멀티미디어정보처리

HW2

컴퓨터공학과

2013043401

**정지성**

제출일:17.04.12

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\_show(Mat,String);

void m\_Blur(Mat,float);

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

void m\_Nois(Mat);

void m\_Edge(Mat);

**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);

Blurring, Sharpening, Noise reduction, Edge detection

그러므로 아래 내용부터는 함수만 보여드리겠습니다.

**Blurring**

**Code**

void m\_Blur(Mat img,float A){

Mat st = cvCreateMat(img.rows,img.cols,CV\_8UC3);

float mask[] = {A,A,A, A,A,A, A,A,A};

Vec3b color;

int sum[3];

int cnt;

for(int y=1;y<img.rows-1;y++){ ->가장자리에 있는 픽셀 때문에

for(int x=1;x<img.cols-1;x++){ 1부터 시작한다.

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

sum[c]=0;

cnt =0;

for(int h=y-1;h<=y+1;h++){ ->배열의 수 만큼 반복한다.

for(int w=x-1;w<=x+1;w++){

color = img.at<Vec3b>(h,w); ->영상의 픽셀을 가져온다.

sum[0]+= color[0]\*mask[cnt]; ->각 픽셀의 RGB 값에 설정한 값을

sum[1]+= color[1]\*mask[cnt]; 곱한다.

sum[2]+= color[2]\*mask[cnt];

cnt++;

}

}

for(int c=0;c<3;c++){

if(sum[c]<0) ->예외처리

sum[c]=0;

else if(sum[c]>255)

sum[c]=255;

st.at<Vec3b>(y,x)[c]=sum[c]; ->Mat 변수 st에 값을 대입한다.

}

}

}

m\_show(st,SAVE+"\_BLUR");

}

**Blurring**

**Image**

|  |  |  |
| --- | --- | --- |
| **원본** | **3X3** | **5X5** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg1** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR_0(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR5_0(2).jpg** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg2** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR5_0(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR_0(2).jpg** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg3** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR5_0(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR_0(2).jpg** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg4** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR_0(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR5_0(2).jpg** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg5** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_BLUR5_0(2).jpg** |

**Blurring**

**Discussion**

Blurring은 정지영상을 흐릿하게 만드는 효과가 있다.

홀수 N 으로 N X N 행렬을 만들어 convolution한다.

이웃한 픽셀끼리 평균을 내어 동일한 값을 갖게 만들어 픽셀 간 값 차이를 줄인다.(경계선의 줄인다.) 근접 픽셀간 값이 동일하기 때문에 전체적인 영상이 흐릿하게 보이는 것이다.

Convolution을 할 때 로 한다. 예를 들어 3X3 인 경우 1/9 이고 5X5 인 경우 1/25를 곱하여 Blurring 효과를 얻는다. 즉 마스크의 총 합이 1이 되도록 만드는 것이다.

이 곱셈 값을 바꿔도 근접 픽셀 값은 같기 때문에 흐릿한 효과는 갖을 수 있지만 영상의 명도가 달라진다. 값이 기준보다 높으면 전체적으로 값을 올리는 것이기 때문에 영상이 밝아지고 낮으면 어두워진다.

(5X5에 1/5 와 1/125를 곱함)

Blurring은 convolution 크기가 클수록 더 강하게 나타난다.

더 많은 픽셀끼리 값이 비슷해 지기 때문에 전체적으로 더 흐릿해 지는 것이다. 위의 비교 영상에서 보이는 것처럼 3X3보다 5X5가 더 흐릿하다.

추가적으로 구현 시 관련하여 N x N 의 mask를 사용하기 위해서는 for문에 2/N 을 시작과 종료 조건으로 사용하여야 한다.

Blurring 은 smoothing 이라고도 부르고 위의 개념을 응용하여 아래 나올 noise 제거를 효과적으로 할 수 있다. 이에 관련된 내용은 아래에서 언급하겠다.

