Advanced Programming

Programming Assignment #3

Kang Hoon Lee

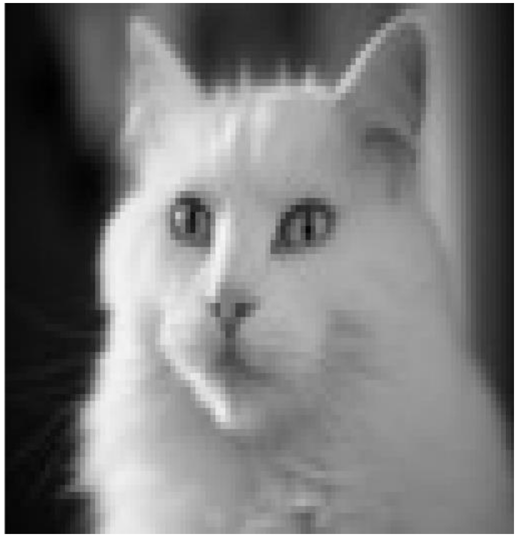
Kwangwoon University

Bitmaps and Filters

- □ `BitmapImage`

Bitmaps and Filters

- BitmapImage
  - bool loadPGM(const std::string& path);
  - bool savePGM(const std::string& path);

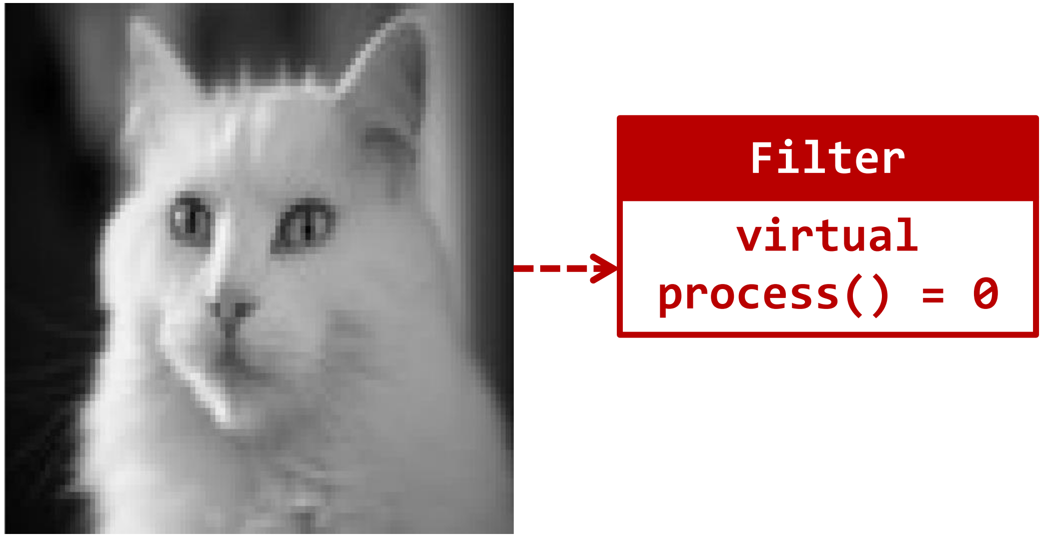


```
P2
102 76
255
7 9 10 12 12 14 21 27 33 ...
107 114 119 119 117 109 ...
42 41 38 35 32 31 32 35 ...
69 80 89 100 108 117 122 ...
17 13 8 9 10 11 11 13 19 ...
110 117 118 120 120 119 ...
47 45 43 42 41 39 36 34 ...
...
```

