[3 일차]

[threshold 조정 : 도로 라인 따기　threshold.cpp]

threshold 조정해도 도로 라인 안잡히면? → 광이 있든 없든 도로라인 경계선에서는 색 변화가 급격함
→ 미분해서 변화량 큰 부분 찾는다.
참고: 이미지센싱 - 가시광선류 센싱이 안잡히는 경우 레이더(전자기파)로 잡아야함.
→ 미분 픽셀 따고, 선형회기(linear regression) c 로 만들어서 분석 → 도로라인잡힘

<코드>

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

Mat edge_detect(Mat blur)
{
	Mat output;
	cvtColor(blur, output, COLOR_RGB2GRAY);
	threshold(output, output, 140, 255, THRESH_BINARY);
	// https://www.tutorialspoint.com/opencv/opencv_simple_threshold.htm
	return output;
}

int main(int argc, char **argv)
{
	Mat img = imread("sample.jpg", -1);
	Mat blur;
	Mat edge;

	GaussianBlur(img, blur, Size(3,3),0,0);
	edge = edge_detect(blur);

	imwrite("org_img.jpg", img);
	imwrite("edge.jpg", edge);

	imshow("Origin Image", img);
	imshow("Edge Image", edge);

	waitKey(0);

	destroyWindow("Origin Image");
	destroyWindow("Edge Image");

	return 0;
}
```

<도로선 잘 나왔다>

[sobel_differential_filter.cpp]

소벨 필터 : 1 계 미방임. 1 계 특징 : 노이즈에 민감. 가장 강한 엣지 검출에 사용.
2 계는 라플라시안이라고함. 역시 노이즈에민감. 관심부분 엣지 검출에 사용. 둘다 노이즈때문에 gaussian blur 사용.

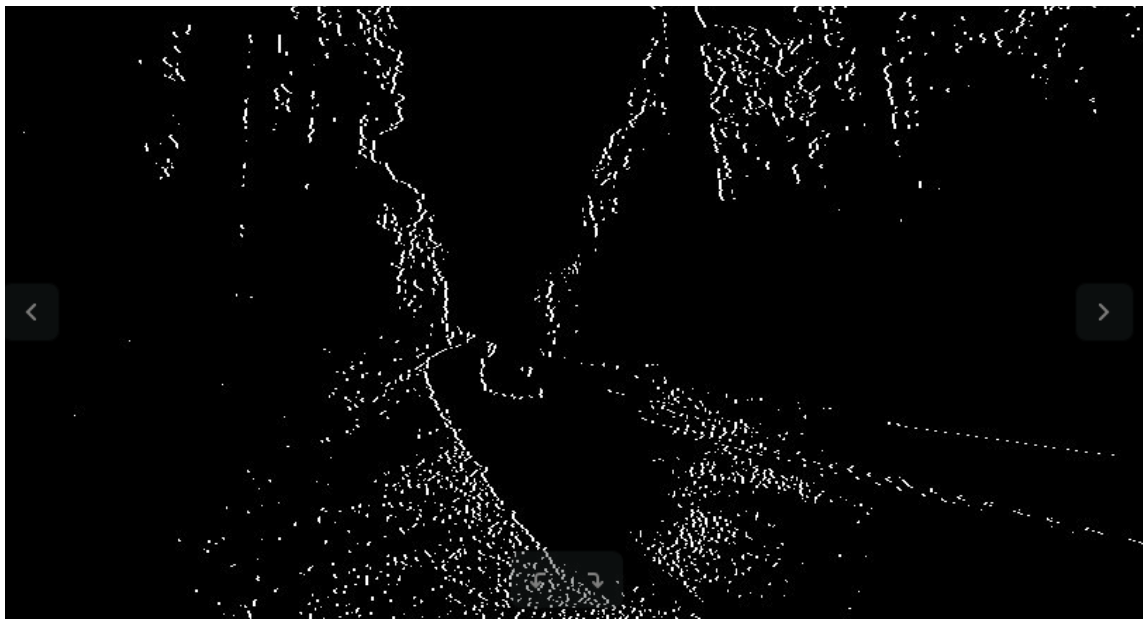
https://www.researchgate.net/post/What_are_the_differences_in_first_order_derivative_edge_detection_algorithms_and_second_order_edge_detection_algorithms

<코드>

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

Mat edge_detect(Mat blur)
{
	Mat output;
	cvtColor(blur, output, COLOR_RGB2GRAY);		//gray 로바꿈
	threshold(output, output, 140, 255, THRESH_BINARY);	// 140->40 해서 흐린 사진 도 라인 땄다.
	// https://www.tutorialspoint.com/opencv/opencv_simple_threshold.htm
	Point anchor = Point(-1,-1);  //-1,1 해야하는데 지금 안돌아가서 -1,-1 로함. kernel 의중앙으로?
	Mat kernel = Mat(1,3,CV_32F); // Mat(rows, cols, type);
			//CV_[The number of bits per item][Signed or Unsigned][Type Prefix]C[The channel number]

	kernel.at<float>(0,0) = -1;
	kernel.at<float>(0,1) = 0;
	kernel.at<float>(0,2) = 1;

	filter2D(output, output, -1, kernel, anchor, 0, BORDER_DEFAULT);  //mask matrix=kernel
	//이 다음에는 직접 filter2D 를 만들 것이다. 라플라스변환이랑 수학 부분 알아야 이해 가능.
	// https://www.tutorialspoint.com/opencv/opencv_filter2d.htm
	return output;
}

int main(int argc, char **argv)
{
	Mat img = imread("sample.jpg", -1);
	Mat blur;
	Mat edge;

	GaussianBlur(img, blur, Size(3,3),0,0);

	edge = edge_detect(blur);

	imwrite("org_img.jpg", img);
	imwrite("differential.jpg", edge);

	imshow("Origin Image", img);
	imshow("First Order Differential Image", edge);

	waitKey(0);

	destroyWindow("Origin Image");
	destroyWindow("First Order Differential Image");

	return 0;
}
```

