영상처리 실제 - 5주차 과제

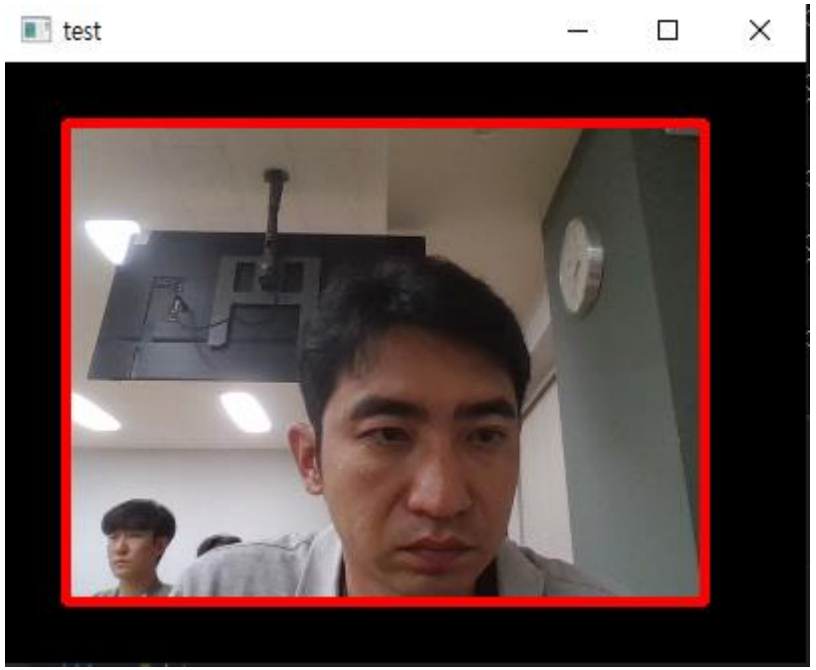
2023254009 최현동

: (5) - p.20

```
    //(5) - 20 Page
#if 1

    Mat image = imread("D:\\1.개인폴더\\2.산업인공지능학과\\2.23년2학기(석사2학기)\\2.영상처리실제\\2.과제\\1.과제\\3.Image\\logo.jpg");

    Mat bgr[3], blue_img, red_img, green_img, zero(image.size(), CV_8U, Scalar(0));
    split(image, bgr);

    Mat buleImage[] = { bgr[0], zero, zero };
    Mat greenImage[] = { zero, bgr[1], zero };
    Mat redImage[] = { zero, zero, bgr[2] };

    merge(buleImage, 3, blue_img);
    merge(greenImage, 3, green_img);
    merge(redImage, 3, red_img);

    imshow("image", image);
    imshow("blue_img", blue_img
    imshow("red_img", red_img);
    imshow("green_img", green_i
    waitKey(0);
#endif
```

