[一、 实验题目 3](#_Toc9722)

[二、 实验目的 3](#_Toc32015)

[三、 实验设计与分析 3](#_Toc25779)

[1. io部分 3](#_Toc27149)

[2. Merge和Sort 3](#_Toc23650)

[4. Bisearch 和 check 4](#_Toc6311)

[四、 实验环境 5](#_Toc6844)

[五、 项目测试（功能与性能） 5](#_Toc24004)

# 实验题目 Shading计算

# 实验目的

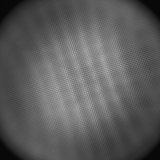
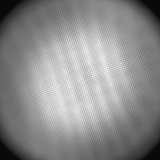
**问题描述**

Shading即我们常说的暗角，为图像呈现出中心区域较亮，四周偏暗的现象（如下图）。

Shading的成因有以下原因引起：

1. 由镜头（Lens）的光学特性引起。对于整个镜头，可将其视为一个凸透镜。由于凸透镜中心的聚光能力远大于其边缘，从而导致Sensor中心的光线强度大于四周。此种现象也称之为边缘光照度衰减。

2. 由摄像头本身的机械结构导致产生。由于摄像头各模块在制作和组装的过程中，均存在一定的工艺误差，从而影响物体光线在摄像头内的传播。



Shading = (四角暗处的灰度) / (中心亮处的灰度)

**输入描述**

Test数据中所有校准得到的基准图；

数据路径：[\\192.168.1.45\file-input\常用软件+培训资料\【培训文档】\光学](\\\\192.168.1.45\\file-input\\常用软件+培训资料\\【培训文档】\\光学) silead-fpdata-caldata-0402压缩包

**输出描述**

.txt 输出n张校准图像对应的shading值；

# 三、实验设计与分析

## 1.文件遍历与图片读取

文件采用\_finddata\_t结构体实现深度优先遍历和.bmp图片读取。

伪代码如下：

DfsProcess(folderPath,imageBuf);

Dowhile(有尚未遍历的文件节点时){

若有子目录

更新路径并迭代DFS遍历DfsProcess(dfsPath, dfsBuf);

否则为根节点

则查找第一个.bmp文件

没找到则return

Dowhile（是.bmp文件）{

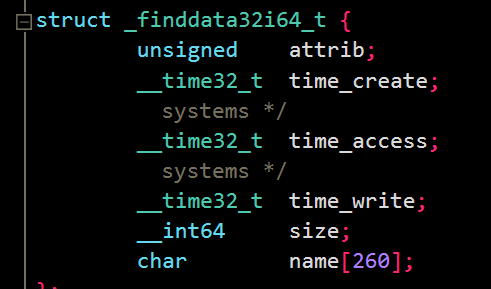
GFPLoadImage;

shading = GetShading(imageBuf, height, width);

return 0;

}

关键还是finddata\_t的文件信息，以及findfirst，findnext和findclose的基于文件句柄的可操作性。



相比之下，深度优先遍历倒是老生常谈的基础，这里就不展开了。

## 2.shading计算

Shading的计算也比较简单。

for (i = 0; i < height; ++i)

for (j = 0; j < width; ++j)

if (i > heightPar \* 3 && i < heightPar \* 4 && j > widthPar \* 3 && j < widthPar \* 4)//center

mean[0] += imageBuf[j + i \* width];

if (i > heightPar \* 1 && i < heightPar\* 2 && j > widthPar \* 1 && j < widthPar\* 2)//leftup

mean[1] += imageBuf[j + i \* width];

if (i > heightPar \* 1 && i < heightPar \* 2 && j > widthPar \* 5 && j < widthPar \* 6)//rightup

mean[2] += imageBuf[j + i \* width];

if (i > heightPar \* 5 && i < heightPar \* 6 && j > widthPar \* 1 && j < widthPar \* 2)//leftdown

mean[3] += imageBuf[j + i \* width];

if (i > heightPar \* 5 && i < heightPar \* 6 && j > widthPar \* 5 && j < widthPar \* 6)//rightdown

mean[4] += imageBuf[j + i \* width];

