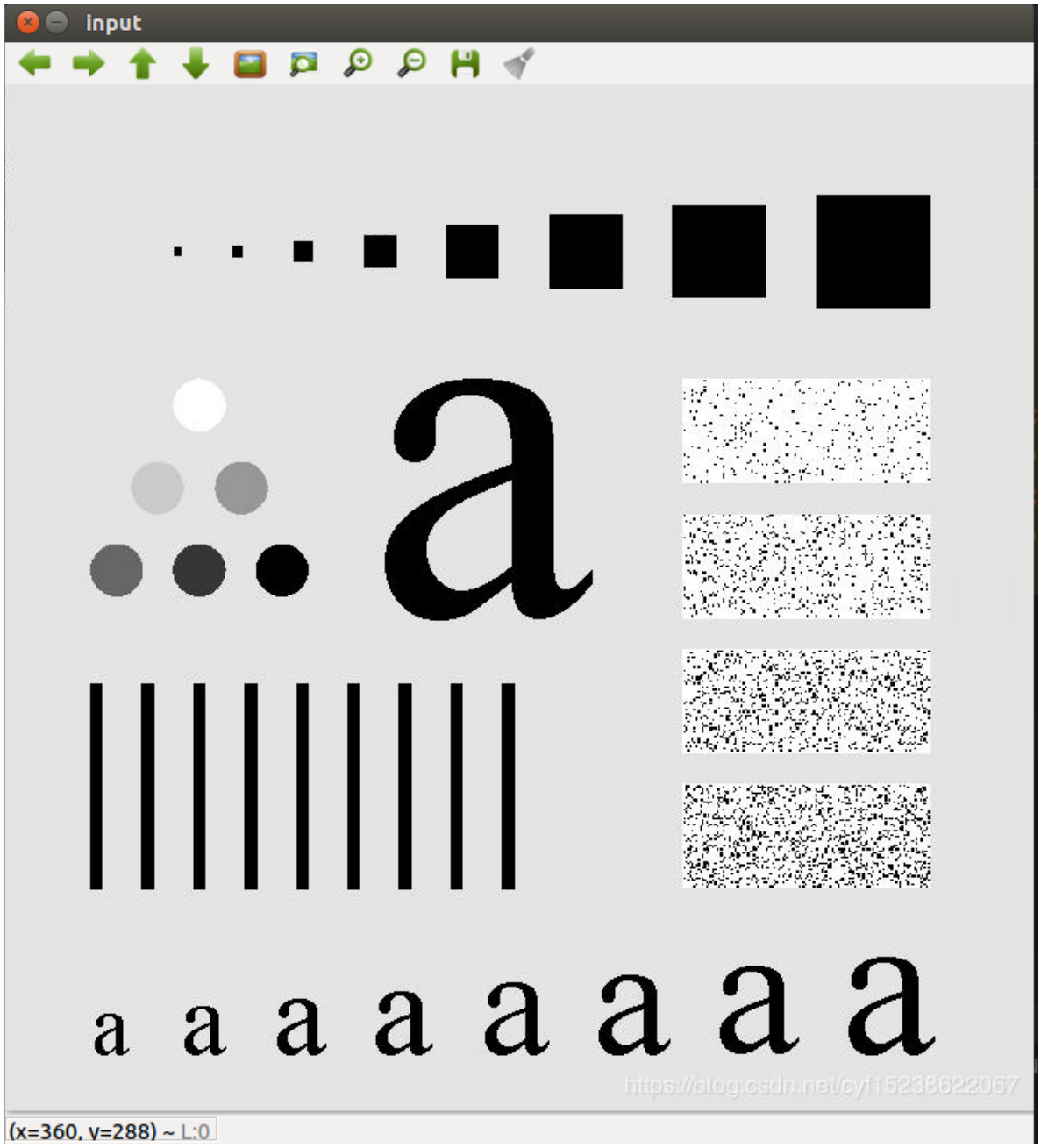
机器视觉作业6

SZ170320207

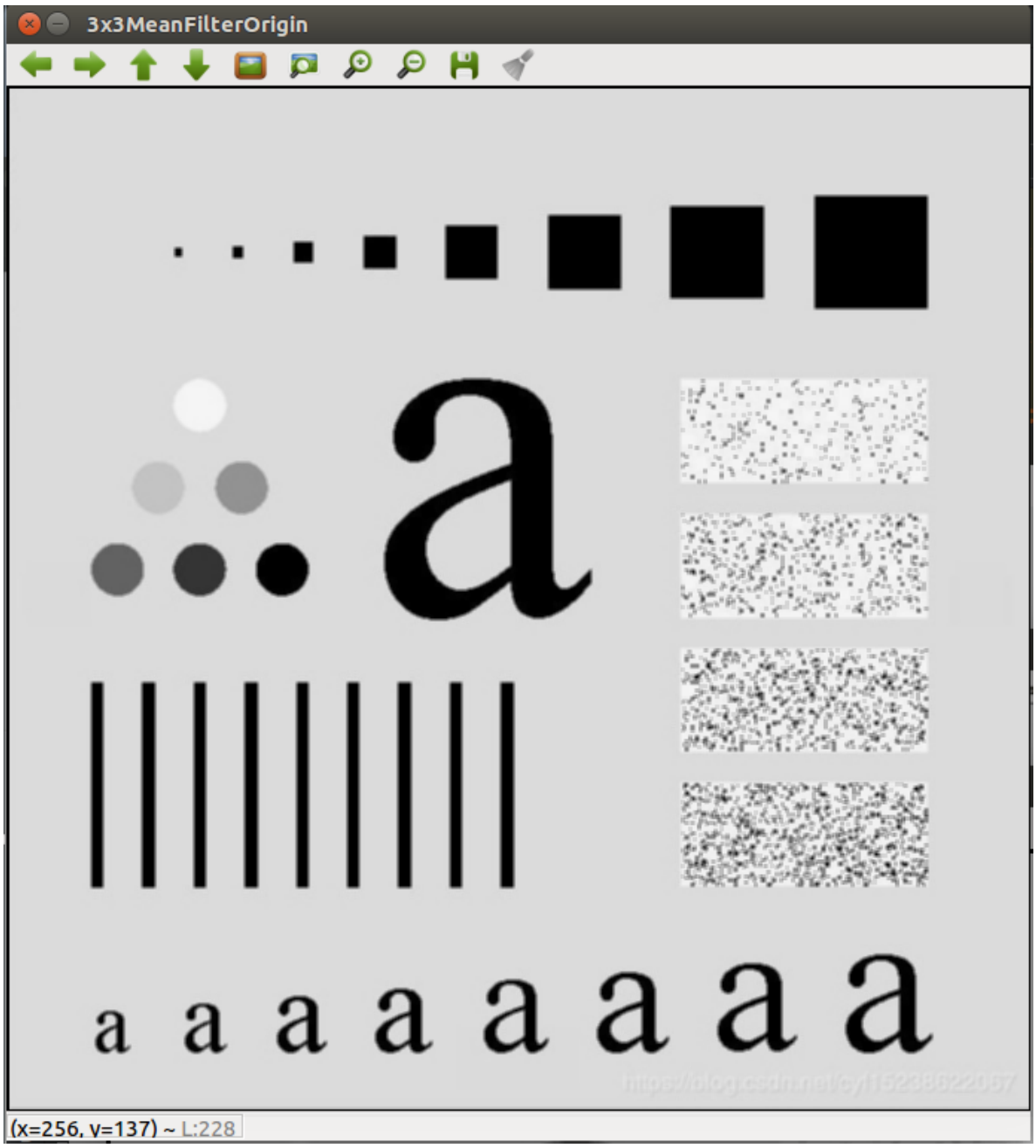
