图像处理课程

设计报告

基于 mfc 的细胞识别程序

专　　业 电子信息工程

班　　级 电 183 班

学　　号 189064238

姓　　名 刘航宇

指导教师 杜培明

安徽工业大学电气与信息工程学院

2021 年 07 月 12 号

目录

一、图像处理课程设计目标.....3
1.1 课题与技术指标.....3
1.2 实践情况.....3
1.3 任务点.....3
二、设计思路.....3
三、创建工程及算法分析.....4
3.1 预备工作及工程创建.....4
3.2 打开 BMP 图片与显示 HIS 值.....5
3.3 Mark-细胞标记.....7
3.4 OnTwoVaule()-二值化.....8
3.5 OnFillHoles()-填洞.....10
3.6 shrink()-收缩.....11
3.7 findcenter()-中心点获取.....12
3.8 count()-统计.....14
3.9 OnCellprgAll( )-按序执行全部.....16
四、错误情况及分析.....16
4.1 未声明全局变量信息头.....16
4.2 错把指针当作 int 型.....16
4.3 软件菜单栏突然丢失.....17
4.4 矩形框在鼠标未按下就出现了.....17
五、心得体会.....17
六、致谢.....19
七、参考文献.....19

一、图像处理课程设计目标

1.1 课题与技术指标

课题：利用图像处理技术设计细胞识别程序。
技术指标：
1. 实验 VS2019 MFC 开发平台
2. 待识别图像为 24bit 的真彩色细胞图像进行处理
3. 要求识别出细胞，并且保证准确度情况下，统计出细胞的个数和大小。

1.2 实践情况

1.课堂教学：布置课程设计任务，相关知识介绍等环节。

2.上机实验：根据课程设计任务要求，设计步骤；根据数字图像处理基本知识逐步完成设计步骤。

1.3 任务点

1.完成图像标注：添加 cidb.h,cdib.cpp、添加函数、添加消息响应函数、添加菜单、打开图像。

2.完成细胞的判别：持续判断 Maybe 点邻域有无 Mak 点、Sobel 计算边缘、删除孤立边缘、生成黑白图像。

3.细胞收缩：利用 stack、vector 获取孔洞坐标、填充孔洞、Edge 处置 0、4 方向、8 方向交替生成边缘。

4.细胞中心点信息获取：递归算法、判别局部是否全是边缘、储存全部中心位置。

5.错误信息的删除：计算中心点均值，半径、去除半径过小的点、包含圆的剔除、相交过大圆的剔除、信息统计与显示

二、设计思路

图像处理之细胞识别的主要工作就是识别细胞数目以及选择出指定大小细胞。设计思路如下图 1 所示。同时我们要去除不符合要求的中心点，来确定最终的细胞数目，去除太小和太大的细胞[1-2]，并结合实际来修改各种参数，设计合理。

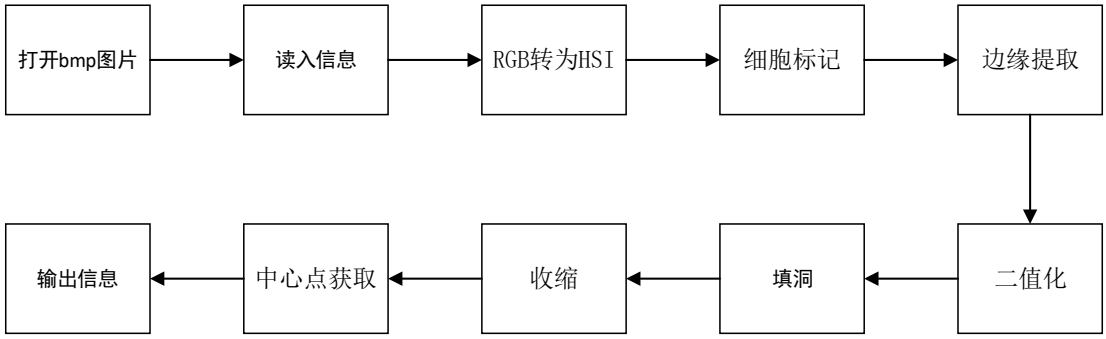

图 1 系统设计思路

三、创建工程及算法分析

3.1 预备工作及工程创建

（1）VS2019 安装 MFC 与 C++开发环境
（2）创建 MFC 工程如下图所示

图 2 创建工程

图 3 MFC 配置

3.2 打开 BMP 图片与显示 HIS 值

代码分析:

（1）在 onDraw 函数实现显示图片；在 messages 中添加 dispaly()实现打开一幅图像的功能。在 LoadBmp()函数中利用 fopen 实现自动打开图片，利用 fread 读取图片信息。

（2）通过：查看—>建立类向导—>添加 OnMouseMove()函数，添加代码实现获取所要信息;实现鼠标移动显示 HSI。

（3）通过函数 RgbtoHsi 实现 RGB 向 HSI 的转化。说明 RGB 向 HSI 模型的转换是由一个基于笛卡尔直角坐标系的单位立方体向基于圆柱极坐标的双锥体的转换。基本要求是将 RGB 中的亮度因素分离，将色度分解为色调和饱和度，并用角向量表示色调[3]。如果直接对 R、G、B 处理，其处理过程中很可能会引起三个量不同程度的变化，这样就会产生色差问题，甚至带来颜色上的失真。HSI 模型的出现，使得在保持色彩无失真的情况下实现图像处理成为可能。

