1.提取手部

肤色分布阈值界定法

将照片转为YCbCr格式，读取每个像素的Cb、Cr值，设定阈值区间Cb=[110,130]，Cr=[140,175]，在此区间内的像素点判断为肤色，将提取出来的部分进行形态学滤波处理。

void cvtColor

(

InputArray src, src：输入图像

OutputArray dst, dst：输出YCbCr格式图像

int code, code：转换形式，此处为CV\_RGB2YCrCb

int dstCn=0 ); dstCn：dst图像的波段数，默认是0

void split(const Mat& src, vector<Mat\_<\_Tp> >& mv)

src：输入图像

mv：保存各个通道，每个通道存放到一个mat 中

运用迭代器访问每个像素的YCbCr值：

MatIterator\_<uchar> it\_Cb = planes[1].begin<uchar>(),

it\_Cb\_end = planes[1].end<uchar>();

MatIterator\_<uchar> it\_Cr = planes[2].begin<uchar>();

MatIterator\_<uchar> it\_skin = mask.begin<uchar>();

for (; it\_Cb != it\_Cb\_end; ++it\_Cr, ++it\_Cb, ++it\_skin)

{

//根据肤色阈值获得mask（Cb=[110,130]，Cr=[140,175]）

if (\*it\_Cb >= 110 && \*it\_Cb <= 130 && \*it\_Cr >= 140 && \*it\_Cr <= 175)

\*it\_skin = 255;

else

\*it\_skin = 0;

}

形态学滤波：腐蚀，膨胀，中值滤波

void erode

(

const Mat& src,

Mat& dst,

const Mat& element,

Point anchor=Point(-1,-1),

int iterations=1,

int borderType=BORDER\_CONSTANT,

const Scalar& borderValue=morphologyDefaultBorderValue() );

src：输入图像

dst：输出图像

element:腐蚀操作的内核，此处为Mat(5, 5, CV\_8UC1)

anchor:默认为Point(-1,-1),内核中心点

iterations:腐蚀次数。省略时为默认值1。

borderType:推断边缘类型，具体参见borderInterpolate函数。默认为BORDER\_DEFAULT，省略时为默认值。

borderValue:边缘值，具体可参见createMorphoogyFilter函数。可省略

void dilate

(

const Mat& src,

Mat& dst,

const Mat& element,

Point anchor=Point(-1,-1),

int iterations=1,

int borderType=BORDER\_CONSTANT,

const Scalar& borderValue=morphologyDefaultBorderValue() );

参数一致

中值滤波：

void medianBlur

(

const Mat& src, src：输入图像

Mat& dst, dst：输出图像

int ksize ); ksize：滤波器的大小，此处为Size(5,5)

1. 分割指尖
2. 画出手部外轮廓，计算图像质心（手心），在外轮廓上找出五个离质心最远的点（五个指尖），分割出指尖的图像

画出手部外轮廓

1. 调用findContours函数找出所有轮廓：

函数原型：void findContours//提取轮廓，用于提取图像的轮廓

(

InputOutputArray image,

OutputArrayOfArrays contours,

OutputArray hierarchy,

int mode,

int method,

Point offset = Point()

)

·第一个参数：image，要处理的图像。

·第二个参数：contours，是一个向量，里面每个元素保存了一组由连续的Point点构成的点的集合的向量，每一组Point点集就是一个轮廓。

·第三个参数：hierarchy，可选的输出向量，包含图像的拓扑信息。

·第四个参数：int型的mode，定义轮廓的检索模式：

这里选用：CV\_RETR\_TREE， 检测所有轮廓，所有轮廓建立一个等级树结构。外层轮廓包含内层轮廓，内层轮廓还可以继续包含内嵌轮廓。

·第五个参数：int型的method，定义轮廓的近似方法：

这里选用:CV\_CHAIN\_APPROX\_SIMPLE 仅保存轮廓的拐点信息，把所有轮廓拐点处的点保存入contours向量内，拐点与拐点之间直线段上的信息点不予保留

·第六个参数：Point偏移量。

这里偏移量为0即可

1. 找出覆盖面积最大的轮廓的编号

for (int i = 0; i < contours.size(); i++)

{

area = contourArea(Mat(contours[i]));

if (area > maxArea)

{

maxArea = area;

index = i;

}

}

(3)调用drawContours函数绘制最大轮廓（即手部外轮廓）：

函数原型：

void drawContours//绘制轮廓，用于绘制找到的图像轮廓

(

 InputOutputArray image,

 InputArrayOfArrays contours,

 int contourIdx,

 const Scalar& color,

 int thickness = 1,

 int lineType = 8,

 InputArray hierarchy = noArray(),

 int maxLevel = INT\_MAX,

 Point offset = Point()

)

·第一个参数：image是要绘制轮廓的图像

·第二个参数：contours，所有输入的轮廓

·第三个参数：contourIdx，指定要绘制轮廓的编号，如果是负数，则绘制所有的轮廓。此处编号为上一步得到的index

·第四个参宿：color，绘制轮廓所用的颜色

第五个参数：绘制轮廓的线条的粗细

·其他参数均可以省略，设为默认值。

寻找手心——计算图像质心

矩——简单描述图像特征的一组数据

从图像中计算出来的矩通常描述了图像不同种类的几何特征，如大小、灰度、方向、形状等

对于一幅M\*N的图像，计算其原点矩



为图像坐标（i,j）处的值

设图像质心为（，）

则有，

Void Moments

（

InputArray src, 输入图像

Bool binaryImage=false 默认值为false

）

Point center(moment.m10 / moment.m00, moment.m01 / moment.m00);//质心坐标

寻找五个指尖坐标点

遍历外轮廓所有点，找出离手心最远的五个点，即为五个指尖点

限制条件：

1. 计算最大值保持的点数，如果大于200,那么就认为这个是指尖
2. 排除低于手心的点

分割出指尖（以中指为例）

将指尖坐标点左移20像素，上移10像素，以移动后的点为基准框出一个60\*60的区域

int a = fingerTips[0].x;

int b = fingerTips[0].y;

Rect rect(a - 20, b - 10, 60, 60);

Mat roi = result(rect);

//此处roi是引用的result，对roi进行改动，result也会随之变化

指甲贴图

在上一步切割出的指甲区域内对像素进行赋值操作