unesco
University Network
Member

Ministry of
Education

Pixel Access

Sung Soo Hwang

Pixel access

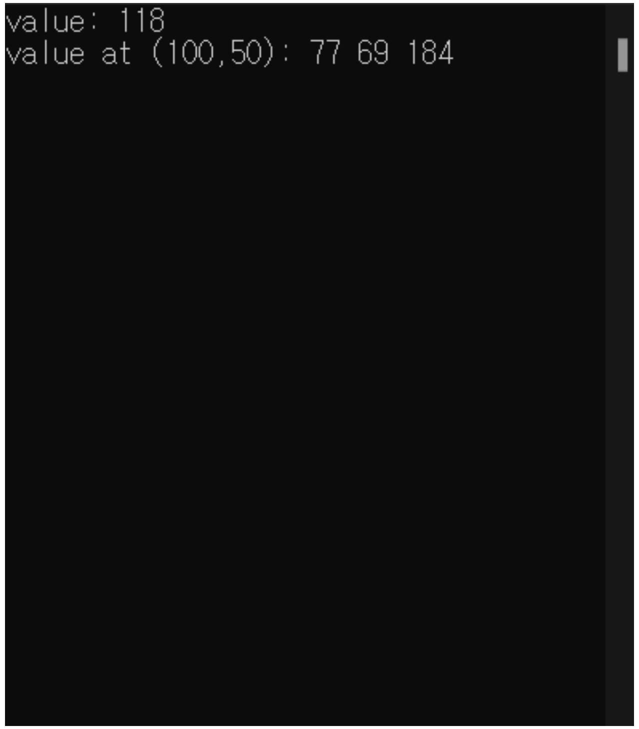
- By using at operator
    - image.at<DATA_TYPE>(WANT_ROW, WANT_COL)
        - DATA_TYPE: data type for a Mat (Ex: float, unsigned char)
        - WANT_ROW: the number of row to access
        - WANT_COL: the number column to access

Pixel access

- By using at operator
  - Example code

```
int main() {
        Mat image, image_gray;
        int value, value_B, value_G, value_R, channels;

        image = imread("lena.png");
        image_gray = imread("lena.png", 0);

        //at operator
        value = image_gray.at<uchar>(50, 100);
        cout << "value: " << value << endl;

        value_B = image.at<Vec3b>(50, 100)[0];
        value_G = image.at<Vec3b>(50, 100)[1];
        value_R = image.at<Vec3b>(50, 100)[2];
        cout << "value at (100,50): " << value_B    << " " << value_G << " " << value_R << endl;

        waitKey(0);
}
```

output



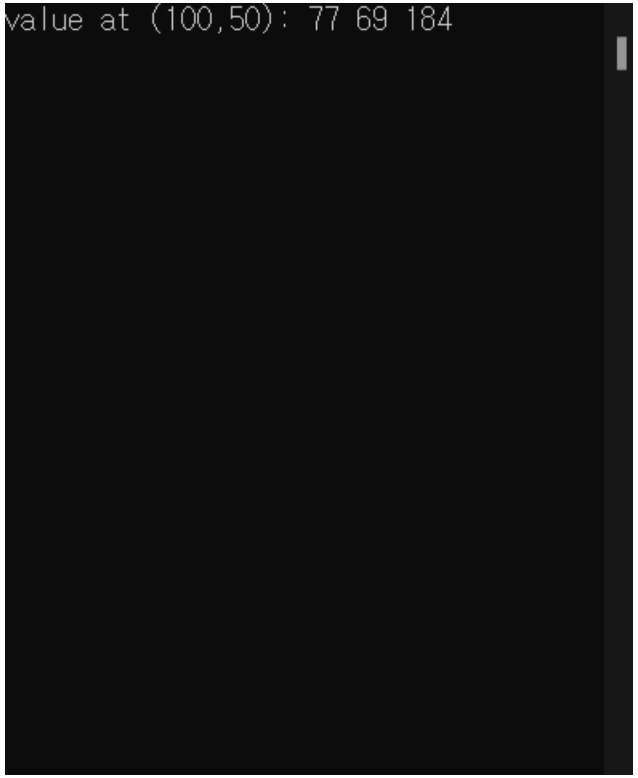
```
value: 118
value at (100,50): 77 69 184
```