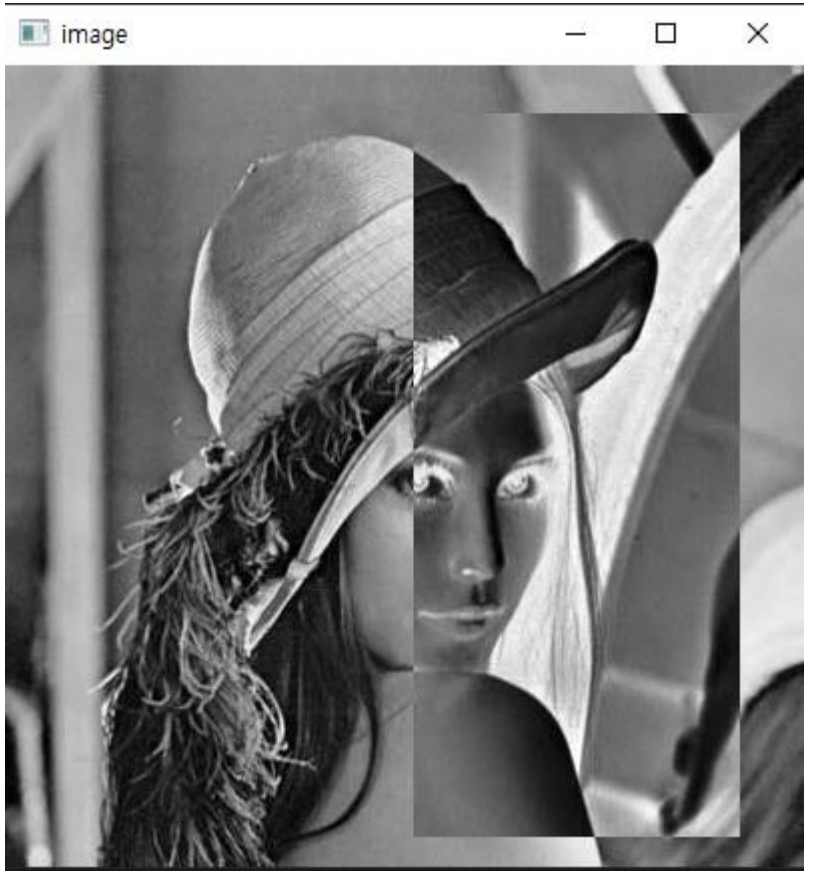
: (5) - p.21

```cpp
    //(5) - 21 Page
#if 1
    namedWindow("test", WINDOW_NORMAL);
    resizeWindow("test", 400, 300);

    VideoCapture capture(0);
    if (!capture.isOpened())
    {
        cout << "카메라가 연결 되지 않았습니다." << endl;
        exit(1);
    }

    Rect roi(30, 30, 320, 240);
    Scalar red(0, 0, 255);

    Mat backImage(300, 400, CV_8UC3, Scalar(0,0,0));

    capture.set(CAP_PROP_FRAME_WIDTH, 320);
    capture.set(CAP_PROP_FRAME_HEIGHT, 240);

    while(true)
    {
        Mat frame;

        capture.read(frame);

        Mat roiImage(backImage, roi);

        frame.copyTo(roiImage);

        rectangle(backImage, roi, red, 3); //사각형 그리기

        imshow("test", backImage);

        if (waitKey(30) >= 0) break;
    }
#endif
```



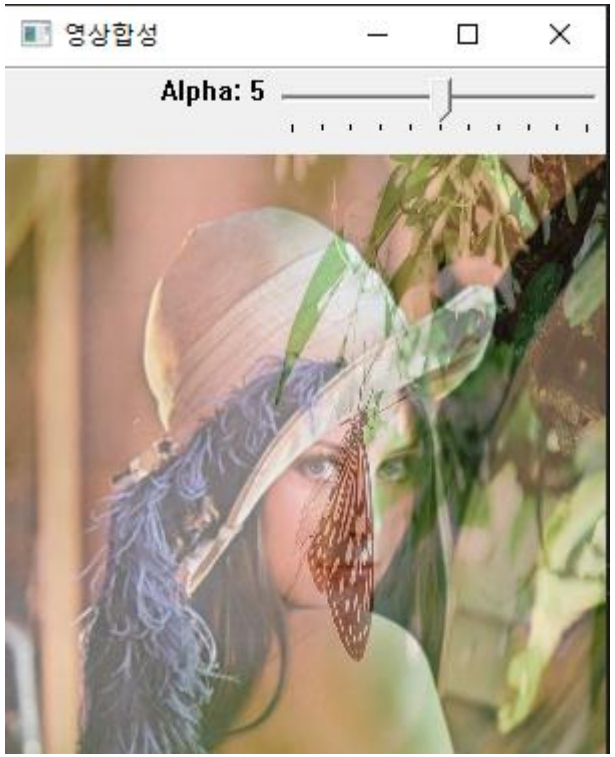
: 화소처리- p.38 – HW1

```
void onMouse_Pixel_Processing(int event, int x, int y, int flags, void* param)
{
    //화소처리 HW1
#if 1
    if (event == EVENT_LBUTTONDOWN)
    {
        // 마우스의 왼쪽 버튼을 누르면
        mx1 = x; // 사각형의 좌측 상단 좌표 저장
        my1 = y;
        cropping = true;
    }
    else if (event == EVENT_MOUSEMOVE)
    {
    }
    else if (event == EVENT_LBUTTONUP)
    {
        // 마우스의 왼쪽 버튼에서 손을 떼면
        mx2 = x; // 사각형의 우측 하단 좌표 저장
        my2 = y;
        cropping = false;
        //rectangle(img, Rect(mx1, my1, mx2 - mx1, my2 - my1), Scalar(0, 255, 0), 2);


        Mat dst(img_Pixel_Processing, (Rect(mx1, my1, mx2 - mx1, my2 - my1)));
        dst = 255 - dst;

        imshow("image", img_Pixel_Processing);
    }
#endif

    //화소처리 - p.38
    //HW1
#if 1
    img_Pixel_Processing = imread("D:\\999.Image\\lenna.jpg", IMREAD_GRAYSCALE);
    if (img_Pixel_Processing.empty())
    {
        cout << "영상을 읽을 수 없음" << endl;
    }
    imshow("image", img_Pixel_Processing);
    setMouseCallback("image", onMouse_Pixel_Processing, 0);
    waitKey(0);
#endif
```



