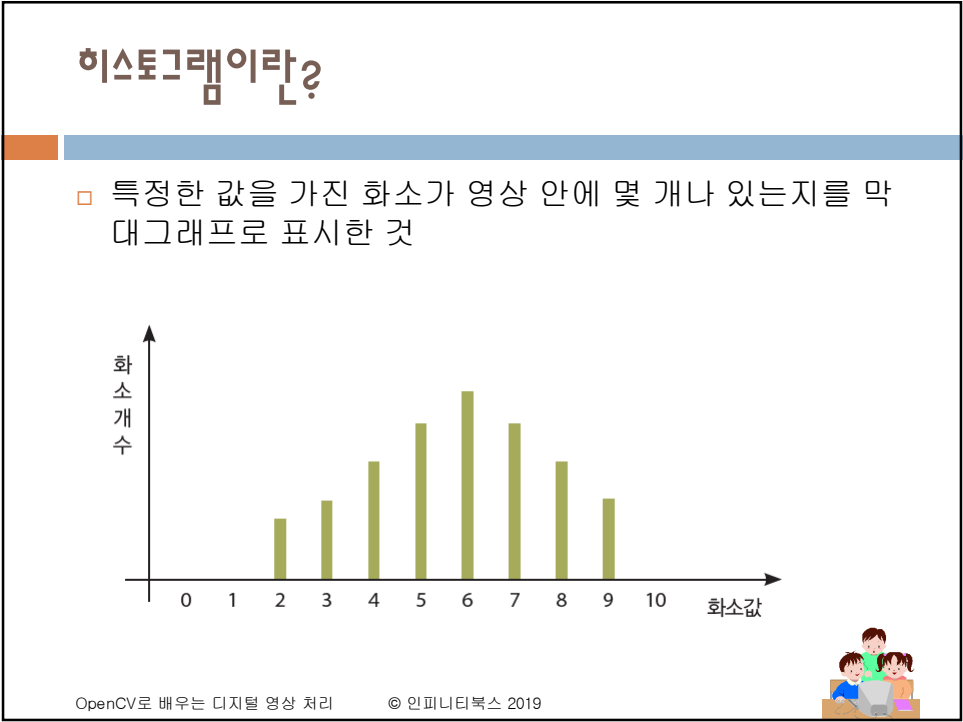


1



2

### 그레이스케일 히스토그램



OpenCV로 배우는 디지털 영상 처리      © 인피니티북스 2019



3

### 컬러 영상 히스토그램



OpenCV로 배우는 디지털 영상 처리      © 인피니티북스 2019



4

## 히스토그램으로 알 수 있는 것

- 화소값들의 분포를 한눈에 볼 수 있다.



어두운 영상



밝은 영상

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



5

## 히스토그램 계산하기

- 히스토그램 알고리즘

```
for each pixel of the image
    value = intensity(pixel)
    histogram[value]++
end
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



6

## OpenCV 프로그램

```
int main()
{
    Mat src = imread("d:/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)]++;

    for (int count : histogram)
        cout << count << ", ";
    waitKey(0);
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



7

## 실행 결과

```
선택 C:\WINDOWS\system32\cmd.exe
173,58,71,73,101,136,129,190,177,218,240,305,307,359,409,490,552,620,700,754,819,847,934,986,925,100
9,944,1000,991,965,939,926,928,814,798,759,689,655,605,576,533,467,466,440,493,415,422,399,362,392,3
73,397,374,388,380,420,406,407,403,387,435,449,462,467,443,480,420,425,473,443,393,414,450,412,447,4
99,459,505,534,524,556,584,588,579,613,689,662,663,803,882,950,1046,1068,1026,1020,941,937,914,869,7
71,728,682,645,662,639,570,595,658,641,627,673,651,692,629,694,699,696,699,812,851,805,841,888,954,9
95,1066,1080,1261,1132,1272,1228,1191,1200,1062,975,933,967,827,896,913,961,992,1040,1076,1095,1145,
1303,1197,1179,1215,1202,1201,1061,1132,1139,985,1083,1193,1152,1258,1280,1452,1342,1483,1252,1183,1
131,1078,1029,804,855,752,789,762,670,720,641,573,601,561,488,602,627,594,713,567,656,579,649,640,56
5,542,475,465,384,364,344,322,314,304,309,278,339,315,350,402,399,405,407,425,415,458,424,417,385,38
5,402,384,403,431,436,389,433,437,451,526,522,535,516,485,482,481,478,430,390,387,299,269,255,206,17
7,151,127,117,119,107,82,91,88,39,44,24,26,23,23,52,
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



8

## 히스토그램 그리기