cat.pgm

Bitmaps and Filters

- BitmapImage → Filter

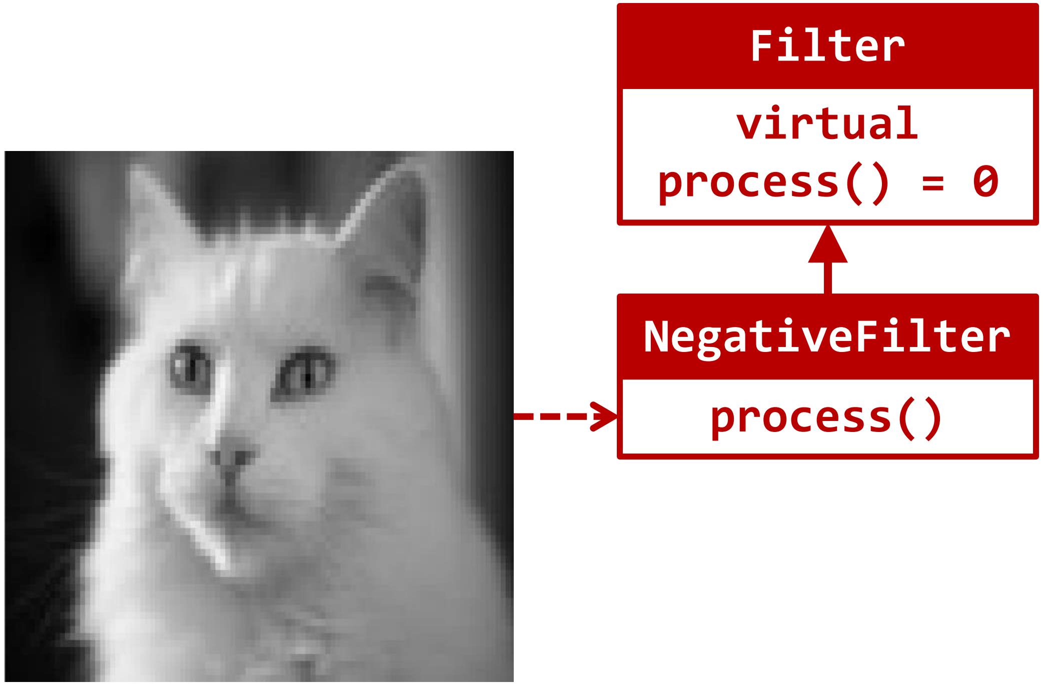


Filter

virtual process() = 0

Bitmaps and Filters

- BitmapImage ➔ NegativeFilter



Filter

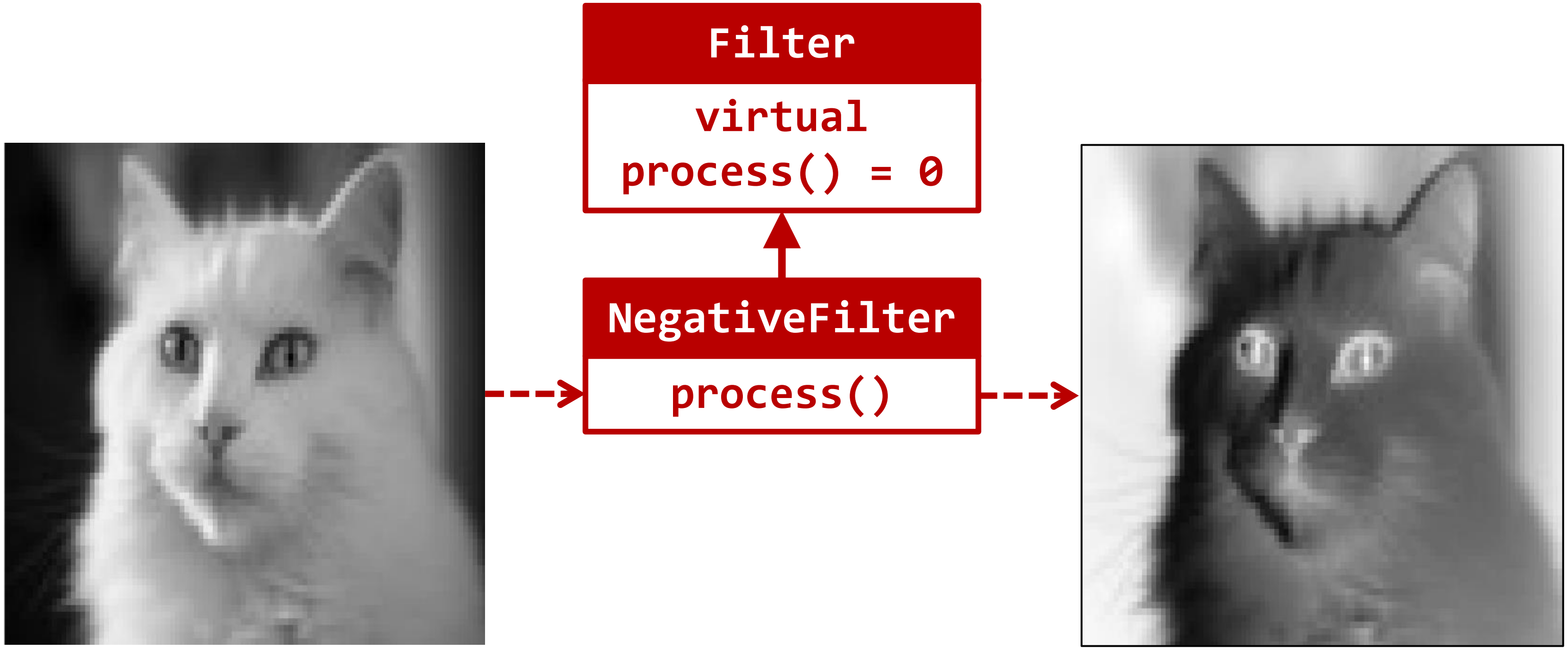
virtual process() = 0

NegativeFilter

process()

Bitmaps and Filters

- `BitmapImage` → `NegativeFilter` → `BitmapImage`



Filter

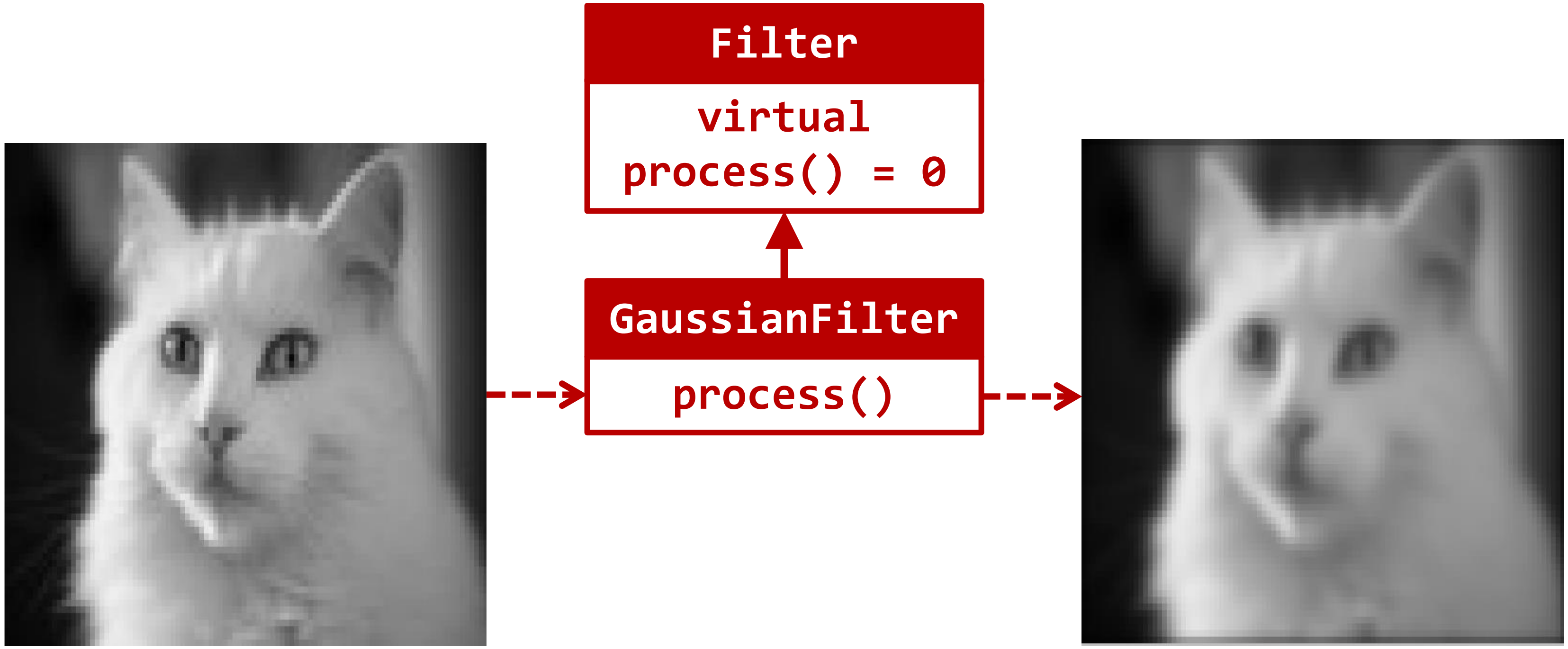
virtual process() = 0

NegativeFilter

process()

Bitmaps and Filters

- BitmapImage → GaussianFilter → BitmapImage



Filter

virtual process() = 0

GaussianFilter

process()

Bitmaps and Filters

- **`BitmapImage`**
  - A class for representing a grayscale bitmap image
  - Dynamically allocate memory for storing an image of an arbitrary size
  - Constructor, copy constructor, copy assignment operator, move constructor, move assignment operator, destructor

- **`Filter`**
  - An abstract base class for representing a general filter applicable to BitmapImage objects
  - Two virtual functions: **`getName()`**, **`process()`**

- **`NegativeFilter`**, **`GaussianFilter`**, **`ZoomInFilter`**, **`ZoomOutFilter`**
  - Derived classes from Filter for representing specific, pre-defined filters

- **`Free1Filter`**, **`Free2Filter`**
  - Two additional derived filters you freely select, define and implement

