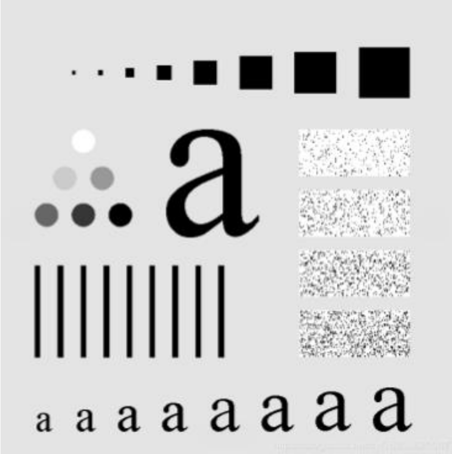
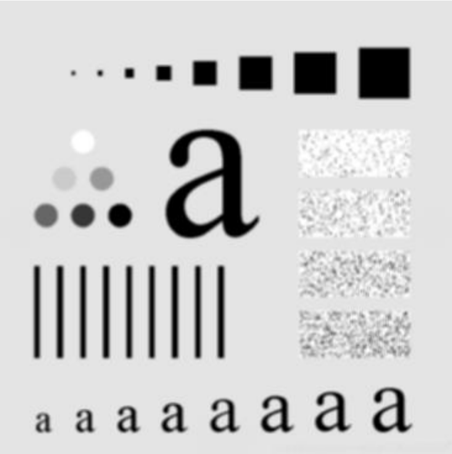
Hw-5 190410102 方尧 自动化一班

编程实现均值滤波器对图像的滤波，滤波器大小3x3，5x5，要求实现两种方式：
1.根据均值滤波定义计算；
2.利用分离滤波器的思想计算, (有兴趣的同学鼓励使用递归方法实现)、
获得滤波计算时间，对比两种滤波计算方法的效果和效率。要求用C语言实现，不能直接调用OpenCV的均值滤波函数，但可以与OpenCV的滤波结果做对比。

