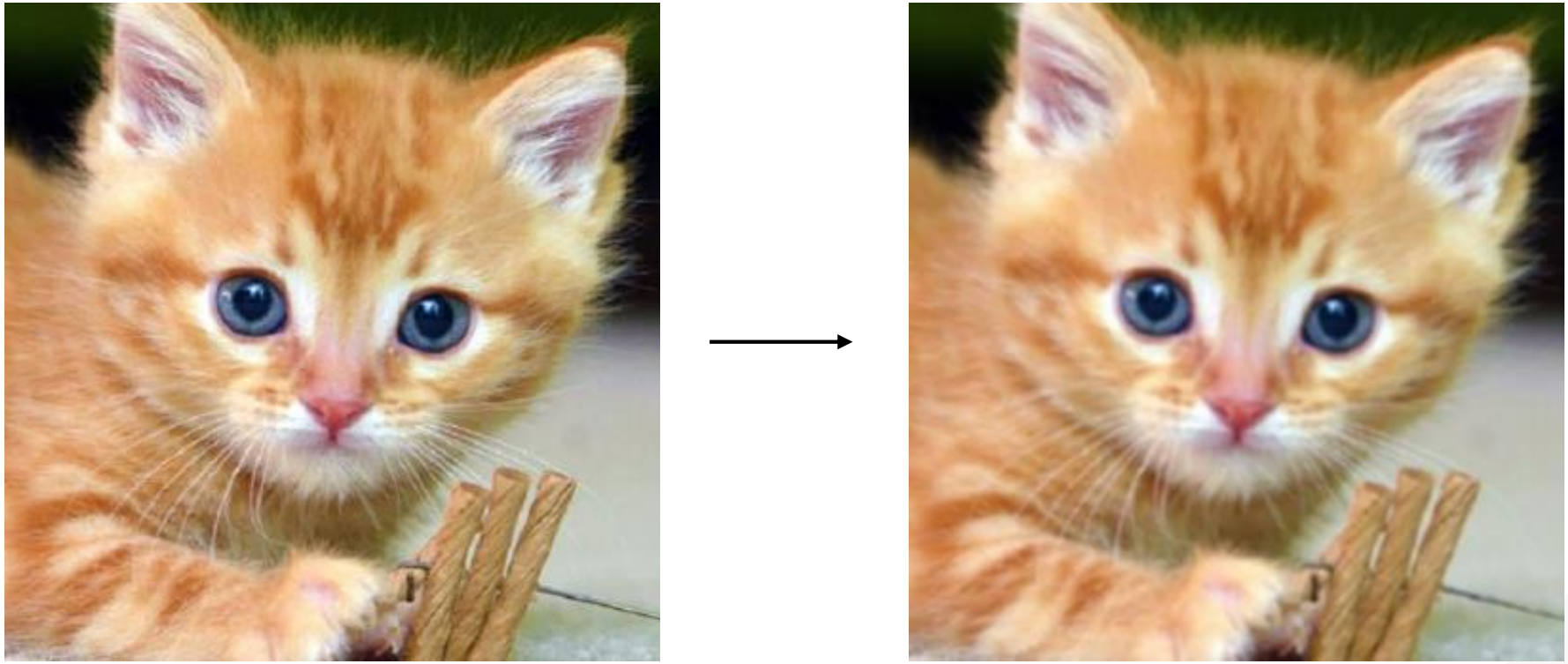
Example: Blur

- In computer vision, image blurring is an important operation to reduce the impact of noise on an image in some vision tasks such as edge detection and object recognition.



Output pixel is the average of the corresponding input pixel and the pixels around it