刘健恒

0.初始输入图片

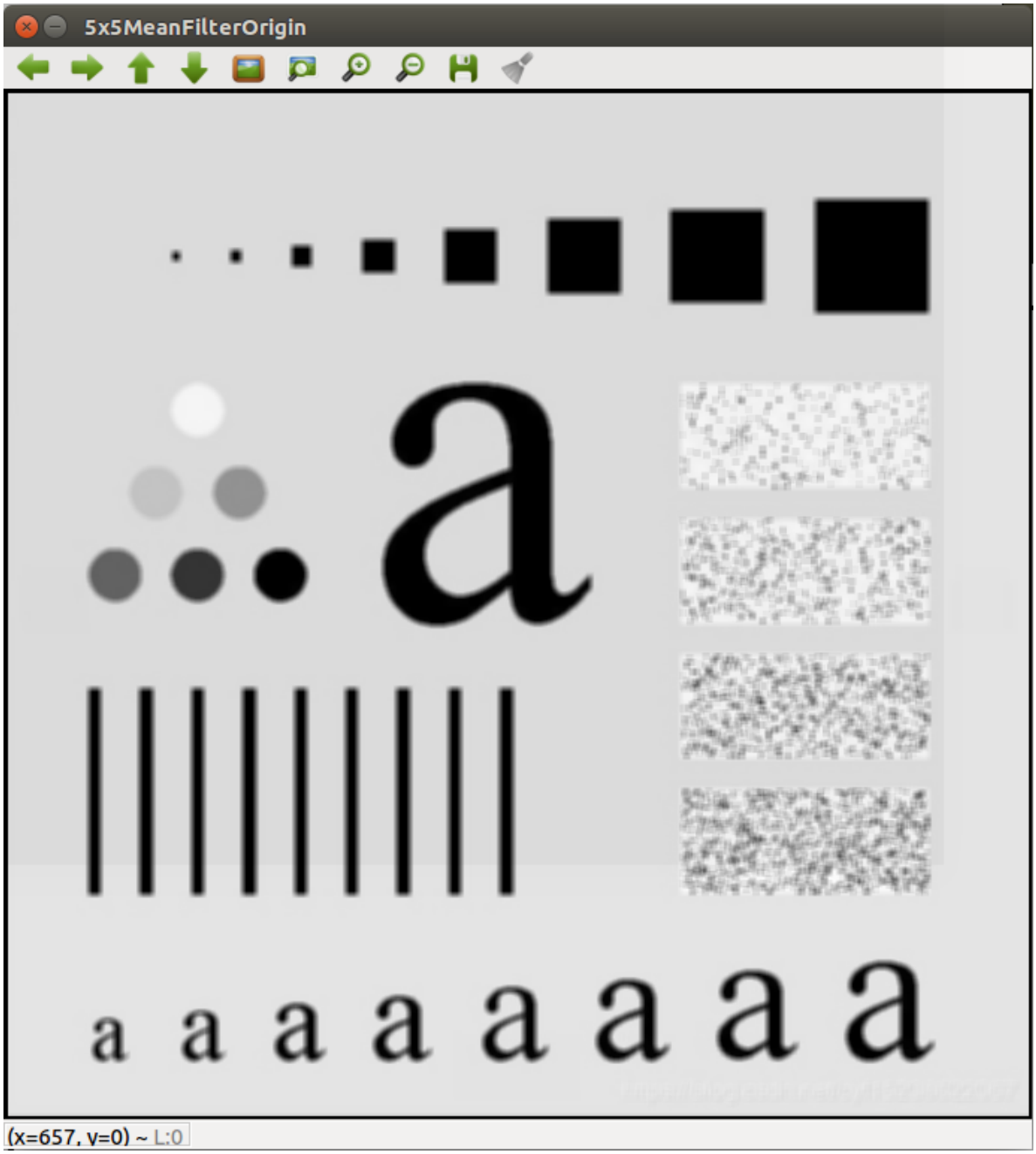


1.定义法均值滤波

3x3