BitmapImage

- **Public interface (you have to define)**

  ```
  BitmapImage(int w, int h);
  BitmapImage(const std::string& path);
  ~BitmapImage();

  BitmapImage(const BitmapImage& im);
  BitmapImage(BitmapImage&& im);

  BitmapImage& operator=(const BitmapImage& im);
  BitmapImage& operator=(BitmapImage&& im);

  void setPixel(int x, int y, double v);
  double getPixel(int x, int y) const;
  void clear();
  ```

BitmapImage

- **Public interface (pre-defined)**
  - `bool loadPGM(const std::string& path);`
  - `bool savePGM(const std::string& path) const;`

  - `inline int getWidth() const { return width; }`
  - `inline int getHeight() const { return height; }`

- **Private data (pre-defined)**
  - `int width;`
  - `int height;`
  - `double* bitmap;`

`BitmapImage`

- **`BitmapImage(int w, int h);`**
  - Initialize the **`width`** and **`height`** as **`w`** and **`h`**
  - Allocate memory space for storing the entire pixel values

- **`BitmapImage(const std::string& path);`**
  - Initialize **`width`**, **`height`**, and **`bitmap`** as **`0`**, **`0`**, and **`nullptr`**
  - Call **`loadPGM(path)`**

- **`~BitmapImage();`**
  - Deallocate the memory space for pixel values

`BitmapImage`

- **`BitmapImage(const BitmapImage& im);`**
  - Initialize **`width`** and **`height`** as **`im.width`** and **`im.height`**
  - Allocate memory space for storing the entire pixel values
  - Copy the pixel values from the image stored in **`im`**

- **`BitmapImage& operator=(const BitmapImage& im);`**
  - If **`this`** equals to the address of **`im`**, return **`this`** immediately
  - If the size of **`this`** image doesn't equal to **`im`**'s image size, deallocate the existing memory space and re-allocate new memory space for copying the image stored in **`im`**
  - Initialize width and height as **`im.width`** and **`im.height`**
  - Copy the pixel values from the image stored in **`im`**
  - Return **`this`** object

BitmapImage

- **BitmapImage(BitmapImage&& im);**
  - Initialize **width**, **height**, and **bitmap** as **im.width**, **im.height**, and **im.bitmap**, respectively
  - Set **im.width**, **im.height**, **im.bitmap** to **0**, **0**, **nullptr**

- **BitmapImage& operator=(BitmapImage&& im);**
  - Deallocate the existing memory space
  - Initialize **width**, **height**, and **bitmap** as **im.width**, **im.height**, and **im.bitmap**, respectively
  - Set **im.width**, **im.height**, **im.bitmap** to **0**, **0**, **nullptr**
  - Return **this** object

`BitmapImage`

- **`void setPixel(int x, int y, double v);`**
  - If the given coordinates are out of boundary, just return
  - Otherwise, set the pixel value at **`(x, y)`** to **`v`**

- **`double getPixel(int x, int y) const;`**
  - If the given coordinates are out of boundary, return **`0.0`**
  - Otherwise, return the pixel value at **`(x, y)`**

- **`void clear();`**
  - Set all the pixel values in the image to **`0.0`**

Filter

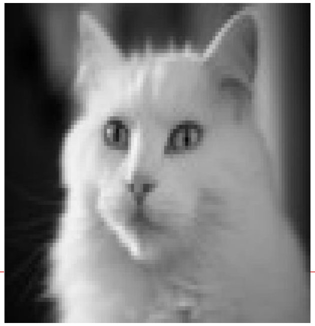
- **Declare only pure virtual functions**
  - Make this an abstract base class, which cannot be instantiated
  - Derived classes must implement these functions

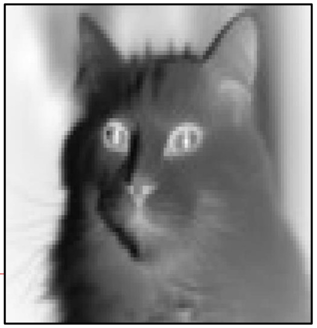
- **Function declarations**
  - `virtual std::string getName() = 0;`
  - `virtual BitmapImage process(const BitmapImage& im) = 0;`

NegativeFilter

- **`std::string getName();`**
  - Return “Negative”

- **`BitmapImage process(const BitmapImage& im);`**
  - Create a **`BitmapImage`** object such that its **`width`** and **`height`** are set to **`im.width`** and **`im.height`**
  - Set the image of the new object as a negative image of **`im`** object
    - Original pixel value at **`(x, y)`** in the **`im`** object: **`v`**
    - New pixel value at **`(x, y)`** in the new object: **`(1.0-v)`**

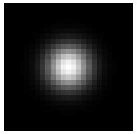


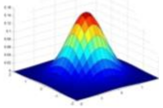


GaussianFilter

- **`std::string getName();`**
  - Return “Gaussian”

- **`BitmapImage process(const BitmapImage& im);`**
  - Create a **`BitmapImage`** object such that its **`width`** and **`height`** are set to **`im.width`** and **`im.height`**
  - Apply convolution operator with the following 5x5 Gaussian kernel