Motivation - A Computer Vision Example: Contour Detection

Original Image

Grayscale Counterpart

Blurred image

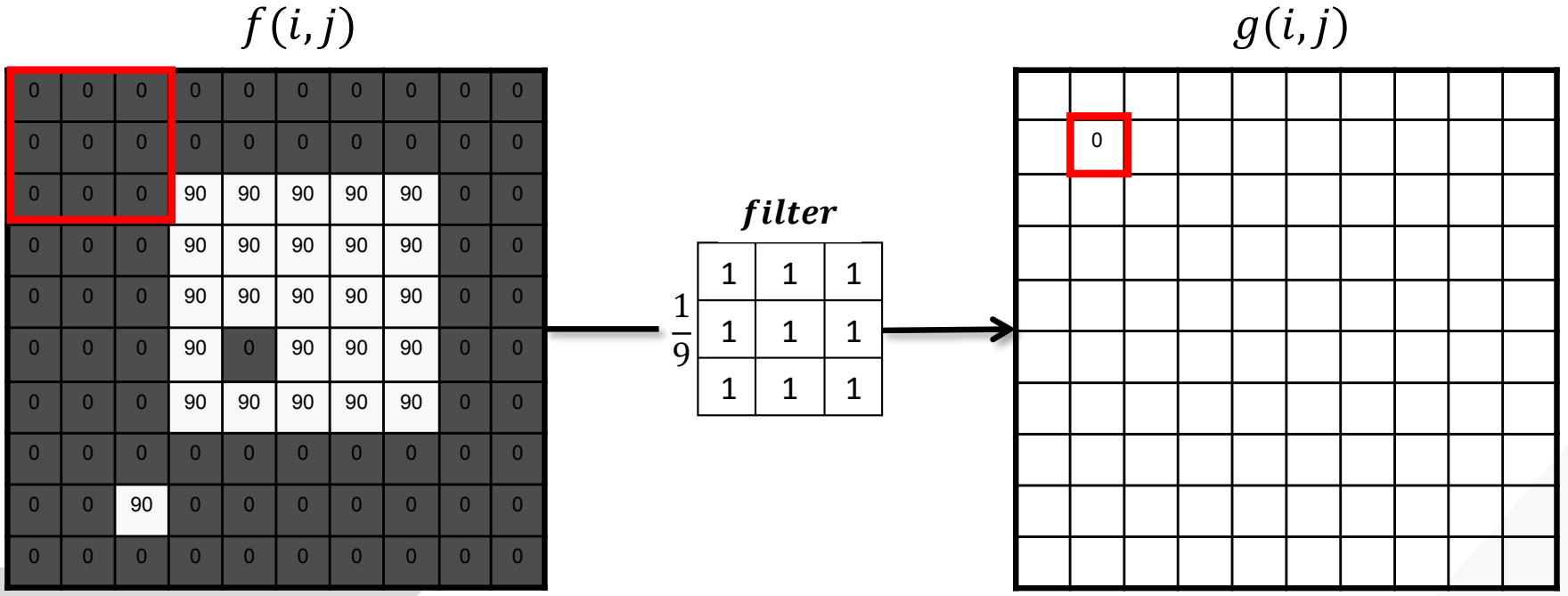
Edges

Contours

Example: Blur

- Mathematically, an image-blurring function calculates the value of an output image pixel as a weighted sum of a patch of pixels encompassing the pixels in the input image.

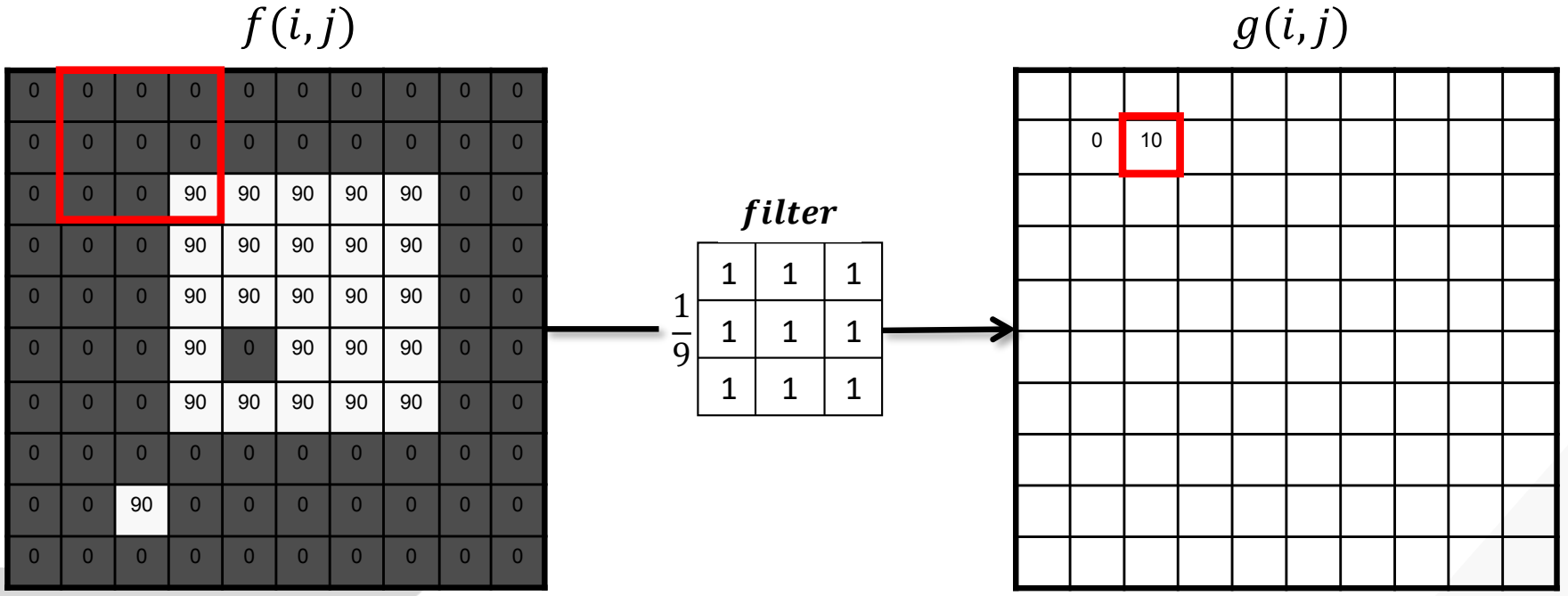
- For example, the following presents an image blurring processing using a box filter (where weight are all 1s; a simplest blurring filter).