```
// 히스토그램을 받아서 막대그래프로 그린다.
void drawHist(int histogram[])
{
    int hist_w = 512; // 히스토그램 영상의 폭
    int hist_h = 400; // 히스토그램 영상의 높이
    int bin_w = cvRound((double)hist_w / 256); // bin의 폭

    // 히스토그램이 그려지는 영상(컬러로 정의)
    Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(255, 255, 255));

    // 히스토그램에서 최대값을 찾는다.
    int max = histogram[0];
    for (int i = 1; i < 256; i++) {
        if (max < histogram[i])
            max = histogram[i];
    }
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



9

## 히스토그램 그리기

```
// 히스토그램 배열을 최대값으로 정규화한다(최대값이 최대 높이가 되도록).
for (int i = 0; i < 255; i++) {
    histogram[i] = floor(((double)histogram[i] /
max)*histImage.rows);
}

// 히스토그램의 값을 빨강색 막대로 그린다.
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);
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



10

## 히스토그램 그리기

```
int main()
{
    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);
    return 0;
}
```

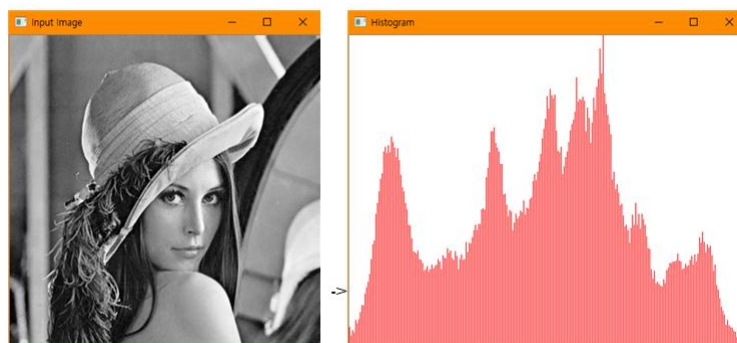
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



11

## 실행결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



12

## OpenCV 함수

□ `void calcHist(const Mat* images, int nimages, const int* channels, InputArray mask, OutputArray hist, int dims, const int* histSize, const float** ranges, bool uniform=true, bool accumulate=false )`

OpenCV로 배우는 디지털 영상 처리

© 인피니티박스 2019



13

## 히스토그램 그리기

```
int main(int argc, char** argv)
{
    Mat src = imread("d:/lenna.jpg", IMREAD_COLOR);
    if (src.empty()) { return -1; }

    vector<Mat> bgr_planes; // 영상들의 벡터
    split(src, bgr_planes); // 입력 영상을 색상별로 분리한다.
    int histSize = 256; // 히스토그램에서 사용되는 상자의 개수
    float range[] = { 0, 256 }; // 화소값의 범위
    const float* histRange = { range };
    bool uniform = true, accumulate = false;
    Mat b_hist, g_hist, r_hist;
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티박스 2019



14

## 히스토그램 그리기

```

    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);

    // 막대그래프가 그려지는 영상을 생성한다.
    int hist_w = 512, hist_h = 400;
    int bin_w = cvRound((double)hist_w / histSize); // 상자의 폭
    Mat histImage(hist_h, hist_w, CV_8UC3, Scalar(0, 0, 0));

    // 값들이 영상을 벗어나지 않도록 정규화한다.
    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());

