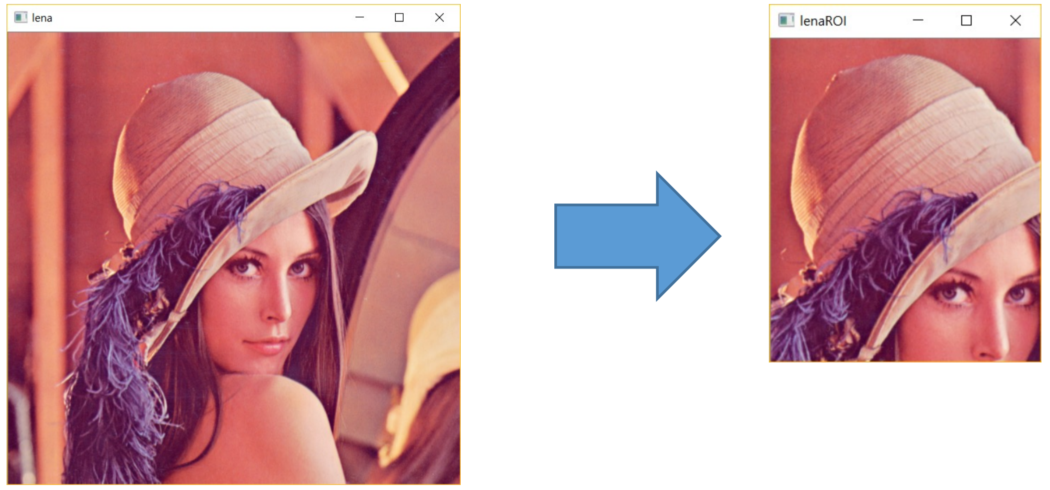
ROI Setting

Sung Soo Hwang

Mat Operator

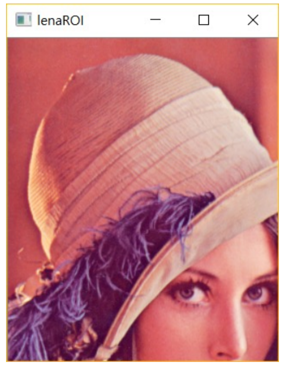
- ROI(Region of Interest)
  - A sub-region in an image that we are interested in



Mat Operator

- ROI(Region of Interest)
  - Example code

```
int main() {
    Mat image = imread("lena.png");
    Rect rect(100, 30, 250, 300);
    Mat rect_roi = image(rect);
    imshow("rectROI", rect_roi);

    waitKey(0);
}
```

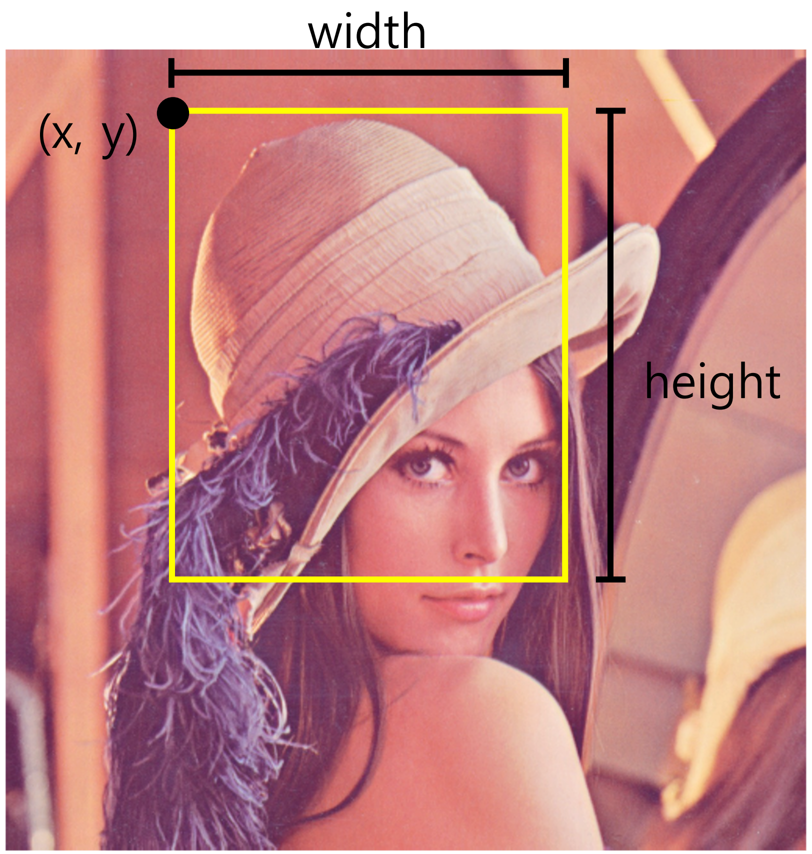


Rect(x, y, width, height)
x: x coordinate of left-top corner
y: y coordinate of left-top corner
width: width of rectangle
height: height of rectangle