Example: Blur

- Mathematically, an image-blurring function calculates the value of an output image pixel as a weighted sum of a patch of pixels encompassing the pixels in the input image.

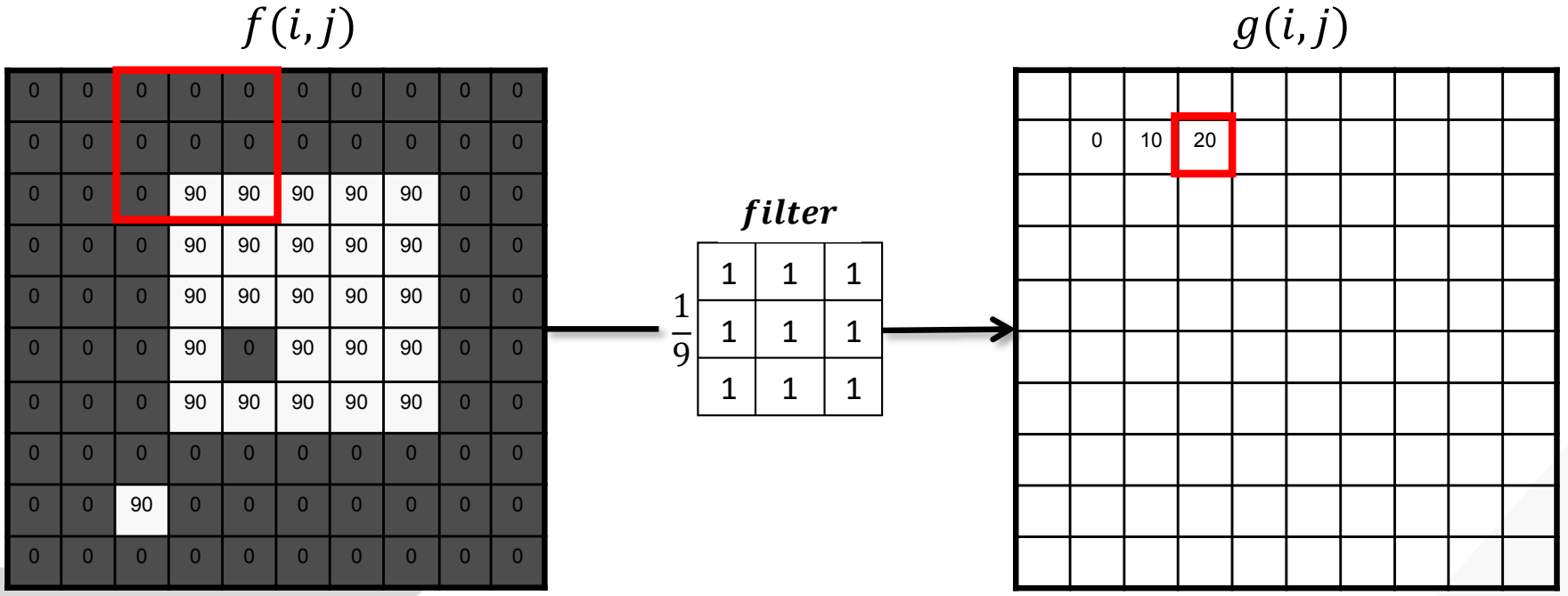
- For example, the following presents an image blurring processing using a box filter (where weight are all 1s; a simplest blurring filter).