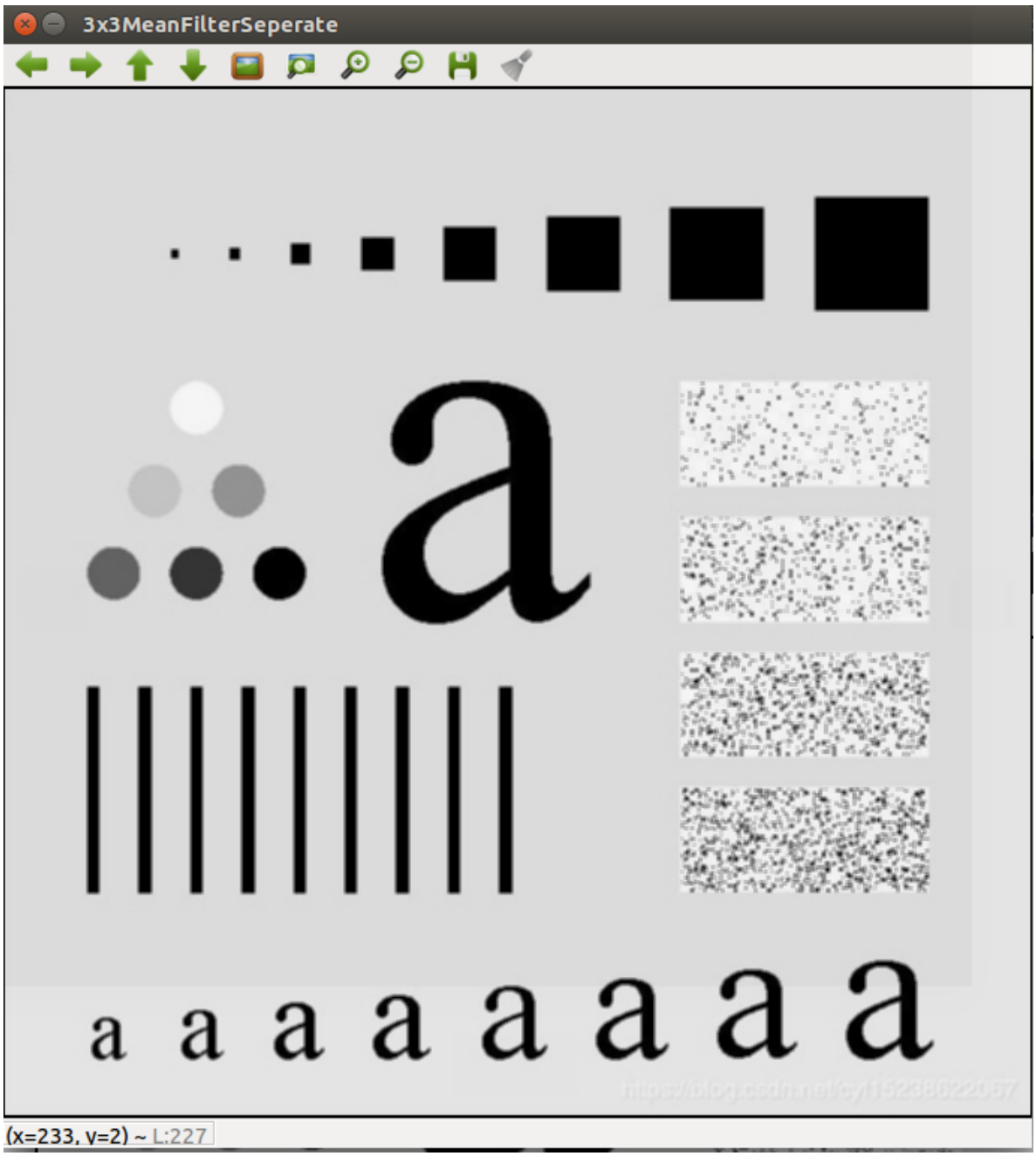


5x5



2.分离法均值滤波

3x3



5x5

3.滤波计算时间比较

```
3x3定义法均值滤波耗时：0.0276813s
5x5定义法均值滤波耗时：0.0721933s

3x3分离法均值滤波耗时：0.0185196s
5x5分离法均值滤波耗时：0.0245297s
```

