영상처리 실제 5주차 실습_히스토그램

2023254015 장욱진

```
#include <opencv2/opencv.hpp>

using namespace std;
using namespace cv;

void calc_Histo(const Mat& image, Mat& hist, int bins, int range_max = 256) {
	int histSize[] = { bins };
	float range[] = { 0, (float)range_max };
	int channels[] = { 0 };
	const float* ranges[] = { range };
	calcHist(&image, 1, channels, Mat(), hist, 1, histSize, ranges);
}

void draw_Histo(Mat hist, Mat& hist_img, Size size = Size(256, 200)) {

	hist_img = Mat(size, CV_8U, Scalar(255));
	float bin = (float)hist_img.cols / hist.rows;
	normalize(hist, hist, 0, hist_img.rows, NORM_MINMAX);

	for (int i = 0; i < hist.rows; i++)
	{
		float start_x = i * bin;
		float end_x = (i + 1) * bin;
		Point2f pt1(start_x, 0);
		Point2f pt2(end_x, hist.at<float>(i));

		if (pt2.y > 0) {
			rectangle(hist_img, pt1, pt2, Scalar(0), -1);
		}
		flip(hist_img, hist_img, 0);
	}
}

void create_hist(Mat img, Mat& hist, Mat& hist_img) {
	int histsize = 256, range = 256;
	calc_Histo(img, hist, histsize, range);
	draw_Histo(hist, hist_img);
}

void drawHist(int histogram[])
{
	int hist_w = 512;
	int hist_h = 400;
	int bin_w = cvRound((double)hist_w / 256);

	Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(255, 255, 255));
	int max = histogram[0];

	for (int i = 1; i < 256; i++) {
		if (max < histogram[i])
			max = histogram[i];
	}

	for (int i = 0; i < 255; i++) {
		histogram[i] = floor(((double)histogram[i] / max) * histImage.rows);
	}
```

```
	for (int i = 0; i < 255; i++) {
		line(histImage, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - histogram[i]),
			Scalar(0, 0, 255));
	}

	imshow("Histogram", histImage);
}

int stretch(int x, int r1, int s1, int r2, int s2)
{
	float result;
	if (0 <= x && x <= r1) {
		result = s1 / r1 * x;
	}
	else if (r1 < x && x <= r2) {
		result = ((s2 - s1) / (r2 - r1)) * (x - r1) + s1;
	}
	else if (r2 < x && x <= 255) {
		result = ((255 - s2) / (255 - r2)) * (x - r2) + s2;
	}
	return (int)result;
}


void page11()
{
	Mat src = imread("./lenna.jpg", IMREAD_GRAYSCALE);
	imshow("Input Image", src);
	int histogram[256] = { 0 };
	for (int y = 0; y < src.rows; y++)
		for (int x = 0; x < src.cols; x++)
			histogram[(int)src.at<uchar>(y, x)]++;
	drawHist(histogram);
	waitKey(0);
}

int page15()
{
	Mat src = imread("./lenna.jpg", IMREAD_COLOR);
	if (src.empty()) { return -1; }
	vector<Mat> bgr_planes;
	split(src, bgr_planes);
	int histSize = 256;
	float range[] = { 0, 256 };
	const float* histRange = { range };
	bool uniform = true, accumulate = false;

	Mat b_hist, g_hist, r_hist;
	calcHist(&bgr_planes[0], 1, 0, Mat(), b_hist, 1, &histSize, &histRange, uniform, accumulate);
	calcHist(&bgr_planes[1], 1, 0, Mat(), g_hist, 1, &histSize, &histRange, uniform, accumulate);
	calcHist(&bgr_planes[2], 1, 0, Mat(), r_hist, 1, &histSize, &histRange, uniform, accumulate);

	int hist_w = 512, hist_h = 400;
	int bin_w = cvRound((double)hist_w / histSize);
	Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));
```