Pixel access

- By using pointer
  - Faster than using at operator
  - Example code

```
int main() {
        Mat image = imread("lena.png");
        int value, value_B, value_G, value_R, channels;
        channels = image.channels();

        //pointer
        uchar* p;
        p = image.ptr<uchar>(50);
        value_B = p[100 * channels + 0];
        value_G = p[100 * channels + 1];
        value_R = p[100 * channels + 2];

        cout << "value at (100,50): " << value_B << " " << value_G << " " << value_R << endl;

        waitKey(0);
}
```

output



```
value at (100,50): 77 69 184
```

Pixel access

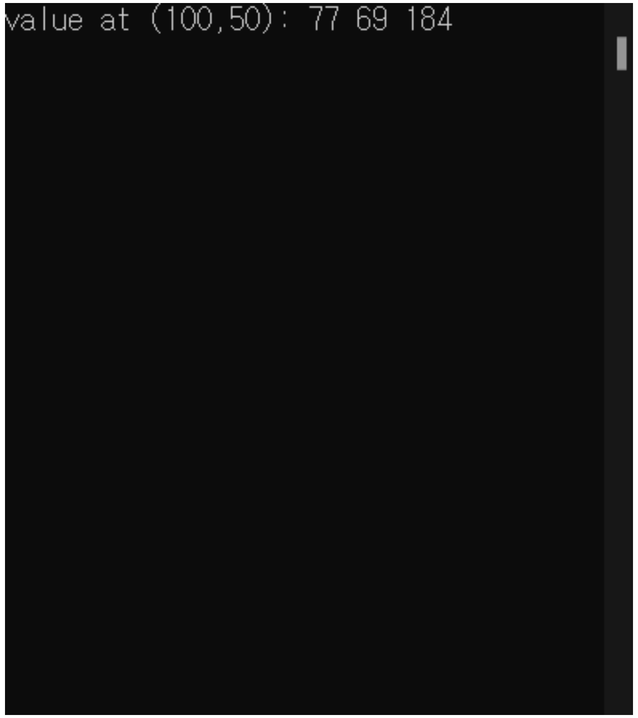
- By using data member function
  - Fast
  - Mat image(ROW, COL, CV_TYPE);
  - DATA_TYPE* data = (DATA_TYPE*)image.data;
  - data[WANT_ROW * image.cols + WANT_COL]
    - ROW : Number of Rows(Height)
    - COL : Number of Columns(Width)
    - CV_TYPE: Type type (ex: CV_8UC3 = 8 bit 3 channels)
    - DATA_TYPE: Mat Date Type(Ex float, unsigned char)
    - WANT_ROW: The row to access
    - WANT_COL: The column to access

Pixel access

- By using data member function
- Example code

```
int main() {
        Mat image;
        int value, value_B, value_G, value_R, channels;

        image = imread("lena.png");
        channels = image.channels();

        //Data member function
        uchar* data = (uchar*)image.data;
        value_B = data[(50 * image.cols + 100) * channels + 0];
        value_G = data[(50 * image.cols + 100) * channels + 1];
        value_R = data[(50 * image.cols + 100) * channels + 2];
        cout << "value at (100,50): " << value_B << " "
           << value_G << " " << value_R << endl;

        waitKey(0);
}
```

output



```
value at (100,50): 77 69 184
```

Intensity Transformation

Sung Soo Hwang

Intensity transformation

- Read an image “lena.png” as a gray-scale image
- Perform the following operations
  - Negative transformation (Result mat: negative_img)
  - Log transformation (Result mat: log_img)
    - Use log(mat a) function to perform log operation
    - To use log function, pixel type of input should be floating point
    - Also use normalize(img, img, 0, 255, NORM_MINMAX)
      - normalize img to (0~255)
  - Gamma transformation with gamma as 0.5 (Result mat gamma_img)
    - Make sure you normalize pixel values from 0 to 1.0

