成像算法基础

滤波

高思成像科技
Goss Imaging Technology

滤波卷积

滤波>卷积

17	24	12	28	31	36
21	5	7	29	18	36
4	6	13	12	15	14
16	18	21	13	15	11
22	22	31	24	26	36

卷积

17 9	24 8	12 7	28	31	36
21 4	5 5	7 6	29	18	36
4 1	6 2	13 3	12	15	14
16	18	21	13	15	11
22	22	31	24	26	36

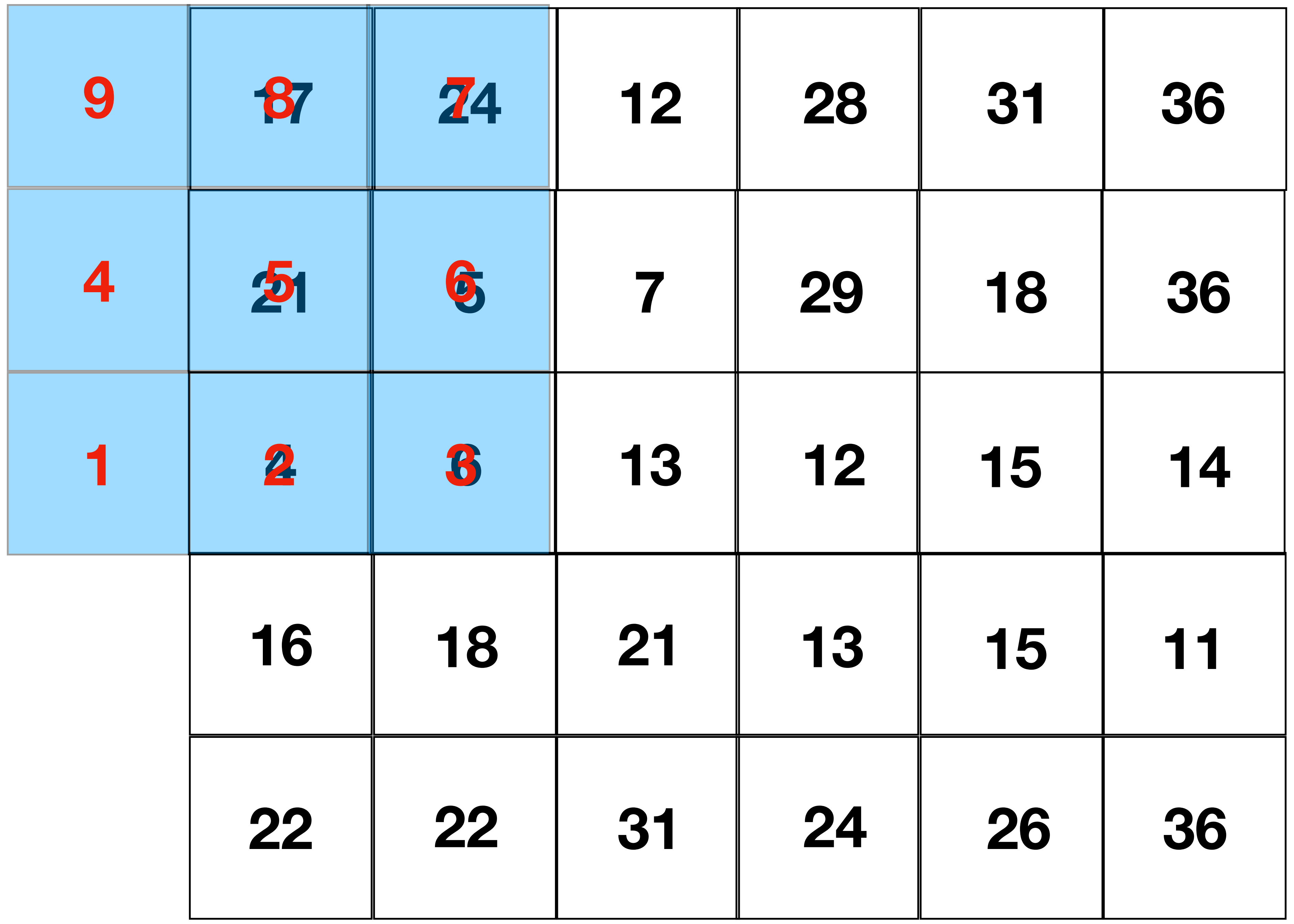
9X17+8X24+7X12+4X21+5X5+6X7+1X4

扩展的滤波应用

- 去噪
- 锐化
- Demosaic
- 畸变调整
- 统计
- ……..

