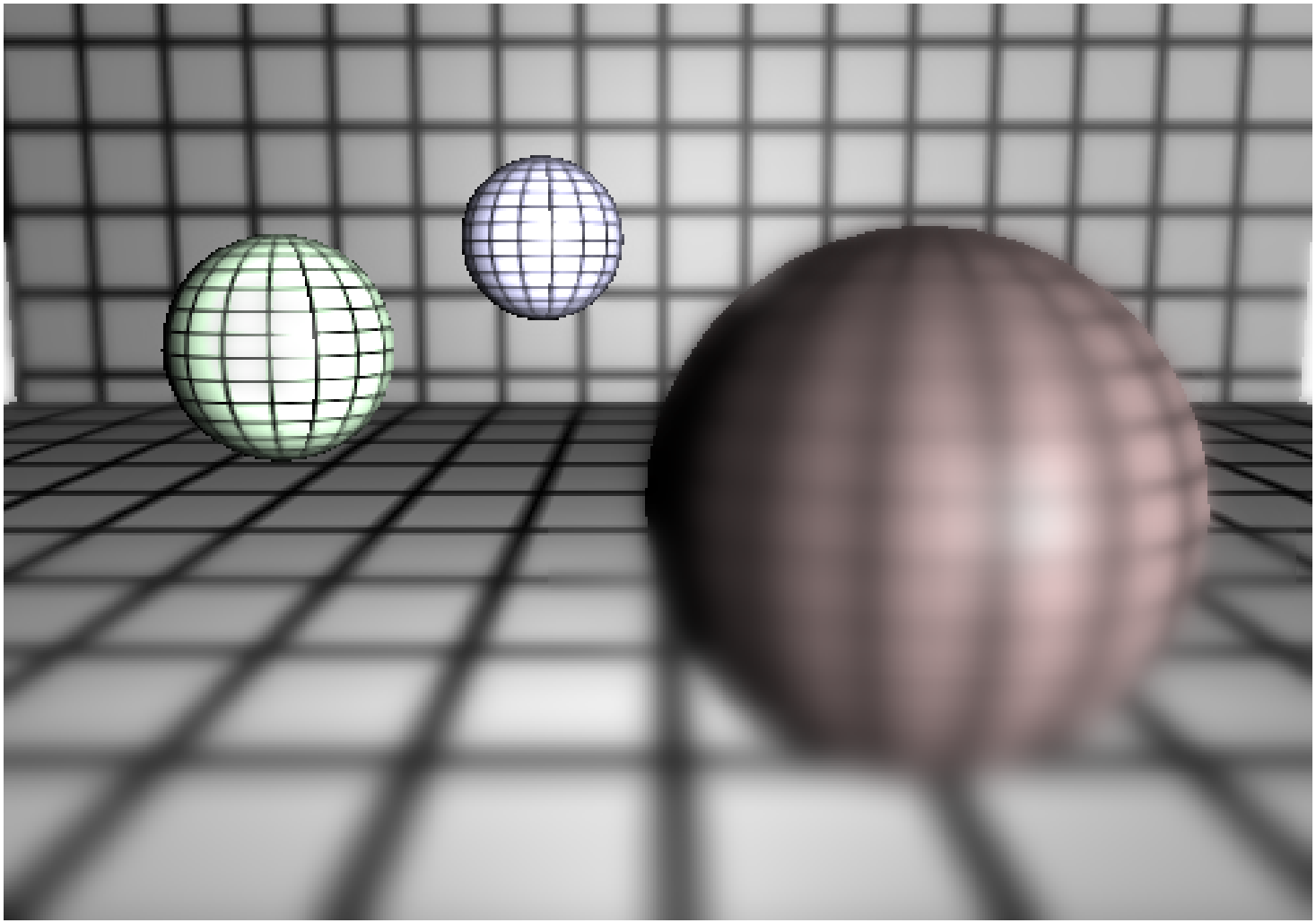
Depth of Field (6)

Sharp large edge around objects with different CoC values.



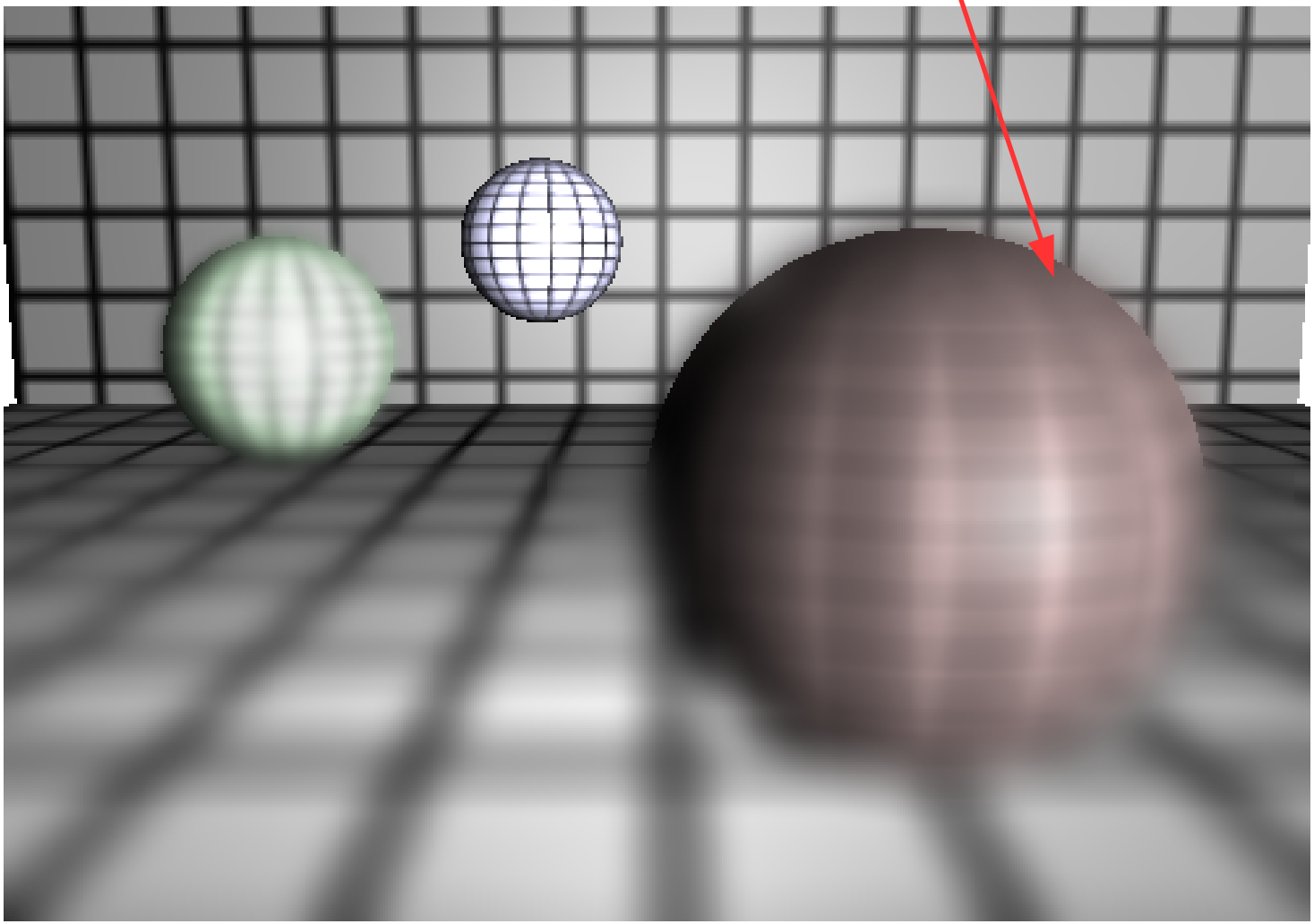
Depth of Field (7)

Take CoC to be a mix between the MAX CoC and AVG CoC from the neighbourhood.
Mix by the percentage of 0 transparency pixels.



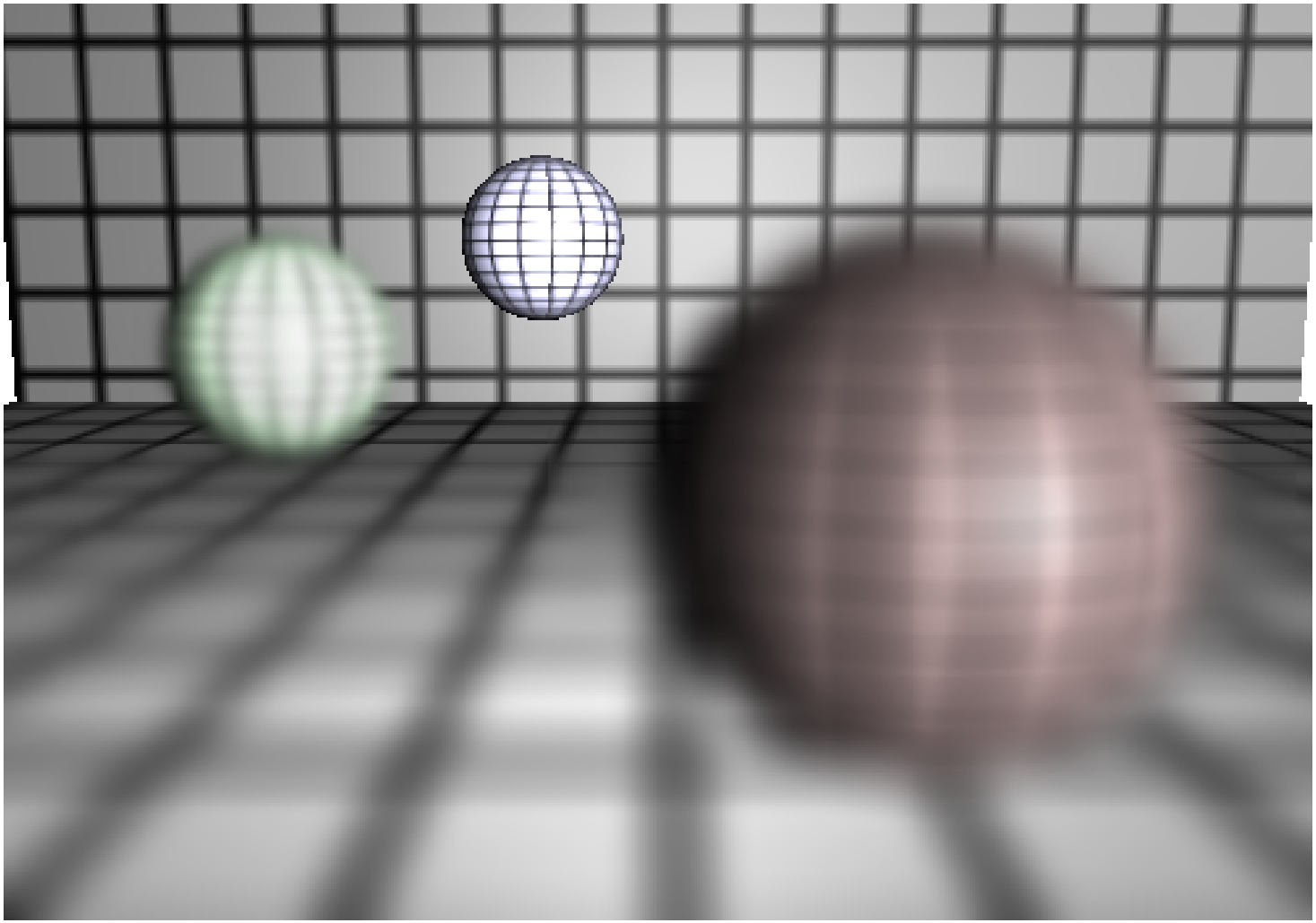
Depth of Field (7)

No visible inside blur on the foreground. There is no data to blend inside pixels with.



Depth of Field (8)

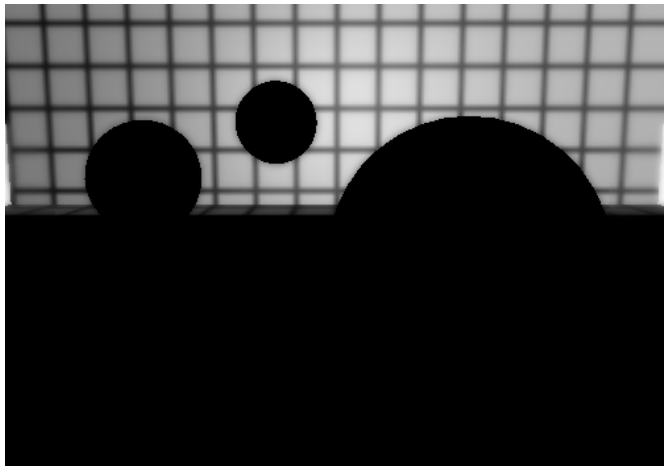
Render the midground with a clipping plane at the midground-foreground border.



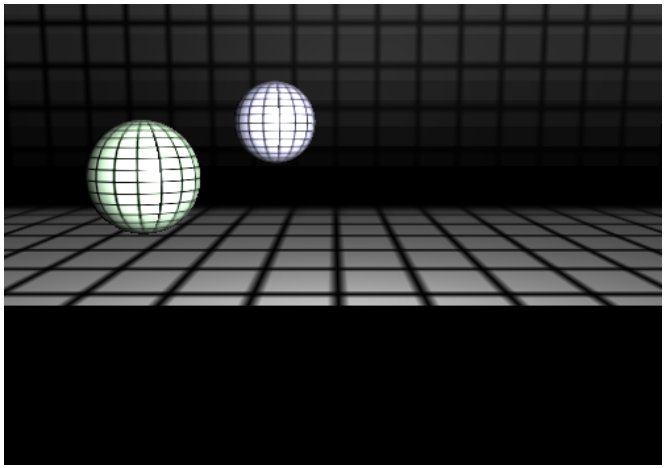
Now the blur is dependent also on the scene's complexity a bit.

Depth of Field (8)

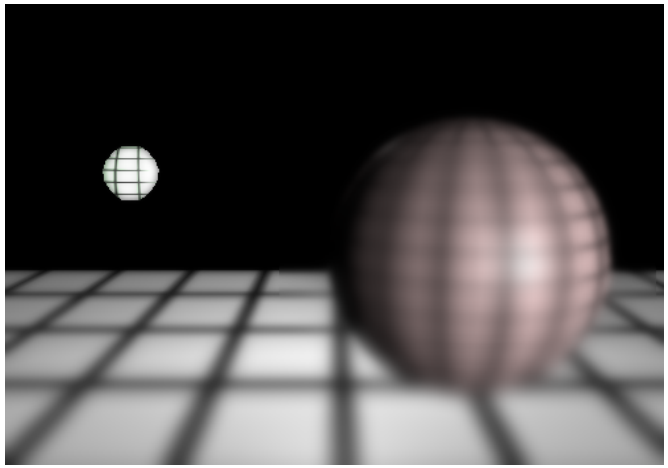
Varyingly blurred background

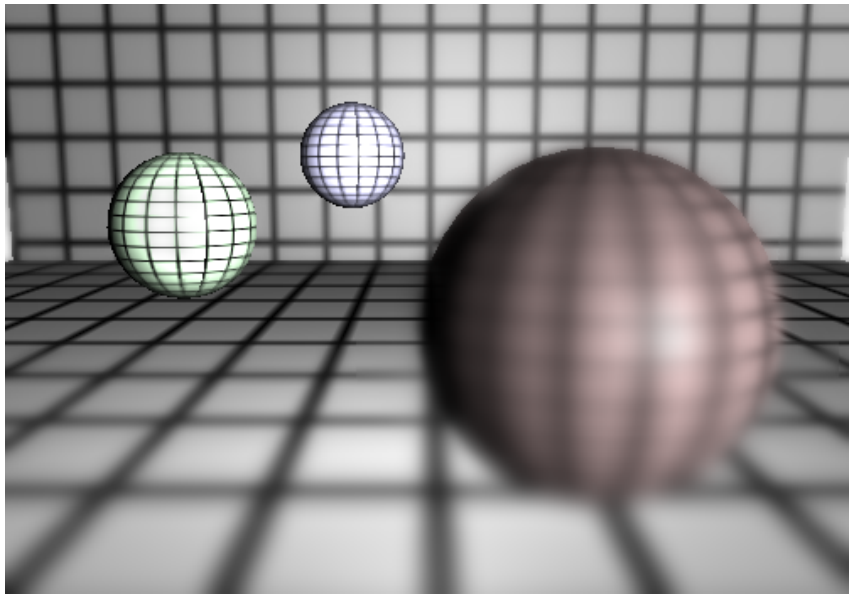


Clipped midground

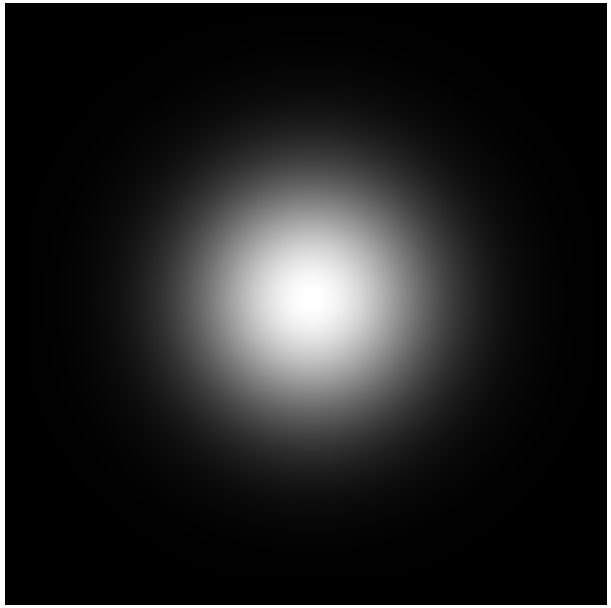


Varyingly blurred foreground



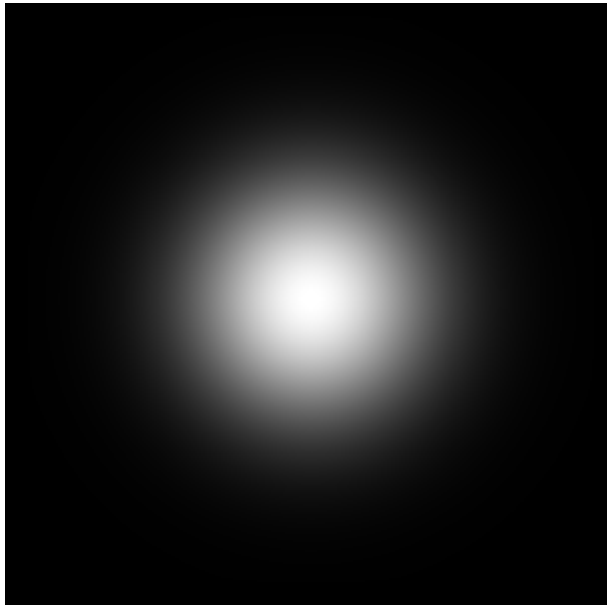


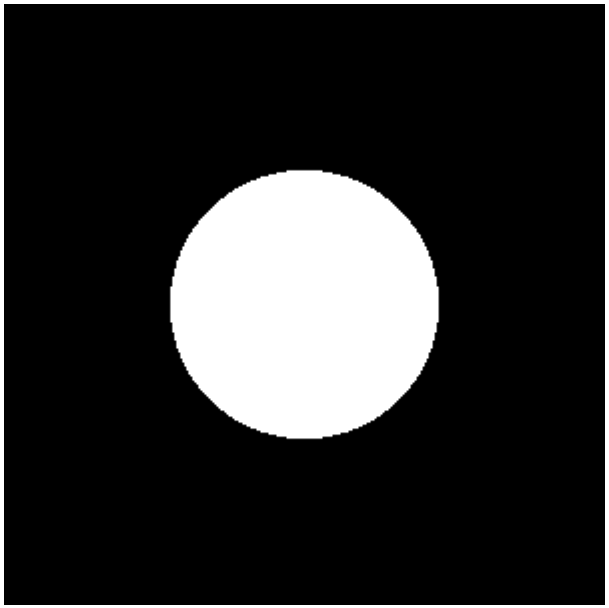
Circular Depth of Field



Instead of modelling the circle of confusion with this Gaussian blur kernel ...

Circular Depth of Field

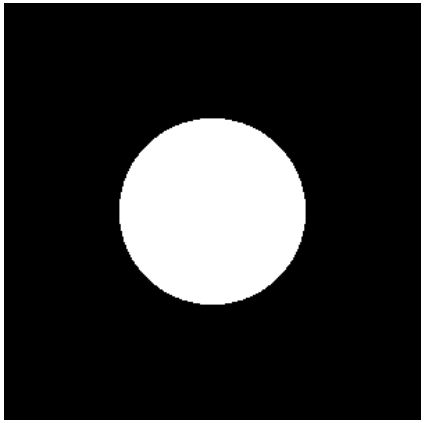




Instead of modelling the circle of confusion with this Gaussian blur kernel ...

… it should rather be like this.