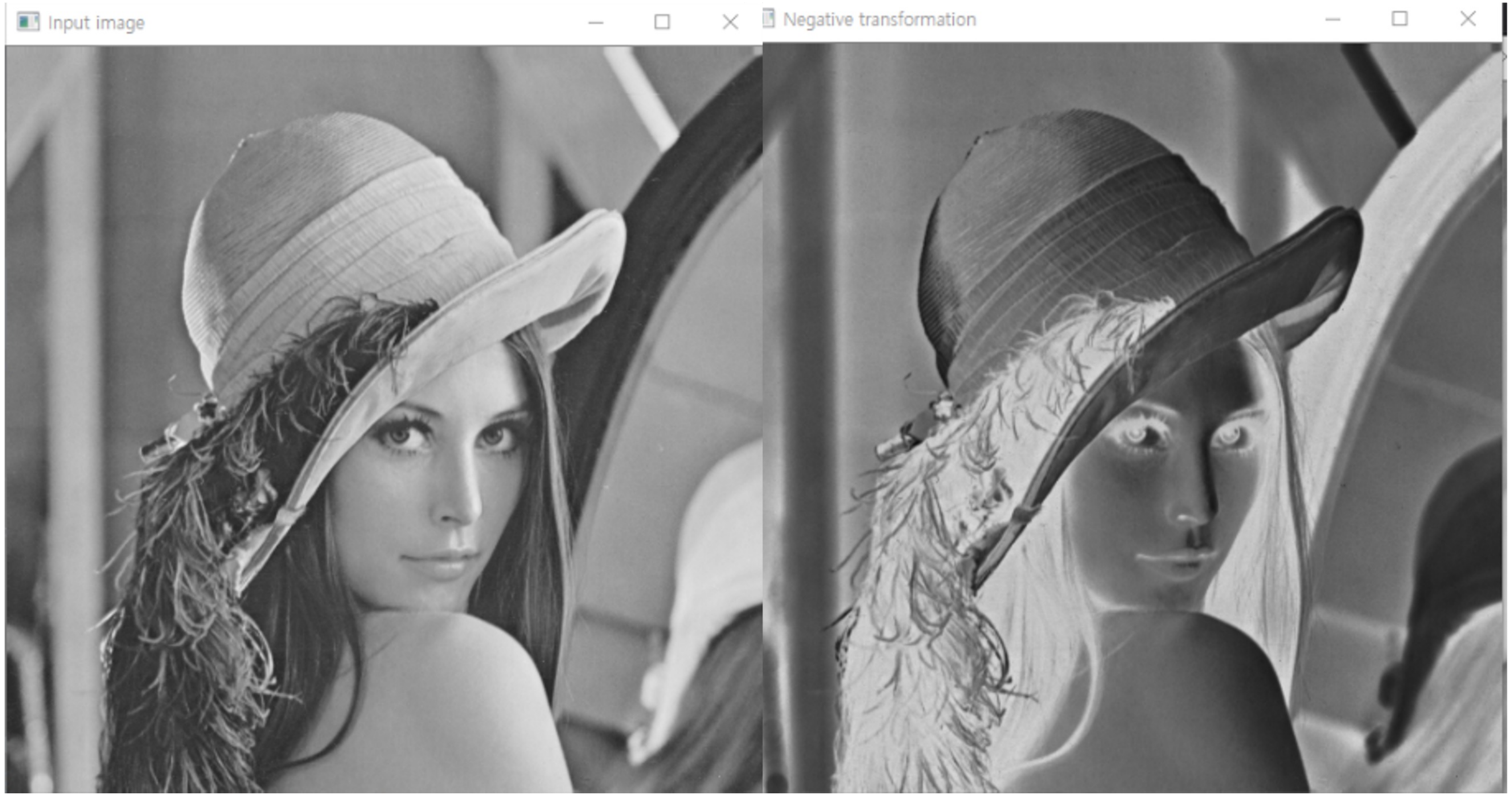
Image negative

- Example code(Image negative)

```
int main() {
        Mat image = imread("lena.png", 0);
        Mat negative_img = image.clone();
        for (int j = 0; j < image.rows; j++)
                for (int i = 0; i < image.cols; i++)
                        negative_img.at<uchar>(j, i) = 255 - image.at<uchar>(j, i);