```

OpenCV로 배우는 디지털 영상 처리

© 인피니티박스 2019



15

## 히스토그램 그리기

```

// 히스토그램의 값을 막대로 그린다.
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));
}

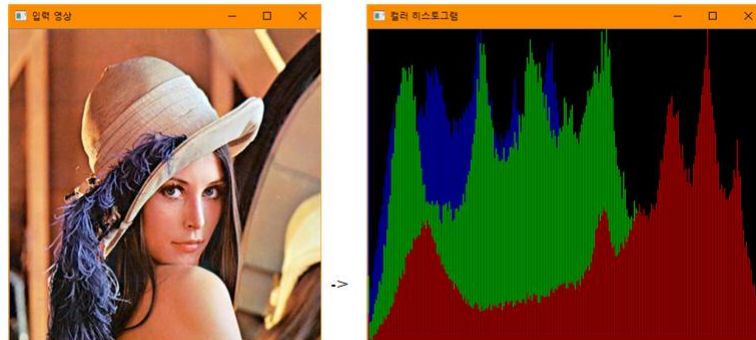
imshow("입력 영상", src);
imshow("컬러 히스토그램", histImage);
waitKey();
return 0;
}

```

16



## 실행 결과



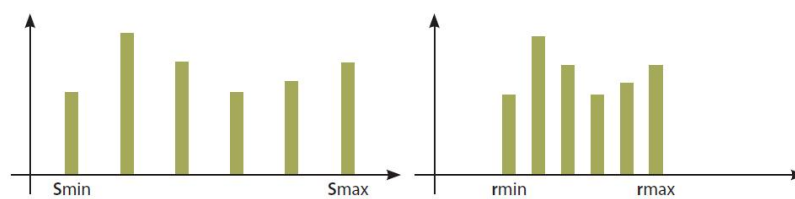
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



17

## 히스토그램 스트레칭

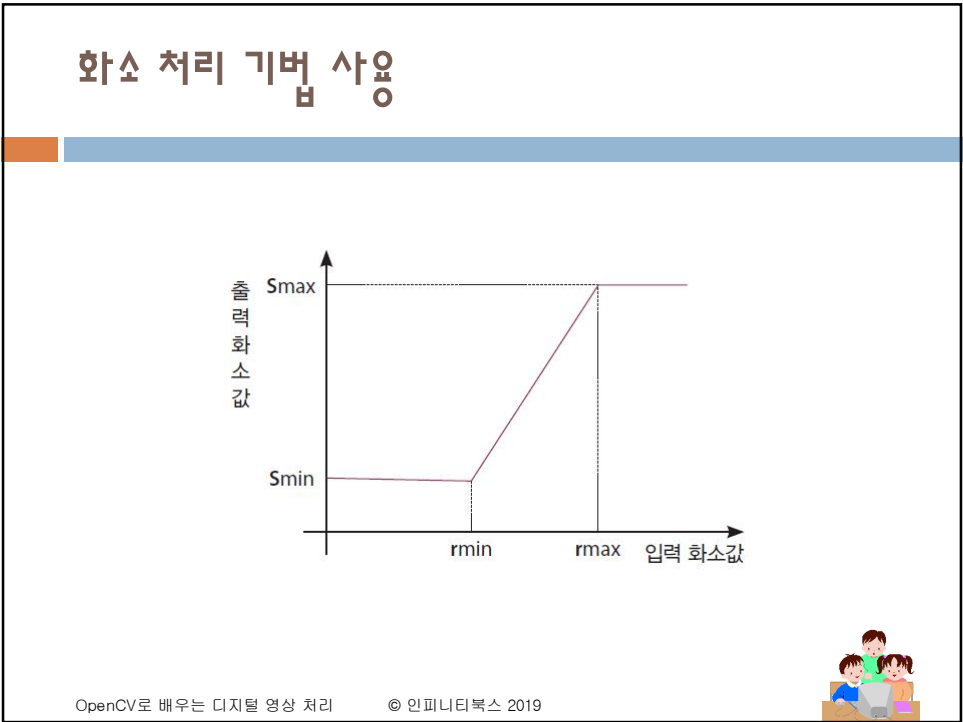


OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



18



19

### 화소 변환 수식

$$s = \frac{(s_{\max} - s_{\min})}{(r_{\max} - r_{\min})} \times (r - r_{\min}) + s_{\min}$$