<결과> // treshold 145

threshold 40 아래 사진



[custom filter2D : laplace_conveolution_filter2d.cpp]

//sobel x filter. -> y 로 하려면 가운데에서 i 위치를 바꾸는형태로가면됨

<코드>

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

void custom_filter2d(Mat in, Mat out, int depth, Mat kernel, Point anchor, int blas, int
border_type)
{

        int i, j, k;
        int tmp=0;
        Mat t=in.clone();      // = copyTo(); but with extra memory allocation.
                               // https://blog.naver.com/pckbj123/100202476334 원본 메모리그대로참조 vs 새로할당
        for(i=0;i<5;i++)       // 5x5 매트릭스를 1x3 으로 스캔하면서 convolution 하고 있다.
        {
                out.at<float>(i,0)= //테두리
                            t.at<float>(i,1)*kernel.at<float>(0,0)+
                            t.at<float>(i,0)*kernel.at<float>(0,1)+
                            t.at<float>(i,1)*kernel.at<float>(0,2);
                for(j=1;j<4;j++)
                {
                out.at<float>(i,j)= //가운데.엥커가하는일이 테투리할지거를지결정.(-1,-1)이면테두리거름
                            t.at<float>(i,j-1)*kernel.at<float>(0,0)+
                            t.at<float>(i,j)*kernel.at<float>(0,1)+
                            t.at<float>(i,j+1)*kernel.at<float>(0,2);
                }
                out.at<float>(i,4)= //테두리. 테두리가있는 영상 처리에 필요함.
                            t.at<float>(i,3)*kernel.at<float>(0,0)+
                            t.at<float>(i,4)*kernel.at<float>(0,1)+
                            t.at<float>(i,3)*kernel.at<float>(0,2);//5 못하니까 이거로해놈.
anchor-1,1 이면 이전값쓴다.