通过像素及像素周周边值得到新的像素值的操作

边界效应



9	8	7
4	5	6
1	2	3

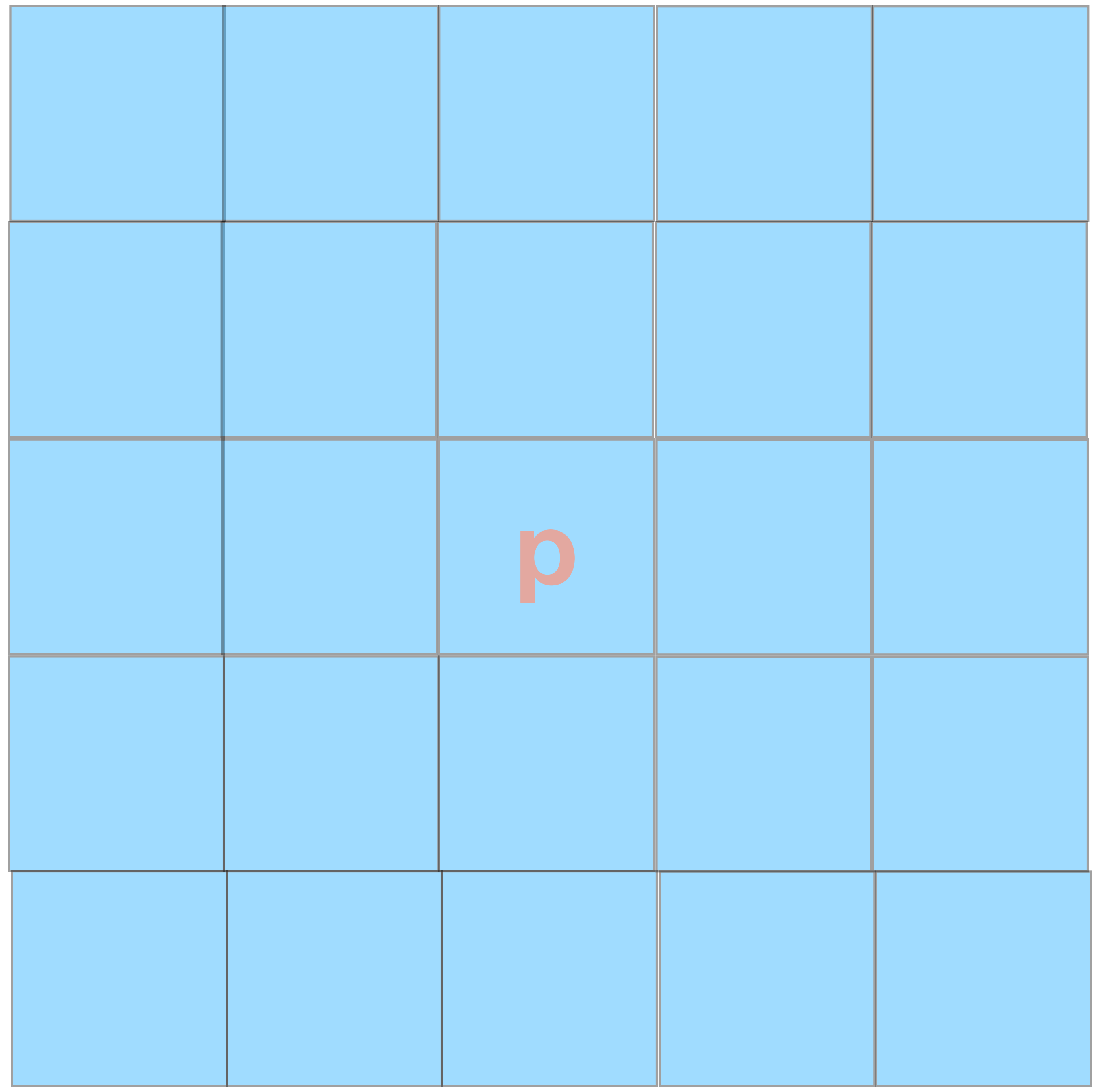
17	24	12	28	31	36
21	5	7	29	18	36
4	6	13	12	15	14
16	18	21	13	15	11
22	22	31	24	26	36

