멀티미디어정보처리

컴퓨터공학과

2013043401

**정지성**

제출일:17.04.02

과제에 들어가기 앞서서

코드의 가독성을 높이기 위해

파일위치를 String 변수로 선언하였고,(변수명은 대문자로 선언하였습니다.)

반복적인 코드와 기능별로 함수를 만들었습니다.

void m\_show(Mat img, String saveName){

winNum++; ->전역변수 입니다.

**namedWindow**("Window"+winNum,CV\_WINDOW\_AUTOSIZE);

**imshow**("Window"+winNum,img);

String locate = saveName+".jpg";

**imwrite**(locate, img);

}

void m\_invert(Mat);

void m\_stretching(Mat,float);

void m\_equalization(Mat);

void m\_thresholding(Mat,int);

void m\_Add(Mat,Mat,float,float);

void m\_Sub(Mat,Mat,float,float);

**예시)**

String LYALC\_D = "D:\\실습자료\\hw1\\lya-lc-dark.bmp";

String LYALC\_X = "D:\\실습자료\\hw1\\lya-lc-안되는예.bmp";

Mat img = imread(LYALC,CV\_32FC3);

m\_invert(img); ->영상반전 함수

그러므로 아래 내용부터는 함수만 보여드리겠습니다.

**Inverse**

**Code**

void m\_invert(Mat img){

Mat Output(img.rows,img.cols,CV\_8UC3);

Mat invImg[3];

split(img,invImg); ->3차원 배열인 img를 3개로 분리한다.

for(int i=0;i<3;i++)

invImg[i]=**255-invImg[i];** ->영상의 RGB 값을 각각 255에서 뺀다.

merge(invImg,3,Output); ->합친다.

m\_show(Output,SAVE+”\_Invert”); ->출력

}

**Inverse**

**Image**

|  |  |
| --- | --- |
| **원본** | **변환** |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (1).png | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31.png**1** |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (4).png | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (5).png**2** |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (8).png | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (10).png**3** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (11).pngC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (13).png** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (16).pngC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (17).png4** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Invert(1).jpg5** |

**Inverse**

**discussion**

invert는 정지영상의 보색 대비색을 보여준다.

이를 구현하기 위해서는 컬러영상에 대한 이해가 필요하다.

컬러영상의 경우 흑백영상과는 다르게 3개의 채널을 사용한다.

컬러영상은 Red Green Blue 의 3개의 채널로 구성되어 있고 각 채널 당 8bit - 28개의 색을 표현 할 수 있어 0부터 255의 값을 갖는다. 이 값은 클수록 R,G,B에 가까워지고 3개의 채널 값의 합을 통해 색을 표현할 수 있다. 그러므로 224개의 색을 표현할 수 있는 것이다.

위의 5번째 영상이 RGB를 좌표로 표현한 영상이다.

이러한 채널을 특성을 활용하여 영상의 보색을 출력할 수 있다.

여기서 보색이란 반대되는 색으로, 두 색을 합칠 때 **무채색(흰색)**이 되는 색의 조합을 말한다.

그러므로 특정 값 A의 보색 값 B는 A+B=255 -> **B=255-A** 가 된다.

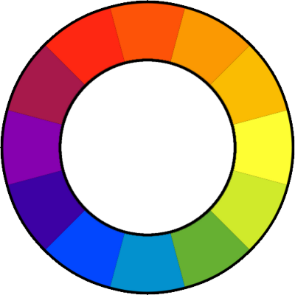
정지 영상의 모든 픽셀에 각 3개의 채널에 위의 연산을 수행하면 보색 영상이 생성된다, 위의 코드에서

for(int i=0;i<3;i++)

invImg[i]=**255-invImg[i];**

이 부분이 invert 하는 부분이다.

아래 사진은 보색표로 반대편에 있는 색이 보색이다.



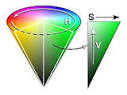
위의 4번 영상을 보면 보색 전환하여 대칭을 이루는 것을 볼 수 있다.