Example: Blur

- Mathematically, an image-blurring function calculates the value of an output image pixel as a weighted sum of a patch of pixels encompassing the pixels in the input image.

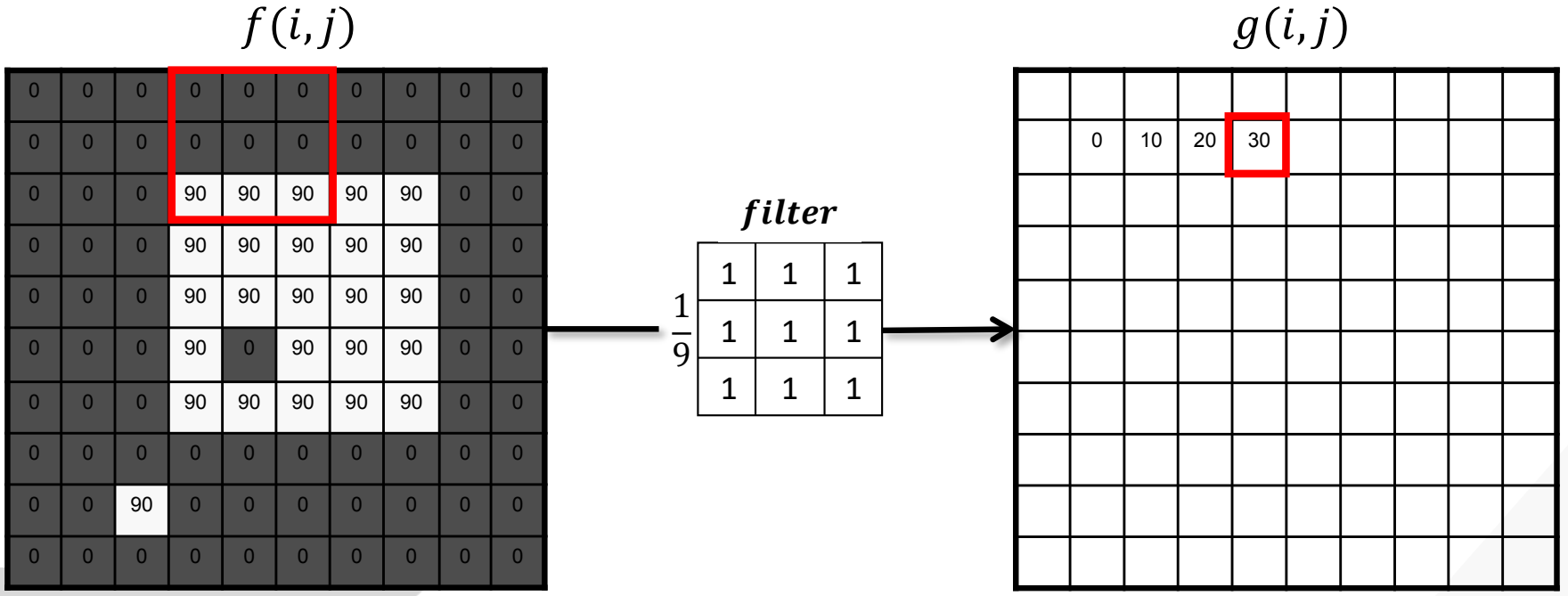
- For example, the following presents an image blurring processing using a box filter (where weight are all 1s; a simplest blurring filter).



Example: Blur

- Mathematically, an image-blurring function calculates the value of an output image pixel as a weighted sum of a patch of pixels encompassing the pixels in the input image.