Circular Depth of Field



0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

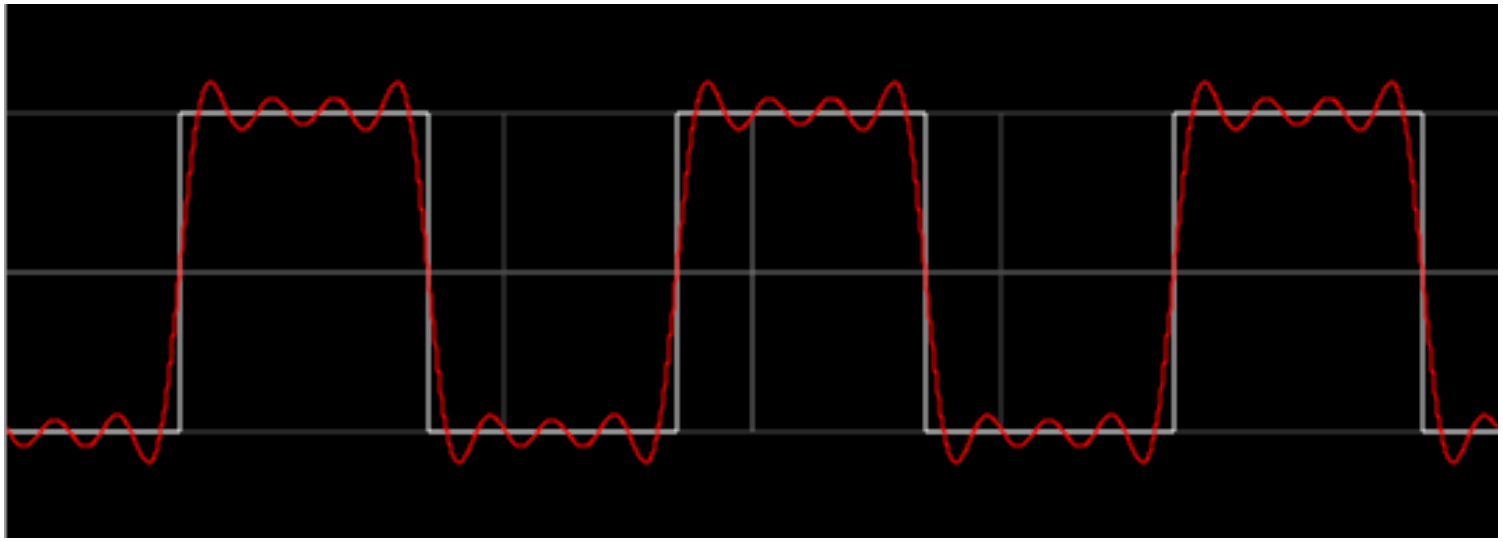
This is not of rank 1.

Thus not separable.

When the cross-section increases, there are new affected dimensions.

Circular Depth of Field

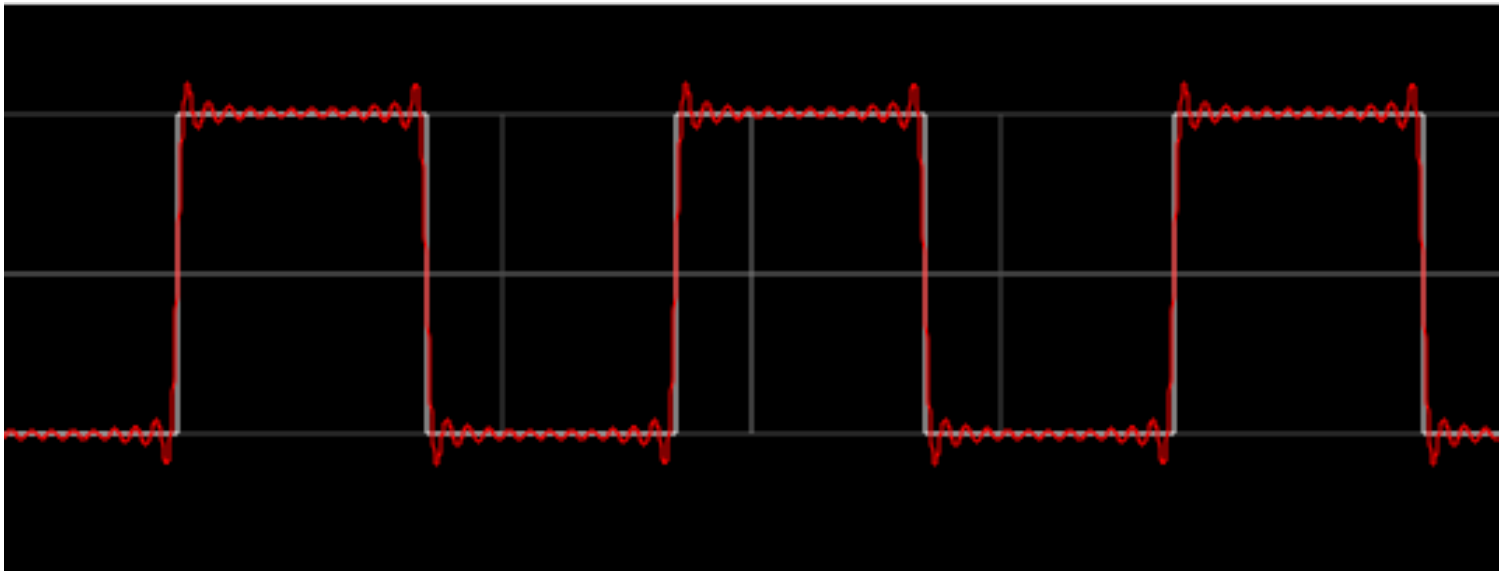
- Different shapes can be approximated via a combination of sinusoids (sines and cosines).



http://www.falstad.com/fourier/

Circular Depth of Field

- Different shapes can be approximated via a combination of sinusoids (sines and cosines).
- More waves (components) means a better approximation.



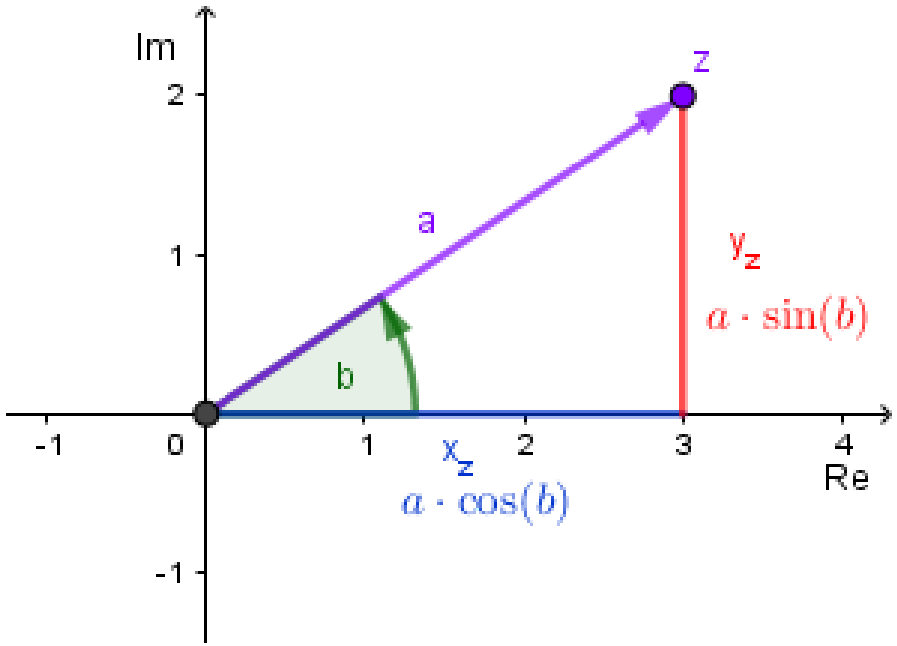
Circular Depth of Field

- Complex numbers

$$z = x_z + i y_z$$

$$z = a \cdot \left(\cos(b) + i \sin(b) \right)$$

$$z = a \cdot e^{ib}$$ (via Euler's formula)



Circular Depth of Field

- Complex numbers

We have some cosines and sines here

$$z = a \cdot \left(\cos\left(b\right) + i \sin\left(b\right) \right)$$

Circular Depth of Field

- Complex numbers

This function outputs complex numbers

$$z = a \cdot \left(\cos\left(b\right) + i \sin\left(b\right) \right)$$

$$z\left(x\right) = \underbrace{a}_{\text{Amplitude}} \cdot \left(\cos\left(\underbrace{bx}_{\text{Frequency}}\right) + i \sin\left(\underbrace{bx}_{\text{Frequency}}\right) \right)$$

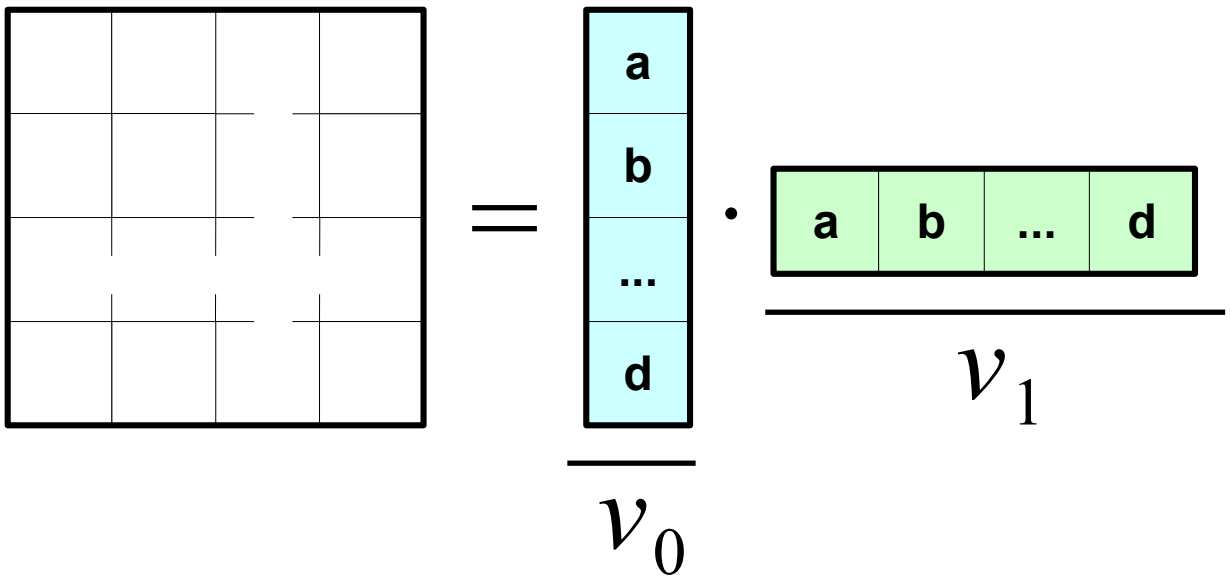
Circular Depth of Field

- We are looking for conditions for a **circularly symmetric separable kernel**...

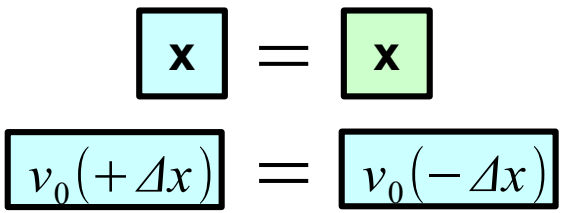
  ... to see if we could create one out of complex numbers (because they include sinusoids)

Circular Depth of Field

- We are looking for conditions for a **circularly symmetric separable kernel**.



When the kernel is **circularly symmetric**:

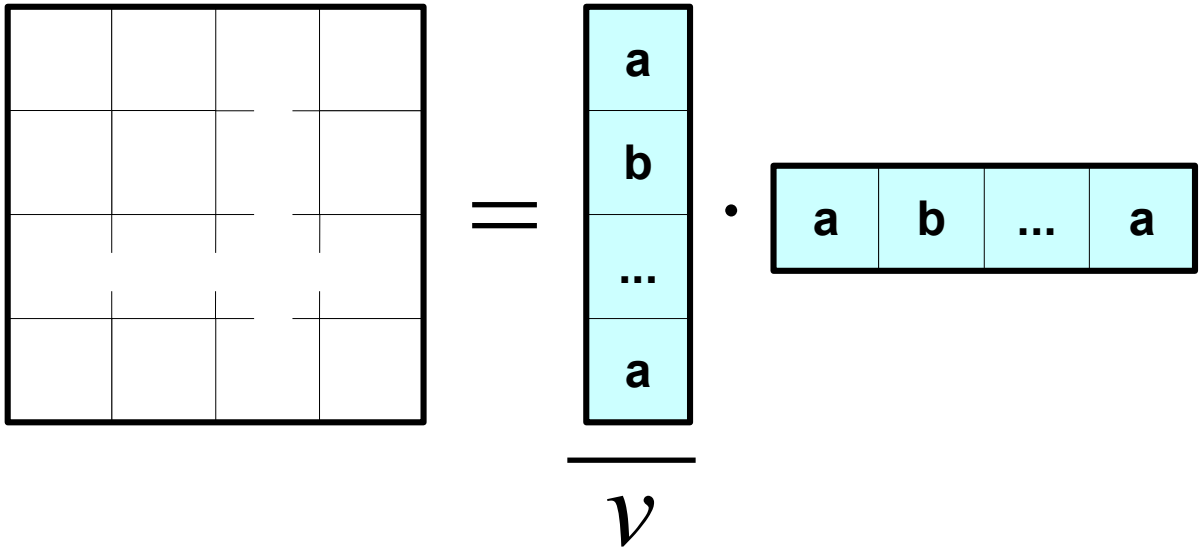


$$K = K^{\mathrm{T}}$$

$$K = v_0 \cdot v_1 = v_0 \cdot v_0^{\mathrm{T}}$$

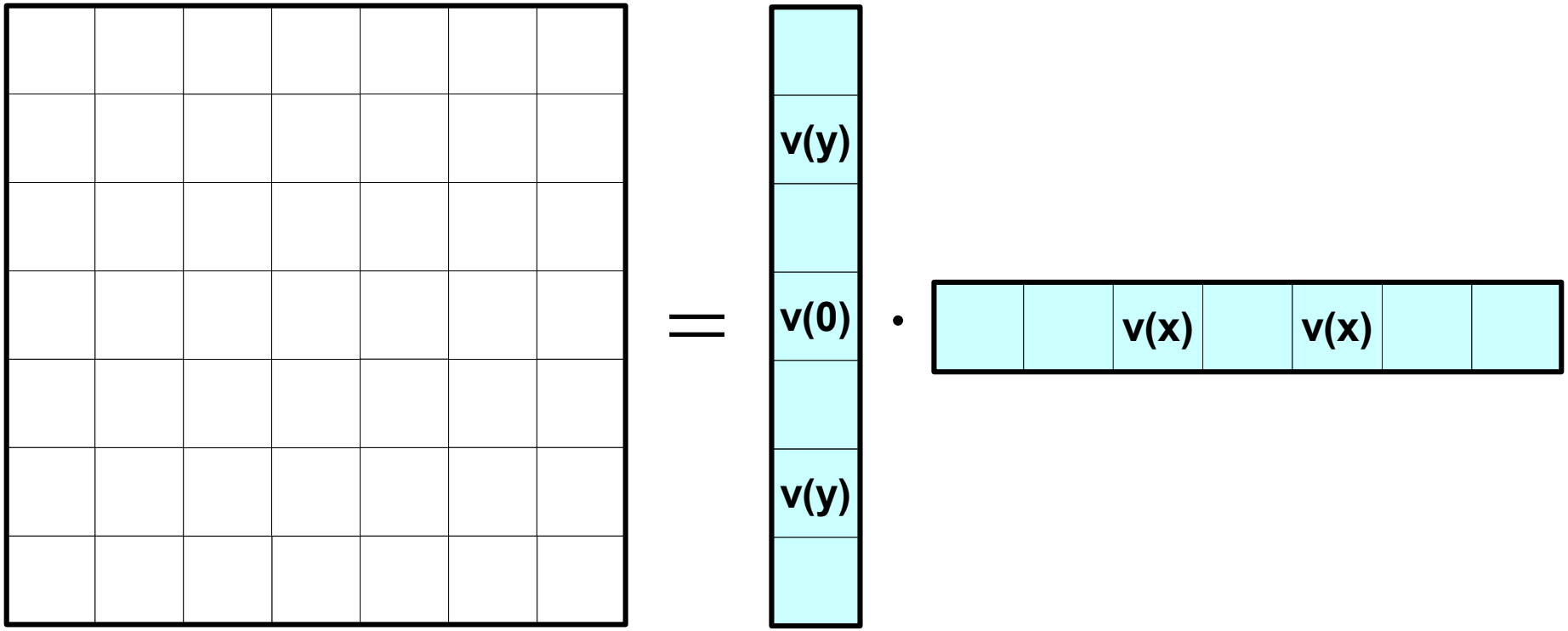
Circular Depth of Field

- We are looking for conditions for a **circularly symmetric separable kernel**.



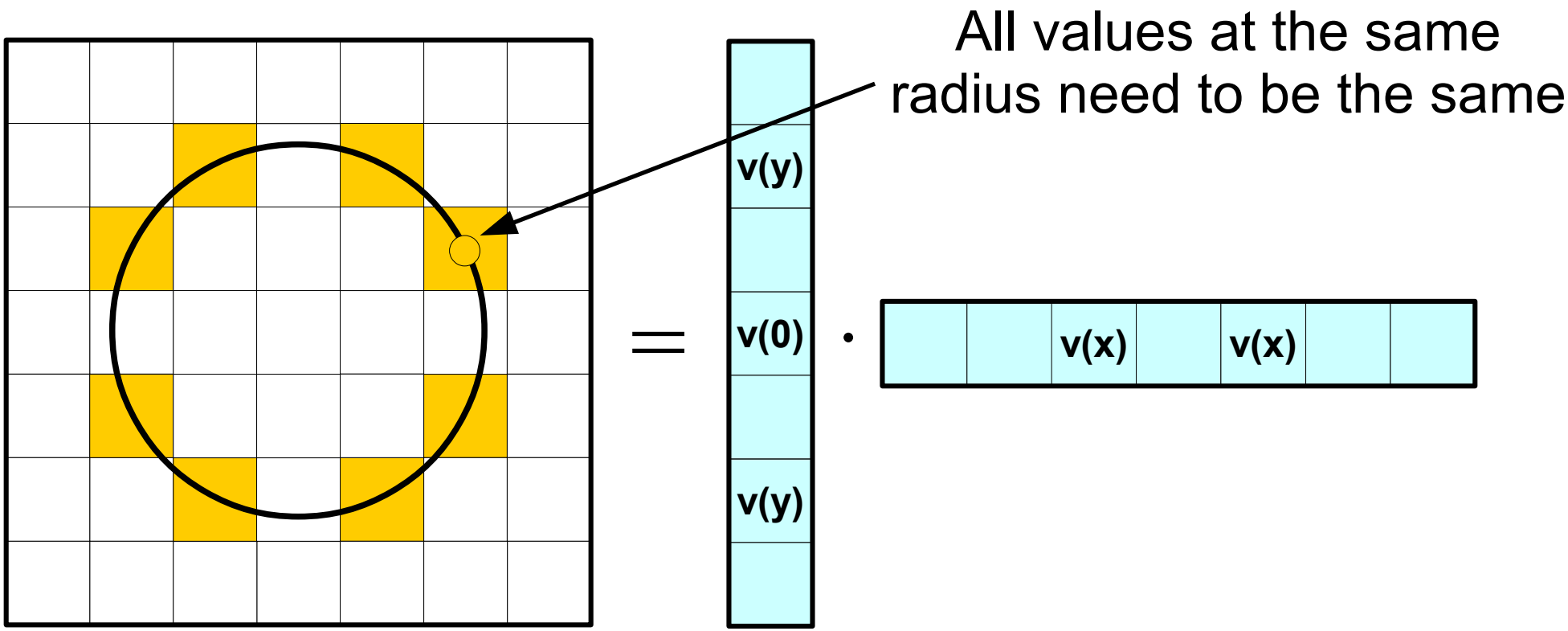
Circular Depth of Field

- We are looking for conditions for a **circularly symmetric separable kernel**.



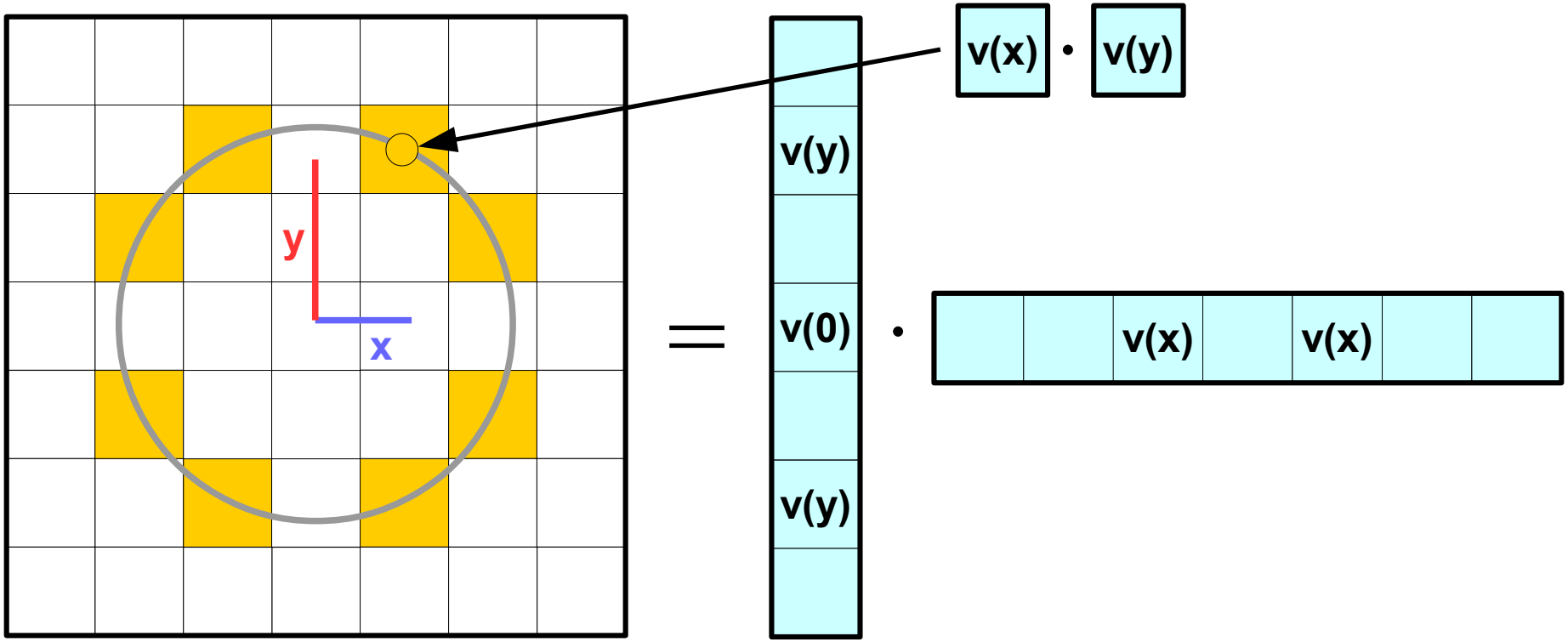
Circular Depth of Field

- We are looking for conditions for a **circularly symmetric separable kernel**.