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

20

## 히스토그램 스트레칭

```
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;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



21

```
int main()
{
    Mat image = imread("d:/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);
            }
        }
    }
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



22

### 실행 결과



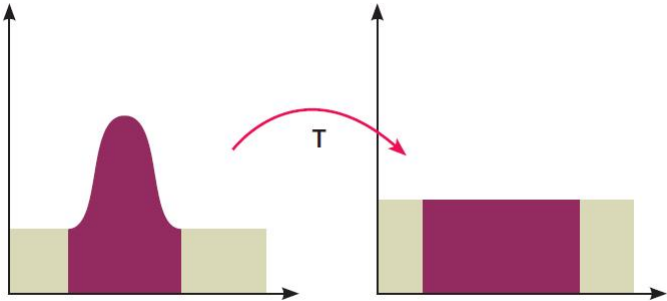
OpenCV로 배우는 디지털 영상 처리      © 인피니티북스 2019



23


### 히스토그램 평활화

□ 화소값의 분포를 나타내는 히스토그램이 균일하게 되도록 변환하는 처리이다.



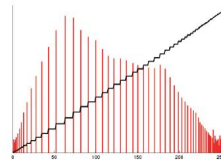
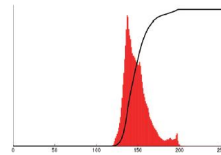
By Zefram – Own work, Public Domain,

OpenCV로 배우는 디지털 영상 처리      © 인피니티북스 2019



24

## 히스토그램 평활화



By original Phillip Capper, modified by User: Konstable-modified Hawkes Bay NZ.jpg, CC BY 2.0(출처: 위키백과)

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



25

## 어떤 화소 변환 함수를 사용할 것인가?

- **CDF**를 변환 함수로 사용한다.

$$CDF(i \leq t) = \sum_{k=0}^t p_k$$

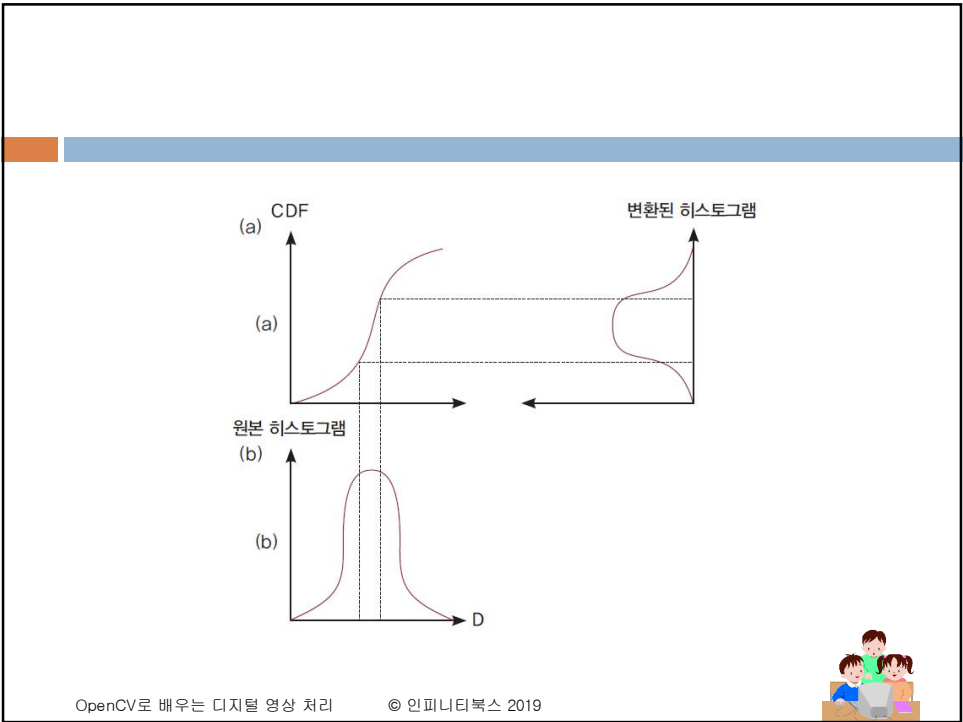
$$p_k = \frac{\text{밝기값 } k \text{를 가진 화소수}}{\text{전체 화소수}}$$