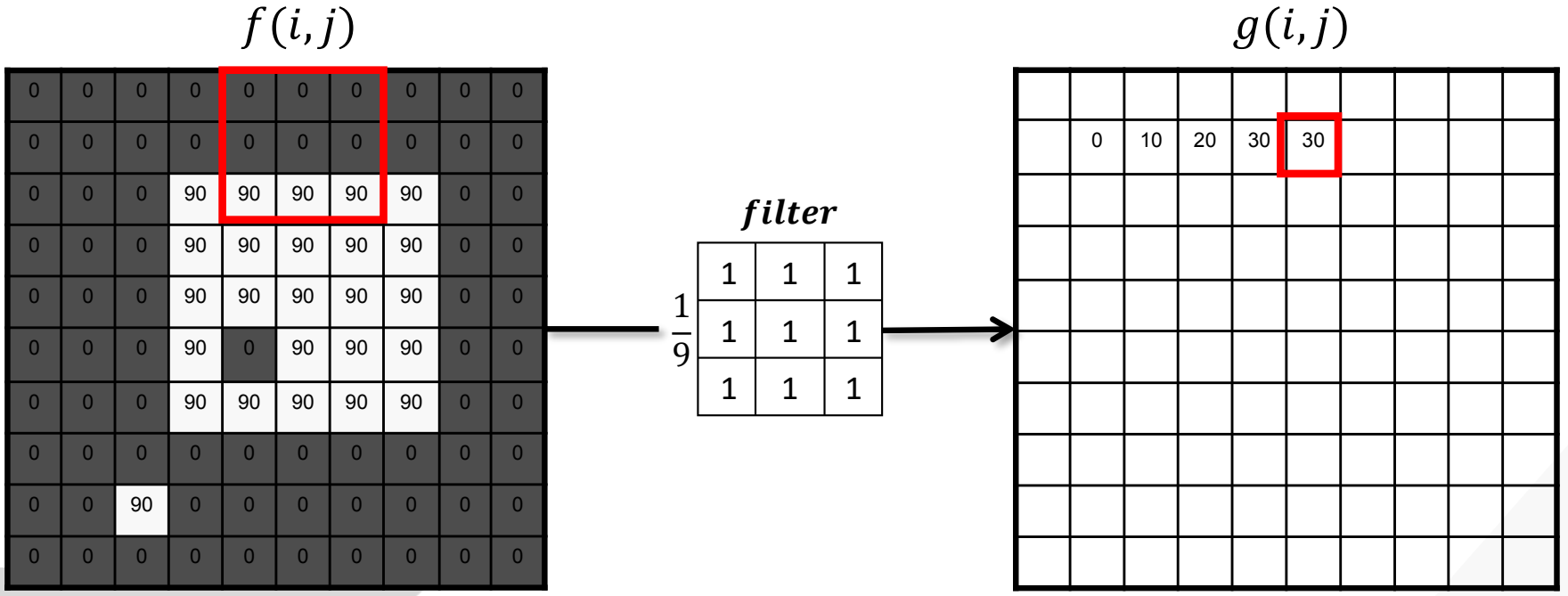
- For example, the following presents an image blurring processing using a box filter (where weight are all 1s; a simplest blurring filter).



Example: Blur

- Mathematically, an image-blurring function calculates the value of an output image pixel as a weighted sum of a patch of pixels encompassing the pixels in the input image.

