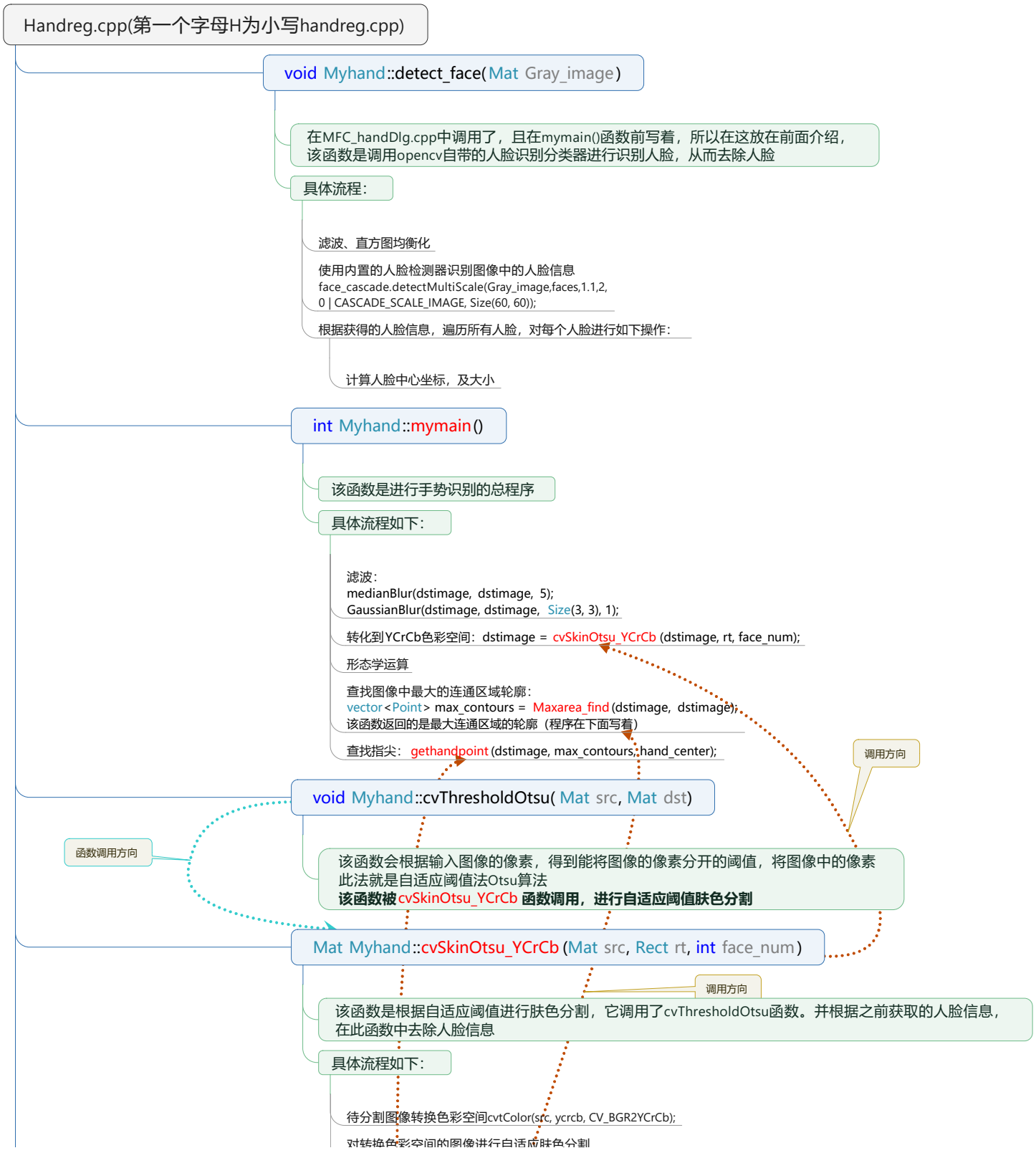
Handreg.cpp(第一个字母H为小写handreg.cpp)
void Myhand::detect_face(Mat Gray_image)
在MFC_handDlg.cpp中调用了，且在mymain()函数前写着，所以在这放在前面介绍，
该函数是调用opencv自带的人脸识别分类器进行识别人脸，从而去除人脸
具体流程：
滤波、直方图均衡化
使用内置的人脸检测器识别图像中的人脸信息
face_cascade.detectMultiScale(Gray_image,faces,1.1,2,
0 | CASCADE_SCALE_IMAGE, Size(60, 60));
根据获得的人脸信息，遍历所有人脸，对每个人脸进行如下操作：
计算人脸中心坐标，及大小
int Myhand::mymain()
该函数是进行手势识别的总程序
具体流程如下：
滤波：
medianBlur(dstimage, dstimage, 5);
GaussianBlur(dstimage, dstimage, Size(3, 3), 1);
转化到YCrCb色彩空间：dstimage = cvSkinOtsu_YCrCb (dstimage, rt, face_num);
形态学运算
查找图像中最大的连通区域轮廓：
vector<Point> max_contours = Maxarea_find(dstimage, dstimage);
该函数返回的是最大连通区域的轮廓（程序在下面写着）
查找指尖：gethandpoint(dstimage, max_contours, hand_center);
调用方向
void Myhand::cvThresholdOtsu(Mat src, Mat dst)
函数调用方向
该函数会根据输入图像的像素，得到能将图像的像素分开的阈值，将图像中的像素
此法就是自适应阈值法Otsu算法
该函数被cvSkinOtsu_YCrCb 函数调用，进行自适应阈值肤色分割
Mat Myhand::cvSkinOtsu_YCrCb (Mat src, Rect rt, int face_num)
调用方向
该函数是根据自适应阈值进行肤色分割，它调用了cvThresholdOtsu函数。并根据之前获取的人脸信息，
在此函数中去除人脸信息
具体流程如下：
待分割图像转换色彩空间cvtColor(src, ycrcb, CV_BGR2YCrCb);
对转换色彩空间的图像进行自适应肤色分割


cvThresholdOtsu(temp_channals.at(1),temp_channals.at(1));