均值滤波,中值滤波

所有值和的平均值

可以卷积计算

$$\frac{\sum x_n}{N}$$

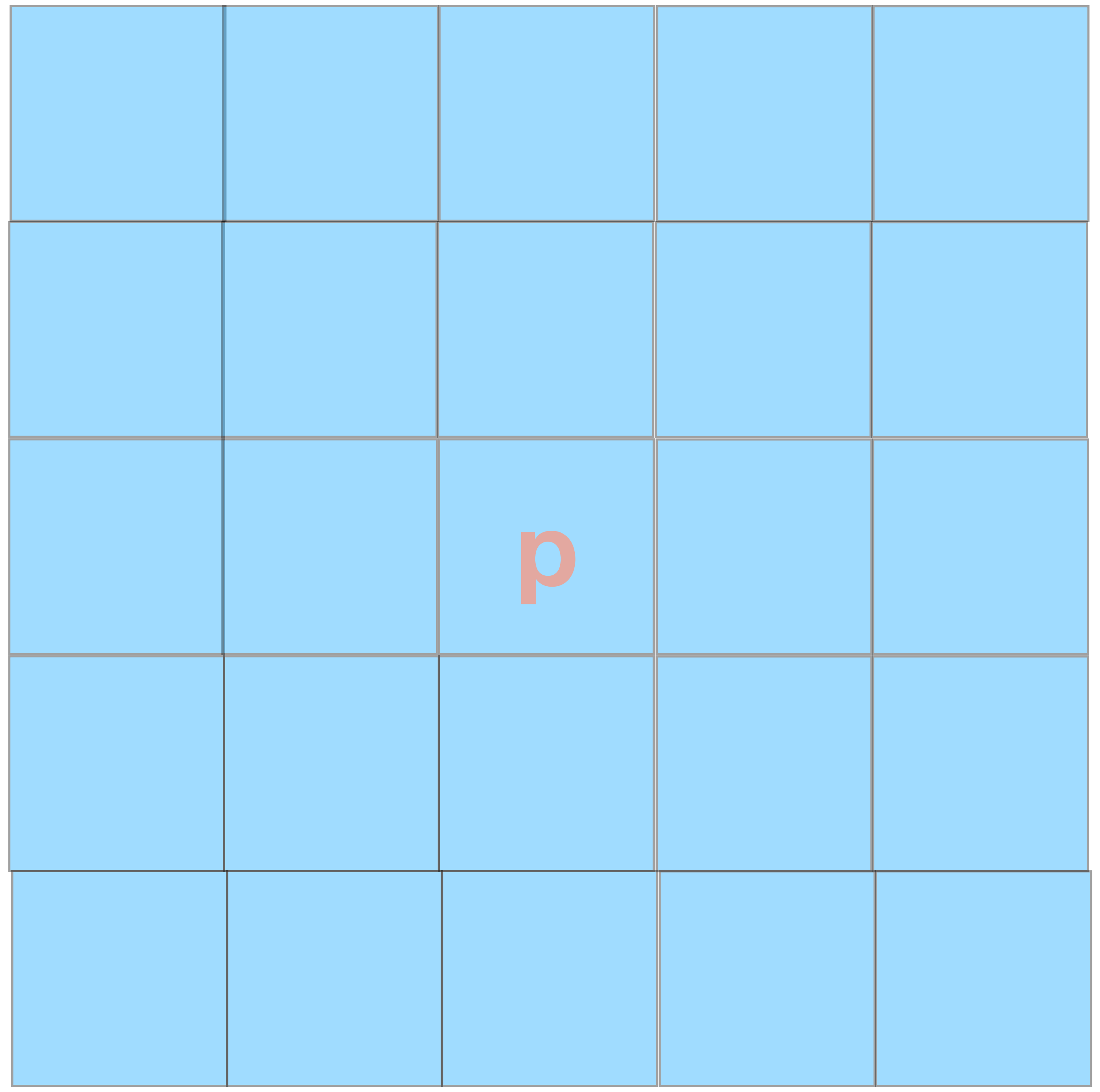