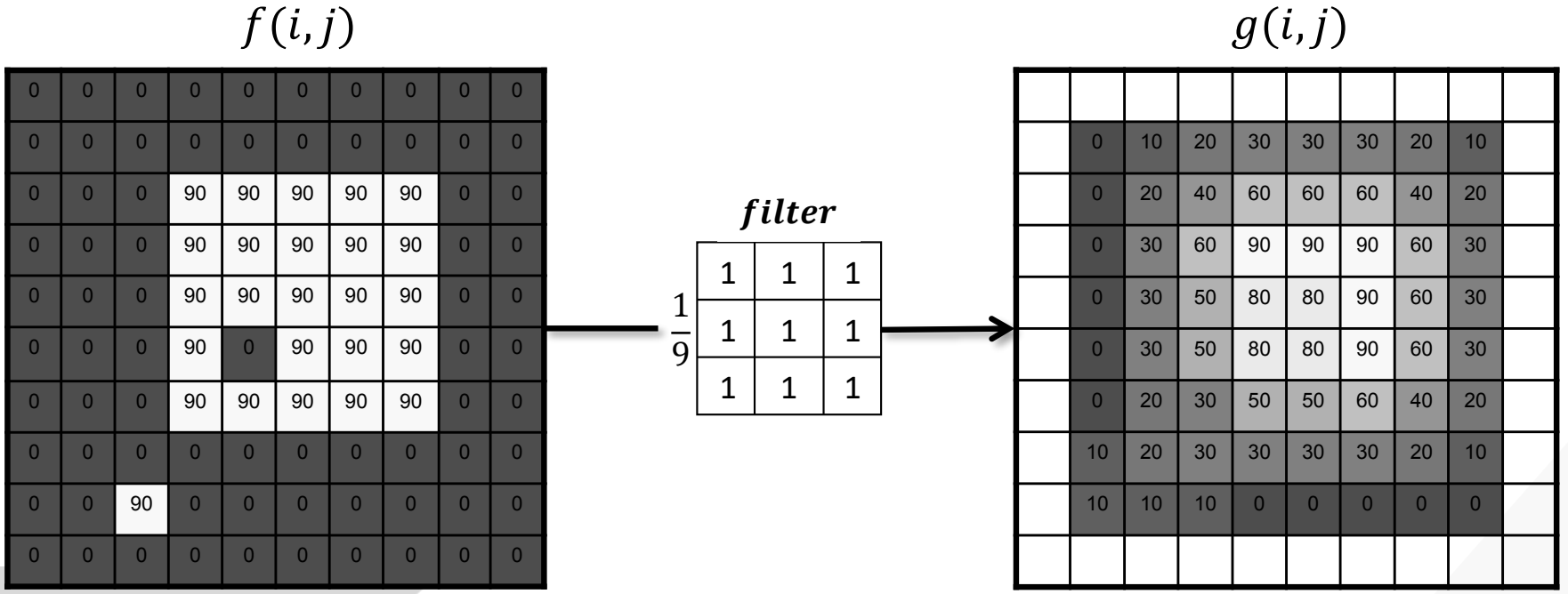
- For example, the following presents an image blurring processing using a box filter (where weight are all 1s; a simplest blurring filter).



Example: Blur

- Mathematically, an image-blurring function calculates the value of an output image pixel as a weighted sum of a patch of pixels encompassing the pixels in the input image.

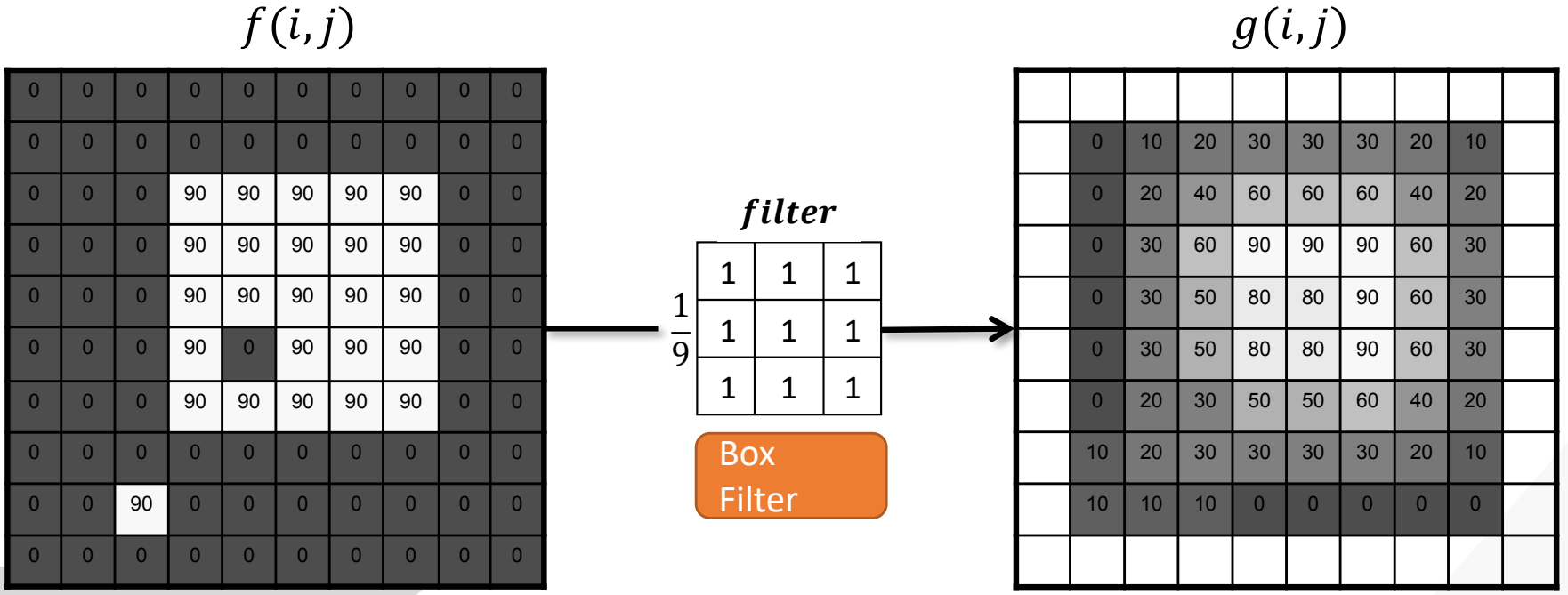
- For example, the following presents an image blurring processing using a box filter (where weight are all 1s; a simplest blurring filter).



