目录

[1.基本元素 2](#_Toc26325)

[1.1 HMat结构 2](#_Toc9893)

[1.2 HRegion结构 2](#_Toc3899)

[2.算子详解 2](#_Toc12636)

[2.1 blob分析 2](#_Toc31615)

[2.1.1 threshold算子 2](#_Toc32498)

[2.2 set(集合运算) 2](#_Toc13489)

[2.2.1 union2算子(并集操作) 2](#_Toc25120)

[2.2.2 intersection算子(交集操作) 3](#_Toc258)

[2.2.3 difference算子(差集操作) 3](#_Toc13631)

[2.3 connection(连通域分析) 3](#_Toc4750)

[2.3.1 connection算子(连通域分析) 3](#_Toc18048)

[2.3.2 select\_shape算子(选择Region) 4](#_Toc22845)

[2.3.3 select\_shape\_max\_std算子(选择面积最大Region) 4](#_Toc7865)

[2.4 morph(形态学运算) 4](#_Toc30934)

[2.4.1 矩形结构元素 4](#_Toc16319)

[2.4.2 圆形结构元素 5](#_Toc11383)

[2.4.3 圆形/矩形获得 7](#_Toc7780)

行程编码算子思路

# 1.基本元素

## 1.1 HMat结构

HMat结构采用的Opencv的Mat结构，存储图片的一种格式，详情请参考Opencv。

## 1.2 HRegion结构

HRegion是存取Region的一种结构，本文采用行程编码的原理，故需要一些容器存储行程编码。

# 

# 2.算子详解

## 2.1 blob分析

### 2.1.1 threshold算子

⑴算法思路：遍历行列所有的元素，查看元素的值是否满足要求。

⑵核心代码：见blob.cpp/threshlod函数

⑶优化思路:使用OpenMP多线程并行计算,使用异步数据处理当前这张图。

## 2.2 set(集合运算)

### 2.2.1 union2算子(并集操作)

⑴算法思路：

Step1：判断两个Region是否均存在数据

Step2：获得当前rle的同一行的所有行程编码(行开始到行结束)

Step3：获得当前Region的同一行的所有行程编码(行开始到行结束)

Step4：在两个Region行信息相同时，计算当前行的并集；若行信息不同时，

取行信息较小的数；若有行超过Region结束值，取另一个Region的信息；

计算并集的原理：将两个region所有的行程编码均取出；

①获取两个Region均为空的最小列开始值作为开始值，获取下一个两个

Region均为空的最大列结束值作为结束值；

②遍历所有的行列信息；

⑵核心代码：见morph.cpp/union2函数

### 2.2.2 intersection算子(交集操作)

⑴算法思路：

Step1：判断两个Region是否均存在数据

Step2：获得当前rle的同一行的所有行程编码(行开始到行结束)

Step3：获得当前Region的同一行的所有行程编码(行开始到行结束)

Step4：在两个Region行信息相同时，计算当前行的交集；

计算交集的原理：对比两个Region同时存在的值，即为交集部分；

①将两个Region合并成一个Region，升序排序(从小到大排序)，获得第一个

列结束的值，获得下一个最小的列开始值和列结束值；

②若当前列开始值小于最小的列结束值并且当前列结束值大于最小结束值；

获得当前交集的长度，以及列开始值；

③遍历所有的行列信息；

⑵核心代码：见morph.cpp/intersection2函数

### 2.2.3 difference算子(差集操作)

⑴算法思路：

Step1：判断两个Region是否均存在数据；

Step2：获得当前rle的同一行的所有行程编码(行开始到行结束)；

Step3：获得当前Region的同一行的所有行程编码(行开始到行结束)；

Step4：在两个Region行信息相同时，计算当前行的差集；若rle信息存在而

Region信息不存在时，取rle信息存在的行程编码；

计算差集原理：寻找rle存在的行程编码而Region不存在的行程编码

⑵核心代码：见morph.cpp/difference函数

## 2.3 connection(连通域分析)

### 2.3.1 connection算子(连通域分析)

⑴算法思路：

Step1：给第一行所有行程编码标号；

Step2：对于除了第一行外的所有行里的团，如果它与前一行中的所有团都没

有重合区域，则给它一个新的标号;如果它仅与上一行中一个团有重合区域，则将上一行的那个团的标号赋给它;如果它与上一行的2个以上的团有重叠区域，则给当前团赋一个相连团的最小标号，并将上一行的这几个团的标记写入等价对，说明它们属于一类；