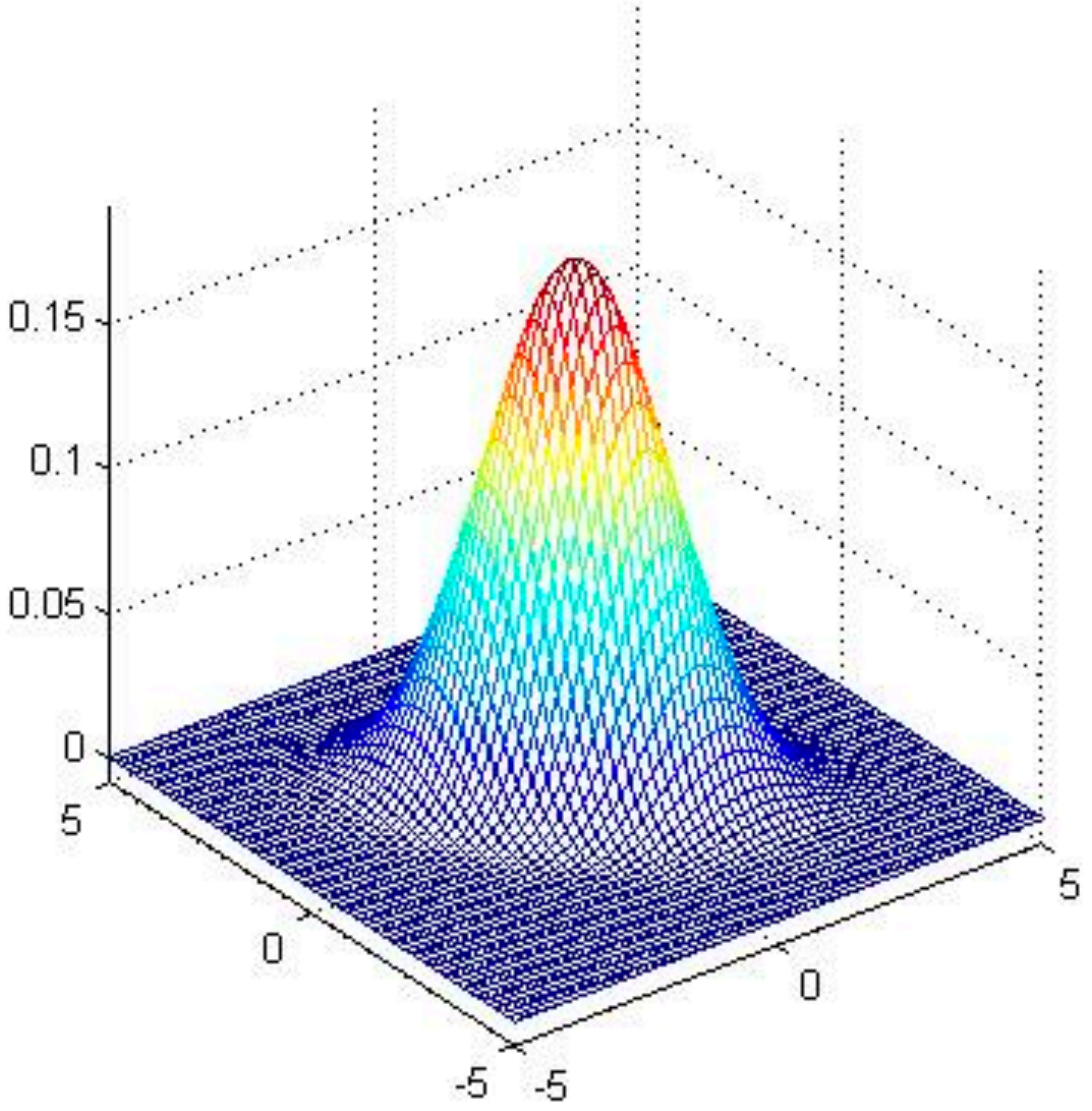
所有值和的中间值

不可以卷积计算

Median(Xn)