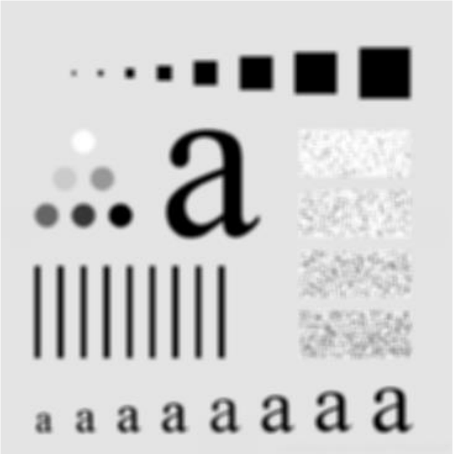
1. 两种滤波计算方法效果对比



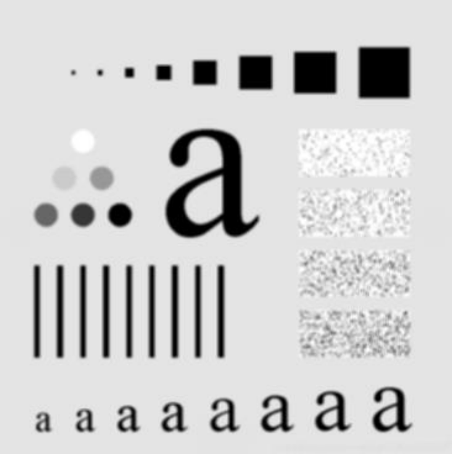
origin



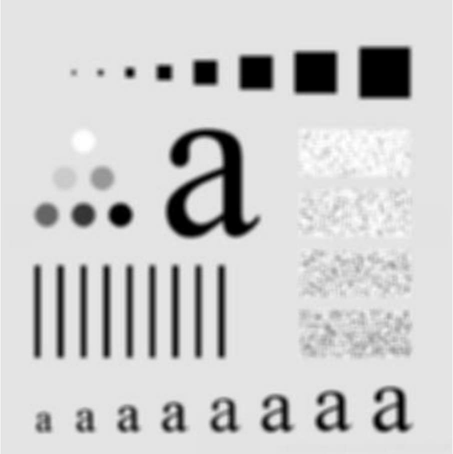
3x3_averge



5x5_averge



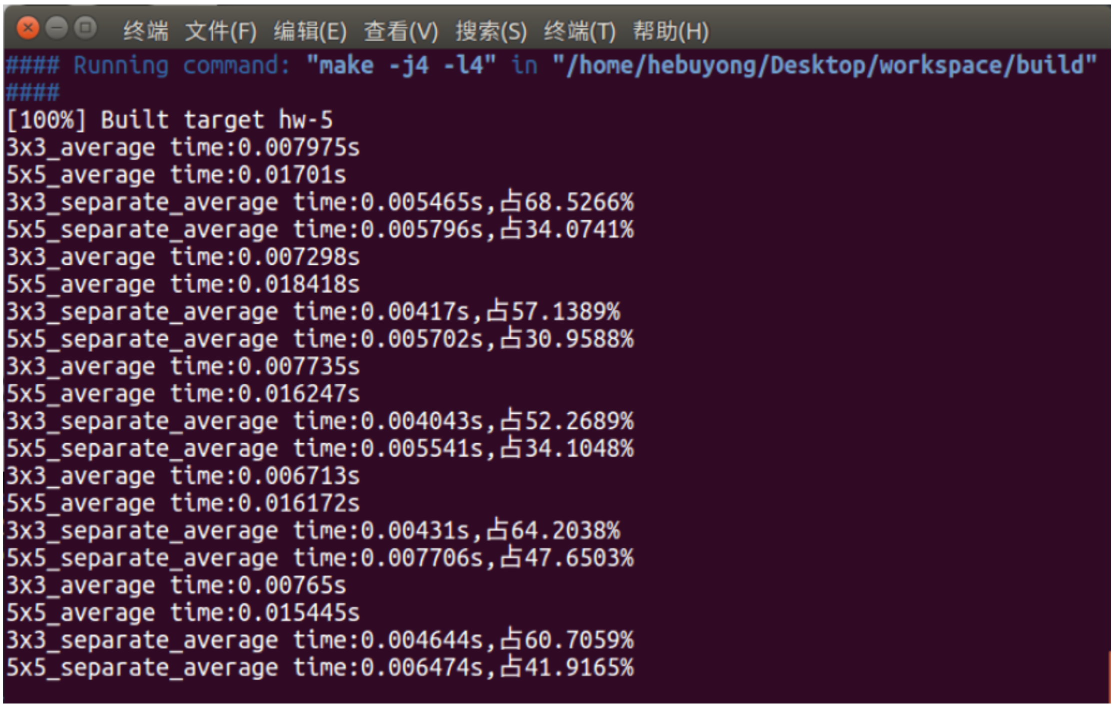
3x3_separate_average



5x5_separate_average

可见两者效果相同。

2. 两种滤波计算方法运行效率对比



```
#### Running command: "make -j4 -l4" in "/home/hebuyong/Desktop/workspace/build"
####
[100%] Built target hw-5
3x3_average time:0.007975s
5x5_average time:0.01701s
3x3_separate_average time:0.005465s,占68.5266%
5x5_separate_average time:0.005796s,占34.0741%
3x3_average time:0.007298s
5x5_average time:0.018418s
3x3_separate_average time:0.00417s,占57.1389%
5x5_separate_average time:0.005702s,占30.9588%
3x3_average time:0.007735s
5x5_average time:0.016247s
3x3_separate_average time:0.004043s,占52.2689%
5x5_separate_average time:0.005541s,占34.1048%
3x3_average time:0.006713s
5x5_average time:0.016172s
3x3_separate_average time:0.00431s,占64.2038%
5x5_separate_average time:0.007706s,占47.6503%
3x3_average time:0.00765s
5x5_average time:0.015445s
3x3_separate_average time:0.004644s,占60.7059%
5x5_separate_average time:0.006474s,占41.9165%
```

核为 3X3，分离均值滤波器用时是普通均值滤波器的 60.57%；
核为 5x5，分离均值滤波器用时是普通均值滤波器的 37.74%。
可见分离设计的滤波器效率高。