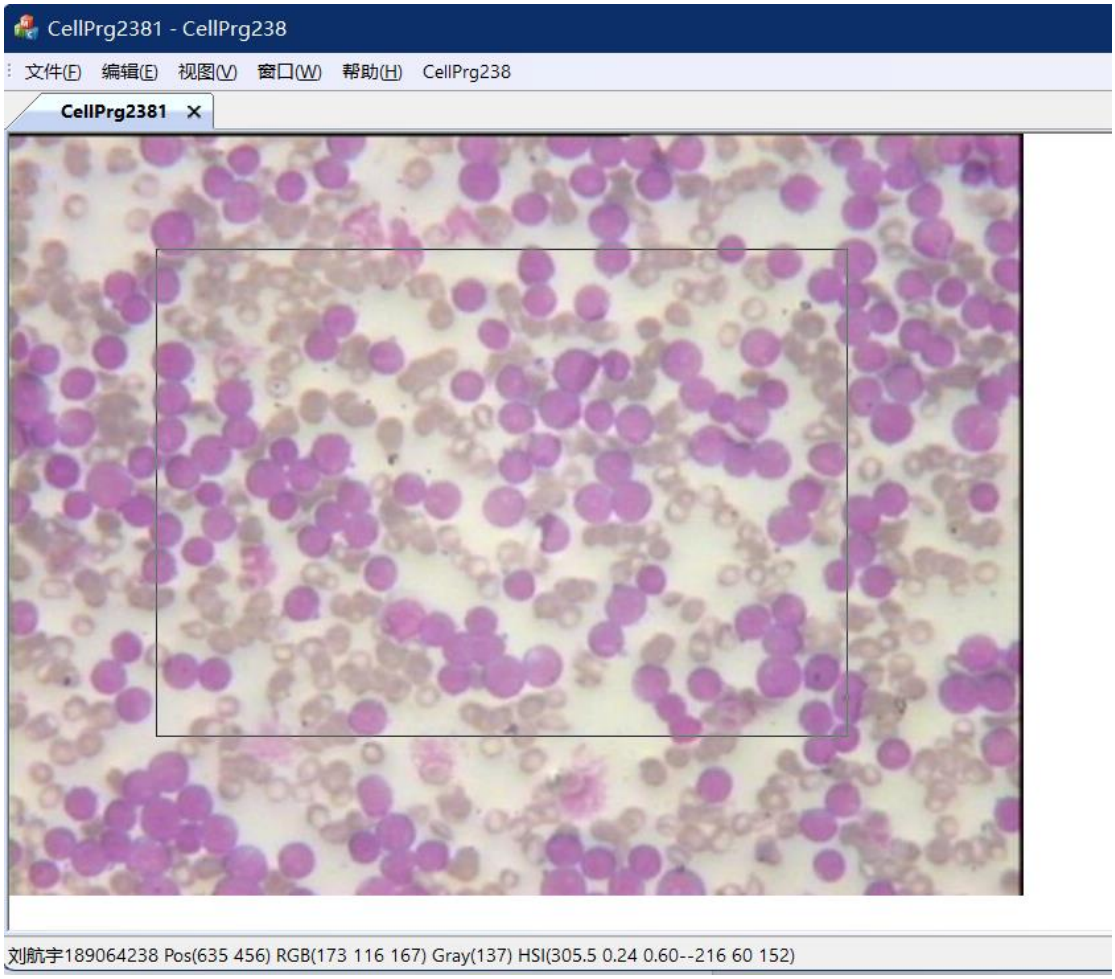

图 4 HSI 显示

主要代码：

```
void CCellPrg238View::RgbtoHsi(RGB* pRgb, HSI* pHsi)
{
    // TODO: 在此处添加实现代码.
    const double ZERO_SATURATION = 0.0;
    const double UNDEFINED_HUE = 0.000;
    const double DEGREES_PER_RADIAN = 180.0 / 3.14159265358979;

    double  R, G, B, Sum, Quotient;
    double  Radians, Angle, MinValue, MaxValue, TempDouble1, TempDouble2;
    R = ((double)pRgb->r) / 255.0;
    G = ((double)pRgb->g) / 255.0;
    B = ((double)pRgb->b) / 255.0;
    Sum = R + G + B;
    pHsi->Intensity = Sum / 3.0;
    MinValue = (R < G) ? R : G;
    MinValue = (B < MinValue) ? B : MinValue;
    MaxValue = (R > G) ? R : G;
    MaxValue = (B > MaxValue) ? B : MaxValue;
    if (pHsi->Intensity < 0.00001)
        pHsi->Saturation = ZERO_SATURATION;
    else
        pHsi->Saturation = 1.0 - (3.0 * MinValue) / Sum;
    if (MinValue == MaxValue)
    {
        pHsi->Hue = UNDEFINED_HUE;
        pHsi->Saturation = ZERO_SATURATION;
        return;
    }

    TempDouble1 = (((R - G) + (R - B)) / 2.0);
    TempDouble2 = (R - G) * (R - G) + (R - B) * (G - B);
    Quotient = (TempDouble1 / sqrt(TempDouble2));
    Radians = acos(Quotient);
    Angle = Radians * DEGREES_PER_RADIAN;
    pHsi->Hue = (B > G) ? 360.0 - Angle : Angle;

    return;
```

图 5 HSI 主要代码

3.3 Mark-细胞标记

代码分析：

确定 Mark 与 mbMark 是通过半径来的，通过计算图像上的某个像素点与细胞内部像素点的欧几里得距离与归一化门限值分别为 0.09 和 0.15 比较，即到圆心（meanH, meanS）r<0.09 是 Mark 红色(细胞)，到圆心 0.09<r<0.15 内是 mbMark 蓝色。OnMouseMove 函数里调用 RgbtoHsi(&rgb, &Hsi)函数，可以在屏幕上显示鼠标所指点的坐标以及 RGB、HSI 和灰度值，通过 HSI 的可以选取合适的阈值来找到细胞以及边界[4]。

边缘提取的方法：先滤波，去除噪声的影响，防止细胞内的噪声的影响。之后用 sobel 算子求出边界，之后画一个 5*5 的矩形，如果没有遇到边界，是噪声，去除。边界点是绿色。细胞边界分别用红色、暗红、蓝色和绿色标记出来。