$$\frac{1}{\sqrt{2\Pi}\,\delta}e^{-\frac{x2+y2}{2\delta^2}}$$

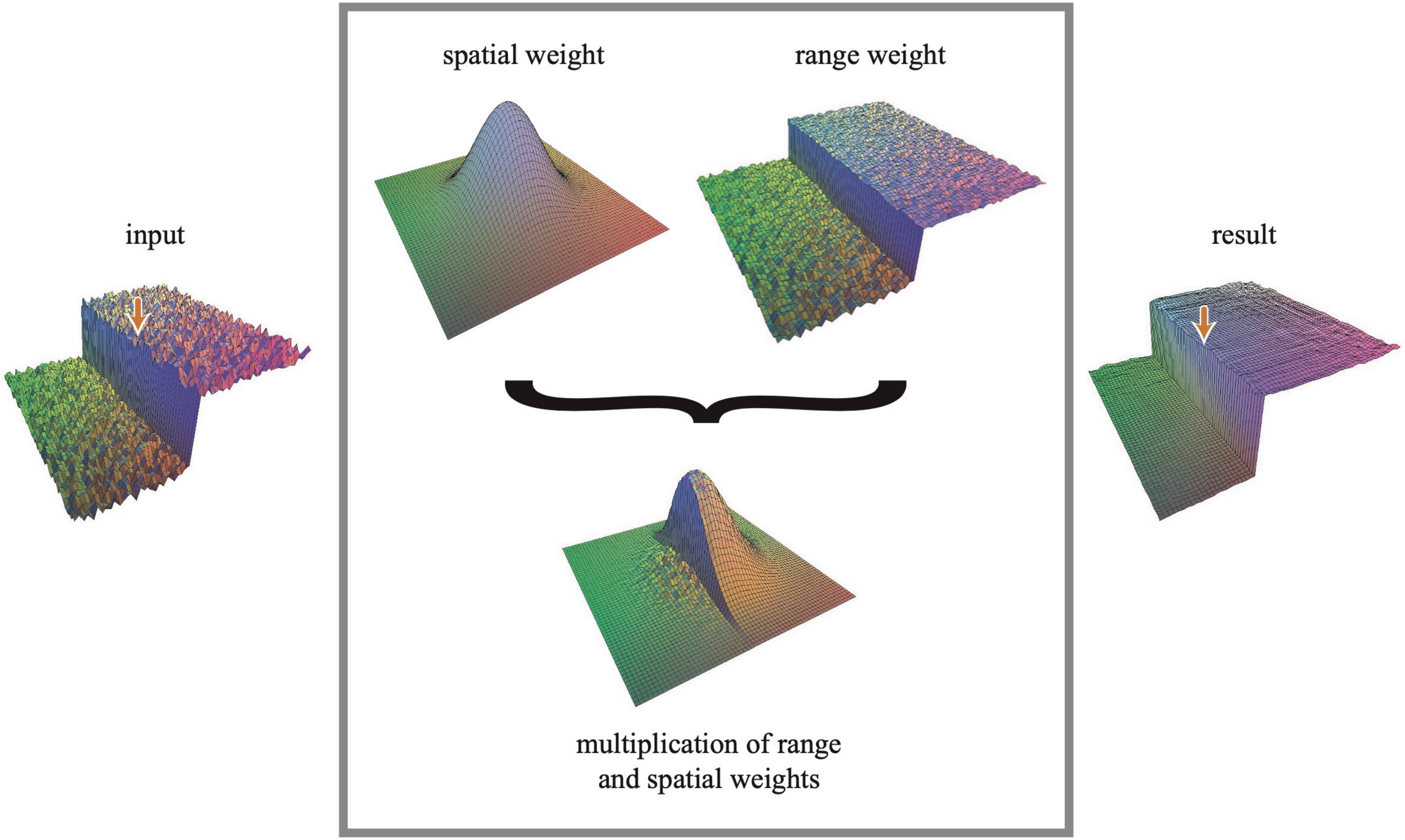




高思成像科技
Goss Imaging Technology

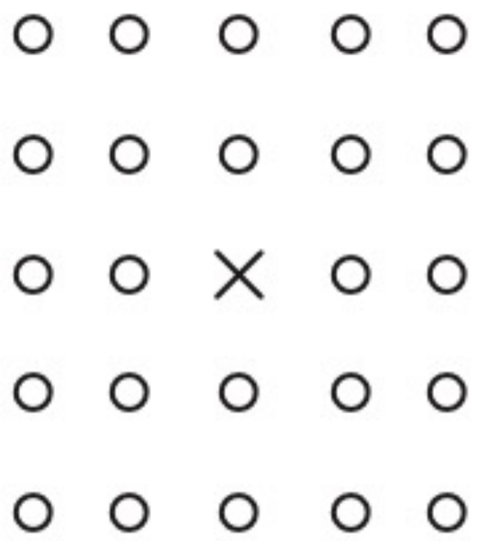
双边滤波

$$BF[I]_{\vec{p}} = \frac{\sum\limits_{\vec{q}\in S} \omega_s(\|\vec{p}-\vec{q}\|)\omega_r(\|I_{\vec{p}}-I_{\vec{q}}\|)I_{\vec{q}}}{\sum\limits_{\vec{q}\in S} \omega_s(\|\vec{p}-\vec{q}\|)\omega_r(\|I_{\vec{p}}-I_{\vec{q}}\|)}$$