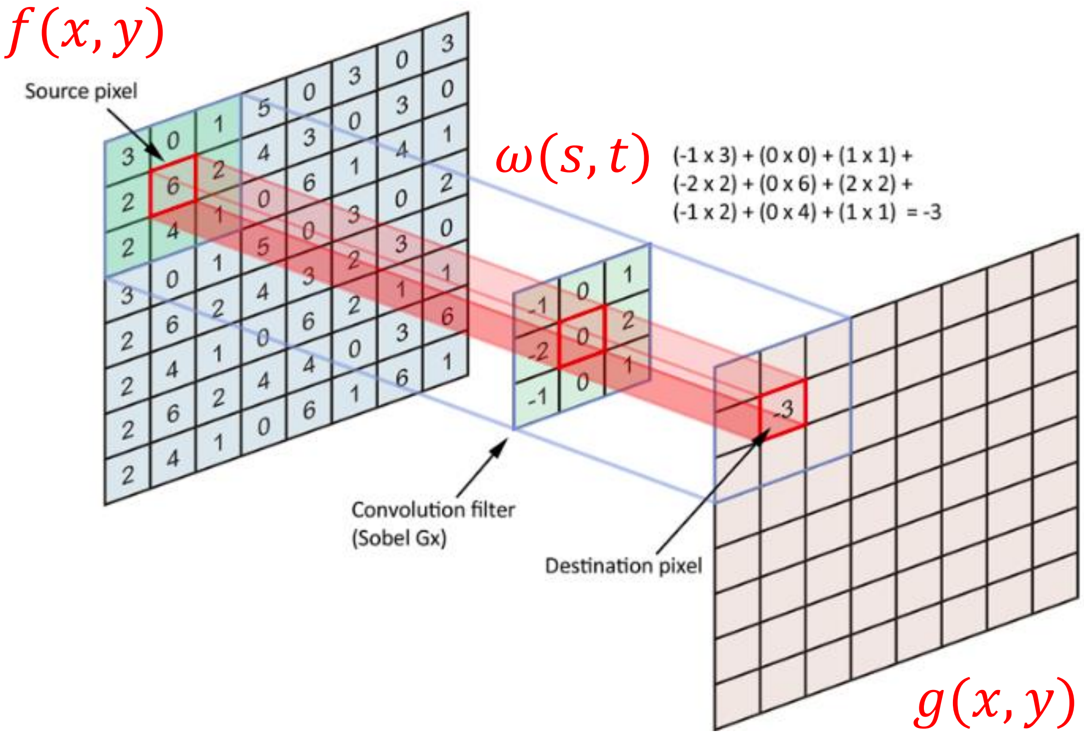




0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

Convolution

$$g(x,y) = \omega * f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t)$$



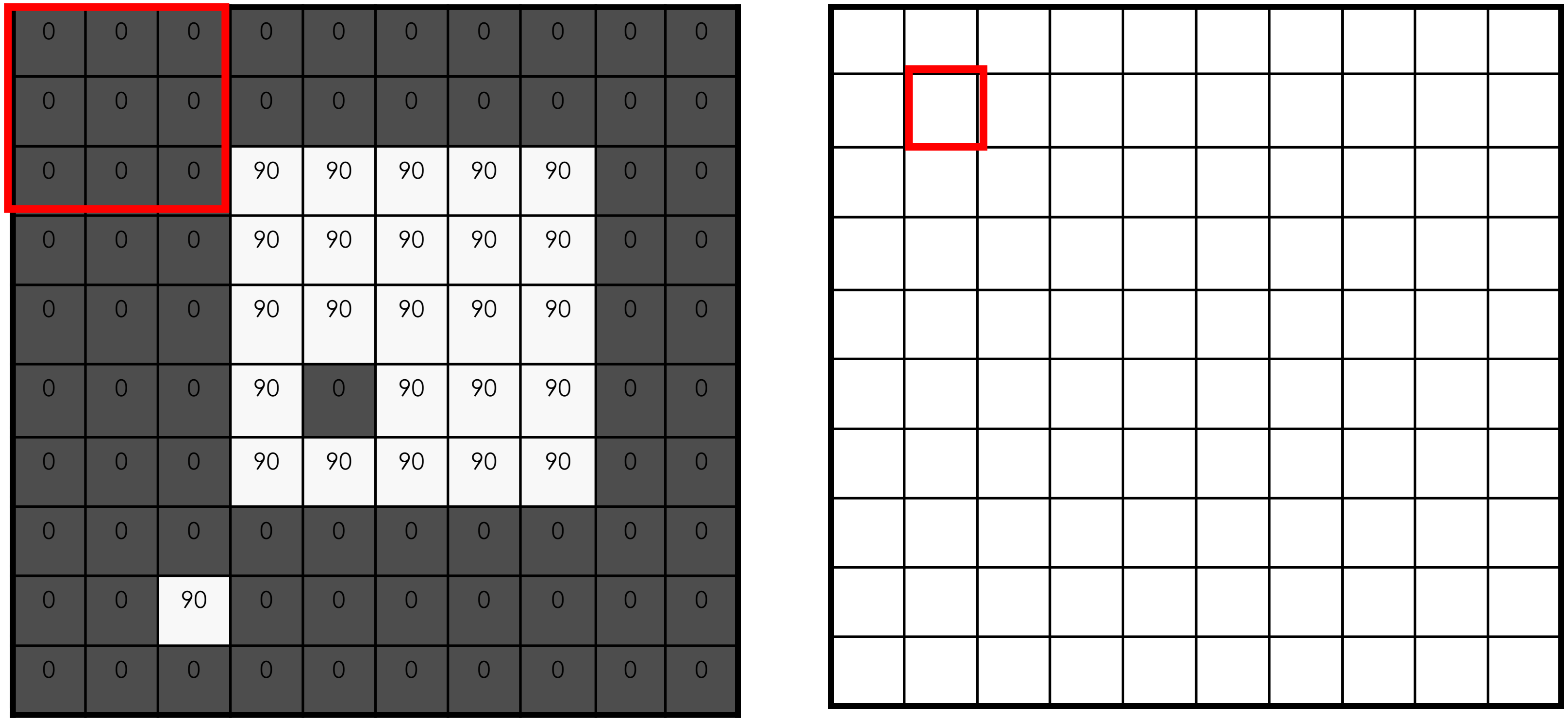
(-1 x 3) + (0 x 0) + (1 x 1) +
(-2 x 2) + (0 x 6) + (2 x 2) +
(-1 x 2) + (0 x 4) + (1 x 1) = -3

Convolution

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$ $h[.,.]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

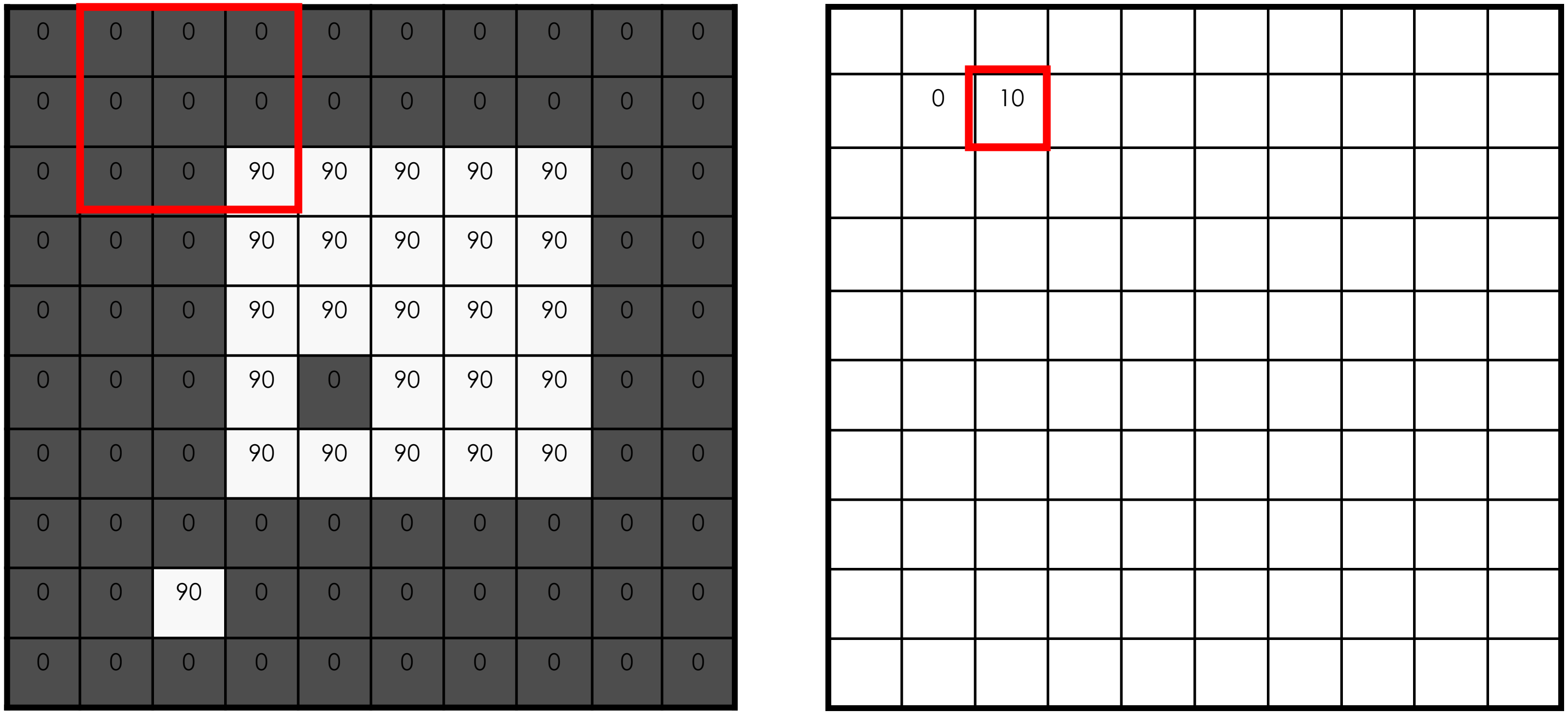
$$h[m,n]=\sum_{k,l} g[k,l]\; f[m+k,n+l]$$