主要代码：

```
            double x1 = Hsi.Hue;//0—360
            double x2 = meanH;//近似
            if (x1 < 90) x1 += 360;
            //归一化
            double y1 = Hsi.Saturation;
            double y2 = meanS;
            x1 /= 180; x2 /= 180;//0--1
            //y1*=2;y2*=2;//0--2
            double dis = DISTANCE(x1, y1, x2, y2);//欧氏距离
            if (dis < MarkDoor) {//Mark
                *lpSrc = 0; *(lpSrc + 1) = 0; *(lpSrc + 2) = 255;//Red
            }
            else if (dis < MayBeMarkDoor) {//may be Mark
                *lpSrc = 255; *(lpSrc + 1) = 0; *(lpSrc + 2) = 0;//Blue
            }
            //为了避开后面判断
            else {//not Mark/maybe Mark
                if (*lpSrc == 0) *lpSrc = 1;//Mark判断；红
                else if (*lpSrc == 255) *lpSrc = 254;//maybe mark判断；蓝
                if (*(lpSrc + 1) == 255) *(lpSrc + 1) = 254; //edge判断

            }
        }
    }

    Invalidate(true);
    MessageBox(TEXT("Mark(Red) & maybe Mark(Blue) "));
    bool MarkChg = true;
    while (MarkChg) {
        MarkChg = false;
```

```
bdelete = true;
for (int m = -M; m <= M; m++)
    for (int n = -M; n <= M; n++)
    {
    if (m == -M || m == M || n == -M || n == M) {
        if (*(lpDst + m_nLineByte * m + n * 3) || (*(lpDst + m_nLineByte * m + n * 3 + 1) == 255))//noMark && no Edge
        {
            bdelete = false;
            m = M + 1; n = M + 1;//out
        }
    }
    }
```

图 6 Mark 主要代码

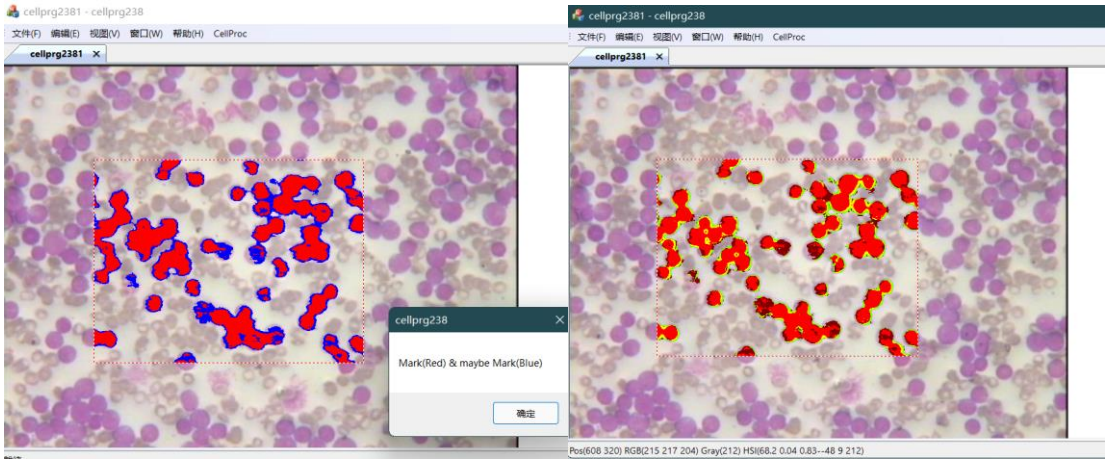

图 7 标记后情况

3.4 OnTwoVaule()-二值化

代码分析:

背景灰度值为 128，细胞灰度值为 240，边缘为 255。二值化便于轮廓提取。

均值滤波的基本原理是用均值代替原图像中的各个像素值，即对待处理的当前像素点（x，y），选择一个模板，该模板由其近邻的若干像素组成，求模板中所有像素的均值，再把该均值赋予当前像素点（x，y），作为处理后图像在该点上的灰度个 g（x，y），即个 g（x，y）=1/m ΣƒΣf（x，y）；m 为该模板中包含当前像素在内的像素总个数。

1. 0x7X　　//edge 边界含义（0x7x 表示边界没 Mark，存在为四边）
2. 0x8X　　//Mark--not edge是细胞
3. 0xfX　　//Mark --edge
4. 0xX1　　//visited
5. 0x2　　//CENTERED

8. m_vCenterPoints.at(j) //存储最后中心点的地方

主要代码：

```
for (int i = 0; i < m_nHeight; i++) {
    for (int j = 0; j < m_nWidth; j++) {
        lpSrc = m_pImage + m_nLineByte * (m_nHeight - 1 - i) + j * 3;//*3彩色
        lpDst = lpNewDIBBits + lNewLineBytes * (m_nHeight - 1 - i) + j;//灰度
        unsigned int v;
        v = 0;
        if (*(lpSrc) == 0)//Mark
        {
            v = TWOVALUE_H;
            if (*(lpSrc + 1))
                v |= EDGEPOINT;//set edge
        }
        *lpDst = (unsigned char)v;
    }

}
```

