

**《计算机图像处理与机器视觉》课程**

**实验报告**

项目名称：机器人收获果实时切割点的提取

专 业：自动化（电气）

参与学生：潘盛琪

指导老师：饶秀勤

提交日期：2019.4.18

浙江大学智能农业装备研究所

二O一三年五月

**一、实验目的和要求**

1. 熟练运用matlab进行综合的图像处理任务
2. 常用的阈值分割处理方法

**二、实验内容和原理**

在草莓、黄瓜、西瓜等果梗细长的果蔬自动收获中，机器人通常切刀等方式切断果梗，实现果实与植株本体的分离。在分离前需要在尽量靠近果实与果梗连接处选择**切割点**。

现以温室立体栽培模式下的小型西瓜为研究对象，该西瓜形状较为匀称，果梗长度大于R/2（R为西瓜半径），其轴线通常沿垂直方向，通过或接近果实形心。据此特征，通常的方法是将切割点限制在采摘点（果实形心）正上方的矩形兴趣区内，如下图a所示，该矩形区域长为R，宽为R/2，其对称中心线通过采摘点，下边缘与采摘点相距R/2。

图a是温室立体栽培模式下的小型西瓜的近红外图像（850 nm），将上述矩形区域作为感兴趣的区域，**通常采用分块定位法来计算矩形兴趣区内的切割点坐标**。

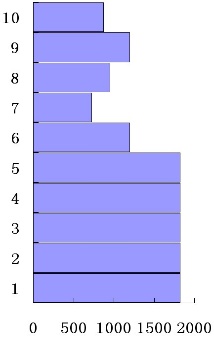
问题：

（1）将小型西瓜的近红外图像进行阈值分割，得到二值化图像（0，255），确定西瓜形心以及矩形区域。

（2）进行行坐标计算。将矩形兴趣区按行等分为10个分块，自下向上对各分块进行编号（如图b所示），计算各分块内白像素点（灰度值为255）的个数Ci（i=1、2、3……10），得到其统计直方图（如图c所示）。按编号顺序，将首次出现黑白相间（即Ci ≠ R2 /20）的分块定义为过渡块（如图b所示第6分块），将过渡块的下一分块定义为切割点所在块（如图b所示第7分块），其水平中心线即为采摘点所在目标行。



1



R

R/2

R/2

9

8

7

6

5

4

3

2

7

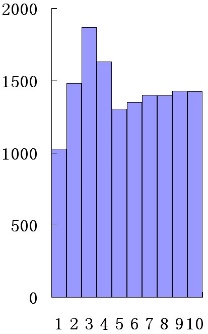
8

9

10

**(a)** **(b)** **(c)**

（3）进行列坐标计算。仿照行坐标计算方法，将矩形兴趣区按列等分为10个分块，统计各分块内白像素点个数（如图d、e所示）。将所含白像素最多的分块定义为果梗所在块，其竖直中心线为采摘点所在目标列。