Credit: S. Seitz

Convolution

$g[\cdot,\cdot]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1



$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

	0	10							

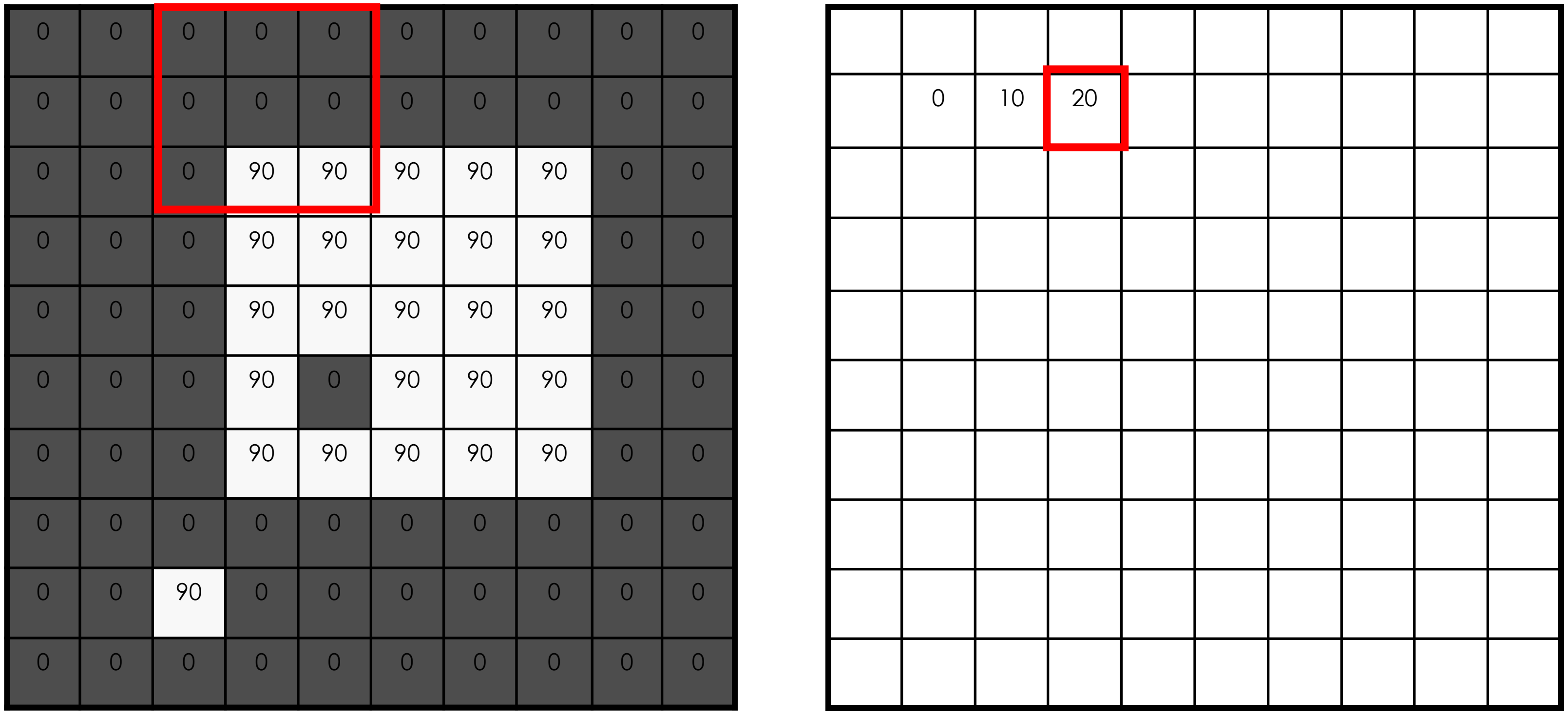
$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Convolution

$$g[\cdot\,,\cdot\,]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$f[.,.]$ $h[.,.]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

	0	10	20						

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Convolution

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

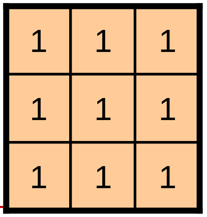
	0	10	20	30					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