图 8 二值化主要代码

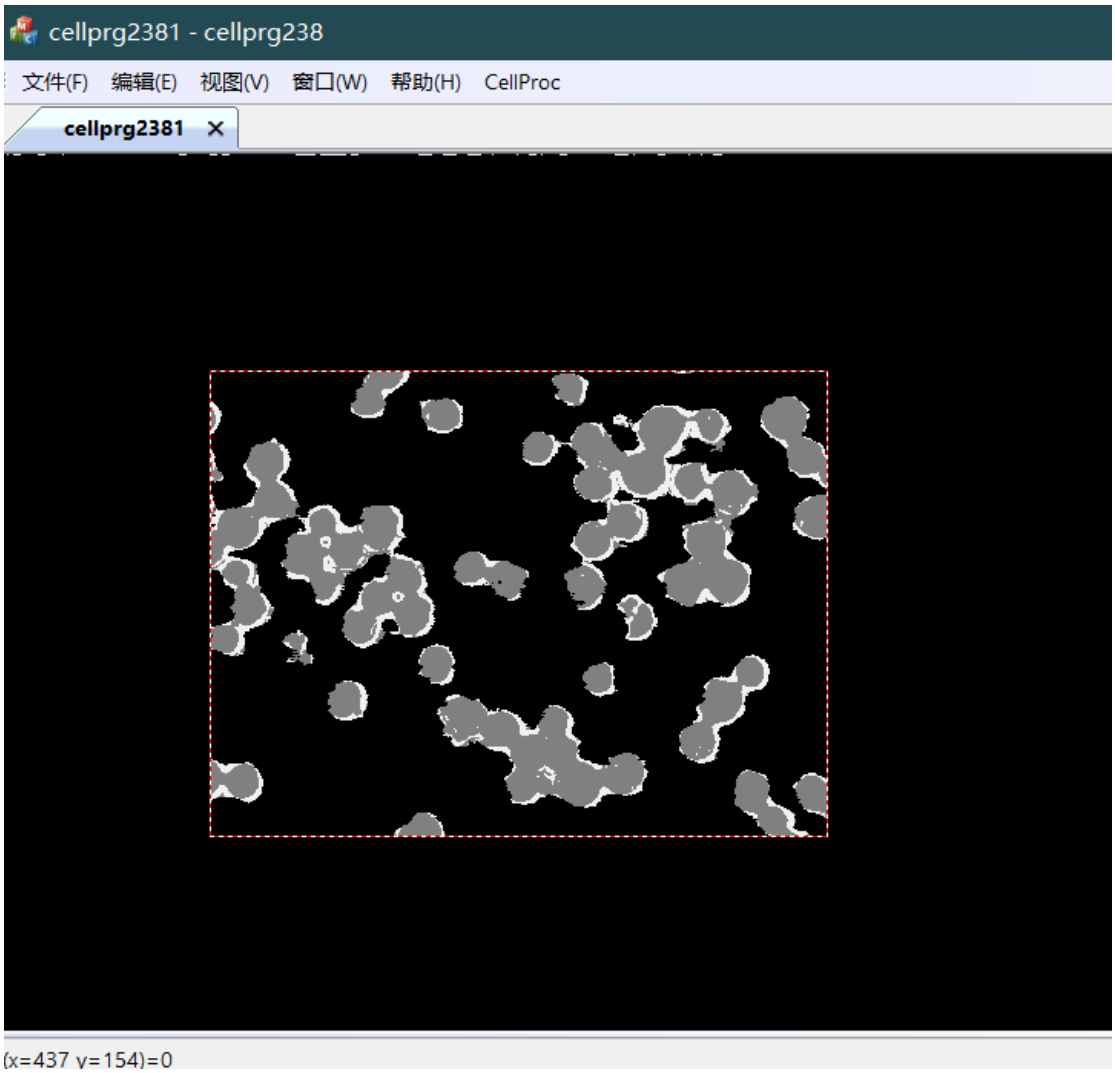

图 9 二值化现象

3.5 OnFillHoles()-填洞

代码分析：

在阈值处理时，如果像素在阈值范围内，则像素将被标志。孔洞填充将先统计所有连通的非标志区域面积，总会有一个或者几个面积特别大的区域，其它的都是面积相对较小的区域。较小或者很小的往往就是系统所要填充的孔洞了。填细胞中的洞，（即为 0 的值），填完后，处理非 Mark 的，边界的值（目的是打开边缘），设为 0.填洞方法：用栈，向量处理，（从四个方向）没访问过的黑点进栈，直到边和走完。之后读值到 xt,yt 并去值，v 放的是点的位置。没有碰到边界和小于 100，则填洞，即令洞的各点值为 Mark 值。将细胞中灰度值为 128 的部分的灰度值设置为 240[5]。

```
if (v.size()<MAX_HOLE && !bBorder)//没有碰到边缘
{
    //CString msg;
    //msg.Format("\n%d--(%d %d)", v.size(), wd, ht);
    //for see
    //if (v.size()>50){
    //Invalidate(true);
    //MessageBox(msg);//显示大于50小于100的，每次要回车，即对话框显示
    //}
    //else
    //TRACE(msg);//打印屏幕上

    for (UINT k = 0; k<v.size(); k++){
        xt = v[k].x;
        yt = v[k].y;
        lpSrc = m_pImage + m_nLineByte * (m_nHeight - 1 - yt) + xt;
        *lpSrc |= MARKED;//0x00填成0x80
    }
}
```