지금까지는 RGB컬러모델을 사용한 보색 전환 방법이었고 논의를 확장해보도록 하겠다.

여러 컬러 모델 중HSV컬러모델에 대해서 살펴보겠다.

HSV는 Hue, Saturation, Value의 약자로 색상, 채도, 명도로 구성된다.

이 모델은 아래 사진과 같이 원뿔모양의 좌표계로 표현된다

.

RGB의 0-255 범위 대신에

H는 0~360의 범위(0=Red, 120=Green, 240=Blue)

S는 0~100 의 범위

V 는 0-100의 범위(0=흰색,100=검정색)

를 갖는다

OpenCV에서는 HSV 모델을 표현할 때

H/S/V -> 0~180 / 0~255 / 0~255 의 범위로 표현한다.

HSV이미지로 변환하기 위해서는

cvCvtColor(RGBimg,HSVimg,CV\_BGR2HSV) 함수를 사용해

RGN 이미지를 HSV 이미지로 받아온다.

**cvCvtPixToPlane**(HSVimg,h,s,v,0) 함수로 HSV 각 채널을 받아올 수 있다.

그러므로 h= 180-h, s = 255-s, v = 255-v, 이런 형식으로 코드를 구성하면

HSV모델에서 보색 영상을 구할 수 있다.

**Contrast Stretching**

**Code**

void m\_stretching(Mat img,float rate){

Mat **st** = cvCreateMat(img.rows,img.cols,CV\_8UC1);

int x,y;

int min =257, max = -1;

int **MN** = (img.rows \*img.cols);

float **slope** = 0;

int **hist**[256] = {0},sum\_1 = 0, sum\_2 = 0;

for(y=0;y<img.rows;y++){

for(x=0;x<img.cols;x++)

**hist**[(int)img.at<uchar>(y,x)]++;

} ->히스토그램 생성

for(int i=0;i<256;i++){

sum\_1 += hist[i];

if(sum\_1>**MN**\***rate**) ->전체 비율의 ‘rate’만큼을 갖고

{

**min** = i; ->최소값 찾기

break;

}

}

for(int i=255;i>=0;i--){

**sum\_2** +=**hist**[i];

if(sum\_2>**MN**\***rate**){

**max** = i; ->최대값 찾기

break;

}

}

**slope** = 255/(max-min);

for (y=0; y<img.rows;y++)

{

for(x=0;x<img.cols;x++)

{

int k;

k=(int)(img.at<uchar>(y,x));

if(k<min)

st.at<uchar>(y,x) = 0;

else if (k>max)

st.at<uchar>(y,x) = 255;

else

st.at<uchar>(y,x) = **(k-min)\*slope;**

}

}

m\_show(st,SAVE+”\_Stretching”);

}

**Contrast Stretching**

**Image**

|  |  |
| --- | --- |
| **원본** | **변환** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (29).pngC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (24).pngC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (26).pngC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (25).pngC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (23).pngC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (27).pngC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Stretching(1).jpg1** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (31).png** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (32).png2** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (33).png** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (34).png3** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (35).png** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (36).png4** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (38).png** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2017-03-31 (37).png5** |

**Contrast Stretching**

**Discussion**

Contrast Stretching은 히스토그램의 분포를 늘려서 영상 밝기 분포를 개선시키는 방법이다.

이를 위해서 해당 영상 분포의 명암 최소값과 최대값을 찾아 연산을 수행한다.

전체적인 stretching을 위해 255에서 최대값 M과 최소값 m 을 뺀 값을 나누고 픽셀에 최소값을 뺀 값을 곱한다. 수식으로 보면 다음과 같다.

*Stretching\_Value* =

위의 값을 각 픽셀에 대입하면 영상이 Contrast Stretching 이 된다.

이 변환법은 히스토그램의 분포가 한쪽으로 집중해 있을 때 유용하게 사용된다. 예를 들어 전체적으로 어두운 영상이나 밝은 영상을 개선할 때 사용한다.

