

1

전통적인 영상 처리

- 화소 처리(point processing)
- 공간 필터링(filtering)

(a)

입력 영상

출력 영상

(b)

입력 영상

출력 영상

[그림 4.1] 화소 처리와 공간 필터링의 비교 (a) 화소 처리 (b) 공간 필터링

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019

2

1

화소 처리

- 밝기 및 콘트라스트 조정
- 색상 교정
- 색상 변환

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



3

화소를 하나씩 처리하는 방법 #1

- **Mat** 클래스가 가지고 있는 **at()** 함수를 사용하면 영상에서 임의의 화소값을 가져오거나 수정할 수 있다.
- **at(y, x)**의 인수로 화소의 행 번호와 열 번호를 전달하면 된다.

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



4

at() 함수

```
#include "opencv2/opencv.hpp"
using namespace cv;
using namespace std;

int main()
{
    Mat img = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    imshow("Original Image", img);

    for (int r = 0; r < img.rows; r++)
        for (int c = 0; c < img.cols; ++c)
            img.at<uchar>(r, c) = img.at<uchar>(r, c) + 30;

    imshow("New Image", img);
    waitKey(0);
    return 0;
}
```

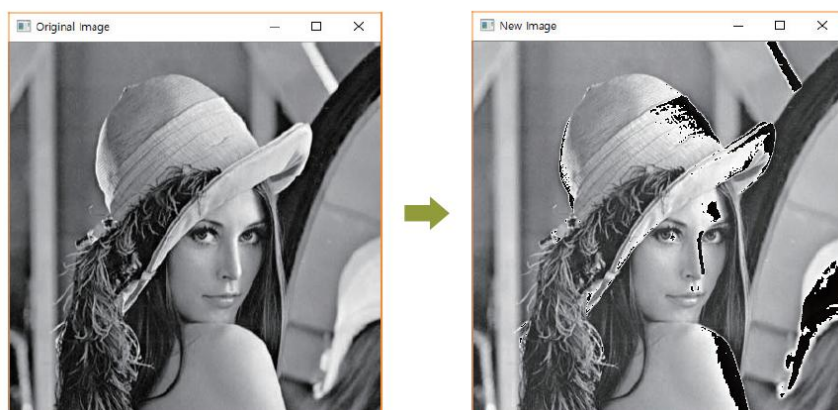
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



5

실행 결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



6

오류 원인

- 이것은 화소의 값에 30이 더해지면 255를 넘게 되어서 오버플로우가 일어난 것이다.
- `img.at<uchar>(r, c) = saturate_cast<uchar>(img.at<uchar>(r, c) + 30);`

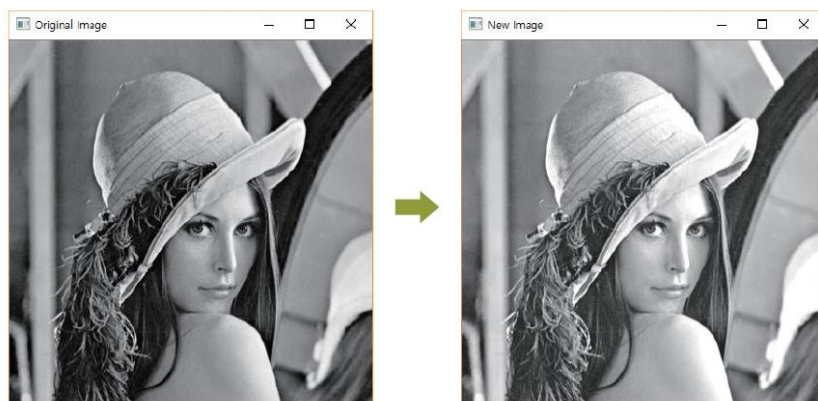
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