图 10 填洞主要代码

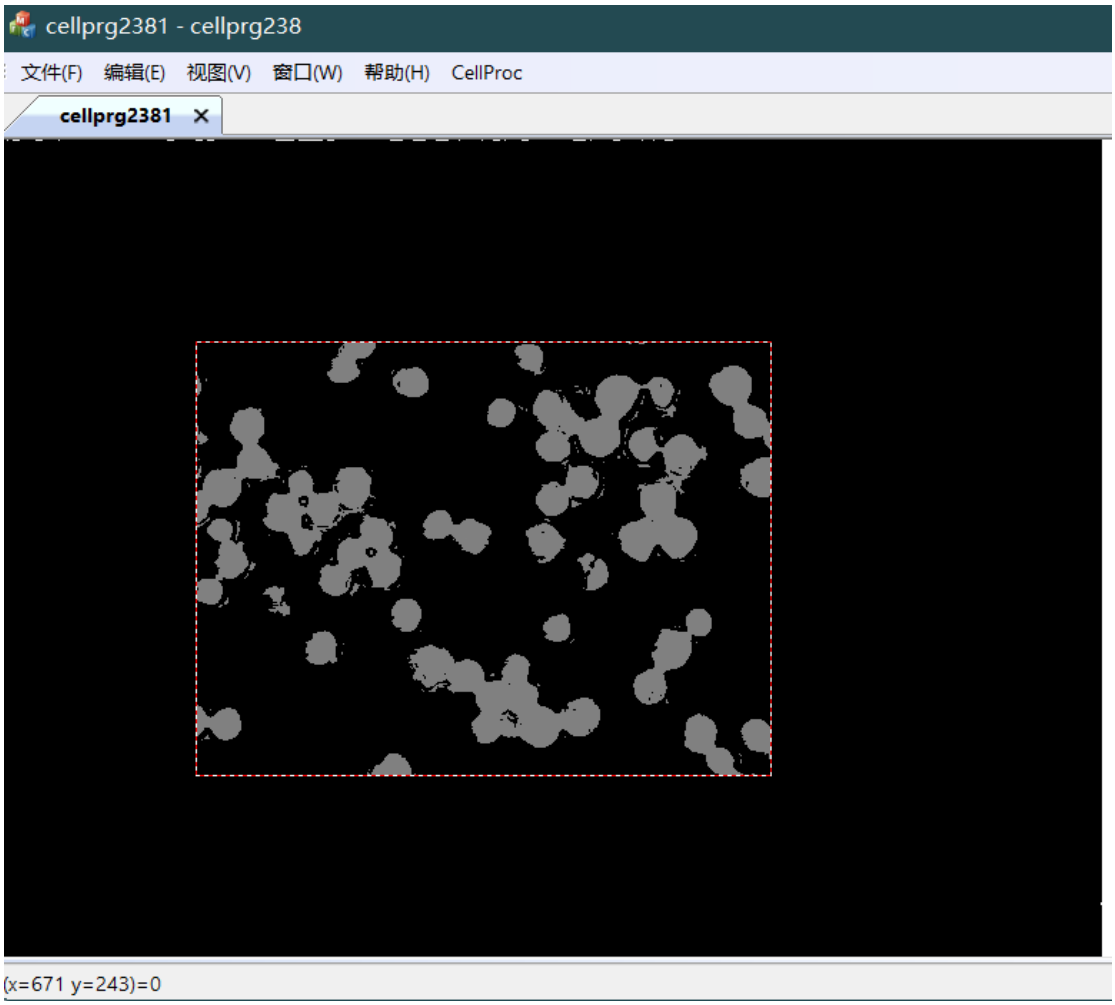

图 11 填洞现象

3.6 shrink()-收缩

代码分析：

收缩的目的是为了方便计数。扫描图像，对图像进行预先的 3 次腐蚀，判断所生成边界点，然后根据原理判定是否标注该点，存放所标志的中心点，便于统计细胞个数及计算细胞半径。

算法：由 Mark 生成边界，我们有四邻域生成边界和八邻域生成边界。判断该点是否为 Mark 点，如果是 Mark 点的话，我们判断 i、j 是否是我们选取图片的边界，如果是的话，我们将该点变成边缘点，否则我们判断它的上下左右(周围八个点)是否有非 Mark 点，如果有，则将这边变成边缘点，反之，不变。最后去掉边界则完成收缩。

```
void CCellPrg238View::OnCellShink()
{
    if (m_CellProc == 3)
    {
        // TODO: 在此添加命令处理程序代码
        m_CellProc = 4;
        //CImgProcDoc* pDoc = GetDocument();
        const int pre_shrink_count = 3;
        unsigned char* lpSrc;
        // 先去掉pre_shrink_count层皮
        GenEdge();
        for (int k = 0; k < pre_shrink_count; k++)
        {
            for (int i = StartPoint.y; i <= EndPoint.y; i++)
            {
                lpSrc = (unsigned char*)m_pImage + m_nLineByte * (m_nHeight - 1 - i) + StartPoint.x - 1;
                for (int j = StartPoint.x; j <= EndPoint.x; j++)//遍历i 做多少次
                {
                    lpSrc++;

                    // 去掉边界!
                    if (*lpSrc & EDGEPOINT)
                        (*lpSrc) &= NO_MARK;//marked=0;变成背景
                }
            }
            if (k == 0)//偶数
                GenEdge4();//4边缘
            else
                GenEdge();

        }
        Invalidate(true);
    }
    else
        MessageBox(TEXT("请先执行Fill Holes"));
}
```

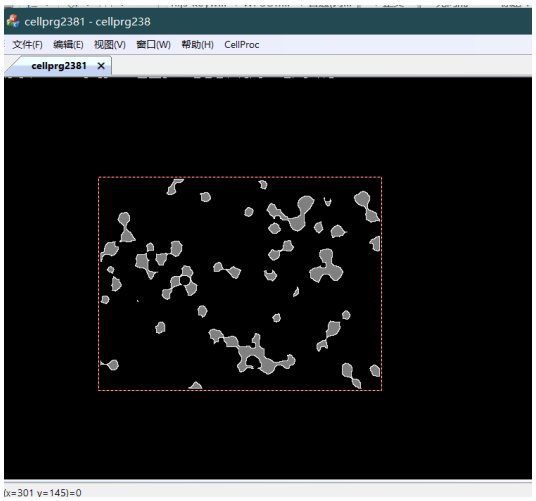

图 12 收缩主要代码及现象

3.7 findcenter()-中心点获取

代码分析：

通过判断 MARK 点上下左右四个方向或者八个方向是否有非 Mark 点，如果有的话即认为是边界，将该点加上边界标志。在收缩的过程（知道没有边界可去）中，清除所有标志点，在不超过边下，没有被访问过的边缘，四个方向全是非 Mark 的话，（收缩 k<2 时）则是孤立点。K>=2，是中心点，保存 x，y，半径。还有除了边界就是背景，这也要保存（就是消失前保存）。

Mark(i,j)：八个方向，没有访问过的 Mark，且是边界点，则保存；SaveIT(I,j)：这点不是中心点（相当于没有访问过）则设为中心点(并保存)；已经是中心点则去除标志，八方向寻找中心点。值得注意的是我们也要去除半径不大于 2 的孤立点，因为我们认为它的半径太小，是噪声。如果是半径大于 2 的孤立点，我们对他进行标记成中心点，对半径做一点补偿（pt.radius=k+pre_shrink_count+4，4 为补偿）。然后在入队。

合并中心点：本点是中心点，则八个方向看都不是中心点，则保存。有相连的话，多点求中心与半径(用到了 CalcCenterArea(i.j))，CalcCenterArea(i.j)：八方向求中心点后去除标志，循环统计。半径取最大，而 x，y 则是取平均。