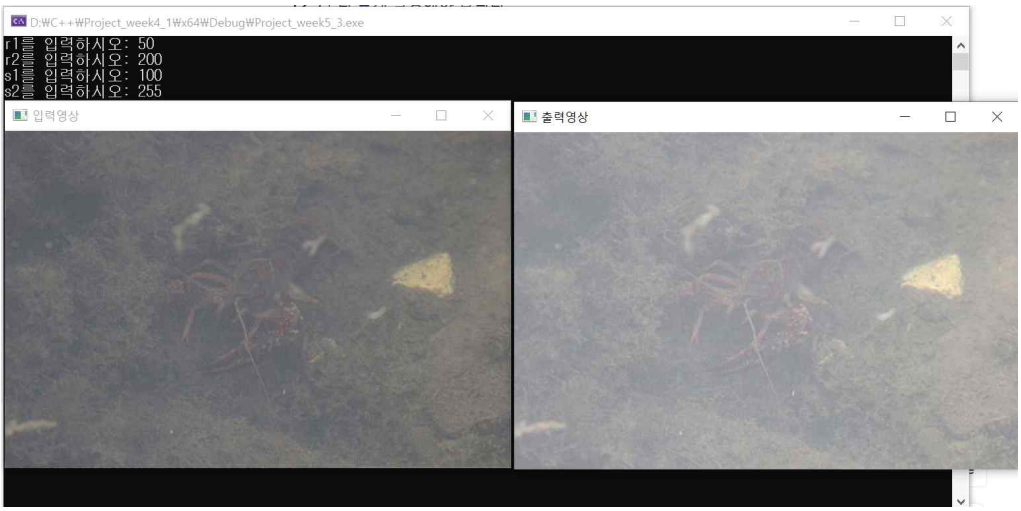
```
	normalize(b_hist, b_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
	normalize(g_hist, g_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
	normalize(r_hist, r_hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());

	for (int i = 0; i < 255; i++) {
		line(histImage, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - b_hist.at<float>(i)), Scalar(255, 0, 0));
		line(histImage, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - g_hist.at<float>(i)), Scalar(0, 255, 0));
		line(histImage, Point(bin_w * (i), hist_h), Point(bin_w * (i), hist_h - r_hist.at<float>(i)), Scalar(0, 0, 255));
	}
	imshow("입력 영상", src);
	imshow("컬러 히스토그램", histImage);
	waitKey();
}

void page19()
{
	Mat image = imread("./crayfish.jpg");
	Mat new_image = image.clone();
	int r1, s1, r2, s2;
	cout << "r1를 입력하시오: "; cin >> r1;
	cout << "r2를 입력하시오: "; cin >> r2;
	cout << "s1를 입력하시오: "; cin >> s1;
	cout << "s2를 입력하시오: "; cin >> s2;
	for (int y = 0; y < image.rows; y++) {
		for (int x = 0; x < image.cols; x++) {
			for (int c = 0; c < 3; c++) {
				int output = stretch(image.at<Vec3b>(y, x)[c], r1, s1, r2, s2);
				new_image.at<Vec3b>(y, x)[c] =
					saturate_cast<uchar>(output);
			}
		}
	}
	imshow("입력영상", image);
	imshow("출력영상", new_image);
	waitKey();
}

void page24()
{
	Mat image = imread("./equalize_test.jpg", 0);
	CV_Assert(!image.empty());
	Mat hist, dst1, dst2, hist_img, hist_img1, hist_img2;
	create_hist(image, hist, hist_img);

	Mat accum_hist = Mat(hist.size(), hist.type(), Scalar(0));
	accum_hist.at<float>(0) = hist.at<float>(0);

	for (int i = 1; i < hist.rows; i++) {
		accum_hist.at<float>(i) = accum_hist.at<float>(i - 1) + hist.at<float>(i);
	}

	accum_hist /= sum(hist)[0];
	accum_hist *= 255;
	dst1 = Mat(image.size(), CV_8U);
	for (int i = 0; i < image.rows; i++)
```

```
        {
                for (int j = 0; j < image.cols; j++)
                {
                        int idx = image.at<uchar>(i, j);
                        dst1.at<uchar>(i, j) = (uchar)accum_hist.at<float>(idx);

                }

        }

        equalizeHist(image, dst2);

        create_hist(dst1, hist, hist_img1);
        create_hist(dst2, hist, hist_img2);

        imshow("image", image), imshow("img_hist", hist_img);
        imshow("dst1-User", dst1), imshow("User_hist", hist_img1);
        imshow("dst2-OpenCV", dst2), imshow("OpenCV_hist", hist_img2);
        waitKey(0);
}

int main()
{
        page11();
        page15();
        page19();
        page24();

        return 0;
}
```

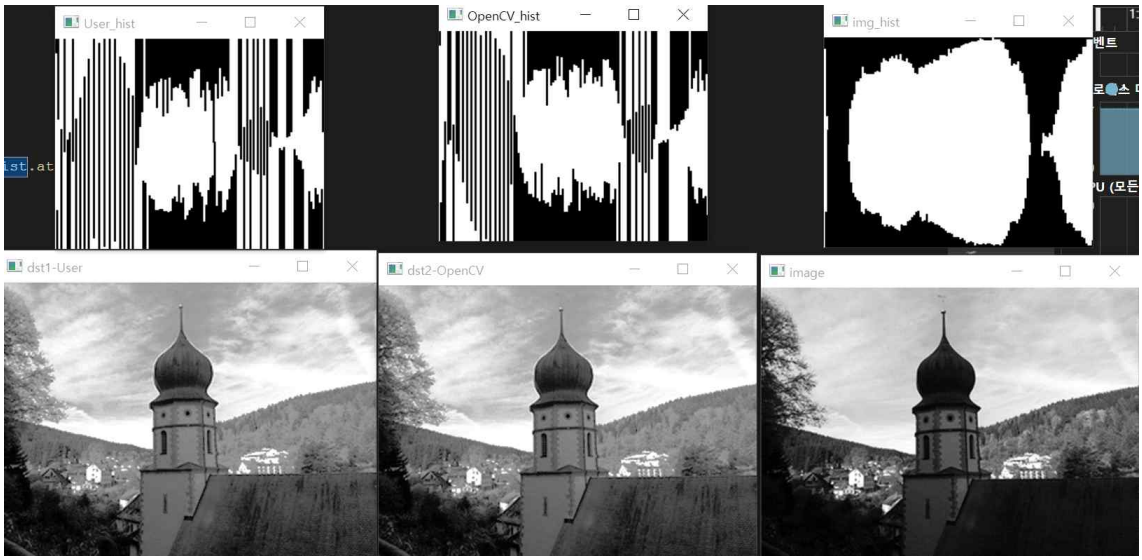
결과화면

<page11 결과화면>

<page15 결과화면>



<page19 결과화면>



<page24 결과화면>