7

실행결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



8

화소를 하나씩 처리하는 방법 #2

- C 스타일 연산자 []를 사용
- 가장 성능이 우수함

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



9

```
#include "opencv2/opencv.hpp"
using namespace cv;
using namespace std;

int main()
{
    Mat img = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    imshow("Original Image", img);

    for (int r = 0; r < img.rows; r++) {
        uchar *p = img.ptr<uchar>(r);
        for (int c = 0; c < img.cols; ++c) {
            p[c] = saturate_cast<uchar>(p[c] + 30);
        }
    }
    imshow("New Image", img);
    waitKey(0);

    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



10

화소를 하나씩 처리하는 방법 #3

- OpenCV 함수 사용하기
- 영상의 밝기를 증가시키는 것은 OpenCV에서 이미 함수로 지원하고 있다. 바로 **convertTo()** 함수이다.

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



11

```
#include "opencv2/opencv.hpp"
using namespace cv;
using namespace std;

int main()
{
    Mat img = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    imshow("Original Image", img);

    Mat oimage;
    img.convertTo(oimage, -1, 1, 30);

    imshow("New Image", oimage);
    waitKey(0);
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



12

영상을 처리하는 함수를 작성해보자.

- 영상의 밝기를 증가시키는 함수 `brighten()`를 작성해보자.

```
void brighten(Mat& img, int value) {
    ....
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



13

```
#include "opencv2/opencv.hpp"
using namespace cv;
using namespace std;

void brighten(Mat& img, int value)
{
    for (int r = 0; r < img.rows; r++)
        for (int c = 0; c < img.cols; ++c)
            img.at<uchar>(r, c) = saturate_cast<uchar>(img.at<uchar>(r, c) +
value);
}

int main()
{
    Mat img = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    imshow("Original Image", img);

    brighten(img, 30);
    imshow("New Image", img);
    waitKey(0);

    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리

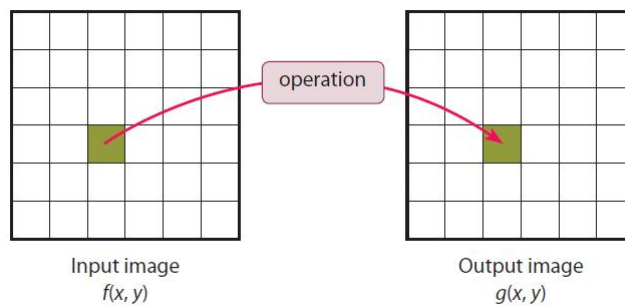
© 인피니티북스 2019



14

밝기 및 콘트라스트 조정

□ $g(x, y) = T[f(x, y)]$



OpenCV로 배우는 디지털 영상 처리

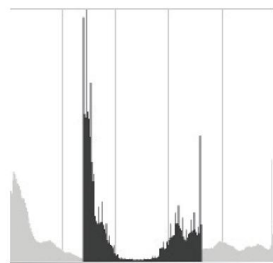
© 인피니티북스 2019



15

밝기 및 콘트라스트 조정

- $g(x, y) = \alpha \cdot f(x, y) + \beta$
- α : 밝기값이 어떻게 퍼지는지를 결정한다
- β : 화소에 더해지는 값



[그림 4.3] 밝은 화색은 원본 영상의 히스토그램, 어두운 화색은 콘트라스트가 줄어든 영상이다.

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



16

화소 접근 방식으로 코드 작성

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace std;
using namespace cv;

int main()
{
    double alpha = 1.0;
    int beta = 0;
    Mat image = imread("d:/contrast.jpg");
    Mat oimage = Mat::zeros(image.size(), image.type());
    cout << "알파값을 입력하시오: [1.0-3.0]: "; cin >> alpha;
    cout << "베타값을 입력하시오: [0-100]: "; cin >> beta;
    for (int y = 0; y < image.rows; y++) {
        for (int x = 0; x < image.cols; x++) {
            for (int c = 0; c < 3; c++) {
                oimage.at<Vec3b>(y, x)[c] =