위의 영상을 보도록 하자. 1번의 1~3번 영상들은 변환 후 육안상 동일한 영상으로 보인다. 이는 히스토그램이 동일한 패턴으로 위치만 다르게 분포되어 있기 때문에 stretching 연산을 하면 모두 같은 분포의 형태를 보인다. 2번 영상에서 뒤의 희색 배경이 변환 후 회색으로 바뀐 것을 볼 수 있다. 연산 식에 의해 흰색(255)보다 작은 값이 되었기 때문에 명도가 낮아지게 되었다. 3번과 4번 또한 같은 원리로 명도가 올라가게 된 것이다.

Contrast stretching은 Max값과 min으로 곱할 값을 결정하기 때문에 치명적인 단점이 있다.

만약 영상이 전체적으로 어둡지만 어느 한 픽셀이 255의 값을 갖는 경우 영상 개선의 효과를 보기 어렵다. 이는 위의 1번의 4번째 영상과 5번의 영상(1-3영상에 흰색 원을 추가한 것)을 보면 알 수 있다. 원본영상과 변환된 영상의 차이를 육안으로는 파악할 수 없다.

이와 같은 단점은 최대,최소값을 수정하여 개선할 수 있다. 위의 코드에서처럼 M,m 값을 영상 전체에서 설정하는 것이 아니라 전체영상의 비율에 따라서 설정을 한다.

예시로 0.05의 비율을 통해 M,m을 찾는다면 전체 영상에서 100% -5% -5% = 90% 의 분포에서 값을 찾아 stretching을 수행한다. 분포의 좌 우 5% 안의 값이 기존 영상의 밀집된 부분과 차이가 많이 나는 경우 이를 무시하고 수행되어 영상의 개선 효과를 늘릴 수 있다.

Contrast stretching을 컬러 영상으로 논의를 확장해 보자

컬러 영상의 경우 3개의 채널을 사용하므로 각 채널 별 Stretching 이 필요하다.

예를 들어 R,G,B모델인 경우 전체 R,G,B 히스토그램에서 각 채널별로 Max,min 값을 찾아 Stretching 연산을 수행한 후 합치면 color contrast stretching을 할 수 있다.

아래는 color contrast stretching 영상이다.  


**Histogram equalization**

**Code**

void m\_equalization(Mat img){

Mat eq = cvCreateMat(img.rows,img.cols,0);

int x,y,i,k;

int acc\_hist =0;

float N = (img.rows\*img.cols)/256;

int hist[256] = {0},sum[256]={0};

for(y=0;y<img.rows;y++)

for(x=0;x<img.cols;x++)

hist[(int)img.at<uchar>(y,x)]++; ->히스토그램 생성

for(i=0;i<256;i++){

acc\_hist+=hist[i];

**sum[i] = ((int)acc\_hist/N);** ->각 픽셀에 정규화 값 대입

}

for(y=0;y<img.rows;y++)

for(x=0;x<img.cols;x++)

eq.at<uchar>(y,x) = sum[img.at<uchar>(y,x)];

->이미지에 대입

m\_show(eq, ,SAVE+”\_Equalization”);

}

**Histogram equalization**

**Image**

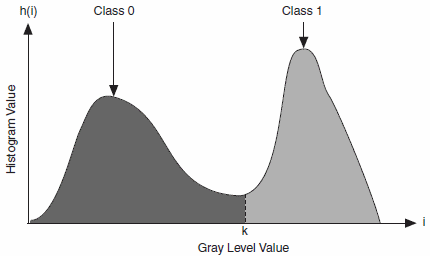
|  |  |
| --- | --- |
| 원본 | 변환 |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Eql(1).jpg1 |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Eql(1).jpg2 |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Eql(1).jpg3 |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Eql(1).jpg4 |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\지문-gray.bmp | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Eql(0).jpg5 |

**Histogram equalization**

**Discussion**

Histogram equalization은 픽셀에 정규화 값을 대입하여 영상을 개선시키는 방법이다.