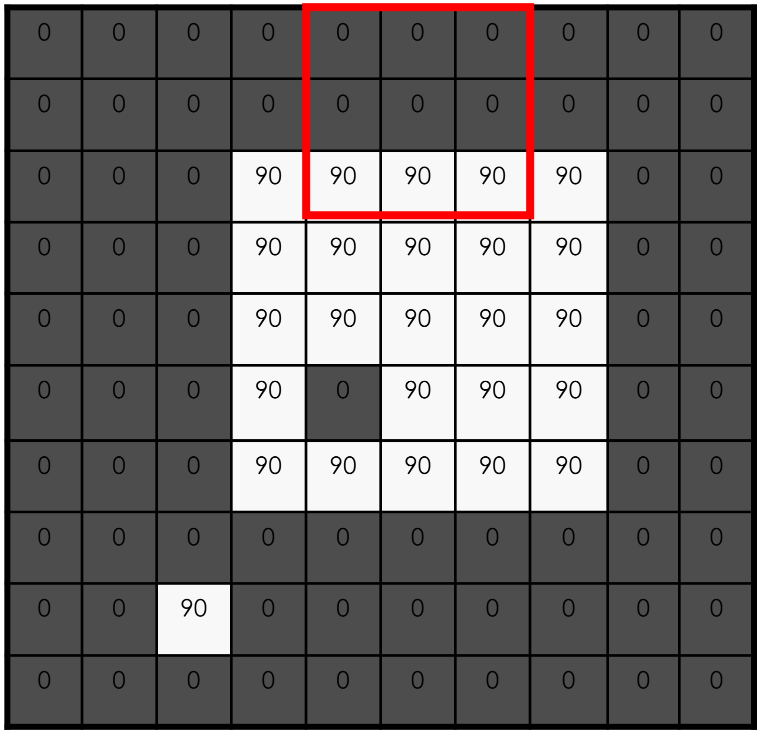
Convolution

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$



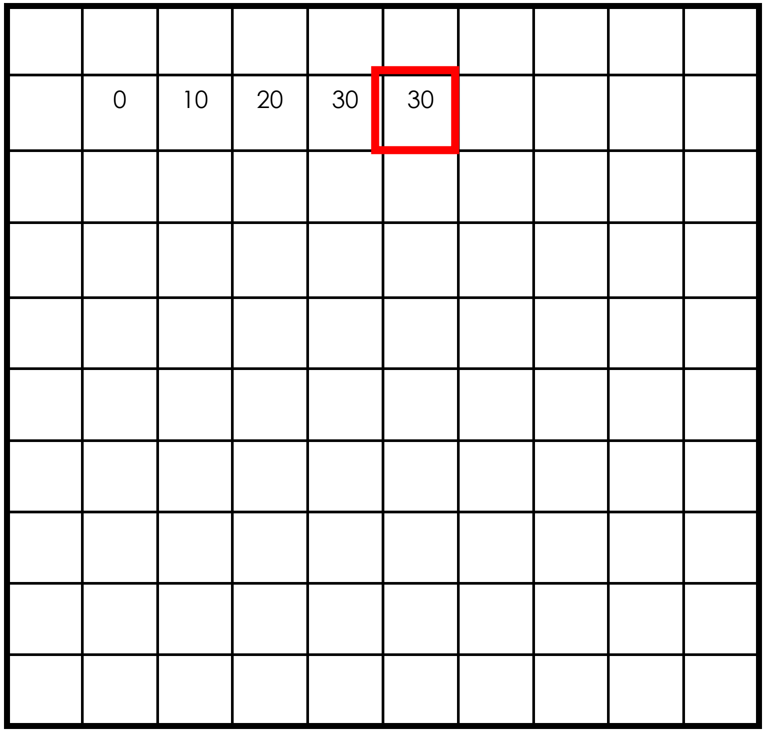
1	1	1
1	1	1
1	1	1

$f\,[.,.]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$



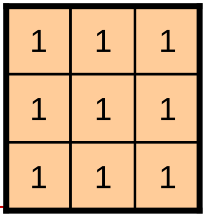
	0	10	20	30	30				

$$h[m,n]=\sum_{k,l} g[k,l]\; f[m+k,n+l]$$

Credit: S. Seitz

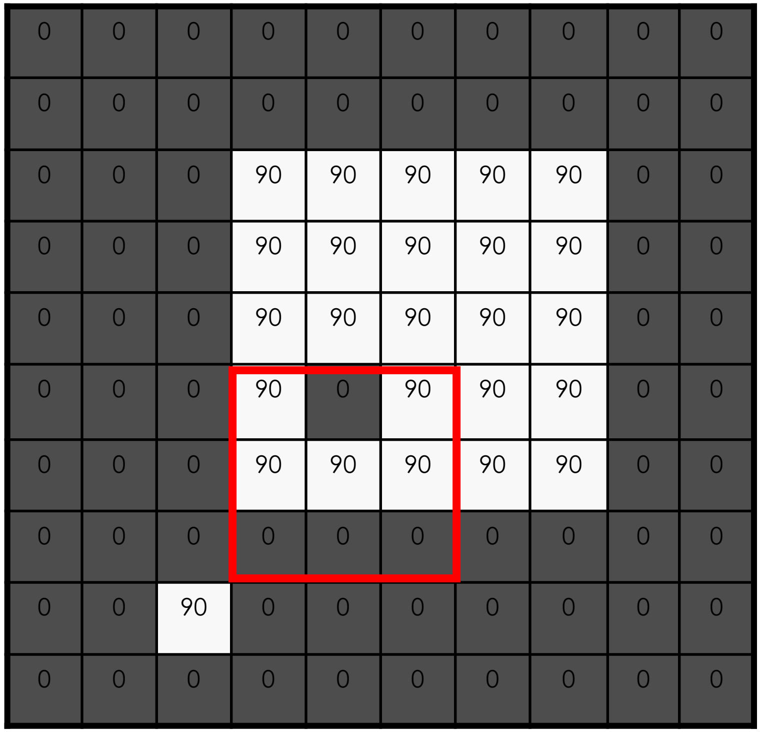
Convolution

$g[\cdot\,,\cdot\,]\ \frac{1}{9}$



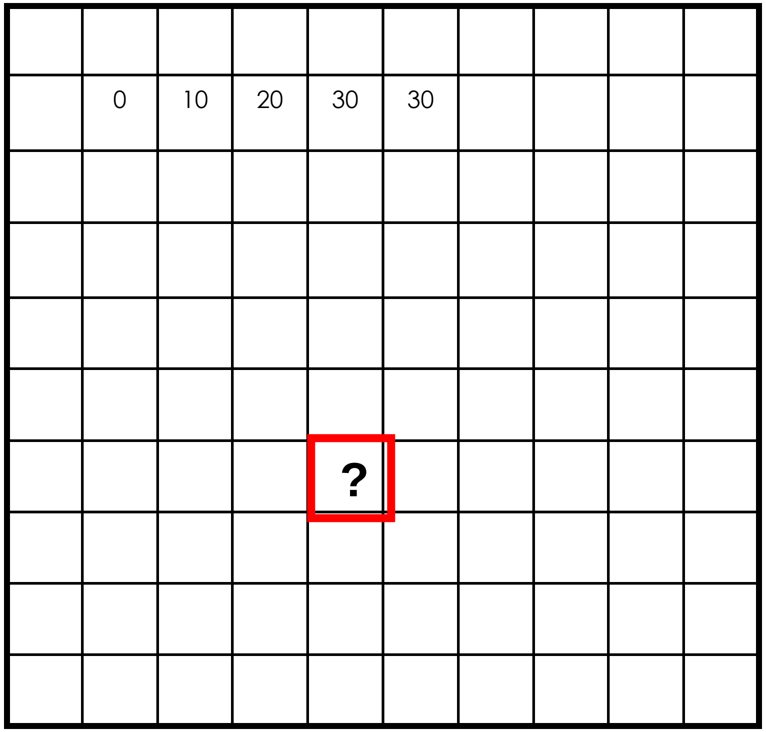
1	1	1
1	1	1
1	1	1

$f[.,.]$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$



	0	10	20	30	30				
				?					