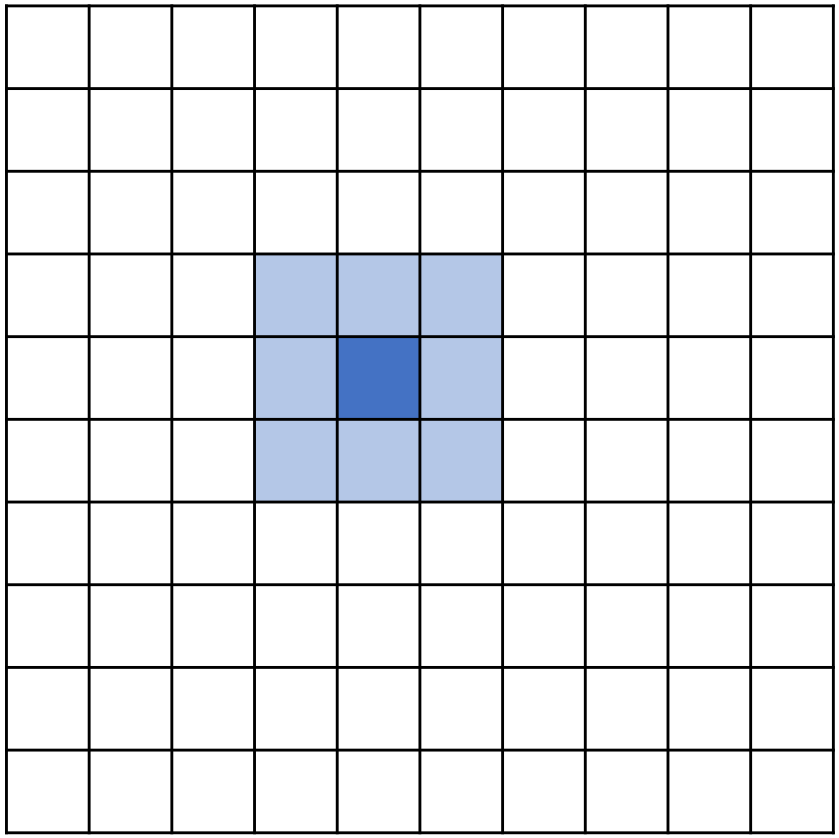
Example: Blur

- Mathematically, an image-blurring function calculates the value of an output image pixel as a weighted sum of a patch of pixels encompassing the pixels in the input image.