对分割后的图像去除人脸信息，即让人脸位置的像素值为0
返回最终的处理结果图像，此时只剩下手部位置，和一些小的噪声点
调用方向
vector <Point > Myhand::Maxarea_find (Mat src, Mat dst)
获取最大的连通区域，并返回最大连通区域的轮廓。
所以他是用来计算分割后只含有手势的二值图现中的最大连通区域，即
cvSkinOtsu_YCrCb()函数的分割结果图
具体流程：
利用opencv里的findContours函数，找到图中所以的连通区域
findContours(src,contours, RETR_EXTERNAL ,CHAIN_APPROX_NONE);
遍历所有的连通区域，获取面积最大的连通区域
使用opencv中的contourArea()函数计算每一个连通区域面积
area_now =
contourArea(contours[i]);
并取得最大连通区域
将上面得到的最大连通区域的轮廓绘制到临时图片上
drawContours(Out_Contours_Max, final_cont, -1, Scalar(255));
同时计算出最大面积连通区域的质心（即手势的质心）
返回最大轮廓return max_contours;
void Myhand::gethandpoint (Mat src, vector <Point > max_contours, Point center)
该函数根据手势轮廓来计算指尖的位置
具体流程如下：
循环遍历轮廓上的每个点，直到所有点被遍历
计算轮廓上第一个到质心最远的点为第一个指尖first_point
= couPoint[spos];
之后同样根据轮廓上的点到质心位置最大的点作为指尖，直到轮廓上的点被遍历完
上面找到的指尖，可能有些并不是指尖，
所以要进行二次判断(进行角度判断的前提是已经找到两个及两个以上的指尖)，
判断依据是两指尖与手势质心连线的夹角不应该大于120度
double angle=getAngle_center(first_point, couPoint[spos]);
if (angle > 120)
{
count--;
fingerTips_single = singer_before;
}
最终在是指尖的位置上画个小圆圈circle(src, fingerTips_single, 15,
Scalar(255), 1, 8, 0);
调用方向
double Myhand::getAngle_center(Point first_p, Point second_p)
计算图片中的两点到手部质心的夹角函数，被gethandpoint函数调用