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

Convolution

$$g[\cdot,\cdot]\ \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$

	0	10	20	30	30				
						?			
				50					

$$h[m,n]=\sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Credit: S. Seitz

$$g[\cdot\,,\cdot\,]\quad \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Convolution

$$f\,[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n]=\sum_{k,l} g[k,l]\; f\,[m+k,n+l]$$

Credit: S. Seitz

$$g[\cdot\,,\cdot\,] \quad \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

Convolution

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$

0									
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$h[m,n] = \sum_{k,l} g[k,l]\; f[m+k,n+l]$$

Credit: S. Seitz

$$g[\cdot\,,\cdot\,] \quad \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Convolution

$$f[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$h[.,.]$$

0	0	0	0	0	0	0	0	0	0
0	0	10	20	30	30	30	20	10	0
0	0	20	40	60	60	60	40	20	0
0	0	30	60	90	90	90	60	30	0
0	0	30	50	80	80	90	60	30	0
0	0	30	50	80	80	90	60	30	0
0	0	20	30	50	50	60	40	20	0
0	10	20	30	30	30	30	20	10	0
0	10	10	10	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

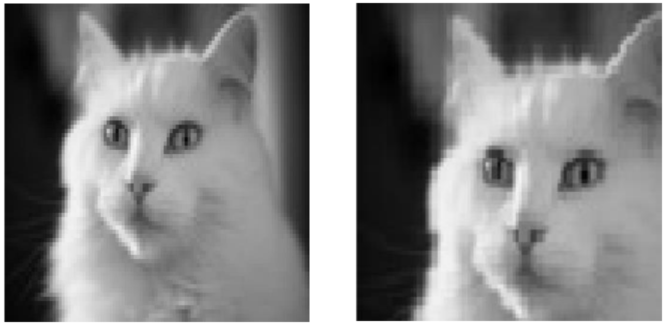
$$h[m,n] = \sum_{k,l} g[k,l]\; f[m+k, n+l]$$

Credit: S. Seitz

ZoomInFilter

- **std::string getName();**
  - Return “Zoom In”

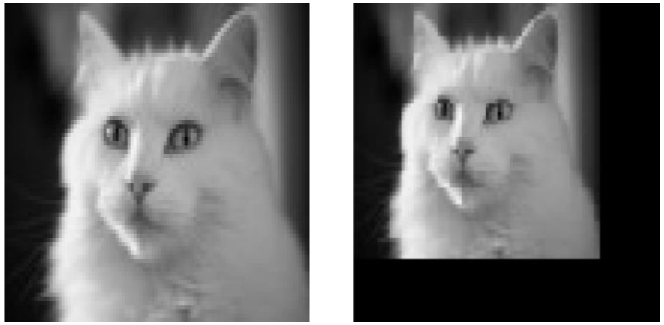
- **BitmapImage process(const BitmapImage& im);**
  - Create a **BitmapImage** object such that its **width** and **height** are set to **im.width** and **im.height**
  - Set the image of the new object as an enlarged image of **im** object
    - New pixel at **(x, y)** = Original pixel at **(x*0.8, y*0.8)**



ZoomOutFilter

- **std::string getName();**
  - Return “Zoom Out”

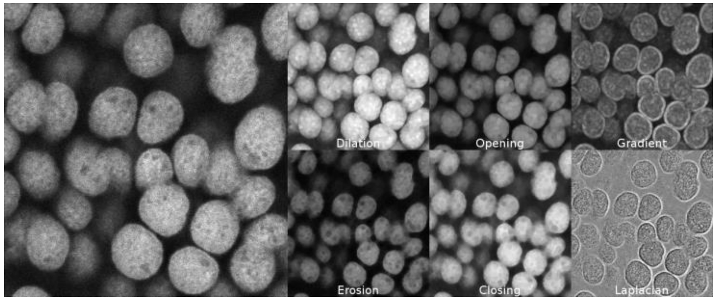
- **BitmapImage process(const BitmapImage& im);**
  - Create a **BitmapImage** object such that its **width** and **height** are set to **im.width** and **im.height**
  - Set the image of the new object as a downsized image of **im** object
    - New pixel at **(x, y)** = Original pixel at **(x*1.25, y*1.25)**



Free1Filter, Free2Filter

- **std::string getName();**
  - Return the name of the filter you selected

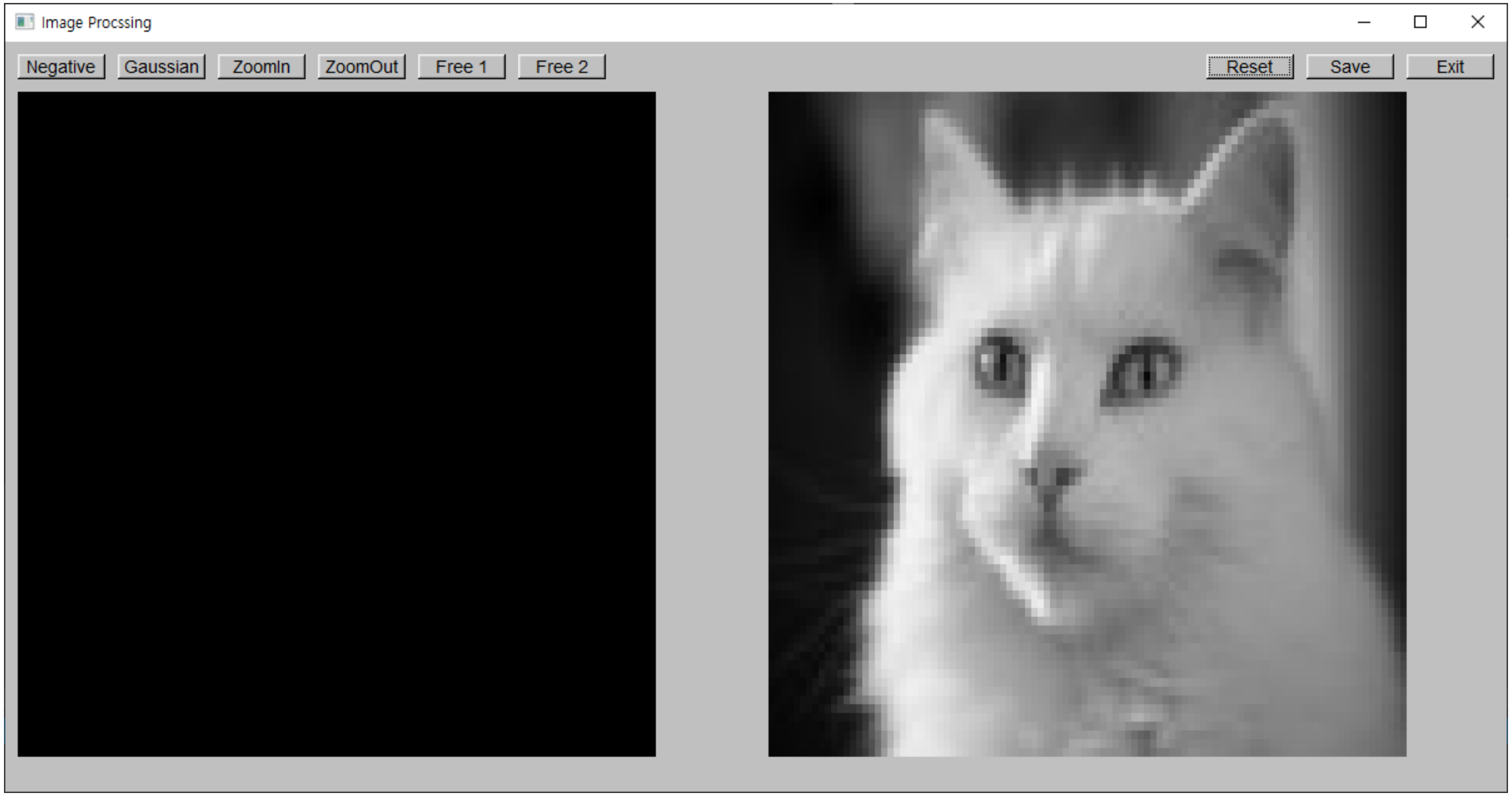
- **BitmapImage process(const BitmapImage& im);**
  - Create a **BitmapImage** object such that its **width** and **height** are set to **im.width** and **im.height**
  - Set the image of the new object as the filtered image of **im** object as you selected, defined, and implemented