        }
        //i,0 i,1 에 convolution 을 때려넣고있다. anchor-1-1 이니까지금 101 근데-1,1 이엇으면 012
}       //값이 변하는 위치 자체는 값이 0 인데 값 변하는 근처 값이 1 로 세팅됨.그래서결과가 급변위치에 점점점이 들어가있었다.
int main(int argc, char **argv)
{
    //DSP 연산 주로 더하기,곱셈 으로 구성되어있는데, 미적분에 활용된다.
    Mat m=Mat::ones(5,5,CV_32F);
    Mat m2=Mat::ones(5,5,CV_32F);
    Mat m3=Mat::ones(5,5,CV_32F);

    Mat kernel=Mat(1,3,CV_32F);     //kernel 1x3 으로 설정. 3x3 보다 이해하기 쉬우니까 이걸로선정함.
    Point anchor = Point(-1,-1);    // -1,-1 하면 테두리 없으니 반대편값가져다가 convolution 한다.

    kernel.at<float>(0,0)=-1;       //kernel 1x3 안에 값 -1, 0, 1 로 함.
    kernel.at<float>(0,1)=0;
    kernel.at<float>(0,2)=1;

    m.at<float>(0,0)=0;         // 대각 에만 변화 값을 줬을 때, convolution 변화가 어떻게 나타나는지 볼 것이다.
    m.at<float>(2,2)=2;
    m.at<float>(4,4)=3;
      //.at https://webnautes.tistory.com/1169 .at<Vec3b>3 채널픽셀접근 .at
    cout << "m: " << endl << m << endl;
```

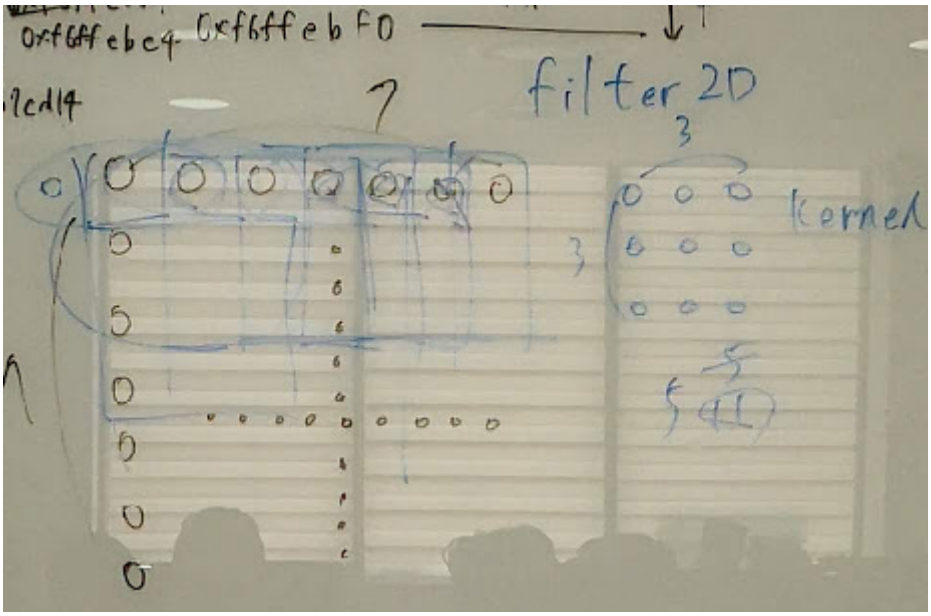
```
    custom_filter2d(m,m2,-1,kernel,anchor,0,BORDER_DEFAULT);
    cout<<"m2: "<< endl << m2 << endl;
    filter2D(m,m3,-1,kernel,anchor,0,BORDER_DEFAULT);
    cout<<"m3: "<< endl << m3 << endl;
      return 0;
}
```

