计算机视觉第三次作业

姓名：邹佳坤

学号：21951159

指导教师：刘二腾

language：C++

前处理

通过双边滤波去除高斯噪声。

使用opencv实现：

```
cv:: bilateralFilter(img, out2, 25, 25 * 2, 25 / 2);
```

自适应维纳滤波

通过自适应维纳滤波处理运动模糊。

$$\mu = \frac{1}{NM} \sum_{n_1,n_2 \in \eta} src(n_1, n_2)$$

$$\sigma^2 = \frac{1}{NM} \sum_{n_1,n_2 \in \eta} src^2(n_1, n_2) - \mu^2$$

$$dst(n_1, n_2) = \mu + \frac{\max(0, \sigma^2 - \nu^2)}{\max(\sigma^2, \nu^2)}(src(n_1, n_2) - \mu)$$

维纳滤波实现代码如下（opencv）：

```
double WienerFilterImpl(const cv::Mat &src, cv::Mat &dst, double noiseVariance,
const cv::Size &block) {

    assert(("Invalid block dimensions", block.width % 2 == 1 && block.height % 2
== 1 && block.width > 1 &&
                                        block.height > 1));
    assert(("src and dst must be one channel grayscale images", src.channels() ==
1, dst.channels() == 1));

    int h = src.rows;
    int w = src.cols;

    dst = cv::Mat1b(h, w);

    cv::Mat1d means, sqrMeans, variances;
    cv::Mat1d avgVarianceMat;
```

```
    cv::boxFilter(src, means, CV_64F, block, cv::Point(-1, -1), true,
cv::BORDER_REPLICATE);
    cv::sqrBoxFilter(src, sqrMeans, CV_64F, block, cv::Point(-1, -1), true,
cv::BORDER_REPLICATE);

    cv::Mat1d means2 = means.mul(means);
    variances = sqrMeans - (means.mul(means));

    if (noiseVariance < 0) {
        cv::reduce(variances, avgVarianceMat, 1, CV_REDUCE_SUM, -1);
        cv::reduce(avgVarianceMat, avgVarianceMat, 0, CV_REDUCE_SUM, -1);
        noiseVariance = avgVarianceMat(0, 0) / (h * w);
    }

    for (int r = 0; r < h; ++r) {
        uchar const *const srcRow = src.ptr<uchar>(r);
        uchar *const dstRow = dst.ptr<uchar>(r);
        double *const varRow = variances.ptr<double>(r);
        double *const meanRow = means.ptr<double>(r);
        for (int c = 0; c < w; ++c) {
            dstRow[c] = cv::saturate_cast<uchar>(
                    meanRow[c] +
                    std::max(0., varRow[c] - noiseVariance) / std::max(varRow[c],
noiseVariance) *
                    (srcRow[c] - meanRow[c])
            );
        }
    }

    return noiseVariance;
}

void WienerFilter(const cv::Mat &src, cv::Mat &dst, double noiseVariance = 10000,
const cv::Size &block = cv::Size(3, 3)) {
    WienerFilterImpl(src, dst, noiseVariance, block);
    return ;
}
```

效果展示

处理前：

处理后：