程序代码如下：

```
#include <stdlib.h>
#include<iostream>
#include <cv.h>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include "ros/ros.h"
#include <math.h>
#include<ctime>
#define M pi 3.14159265
using namespace std;
using namespace cv;
//////////////////////滤波//////////////////
// axa 均值滤波
void Average filter(Mat input, Mat output,int a){
    int x0=(a-1)/2;
    for(int i=0;i<input.rows;i++){
        for(int j=0;j<input.cols;j++){
            //图形边缘采用原图灰度值
            if(i<x0||i>=input.rows-x0||j<x0||j>=input.cols-x0)
            {
                output.at<uchar>(i,j)=input.at<uchar>(i,j);
            }else{
                double sum=0;
                for(int x=-x0;x<=x0;x++){
                    for(int y=-x0;y<=x0;y++){
                        sum=sum+input.at<uchar>(i+x,j+y);
                    }
                }
                output.at<uchar>(i,j)=sum/a/a;
            }

        }
    }

}
// axa 分离均值滤波
void Average_filter_2(Mat input, Mat output,int a){
    int x0=(a-1)/2;

    double output1[input.rows][input.cols];
    for(int i=0;i<input.rows;i++){
        for(int j=0;j<input.cols;j++){
            output1[i][j]=input.at<uchar>(i,j);
        }
    }
    double output2[input.rows][input.cols];
    memcpy(output2,output1,sizeof(output1));
    //列平均
    for(int i=0;i<input.rows;i++){
        for(int j=0;j<input.cols;j++){
            //图形边缘采用原图灰度值
            if(i<x0||i>=input.rows-x0||j<x0||j>=input.cols-x0)
            {
                output2[i][j]=output1[i][j];
            }else{
                double sum=0;
                for(int y=-x0;y<=x0;y++){
                    sum+=output1[i+y][j];
                }
                output2[i][j]=sum/a;
            }
        }
    }
    double output3[input.rows][input.cols];
    memcpy(output3,output2,sizeof(output2));
    //行平均
    for(int i=0;i<input.rows;i++){
        for(int j=0;j<input.cols;j++){
            //图形边缘采用原图灰度值
            if(i<x0||i>=input.rows-x0||j<x0||j>=input.cols-x0)
            {
                output3[i][j]=output2[i][j];
            }else{
                double sum=0;
                for(int y=-x0;y<=x0;y++){
                    sum+=output2[i][j+y];
                }
                output3[i][j]=sum/a;
            }
        }
    }
    for(int i=0;i<input.rows;i++){
        for(int j=0;j<input.cols;j++){
            output.at<uchar>(i,j)=output3[i][j];
        }
    }
```

```
}

int main(int argc, char **argv)
{

    ros::init(argc,argv,"trafficLaneTrack");//初始化 ROS 节点

    while (ros::ok())
    {
        clock_t start,end;
        double endtime,endtime2;

        //使用本地图像文件
        Mat IMG=imread("./src/hw-5/src/1.png");
        Mat gray;//灰度图
        cvtColor(IMG, gray, COLOR_BGR2GRAY);
        imshow("origin",IMG);

        Mat output = gray.clone();

        start=clock();
        Average_filter(gray,output,3);
        end=clock();
        endtime=(double)(end-start)/CLOCKS_PER_SEC;
        cout<<"3x3 average time:"<<endtime<<"s"<<endl;
        imshow("3x3 averge",output);

        start=clock();
        Average_filter(gray,output,5);
        end=clock();
        endtime2=(double)(end-start)/CLOCKS_PER_SEC;
        cout<<"5x5 average time:"<<endtime2<<"s"<<endl;
        imshow("5x5 average",output);

        start=clock();
        Average_filter_2(gray,output,3);
        end=clock();
        double per=((double)(end-start)/CLOCKS_PER_SEC)/endtime*100;
        endtime=(double)(end-start)/CLOCKS_PER_SEC;
        cout<<"3x3 separate average time:"<<endtime<<"s,占
"<<per<<"%"<<endl;
        imshow("3x3_separate_average",output);

        start=clock();
        Average_filter_2(gray,output,5);
        end=clock();
        per=((double)(end-start)/CLOCKS_PER_SEC)/endtime2*100;
        endtime2=(double)(end-start)/CLOCKS_PER_SEC;
        cout<<"5x5_separate_average_time:"<<endtime2<<"s,占
"<<per<<"%"<<endl;
        imshow("5x5_separate_average",output);
        ros::spinOnce();
        waitKey(0);
    }
    return 0;
}
```