去除多余圆或者错误圆：

1、相近圆：圆心距离小于 10，则取半径小的中心点为中心点，取半径最大值+2；

2、去除潜在的错误，即半径小于 9 的去掉。但四个方向，边部补偿，防止边部的细胞被去除。

3、多圆相交时，不相交部分小于则去除。具体操作:首先，相交部分先取出来保存。之后，单个圆内的交上了被标志(画一个方框然后统计之内的点数)，之后单个圆不相交的进行统计。不相交的占 50%以下则清除本圆。

```
//多次去噪、收缩初步获得的中心点个数
for (int j = StartPoint.y; j <= EndPoint.y; j++)
{
    lpSrc = (unsigned char*)m_pImage + m_nLineByte * (m_nHeight - 1 - j) + StartPoint.x - 1;
    for (int i = StartPoint.x; i <= EndPoint.x; i++)
    {
        lpSrc++;
        {
            if (j > StartPoint.y && j<EndPoint.y && i>StartPoint.x && i < EndPoint.x) // 最边上的不用处理
            {
                m_bFullEdge = true;
                if (*lpSrc & EDGEPOINT && !(*lpSrc & VISITED))  // 没有访问过的边界
                {
                    if (!(*(lpSrc - 1) & MARKED) &&
                        !(*(lpSrc + 1) & MARKED) &&
                        !(*(lpSrc + m_nLineByte) & MARKED) &&
                        !(*(lpSrc - m_nLineByte) & MARKED))
                    {
                        if (k == 0) // 基本上这种是噪音
                        {
                            continue;
                        }
                        // 孤立的点

                        *lpSrc |= CENTERED;
                        // 保存一下CENTER_POINT信息
                        pt.x = i;
                        pt.y = j;
                        pt.radius = k + pre_shrink_count + 4;//circle adjust
                        points_temp.push_back(pt);

                        continue;
                    }
                    else
                        MarkIt(i, j); // 判断是否需要保存
                // 没有访问过 标志了 并且是非边缘邻域
```

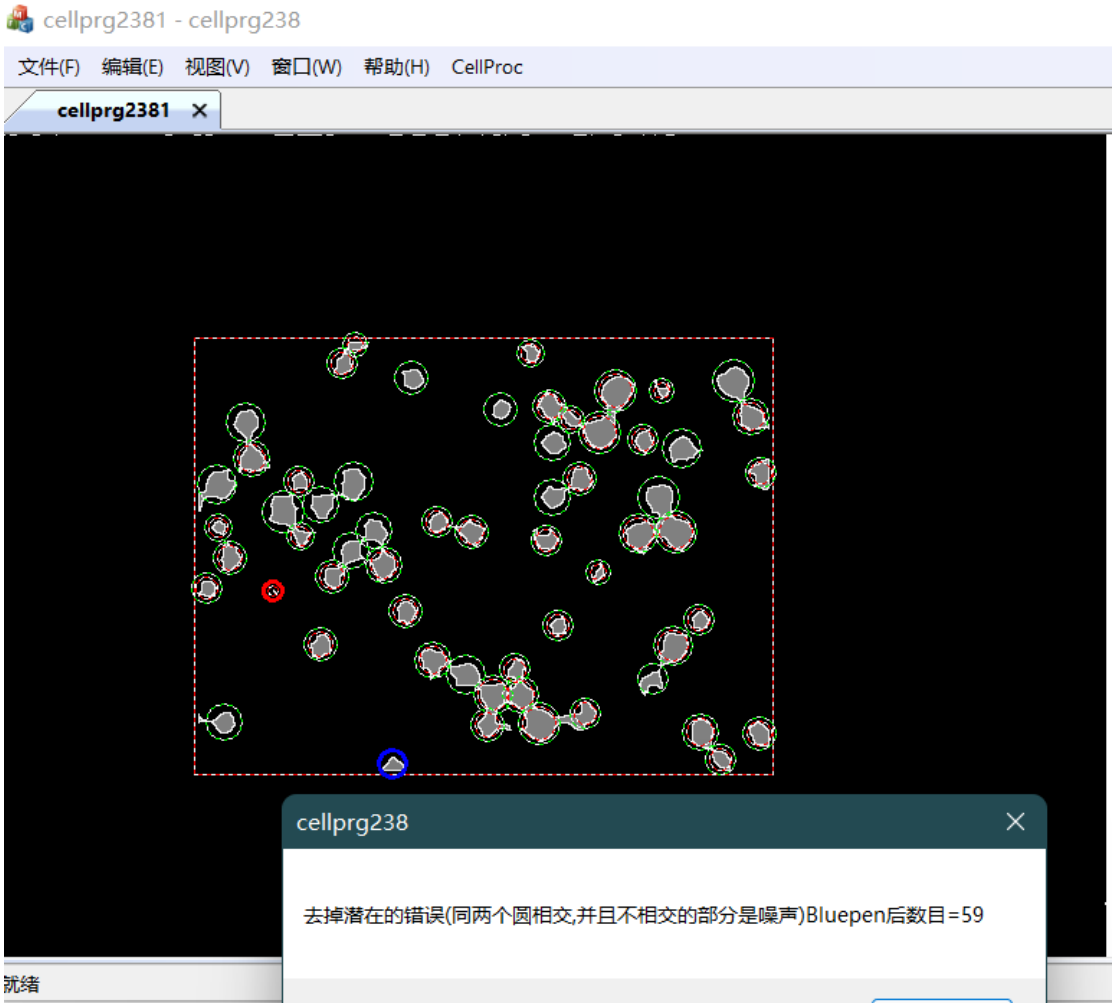

图 13 中心点获取主要代码及现象

3.8 count()-统计

代码分析：

首先图像重新打开，然后我们获取细胞内部的 HSI 的最大值和最小值，并且计算细胞，平均半径，平均面积等值。Count()函数中首先调用了重载函数 reload(),将图片还原，随后将之前细胞识别所保留下来的信息(细胞个数)，从 view.h 中提取信息，将细胞数、半径、面积 、显示在对话框中。