**sharpening**

**Code**

void m\_Shar(Mat img,float A,float B,float C){

Mat st = cvCreateMat(img.rows,img.cols,CV\_8UC3);

float mask[] = {A,B,A,

B,C,B,

A,B,A};

Vec3b color;

int sum[3];

int cnt;

for(int y=1;y<img.rows-1;y++){

for(int x=1;x<img.cols-1;x++){

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

sum[c]=0;

cnt =0;

for(int h=y-1;h<=y+1;h++){

for(int w=x-1;w<=x+1;w++){

color = img.at<Vec3b>(h,w);

sum[0]+= color[0]\*mask[cnt];

sum[1]+= color[1]\*mask[cnt];

sum[2]+= color[2]\*mask[cnt];

cnt++;

}

}

for(int c=0;c<3;c++){

if(sum[c]<0)

sum[c]=0;

else if(sum[c]>255)

sum[c]=255;

st.at<Vec3b>(y,x)[c]=sum[c];

}

}

}

m\_show(st,SAVE+"\_Shar");

}

**sharpening**

**Image**

|  |  |  |
| --- | --- | --- |
| **원본** | **Mask1** | **Mask2** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg1** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar-1(2).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar0(1).jpg** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg2** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar0(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar-1(2).jpg** |
| **3C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar0(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar-1(2).jpg** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg4** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar0(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar-1(2).jpg** |
| **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg5** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar0(1).jpg** | **C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Shar-1(2).jpg** |

**sharpening**

**Discussion**

Sharpening 은 영상을 선명하게 보여주는 효과가 있다.

Blurring 과는 반대로 영상의 경계선 값을 더 높여 선을 날카롭게 만든다.

영상에 mask convolution을 통해 구현할 수 있다. Mask의 가운데에 있는 픽셀에 주변 픽셀을 수치만큼 곱하여 대입한다. 곱셈 연산 후 덧셈연산을 수행하기 때문에 근접 픽셀 간 수치의 차가 높을수록 결과값이 더 높아져서 경계 값, 윤곽선이 도드라져 보이는 것이다.

경계값을 높이기 위해서는 mask 값을 설정하는 것이 중요하다.

위의 영상에서 mask는 이렇게 설정하였다.

|  |  |  |
| --- | --- | --- |
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| -1 | 9 | -1 |
| -1 | -1 | -1 |

Mask1 Mask2

Mask 의 수치를 보면, 단순히 가운데 값이 높은 것이 아니라 전체의 합이 1이 된다.

Mask의 총 합이 1이 아닌 경우 영상의 결과가 크게 차이가 난다.

합이 0인 경우를 보면(5번 영상)

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |



이렇게 윤곽선만 찾기 위해서는 합을 0으로 설정하면 된다.

합을 음수로 설정하면 이렇게 윤곽선이 잘 보이지 않는다. (더 내리면 검은색 화면)

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| -1 | 8 | -1 |
| -1 | -1 | -1 |



총 합을 3으로 해 보았다.

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| -1 | 11 | -1 |
| -1 | -1 | -1 |



이렇게 윤곽은 보이지만 기존 영상보다 명도가 더 높아진다.

그러므로 기존 영상에 선명한 효과를 주기 위해서는 합이 1이 되도록 설정하여야 한다.

추가적으로 sharpening으로 선명해진 영상에 sharpening을 한번 더 수행해 보았다.(3번영상)

|  |  |  |
| --- | --- | --- |
| 0 | -1 | 0 |
| -1 | 5 | -1 |
| 0 | -1 | 0 |



보이는 것과 같이 선명도가 개선되지만 근접 픽셀간 차이가 심하게 나기 때문에 좋은 효과를 보긴 어렵다.

**Noise reduction**

**Code**

void m\_Nois\_M(Mat img){ ->Median filtering

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

int sort\_array[9];

int cnt=0;

for(int y=1;y<img.rows-1;y++){

for(int x=1;x<img.cols-1;x++) {

cnt = 0;

for(int h=y-1;h<=y+1;h++){

for(int w=x-1;w<=x+1;w++){

sort\_array[cnt]= img.at<uchar>(h,w);

cnt++;

}

}

vector<int>myvector(sort\_array,sort\_array+9);

sort(myvector.begin(),myvector.end());

st.at<uchar>(y,x) = myvector.at(4);

}

}

m\_show(st,SAVE+"\_Med");

}

void m\_Nois\_A(Mat img, float A){ ->Averaging filtering

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

float mask[]={A,A,A,A,A,A,A,A,A};