https://en.wikipedia.org/wiki/Digital_image_processing
https://en.wikipedia.org/wiki/Kernel_(image_processing)
https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

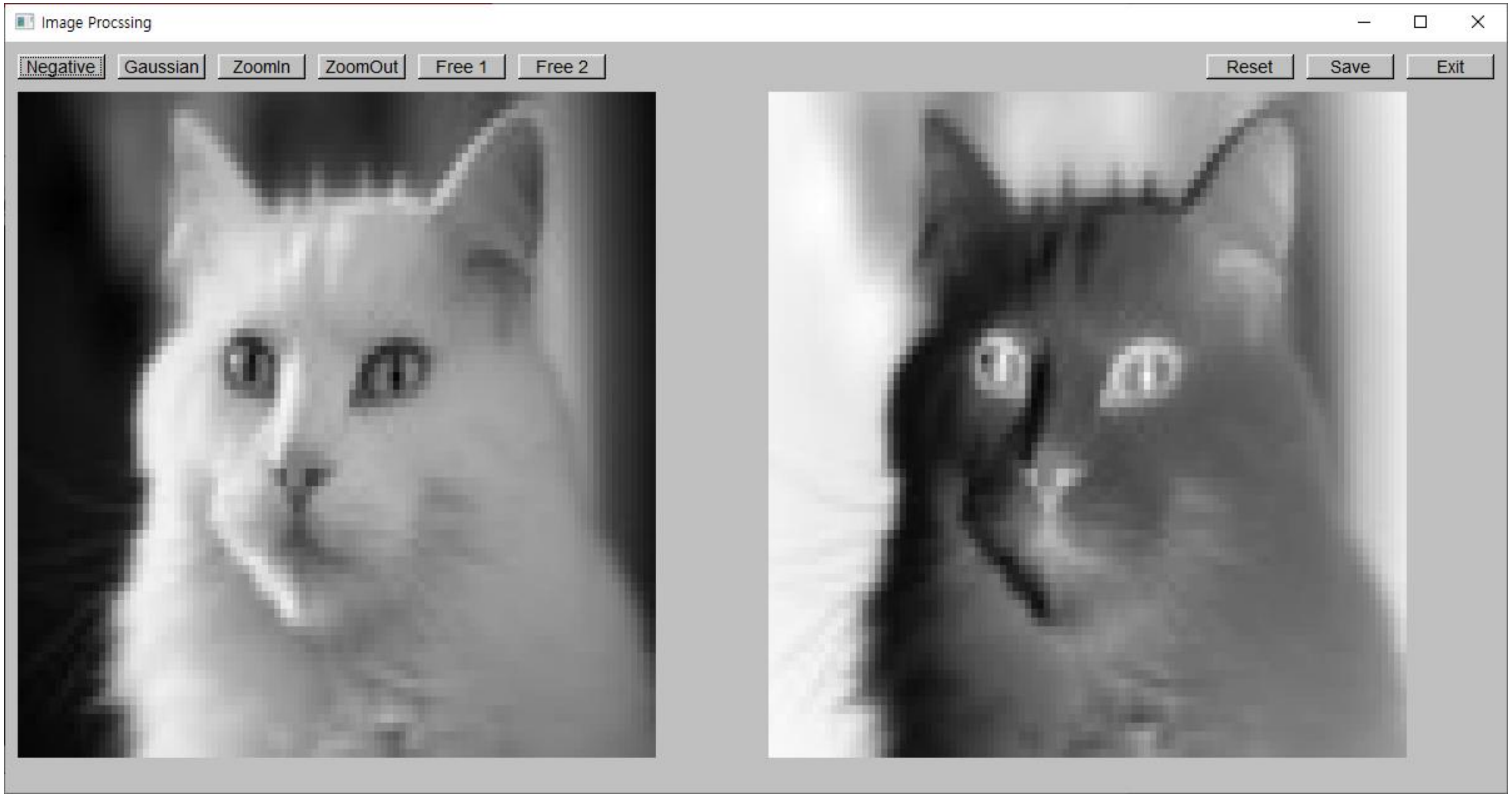
Test with Pre-defined GUI Application

- □ Pressing each filter button copies the right image to the left, processes the left image, and copies the result to the right



Test with Pre-defined GUI Application

- Pressing a filter button copies the right image to the left, filters the image, and presents the filtered image to the right



Submission

- **Report**
  - Title page
    - Course title, submission date, affiliation, student ID, full name
  - Explain how you implemented in detail
    - **`BitmapImage`** class (**`BitmapImage.cpp/.h`**)
    - **`Filter`** classes (**`Filter.cpp/.h`**)
  - Demonstrate the correctness of your class, focusing on the following functions:
    - Copy/move constructors, copy/move assignment operators, destructor of **`BitmapImage`**
    - **`process()`** functions implemented in **`NegativeFilter`**, **`GaussianFilter`**, **`ZoomInFilter`**, **`ZoomOutFilter`**, **`Free1Filter`**, and **`Free2Filter`**
  - For each additional feature, if exist, explain what it is and how you implemented it
    - e.g. additional filters, improved quality (**`ZoomInFilter`**, **`ZoomOutFilter`**), colors, etc.
  - Conclude with some comments on your work
    - Key challenges you have successfully tackled
    - Limitations you hope to address in the future

Submission

- **Compress your code and report into a single `*.zip` file**
  - Code
    - The entire project folder including **`*.sln`**, **`*.cpp`**, **`*.h`**, **`*.jpg`**, etc.
    - Remove unnecessary folders such as **`.vs`** and **`Debug`**
    - The grader should be able to open the **`*.sln`** and build/run the project immediately without any problems
  - Report
    - A single **`*.pdf`** file
    - You should convert your word format (**`*.hwp`**, **`*.doc`**, **`*.docx`**) to PDF format (**`*.pdf`**) before zipping
  - Name your zip file as your student ID
    - ex) **`2012726055.zip`**

- **Upload to homework assignment menu in KLAS**

- **Due at 6/19 (Sat), 11:59 PM**