**(d) (e)**

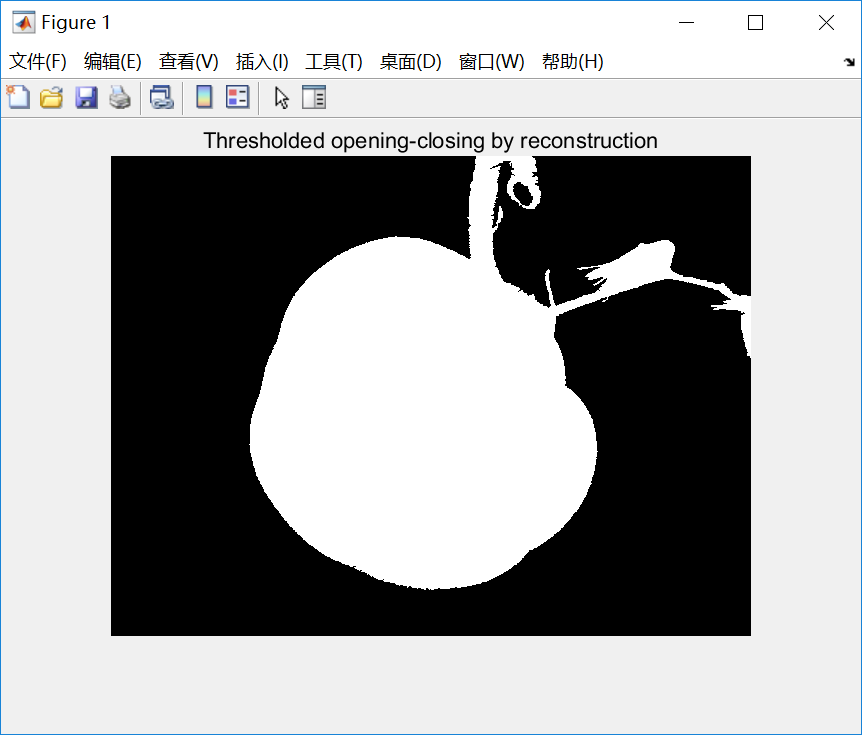
（4）确定切割点.确定目标行、列的交点为切割点，将求取结果表示在原始图像中。



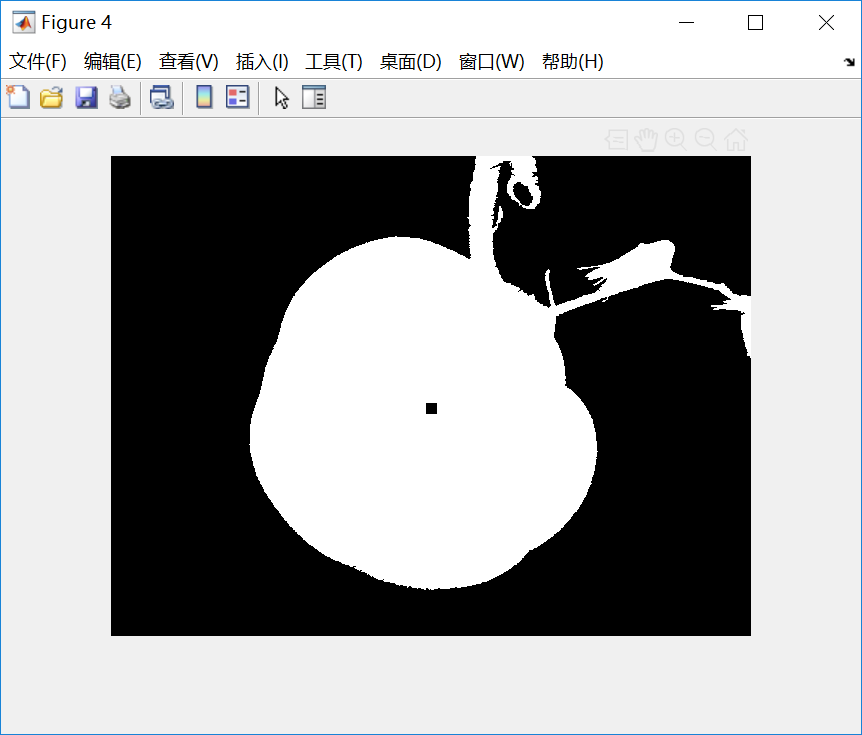
**(f)**

**三、实验结果**

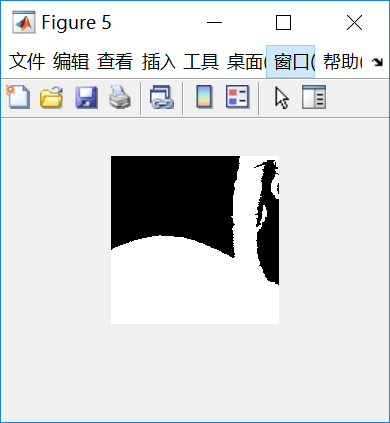
1. 阈值分割



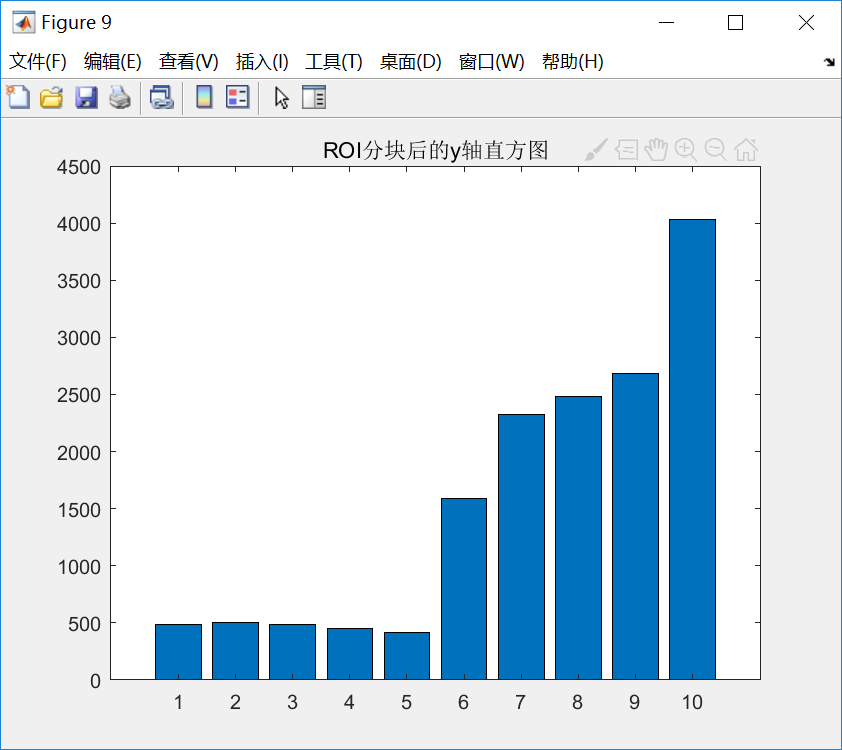
1. 中心点计算



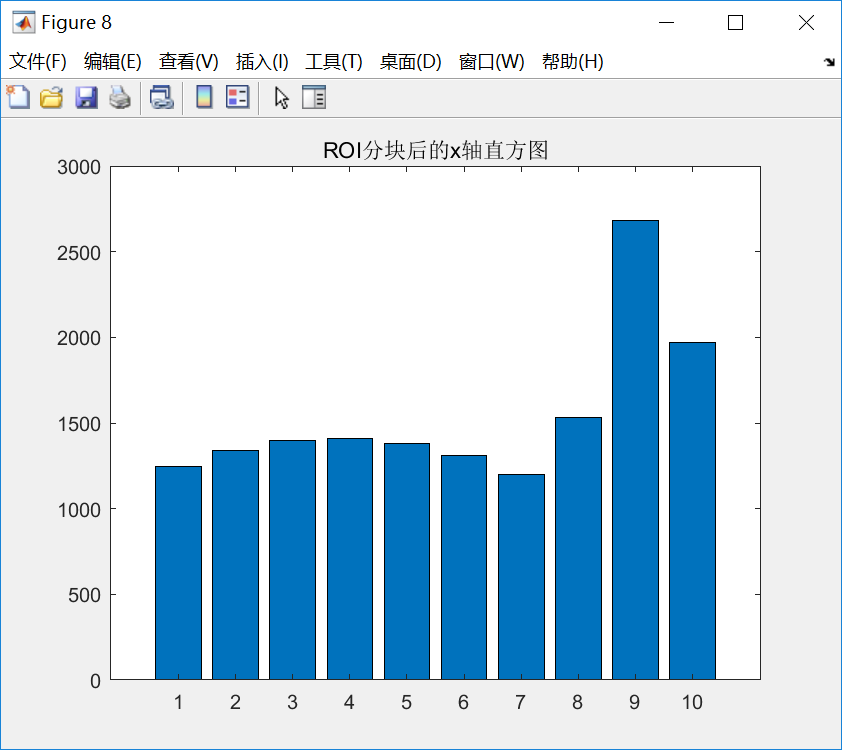
1. ROI提取