```
m:
[0, 1, 1, 1, 1;
 1, 1, 1, 1, 1;
 1, 1, 2, 1, 1;
 1, 1, 1, 1, 1;
 1, 1, 1, 1, 3]
m2:
[0, 1, 0, 0, 0;
 0, 0, 0, 0, 0;
 0, 1, 0, -1, 0;
 0, 0, 0, 0, 0;
 0, 0, 0, 2, 0]
m3:
[0, 1, 0, 0, 0;
 0, 0, 0, 0, 0;
 0, 1, 0, -1, 0;
 0, 0, 0, 0, 0;
 0, 0, 0, 2, 0]
```

다 ones(,)인데, 대각성분 0,0 2,2 4,4 만 값 변화를 줬다.

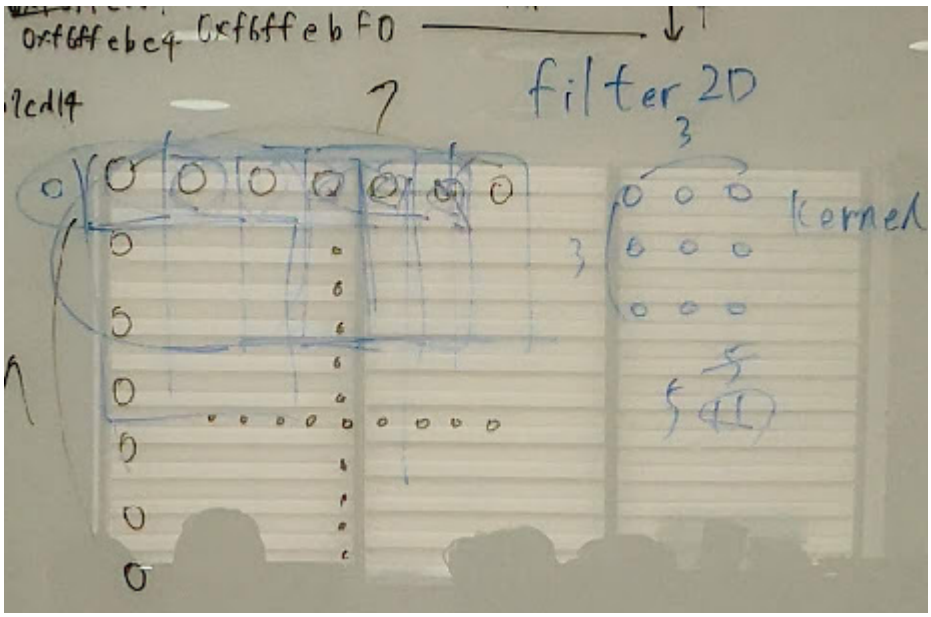
1x3 으로 스캔하면서 convolution 할 때, 값 변한애 주변으로 1,-1,2 생긴걸 확인할 수 있다.

filter2D 인데, custo_filter2D 와 결과가 같다.



[roi_sobel : roi_sobel.cpp]

convolution 개념. 인터넷에 쳐보면 다 1 차만 나온다. 영상은 3 차원이고 우리 하는건 2 차 그 개념은 아래와같다. 쭉 스캔
위의 예에서는 5x5 에서 1x3 으로 보기 쉽게 convolution 했다. convolution 해서 한칸에 저장. 이렇게 전체 이미지에대해수행
테두리 없으니까 가져와서 한부분있음→ 테두리지금표시안되어있다.



이제
roi 적용해서 차선을 실제로 뽑아야한다.

<코드>

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <stdio.h>

using namespace cv;
using namespace std;

Mat custom_roi(Mat img)
{
    Mat output;
    Mat mask = Mat::zeros(img.size(),img.type());

    Point pts[4]={
                Point(50,230),
                Point(820,230),
                Point(820,400),
                Point(50,400)
    };

    fillConvexPoly(mask, pts, 4, Scalar(255,255,255));
    bitwise_and(img, mask, output);
    return output;

}

Mat edge_detect(Mat blur)
{
      Mat output;
      cvtColor(blur, output, COLOR_RGB2GRAY);
      threshold(output, output, 40, 255, THRESH_BINARY);

      Point anchor = Point(-1,-1);        //앵커개 념//
```