Step3：将等价对转换为等价序列，每一个序列需要给一相同的标号，因为它们都是等价的；

Step4：遍历开始团的标记,查找等价序列,给予新的标记；

计算团、等价对原理：

①完成团的标记和等价对列表的生成，将列表P改成等价对中最小的值；

②给列表P重新排序；

③将团的标记重新更新；

⑵核心代码：见morph.cpp/connection函数

### 

### 2.3.2 select\_shape算子(选择Region)

⑴算法思路：

Step1：计算所有Region的最大宽度值/最小宽度值，最大高度值/最小高度值和面积；

Step2：获得Region的宽度值和高度值；

Step3：判断当前宽/高/面积属性是否存在；

Step4：若当前宽/高/面积属性同时满足要求，输出满足要求的Region以及给Region重新编码；

⑵核心代码：见morph.cpp/select\_shape函数

### 2.3.3 select\_shape\_max\_std算子(选择面积最大Region)

⑴算法思路：

Step1：计算所有Region面积；

Step2：获得最大面积的ID；

Step3：获得最大面积的所有行程编码；

⑵核心代码：见morph.cpp/select\_shape\_max\_std函数

## 2.4 morph(形态学运算)

### 2.4.1 矩形结构元素

#### 2.4.1.1 erosion\_rectangle1算子(矩形结构腐蚀)

⑴算法思路：

Step1：计算需要做交集的次数，时间复杂度为(log2^(n))，n为高度；

Step2：计算去掉宽度后的行程编码(不满足宽度要求的除去)，原点在左边，获得新的行程编码；

Step3：使用当前级数取交集获得下一级数的Region(同一级Region的列开始信息和列结束信息均一样，行信息不一样)；

①获取另一个Region的行信息参数(行信息相差2^(n-1)，n表示当前级数)(当树为不完全树，最后一级差height-2^(n-1)，height为当前矩形元素的高度)；

②两个Region取交集；

Step4：将行程的值移动回原点，x的偏移为log2^(width -1)，y的偏移为log2^(height -1)；

⑵核心代码：见morph.cpp/erosion\_rectangle1函数

#### 2.4.1.2 dilation\_rectangle1算子(矩形结构膨胀)

⑴算法思路：

Step1：计算需要做并集的次数，时间复杂度为(log2^(n))，n为高度；

Step2：计算加上宽度后的行程编码，原点在左边，获得新的行程编码；

Step3：合并当前Region，输出新的行程编码；

Step4：使用当前级数取并集获得下一级数的Region(同一级Region的列开始信息和列结束信息均一样，行信息不一样)；

①获取另一个Region的行信息参数(行信息相差2^(n-1)，n表示当前级数)(当树为不完全树，最后一级差height-2^(n-1)，height为当前矩形元素的高度)；

②两个Region取并集；

Step5：将行程的值移动回原点，x的偏移为log2^(width)，y的偏移为log2^(height)；

⑵核心代码：见morph.cpp/dilation\_rectangle1函数

#### 2.4.1.3 opening\_rectangle1算子(矩形结构开运算)

⑴算法思路：

Step1：先腐蚀；

Step2：后膨胀；

⑵核心代码：见morph.cpp/opening\_rectangle1函数

#### 2.4.1.4 closing\_rectangle1算子(矩形结构开运算)

⑴算法思路：

Step1：先膨胀

①操作流程如同矩形结构膨胀，最终行程移动回原点，需保留膨胀后的所有信息；

Step2：后腐蚀

⑵核心代码：见morph.cpp/closing\_rectangle1函数

### 2.4.2 圆形结构元素

#### 2.4.2.1 erosion\_circle算子(圆形结构腐蚀)

⑴算法思路：

Step1：根据半径获得圆形结构元素的参数；

Step2：将圆形结构元素中相同长度且相邻的行程编码放在一起(记录当前行程编码的长度/列开始/列结束/行值)；

Step3：相同长度的矩形块算出减去宽度后对应的行程编码的数据(这里的个数为step2的一半加1，因为圆形结构元素是对称的,算一半即可推出另一半)，计算满足宽度的行程编码的值(原点为中心点)；

