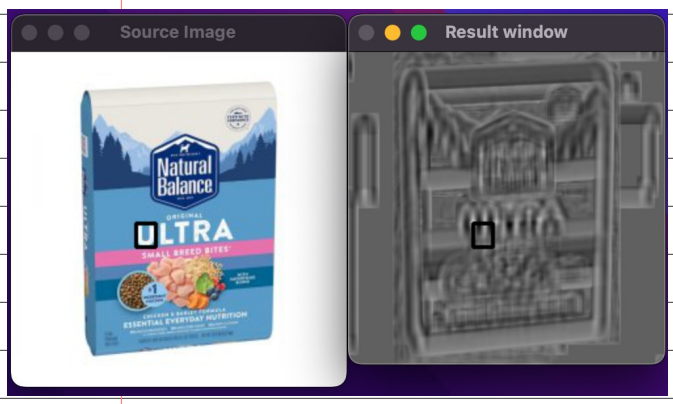
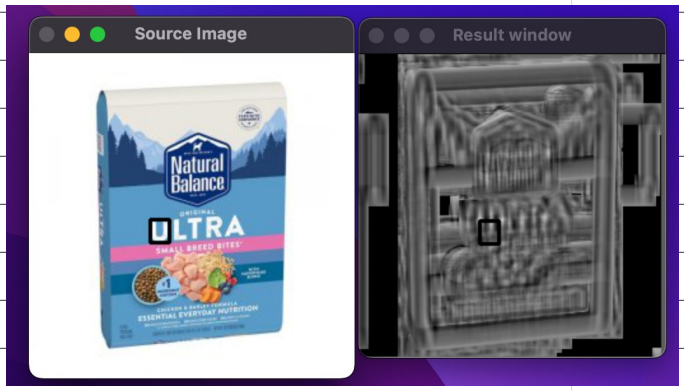
멀티미디어 처리 및 HW#3

2016114463
이인형

1. 전체코드 (2~3p)

2.