https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/filter_2d/filter_2d.html

```
	Mat kernel = Mat(1,3,CV_32F);

	kernel.at<float>(0,0) = -1;
	kernel.at<float>(0,1) = 0;
	kernel.at<float>(0,2) = 1;

	filter2D(output, output, -1, kernel, anchor, 0, BORDER_DEFAULT);

	return output;
}

int main(int argc, char **argv)
{
	Mat img = imread("sample.jpg", -1);
	Mat blur;
	Mat edge;
	Mat croi;

	GaussianBlur(img, blur, Size(3,3),0,0);

	edge = edge_detect(blur);
	croi = custom_roi(edge);

	imwrite("org_img.jpg", img);
	imwrite("sobel_roi.jpg", croi);

	imshow("Origin Image", img);
	imshow("Filtered Image", croi);

	waitKey(0);

	destroyWindow("Origin Image");
	destroyWindow("Filtered Image");

	return 0;
}
```

[dsp_performance.cpp]//지금까지는 cortex A 를쓴거고 DSP 를 쓰지 않았다. 이제 써본다.

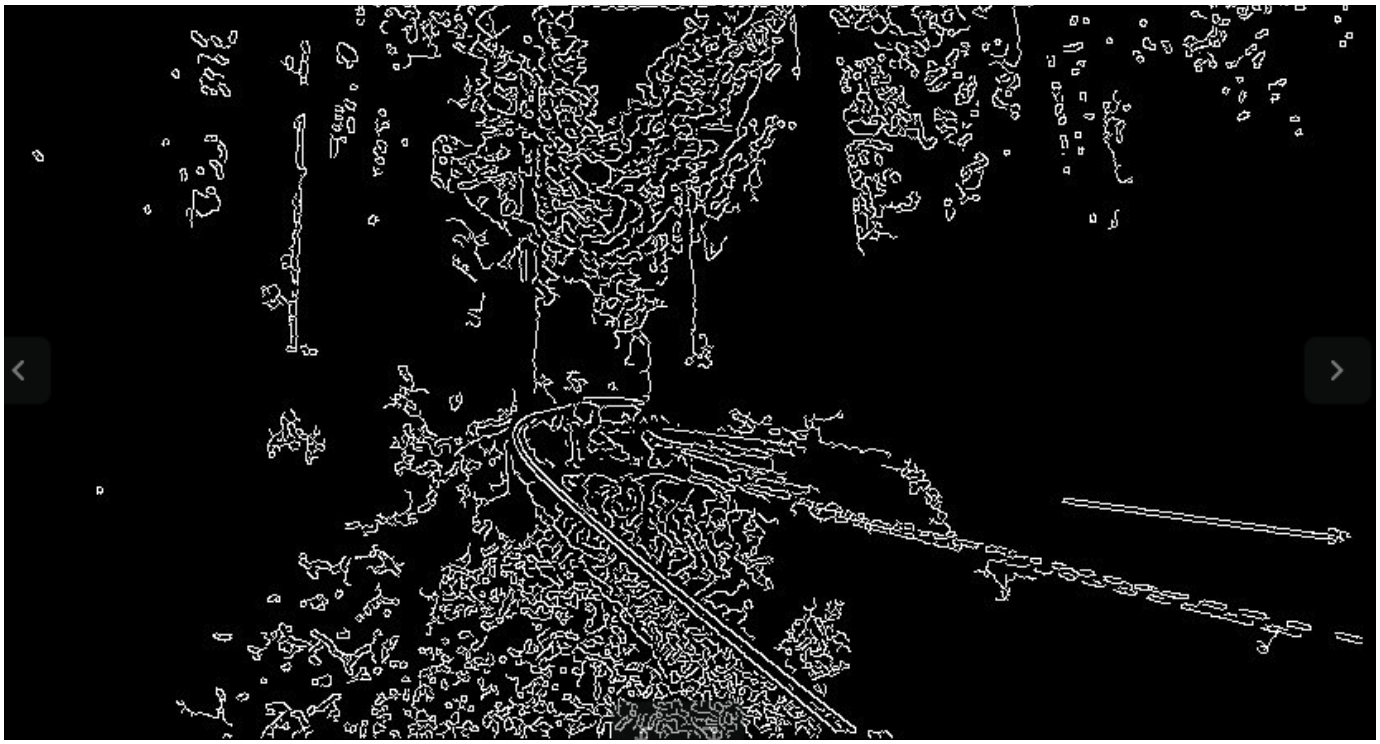
<코드>

```
// now we only use cortex-a15
//     we will use C6678 DSP next time

#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/core/ocl.hpp>
#include <unistd.h>
#include <stdio.h>
#include <time.h>

double tdiff_calc(struct timespec &tp_start, struct timespec &tp_end)
{
      return (double)(tp_end.tv_nsec - tp_start.tv_nsec) * 0.000001 +
            (double)(tp_end.tv_sec - tp_start.tv_sec) *1000.0;
}

using namespace cv;

int main(int argc, char **argv)
{
        struct timespec tp0, tp1, tp2, tp3;
        UMat img, gray; //cv::Umat c++class similar to cv::Mat
                        // enables the same APIs to be implemented usingCPU or OpenCL code.
                        // 관련링크
                //      https://stackoverflow.com/questions/33602675/what-is-the-difference-between-umat-and-mat-in-opencv or
                //
      https://software.intel.com/sites/default/files/managed/2f/19/inde_opencv_3.0_arch_guide.pdf
        imread(("sample.jpg"), 1).copyTo(img);

        clock_gettime(CLOCK_MONOTONIC, &tp0);
        cvtColor(img, gray, COLOR_BGR2GRAY);
        clock_gettime(CLOCK_MONOTONIC, &tp1);
        GaussianBlur(gray, gray, Size(5,5), 1.25);
        clock_gettime(CLOCK_MONOTONIC, &tp2);
        Canny(gray, gray, 0, 30);
        clock_gettime(CLOCK_MONOTONIC, &tp3);

        printf("RGB2GRAY tdiff = %lf ms\n", tdiff_calc(tp0,tp1));
        printf("Gauss Blur tdiff = %lf ms\n", tdiff_calc(tp1,tp2));
        printf("Canny tdiff = %lf ms\n", tdiff_calc(tp2,tp3));

        imwrite("dsp_performance.jpg", gray);


        return 0;
}
```

```
root@am57xx-evm:~/gihwahong# ./a.out
RGB2GRAY tdiff = 5.378898 ms
Gauss Blur tdiff = 4.358003 ms
Canny tdiff = 16.083666 ms
```