                saturate_cast<uchar>(alpha*(image.at<Vec3b>(y, x)[c]) + beta);
            }
        }
    }
}
```

17

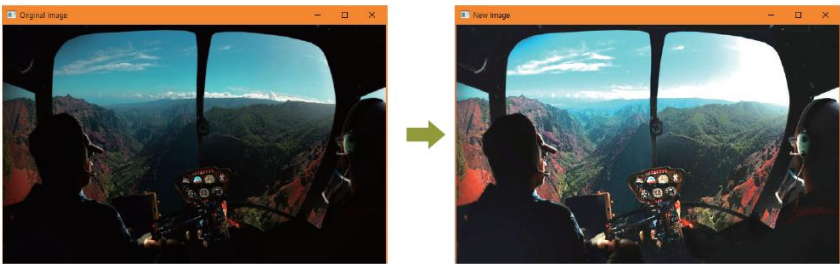
화소 접근 방식으로 코드 작성

```
imshow("Original Image", image);
imshow("New Image", oimage);
waitKey();
return 0;
}
```



18

실행 결과



OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019



CONVERTTO() 함수 사용

```
void convertTo(OutputArray m, int rtype, double alpha = 1, double beta = 0)
```

매개 변수	설명
m	출력 행렬. 작업 전에 적절한 크기나 유형이 없으면 다시 할당된다.
rtype	원하는 출력 행렬 유형을 지정한다. rtype이 음수면 출력 행렬은 입력과 동일한 유형을 갖는다.
alpha	화소값에 곱해지는 수
beta	화소값에 더해지는 수

OpenCV로 배우는 디지털 영상 처리 © 인피니티박스 2019



CONVERTTO() 함수 사용

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace std;
using namespace cv;

int main()
{
    double alpha = 1.0;
    int beta = 0;
    Mat image = imread("d:/contrast.jpg");
    Mat oimage;
    cout << "알파값을 입력하시오: [1.0-3.0]: "; cin >> alpha;
    cout << "베타값을 입력하시오: [0-100]: "; cin >> beta;

    image.convertTo(oimage, -1, alpha, beta);
    imshow("Original Image", image);
    imshow("New Image", oimage);
    waitKey();
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

21

행렬의 중복정의된 덧셈과 곱셈을 이용해보자.

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace std;
using namespace cv;

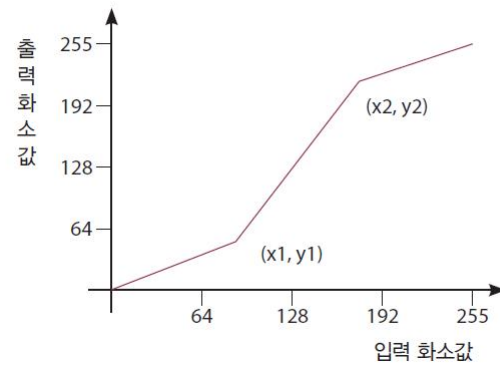
int main()
{
    double alpha = 1.0;
    int beta = 0;
    Mat image = imread("d:/contrast.jpg");
    Mat oimage;
    cout << "알파값을 입력하시오: [1.0-3.0]: "; cin >> alpha;
    cout << "베타값을 입력하시오: [0-100]: "; cin >> beta;

    oimage = image * alpha + beta;
    imshow("Original Image", image);
    imshow("New Image", oimage);
    waitKey();
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

22

선형 콘트라스트 확대



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



23

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace std;
using namespace cv;

int contrastEnh(int input, int x1, int y1, int x2, int y2)
{
    double output;
    if (0 <= input && input <= x1) {
        output = y1 / x1 * input;
    }
    else if (x1 < input && input <= x2) {
        output = ((y2 - y1) / (x2 - x1)) * (input - x1) + y1;
    }
    else if (x2 < input && input <= 255) {
        output = ((255 - y2) / (255 - x2)) * (input - x2) + y2;
    }
    return (int)output;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



24