영상 전체적인 명암을 고르게 만들어주는 기능을 한다. 그렇기 때문에 한쪽으로 치우쳐진 명도 값을 개선해준다.

Stretching 과의 명확한 차이는 **고르게** 해준다는 것에 있다. 만약에 분포가 2개 이상의 부분으로 쏠려 있는 경우 효과적인 개선을 기대할 수 있다. Stretching의 경우 0부분과 255, 2부분에 치우쳐진 영상에서는 효과를 기대할 수 없다. 그러나 equalization 은 개선 효과가 나타난다.

이와 같은 histogram에서는 equalization 이 효과적이다.

평활화는 2,5번 영상에서 보이다시피 밝은 부분까지 전체적으로 낮춰준다. 그러므로 텍스트나 지문 영상을 개선할 때에는 사용할 수 없다.

만약 평활화를 부분적으로 하고 싶다면

for(i=0;i<256;i++){

acc\_hist+=hist[i];

**sum[i] = ((int)acc\_hist/N);**

여기서 i의 값을 변경하면 되고

전체적으로 밝거나 어두운 영상을 만들고 싶으면 정규화 값 **(acc\_hist)** 를 조정하면 된다.

**Thresholding**

**Code**

void m\_thresholding(Mat img,int threshold\_value){

Mat thresh\_img = cvCreateMat(img.rows,img.cols,CV\_8UC1);

int x,y;

for(y=0;y<img.rows;y++)

{

for(x=0;x<img.cols;x++)

{

int k=(int)(img.at<uchar>(y,x));

if(**k<=threshold\_value**)

**thresh\_img.at<uchar>(y,x) = 0;**

else

**thresh\_img.at<uchar>(y,x) = 255;**

}

}

m\_show(thresh\_img, ,SAVE+”\_Thresholding”);

}

**Thresholding**

**Image**

|  |  |
| --- | --- |
| **원본** | **변환** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(1).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(2).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(3).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(4).jpg1** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(1).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(2).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(3).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(4).jpg2** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(1).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(2).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(3).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(4).jpg3** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Threshold(2).jpg4** |

**Thresholding**

**Discussion**

Thresholding 은 영상을 흰색(255)과 검은색(0) 값을 갖는 영상으로 전환한다.

이를 구현하기 위해 특정 값 ‘x’를 지정하고 영상의 각 픽셀마다 비교연산을 한다.

픽셀의 값이 x 보다 작으면 0, 크면 255로 값을 변경하여 생성하면 위와 같은 영상이 나타난다.

위의 표에서 변환된 4개의 사진들은 x값을 각 120/150/180/210으로 두어 변환한 영상들이다.

X의 값이 높을수록 전체적으로 검은색이 많다. 당연히도 기준이 올라가기 때문에 0 이 되는 픽셀들이 많아진 것이다.

Tresholding 기법은 1번 영상처럼 지문을 분석하거나 문자를 분석할 때 활용할 수 있다.

또한 2번과 3번 영상에서 보이는 것과 같이 이진화를 통해 선, 길을 구분할 수 있어 활용성이 높다.

Thresholding에서 가장 중요한 것은 x 값을 무엇으로 할 것이냐 이다.

지금부터 x 값을 찾기 위한 여러 방법들에 대해 논의해보겠다.

**OTSU**

OTSU 알고리즘은 특정 값 T를 기준으로 두개의 부분으로 나누었을 때 각 부분의 분포의 균일성이 높을수록 이진화에 적합한 값이 된다는 원리로 T를 찾는 알고리즘이다.

여기서 균일성은 각 부분의 분산으로 측정하고 분산이 작을수록 균일성이 높다.

알고리즘은 다음과 같다

1. 히스토그램을 계산하고
2. 초기값 A(0) B(0)를 설정한다.
3. T=1 … 255까지 비교

1.Ai, Bi를 갱신한다

2.T에 대한 분산 C(t) 계산한다.

4. 최대값 C(t)를 찾는다.

.