前面只需要遍历仅仅一次image，就可以计算出中心和四角的灰阶均值，并借此计算出shading：

mean[i] /= (widthPar \* heightPar);

shading = (mean[1] + mean[2] + mean[3] + mean[4]) \* 100 / (mean[0] \* 4);;

# 四、实验环境

Microsoft Visual Studio Ultimate 2013

2020-4-30 9:50:15

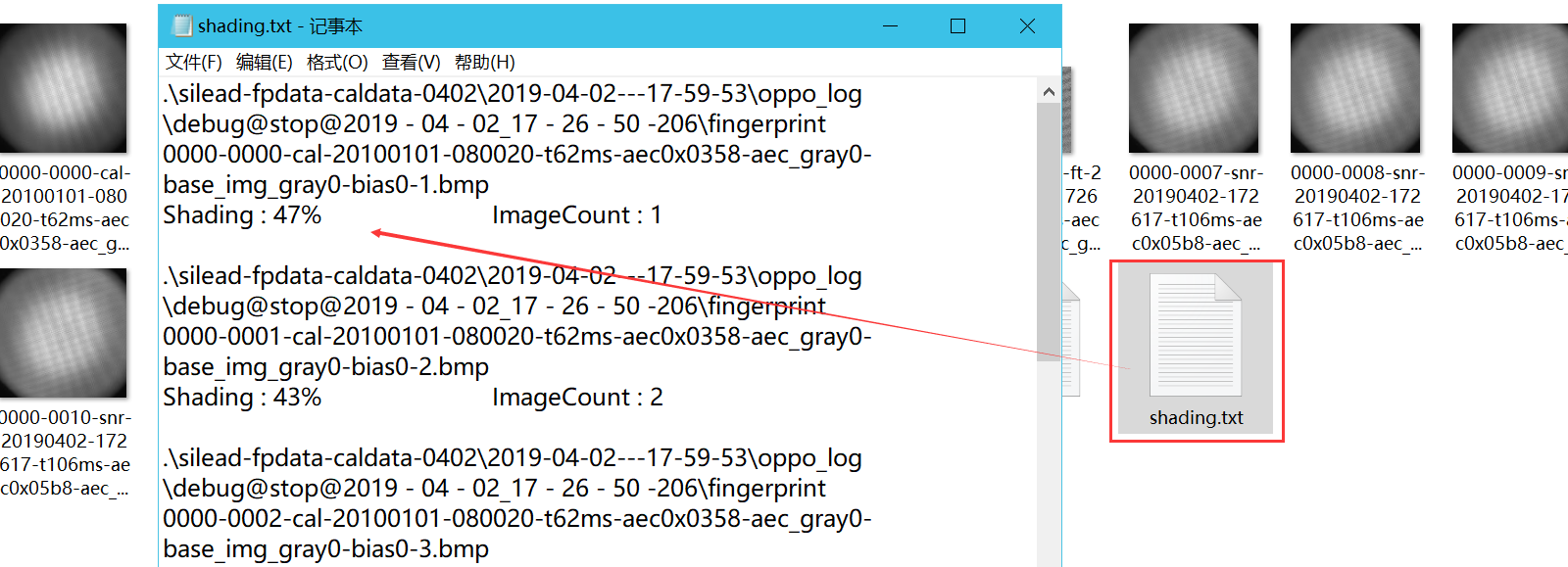
Windows 10.0 1903

# 五、实验测试

手机屏幕的截图

描述已自动生成

如图，控制台输出文件信息和计算的Shading值。



每个图片文件夹存储为txt文档保存shading值

# 六、实验分析与评估

这次实验题主要花费时间在文件的操作上，学会了文件句柄的操作方法和finddata对于文件信息的存储和使用。

另外在shading的计算上也体会到了shading的实际应用意义，以及部分文件比如ft图像就显得shading较黑。

此外，在文件的存储上手有点生，和之前java用的库有点不同，最后存文件如果多次存储会在原有信息的基础上append而不是trunc，这个是可以加以改进的。

此外外，最后一天临时补了点git相关，把项目push到了我的[github](https://github.com/pqsworld/optics)上面。