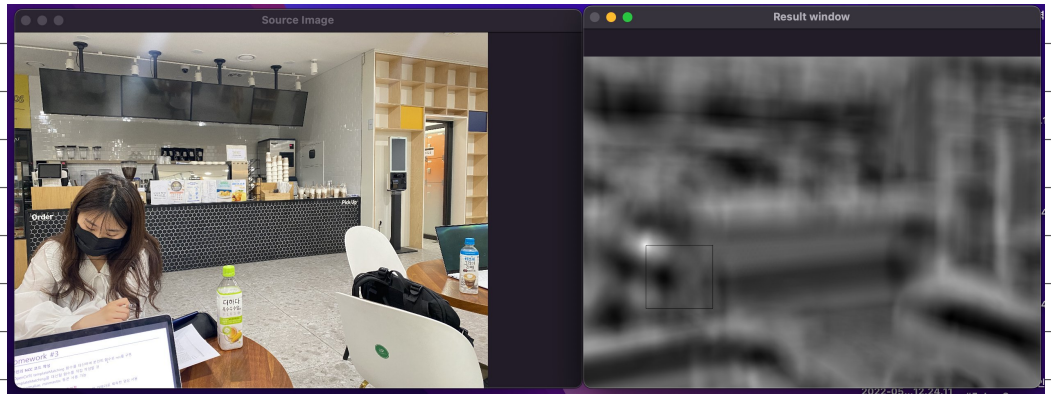


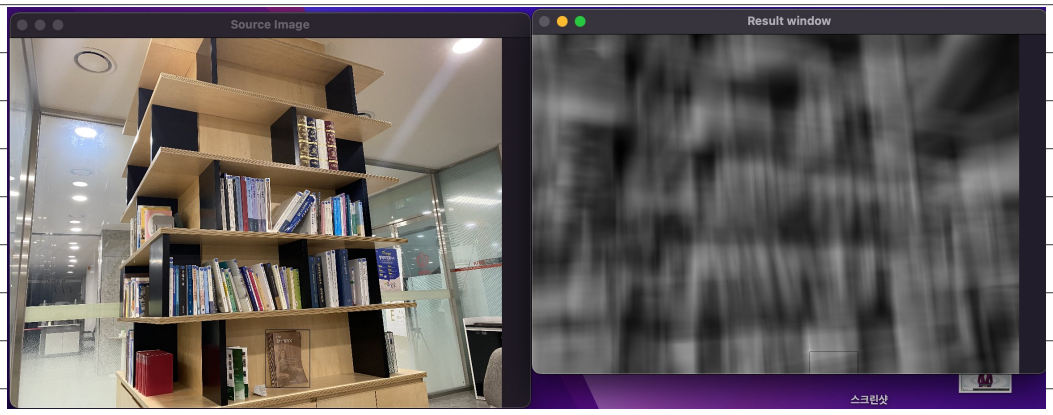
↳ 내가만든 NCC 결과

↳ opencv NCC 결과.

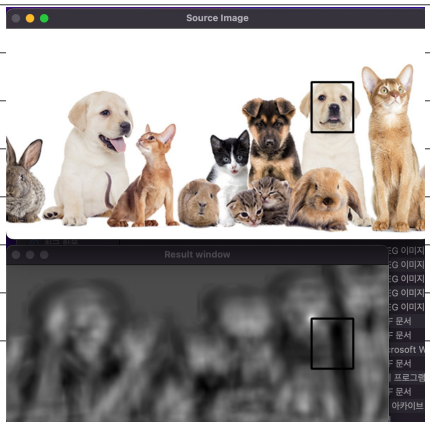
⇒ 결과에 차이는 있었지만 opencv NCC가 더 빨리 작동.

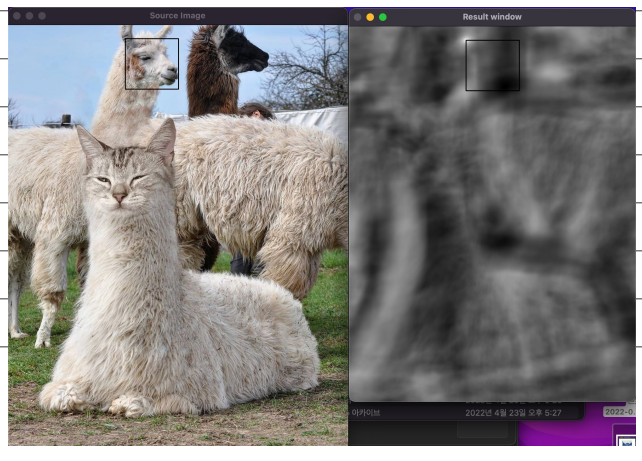
3.





3.