: 화소처리- p.38 – HW2

```
void on_trackbar_Pixel_Processing(int nAlpha, void* pUserdata)
{
    double dTemp = ((double)nAlpha / 10);

    double beta = (1.0 - dTemp);

    addWeighted(img_Pixel_Processing_src1, dTemp, img_Pixel_Processing_src2, beta, 0.0, img_Pixel_Processing_dst);
    imshow(title_Pixel_Processing_HW2, img_Pixel_Processing_dst);
}
    //HW2
#if 1
    img_Pixel_Processing_src1 = imread("D:\\999.Image\\lenna.jpg");
    img_Pixel_Processing_src2 = imread("D:\\999.Image\\bug.jpg");

    Size sz1(300, 300);
    resize(img_Pixel_Processing_src1, img_Pixel_Processing_src1, sz1);
    resize(img_Pixel_Processing_src2, img_Pixel_Processing_src2, sz1);

    namedWindow(title_Pixel_Processing_HW2, WINDOW_AUTOSIZE);
    createTrackbar("Alpha", title_Pixel_Processing_HW2, &nAlpha_Pixel_Processing, 10, on_trackbar_Pixel_Processing);

    waitKey();
#endif
```



: 히스토그램- p.31 – HW1

화소값	0	1	2	3	4	5	6	7
화소수	0	0	50	60	50	20	10	0

- 스트레칭

화소값	0	1	2	3	4	5	6	7
화소수	50	60	50	60	0	20	0	10

- 평활화

화소값	0	1	2	3	4	5	6	7
화소수	0	0	50	60	50	20	10	0
누적	0	0	50	110	160	180	190	190
새로운 명도	0	2	67	148	215	242	255	255

: 히스토그램- p.31 – HW2

```
//히스토그램 - p.31
//HW2
#if 1
Mat src_Image = imread("D:\\999.Image\\lenna.jpg", IMREAD_GRAYSCALE);
Mat temp_row;
Mat temp_col;

int nDim = 0; //0 - row(↓), 1 - col(->)
reduce(src_Image, temp_row, 0, REDUCE_SUM, CV_32F);

reduce(src_Image, temp_col, 1, REDUCE_SUM, CV_32F);

int hist_row[400] = { 0 };
int hist_col[400] = { 0 };

for (int y = 0; y < temp_row.rows; y++)
{
    for (int x = 0; x < temp_row.cols; x++)
    {
        hist_row[x] = (int)temp_row.at<float>(y, x);
    }
}

for (int y = 0; y < temp_col.rows; y++)
{
    for (int x = 0; x < temp_col.cols; x++)
    {
        hist_col[y] = (int)temp_col.at<float>(y, x);
    }
}
int hist_w = src_Image.cols; //히스토그램 영상의 폭
int hist_h = src_Image.rows; //히스토그램 영사의 높이
int bin_w = cvRound((double)hist_w / 256); //빈의 폭

//히스토그램이 그려지는 영상(칼라로 정의)
Mat histImage_row(hist_h, hist_w, CV_8UC3, Scalar(255, 255, 255));
Mat histImage_col(hist_h, hist_w, CV_8UC3, Scalar(255, 255, 255));
```

```
//히스토그램의 최대값을 찾는다.
int max_row = hist_row[0];
for (int i = 1; i < 400; i++)
{
    if (max_row < hist_row[i])
    {
        max_row = hist_row[i];
    }
}
int max_col = hist_col[0];
for (int i = 1; i < 400; i++)
{
    if (max_col < hist_col[i])
    {
        max_col = hist_col[i];
    }
}

//히스토그램 배열을 최대값으로 정규화 한다.(최대값이 최대높이가 되도록)
for (int i = 0; i < 399; i++)
{
    hist_row[i] = floor(((double)hist_row[i] / max_row) * histImage_row.rows);
}

for (int i = 0; i < 399; i++)
{
    hist_col[i] = floor(((double)hist_col[i] / max_col) * histImage_col.rows);
}

//히스토그램의 값을 그린다.
for (int i = 0; i < 399; i++)
{
    line(histImage_row, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - hist_row[i]), Scalar(100, 20, 100));
}

for (int i = 0; i < 399; i++)
{
    line(histImage_col, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - hist_col[i]), Scalar(100, 100, 100));
}

imshow("src_Image", src_Image);

imshow("Histogram_row", histImage_row);

rotate(histImage_col, histImage_col, ROTATE_90_COUNTERCLOCKWISE);
imshow("Histogram_col", histImage_col);
```