현재 cortex-a15 썼을때 속도.

<결과 .jpg 파일 >



<cpu 결과>

```
root@am57xx-evm:~/gihwahong# ./a.out
RGB2GRAY tdiff = 5.378898 ms
Gauss Blur tdiff = 4.358003 ms
Canny tdiff = 16.083666 ms
```

<dsp 결과>
export TI_OCL_LOAD_KERNELS_ONCHIP=Y
export TI_OCL_LOAD_KERNELS_ONCHIP=Y
export OPENCV_OPENCL_DEVICE='TI AM57:ACCELERATOR:TI Multicore C66 DSP'
이 코드는 하드웨어를 바꾸는거라 네트워크 물린 컴 하나에서 해도 다른 컴 다 적용된다.

```
root@am57xx-evm:~/gihwahong# ./a.out
RGB2GRAY tdiff = 14.120443 ms
Gauss Blur tdiff = 6.854936 ms
Canny tdiff = 6.530741 ms
root@am57xx-evm:~/gihwahong#
```

DSP Disable 하고 싶으면 아래 명령어 입력하면된다.
#export OPENCV_OPENCL_DEVICE=’disabled’

** 처음실행 캐시가안되어있어서 오래걸린다.
** RGB2GRAY 는 CPU 가 빠르다.
** canny 는 dsp 를 써야겠다.
** gaussian 은 비슷

위의예에서, task4 개나눠서 dsp2 개 cpu2 개 해서 canny,gaussian 쓰고, cpu 는 일반용+RGB2GRAY 찢어서 쓰면 굉장히 속도 빨리할수있다.