FIR

$$y(m,n)=\sum_{k=-N}^{N}\sum_{l=-N}^{N}h(k,l)x(m-k,n-l)$$



$$\begin{matrix}1&2&1\\2&\boxed{4}&2\\1&2&1\end{matrix}\cdot\frac{1}{16}$$

$$\underbrace{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&16&16&16&16\\0&0&0&16&16&16&16\\0&0&0&16&16&16&16\\0&0&0&16&16&16&16\end{matrix}}_{\text{Input Image}}\Rightarrow\underbrace{\begin{matrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&1&3&4&4&3\\0&0&3&9&12&12&9\\0&0&4&12&16&16&12\\0&0&4&12&16&16&12\\0&0&3&9&12&12&9\end{matrix}}_{\text{Output Image}}$$

IIR

$$y(m,n) = x(m,n) + ay(m-1,n) + ay(m,n-1)$$

$$\begin{array}{cc} & \circ \\ \circ & \times \end{array} \qquad\qquad \begin{array}{cc} & 1/2 \\ 1/2 & \times \end{array}$$

$$\underbrace{\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 64 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}}_{\text{Input Image}} \Rightarrow \underbrace{\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 64 & 32 & 16 & 8 \\ 0 & 0 & 0 & 32 & 32 & 24 & 16 \\ 0 & 0 & 0 & 16 & 24 & 24 & 20 \\ 0 & 0 & 0 & 8 & 16 & 20 & 20 \end{array}}_{\text{Output Image}}$$

API的选择

- Scipy
- Opencv
- Skimge
- ………

Scipy中的滤波

gaussian_filter(input, sigma[, order, …])   Multidimensional Gaussian filter.

gaussian_filter1d(input, sigma[, axis, …]) One-dimensional Gaussian filter.

gaussian_gradient_magnitude(input, sigma[, …])   Multidimensional gradient magnitude using Gaussian derivatives.

gaussian_laplace(input, sigma[, output, …])   Multidimensional Laplace filter using gaussian second derivatives.

generic_filter(input, function[, size, …])   Calculate a multi-dimensional filter using the given function.

generic_filter1d(input, function, filter_size) Calculate a one-dimensional filter along the given axis.

generic_gradient_magnitude(input, derivative) Gradient magnitude using a provided gradient function.

generic_laplace(input, derivative2[, …])   N-dimensional Laplace filter using a provided second derivative function.

laplace(input[, output, mode, cval])   N-dimensional Laplace filter based on approximate second derivatives.

maximum_filter(input[, size, footprint, …]) Calculate a multi-dimensional maximum filter.

maximum_filter1d(input, size[, axis, …])   Calculate a one-dimensional maximum filter along the given axis.

median_filter(input[, size, footprint, …])   Calculate a multidimensional median filter.

minimum_filter(input[, size, footprint, …])   Calculate a multi-dimensional minimum filter.

minimum_filter1d(input, size[, axis, …])   Calculate a one-dimensional minimum filter along the given axis.

percentile_filter(input, percentile[, size, …]) Calculate a multi-dimensional percentile filter.

prewitt(input[, axis, output, mode, cval]) Calculate a Prewitt filter.

rank_filter(input, rank[, size, footprint, …]) Calculate a multi-dimensional rank filter.

sobel(input[, axis, output, mode, cval])   Calculate a Sobel filter.

uniform_filter(input[, size, output, mode, …])   Multi-dimensional uniform filter.

uniform_filter1d(input, size[, axis, …])   Calculate a one-dimensional uniform filter along the given axis.

…………

•

高思成像科技
Goss Imaging Technology

Opencv中的滤波

```
cv.blur(src, ksize[, dst[, anchor[, borderType]]])
cv.bilateralFilter(src, d, sigmaColor, sigmaSpace[, dst[, borderType]])
cv.dilate(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]])
cv.erode(src, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]]])
cv.filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]])
cv.GaussianBlur(src, ksize, sigmaX[, dst[, sigmaY[, borderType]]])
cv.getGaussianKernel(ksize, sigma[, ktype])
cv.boxFilter(src, ddepth, ksize[, dst[, anchor[, normalize[, borderType]]]])
.........
```

https://docs.opencv.org/4.2.0/d4/d86/group__imgproc__filter.html#gad533230ebf2d42509547d514f7d3fbc3

自己实现的滤波

```
generic_filter(input, function[, size, ...])
cv.filter2D(src, ddepth, kernel[, dst[, anchor[, delta[, borderType]]]])
```

17	24	12	28	31	36
21	5	7	29	18	36
4	6	13	12	15	14
16	18	21	13	15	11
22	22	31	24	26	36

高思成像科技
Goss Imaging Technology