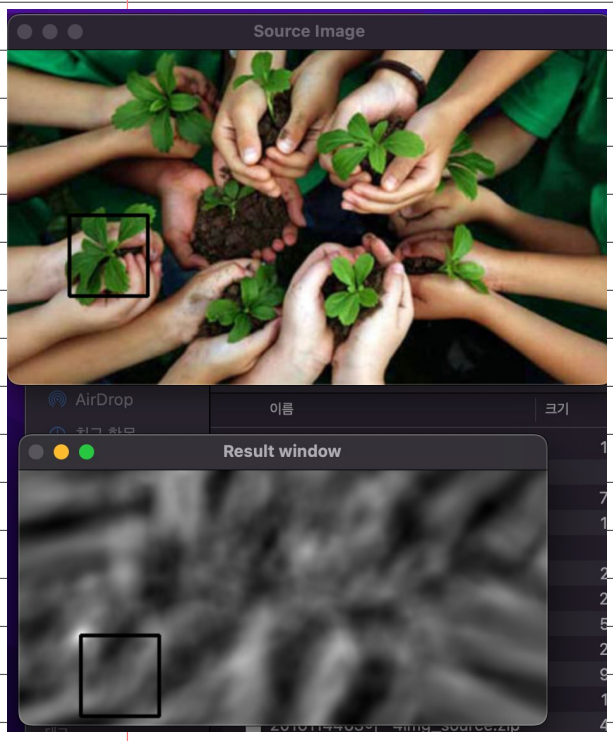


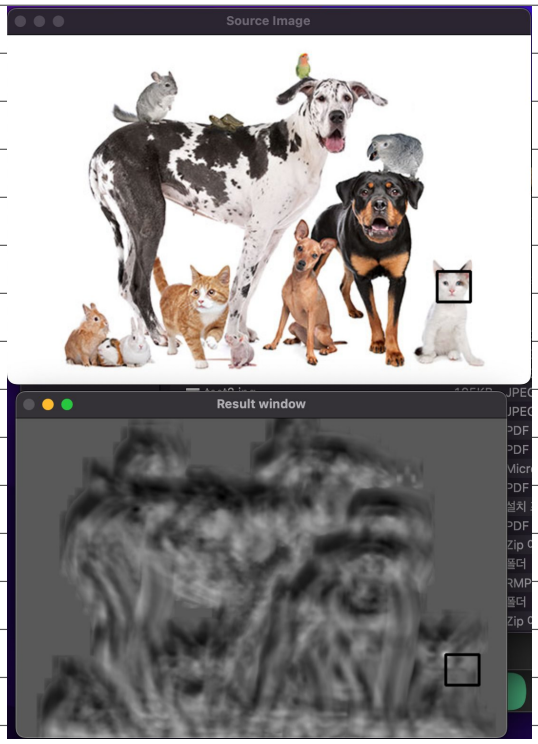
```cpp
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include <iostream>
#include <cmath>
using namespace std;
using namespace cv;
bool use_mask;
Mat img; Mat templ; Mat mask; Mat result;
const char* image_window = "Source Image";
const char* result_window = "Result window";
int match_method;
int max_Trackbar = 5;
int function_1(string image_name, string template_name, int method);
void MatchingMethod( int, void* ,int method);
void matchTemplate_my(Mat img1, Mat img2,Mat result);


int main( int argc, char** argv )
{
    cout <<argv[0]<<endl;
    if(function_1("source4.jpeg","crop4.jpeg",0))return EXIT_FAILURE;
    if(function_1("source1.jpeg","crop1.jpeg",1))return EXIT_FAILURE;
    if(function_1("source2.jpeg","crop2.jpeg",1))return EXIT_FAILURE;
    if(function_1("source3.jpeg","crop3.jpeg",1))return EXIT_FAILURE;
    if(function_1("source4.jpeg","crop4.jpeg",1))return EXIT_FAILURE;
    return 0;

}

int function_1(string image_name, string template_name, int method){
    //method 1 : opencv , 0 : myfunction
    img = imread( image_name, IMREAD_COLOR );
    templ = imread( template_name, IMREAD_COLOR );
    if(img.empty() || templ.empty() || (use_mask && mask.empty()))
    {
        cout << "Can't read one of the images" << endl;
        return EXIT_FAILURE;
    }
    namedWindow( image_window, WINDOW_AUTOSIZE );
    namedWindow( result_window, WINDOW_AUTOSIZE );
    MatchingMethod( 0, 0 ,method);
    waitKey(0);
    return EXIT_SUCCESS;
}

void MatchingMethod( int, void* ,int method)
{
    Mat img_display;
    img.copyTo( img_display );
    int result_cols = img.cols - templ.cols + 1;
    int result_rows = img.rows - templ.rows + 1;
    result.create( result_rows, result_cols, CV_32FC1 );
    cout << "시작" <<endl;
    if(!method){
```