Vec3b color;

int sum,cnt;

for(int y=1;y<img.rows-1;y++){

for(int x=1;x<img.cols-1;x++){

sum = 0;

cnt = 0;

for(int h=y-1;h<=y+1;h++){

for(int w=x-1;w<=x+1;w++){

sum += img.at<uchar>(h,w)\*mask[cnt];

cnt++;

}

}

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

}

}

m\_show(st,SAVE+"\_Ave");

}

**Noise reduction**

**Image**

|  |  |
| --- | --- |
| Impulse noise | Noise reduction |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Med(1).jpg |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Med(1).jpg |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Med(1).jpgC:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Med(1).jpg |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Med(1).jpg |

|  |  |
| --- | --- |
| Gaussian noise | Noise reduction |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Ave(1).jpg |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Ave(1).jpg |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\lena(GausiaanNoise).bmp | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Ave(1).jpg |
| C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save(0).jpg | C:\Users\jisung\AppData\Local\Microsoft\Windows\INetCache\Content.Word\save_Ave(1).jpg |

**Noise reduction**

**Discussion**

일단 영상에서의 noise의 종류에 대해 알아보겠다.

영상에서 노이즈는

uniform noise / Gaussian noise / impulse noise 등이 있다.

Uniform 노이즈는 양자화 노이즈 라고도 불리며 신호의 변환과정(아날로그->디지털)에서 0~255의 정수형으로 변환되기 때문에 소수부분의 값이 반올림되어 잡음이 발생하는 것이다.

Impulse 노이즈는 0과 255의 값을 가진 noise 픽셀이 생기는 것으로 소금과 후추가 뿌려진 것 같다고 해서 salt and pepper noise 라고도 불린다. 이러한 노이즈는 디지털화 과정에서 발생한다.

Gaussian 노이즈는 카메라의 렌즈로부터 들어오는 광학 신호를 변환하는 과정에서 센서에 생기는 노이즈이다. 이 노이즈는 가우시안 분포함수의 형태를 따르기 때문에 Gaussian 노이즈라고 불린다

영상에서 이러한 노이즈를 제거하기 위해 필터를 사용하는데 위에서 언급한 blurring의 기본적인

원리를 사용한다. 여기서는 여러 필터 중 Impulse noise와 Gaussian noise를 제거할 수 있는 중간

값 필터와 평균값 필터에 대해서 알아보겠다.

중간값 필터는 NxN의 마스크에서 모든 요소를 정렬한 후 중간값을 찾아 가운데 픽셀에 저장하는 것이다.

예를들어 3x3 mask에서

|  |  |  |
| --- | --- | --- |
| 51 | 58 | 57 |
| 50 | 44 | 65 |
| 45 | 46 | 49 |

중간값이 50 이 되어 가운데 픽셀을 50으로 저장한다.(44-45-46-49-50-51-57-58-65)

만약에 impulse noise로 인해 값이 0이 된 경우 중간값으로 바뀌기 때문에 노이즈를 제거할 수 있는 것이다.

이러한 방법에는 몇가지 문제점이 있다.

위의 예시처럼 노이즈가 발생하지 않은 경우에도 픽셀을 변경시키게 될 수 있고 노이즈가 연달아 있을 때 제거가 안될 수 있다. 첫번째 문제는 알고리즘 상 어쩔 수 없는 문제점이지만 두번째 문제의 경우 mask 크기를 크게 만들어 해결할 수 있다. (하지만 크기가 커지면 blurring 효과가 심해지기 때문에 주의해야한다.)

4번째 영상을 보면 작업 후 영상에 노이즈가 여전히 존재하는 것을 볼 수 있다.

이러한 경우 앞서 말한 것 처럼 mask의 size를 넓히면 된다.



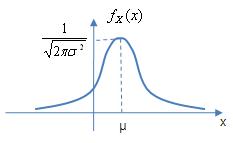
이렇게 노이즈는 제거했지만 blurring 효과가 있다.

평균값 필터는 중간값 필터와 유사하지만 근접 픽셀의 평균값을 내어 설정하는 것이다.

예를 들어 아래와 같은 경우는 50이다.

|  |  |  |
| --- | --- | --- |
| 50 | 40 | 60 |
| 30 | 20 | 70 |
| 40 | 60 | 80 |

가우시안 노이즈는



이렇게 종의 모양으로 분포 되어있다. 그렇게 때문에 주변 픽셀에 평균을 구하여 노이즈를 제거하는 것이다.