책 - 딥러닝라이브러리들 을 nvdia 이외의 환경에서 사용못한다. 즉 pc 에서는 책의 10 장이후부터는 테스트를 해볼수있으나 dsp 에서는 못 함. 그래서 딥러닝라이브러리들을 변환해서 dsp 에 쓸수있게해야한다.
[과제 : 2 일차과제(1 점분석)을 여러점찍어 평균내서 색 분석으로 바꾸기]

예)R,G,B mean value 구하고, R>200 시 RED 로 인식
R>200, G>200 YELLOW 인식
G>200 시 GREEN 인식
해서 0,1,2 값 Serial 던진다.

<코드> // R,G,B mean value 구하기 까지

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/opencv.hpp>
#include <iostream>
#include <sys/types.h>
#include <sys/poll.h>
#include <termios.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <setjmp.h>
#include <fcntl.h>
#include <stdio.h>
#include <time.h>
#include "serial.h"

using namespace std;
using namespace cv;

extern char *dev0;
jmp_buf env;
int fd;
int flag = 0;
Mat img;
Mat croi;
int idx;
char name[32] = "";
char *traffic[4] = {"red_traffic.jpg", "yellow_traffic.jpg", "green_traffic.jpg"};
Point pts[3][4] = {
    {
        Point(150, 54),
        Point(183, 53),
        Point(181, 82),
        Point(149, 81)
    },
    {
        Point(60, 95),
        Point(85, 97),
        Point(86, 125),
        Point(60, 123)
    },
    {
        Point(332, 127),
        Point(352, 130),
        Point(353, 150),
        Point(334, 150)
    }
};

const Point *ppt1[1] = { pts[0] };
const Point *ppt2[1] = { pts[1] };
const Point *ppt3[1] = { pts[2] };

void call_exit(int signo)
{
```

```
    longjmp(env, 1);
}

void traffic_chg(int signo)
{
    idx = rand() % 3;
    strcpy(name, traffic[idx]);
    printf("idx = %d, name = %s\n", idx, name);

    img = cv::Mat::zeros(img.size(), img.type());
    img = imread(name, -1);

    flag = 1;
}

Mat custom_roi(Mat img, int idx)
{
    Mat output;
    Mat mask = Mat::zeros(img.size(), img.type());
    //Mat mask(img.size(), CV_8UC3);
    //Mat mask = Mat::zeros(img.size(), img.type());

#if 0
    Point pts[3][4];
    pts[0][0] = Point(328, 175);
    pts[0][1] = Point(384, 173);
    pts[0][2] = Point(385, 111);
    pts[0][3] = Point(330, 116);

    const Point *ppt[1] = { pts[0] };
#endif
    int npt[] = { 4 };
    switch(idx)
    {
        case 0:
            cv::fillPoly(mask, ppt1, npt, 1, cv::Scalar(255, 255, 255));
            break;
        case 1:
            cv::fillPoly(mask, ppt2, npt, 1, cv::Scalar(255, 255, 255));
            break;
        case 2:
            cv::fillPoly(mask, ppt3, npt, 1, cv::Scalar(255, 255, 255));
            break;
    }

    bitwise_and(img, mask, output);

    cout << output.type() << endl;
    return output;
}

#if 0
Point(332, 127),
Point(352, 130),
Point(353, 150),
Point(334, 150)
#endif

void chk_traffic_color(Mat croi, int idx)
{
    int i, j, div_factor;
    float tmp_r = 0, tmp_g = 0, tmp_b = 0;
    char buf[32] = "";
    printf("croi rows = %d, croi cols = %d\n", croi.rows, croi.cols);