Mat Operator

- ROI(Region of Interest)
  - class Rect

Rect(x, y, width, height)
x: x coordinate of left-top corner
y: y coordinate of left-top corner
width: width of rectangle
height: height of rectangle



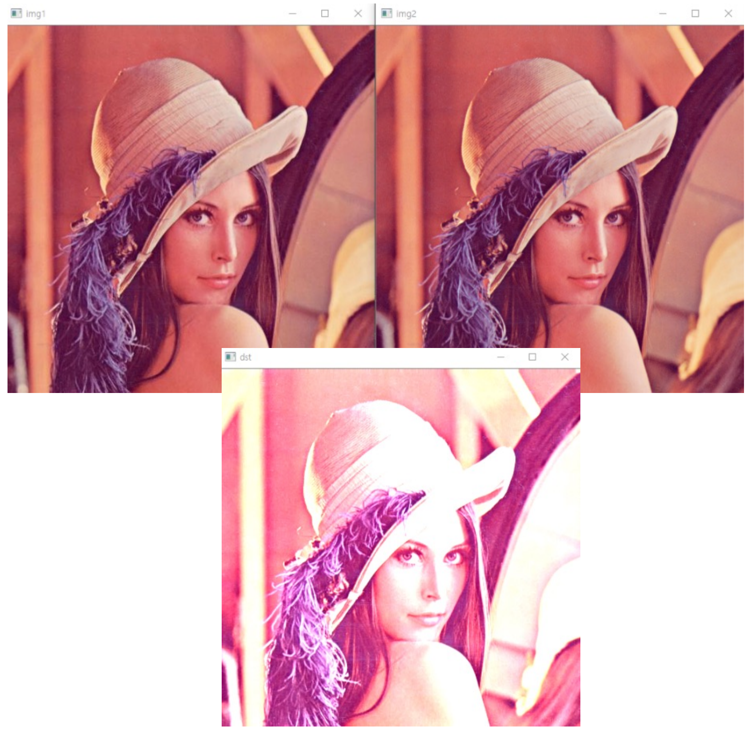
Mat Addition_Subtraction

Sung Soo Hwang

Mat Operator

- Addition/Subtraction operation
  - void add (Mat src1, Mat src2, Mat dst, Mat mask= noArray(), int dtype = -1)
    - Save the result of src1 + src2 to dst
    - mask: optional operation mask(8-bit single channel array)
    - dtype : optional depth of output array
    - dst(I) = saturate(src1(I)+src2(I)) if mask(I) != 0
  - Example code

```
int main() {
        Mat img1 = imread("lena.png");
        Mat img2 = imread("lena.png");
        Mat dst;
        add(img1, img2, dst);
        imshow("dst", dst);
        waitKey(0);
}
```



unesco
University Network
Member

Ministry of
Education

Mat Operator

- Addition/Subtraction operation
  - void scaleAdd(Mat src1, double scale, Mat src2, Mat dst)
    - dst(I) = scale * src1(I) + src2(I)
  - void absdiff(Mat src1, Mat src2, Mat dst)
    - dst(I) = saturate(| src1(I)-src2(I) |)
  - void subtract(Mat src1, Mat src2, Mat dst, Mat mask=noArray(), int dtype = -1)
    - dst(I) = saturate(src1(I) – src2(I)) if mask(I) != 0

* saturate : it prevents any pixel values not to be larger than 255 or smaller than 0. In other words, if any added/subtracted pixel value is greater than 255, then it is set to be 255. Similarly, when it is smaller than 0, then it is set to be 0.