```

int main()
{
    Mat image = imread("d:/lenna.jpg");
    Mat oimage = image.clone();
    int x1, y1, x2, y2;
    cout << "x1 값을 입력하시오: "; cin >> x1;
    cout << "y1 값을 입력하시오: "; cin >> y1;
    cout << "x2 값을 입력하시오: "; cin >> x2;
    cout << "y2 값을 입력하시오: "; cin >> y2;

    for (int r = 0; r < image.rows; r++) {
        for (int c = 0; c < image.cols; c++) {
            for (int ch = 0; ch < 3; ch++) {
                int output = contrastEnh(image.at<Vec3b>(r, c)[ch], x1,
y1, x2, y2);

                oimage.at<Vec3b>(r, c)[ch] =
saturate_cast<uchar>(output);
            }
        }
    }
    imshow("원영상", image);
    imshow("결과영상", oimage);
    waitKey();
    return 0;
}

```

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

25

실행 결과

```

C:\WINDOWS\system32\cmd.exe
r1 값을 입력하시오: 70
s1 값을 입력하시오: 0
r2 값을 입력하시오: 150
s2 값을 입력하시오: 255

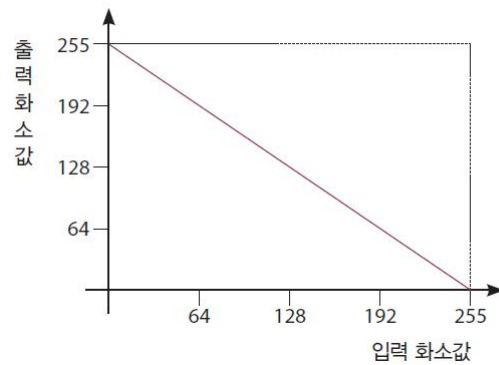
```



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019

26

바지



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



27

```

...
int main()
{
    Mat src;
    src = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    imshow("원영상", src);

    Mat dst;
    dst = 255 - src;
    imshow("변경된 영상", dst);

    waitKey(0);
    return 0;
}

```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



28

실행결과



→

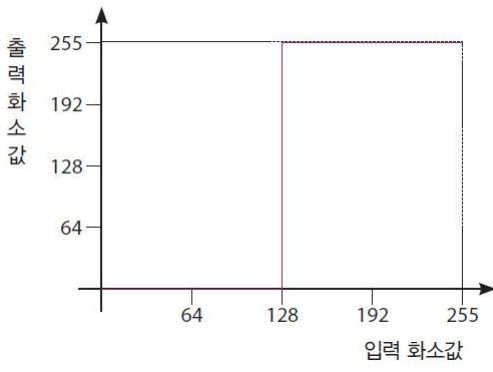


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019




29

이진화



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



30

```
double threshold(InputArray src, OutputArray dst, double thresh,
                double maxval, int type)
```

매개 변수	설명
src	입력 영상. 1채널이어야 한다(8비트 또는 32-bit floating point).
dst	출력 영상
thresh	임계값
maxval	가능한 최대 출력값
type	이진화 종류. 우리는 THRESH_BINARY만 사용한다.

OpenCV로 배우는 디지털 영상 처리

© 인피니티박스 2019

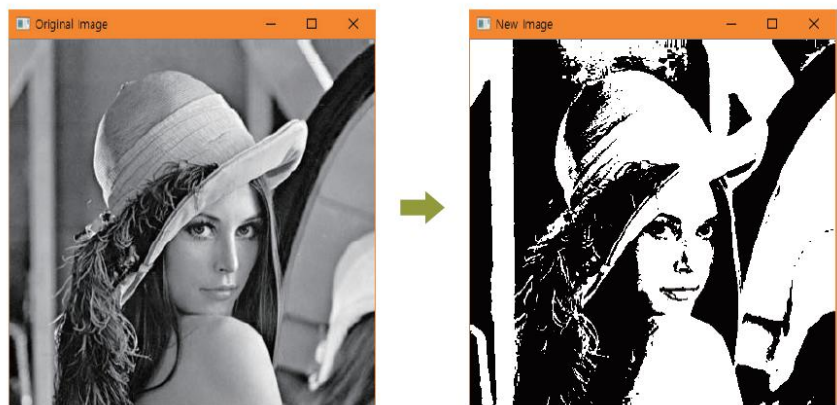
31