**Locally adaptive thresholding**

이 방법은 x 값을 픽셀마다 다르게 하여 이진화 하는 방법이다.

특정 픽셀을 기준으로 n X n 개의 픽셀에 대하여 이진화를 개별적으로 진행하는 것이다.

그렇기 때문에 n과 개별 x의 값에 따라 이진화 결과가 달라진다.

이 두가지 방법은 openCV에서 함수를 제공한다.

Otsu 는

**void threshold( , , , , );**

- 입력 array

- 출력 array

– threshold 값보다 작을 경우 지정될 값 (ex-0)

– threshold 값보다 클 경우 지정될 값(ex-255)

– threshold 값, OTSU를 사용하기 위해서 ‘THRESH\_BINARY | THRESH\_OTSU’ 라는 인자를 사용해야 한다.

Adaptive threhsholding 은

**void adaptivethreshold( , , , , , , );**

- 입력 array

- 출력 array

– threshold 값보다 클 경우 지정될 값 (ex-255)

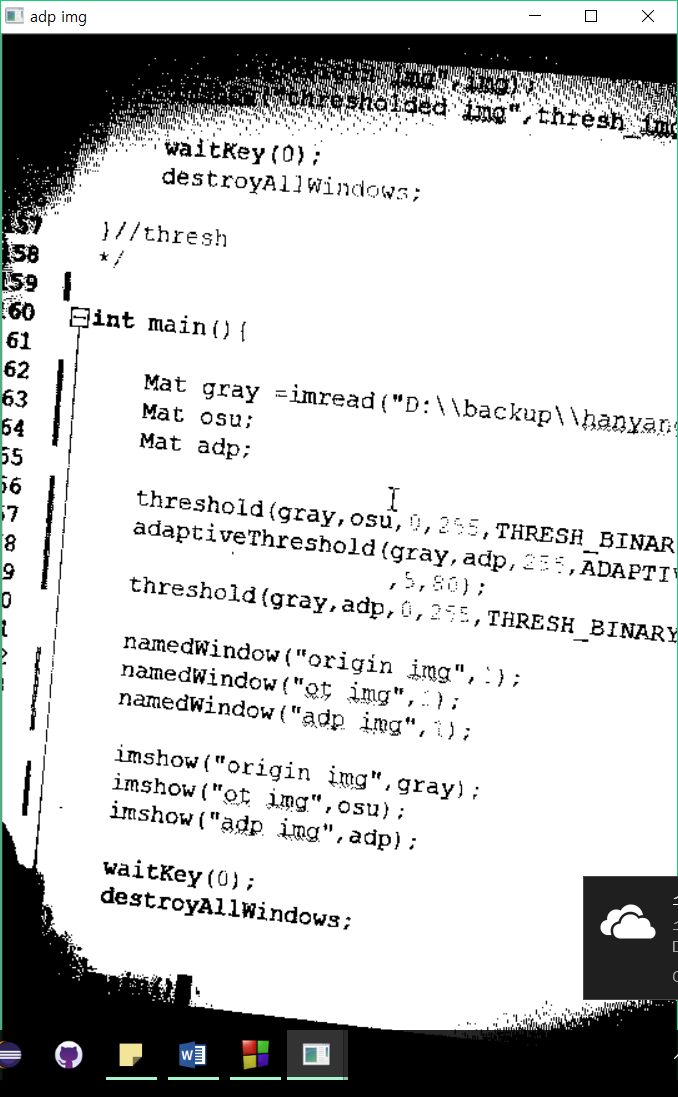
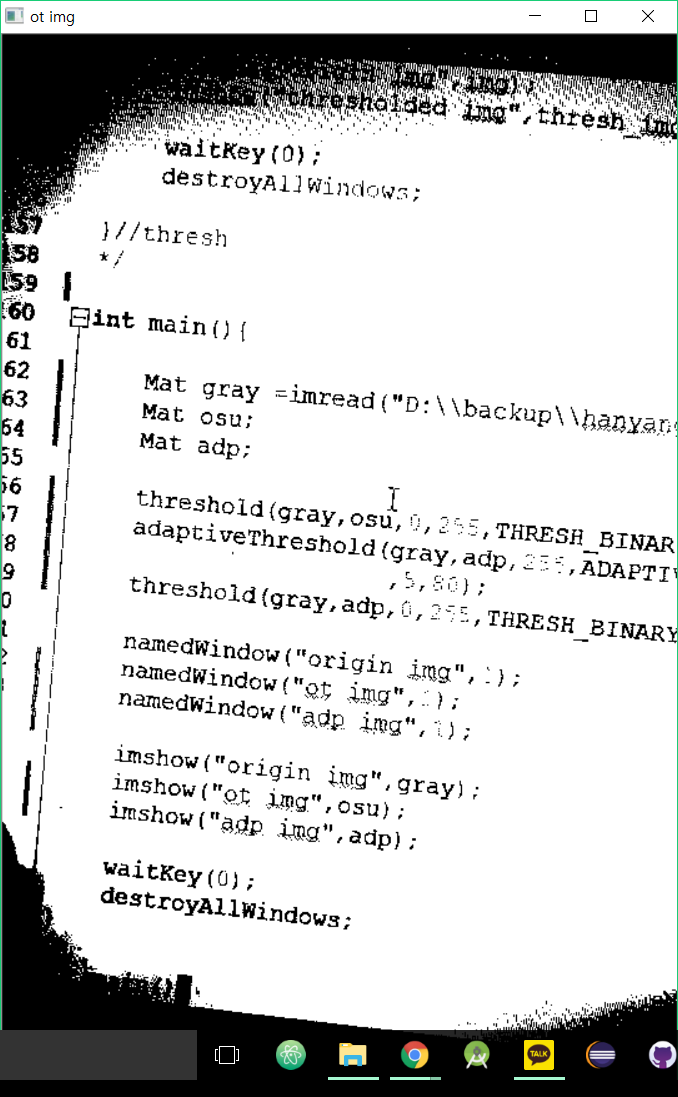
– thresholding 알고리즘 (ex -ADAPTIVE\_THRESH\_MEAN\_C)