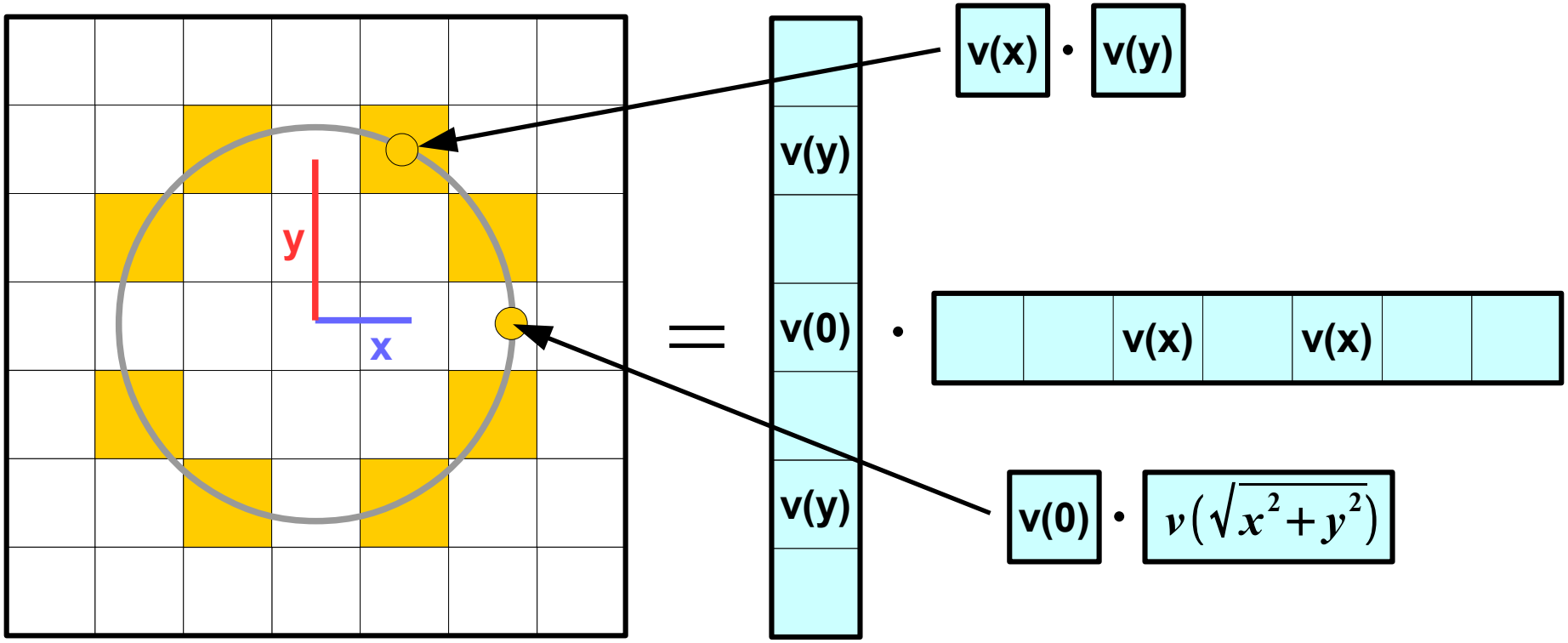
Circular Depth of Field

- We are looking for conditions for a **circularly symmetric separable kernel**.



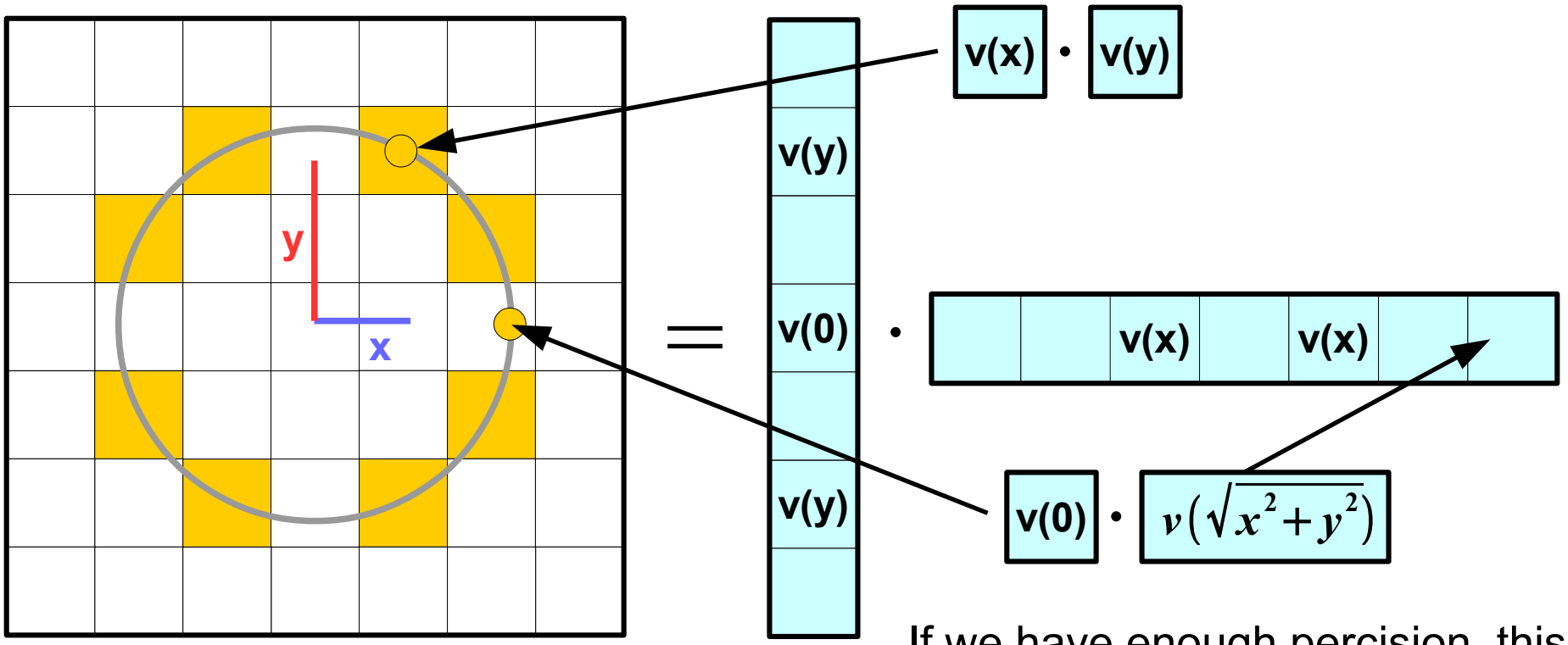
Circular Depth of Field

- We are looking for conditions for a **circularly symmetric separable kernel**.



Circular Depth of Field

- We are looking for conditions for a **circularly symmetric separable kernel**.



If we have enough percision, this value is somewhere on the 1D kernel.

Circular Depth of Field

- Condition for a **circularly symmetric separable kernel:**

$$\boxed{v(x)} \cdot \boxed{v(y)} = \boxed{v(0)} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

(constant) (value at the radius)

Circular Depth of Field

- Condition for a **circularly symmetric separable kernel:**

$$\boxed{v(x)} \cdot \boxed{v(y)} = \boxed{v(0)} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

Checking the Gaussian blur kernel.

Circular Depth of Field

- Condition for a **circularly symmetric separable kernel:**

$$\boxed{v(x)} \cdot \boxed{v(y)} = \boxed{v(0)} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

$$g(x) = \alpha \cdot e^{-a x^2}$$

(simplification)

Circular Depth of Field

- Condition for a **circularly symmetric separable kernel:**

$$\boxed{v(x)} \cdot \boxed{v(y)} = \boxed{v(0)} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

$$\alpha \cdot e^{-a\,x^2} \cdot \alpha \cdot e^{-a\,y^2} = \alpha \cdot e^{0} \cdot \alpha \cdot e^{-a\left(\sqrt{x^2+y^2}\right)^2}$$

Circular Depth of Field

- Condition for a **circularly symmetric separable kernel:**

$$\boxed{v(x)} \cdot \boxed{v(y)} = \boxed{v(0)} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

$$\alpha \cdot e^{-a\,x^2} \cdot \alpha \cdot e^{-a\,y^2} = \alpha \cdot e^{0} \cdot \alpha \cdot e^{-a\left(\sqrt{x^2+y^2}\right)^2}$$

$$\alpha \cdot e^{-a\,x^2} \cdot \alpha \cdot e^{-a\,y^2} = \alpha \cdot \alpha \cdot e^{-a\left(x^2+y^2\right)}$$

Circular Depth of Field

- Condition for a **circularly symmetric separable kernel:**

$$\boxed{v(x)} \cdot \boxed{v(y)} = \boxed{v(0)} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

$$\alpha \cdot e^{-a x^2} \cdot \alpha \cdot e^{-a y^2} = \alpha \cdot e^{0} \cdot \alpha \cdot e^{-a\left(\sqrt{x^2+y^2}\right)^2}$$

$$\alpha \cdot e^{-a x^2} \cdot \alpha \cdot e^{-a y^2} = \alpha \cdot \alpha \cdot e^{-a\left(x^2+y^2\right)}$$

$$\alpha \cdot e^{-a x^2} \cdot \alpha \cdot e^{-a y^2} = \alpha^2 \cdot e^{-a\left(x^2+y^2\right)}$$

Satisfies the condition

Circular Depth of Field

- What about our function with complex numbers?

$$z(x) = a \cdot (\cos(bx) + i \sin(bx))$$

$$\boxed{\text{v(x)}} \cdot \boxed{\text{v(y)}} = \boxed{\text{v(0)}} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

Circular Depth of Field

- What about our function with complex numbers?

$$z(x) = a \cdot (\cos(bx) + i \sin(bx))$$

$$z(x) = a\, e^{ibx}$$

$$\boxed{\text{v(x)}} \cdot \boxed{\text{v(y)}} = \boxed{\text{v(0)}} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

Circular Depth of Field

- What about our function with complex numbers?

$$z(x) = a \cdot (\cos(bx) + i \sin(bx))$$

$$z(x) = a\, e^{ibx}$$

$$a\, e^{ibx} \cdot a\, e^{iby} = a\, e^{0} \cdot a\, e^{ib(x+y)}$$

$$\boxed{\text{v(x)}} \cdot \boxed{\text{v(y)}} = \boxed{\text{v(0)}} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

Circular Depth of Field

- What about our function with complex numbers?

$$z(x) = a \cdot (\cos(bx) + i \sin(bx))$$

$$z(x) = a\, e^{ibx}$$

$$a\, e^{ibx} \cdot a\, e^{iby} = a\, e^{0} \cdot a\, e^{ib\underline{(x+y)}}$$

$$\neq a\, e^{0} \cdot a\, e^{ib\underline{(\sqrt{(x^2+y^2)})}}$$

Does not satisfy the condition

$$\boxed{\text{v(x)}} \cdot \boxed{\text{v(y)}} = \boxed{\text{v(0)}} \cdot \boxed{v(\sqrt{x^2+y^2})}$$

Circular Depth of Field

- What about our function with complex numbers?

$$z(x) = a \cdot \left(\cos\left(bx^2\right) + i \sin\left(bx^2\right)\right)$$

$$z(x) = a\, e^{ibx^2}$$

$$a\, e^{ibx^2} \cdot a\, e^{iby^2} = a\, e^{0} \cdot a\, e^{ib\left(x^2+y^2\right)}$$

$$= a\, e^{0} \cdot a\, e^{ib\left(\sqrt{\left(x^2+y^2\right)}\right)^2}$$

Satisfies the condition

$$\boxed{\text{v(x)}} \cdot \boxed{\text{v(y)}} = \boxed{\text{v(0)}} \cdot \boxed{v\left(\sqrt{x^2+y^2}\right)}$$

Circular Depth of Field

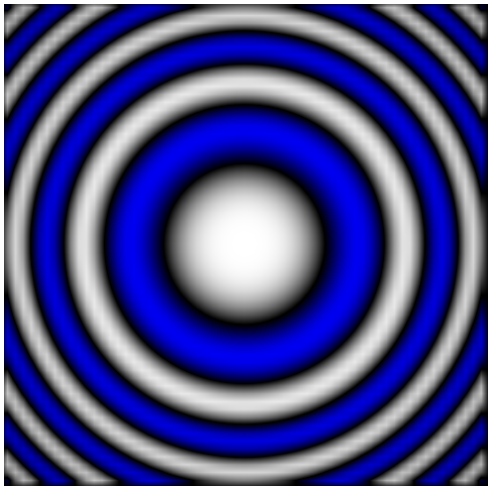
$$z(x) = a \cdot \left(\cos\left(bx^2\right) + i \sin\left(bx^2\right)\right)$$

1D

$$z(x) = a \cdot e^{ibx^2}$$

2D

$$K(x, y) = a^2 \cdot e^{ib\left(x^2 + y^2\right)}$$



Real part.

Values scaled. Blue is negative.

Circular Depth of Field

$$z(x) = a \cdot \left(\cos\left(bx^2\right) + i \sin\left(bx^2\right)\right)$$

2D

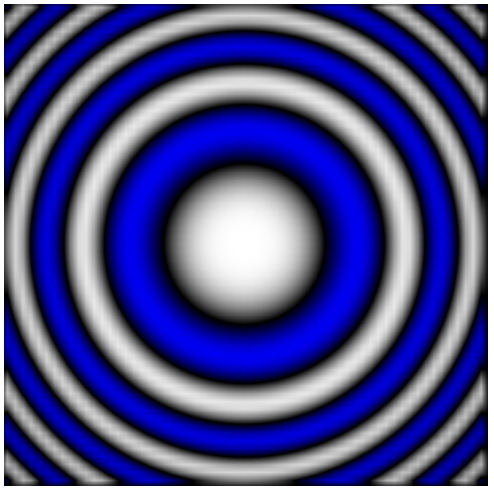
$$K(x, y) = a^2 \cdot e^{ib\left(x^2 + y^2\right)}$$

$$\underline{B} = \underline{K} * \underline{P}$$

Blurred picture (real)

Kernel (cmplx)

Input picture (real)



Real part.

When we do the 2D convolution, only the real part matters.

Circular Depth of Field

$$z(x) = a \cdot \left(\underline{\cos\left(bx^2\right)} + i \sin\left(bx^2\right) \right)$$

We are currently only using the cosine wave (real part).

2D

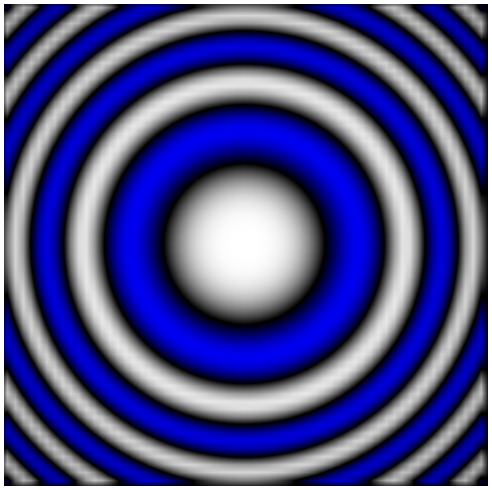
$$K(x, y) = a^2 \cdot e^{ib\left(x^2 + y^2\right)}$$

$$\underline{B} = \underline{K} * \underline{P}$$

Blurred picture (real)

Kernel (cmplx)

Input picture (real)



Circular Depth of Field

$$z(x) = \underline{a} \cdot \left(\cos\left(bx^2\right) + i \sin\left(bx^2\right)\right)$$

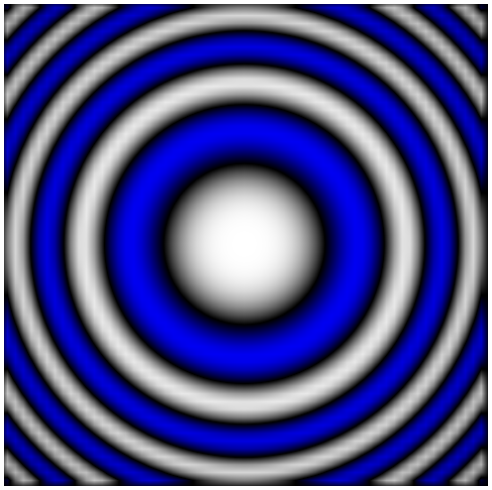
$$a = (c - id)$$

2D

$$K(x, y) = \underline{a}^2 \cdot e^{ib\left(x^2 + y^2\right)}$$

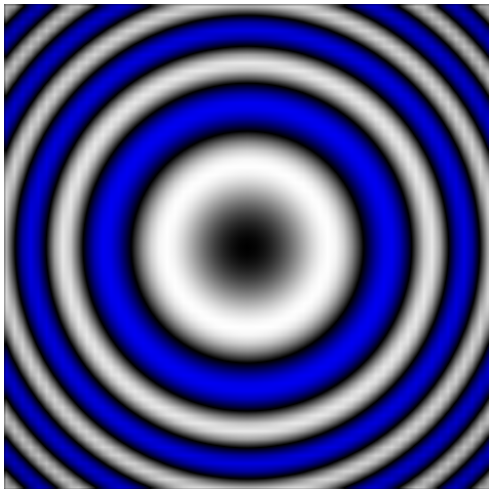
Multiplication of two complex numbers is still a complex number.

cos



+

sin



Circular Depth of Field

$$z(x)=(c-id)\cdot(\cos(bx^2)+i\sin(bx^2))=$$

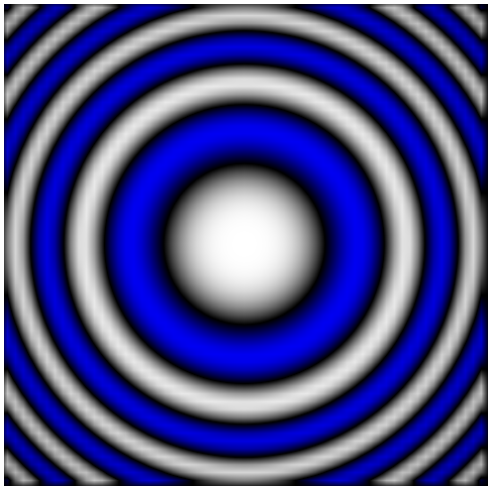
$$=c\cos(bx^2)+d\sin(bx^2)+i(c\sin(bx^2)-d\cos(bx^2))$$

Now we have both a cosine and a sine in the real part.

This is also important for a separated kernel.

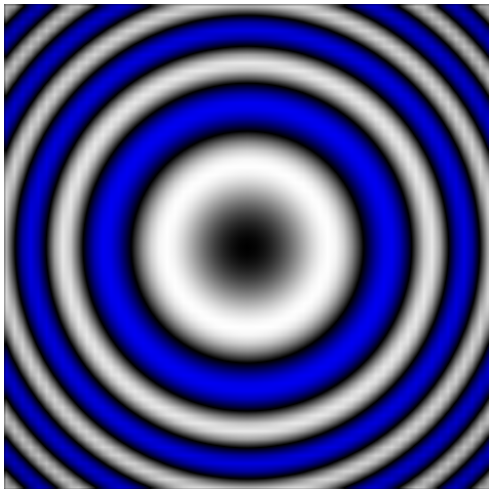
$$K(x,y)=(c-id)^2\cdot e^{ib(x^2+y^2)}$$

cos



+

sin

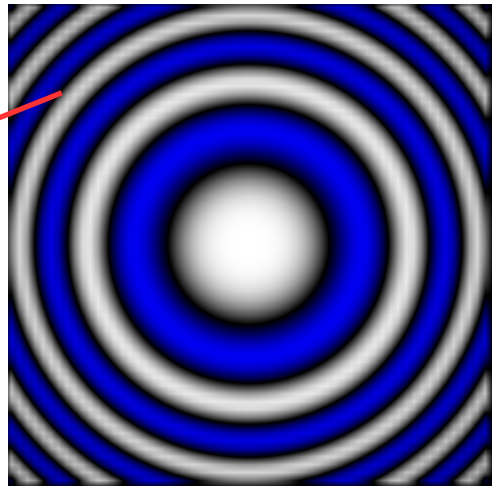


Circular Depth of Field

$$z(x)=(c-id)\cdot(\cos(bx^2)+i\sin(bx^2))$$

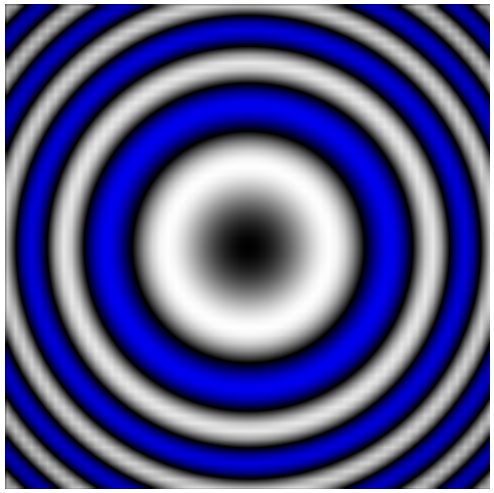
$$K(x,y)=(c-id)^2\cdot e^{ib(x^2+y^2)}$$

cos



+

sin



We want a bound circle instead of a wavy pattern.

