**切分、识别等模块与软件协调模块运行流程**

2018.12.26

一、算法流程

染色体切割并识别的总体运行流程如下图所示（**仅供参考，具体以正文为准**）：



注：上图中两次提到“46条单条染色体”，实际上是假设患者是正常人的前提，如果患者为染色体数目异常，则此处数值上下浮动也有可能。

二、各模块功能与接口

1.打分模块（module0）

功能：输入病人的原始核型图，给出评分结果，用于挑选符合进一步分割的图片。

函数原型：int moduleScoring(Mat originPicture, String pictureType, float & pictureScore, int & singleNum)

参数列表：

Mat originPicture 输入，待处理原始图像

String pictureType 输入，表明图像类型的字符串，为“raw”或“tif”，“aipng”等

float & pictureScore 输出，打分结果，打分越高表明该图片质量越高

int & singleNum 输出，初步分割得到的单条染色体数量

返回值 输出，程序正常为0，异常为1

2.优化模块（module0pti）

功能：用于优化原始图像，进行对比度调整。

函数原型：Mat moduleOpti(Mat originPicture, String pictureType)

参数列表：

Mat originPicture 输入，待处理原始图像

String pictureType 输入，表明图像类型的字符串，为“raw”或“tif”，“aipng”等

返回值 输出，Mat类型，处理后的图像

3.自动分割模块（moduleSeg）

功能：将病人的原始核型图进行初步自动切割，用于“**自动切割识别**”业务中的从显微镜读取图片后的切割过程。

函数原型：int moduleSeg(Mat originPicture, String pictureType, String patientId, String glassId, String karyoId, Mat& optiPicture, String& optiPictureType, vector<chromo>& chromoData)

参数列表：

Mat originPicture 输入，待处理原始图像

String pictureType 输入，表明图像类型的字符串，为“raw”或“tif”，“aipng”等

String patientId 输入，患者id，为100位字符串

String glassId 输入，玻片id，为100位字符串

String karyoId 输入，核型id，为100位字符串

Mat& optiPicture 输出，Mat类型，优化对比度处理后的图像

String& optiPictureType 输出，优化图图像格式，目前为“tif”

vector<chromo>& chromoData 输出，存储染色体数据的结构体容器，其中每个结构体chromo对应一条染色体，结构体按如下方式定义：

struct chromo{

int index：一个索引编号，两位整数，与其实际是几号染色体并无关系，索引数据；

int relatedIndex: 相关的索引编号，如果分割类型是single的时候，有可能是交叉染色体分割出来的单条染色体，通过该字段关联原交叉染色体。如果是直接分割出来的单条染色体，该字段为索引本身。

int chromoId：该染色体被识别为第几条染色体。在分割过程中该值始终为0

bool cImgType：分割类型(如果是染色体单体则为true（但需注意，即使软件预判为单条类型，但仍有可能实际上是粘连染色体，需人机交互调用模块2做进一步人工切分）；如果是认为交叉粘连则为false)；

Mat cImg：分割出的原始图片

Mat cImgRotated：分割出旋转后的图片

结构体struct cImgPosition{

Mat cImgMask：该分割图像的掩码（蒙版），分割图像所在的位置标记为1，其余位置标记为0。（未旋转的图片）

int cImgBoundingbox[4]:数组，存储该分割图片的在核型图上外接矩形框位置，分别为y,x,h,w（未旋转的图片）

float cImgOrientation：存储该分割图片的旋转角，用于旋转得到竖直的图片。

}

float chromo\_categoryinfo[25]（一个浮点数数组，从第0个到第24个，其中第0个位置的数值代表系统判断出的其是第几号染色体，存的数值就是其染色体编号，例如“22”说明系统认为其就是22号染色体，“23”说明是“X”染色体，“24”说明是“Y”染色体；数组中第1号到第24号位置分别代表系统判断其是第“1”号染色体的概率，这个数组的所有参数在默认时全为0，当调用染色体识别函数后，改写其第一位为确认的染色体编号，其他位置为相应位置下的染色体的概率值）

int chromoUpright：染色体长臂向下短臂随体向上信息chromoUpright（如正立则为1，倒立为2，未知未判断为0）

}

返回值 输出，程序正常为0，异常为1

4.手动拼接模块（module2）

功能：对于误分割的染色体进行手动拼接。

函数原型：int moduleSplit(Mat originPicture, String pictureType, vector<chromo> chromoDataArray, int newCutIndex, chromo& chromoData)

参数列表：

Mat originPicture 输入，待处理原始图像

String pictureType 输入，表明图像类型的字符串，为“raw”或“tif”，“aipng”等

vector<chromo> chromoDataArray 输入，待拼接的数据结构体容器

int newCutIndex 输入，拼接后新生成的染色体的起始编号

chromo& chromoData 输出，染色体拼接结果

返回值 int类型的错误代码，0为程序运行正常，1为异常

注1：当软件调用该模块后，应及时修改原chromo\_data[]中的数据，具体来说是，添加输出参数chromoData所对应的数据行，删除输入参数chromoDataArray里所对应的数据行。