Step4：计算圆形结构元素最中间长度相等且相邻的行程编码相互取交集后的结果(方法参考矩形元素)；

①获得需要做交集的次数，时间复杂度为(log2^(n))，n为高度；

②根据当前级数取交集获得下一个级数的Region；

③输出最终Region的行程编码；

Step5：计算圆形结构元素长度相等且相邻的行程编码相互取交集后的结果(方法参考矩形元素)(圆形结构元素上下对称,移动y值可以获得另一半)。

①获得需要做交集的次数，时间复杂度为(log2^(n))，n为高度；

②根据当前级数取交集获得下一个级数的Region；

③输出最终Region的行程编码；

Step6：计算剩下所有Region的交集，得出最终的行程编码；

Step7：将行程的值移动到原点的值(在n.0<Radius<n.5之间x向左移动一，y无偏移；n.5<=Radius<n.0+1之间x，y无偏移；n为整数；n>0；在圆形结构元素中做腐蚀，半径为整数时，腐蚀宽高为偶数，中心应稍偏左一点；其他不变)。

⑵核心代码：见morph.cpp/ erosion\_circle函数

#### 2.4.2.2 dilation\_circle算子(圆形结构膨胀)

⑴算法思路：

Step1：根据半径获得圆形结构元素的参数；

Step2：将圆形结构元素中相同长度且相邻的行程编码放在一起(记录当前行程编码的长度/列开始/列结束/行值)；

Step3：相同长度的矩形块算出加上宽度后对应的行程编码的数据(这里的个数为step1的一半加1，因为圆形结构元素是对称的，算一半即可推出另一半)。(原点为中心点)；

Step4：遍历step3所有Region的行程编码取并集之后的信息；

Step5：计算圆形结构元素最中间长度相等且相邻的行程编码相互取并集后的结果(方法参考矩形元素)；

①获得需要做并集的次数，时间复杂度为(log2^(n))，n为高度

②根据当前级数取并集获得下一个级数的Region

③输出最终Region的行程编码

Step6：计算圆形结构元素长度相等且相邻的行程编码相互取交集后的结果(方法参考矩形元素)(圆形结构元素上下对称,移动y值可以获得另一半)

①获得需要做并集的次数，时间复杂度为(log2^(n))，n为高度

②根据当前级数取并集获得下一个级数的Region

③输出最终Region的行程编码

Step7：计算剩下所有Region的并集，得出最终的行程编码；

Step8：将行程的值移动到原点的值(n.0<Radius<n.5之间x,y无偏移；n.5<=

Radius<n.0+1之间x,y无偏移； n为整数；n>0；在圆形结构元素中做膨胀,中心均不变)。

⑵核心代码：见morph.cpp/ dilation\_circle函数

#### 2.4.2.3 opening\_circle算子(圆形结构开运算)

⑴算法思路：

Step1：先腐蚀；

Step2：后膨胀；

①操作流程如同圆形结构膨胀；在开运算的膨胀需考虑原点的移动。在获得圆形结构元素中，若n.0<Radius<n.5之间，圆形结构元素x右偏1，y右偏1;若n.5<=Radius<n.0+1之间，圆形结构元素原点不变；n为整数；n>0；

其余操作流程如圆形结构元素的膨胀。

⑵核心代码：见morph.cpp/ opening\_circle函数

#### 2.4.2.3 closing\_circle算子(圆形结构闭运算)

⑴算法思路：

Step1：先膨胀；

①操作流程如同圆形结构膨胀；在开运算的膨胀需考虑原点的移动。在获得圆形结构元素中，若n.0<Radius<n.5之间，圆形结构元素x右偏1，y右偏1;若n.5<=Radius<n.0+1之间，圆形结构元素原点不变；n为整数；n>0；

②最终行程移动回原点，需保留膨胀后的所有信息；

其余操作流程如圆形结构元素的膨胀。

Step2：后腐蚀；

⑵核心代码：见morph.cpp/ closing\_circle函数

### 2.4.3 圆形/矩形获得

#### 2.4.3.1 gen\_rectangle1算子

⑴算法思路：

Step1：根据给定的行开始和行结束，遍历所有的行，获得列开始值和列结束值；

⑵核心代码：见morph.cpp/ gen\_rectangle1函数

#### 2.4.3.2 gen\_circle算子

⑴算法思路：

Step1：规划圆形半径选取范围(n.0<Radius<n.5之间Radius=n.0；n.5<=Radius

<n.0+1之间Radius=n.5；n为整数；n>0)。

Step2：若圆形半径为1.5，输出对应的行列信息；

Step3：根据当前点的dy，计算当前点的dx，从而计算当前圆形的行程编码。当前偏移量为(j1+(j2-1))/2，中心偏左。

⑵核心代码：见morph.cpp/ gen\_circle函数