OpenCV로 배우는 디지털 영상 처리

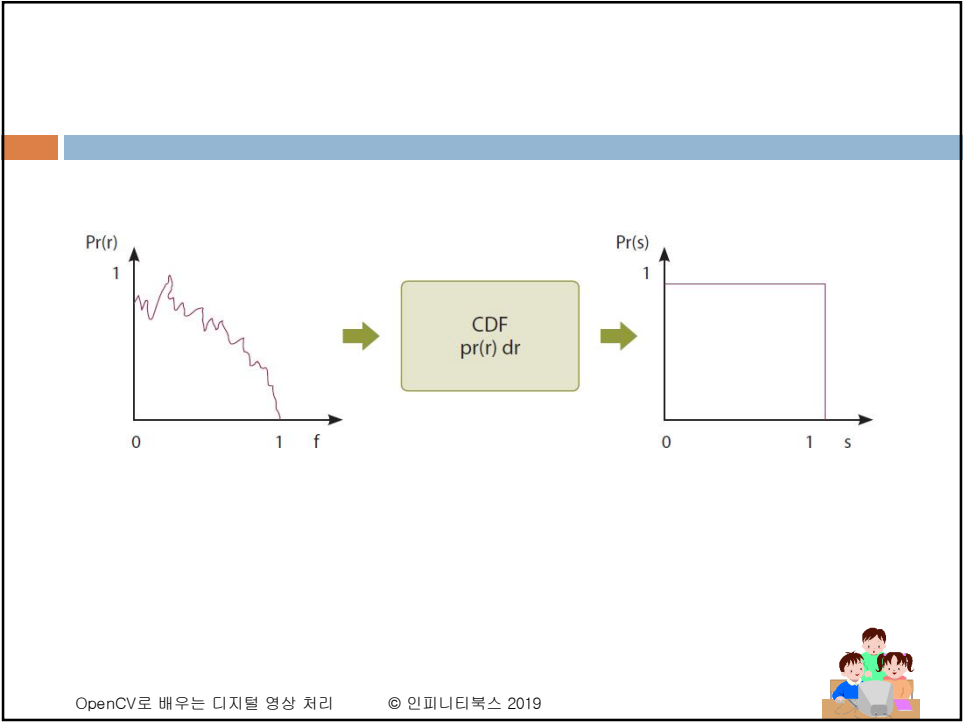
© 인피니티북스 2019



26



27



28

## 히스토그램 평활화의 절차

1. 평활화되어야 할 영상의 누적 히스토그램을 구한다

$$H'(i) = \sum_{k=0}^i H(k)$$

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



29

## 히스토그램 평활화의 절차

2. 누적 히스토그램의 값을 0.0과 1.0 사이로 정규화 한다.

$$\text{정규화된 누적 히스토그램} = \frac{H'(i)}{n_i}$$

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



30

## 히스토그램 평활화의 절차

3. 정규화 된 누적 히스토그램을 변환 함수로 이용하여 화소 값을 변환한다.

