目录

[1. D5切图算法分析 2](#_Toc8026)

[2. 优化路径 2](#_Toc7480)

[2.1 行程编码 2](#_Toc14587)

[2.2 OpenCV-CCA 3](#_Toc16908)

[2.3 OpenCV-Contour 3](#_Toc31867)

[3. 形态学运算耗时分析 3](#_Toc19520)

[4. 连通域提取耗时分析 8](#_Toc8170)

[5. 集合运算耗时分析 9](#_Toc23407)

[5.1 Union1函数耗时分析 9](#_Toc18179)

[5.2 union2函数耗时分析 12](#_Toc9426)

[5.3 intersection函数耗时分析 12](#_Toc30966)

[5.4 difference函数耗时分析 13](#_Toc1603)

[6. 通道拆分耗时分析 13](#_Toc2525)

[7. 颜色空间转换耗时分析 14](#_Toc25214)

[8. 图片读取耗时分析 15](#_Toc15489)

1. D5切图算法分析

行程编码版连通域分析库比OpenCV版性能更优，但与Halcon仍然存在较大差距，主要体现在形态学腐蚀和膨胀两个算子上。使用行程编码、OpenCV和Halcon实现D5切图算法，部分函数耗时如图1.1所示。

由图1.1可见，不论是行程编码还是OpenCV，与Halcon的差距主要在形态学运算。基于基础版行程编码的腐蚀和膨胀是直接套用的数学公式实现，无任何优化，与Halcon的性能对比也说明存在较大优化空间。基于Jump-Miss版行程编码算法针对有很多小物体的图片进行了优化，主打任意形状的结构元素，与实际应用场景不符。OpenCV版对于使用矩形作为结构元素的腐蚀和膨胀可能使用了一些优化方法，如将一个二维滤波器分解成两个一维滤波器，表现比行程编码更好；但是当结构元素是半径为50的圆时，处理时间高达3968ms，直接导致处理一张图需要5s以上。除此之外，OpenCV的连通域提取也耗时较长，轮廓查找耗时32.96ms，轮廓填充便于之后的形态学运算耗时319.16ms，共耗时354.12ms。

1. 优化路径
   1. 行程编码

该方案的关键在于对腐蚀和膨胀算法进行优化。测试表明当前算法和Halcon对同一区域的行程编码是一致的，因此可以暂时不考虑使用列行程而不是行行程可能效率更高的情况。需要考虑的优化项如下。

1. 寻找新的基于行程编码的腐蚀及膨胀算法。理解Jump-Miss算法原理，考虑能否对其进行优化。
2. 优化图片数据的访问速度。行程编码版可以使用任意区域为图片设定ROI，如同Halcon，但是这样访问图片数据的速度会变慢，并直接影响threshold，intensity等一系列算子。以intensity例，使用矩形区域的行程编码作为ROI与OpenCV中矩形ROI相比，速度慢了一倍。除此之外，对threshold进行测试，将二值化阈值设为255，Halcon中遍历一张图所需时间为1.16ms，而在行程编码版中遍历图片需要12.09ms，正常执行threshold的时间为1.74ms v.s. 14.26ms，这表明时间主要消耗在图片遍历上。需要在弄清原因的基础上，考虑是否应当尝试OpenCV中图片像素遍历的方式，使用矩形ROI，并配一张等大掩码图片。
3. 优化行程编码的速度。Halcon中一个行程使用了3个short类型编码，row,start\_col,end\_col,共6字节，并需要288个字节的额外空间。当前行程编码中的一个行程需要2个short和一个int，共8个字节。需要考虑其中字节对齐的问题，并对行程的插入速度和读取速度进行测试。
4. 对于HRegion和HMat,赋值运算符与复制构造函数设定为浅拷贝，即指向同一份图像或行程编码数据；添加成员函数copyTo实现深层复制，即数据的复制。
5. 计算区域的属性，如area\center\circularity等，为了便于添加新属性，应当使用strategy模式。为了计算效率，有些属性，如area和center，应当一同计算。这些属性缓存在对象中，需要的时候才计算。
6. 边缘检测系列算子需要short甚至float类型HMat,因此处理HMat的系列算子，如二值化等，是否需要写成模板函数？或在函数内加入型别判断？
   1. OpenCV-CCA

该方案核心是使用OpenCV完成形态学运算，适用于进行腐蚀和膨胀的结构元素是大小在20\*20以下的矩形（20ms），或半径在5以内的圆形(35ms)的项目需求。针对此类项目需求，还需要将连通域提取算法进行优化，使用OpenCV 4.0中的连通域标注算法取代当前基于轮廓查找和轮廓绘制的算法。除此之外，在形态学运算之前，需要添加padding的相应代码。

* 1. OpenCV-Contour

该方案的核心是使用OpenCV中的轮廓查找来进行连通域分析，优势在于区域属性，如面积、周长、矩等已有实现。难点在于如何对轮廓进行腐蚀和膨胀，但此类算法也有相应文献可以参考。

1. 形态学运算耗时分析

当结构元素是大小在20\*20以下的矩形（20ms），或半径在5以内的圆形(35ms)，使用OpenCV完成形态学运算是比较经济的。其它情况下，行程编码是更优的选择。当待处理的区域面积较大，使用基础版行程编码算法可以满足需求；当待处理的区域数量较多，面积较小，使用Jump-Miss版行程编码算法可以满足需求；实际需要处理的图片在两者之间，需要在理解算法的基础上进行优化才能满足需求。

由图3.1和图3.2可见，行程编码-basic对大区域的处理速度比Halcon差，但基本能满足需求。OpenCV仅在结构元素较小(20\*20)时可以满足需求。行程编码-Jump-Miss在区域面积较大的情况下完全不能用。