    switch(idx)
    {
        case 0:
```

```
            div_factor = 225;
            for(i = 0; i < 15; i++)
            {
                for(j = 0; j < 15; j++)
                {
                    tmp_b += croi.at<Vec3b>(61 + i, 159 + j)[0];
                    tmp_g += croi.at<Vec3b>(61 + i, 159 + j)[1];
                    tmp_r += croi.at<Vec3b>(61 + i, 159 + j)[2];
                }
            }
#if 0
            printf("r = %d, g = %d, b = %d\n",
                   croi.at<Vec3b>(68, 166)[0],
                   croi.at<Vec3b>(68, 166)[1],
                   croi.at<Vec3b>(68, 166)[2]);
#endif
                //croi.at<Vec3b>(166, 68)[0],
                //croi.at<Vec3b>(166, 68)[1],
                //croi.at<Vec3b>(166, 68)[2]);

            sprintf(buf, "%d", 1);
            printf("buf = %s\n", buf);
            //send_data(fd, buf, 1, 0);

            break;

        case 1:

        div_factor = 100;
            for(i = 0; i < 10; i++)
            {
                for(j = 0; j < 10; j++)
                {
                    tmp_b += croi.at<Vec3b>(105 + i, 68 + j)[0];
                    tmp_g += croi.at<Vec3b>(105 + i, 68 + j)[1];
                    tmp_r += croi.at<Vec3b>(105 + i, 68 + j)[2];
                }
            }
#if 0
            printf("r = %d, g = %d, b = %d\n",
                croi.at<Vec3b>(109, 76)[0],
                croi.at<Vec3b>(109, 76)[1],
                croi.at<Vec3b>(109, 76)[2]);
#endif

            sprintf(buf, "%d", 2);
            printf("buf = %s\n", buf);
            //send_data(fd, buf, 1, 0);

            break;

        case 2:
#if 1
            div_factor = 25;
            for(i = 0; i < 5; i++)
            {
                for(j = 0; j < 5; j++)
                {
                    tmp_b += croi.at<Vec3b>(135 + i, 339 + j)[0];
                    tmp_g += croi.at<Vec3b>(135 + i, 339 + j)[1];
                    tmp_r += croi.at<Vec3b>(135 + i, 339 + j)[2];
                }
            }
#endif
#if 0
            for(i = 0; i < 3; i++)
            {
                printf("b = %d, g = %d, r = %d\n",
```

```
                        croi.at<Vec3b>(135 + i, 341)[0],
                        croi.at<Vec3b>(135 + i, 341)[1],
                        croi.at<Vec3b>(135 + i, 341)[2]);
            }
#endif

            sprintf(buf, "%d", 3);
            printf("buf = %s\n", buf);
            //send_data(fd, buf, 1, 0);

            break;
    }

    tmp_r /= div_factor;
    tmp_g /= div_factor;
    tmp_b /= div_factor;

    printf("mean r = %f, mean g = %f, mean b = %f\n", tmp_r, tmp_g, tmp_b);
}

int main(int argc, char **argv)
{
    int nr, fd;
    int x, y, w, h;
    char buf[32] = "";
    char test_img[32] = "green_traffic.jpg";

    int ret;
    int wait_time;

    signal(SIGINT, call_exit);
    signal(SIGALRM, traffic_chg);

    srand(time(NULL));
    //fd = serial_config(dev0);
    fd = 1;

    printf("Automatic Traffic Light\n");
    img = imread(test_img, -1);
    //imshow("Green", img);
    //waitKey(0);
    //destroyWindow("Green");

    if(!(ret = setjmp(env)))
    {
        for(;;)
        {
            alarm(0);
            wait_time = rand() % 1 + 2;
            alarm(wait_time);

            while(!flag)
                ;

            waitKey(wait_time * 1000);

            flag = 0;

            //cvtColor(img, img, COLOR_BGR2HSV);

            croi = custom_roi(img, idx);

            //send_data(fd, buf, 1, 0);
            //printf("\n");

            // TODO - Something wrong
            chk_traffic_color(croi, idx);
```

```
            imshow("Custom ROI Image", croi);

#if 0
            red.png -> 219, 17, 29
            yellow.jpg -> 251, 246, 84
            green.jpg -> 145, 145
#endif

            //memset(buf, 0x0, 32);
        }
    }

    //close_dev(fd);

    return 0;
}
```