1）利用前面修正的结果，在 Count 模块进行总的计数，计算出平均半径和平均面积并输出。

2）文件重新加载模块。如果 noclick 置为 true，则进行重新加载图片

```
min[0] = min[1] = min[2] = 255;
max[0] = max[1] = max[2] = 0;
double add;
for (int j = 0; j < 3; j++)
{
    add = 0;
    for (int i = 0; i < 256; i++)
    {
        // 清零
        add += m_nHistHSI[j * 256 + i];
        if (add > tota / 20) {//>5%
            min[j] = i;
            i = 256;//out
        }

    }
}
for (int j = 0; j < 3; j++)
{
    add = 0;
    for (int i = 255; i > 0; i--)
    {
        // 清零
        add += m_nHistHSI[j * 255 + i];
        if (add > tota / 20) {//>5%
            max[j] = i;
            i = 0;//out
        }

    }
}

Invalidate(true);
MessageBox(TEXT("ReLoad Image"));
```

```
void CCellPrg238View::OnDraw(CDC* pDC)//绘图
{
    CCellPrg238Doc* pDoc = GetDocument();  //是标准框架代码（不是自己增加的代码）的pDoc，是文档类（(
    ASSERT_VALID(pDoc);
    if (!pDoc)
        return;

    // TODO: 在此处为本机数据添加绘制代码
    if (noclick)
        LoadBmp();
    noclick = false;
    //ShowBitmap(pDC, BmpName);              //显示图片
//}
// TODO: 在此处为本机数据添加绘制代码
    CSize size;
    size.cx = bih.biWidth; size.cy = bih.biHeight;
    dispaly(pDC, CPoint(0, 0), size);   //pdcpDC是CDC类的一个指针时
    //在onDraw函数中添加代码实现打开一幅图像的功能。获取RGB、HSI信息


}
```

图 14 重新加载图片

图 15 最终标记现象

3.9 OnCellprgAll( )-按序执行全部

```
void CCellPrg238View::OnCellprgAll()
{
    // TODO: 在此添加命令处理程序代码
    OnCellprgMark();
    OnCellprgErzhihua();
    OnCellprgFillhole();
    OnCellprgCellshink();
    OnCellFindCenter();
    OnCellprgCountt();
}
```

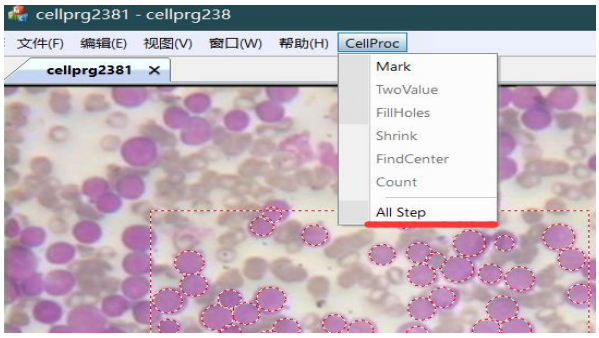

图 16 OnCellprgAll( )

代码分析：不难看出调用了前面各个函数可以依次实现细胞识别的所有操作步骤。

四、错误情况及分析

4.1 未声明全局变量信息头

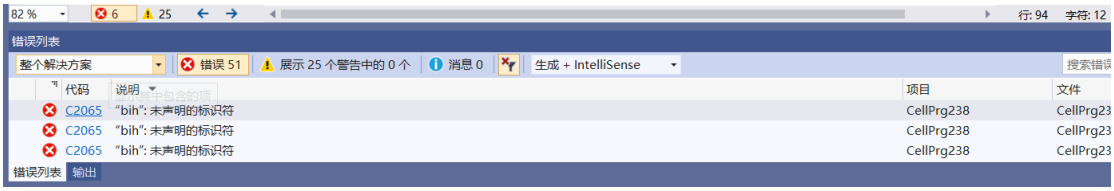

图 17 错误案例 1

4.2 错把指针当作 int 型

修正:应将 int m_pImage;改为 BYTE* m_pImage;