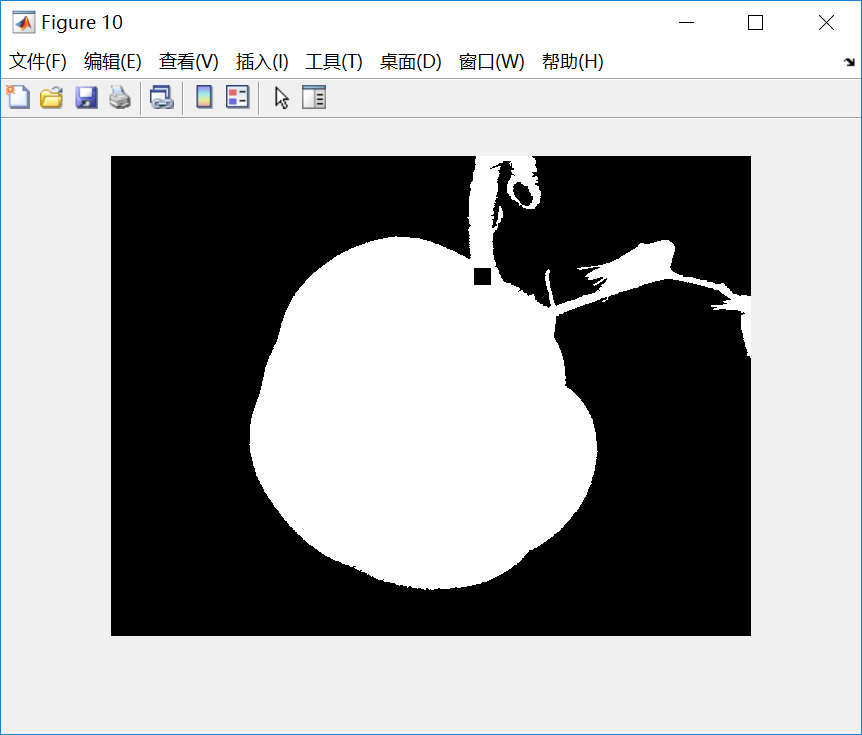
1. 行坐标计算



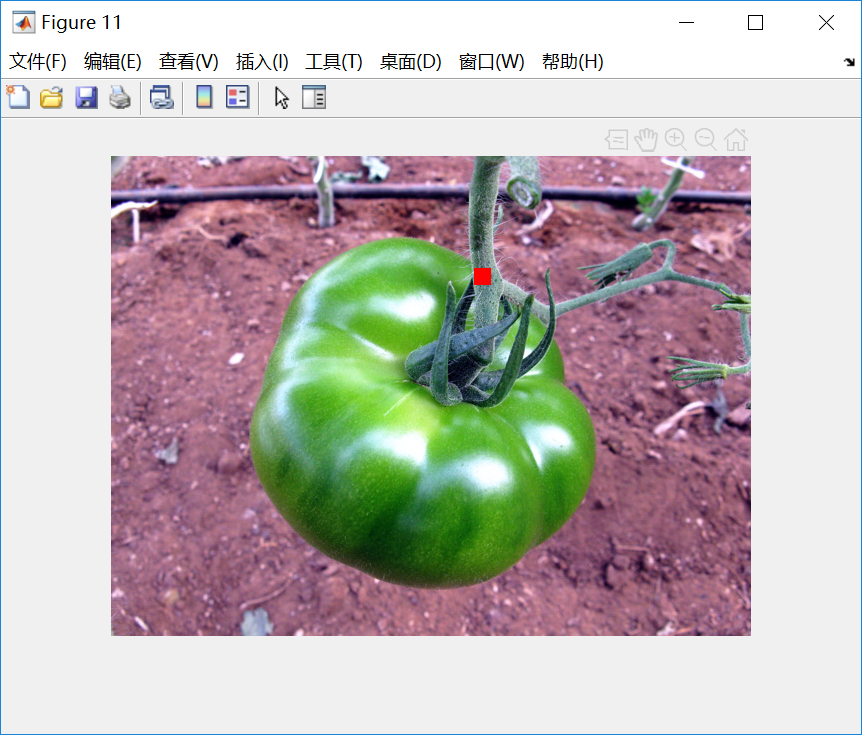
1. 列坐标计算



1. 在二值图上显示切割点



1. 确定切割点



**四、讨论**

这一实验的重点也是最大的难点在于阈值分割，也就是如何能最好地将图片二值化，把所需要的水果提取出来。由于水果的生长环境导致图片背景较为杂乱，直接利用灰度图像进行阈值分割不太能行得通，效果非常糟糕。而可以看到水果切割这一类待图像有另一个特点，就是水果的颜色往往与背景相差较大。因此将图片转到HSV色彩空间进行阈值分割，效果会好很多。

当然，对于一部分水果，比如黄瓜，它的颜色与叶子很相近，对于这类水果就不能采用HSV色彩空间进行阈值分割了。

**五、心得体会与课程建议**

在这门课上学到了一些图像处理的基本算法。其中令我影响很深的有灰度变换、阈值分割、伪彩色变换等。但课程内容毕竟有限，这学期学到的知识需要完整地融合起来，并结合一些其他的处理技术，才能发挥实际的用处。

最开始学的时候我也不是很明白阈值分割究竟有什么意义，似乎阈值分割以后也不知道后续的处理该怎么进行。但是经过这次大作业是实际操作以后，完整地感受到了图像处理的整个过程，我也进一步了解了阈值分割的用处。上完这门课以后，我深深地感受到数字图像处理是一个成体系的完整理论，要说一个单一的算法具体有什么用，可能真的很难说上来。但是将所学的一些基础的算法结合起来，就能写出一些比较有实际应用价值的程序来。