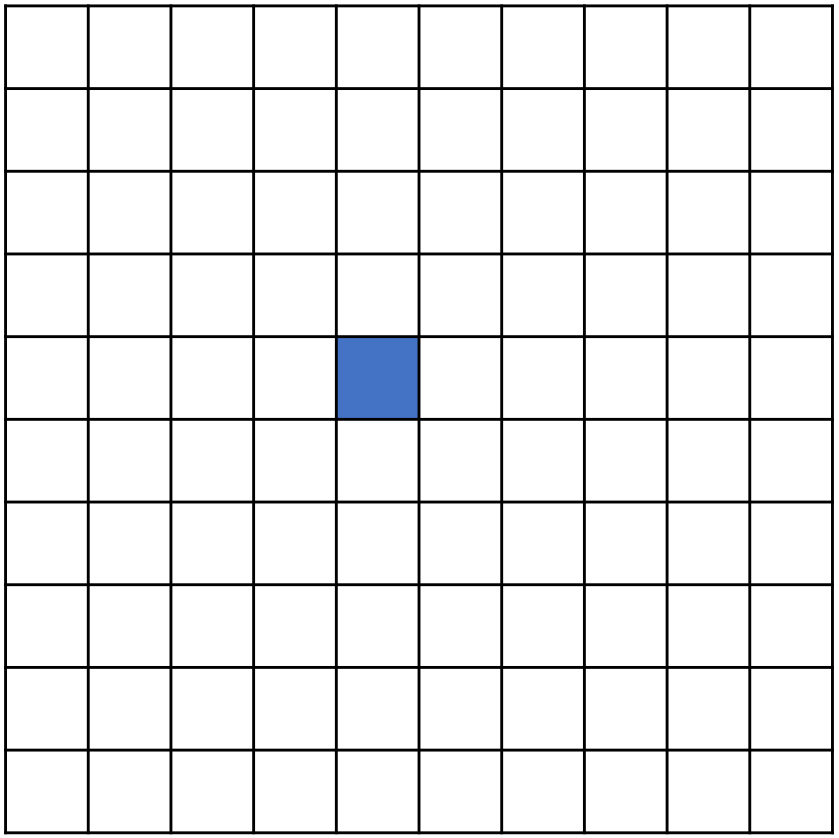
- For example, the following presents an image blurring processing using a box filter (where weight are all 1s; a simplest blurring filter).



Example: Blur



Input Image

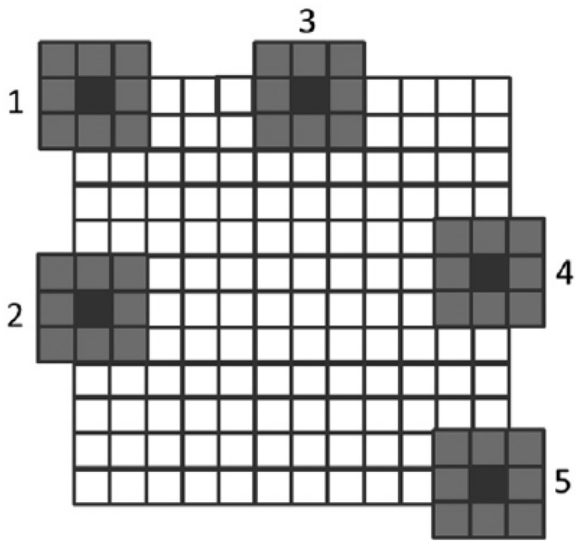


Output Blurred Image

Parallelization approach: assign one thread to each *output* pixel, and have it read multiple *input* pixels

Example: Blur

```
67  __global__ void blurImage_Kernel(unsigned char * Pout, unsigned char * Pin, unsigned int width, unsigned int height, int filter_size)
68  {
69      int colIdx = blockIdx.x * blockDim.x + threadIdx.x;
70      int rowIdx = blockIdx.y * blockDim.y + threadIdx.y;
71
72      // In C/C++, 2D arrays are linearized following the row-major layout
73      if(colIdx < width && rowIdx < height){
74          float sumPixVal = 0.0;
75          int filterRadius = filter_size/2;
76
77          for(int blurRowOffset = -1*filterRadius; blurRowOffset < (filterRadius+1); blurRowOffset++){
78              for(int blurColOffset = -1*filterRadius; blurColOffset < (filterRadius+1); blurColOffset++){
79                  int curRowIdx = rowIdx + blurRowOffset;
80                  int curColIdx = colIdx + blurColOffset;
81
82                  if( (curRowIdx>=0) && (curRowIdx<height) && (curColIdx>=0) && (curColIdx < width) ){
83                      // As we use the linearized index, "curRowIdx * width + curColIdx" represents the offset for each matrix element when working with the filter.
84                      sumPixVal += Pin[curRowIdx * width+ curColIdx]/255.0;
85                      //nPixels++;
86                  }
87              }
88          }
89          // compute the average. Note that as we use the linearized index, "rowIdx * width + colIdx" represents the offset for each element in the result matrix.
90          Pout[rowIdx * width+ colIdx] = (unsigned char)((sumPixVal/(filter_size*filter_size))*255.0);
91      }
92  }
```



Boundary Conditions

- Even if output pixel is in bounds, might be accessing out of bounds input pixel

Rule of thumb: every memory access must have a corresponding guard the compares its indexes to the array dimensions