```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace std;
using namespace cv;

int main()
{
    Mat image = imread("d:/lenna.jpg", IMREAD_GRAYSCALE);
    Mat dst;
    int threshold_value = 127;
    threshold(image, dst, threshold_value, 255, THRESH_BINARY);
    imshow("Original Image", image);
    imshow("New Image", dst);
    waitKey(0);
    return 0;
}
```

32

실행 결과

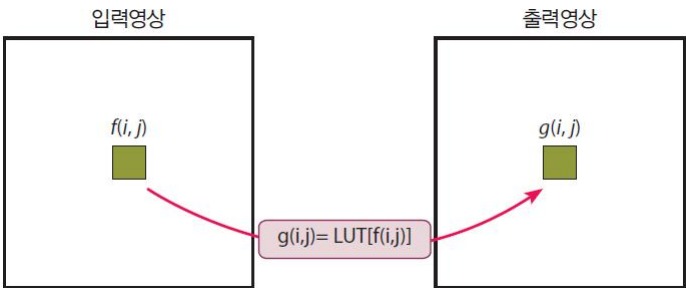


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



33

LUT 사용 방법

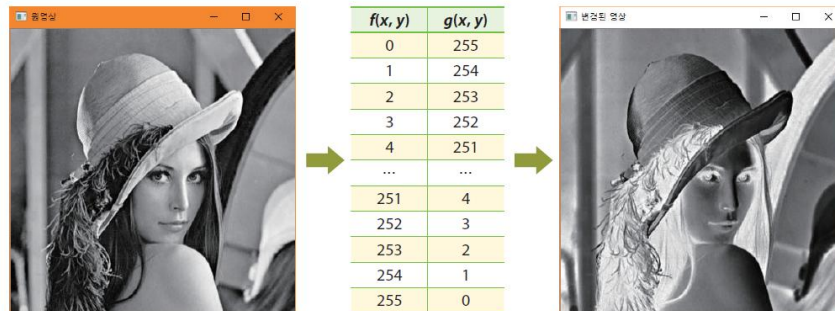


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



34

LUT 사용 방법



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



35

```

#include "opencv2/opencv.hpp"
using namespace cv;
using namespace std;

int main()
{
    Mat img1 = imread("d:/Lenna.jpg", IMREAD_GRAYSCALE);
    imshow("Original Image", img1);

    Mat table(1, 256, CV_8U);      // ①

    uchar* p = table.ptr();
    for (int i = 0; i < 256; ++i)
        p[i] = (i / 100) * 100;

    Mat img2;
    LUT(img1, table, img2);

    imshow("New Image", img2);
    waitKey(0);

    return 0;
}

```

OpenCV로 배우는 디지털 영상 처리

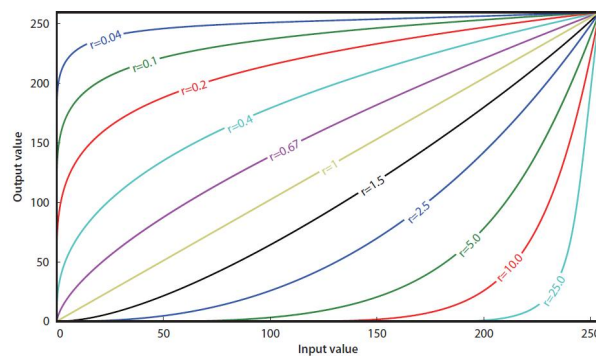
© 인피니티북스 2019



36

감마보정

$$g(x, y) = \left(\frac{f(x, y)}{255} \right)^\gamma \times 255$$



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



37

```
int main()
{
    Mat src1, src2, dst;
    double gamma = 0.5;

    src1 = imread("d:/gamma1.jpg");
    if (src1.empty()) { cout << "영상을 읽을 수 없습니다." << endl; return -1; }

    Mat table(1, 256, CV_8U);
    uchar * p = table.ptr();
    for (int i = 0; i < 256; ++i)
        p[i] = saturate_cast<uchar> (pow(i / 255.0, gamma) * 255.0);