注2：软件人员在调用该模块生成新的染色体后，应将该新染色体放置在软件界面的待判断区，等待由用户自行将其拖动到正确的位置，或等待用户运行识别模块后，重新显示排列图。

5.手动切割模块（module3）

功能：完成手动进行染色体的切割。

函数原型：int ManualSegment(chromo chromoData, vector<chromo>& newChromoDataList, vector<Mat> allLines, Point clickPosi, int segType, int newCutNum)

参数列表：

chromo chromoData 输入，待处理原始染色体结构体数据。该数据需要和chromo\_data中的cimg数据大小保持一致

vector<chromo>& newChromoDataList 输出，处理后染色体结构体容器

vector<Mat> allLines 输入，人工所画线的列表，边缘切割时只有1个元素，骨架切割时每条画线为容器中的一个元素

Point clickPosi 输入，鼠标点击坐标，边缘切割时鼠标点击确定了经切割后所要提取出来的染色体是哪一条

int segType 输入，染色体切割方式，0为边缘切割，1为骨架切割

int newCutNum 输入，分割后新生成的染色体的起始编号

返回值 int类型的错误代码，0为程序运行正常，1为异常

6.十字交叉切割模块（moduleCrossSeg）

功能：完成十字交叉类型染色体的切割。

函数原型：int moduleCrossSeg(chromo chromoData, int newCutNum, vector<vector<chromo>>& chromoDataList)

参数列表：

chromo chromoData 输入，染色体数据结构体

int newCutNum 输入，分割后新生成的染色体的起始编号

vector<vector<chromo>>& chromoDataList 输出，分割结果，其中第一个vector<chromo>中存放的是按对角线切割出的两个染色体；第二个vector<chromo>中存放的是按“)(”切割出的两个染色体；第三个存放的是按“^v”切割出的两个染色体

返回值 int类型的错误代码，0为程序运行正常，1为异常

7.识别模块（module4）

功能：对输入的多条单条染色体数据行进行其染色体编号识别。

函数原型：int moduleRecognition(vector<chromo>&chromoData,

Mat &sortedKaryoImg, String model\_path)

参数列表：

vector<chromo>& chromoData 输入输出，待识别的数据结构体容器

Mat &sortedKaryoImg 输入输出，核型图画布

String model\_path 输入，模型文件名

返回值 0

使用方法：

//定义模型路径：更换模型只需要改一下对应模型路径即可

String modelPath = "model-016-0.0601.pb";

//定义空白画布，大小可以设置很大，

//模块内部自动根据识别情况调整输出图片大小

Mat sortedKaryoImg(4000, 4000, CV\_8UC3, Scalar(255, 255, 255));

//执行识别模块

//识别模块会将每一个染色体的类别，概率写入结构体

//并将sortedKaryoImg输出为染色体排列图

int errorCodeModule4 = moduleRecognition(chromoVec, sortedKaryoImg, modelPath);

//保存染色体排列图

imwrite("result1.jpg", sortedKaryoImg);

注：模块应通过识别后，主动改写所有图像类型值为single的数据行所对应的chromo\_num[]数据和chromo\_upright数据。

三、代码修改与库更新日志

1226

修改了module1和module0的接口，修改了module1的去噪部分，针对去核做了相应修改

1204

修改了module1自动分割模块，针对十字交叉型染色体尽可能按照对角线的形式切割；

修改了moduleCrossSeg模块中的函数，不影响使用和切割效果

1126

工程中的源代码保存在code文件夹中，包括moduleOpti优化模块、module0打分模块、module1自动分割模块

module2手动拼接模块、module3手动分割模块、moduleCrossSeg十字分割模块。

opencv版本为3.4.0

dll包括release版本和debug版本

1122

1.修复了优化模块中多次执行结果不同的问题

2.加入了自动分割的异常处理

1115

1.对Module优化模块进行了转码；效果有较好提升

2.修复了自动分割和打分模块的bug

1108

解决了模块0打分针对AIpng报错的问题

1107

模块3多骨架分割的部分进行了较大改进

1106

1.模块0和模块1的预处理部分进行了优化

1105

1.十字交叉分割问题修复bug

1102

模块3手动分割增加了多骨架分割的部分

接口函数进行了相应修改

int ManualSegment(chromo chromoData, vector<chromo>& newChromoDataList, vector<Mat> allLines, Point clickPosi, int segType, int newCutNum);

各参数定义见文件最下面部分

1031

模块1自动分割修复了最后将cluster分割后mask显示不全的bug

1030

十字交叉分割问题修复

1.修复cImgType的问题

2.修复部分染色体切割后没有旋转的问题

3.更新正确坐标

1029

1.打分模块重新拟合

2.十字交叉分割测试完毕