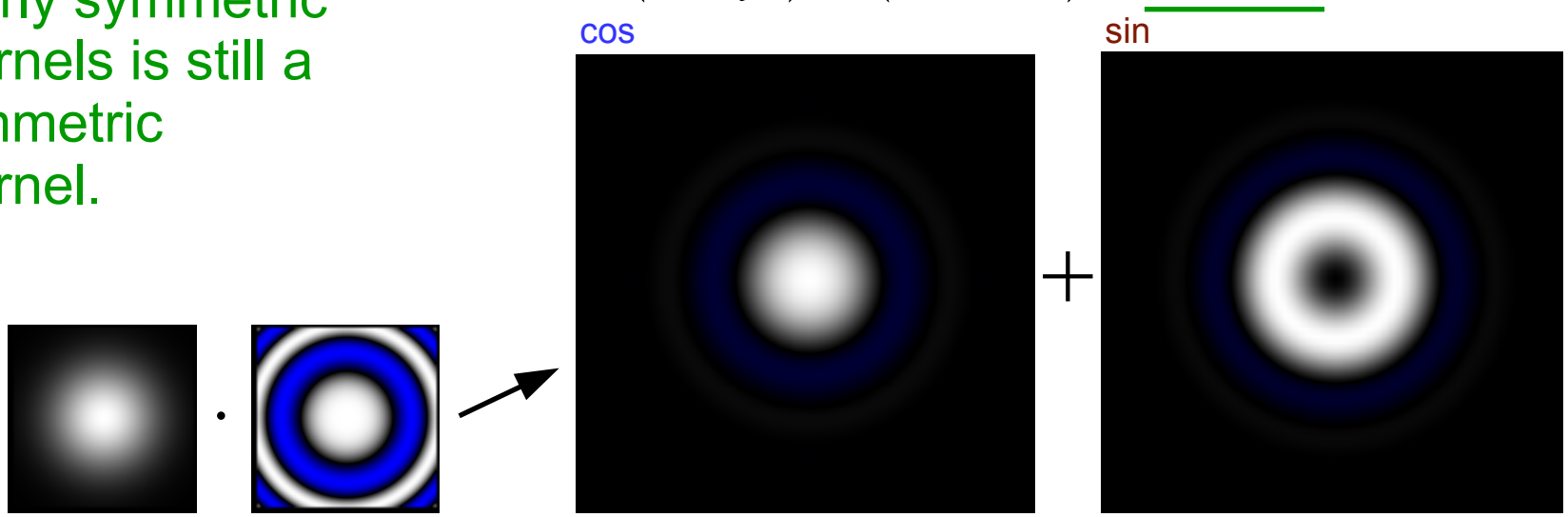
Circular Depth of Field

$$z(x)=(c-id)\cdot e^{ax^2}\cdot\left(\cos\left(bx^2\right)+i\sin\left(bx^2\right)\right)$$

Multiply the kernel with a Gaussian *envelope*.

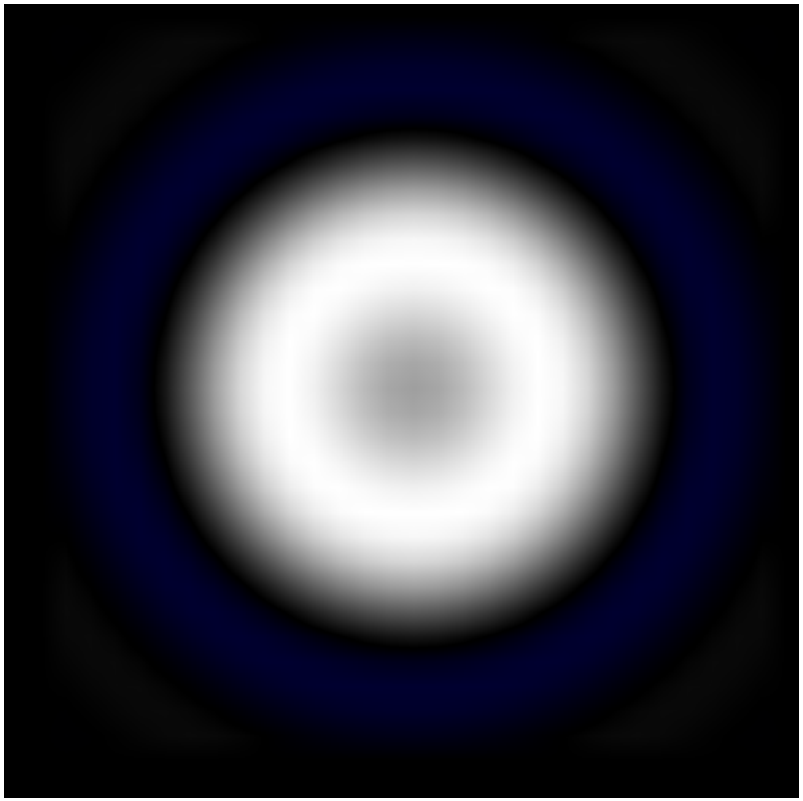
Element-wise multiplication of two circularly symmetric separable kernels is still a circularly symmetric separable kernel.

$$K(x,y)=(c-id)^2\cdot e^{a\left(x^2+y^2\right)}\cdot e^{ib\left(x^2+y^2\right)}$$



Circular Depth of Field

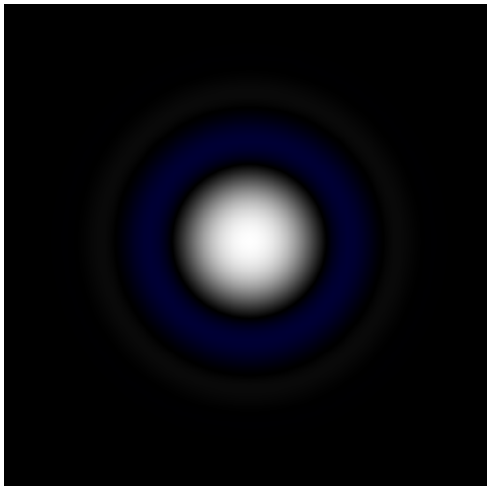
$$z(x)=(c-id)\cdot e^{ax^2}\cdot\left(\cos\left(bx^2\right)+i\sin\left(bx^2\right)\right)$$



$$K(x,y)=(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}$$

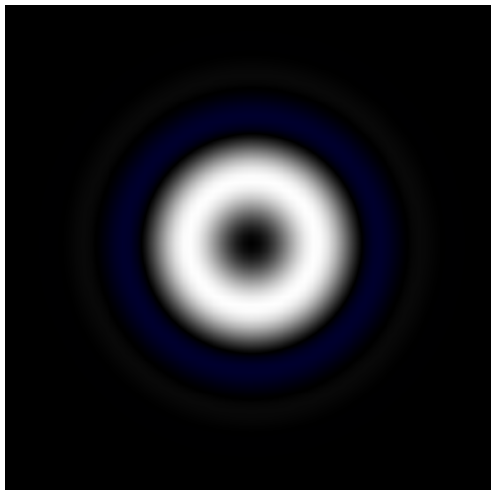
$=$

cos



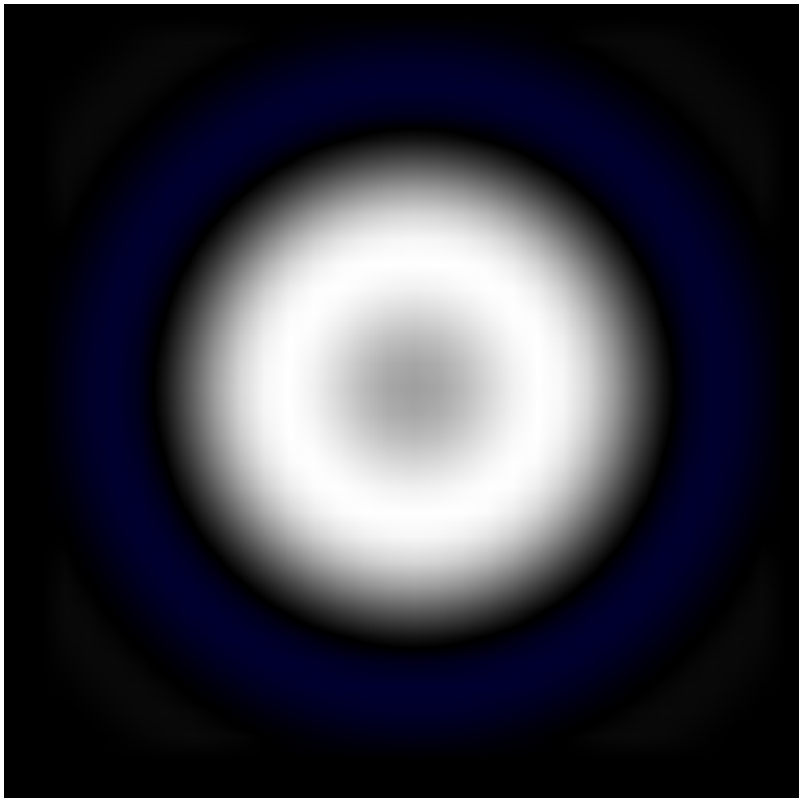
$+$

sin



Circular Depth of Field

$$z(x)=(c-id)\cdot e^{ax^2}\cdot\left(\cos\left(bx^2\right)+i\sin\left(bx^2\right)\right)$$

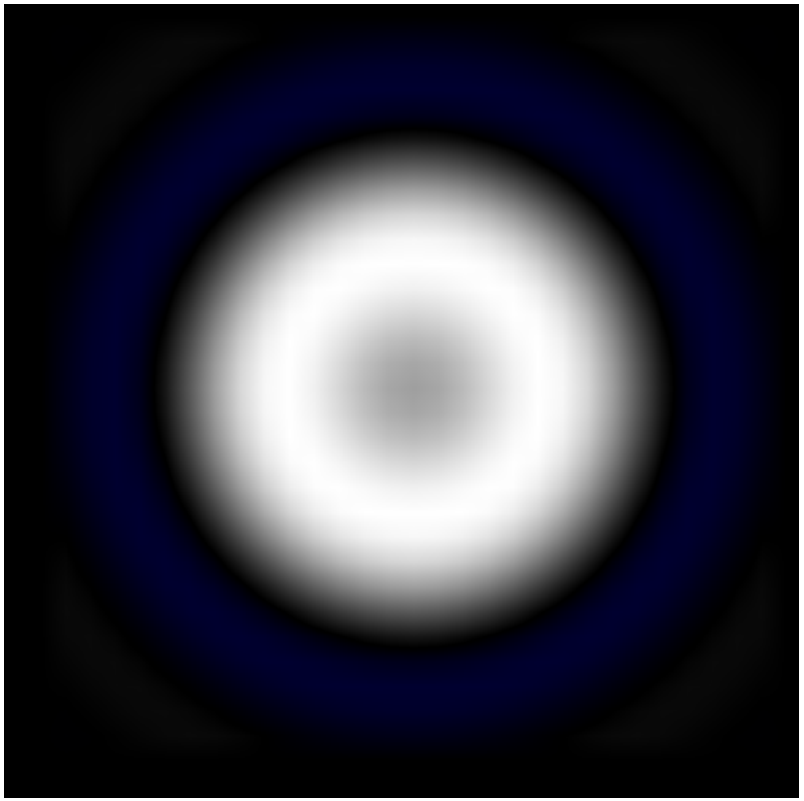


- Parameters:
  - a ~ std of the Gaussian

$$K(x,y)=(c-id)^2\cdot e^{a\left(x^2+y^2\right)}\cdot e^{ib\left(x^2+y^2\right)}$$

Circular Depth of Field

$$z(x)=(c-id)\cdot e^{ax^2}\cdot\left(\cos\left(\underline{b}x^2\right)+i\sin\left(\underline{b}x^2\right)\right)$$

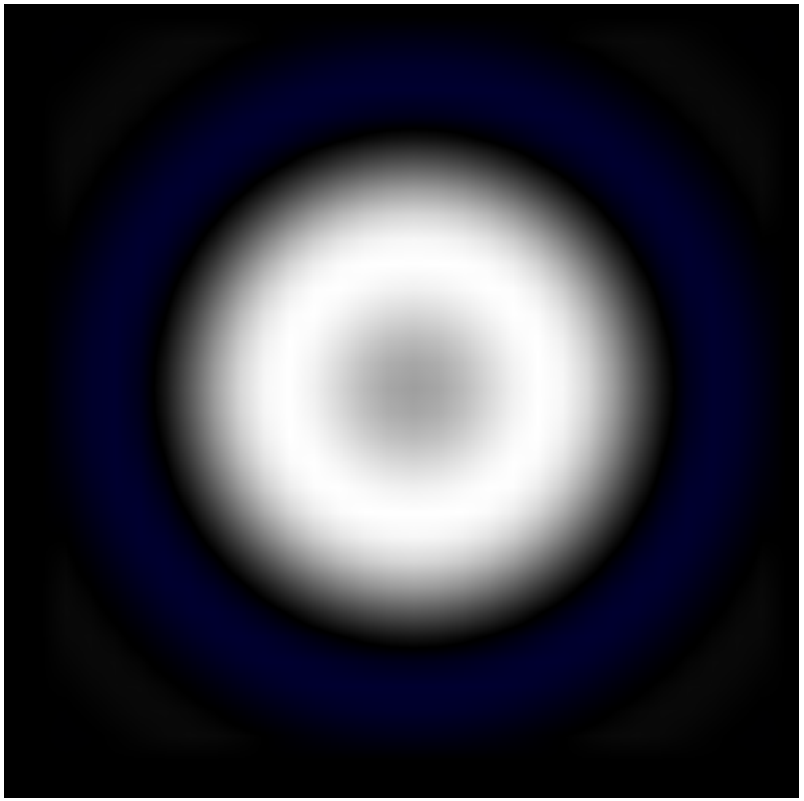


- Parameters:
  - a ~ std of the Gaussian
  - b = frequency of the waves

$$K(x,y)=(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{\underline{ib}(x^2+y^2)}$$

Circular Depth of Field

$$z(x)=(\underline{c}-id)\cdot e^{ax^2}\cdot(\cos(bx^2)+i\sin(bx^2))$$

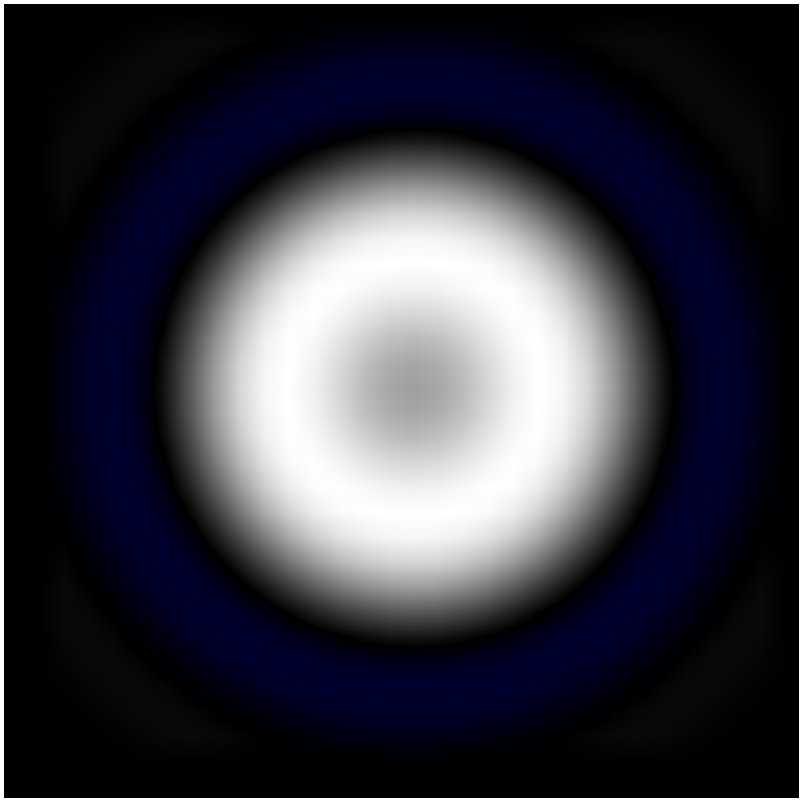


- Parameters:
  - a ~ std of the Gaussian
  - b = frequency of the waves
  - c = contribution of cos() in 1D

$$K(x,y)=(\underline{c}-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}$$

Circular Depth of Field

$$z(x)=(c-id)\cdot e^{ax^2}\cdot\left(\cos\left(bx^2\right)+i\sin\left(bx^2\right)\right)$$



- Parameters:
  - a ~ std of the Gaussian
  - b = frequency of the waves
  - c = contribution of cos() in 1D
  - d = contribution of sin() in 1D

$$K(x,y)=(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}$$

Circular Depth of Field

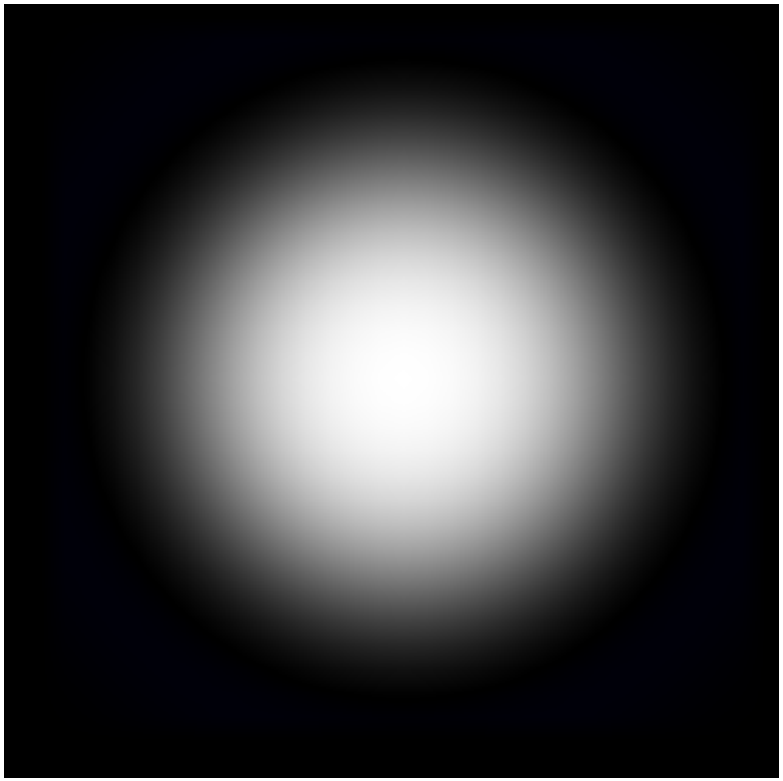
- Too broad falloff, almost like Gaussian blur.

$$a = -1$$

$$b = 0.9$$

$$c = 0.4$$

$$d = 0.2$$



Circular Depth of Field

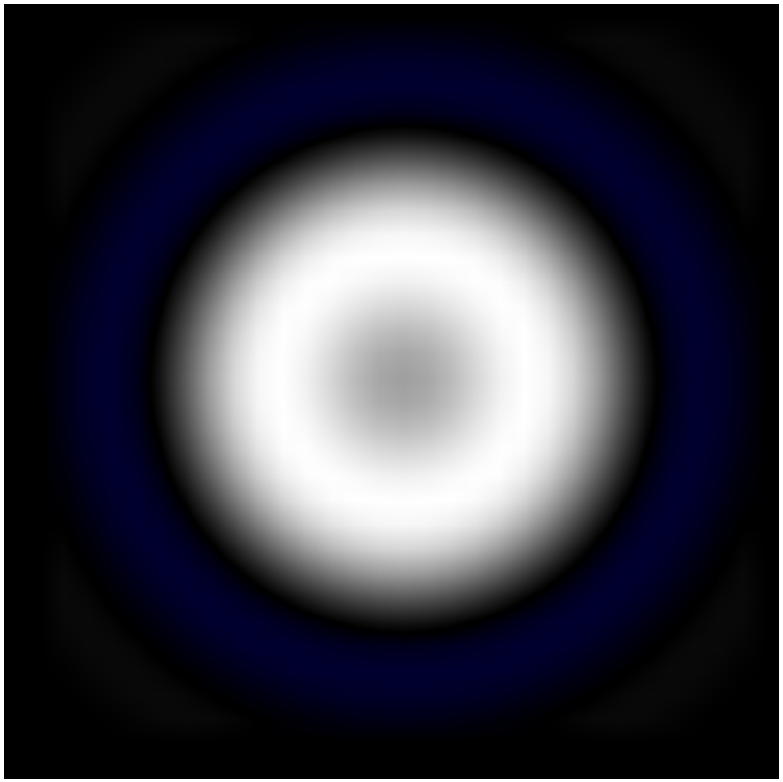
- Sharper falloff, but negative values.
- Negative values create a sharpening effect.

$$a = -0.86$$

$$b = 1.62$$

$$c = 1.18$$

$$d = 0.79$$



Circular Depth of Field

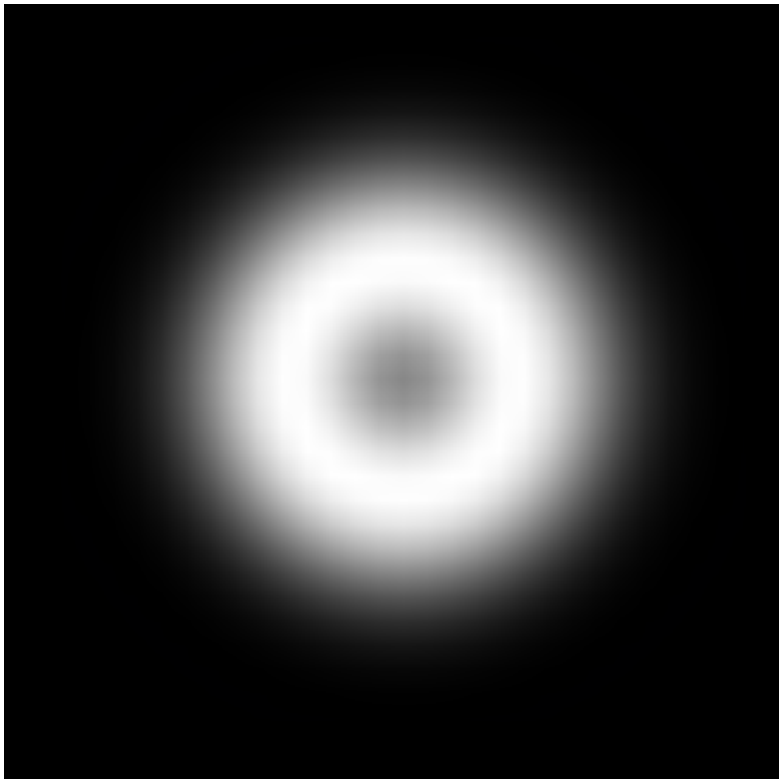
- No negative values, but broader falloff again
- Creates a more doughnut shaped bokeh.

$$a = -2.01$$

$$b = 1.05$$

$$c = 1.78$$

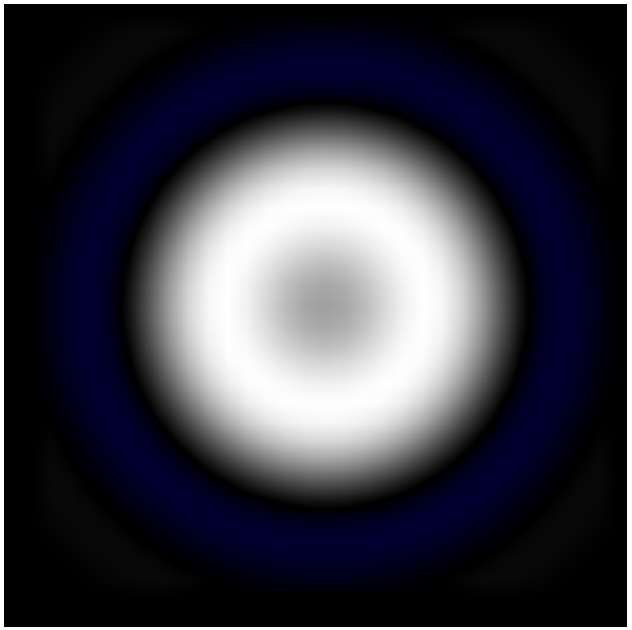
$$d = 1.57$$



Circular Depth of Field

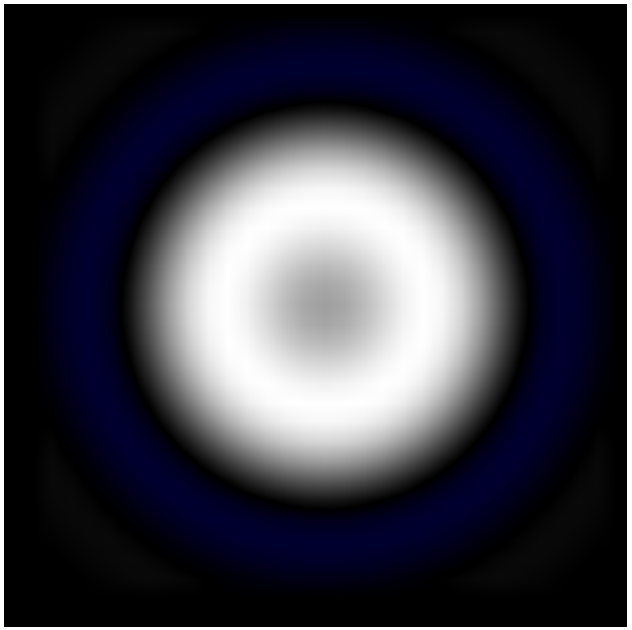
- Normalization:
  - Differently from the Gaussian kernel, this kernel does not have its values sum up to 1.