– thresholding 타입 (ex-THRESH\_BINARY)

– 계산하는 블록의 크기 (위에서 말한 n)

– 계산된 평균으로부터 뺄 상수값

아래 두 영상은 opencv의 내장함수를 사용하여 otsu-threshold 와 adaptive-threshold를 사용한 것이다.



보이는 것과 같이 두 방법은 텍스트를 구분할 수 있는 좋은 성능을 가진다.

**Add & Subtract**

**Code**

void m\_Add(Mat img1,Mat img2,float alpha,float beta){

Mat plus = cvCreateMat(img1.rows,img1.cols,CV\_8UC1);

**plus = img1\*alpha +img2\*beta;**

m\_show(plus, ,SAVE+”\_Add”);

}

void m\_Sub(Mat img1,Mat img2, float alpha,float beta){

Mat minus = cvCreateMat(img1.rows,img1.cols,CV\_8UC1);

**minus = img1\*alpha - img2\*beta;**

m\_show(minus,SAVE+”\_Sub”)

}

**Add & Subtract**

**Image**

|  |  |
| --- | --- |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(3).jpgadd1** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(3).jpgadd2** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(1).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(4).jpgsub1** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(4).jpgsub2** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(4).jpgsub3** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(3).jpgcustom1** |

**Add & Subtract**

**Discussion**

Add

두개 이상의 영상을 가지고 + 연산으로 합쳐진 영상을 만들 수 있다.

이는 단순하게 동일한 위치를 갖는 픽셀끼리 값을 더함으로써 구현할 수 있다.

여기서 주의할 것은 연산을 진행할 영상이 모두 동일한 픽셀을 갖어야 한다.