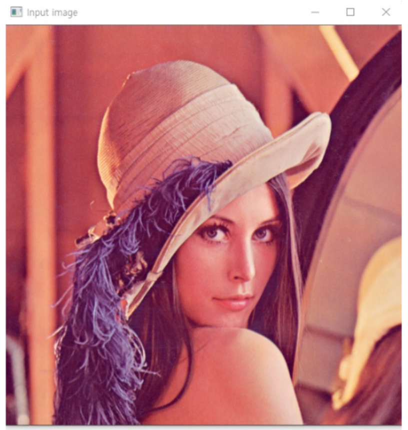
blur_GaussianBlur_Sharpening_medianBlur

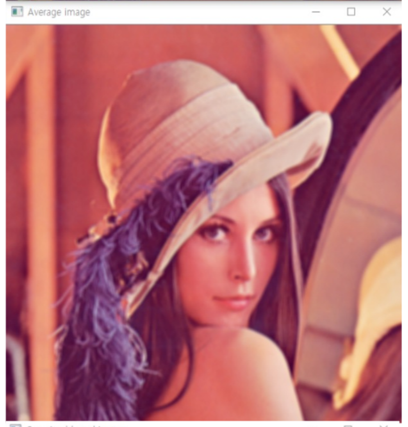
Sung Soo Hwang

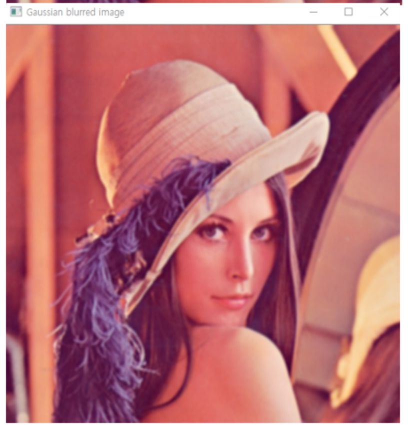
Spatial filtering

- Averaging filter
  - Example code

```
int main() {
    Mat image, AvgImg, GaussianImg;
    image = imread("lena.png");

    // Blurs an image using the normalized box filter
    // image: input image, AvgImg: output image, Size(5, 5): blurring kernel size
    blur(image, AvgImg, Size(5, 5));

    // Blurs an image using a Gaussian filter
    // image: input image, GaussianImg: output image, Size(5, 5): Gaussian kernel size
    // 1.5: Gaussian kernel standard deviation in X direction
    GaussianBlur(image, GaussianImg, Size(5, 5), 1.5);

    imshow("Input image", image);
    imshow("Average image", AvgImg);
    imshow("Gaussian blurred image", GaussianImg);

    waitKey(0);
    return 0;
}
```