The sum here is 214



Circular Depth of Field

- Normalization:
  - Differently from the Gaussian kernel, this kernel does not have its values sum up to 1.
  - That will cause the image to become too bright.



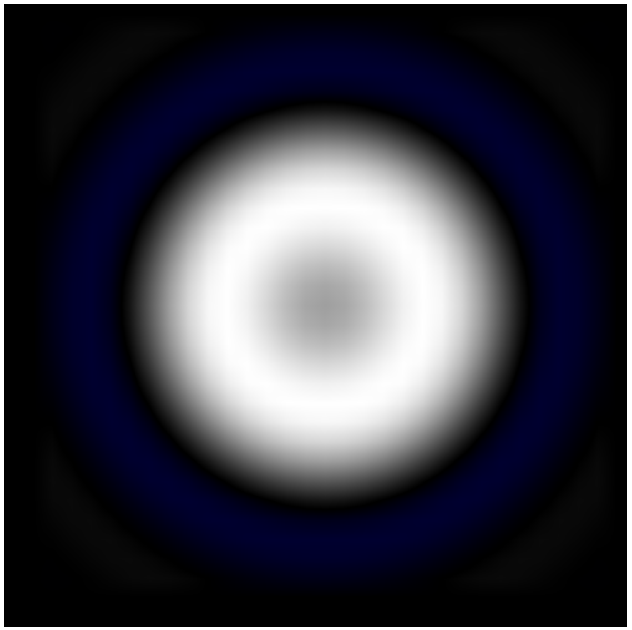
The sum here is 214

Circular Depth of Field

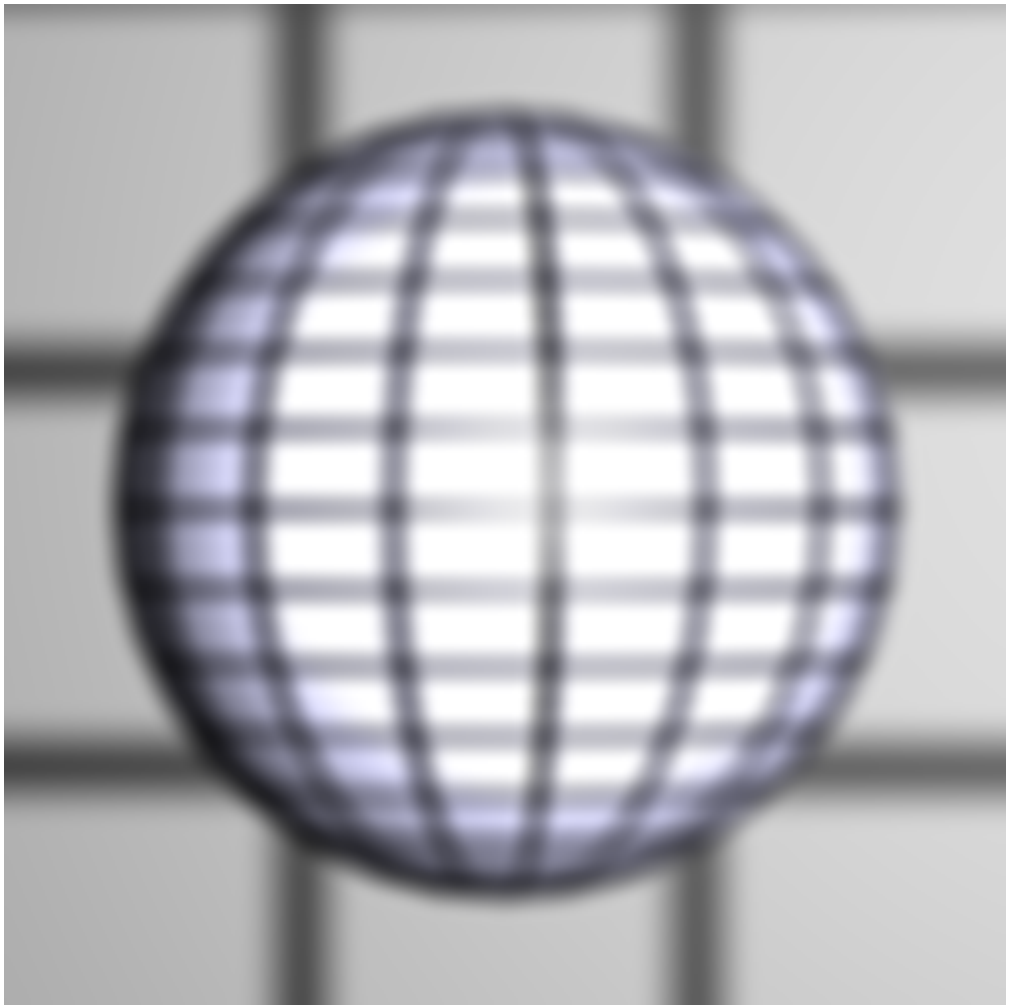
- Normalization:
  - Differently from the Gaussian kernel, this kernel does not have its values sum up to 1.
  - That will cause the image to become too bright.
  - Values must be normalized.

$$\bar{K}(x, y) = \frac{K(x, y)}{\left| \sum_{x', y'} K(x', y') \right|}$$

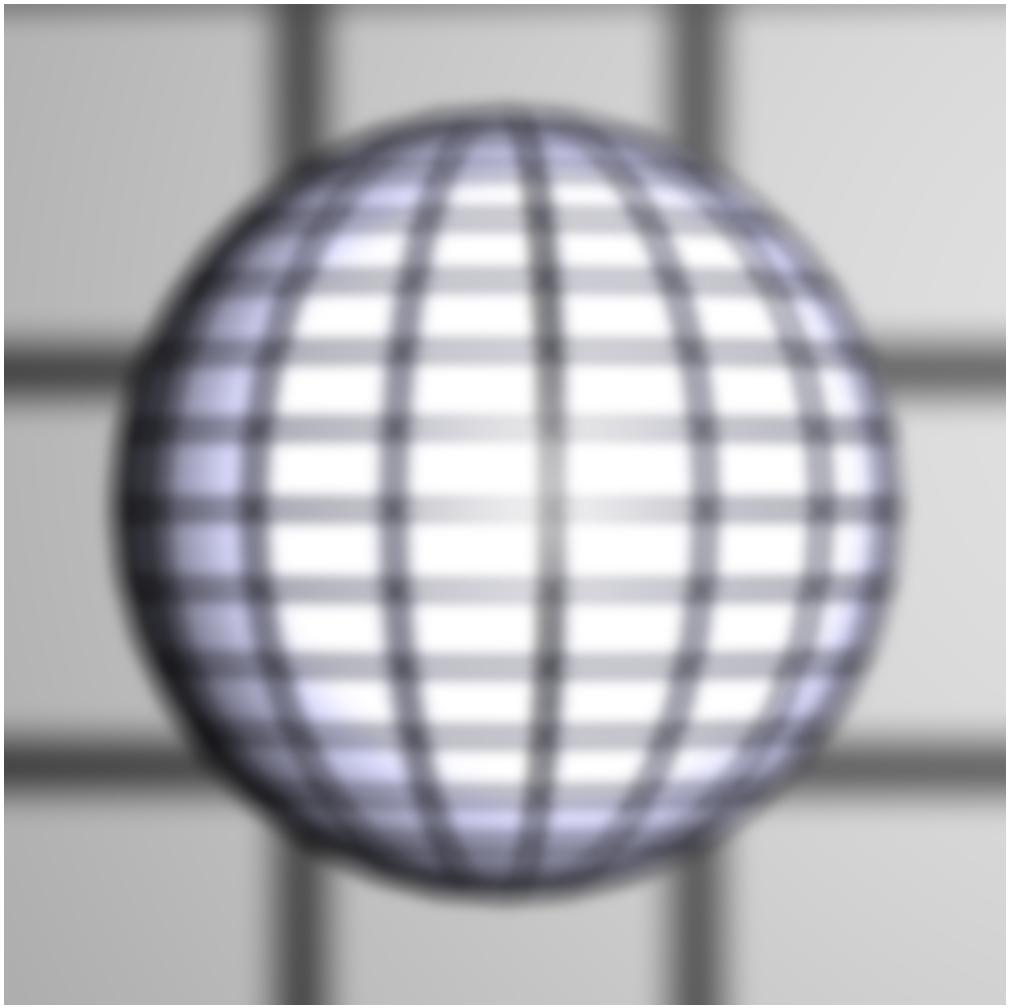
The sum here is 214

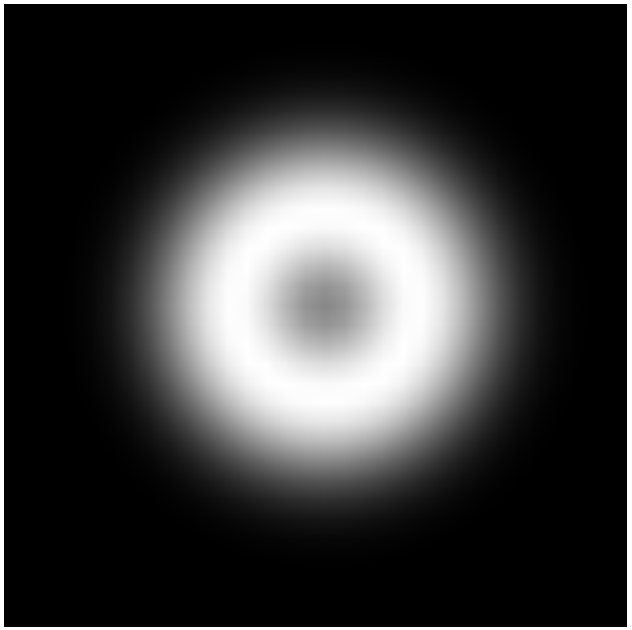


Circular Depth of Field

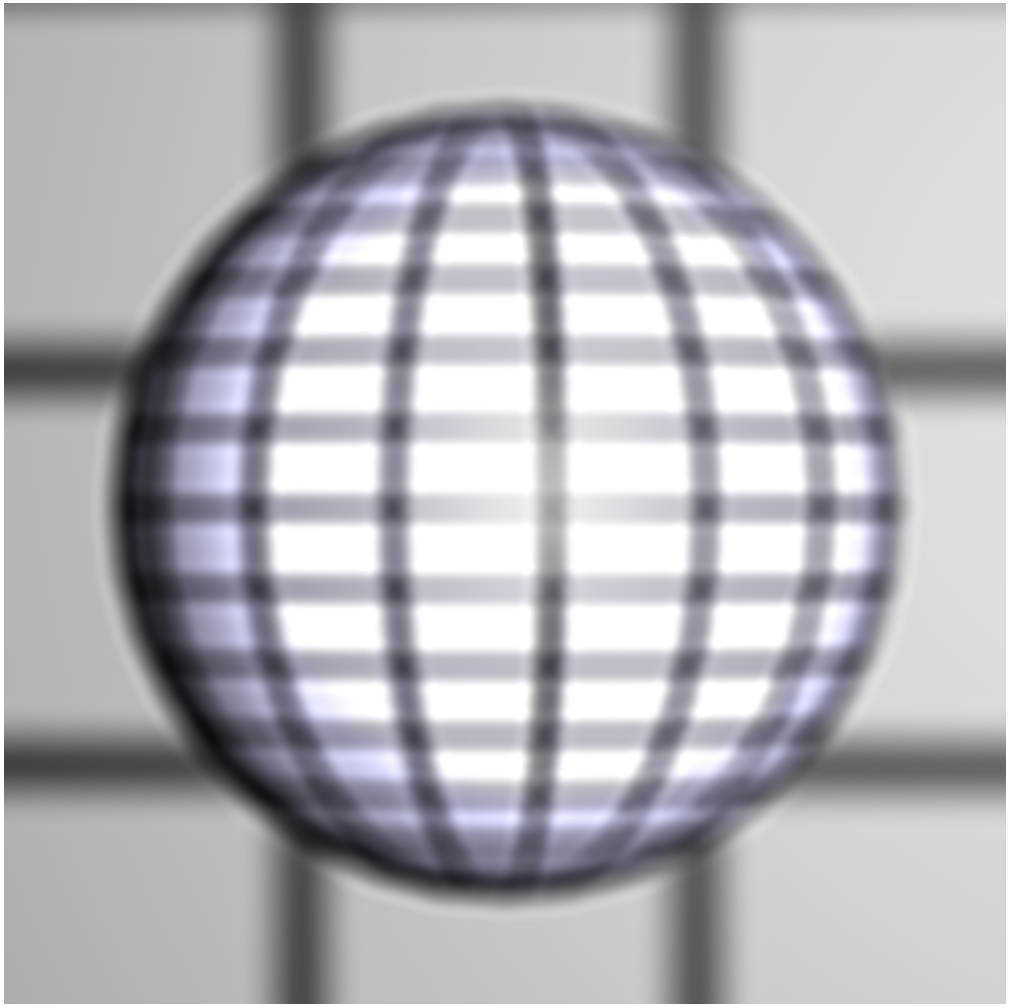


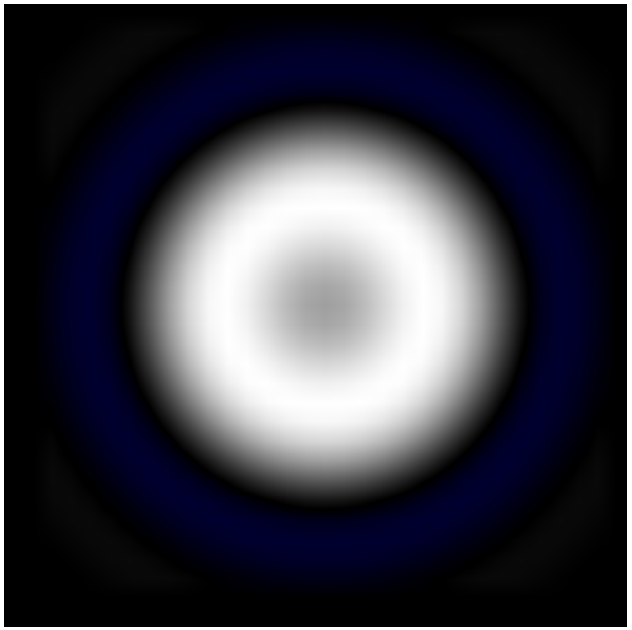
Circular Depth of Field





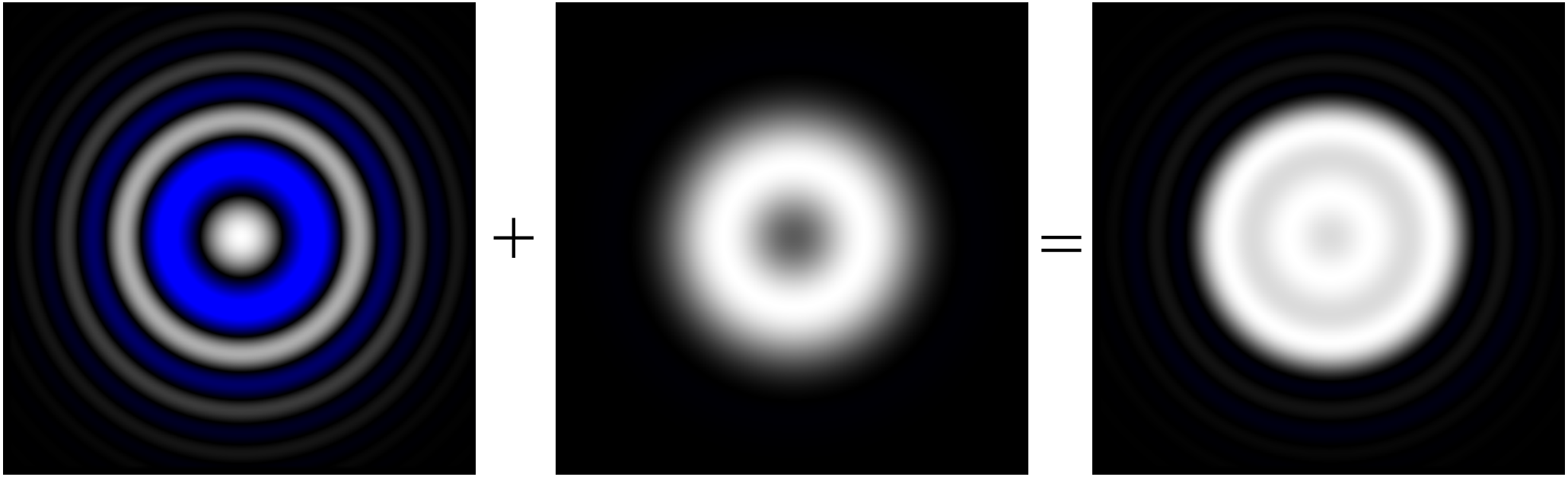
Circular Depth of Field





Circular Depth of Field

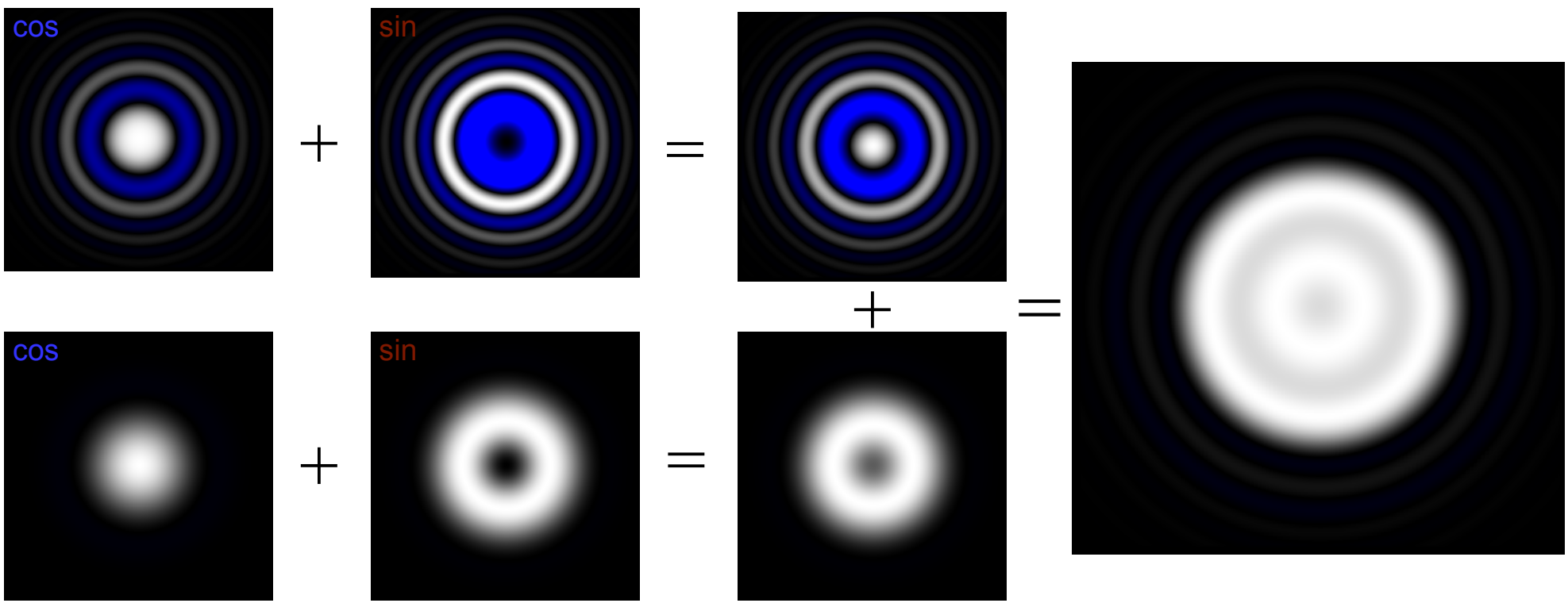
- Components:
  - We can create a better *circle* if we use multiple complex-numbered kernels (components).