就像这次的大程序，能够自动识别果实切割点，在实际生产中非常实用，但把它拆分开，也不过是阈值分割，ROI提取，横向和纵向的投影这么几块罢了。

这次课程对我而言还有一大遗憾，刚开始上课的时候没有意识到图像处理算法间的关联性，所有算法实现的程序都是直接用脚本文件写的，没有封装成函数，这也给最后的大作业的完成带来了不少麻烦。

我对于课程也有一些小建议，我记得刚开始老师没有说过建了班群，后来问同学要课件才知道的，可能是因为选这门课的大部分同学都是一个专业的直接建好了群，我是外专业的，完全不知道。建议老师可以在课程开始的时候建一个群把大家拉进去。

附录一：源程序及程序说明

程序说明见注释

clc;

clear;

close all;

% 初始化

rgbimg = imread('test.jpg');

% figure;

imshow(rgbimg);

r = rgbimg(:, :, 1);

g = rgbimg(:, :, 2);

b = rgbimg(:, :, 3);

grayimg = rgb2gray(rgbimg);

[row, col] = size(grayimg);%行数和列数

hsvimg = rgb2hsv(rgbimg);

H = hsvimg(:, :, 1);

S = hsvimg(:, :, 2);

V = hsvimg(:, :, 3);

% 初始化

I = H;%提取H色彩空间

% hy = fspecial('sobel');%sobel算子

% hx = hy';

% Iy = imfilter(double(I), hy, 'replicate');%滤波求y方向边缘

% Ix = imfilter(double(I), hx, 'replicate');%滤波求x方向边缘

% gradmag = sqrt(Ix.^2 + Iy.^2);%求摸

% 阈值分割

% 分别对前景和背景进行标记：使用形态学重建技术对前景对象进行标记，首先使用开操作，开操作之后可以去掉一些很小的目标。

se = strel('disk', 40);%圆形结构元素

% 腐蚀

Io = imopen(I, se);%形态学开操作

Ie = imerode(I, se);%对图像进行腐蚀

Iobr = imreconstruct(Ie, I);%形态学重建

% 膨胀

Ioc = imclose(Io, se);%形态学关操作

Iobrd = imdilate(Iobr, se);%对图像进行膨胀

Iobrcbr = imreconstruct(imcomplement(Iobrd), imcomplement(Iobr));%形态学重建

Iobrcbr = imcomplement(Iobrcbr);%图像求反

bw = imbinarize(Iobrcbr, graythresh(Iobrcbr));%转化为二值图像

bw = imcomplement(bw);%图像求反

% figure;

imshow(bw) %显示二值图像

title('Thresholded opening-closing by reconstruction')

% 阈值分割

% 求重心

XProject = sum(bw);%二值图在x轴的投影

YProject = sum(bw, 2);%二值图在y轴的投影

XProjectSum = XProject;

YProjectSum = YProject;

for i = 2 : col

XProjectSum(i) = XProjectSum(i) + XProjectSum(i-1);

end

for i = 2 : row

YProjectSum(i) = YProjectSum(i) + YProjectSum(i-1);

end

%画图

XAxis = 1 : col;

YAxis = 1 : row;

figure;

bar(XAxis, XProjectSum);

title('x轴累积直方图');

figure;

bar(YAxis, YProjectSum);

title('y轴累积直方图');

% 分别求x轴y轴重心

XHalf = sum(XProject)/2;%求x轴一半

YHalf = sum(YProject)/2;%求y轴一半

for i = 2 : col

if XHalf > XProjectSum(i-1) && XHalf < XProjectSum(i)

break;

end

end

XCenter = i;%x轴中心

for i = 2 : row

if YHalf > YProjectSum(i-1) && YHalf < YProjectSum(i)

break;

end

end

YCenter = i;%y轴中心

bwCenter = bw;

bwCenter(YCenter - 5 : YCenter + 5, XCenter - 5 : XCenter + 5) = 0;

figure;

imshow(bwCenter);

% 求重心

% 求半径

for i = XCenter : col

if bw(YCenter, i) == 0

break;

end

end

RXR = i - XCenter;

for i = XCenter : -1 : 1

if bw(YCenter, i) == 0

break;

end

end

RXL = XCenter - i;

R = (RXR + RXL) / 2;