(사이즈 변경함수 **resize( )** )

add1 영상은 두개의 영상을 합한 영상이다. 단순히 합이므로 결과 영상이 255에 가깝게 되므로 영상에 흰부분이 많다.

add2 영상은 각 영상에 0.8, 0.2을 곱한 후 덧셈연산을 한 영상이다. 비율을 조절해 합쳤으므로 영상이 자연스럽게 합쳐져 있다.

비율의 합이 1이 넘지않게 설정한다면 결과 영상에서 255를 넘어가는 값은 없게 되므로 그것을 참고하여 각 영상 별 비율을 조절하는 것이 좋다.

sub1은 동일한 두개의 영상에 뺄셈연산을 한 결과이다. 같은 위치에 같은 값이 있으므로 모든 값이 0으로 되어 검은색으로 출력되는 것이다.

Sub2는 서로 다른 두 영상을 뺀 결과이다. Lena 영상에서 패턴 영상의 값을 뺏으므로 결과영상에서 패턴 영상이 어둡게 겹쳐서 보이게 된다.

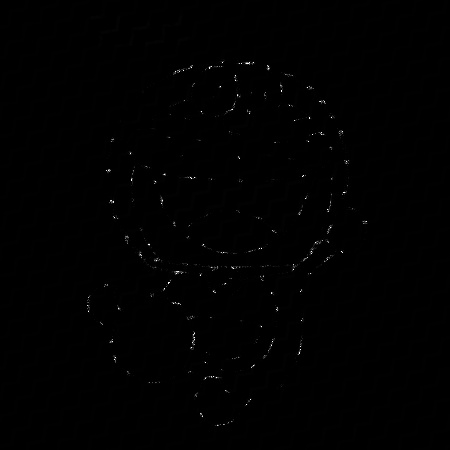
Sub3는 위의 add2와 같이 비율을 곱해서 연산한 결과이다. 영상에 약간 어두운 패턴을 주기 위해 활용할 수 있다.

이러한 영상의 뺄셈 연산은 인식 기술에서 많이 쓰인다. 두개의 영상의 차이가 명확하게 보이기 때문에 동일한 배경에서 특정 객체의 위치가 바뀌는 것을 뺄셈연산으로 인식할 수 있다.

추가적으로 나눗셈 연산을 시도해 보았다.

위의 영상의 도라에몽을 p1 패턴을 p2라 하고

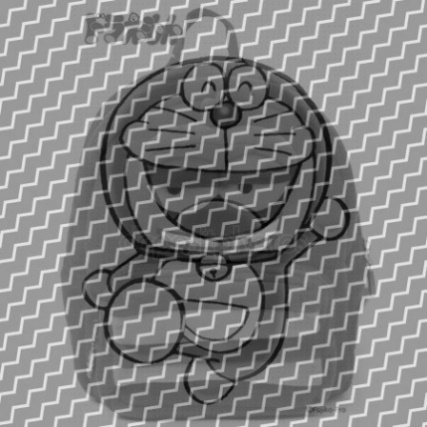
p1/p2와 p2/p1 연산을 해보았다.

****

보이는 것과 같이 형체를 구분할 수 있으나 어두워 형체를 인식하기가 어렵다.

나눗셈 연산은 별다른 특징이 없기 때문에 응용가능성은 적어 보인다.

마지막으로 3개의 영상에 대한 덧셈연산 결과이다.



3개의 영상을 합친 값은 매우 높게 나오기 때문에 각 영상에 0.3 씩 곱하여 합쳤다.

이와 같이 2개 이상의 영상도 덧셈연산으로 합칠 수 있다.

**출처**

<http://vision0814.tistory.com/71>

<http://darkpgmr.tistory.com/115>

<https://ko.wikipedia.org/wiki/HSV_%EC%83%89_%EA%B3%B5%EA%B0%84>

<http://mercury.kau.ac.kr/image_lab/%C4%C3%B7%AF.htm>

<http://blog.daum.net/_blog/BlogTypeView.do?blogid=050RH&articleno=12109315&categoryId=0&regdt=20141215103657>

<http://egloos.zum.com/kimhj8574/v/5728390>

<http://webnautes.tistory.com/1034>