Circular Depth of Field

- We can create a better *circle* if we use multiple complex-numbered kernels (components).

$$K_0(x,y)=(c_0-id_0)^2\cdot e^{a_0(x^2+y^2)}\cdot e^{ib_0(x^2+y^2)} \qquad (a_0, b_0, c_0, d_0)$$

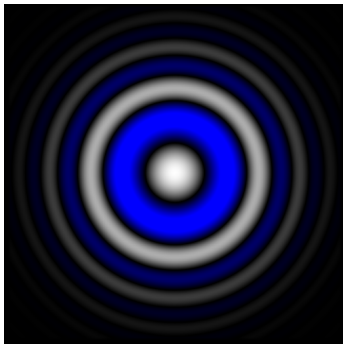


$$K_1(x,y)=(c_1-id_1)^2\cdot e^{a_1(x^2+y^2)}\cdot e^{ib_1(x^2+y^2)} \qquad (a_1, b_1, c_1, d_1)$$

Circular Depth of Field

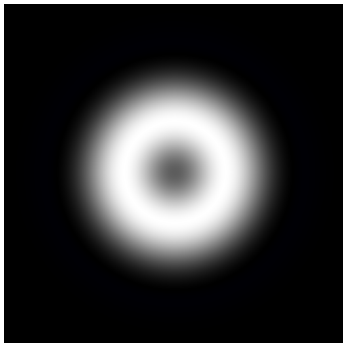
- We can create a better *circle* if we use multiple complex-numbered kernels (components).
- Sum of separable kernels is not separable.

Separable $K_0(x, y) =$



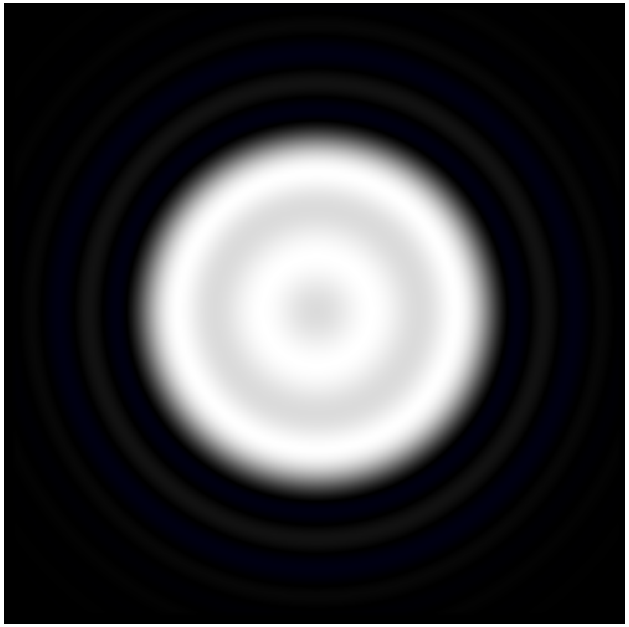
$+$

Separable $K_1(x, y) =$



$=$

Not separable



Circular Depth of Field

- We can create a better *circle* if we use multiple complex-numbered kernels (components).
- Sum of separable kernels is not separable.
- Convolution is distributive:

$$K_0 * P + K_1 * P = (K_0 + K_1) * P$$

Circular Depth of Field

- We can create a better *circle* if we use multiple complex-numbered kernels (components).
- Sum of separable kernels is not separable.
- Convolution is distributive:

$$K_0 * P + K_1 * P = (K_0 + K_1) * P$$

- But if we separate the kernels:

$$(K_0 + K_1) * P = (v_0 * v_0^{\mathrm{T}} + v_1 * v_1^{\mathrm{T}}) * P$$

Circular Depth of Field

- We can create a better *circle* if we use multiple complex-numbered kernels (components).
- Sum of separable kernels is not separable.
- Convolution is distributive.
- But if we separate the kernels:

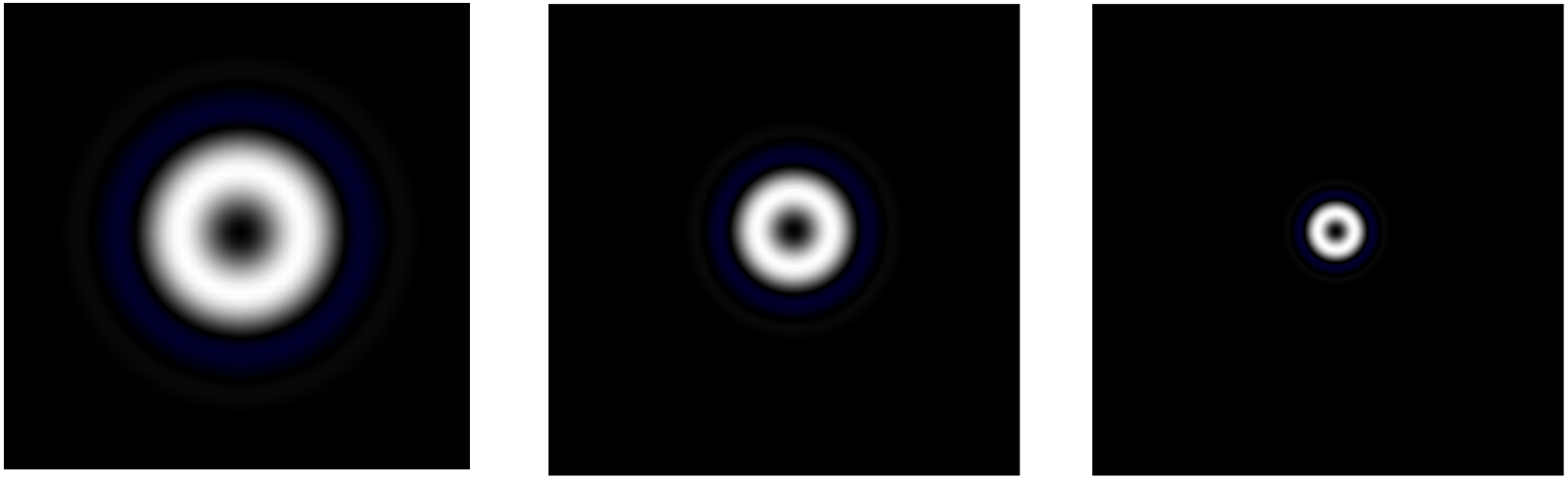
$$\left(K_0 + K_1\right) * P = \left(v_0 * v_0^{\mathrm{T}} + v_1 * v_1^{\mathrm{T}}\right) * P$$

- We see that we need to perfrom 2 × 1D chains of (2×) convolution and add the results:

$$\left(v_0 * v_0^{\mathrm{T}} + v_1 * v_1^{\mathrm{T}}\right) * P = \underbrace{v_0 * v_0^{\mathrm{T}} * P}_{\text{2×1D conv}} + \underbrace{v_1 * v_1^{\mathrm{T}} * P}_{\text{2×1D conv}}$$

Circular Depth of Field

- Scaling
  - Instead of generating multiple kernels, mixing between them and normalizing in the shader...



Circular Depth of Field

- Scaling
  - Instead of generating multiple kernels, mixing between them and normalizing in the shader...
  - ... we can change the sampling offset scale instead.

$$(K * P)(x, y) =$$

$$= \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(\underline{s}\Delta x, \underline{s}\Delta y) \cdot P(x+\Delta x, y+\Delta y)$$

Circular Depth of Field

- Scaling
  - Instead of generating multiple kernels, mixing between them and normalizing in the shader...
  - ... we can change the sampling offset scale instead.

$$(K * P)(x, y) =$$

$$= \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(s\Delta x, s\Delta y) \cdot P(x+\Delta x, y+\Delta y)$$

Changing it for the kernel would require renormalization of the result...

Circular Depth of Field

- Scaling
  - Instead of generating multiple kernels, mixing between them and normalizing in the shader...
  - ... we can change the sampling offset scale instead.

$$(K * P)(x, y) =$$

$$= \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(\Delta x, \Delta y) \cdot P(x + \underline{s}\Delta x, y + \underline{s}\Delta y)$$

So, instead change it for the sampled picture.

Circular Depth of Field

- Scaling
  - Instead of generating multiple kernels, mixing between them and normalizing in the shader...
  - ... we can change the sampling offset scale instead.

$$(K * P)(x, y) =$$

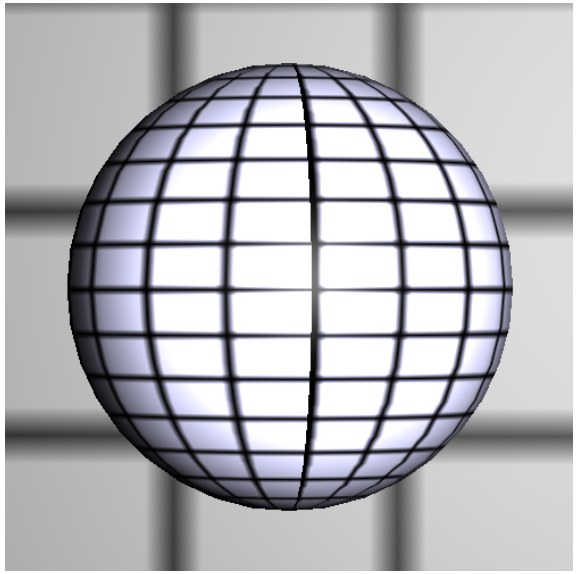
$$= \sum_{\Delta x=-r}^{\Delta x=r} \sum_{\Delta y=-r}^{\Delta y=r} K(\Delta x, \Delta y) \cdot P(x + s\Delta x, y + s\Delta y)$$

Note that we can not change the kernel radius, because loops in shaders need to have fixed conditions.

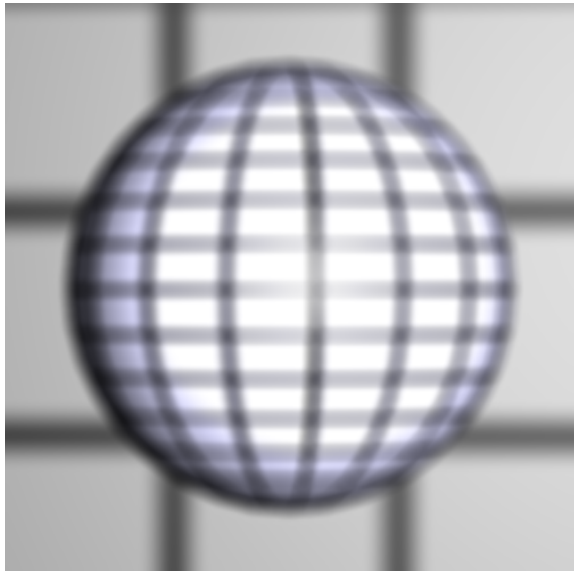
Kernel: 32×32
Output: 500×500
Components: 2

Circular Depth of Field

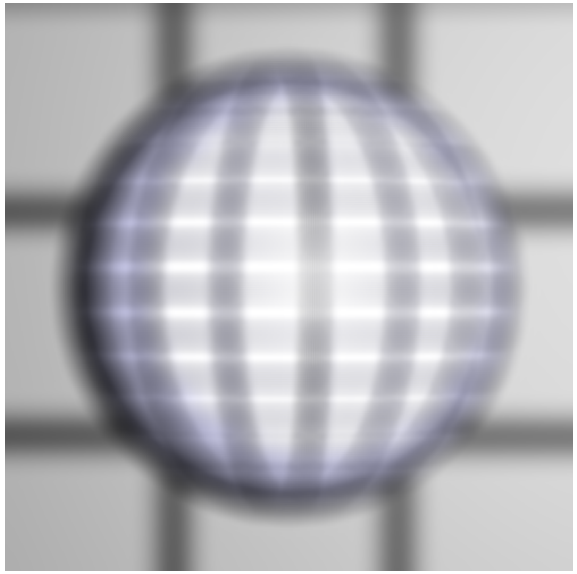
$s=0$



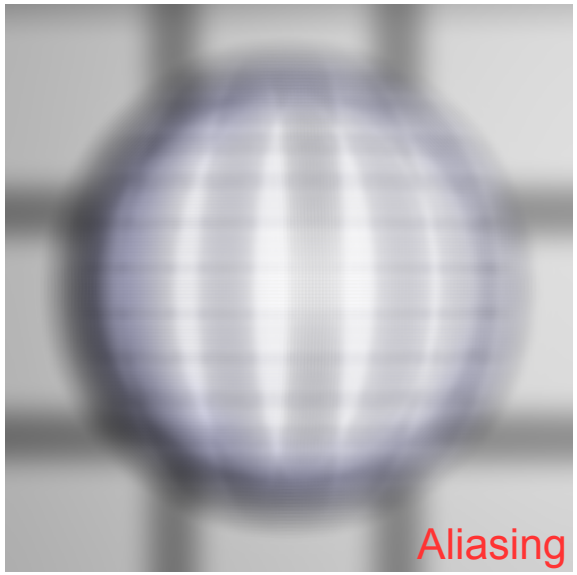
$s=1$



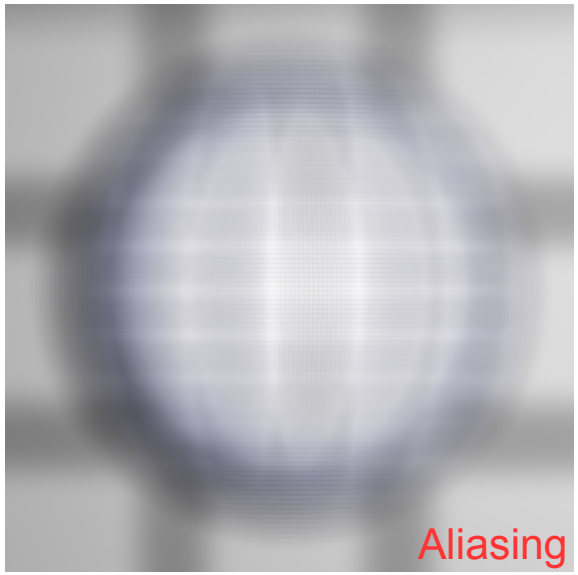
$s=2$



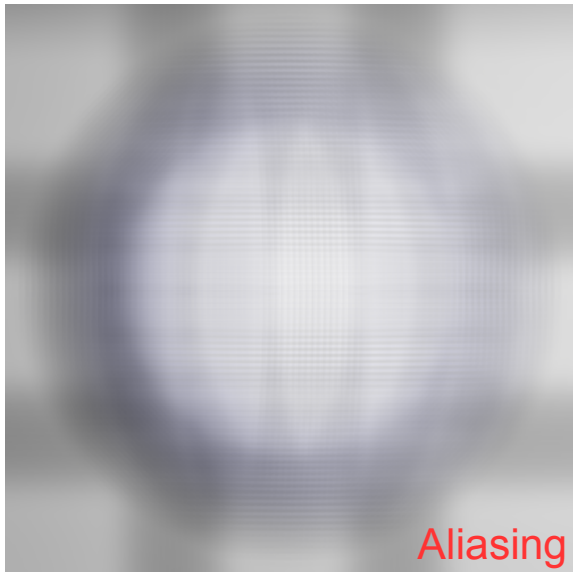
$s=3$



$s=4$



$s=5$



Circular Depth of Field

- Separated kernel

$$K(x, y) = (c - id)^2 \cdot e^{a(x^2+y^2)} \cdot e^{ib(x^2+y^2)}$$

What happens in the calculations with a separated kernel.

Circular Depth of Field

- Separated kernel

$$K(x, y) = \Re[(c - id)^2 \cdot e^{a(x^2+y^2)} \cdot e^{ib(x^2+y^2)}]$$

With a 2D kernel we only used the real part of the values, because both our input and output were reals.

$$\underline{B} = \underline{K} * \underline{P}$$

Blurred picture (real)

Kernel (cmplx)

Input picture (real)

Circular Depth of Field

- Separated kernel

$$K(x,y)=\Re[(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}]=$$

$$=\Re[\underbrace{(c-id)\cdot e^{a(x^2)}\cdot e^{ib(x^2)}}_{\text{1D kernel}}\cdot\underbrace{(c-id)\cdot e^{a(y^2)}\cdot e^{ib(y^2)}}_{\text{1D kernel}}]$$

Circular Depth of Field

- Separated kernel

$$K(x,y)=\Re[(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}]=$$

$$=\Re[\underbrace{(c-id)\cdot e^{a(x^2)}\cdot e^{ib(x^2)}}_{\text{1D kernel}}\ \cdot\ \underbrace{(c-id)\cdot e^{a(y^2)}\cdot e^{ib(y^2)}}_{\text{1D kernel}}]$$

Complex number multiplication

Circular Depth of Field

- Separated kernel

$$K(x,y)=\Re[(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}]=$$

$$=\Re[\underline{(c-id)\cdot e^{a(x^2)}\cdot e^{ib(x^2)}}\ \cdot\ (c-id)\cdot e^{a(y^2)}\cdot e^{ib(y^2)}]$$

$$e^{ax^2}\cdot(c-id)\cdot(\cos(bx^2)+i\sin(bx^2))$$

Circular Depth of Field

- Separated kernel

$$K(x,y)=\Re[(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}]=$$

$$=\Re[\underline{(c-id)\cdot e^{a(x^2)}\cdot e^{ib(x^2)}}\ \cdot\ (c-id)\cdot e^{a(y^2)}\cdot e^{ib(y^2)}]$$

$$e^{ax^2}\cdot(c-id)\cdot(\cos(bx^2)+i\sin(bx^2))=$$

$$=e^{ax^2}\cdot(c\cos(bx^2)+d\sin(bx^2)+i(c\sin(bx^2)-d\cos(bx^2)))$$

Circular Depth of Field

- Separated kernel

$$K(x,y)=\Re[(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}]=$$

$$=\Re[\underline{(c-id)\cdot e^{a(x^2)}\cdot e^{ib(x^2)}}\ \cdot\ (c-id)\cdot e^{a(y^2)}\cdot e^{ib(y^2)}]$$

$$e^{ax^2}\cdot(c-id)\cdot(\cos(bx^2)+i\sin(bx^2))=$$

$$=e^{ax^2}\cdot(c\cos(bx^2)+d\sin(bx^2)+i(c\sin(bx^2)-d\cos(bx^2)))=$$

$$=\underline{e^{ax^2}\cdot(c\cos(bx^2)+d\sin(bx^2))}+i\,\underline{e^{ax^2}\cdot(c\sin(bx^2)-d\cos(bx^2))}$$

1D kernel **real** part

1D kernel **imaginary** part

Circular Depth of Field

- Separated kernel

$$K(x,y)=\Re[(c-id)^2\cdot e^{a(x^2+y^2)}\cdot e^{ib(x^2+y^2)}]=$$

$$=\Re[\underbrace{(c-id)\cdot e^{a(x^2)}\cdot e^{ib(x^2)}}_{V}\cdot(c-id)\cdot e^{a(y^2)}\cdot e^{ib(y^2)}]$$

1D kernel (**complex**)

$$e^{ax^2}\cdot(c-id)\cdot(\cos(bx^2)+i\sin(bx^2))=$$

$$=e^{ax^2}\cdot(c\cos(bx^2)+d\sin(bx^2)+i(c\sin(bx^2)-d\cos(bx^2)))=$$

$$=\underbrace{e^{ax^2}\cdot(c\cos(bx^2)+d\sin(bx^2))}_{R}+i\,\underbrace{e^{ax^2}\cdot(c\sin(bx^2)-d\cos(bx^2))}_{I}$$

1D kernel **real** part

1D kernel **imaginary** part

Circular Depth of Field

- Separated kernel

$$K * P = V * V^{\mathrm{T}} * P$$

$$K * P = \left(R + iI \right) * \left(R + iI \right)^{\mathrm{T}} * P$$

$$\underline{K} * P = \left(\underline{R} + \underline{iI} \right) * \left(\underline{R} + \underline{iI} \right) * P$$

Circular Depth of Field

- Separated kernel

$$K * P = (R + iI) * (R + iI) * P$$

$$R * P \rightarrow P'_R \searrow$$

$$P'$$

(complex numbers)

$$I * P \rightarrow P'_I \nearrow$$

Horizontal pass(es)

Circular Depth of Field

- Separated kernel

$$K * P = (R + iI) * (R + iI) * P$$

$$R * P \rightarrow P'_R$$

$$I * P \rightarrow P'_I$$

$$P'$$

(complex multiplication of elements)

$$V * P' \rightarrow P''$$

Horizontal pass(es)

Vertical pass

Imaginary parts from P' and V affect the real part of P''.