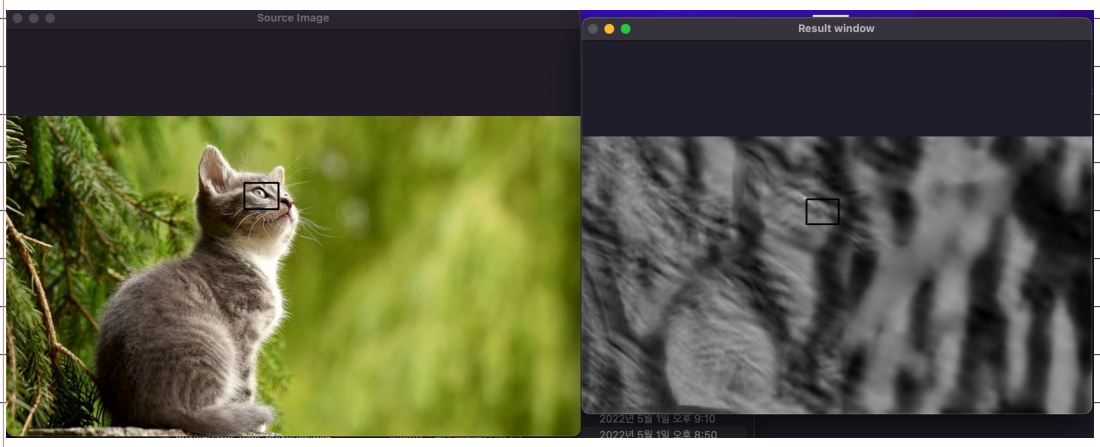
0 : myfunction.

1 : opencv

```
        matchTemplate_my( img, templ, result);
    }else{
        matchTemplate( img, templ, result, TM_CCOEFF_NORMED, mask);
    }
    cout << "matchTemplate완료" <<endl;
    normalize( result, result, 0, 1, NORM_MINMAX, -1, Mat() );
    double minVal; double maxVal; Point minLoc; Point maxLoc;
    Point matchLoc;
    minMaxLoc( result, &minVal, &maxVal, &minLoc, &maxLoc, Mat() );
    matchLoc = maxLoc;
    rectangle( img_display, matchLoc, Point( matchLoc.x + templ.cols ,
     matchLoc.y + templ.rows ), Scalar::all(0), 2, 8, 0 );
    rectangle( result, matchLoc, Point( matchLoc.x + templ.cols ,
     matchLoc.y + templ.rows ), Scalar::all(0), 2, 8, 0 );
    imshow( image_window, img_display );
    imshow( result_window, result );
    return;
}

void matchTemplate_my(Mat img1_c, Mat img2_c,Mat result){
    Mat img1, img2;
    Mat img1_part;

    cvtColor( img1_c, img1, COLOR_RGB2GRAY );
    cvtColor( img2_c, img2, COLOR_RGB2GRAY );
    int rows = img1.rows, cols = img1.cols;
    int rows2 = img2.rows, cols2 = img2.rows;
    img1_part.create(rows2,cols2,CV_8UC1);
    for(int i=0;i<(rows - rows2);i++){
        for(int j=0;j<(cols - cols2);j++){

            for(int k=0;k< rows2 ;k++){
                for(int l=0;l< cols2 ;l++){
                    img1_part.at<uchar>(k,l) = img1.at<uchar>(i+k,j+l);
                }
            }
            float img1mean = mean(img1_part)[0];
            float img2mean = mean(img1)[0];
            float a=0,b=0,c=0;

            for(int k=0;k< rows2 ;k++){
                for(int l=0;l< cols2 ;l++){
                    a += (img2.at<uchar>(k,l) -
                     img2mean)*(img1_part.at<uchar>(k,l)-img1mean);
                    b += pow(img2.at<uchar>(k,l)-img2mean,2);
                    c += pow(img1_part.at<uchar>(k,l)-img1mean,2);
                }
            }
            result.at<float>(i,j) = a/sqrt(b*c);
        }

    }
    return;
}
```







=> 받은 이미지를 compile 한 후가. 결과이 잘 되는 것을 확인할 수 있다.

4. opencv와 내가 만든 ncc로다 결과는 동일했지만, optimalization 측면에서 opencv가 월등히 빠른 결과속도를 보였다.