% 求半径

% 根据半径框出ROI

RecLDown = [floor(YCenter - R / 2), floor(XCenter - R / 2)];%矩形左下角的点为RecLDown

RecRDown = [floor(YCenter - R / 2), floor(XCenter + R / 2)];%矩形右下角的点为RecRDown

RecLUp = [floor(YCenter - R / 2 - R), floor(XCenter - R / 2)];%矩形左上角的点为RECLUp

RecRUp = [floor(YCenter - R / 2 - R), floor(XCenter + R / 2)];%矩形右上角的点为RecRUp

ROI = bw(RecLUp(1) : RecRDown(1), RecLDown(2) : RecRUp(2));

figure;

imshow(ROI);

% 根据半径框出ROI

% 对ROI进行处理

[ROIrow, ROIcol] = size(ROI);

ROIXProject = sum(ROI);%二值图在x轴的投影

ROIYProject = sum(ROI, 2);%二值图在y轴的投影

ROIXProjectSum = ROIXProject;

ROIYProjectSum = ROIYProject;

for i = 2 : ROIcol

ROIXProjectSum(i) = ROIXProjectSum(i) + ROIXProjectSum(i-1);

end

for i = 2 : ROIrow

ROIYProjectSum(i) = ROIYProjectSum(i) + ROIYProjectSum(i-1);

end

% 画图

ROIXAxis = 1 : ROIcol;

ROIYAxis = 1 : ROIrow;

figure;

bar(ROIXAxis, ROIXProject);

title('ROI的x轴直方图');

figure;

bar(ROIYAxis, ROIYProject);

title('ROI的y轴直方图');

% 分块

XStep = floor(ROIcol / 10);%x轴每块的大小

YStep = floor(ROIrow / 10);%y轴每块的大小

%初始化xy轴每一块的值

XBlock = zeros(1, 10);

YBlock = zeros(1, 10);

for i = 1 : 10

if i == 1

XBlock(1) = ROIXProjectSum(XStep \* i);

YBlock(1) = ROIYProjectSum(YStep \* i);

continue;

end

if i == 10

XBlock(10) = ROIXProjectSum(ROIcol) - ROIXProjectSum(XStep \* 9);

YBlock(10) = ROIYProjectSum(ROIrow) - ROIYProjectSum(YStep \* 9);

break;

end

XBlock(i) = ROIXProjectSum(XStep \* i) - ROIXProjectSum(XStep \* (i - 1));%计算x轴每一块的值

YBlock(i) = ROIYProjectSum(YStep \* i) - ROIYProjectSum(YStep \* (i - 1));%计算y轴每一块的值

end

% 画图

ROIXAxis = 1 : 10;

ROIYAxis = 1 : 10;

figure;

bar(ROIXAxis, XBlock);

title('ROI分块后的x轴直方图');

figure;

bar(ROIYAxis, YBlock);

title('ROI分块后的y轴直方图');

% 确定最终的切割点

[ROIXMax, ROIXPlace] = max(XBlock);%确定切割位置的x轴坐标

i = 10;%初始化i

if YBlock(10) == ROIcol \* (ROIrow - 9 \* YStep)

for i = 9 : -1 : 1

if YBlock(i) ~= ROIcol \* YStep

break;

end

end

end

ROIYPlace = i - 1;

% 对ROI进行处理

% 转换到原坐标下并作图

XPlaceL = (ROIXPlace - 1) \* XStep + RecLDown(2) - 1;

XPlaceR = ROIXPlace \* XStep + RecLDown(2) - 1;

YPlaceU = (ROIYPlace - 1) \* YStep + RecLUp(1) - 1;

YPlaceD = ROIYPlace \* YStep + RecLUp(1) - 1;

FinalImg = bw;

FinalImg(YPlaceU : YPlaceD, XPlaceL : XPlaceR) = 0;

figure;

imshow(FinalImg);

figure;

Finalrgbimg = rgbimg;

Finalrgbimg(YPlaceU : YPlaceD, XPlaceL : XPlaceR, 1) = 256;

Finalrgbimg(YPlaceU : YPlaceD, XPlaceL : XPlaceR, 2 : 3) = 1;

imshow(Finalrgbimg);

% 转换到原坐标下并作图

% 显示去除背景后的原图

% bww=im2uint8(bw)/255;

% rgbimg(:, :, 1)=bww.\*rgbimg(:, :, 1);

% rgbimg(:, :, 2)=bww.\*rgbimg(:, :, 2);

% rgbimg(:, :, 3)=bww.\*rgbimg(:, :, 3);

% figure;

% imshow(rgbimg);

参考文献：

无