    LUT(src1, table, dst);
    imshow("src1", src1);
    imshow("dst", dst);
    waitKey(0);
    return 0;
}
```

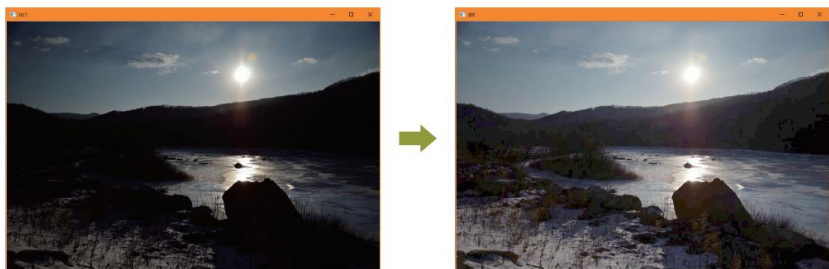
OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



38

실행결과



[그림 4.8] 감마 보정의 효과
(출처: <https://www.codepool.biz/image-processing-opencv-gamma-correction.html>)



영상합성

$$g(x, y) = f_1(x, y) + f_2(x, y)$$



```

#include "opencv2/opencv.hpp"
#include <iostream>
using namespace std;
using namespace cv;

int main()
{
    Mat src1 = imread("d:/test1.jpg");
    Mat src2 = imread("d:/test2.jpg");
    Mat dst;
    dst = src1 + src2;
    imshow("Original Image1", src1);
    imshow("Original Image2", src2);
    imshow("New Image", dst);
    waitKey(0);
    return 0;
}

```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



41

실행 결과



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



42

선형영상합성


LINEAR IMAGE COMPOSITING

$$g(x, y) = (1 - \alpha) * f_1(x, y) + \alpha * f_2(x, y)$$

```
void addWeighted(InputArray src1, double alpha, InputArray src2,
                double beta, double gamma, OutputArray dst, int dtype=-1)
```

매개 변수	설명
src1	첫 번째 입력 영상
alpha	첫 번째 영상의 가중치
src2	두 번째 입력 영상
beta	두 번째 영상의 가중치
gamma	화소의 합계에 더해지는 값
dst	출력 영상


OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



```
#include "opencv2/opencv.hpp"
#include <iostream>
using namespace cv;
using namespace std;

int main()
{
    double alpha = 0.5; double beta; double input;
    Mat src1, src2, dst;
    cout << "알파값을 입력하시오[0.0-1.0]: ";
    cin >> input;
    src1 = imread("d:/test1.jpg");
    src2 = imread("d:/test2.jpg");
    if (src1.empty()) { cout << "영상1을 로드할 수 없습니다." << endl; return -1; }
    if (src2.empty()) { cout << "영상2를 로드할 수 없습니다." << endl; return -1; }
    beta = (1.0 - alpha);
    addWeighted(src1, alpha, src2, beta, 0.0, dst);
    imshow("Original Image1", src1);
    imshow("Original Image2", src2);
    imshow("선형 합성", dst);
    waitKey(0);
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



실행결과



OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



논리적인 영상 합성

- 2개의 영상을 가지고 비트별로 AND, OR, XOR와 같은 논리적인 연산을 적용할 수 있다.

OpenCV로 배우는 디지털 영상 처리 © 인피니티북스 2019



논리적인 영상 합성



OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



47

```
int main()
{
    Mat img1, mask;

    img1 = imread("d:/scene.jpg", IMREAD_COLOR);
    if (img1.empty()) { cout << "영상1을 로드할 수 없습니다." << endl; return -1; }
    mask = imread("d:/mask.png", IMREAD_COLOR);
    if (mask.empty()) { cout << "영상2을 로드할 수 없습니다." << endl; return -1; }

    Mat dst = img1.clone();
    imshow("img1", img1);
    imshow("mask", mask);

    bitwise_and(img1, mask, dst);
    imshow("dst", dst);
    waitKey(0);
    return 0;
}
```

OpenCV로 배우는 디지털 영상 처리

© 인피니티북스 2019



48

Q & A