: 히스토그램- p.31 – HW2

: 히스토그램- p.31 – HW3

```
void onMouse_Histogram(int event, int x, int y, int flags, void* param)
{
#if 1
    if (event == EVENT_LBUTTONDOWN)
    {
        // 마우스의 왼쪽 버튼을 누르면
        mx1 = x; // 사각형의 좌측 상단 좌표 저장
        my1 = y;
        cropping = true;
    }
    else if (event == EVENT_MOUSEMOVE)
    {
    }
    else if (event == EVENT_LBUTTONUP)
    {
        // 마우스의 왼쪽 버튼에서 손을 떼면
        mx2 = x; // 사각형의 우측 하단 좌표 저장
        my2 = y;
        cropping = false;
        Mat dst(img_Histogram, (Rect(mx1, my1, mx2 - mx1, my2 - my1)));


        vector<Mat> bgr_planes;
        split(dst, bgr_planes);
        int histSize = 256;
        float range[] = { 0, 256 };
        const float* histRange = { range };
        bool uniform = true, accumlate = false;

        Mat b_hist, g_hist, r_hist;
        calcHist(&bgr_planes[0], 1, 0, Mat(), b_hist, 1, &histSize, &histRange, uniform, accumlate);
        calcHist(&bgr_planes[1], 1, 0, Mat(), g_hist, 1, &histSize, &histRange, uniform, accumlate);
        calcHist(&bgr_planes[2], 1, 0, Mat(), r_hist, 1, &histSize, &histRange, uniform, accumlate);
        //막대그래프가 그려지는 영상을 생성한다.
        int hist_w = 512, hist_h = 400;
        int bin_w = cvRound((double)hist_w / histSize);// 상자의 폭
        Mat histImage_B(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
        Mat histImage_G(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
        Mat histImage_R(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
        //값들이 영상을 벗어나지 않도록 정규화한다.
        normalize(b_hist, b_hist, 0, histImage_B.rows, NORM_MINMAX, -1, Mat());
        normalize(g_hist, g_hist, 0, histImage_G.rows, NORM_MINMAX, -1, Mat());
        normalize(r_hist, r_hist, 0, histImage_R.rows, NORM_MINMAX, -1, Mat());

        //  히스토그램의 값을 막대로 그린다.
        for (int i = 0; i < 255; i++)
        {
            line(histImage_B, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - b_hist.at<float>(i)), Scalar(255, 0, 0));
            line(histImage_G, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - g_hist.at<float>(i)), Scalar(0, 255, 0));
            line(histImage_R, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - r_hist.at<float>(i)), Scalar(0, 0, 255));
        }
        imshow("컬러 히스토그램 - Blue", histImage_B);
        imshow("컬러 히스토그램 - Green", histImage_G);
        imshow("컬러 히스토그램 - Red", histImage_R);

    }
#endif
}
```

```
    //HW3
#if 1
    img_Histogram  = imread("D:\\999.Image\\lenna.jpg");
    if (img_Histogram.empty())
    {
        return -1;
    }
    imshow("image", img_Histogram);

    setMouseCallback("image", onMouse_Histogram, 0);
    waitKey();
#endif
```

: 히스토그램- p.31 – HW3