由图3.3和图3.4可见，当图片中有大大小小的区域，Halcon依然表现优秀，OpenCV在结构元素较小(20\*20)时依然可以满足需求，但是当前的行程编码算法表现糟糕，与Halcon差了几个数量级。

由图3.5和图3.6可见，对于单一大面积区域，行程编码-basic处理速度比Halcon慢，但基本能满足需求。OpenCV仅在圆形结构元素的半径较小(<5)时可以满足需求。行程编码-Jump-Miss在该情况下完全不能用。

当圆形半径大于或等于10(大小20\*20)，腐蚀的时间几乎为0。如果区域内保存了最长行程长度，而最长行程长度又小于结构元素，就可以直接获得空区域的结果。

由图3.7和图3.8可见，当图片中有大大小小的区域，Halcon依然表现优秀，OpenCV在结构元素较小(半径<5)时依然可以满足需求，但是当前的行程编码算法与Halcon差距较大，Jump-Miss版比basic略好。

1. 连通域提取耗时分析

由图4.1可见，支持灰度区间的二值化函数OpenCV-LUT的耗时是支持单一阈值的OpenCV-threshold的两倍。值得一提的是Halcon对5472\*3648的D5图片执行二值化仅需1.74ms，其中包括了图像数据访问和行程编码生成的时间，相较而言OpenCV-threshold的二值化操作更简单，却需要2.78ms。

其中行程数以千为单位。KFactor = NumRuns/sqrt(Area)。LFactor指每一行的平均行程数。MeanLength 指行程长度的平均值。行程编码-Halcon指图4.1中行程编码与Halcon的耗时之差。由图4.2可见，二者耗时之差与行程数的变化基本一致，可以推测行程编码时的插入速度是需要优化的。

由图4.3可见，使用行程编码进行连通域提取具有一定优势，虽然比Halcon慢，但是基本能满足需求，主要的优化项在threshold算子上。虽然OpenCV-Contour的速度比行程编码慢，但是OpenCV在threshold上的耗时在复杂图片上比行程编码更短，因此可以考虑连通域提取只返回轮廓，随后的腐蚀、膨胀等形态学运算也仅对轮廓进行的方案。至于使用轮廓查找和轮廓填充实现的连通域提取算法，对复杂图片耗时很久，但是如果使用图片标注算法替代轮廓算法，性能会得到显著提升。

1. 集合运算耗时分析
   1. Union1函数耗时分析

使用行程编码完成集合运算在实际项目图片中优势明显。使用OpenCV速度仅与区域外接矩形的面积有关，当图片比较复杂时优势明显，但是实际项目中通常不会出现那么复杂的图片。

使用正态分布(mean =100, std=40)生成的大小为4096\*3000随机图片作为测试图片，以下4组实验说明行程编码版Union1算子比Halcon速度更快，但是对于这样的复杂图片，不论是行程编码还是Halcon,速度都比OpenCV慢很多。OpenCV的耗时只与区域外接矩形的面积有关，与构成该区域的行程数无关。当行程数相对较少，即实际项目图片的常见情形，OpenCV的耗时就会显得较长。

* 1. union2函数耗时分析

由图5.6可见，当前行程编码版union2比Halcon速度慢，由于通常图片的行程数不会太多，更可能出现二值化阈值在40左右的情况，基本满足项目需求。OpenCV对所有4096\*3000的区域处理时间相同，均是7.46ms。

* 1. intersection函数耗时分析

由图5.7可见，当前行程编码版intersection比Halcon速度慢，由于通常图片的行程数不会太多，更可能出现二值化阈值在40左右的情况，基本满足项目需求。OpenCV对所有4096\*3000的区域处理时间相同，均是5.21ms。

* 1. difference函数耗时分析

由图5.8可见，Halcon和行程编码速度相仿，OpenCV对所有4096\*3000的区域处理时间相同，均是3.49ms。

1. 通道拆分耗时分析

OpenCV中图片按像素存贮，通道拆分涉及到图片数据的复制，速度主要跟图片大小有关，复制4096\*3000\*3字节图片数据需要22ms。Halcon中decompose3时直接返回原彩色图片的数据，因此函数返回很快。当某一通道被修改后，如paint\_region(Rectangle, R, &R, 255.0, "fill")时，则先复制原数据，然后对图片进行修改（Proxy模式）。

|  |  |  |  |
| --- | --- | --- | --- |
| 图片大小 | Halcon | 行程编码 | OpenCV |
| 1280\*720 | 0.00ms | 1.46ms | 1.42ms |
| 1920\*1080 | 0.00ms | 3.49ms | 3.40ms |
| 3840\*2160 | 0.01ms | 14.76ms | 14.49ms |
| 4096\*3000 | 0.02ms | 22.28ms | 22.03ms |

后续优化可以考虑通过设置step而不是复制数据实现decompose3，但是可能会因为数据的不连续存贮而影响后续处理函数的速度。OpenCV中执行5472\*3648\*3的图片数据复制22.74ms，Halcon复制4096\*3000速度为3.6ms，折合针对4096\*3000个字节，4.9ms V.S. 3.6ms,可考虑是否可以通过DMA之类的数据传输通道以加快图片复制速度。

1. 颜色空间转换耗时分析

由于计算精度不一样，行程编码和OpenCV中采用的是整形，而Halcon中采用的是浮点型，因此速度比Halcon更快，同时也造成了与Halcon在结果上最多有2个灰度的差别。

1. 图片读取耗时分析

OpenCV和行程编码读取图片的速度比Halcon快。由三者在读取png和jpg速度相近，推测Halcon只是在读bmp图片时速度较慢，并没有在图片读取中做额外的事情。