可以看出通过使用分离法滤波能够大幅提高滤波的效率，并且随着模板的增大，提升性能更明显。

输出的滤波图片效果基本一致，没有发现明显的差别。

由此可见通过分配率进行分离计算滤波能够有效减少重复的运算，大幅的提高滤波的效率。

4.源码

```
//
```

```
// Created by chrisliu on 2020/4/9.
//

#include <iostream>
#include <opencv2/opencv.hpp>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>

using namespace std;
using namespace cv;

void MeanFilterOrigin(Mat input, Mat &output, int size = 3)
{
    int rows = input.rows;
    int cols = input.cols;

    output = Mat::zeros(rows, cols, CV_8UC1);

    int temp = 0;
    int deriation = (size - 1) / 2;

    sizeNumber = size * size;
    //计时
    double time_consumed = cv::getTickCount();
    for (int i = 0; i < rows; i++)
        for (int j = 0; j < cols; j++)
        {
            //边缘检测
            if ((i - deriation) > 0 && (i + deriation) < rows && (j - deriation)
> 0 && (j + deriation) < cols)
            {
                for (int x = 0; x < size; x++)
                {
                    for (int y = 0; y < size; y++)
                    {
                        temp += input.at<uchar>(i - deriation + y, j - deriation
+ x);
                    }
                }
                output.at<uchar>(i, j) = temp / sizeNumber;
                temp = 0;
            }
        }
    time_consumed = (cv::getTickCount() - time_consumed) /
cv::getTickFrequency();
    cout << size << "x" << size << "定义法均值滤波耗时：" << time_consumed << "s"
<< endl;
}

void MeanFilterSeperate(Mat input, Mat &output, int size = 3)
{
    int rows = input.rows;
    int cols = input.cols;

    int mem1_output[rows][cols];
    //将数组归0
    memset(mem1_output, 0, sizeof(mem1_output));
```

```
    int temp = 0;
    int deriation = (size - 1) / 2;

    sizeNumber = size * size;
    //计时
    double time_consumed = cv::getTickCount();
    //计算行
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            //边缘检测
            if ((j - deriation) > 0 && (j + deriation) < cols)
            {
                for (int x = 0; x < size; x++)
                {
                    temp += input.at<uchar>(i, j - deriation + x);
                }
                mem1_output[i][j] = temp;
                temp = 0;
            }
        }
    }

    int mem2_output[rows][cols];
    //将数组归0
    memset(mem2_output, 0, sizeof(mem2_output));
    //计算列
    for (int j = 0; j < cols; j++)
    {
        for (int i = 0; i < rows; i++)
        {
            //边缘检测
            if ((i - deriation) > 0 && (i + deriation) < cols)
            {
                for (int y = 0; y < size; y++)
                {
                    temp += mem1_output[i - deriation + y][j];
                }
                mem2_output[i][j] = temp / sizeNumber;
                temp = 0;
            }
        }
    }
    time_consumed = (cv::getTickCount() - time_consumed) /
cv::getTickFrequency();
    cout << size << "x" << size << "分离法均值滤波耗时：" << time_consumed << "s"
<< endl;
    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < cols; j++)
        {
            output.at<uchar>(i, j) = mem2_output[i][j];
        }
    }
}
```

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

    VideoCapture capture;
    capture.open(0);
    if (!capture.isOpened())
    {
        printf("摄像头没有正常打开,重新插拔工控机上当摄像头\n");
        return 0;
    }

    while (1)
    {
        Mat frIn =
imread("/home/chrisliu/ROS/learn_opencv/src/image_pkg/image/jiaozhun.jpg");
        cvtColor(frIn, frIn, CV_RGB2GRAY);
        imshow("input", frIn);

        Mat output(frIn.rows, frIn.cols, CV_8UC1);
        MeanFilterOrigin(frIn, output, 3);
        imshow("3x3MeanFilterOrigin", output);
        MeanFilterOrigin(frIn, output, 5);
        imshow("5x5MeanFilterOrigin", output);

        MeanFilterSeperate(frIn, output, 3);
        imshow("3x3MeanFilterSeperate", output);
        MeanFilterSeperate(frIn, output, 5);
        imshow("5x5MeanFilterSeperate", output);

        waitKey(0);
    }
    return 0;
}
```