Circular Depth of Field

- Separated kernel

$$K * P = (R + iI) * (R + iI) * P$$

$$R * P \rightarrow P'_R$$

$$I * P \rightarrow P'_I$$

$$P'$$

$$V * P' \rightarrow P''$$

$$\Re[P'']$$

$$B$$

Horizontal pass(es)

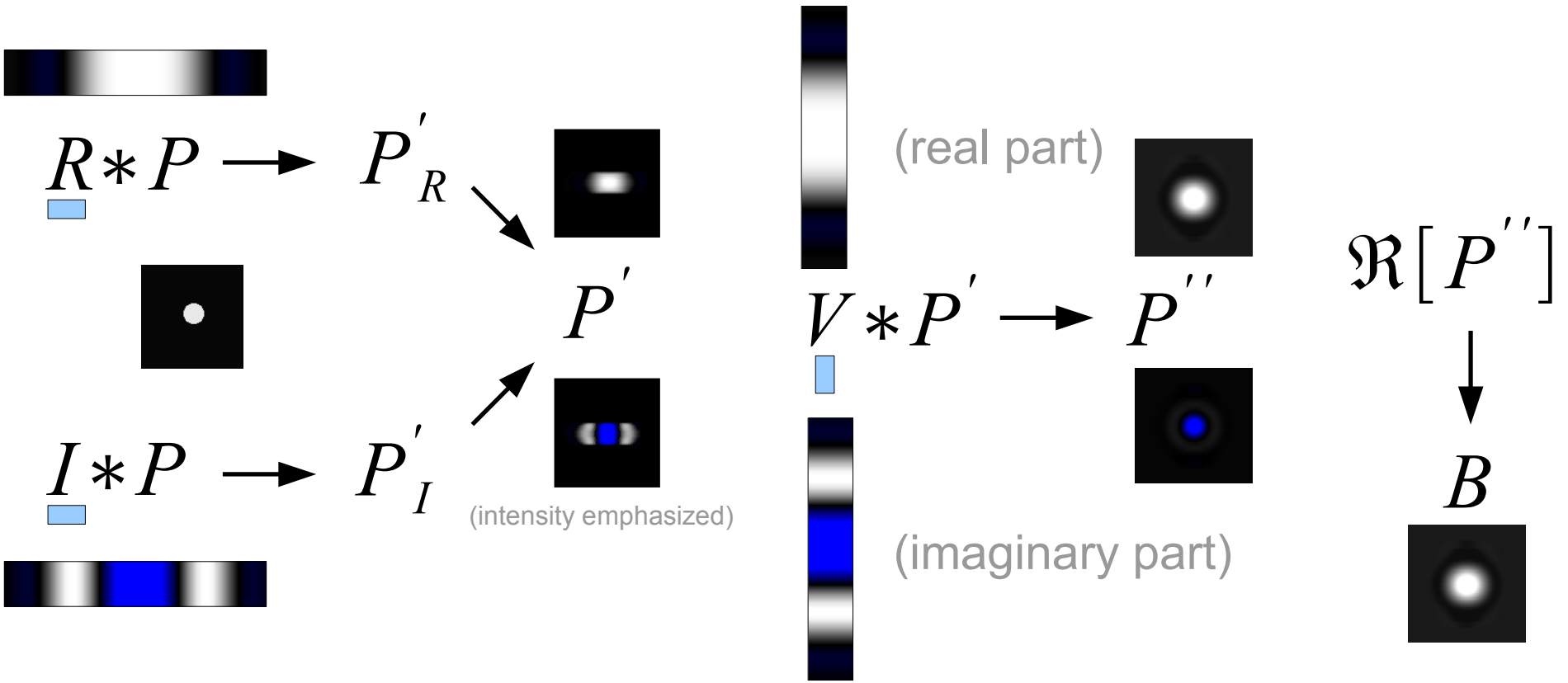
Vertical pass

Result

Circular Depth of Field

- Separated kernel

$$K * P = (R + iI) * (R + iI) * P$$



Horizontal pass(es)

Vertical pass

Result

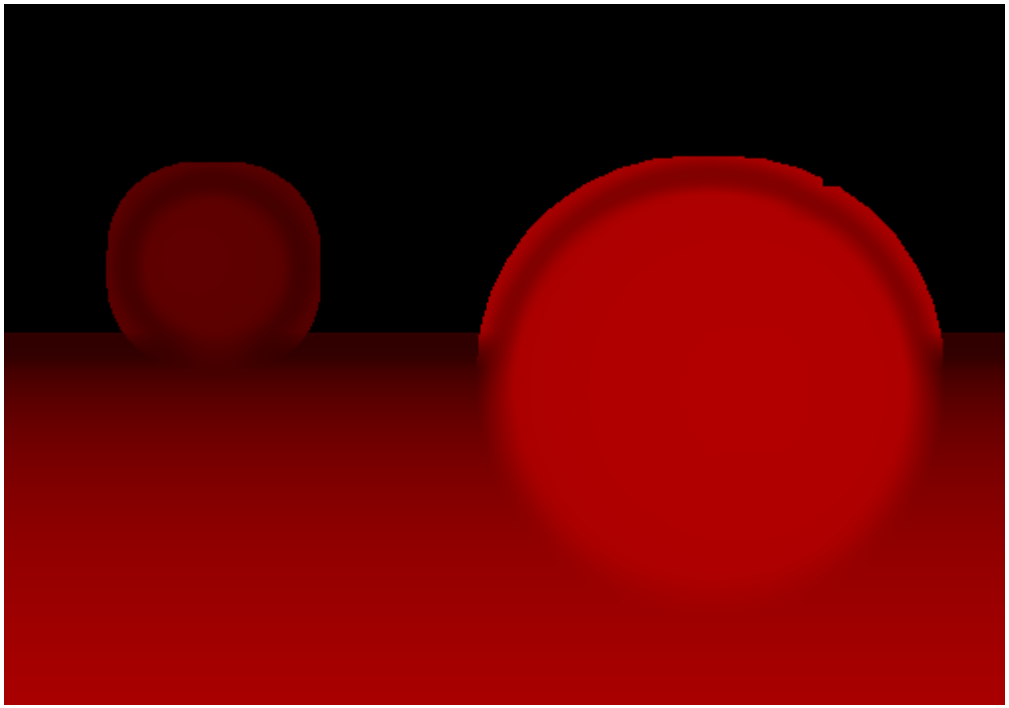
Circular Depth of Field

- Normalization
  - We want to scale the separated kernels such that the sum of the real parts of the final pass will be 1.
  - We need to calculate the real values of the kernel K.

$$\bar{V}(x) = \frac{V(x)}{\left| \sum_{x', y'} \Re[K(x', y')] \right|}$$

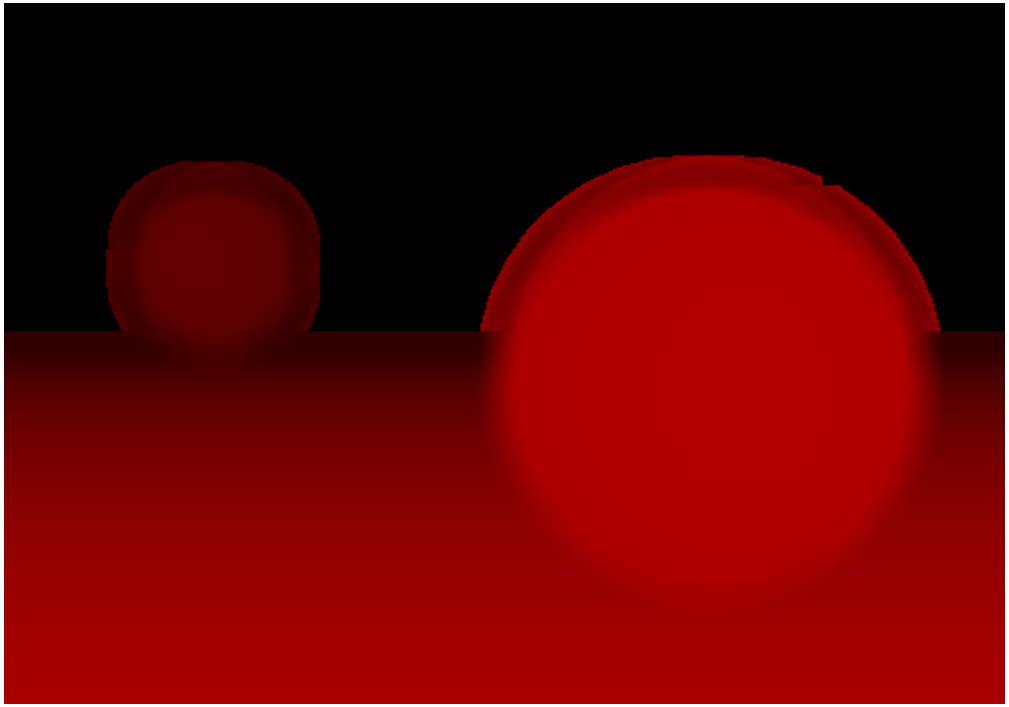
$$\Re[K(x, y)] = e^{a(x^2+y^2)} \cdot \left((c^2 - d^2) \cos(b(x^2+y^2)) + 2cd \sin(b(x^2+y^2)) \right)$$

Examples



2D

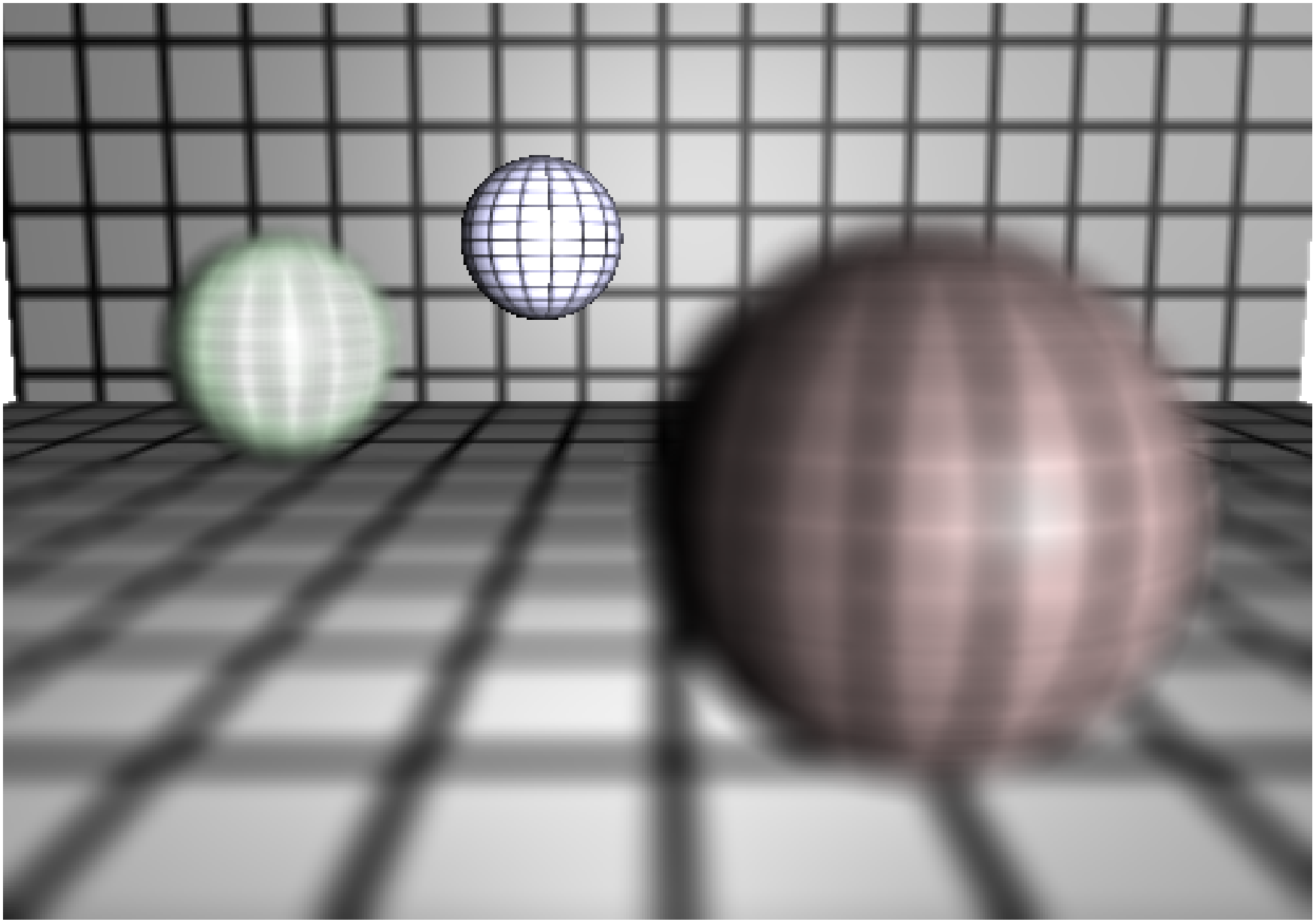
2×1D



The MAX-AVG border mix.