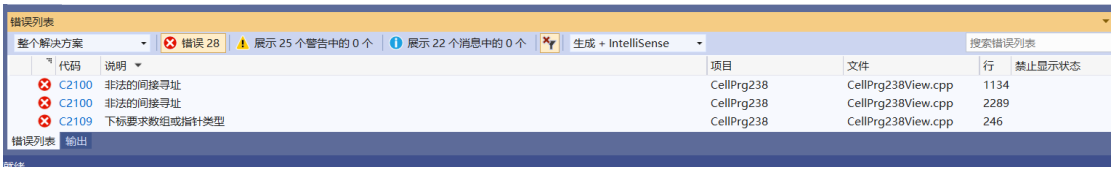

图 18 错误案例 2

4.3 软件菜单栏突然丢失

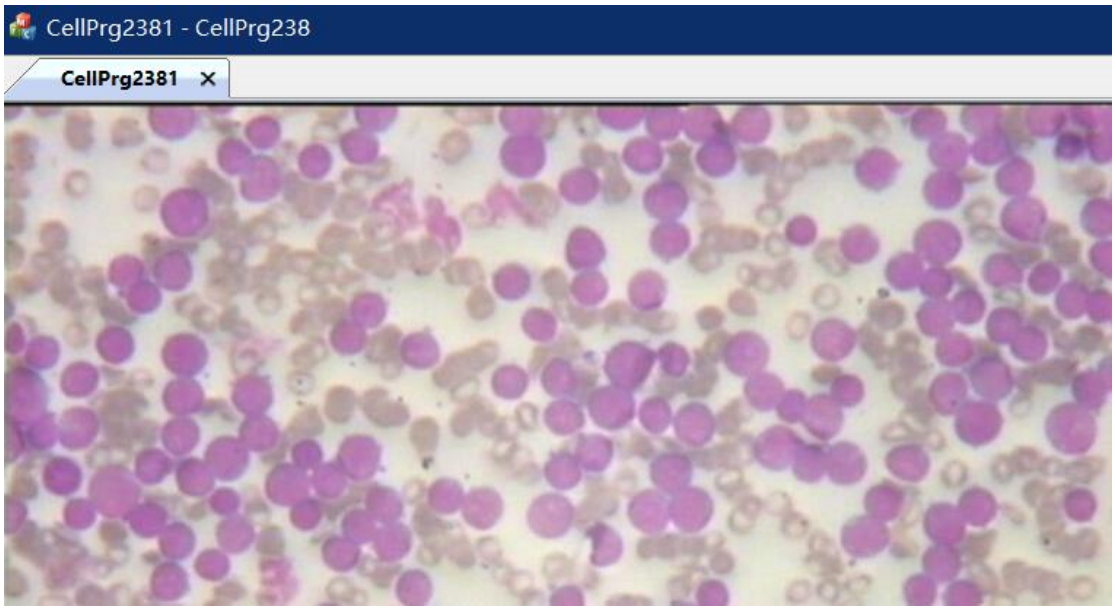

图 19 错误案例 3

原因：误碰软件，把菜单移出去了，可以移动菜单栏使其正常

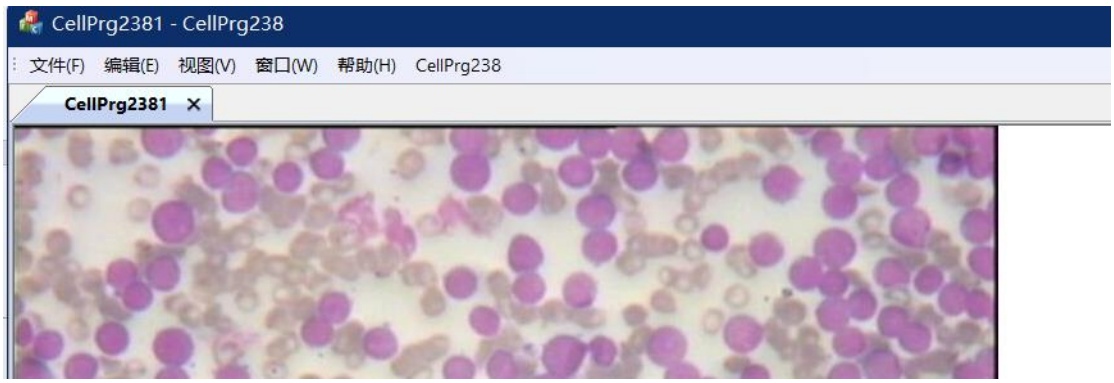

图 20 修正错误

4.4 矩形框在鼠标未按下就出现了

原因:OnMouseMove 中判断条件不对

五、心得体会

在这次为期 2 周的课程设计中，我真的学到了很多，同时学会如何打开和自动打开 bmp 图像，对其进行进一步操作，将图像 RGB 信息转为 HIS，再 Mark 标记,边缘提取，二值化，填洞，收缩，中心点获取，中心点统计等。Mark 实际上是对我们所要的要素进行标记，便于识别，基本思想是按半径进行区分一定是细

胞和可能是细胞的区域。二值化的意义在于便于轮廓提取；实现办法是设置一个阈值，将图像分割成大于阈值和小于阈值的两部分像素群。为什么要填洞？如果不填洞对后续细胞统计干扰很大，说不清到底是一个细胞还是两个细胞，因此我们需要填洞；孔洞填充的原理如下：在阈值处理时，如果像素在阈值范围内，则像素将被标志。孔洞填充将先统计所有连通的非标志区域面积，总会有一个或者几个面积特别大的区域，其它的都是面积相对较小的区域。较小或者很小的往往就是系统所要填充的孔洞了。收缩实质是先去掉边缘点，然后将剩下的 Mark 点生成边缘点，再去掉在生成，直到去掉三次边缘，就这样收缩，如果收缩过程中有些 Mark 点收缩到很小，可以认为那不是细胞，那就可以直接去掉。获得中心点是这次课程设计最关键的一部分，这个算法决定着结果的好坏。中心点的获取就是在每一次的收缩之前先进性一次判断，如果该点是孤立点或者是全边界点则将该点保存起来。因为对于孤立点和全边界点如果不进行保存的话在进行一次收缩之后该点就会消失，因此为了保存该点，就要在其消失前将其保存。最后每一个细胞收缩到最后可能剩下一个点到四个点。因此第一次获取的中心点个数会是真正中心点个数的三倍左右。细胞半径大小的获取是通过判断该中心点是通过多少次收缩之后获得的从而近似获得细胞的半径。统计，实质就是通过前面的中心点，半径来计数即可。Count()函数中首先调用了重载函数 reload(),将图片还原，随后将之前细胞识别所保留下来的信息(细胞个数)，从 view.h 中提取信息，将细胞数、半径、面积 、显示在对话框中。同时细胞处理方法很大程度是与参数设定有关，涉及的影响因素很多，在这次课程设计中真正意义上理解了之前图像处理课程的理论，现在在原有的 C++基础上多学会了一种 MFC 图像处理方式。

在这次图像处理课程设计中，我认为最大的挑战是最开始的打开图片与完成 OnMouseMove 显示 HSI 的过程，虽然并没有后面的算法复杂，但是这是一个从无到有的过程；从零基础的 VS2019 的 MFC 项目到最后可以独立完成一个自己 MFC 项目。在不断上网查询有关技术和询问老师同学的协助下；我终于完成了鼠标移动显示 HSI 的任务，然后根据老师提供的核心参考代码，不断调试，查看报错，完善代码，完善函数与变量的声明，解决报错，一步步走下去，完成了全部任务，虽然辛苦但是很值得，这让我们切实的体会与学习到很多。

六、致谢

在这短暂两周时间内，从接到课题到全部调试完成，经历了很多困难，真心感谢老师在群里各种教导和学长学姐对我的指导，在你们的帮助下顺利完成此次课设。金无足赫，人无完人。由于本人学术水平有限，所写报告难免有错误之处。恳请各位老师和同学与指正!

七、参考文献

[1]刘腾飞，刘威.基于 Matlab 的肿瘤细胞识别系统[J].电子设计工程,2021,29(06):1-5.

[2]黄松. 基于图像识别方法体细胞计数系统的研究[D].山西农业大学,2013.

[3]朱会平，陈志远.基于细胞显微图像的数量统计应用[J].实验室科学,2011,14(05):73-75.

[4]蔡朋杞. 红细胞识别系统的设计与实现[D].电子科技大学,2011.

[5]赵秋影. 人体细胞识别技术研究[D].长春理工大学,2007.