        imshow("Input image", image);
        imshow("Negative transformation", negative_img);

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

Image negative

- Example code(Image negative)



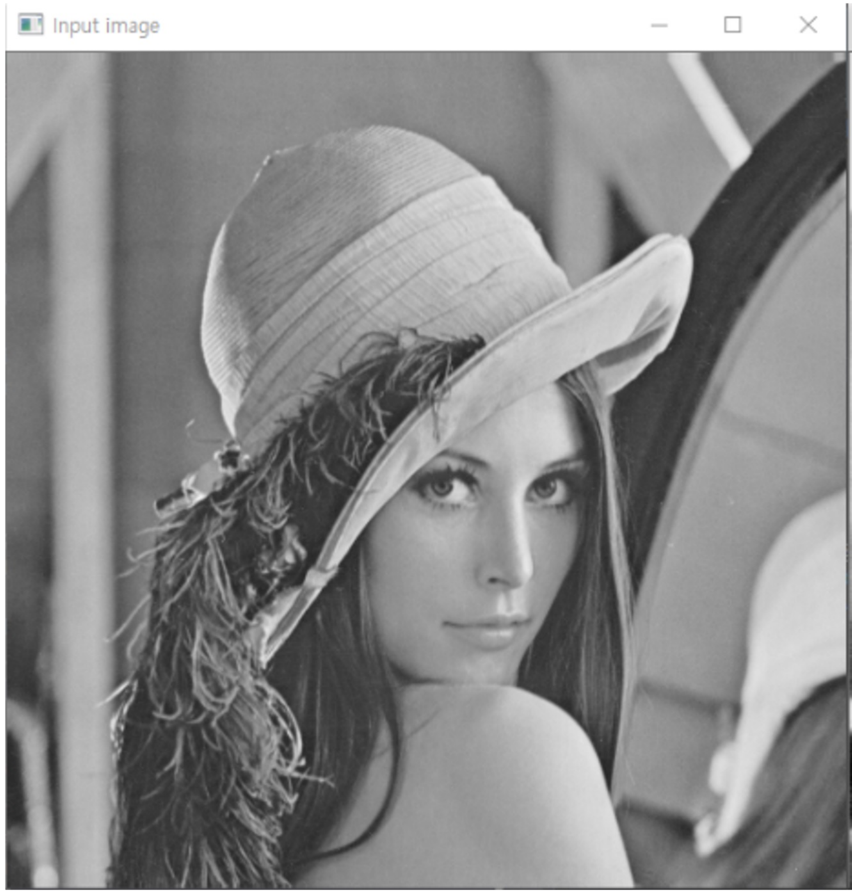
Log transformation

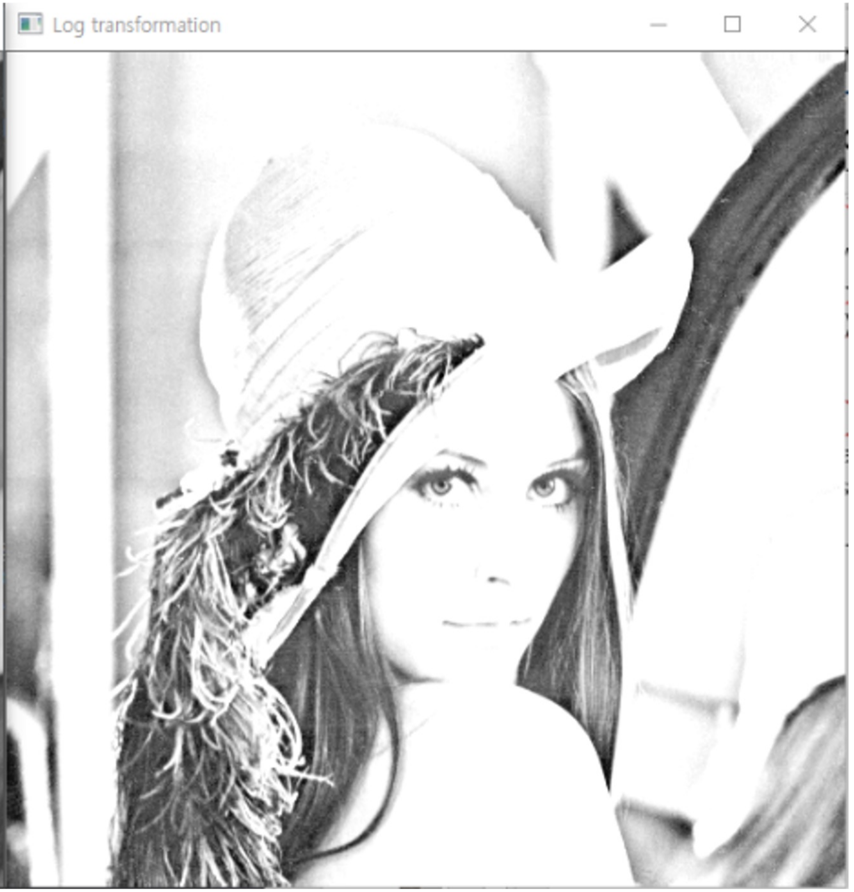
- Example code(Log transformation)