$$s = (L - 1) \times \frac{H'(i)}{n_t}$$

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



31

## 히스토그램 평활화

```
int main()
{
    Mat src = imread("d:/crayfish.jpg", IMREAD_GRAYSCALE);
    if (src.empty()) { return -1; }

    Mat dst;
    equalizeHist(src, dst);

    imshow("Image", src);
    imshow("Equalized", dst);
    waitKey(0);
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



32



### 실행결과

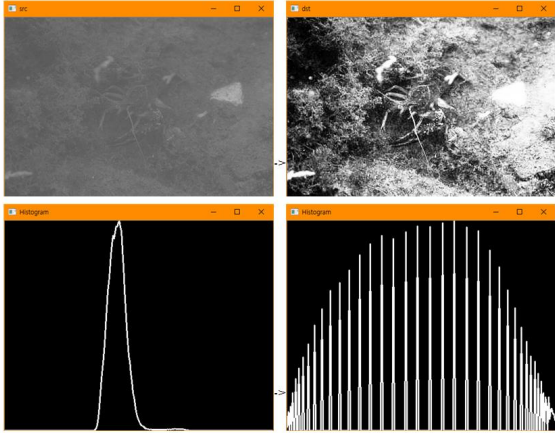


사진 출처: By Biem [Public domain], from Wikimedia Commons

OpenCV로 배우는 디지털 영상 처리      © 인피니티북스 2019



33

### 히스토그램을 이용한 전경과 배경 분리



OpenCV로 배우는 디지털 영상 처리      © 인피니티북스 2019



34

## 히스토그램을 이용한 전경과 배경 분리

```
using namespace std;
using namespace cv;

int main()
{
    Mat src, dst;

    src = imread("d:/plane.jpg", IMREAD_GRAYSCALE);
    imshow("Image", src);
    if (!src.data) { return -1; }

    Mat threshold_image;
    threshold(src, threshold_image, 100, 255, THRESH_BINARY);
    imshow("Thresholded", threshold_image);
    waitKey(0);
    return 0;
}
```

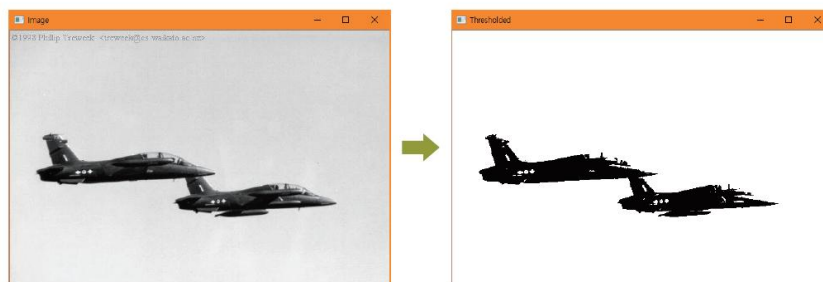
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



35

## 실행 결과



OpenCV로 배우는 디지털 영상 처리

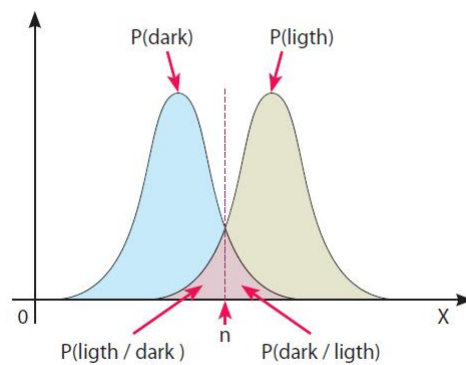
© 인피니티북스 2019



36

## 향상된 이진화 방법

- 컴퓨터가 영상을 분석하여 자동으로 임계값을 결정하게 할 수 있을까?



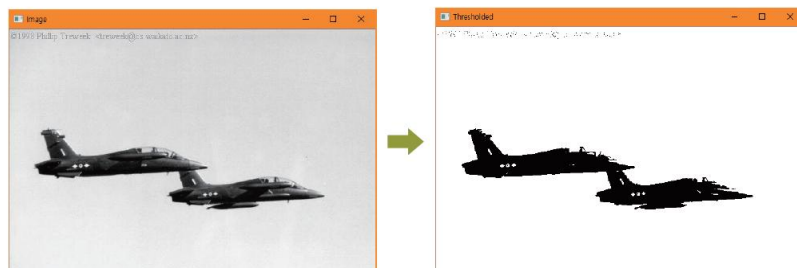
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



37

- `threshold(src, threshold_image, 0, 255, CV_THRESH_BINARY | CV_THRESH_OTSU);`



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



38

Q & A