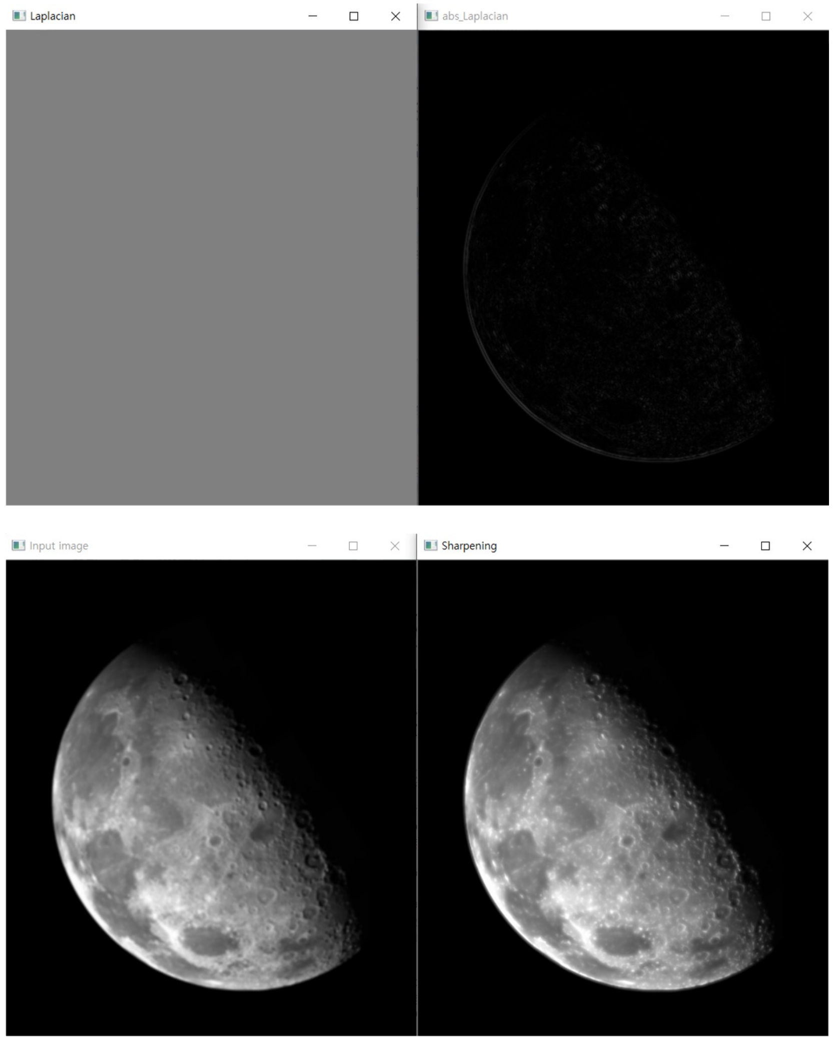




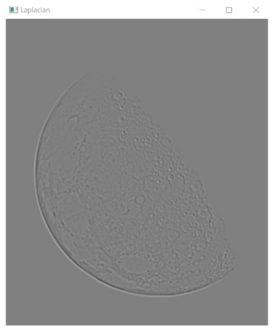
Spatial filtering

- Sharpening using second derivative
  - Example code

```
int main() {
        Mat image, laplacian, abs_laplacian, sharpening;
        image = imread("Moon.png", 0);

        GaussianBlur(image, image, Size(3, 3), 0, 0, BORDER_DEFAULT);
        // calculates the Laplacian of an image
        // image: src, laplacian: dst, CV_16S: desire depth of dst,
        // 1: aperture size used to compute second-derivative (optional)
        // 1: optional scale factor for the computed Laplacian values
        // 0: optional delta value that is added to the result
        Laplacian(image, laplacian, CV_16S, 1, 1, 0);
        convertScaleAbs(laplacian, abs_laplacian);
        sharpening = abs_laplacian + image;

        imshow("Input image", image);
        imshow("Laplacian", laplacian);
        imshow("abs_Laplacian", abs_laplacian);
        imshow("Sharpening", sharpening);

        waitKey(0);
}
```



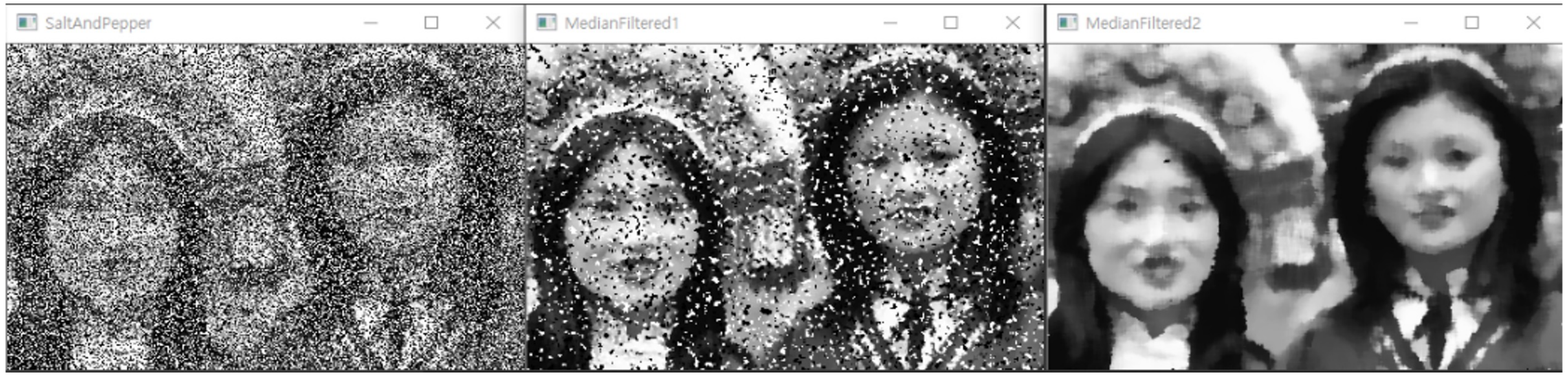
Try putting
Laplacian(image, laplacian, CV_16S, 5, 5, 5);
To visualize the changes of the function



Other filter

- Median filter
  - Example code

```
int main() {
        Mat image = imread("saltnpepper.png", 0);
        imshow("SaltAndPepper", image);
        Mat mf1, mf2;
        // Blurs an image using the median filter
        // image: src, mf1: dst, 3: aperture size(must be odd and greater than 1)
        medianBlur(image, mf1, 3);
        imshow("MedianFiltered1", mf1);

        medianBlur(image, mf2, 9);
        imshow("MedianFiltered2", mf2);

        waitKey(0);
        return 0;
}
```



http://www.fit.vutbr.cz/~vasicek/imagedb/img_corrupted/impnoise_055/189003.png