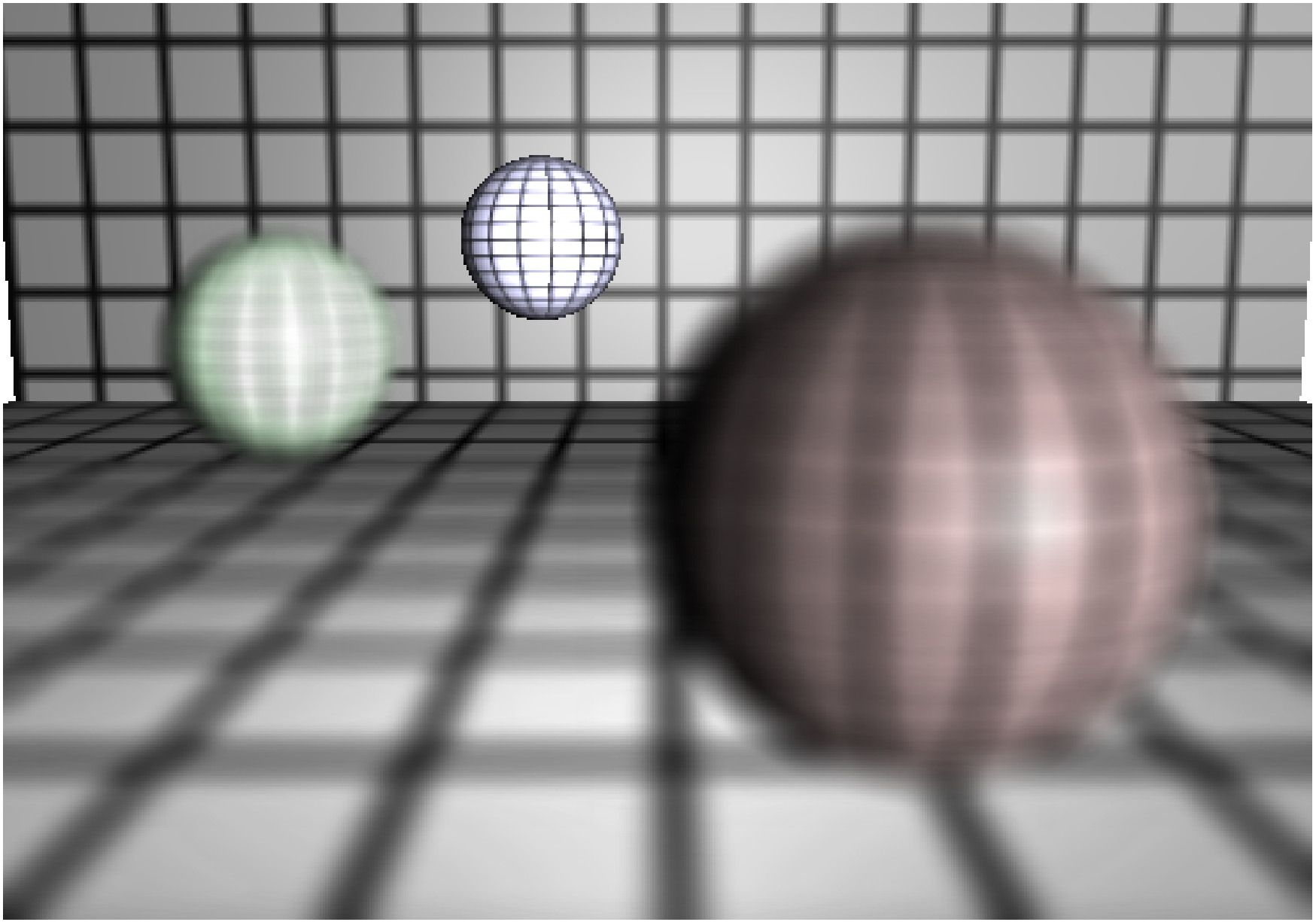
Examples

1×2D Circular DOF



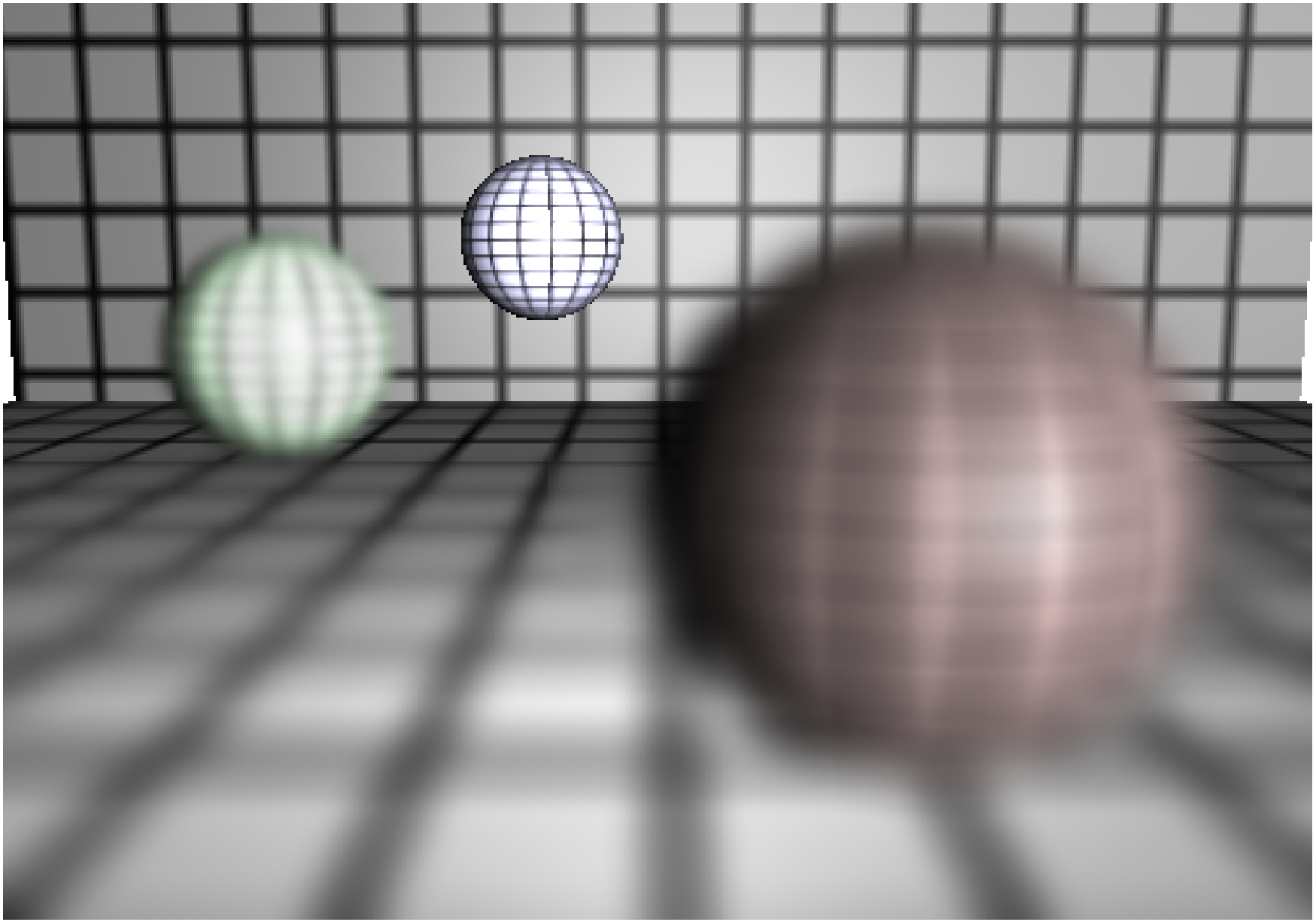
Examples

2×1D Circular DOF



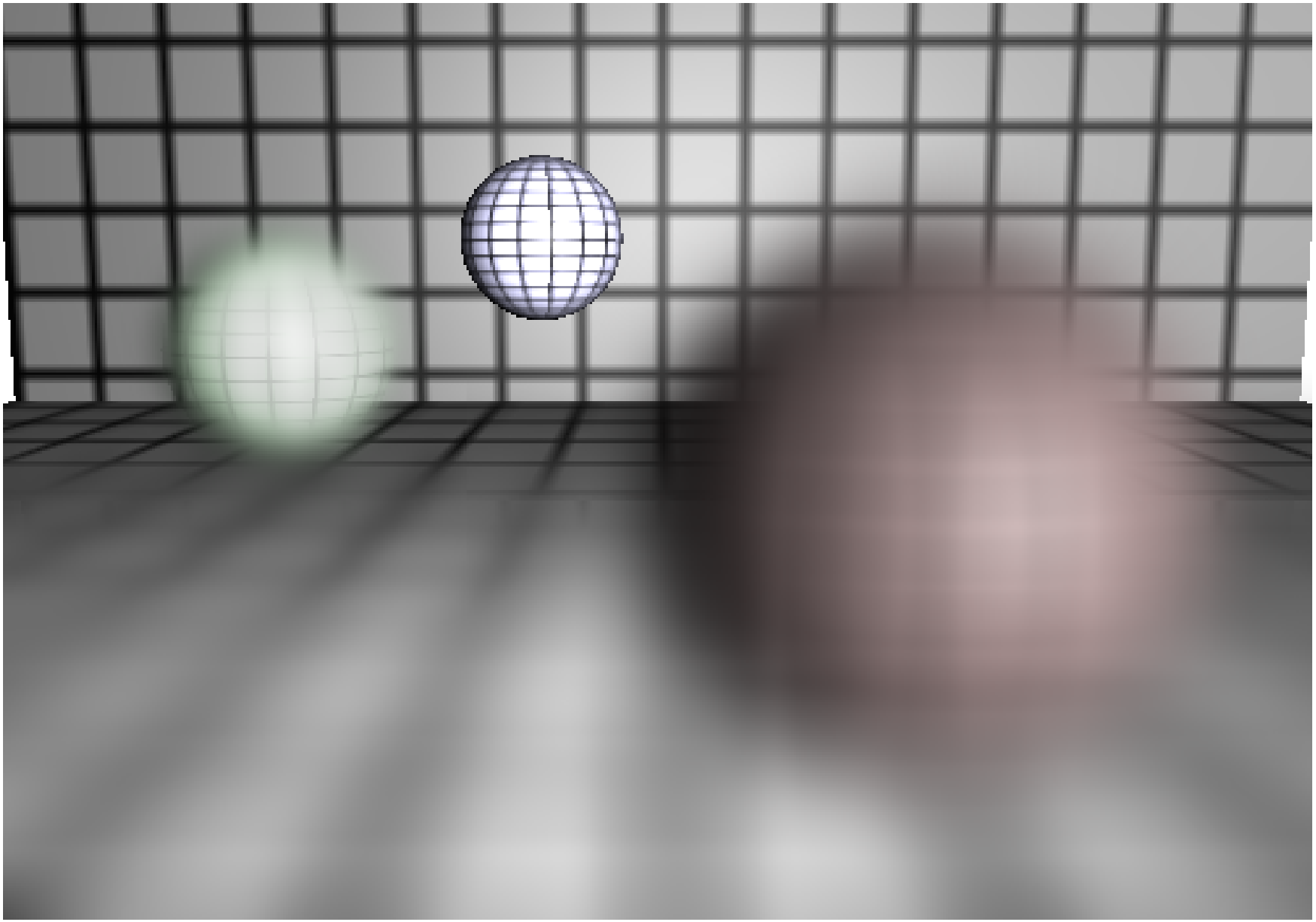
Examples

Gaussian blur



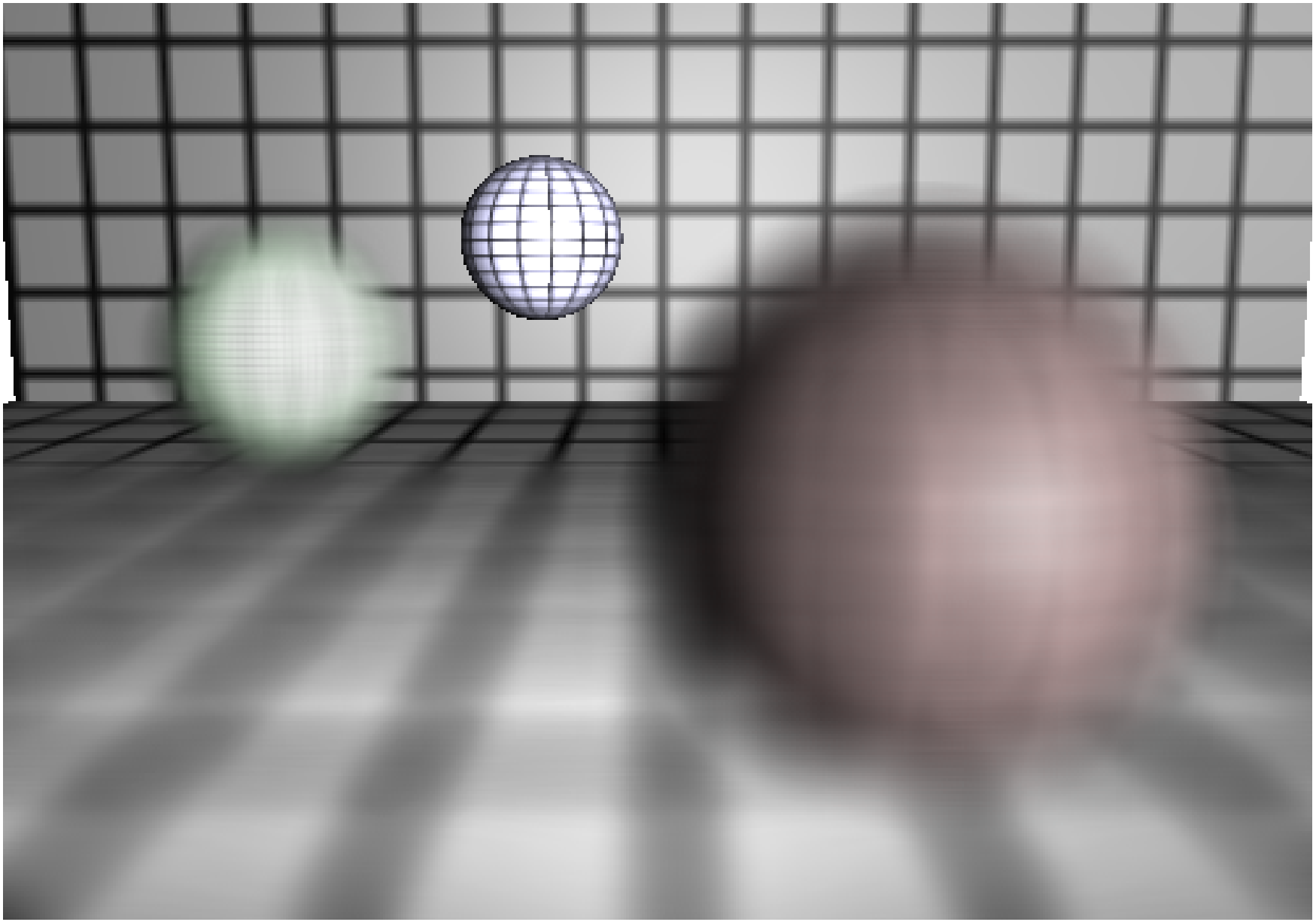
Examples

Gaussian blur



Examples

Circular DOF



Examples

- Very slow (unoptimized) demos.
  - **Gaussian DOF**

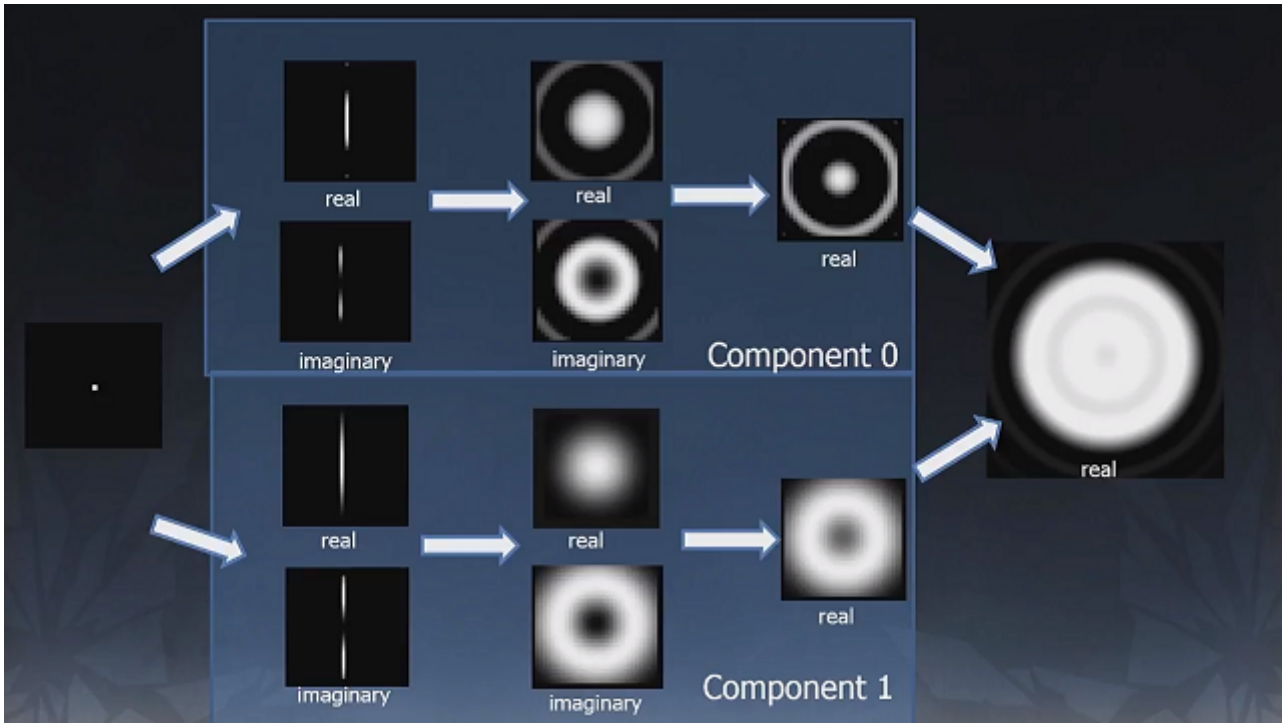
    http://tume-maailm.pri.ee/ylikool/CG/2018/test/dof/gaussian-dof.html
  - **Circular DOF**

    http://tume-maailm.pri.ee/ylikool/CG/2018/test/dof/circular-dof.html

- ~2000 LOC

References

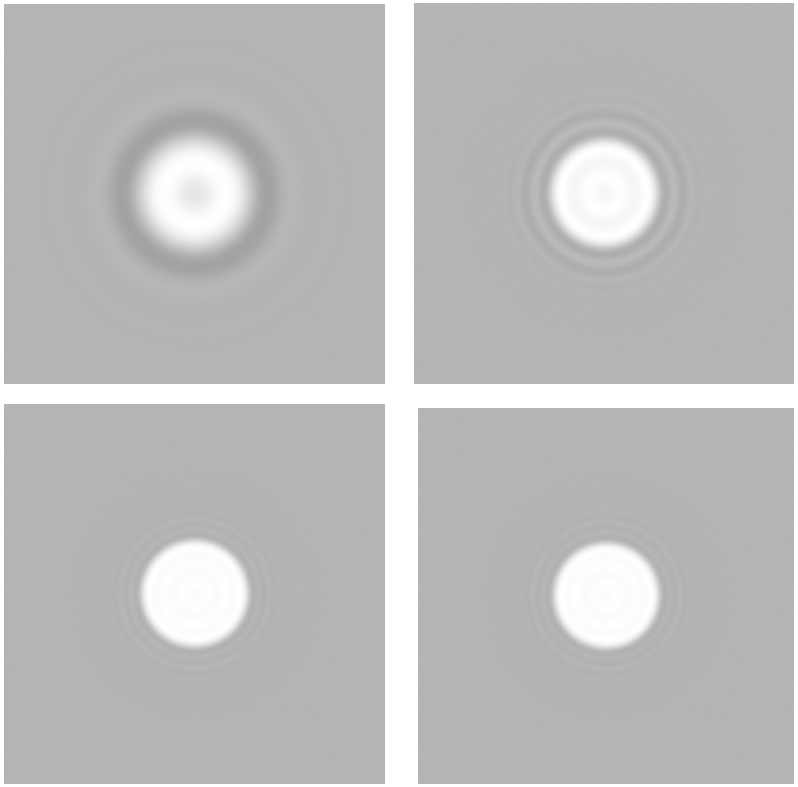
- **Circular Separable Convolution Depth of Field** (Kleber Garcia – EA, SIGGRAPH Talks 2017)

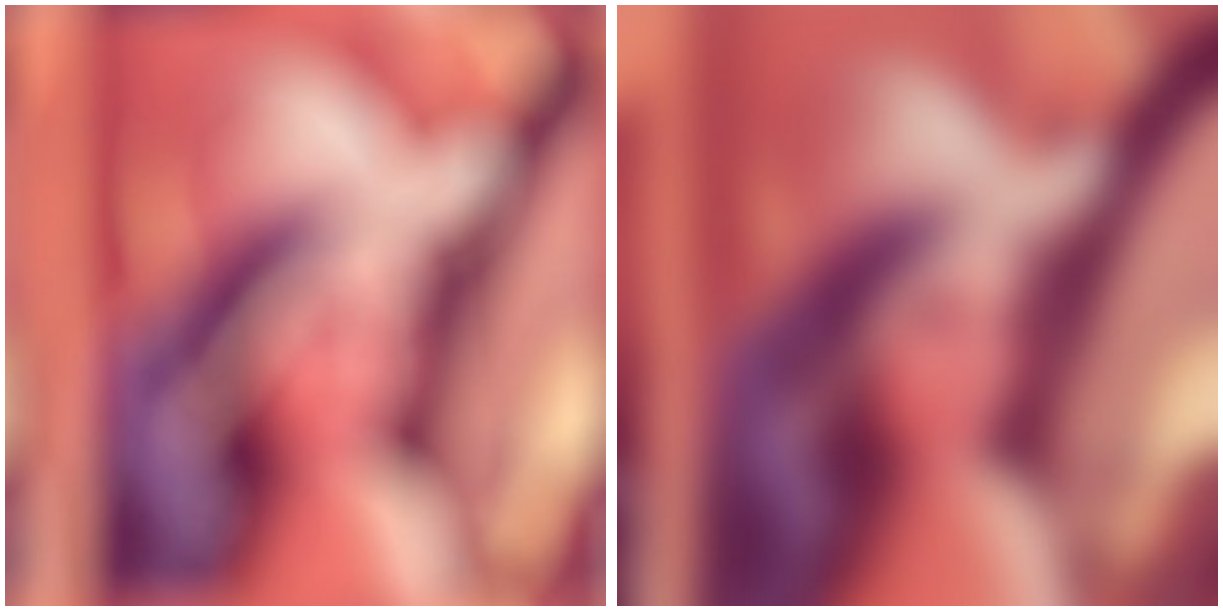


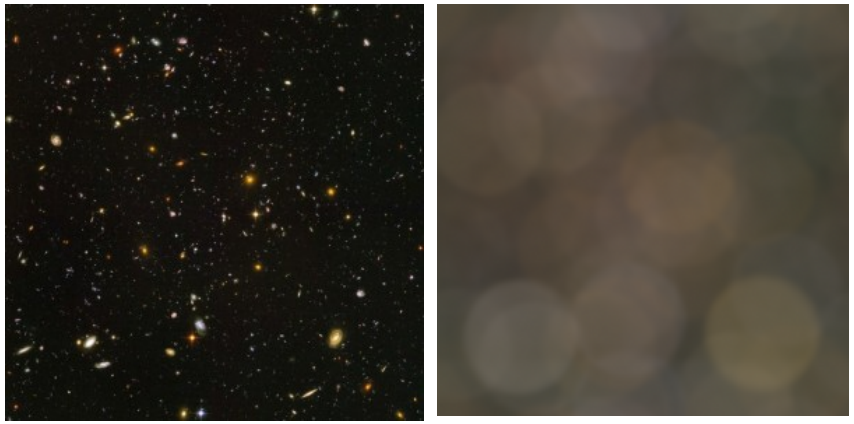
- https://www.youtube.com/watch?v=QKhydJSbcno
- https://dl.acm.org/citation.cfm?id=3085022
- GDC 2018: https://www.ea.com/frostbite/news/circular-separable-convolution-depth-of-field

References

- **Circularly symmetric convolution and lens blur** (Olli Niemitalo)







- http://yehar.com/blog/?p=1495

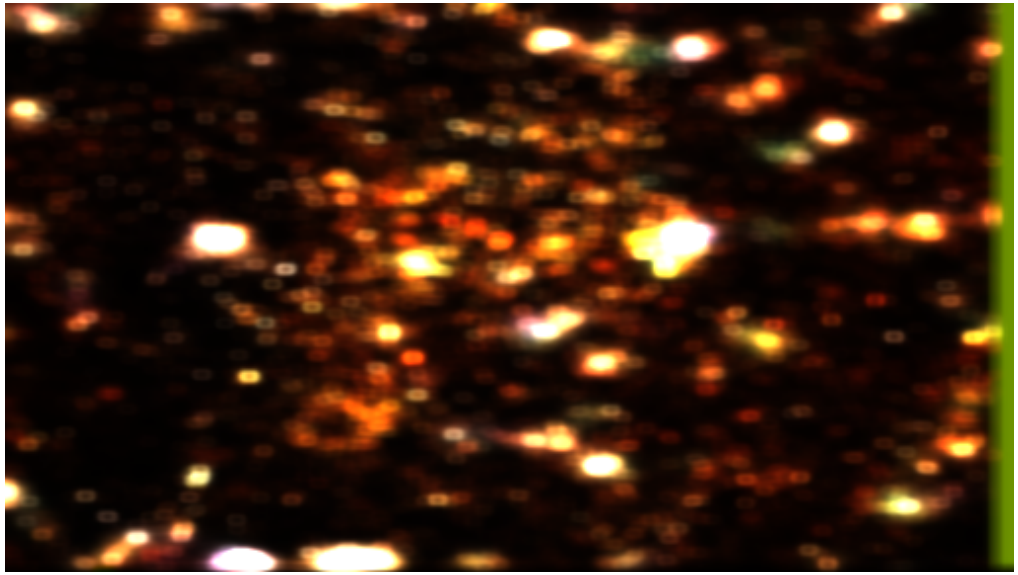
References

- **Video Game & Complex Bokeh Blurs**
  (Dr Mike Pound, Computerphile)

- https://www.youtube.com/watch?v=vNG3ZAd8wCc

References

- **Shadertoy implementations**



Bart Wronski

https://www.shadertoy.com/view/lsBBWy

Blogpost:

https://bartwronski.com/2017/08/06/separable-bokeh/

Kleber Garcia

https://www.shadertoy.com/view/Xd2BWc

Simplified by Egon Elbre

https://www.shadertoy.com/view/wdXXRS

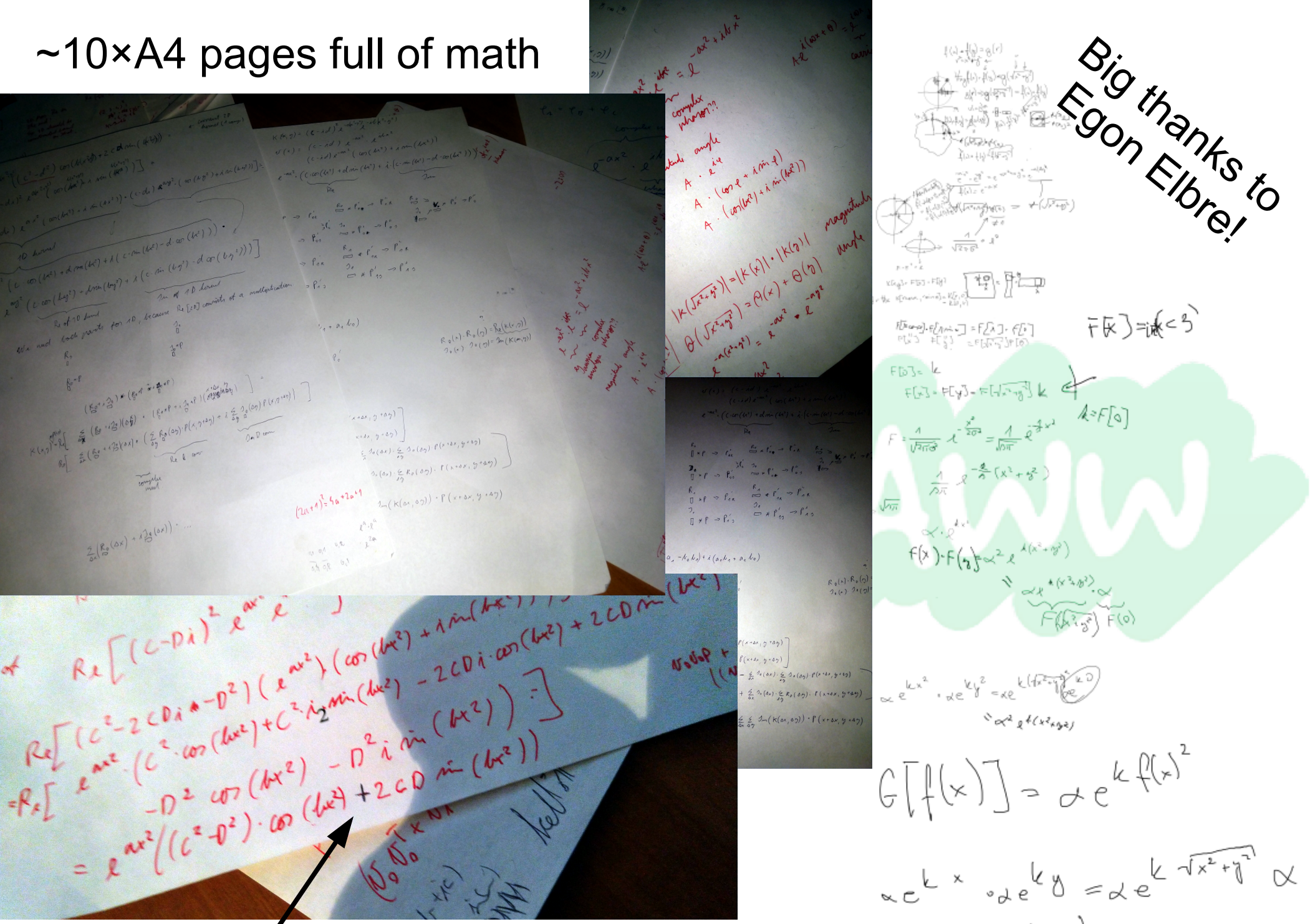
Filter generator

https://github.com/kecho/CircularDofFilterGenerator/blob/master/CircularDofFilterGen.py

References

- Terms
  - https://en.wikipedia.org/wiki/Kernel_(image_processing)
  - https://en.wikipedia.org/wiki/Circle_of_confusion
- GPU Gems
  - **Depth of Field: A Survey of Techniques**
    https://developer.nvidia.com/sites/all/modules/custom/gpugems/books/GPUGems/gpugems_ch23.html
  - **Practical Post-Process Depth of Field**
    https://developer.nvidia.com/gpugems/GPUGems3/gpugems3_ch28.html
- Other Stuff
  - **Algorithms for Rendering Depth of Field Effects in Computer Graphics (Survey)**
    https://people.eecs.berkeley.edu/~barsky/Blur/survey.WSEAS.2008.Algorithms.for.Rendering.Depth.of.Field.Effects.in.Computer.Graphics.pdf
  - **MIT Computer Vision: Linear Filters lecture**
    http://6.869.csail.mit.edu/fa16/lecture/lecture3linearfilters.pdf
  - **Deep Learning Book Series · 2.8 Singular Value Decomposition, Hadrien J.**
    https://hadrienj.github.io/posts/Deep-Learning-Book-Series-2.8-Singular-Value-Decomposition/
  - Another approach to DoF in Unity
    https://catlikecoding.com/unity/tutorials/advanced-rendering/depth-of-field/
  - Thread about DoF
    https://forum.libcinder.org/topic/depth-of-field-blur-weighted-sampling#23286000002682009
  - Single-pass Bokeh DoF
    http://blog.tuxedolabs.com/2018/05/04/bokeh-depth-of-field-in-single-pass.html

~10×A4 pages full of math



This + (instead of -) cost 10h of debugging.