```
int main() {
    Mat image = imread("lena.png", 0);
    Mat f_img, log_img;
    double c = 1.5f; // scale constant
    image.convertTo(f_img, CV_32F);
    f_img = abs(f_img) + 1;
    log(f_img, f_img);
    normalize(f_img, f_img, 0, 255, NORM_MINMAX); // normalize image to (0~255)
    convertScaleAbs(f_img, log_img, c); // scaling by c, conversion to an unsigned 8-bit type
    imshow("Input image", image);
    imshow("Log transformation", log_img);

    waitKey(0);
}
```

Log transformation

- Example code(Log transformation)





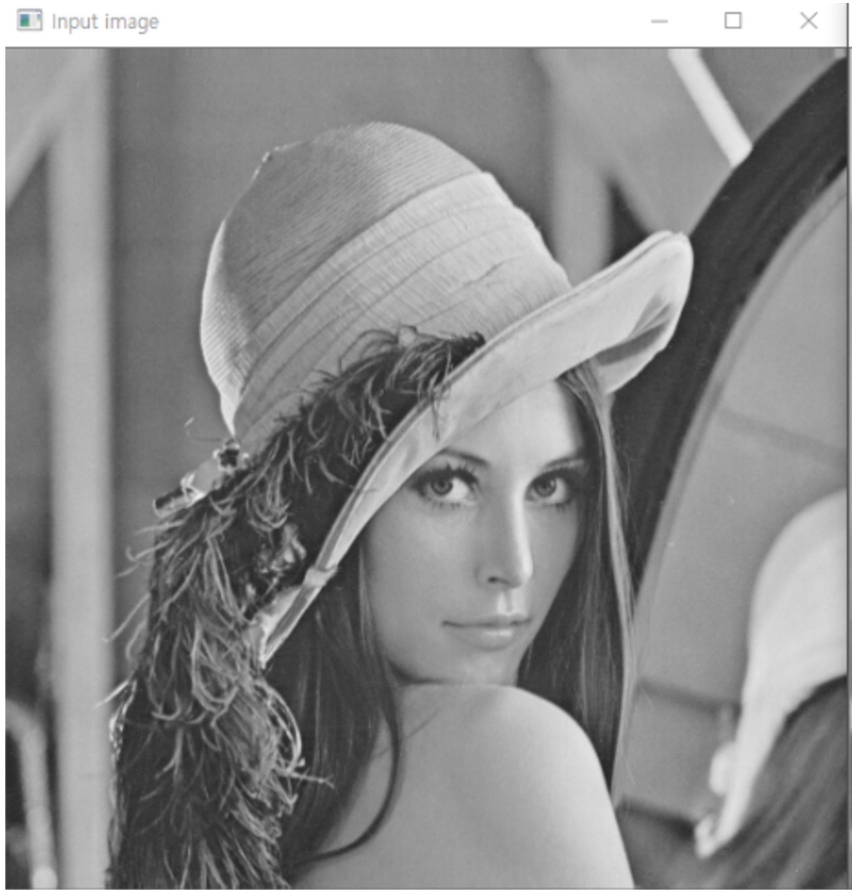
Gamma correction

- Example code(Gamma correction)

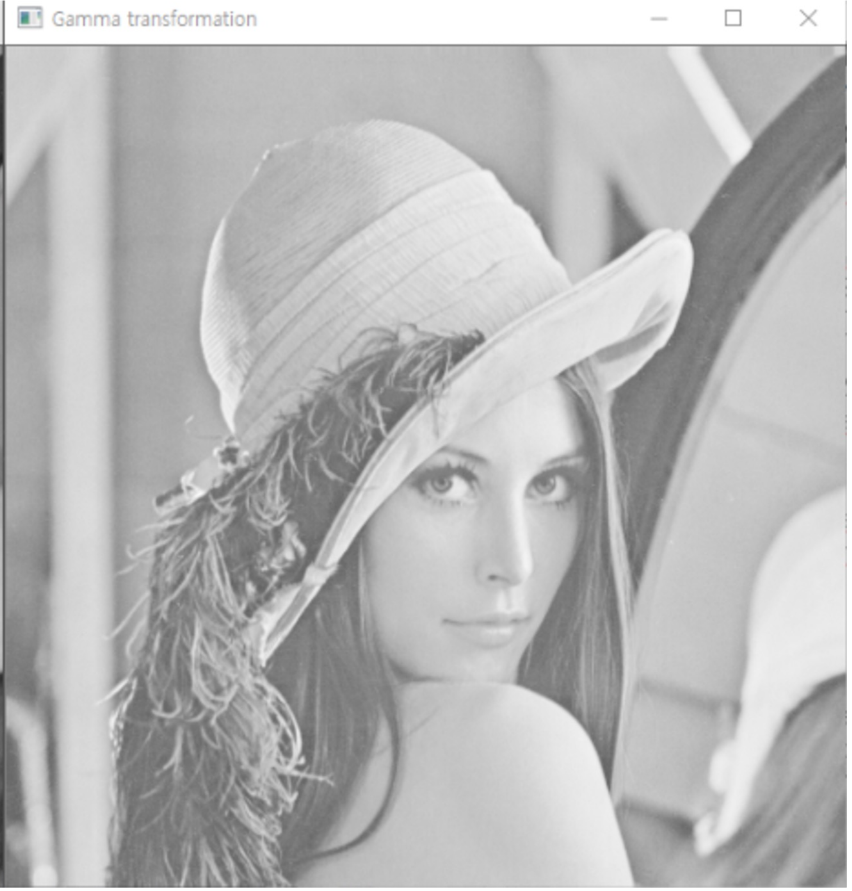
```
int main() {
    Mat image = imread("lena.png", 0);
    Mat gamma_img;
    MatIterator_<uchar> it, end;
    float gamma = 0.5;
    unsigned char pix[256];

    for (int i = 0; i < 256; i++) {
        pix[i] = saturate_cast<uchar>(pow((float)(i / 255.0), gamma) * 255.0f);
    }
    gamma_img = image.clone();

    for (int j = 0; j < image.rows; j++)
      for (int i = 0; i < image.cols; i++)
        gamma_img.at<uchar>(j, i) = pix[gamma_img.at<uchar>(j, i)];

    imshow("Input image", image);
    imshow("Gamma transformation", gamma_img);
    waitKey(0);
}
```

Gamma correction

- Example code(Gamma correction)



Input image



Gamma transformation