**Edge detection**

**Code**

void m\_Edge(Mat img){

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

float maskX[]={-1,0,1,-2,0,2,-1,0,1};

float maskY[]={-1,-2,-1,0,0,0,1,2,1};

Vec3b color;

int sumX,sumY,cnt;

for(int y=1;y<img.rows-1;y++){

for(int x=1;x<img.cols-1;x++)

{

sumX = 0;

sumY = 0;

cnt = 0;

for(int h=y-1;h<=y+1;h++){

for(int w=x-1;w<=x+1;w++){

sumX += img.at<uchar>(h,w)\*maskX[cnt]; ->x방향

sumY += img.at<uchar>(h,w)\*maskY[cnt]; ->y방향

cnt++;

}

}

st.at<uchar>(y,x) = abs(sumX)+abs(sumY); ->절대값을 씌운다.

}

}

}

**Edge detection**

**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_EDGE_(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_EDGE_(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_EDGE_(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_EDGE_(1).jpg4** |

**Edge detection**

**Discussion**

영상에서 Edge를 검출하기 위해서는 앞서 sharpening에서 언급한 것처럼 mask 요소의 합이 0이 되도록 설정해야 한다.

여기서는 여러 검출 방법 중 sobel mask를 사용하여 윤곽선을 검출하였다.

Sobel mask는 수직 마스크와 수평 마스크 2개를 사용하여 x,y축에 대하여 convolution 을 진행한다.

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| -1 | 2 | 1 |

이렇게 2개의 mask를 사용한다.

기본원리를 보자면

가운데 픽셀을 기준으로 왼쪽 픽셀과 오른쪽 픽셀의 값을 a,b 하고,

만약 a=b 이면 픽셀 값이 b-a=0 이 된다.

즉, 픽셀의 차이가 적을수록 명도가 낮아지고 픽셀의 차이가 클수록 명도가 높아져 Edge가 보이게 되는 것이다. (절대값을 사용한다.)

이는 변화율의 개념으로 미분값을 찾는 것과 동일하다.

Soble edge detection은 수직과 수평의 mask를 사용한 결과물을 합친다.

4번째 영상에 대해서 x mask 와 y mask 만 사용해서 결과를 뽑아보았다.



이 영상에서 보면 머리 왼쪽의 머리카락 부분에서 edge 가 많이 검출된 것을 볼 수 있다.

그리고 소파 부분에는 검출이 거의 안되었다.

이는 수평축 edge는 잘 찾으나 수직축 edge를 찾지 못한 것을 보여준다.



이 영상은 왼쪽의 머리카락부분에서는 거의 찾지 못했지만 소파부분의 edge를 검출하였다.

그러므로 Edge를 좀 더 잘 찾기 위해서 두개의 영상을 합치는 연산을 한다.

Sobel mask은 수직 수평의 edge보다 대각선 edge를 더 잘 검출한다. 대각선의 경우 두가지 mask 에서 검출이 되고 그 값이 합쳐지므로 잘 검출되는 것이다.

Edge 검출에 사용하는 mask 는 sobel 외에도 많은데 대표적으로 Prewitt 와 Robert 가 있다.

마스크는 다음과 같다.

Prewitt

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Robert

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| -1 | 0 | 0 |

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | -1 |

Prewitt mask 의 특징은 sobel 에 비해 크기 변화 값이 적기 때문에 edge 가 약하게 보인다(비교적).

Robert mask는 연산 수가 적기 때문에 다른 마스크에 비해 빠른 속도를 갖는다. 그러나 앞서 말한 이유로 edge 가 약한 특징이 있다

출처

<http://blog.daum.net/trts1004/12109067>

<http://m.blog.naver.com/roboholic84/220482877717>

<http://webnautes.tistory.com/1044>

<http://blog.naver.com/PostView.nhn?blogId=wlsgk123123&logNo=10093294588&redirect=Dlog&widgetTypeCall=true>

<http://m.blog.naver.com/neverabandon/100053914488>

<http://forum.falinux.com/zbxe/index.php?document_srl=547769&mid=lecture_tip>

<http://sensitive04.tistory.com/11>

<http://terms.naver.com/entry.nhn?docId=854058&cid=42346&categoryId=42346>

<http://linecard.tistory.com/59>

<http://iskim3068.tistory.com/50>