**实验报告**

装 订 线

专业：\_自动化（电气）\_

姓名：\_\_\_\_\_潘盛琪\_\_\_\_\_

学号：\_\_\_3170105737\_\_\_

日期：\_\_\_\_\_4.11\_\_\_\_

地点：**生工食品学院机房**

课程名称： **计算机图像处理与机器视觉** 指导老师： **饶秀勤**  成绩：

实验名称：  **最小外接矩形的求取** 实验类型： **设计型**

**一、实验目的和要求**

求二值图像的最小外接矩形

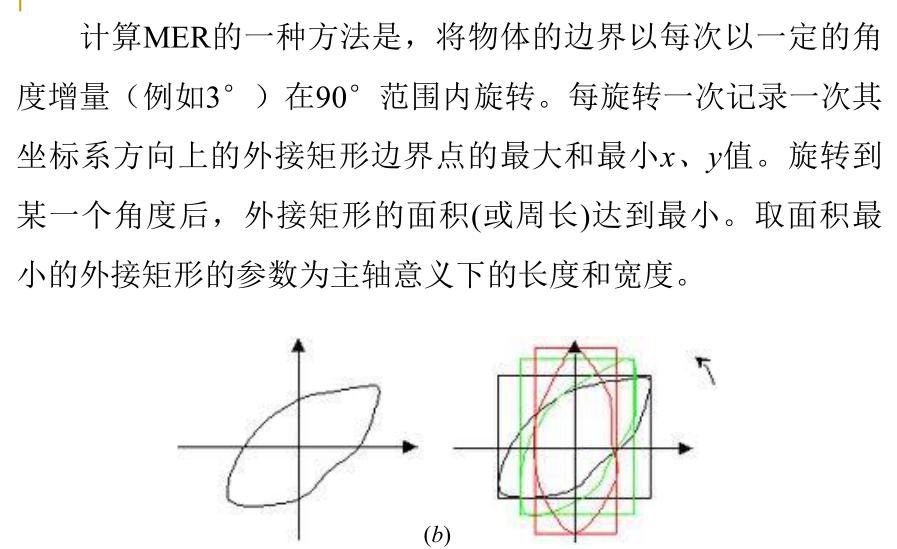
**二、计算机配置与软件处理平台**

硬件：



软件：基于matlab

**三、算法描述**



**四、结果与讨论**

**4.1 输入图像**

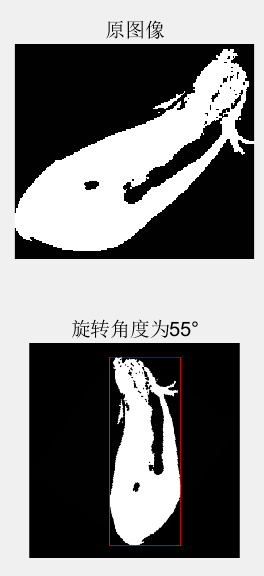


**4.2 输出图像**

旋转角度从0°~90°，步长为10°，求得的MER



旋转角度从0°~90°，步长为1°，求得的MER



**4.3 讨论**

从输入输出图像中可以看到，通过旋转图像法求最小外接矩形，实验效果较好。通过改变旋转角度的步长可以使MER的计算变得更为精确，但也会使得程序运行时间增加。此外通过这一方法求MER，阈值分割的准确性也很重要，本次实验中选区的图像背景较为简洁，因此阈值分割效果也比较好。但在实际应用中，背景通常没有这么间接，因此在进行阈值分割时应当进行形态学的腐蚀与膨胀，以确保阈值分割的效果。一旦阈值分割出现一点点小瑕疵，整个程序都会出问题。

**五、结论**

原始图像在旋转55°之后的坐标系内，有主轴下的MER

**六、源程序**

%%%寻找图像最小外接矩形

%%%先用迭代式阈值求法将图像变为二值图

img = imread('test0.png');

img = rgb2gray(img);

Th = mean(img( : )); %将灰度均值设为初始阈值

newTh = 0;

i = 1;

while (Th - newTh) > 1

pic1 = img;

pic1(pic1 > Th) = 0; %大于阈值置零方便求均值

miu1 = mean(pic1(:)); %小于阈值的像素的灰度均值

pic2 = img;

pic2(pic2 < Th) = 0; %小于阈值置零方便求均值

miu2 = mean(pic2( : )); %大于阈值的像素的灰度均值

if i ~= 1

Th = newTh;

end

i = i + 1;

newTh = (miu1 + miu2) / 2;

end

imgTh = img;

imgTh(imgTh < Th) = 1;

imgTh(imgTh > Th) = 256;

imgTh = 256 - imgTh;

%%%先用迭代式阈值求法将图像变为二值图

%%%旋转图像求其MER

for angle = 0 : 10 : 90

imgRotated = double(imrotate(imgTh,angle,'bicubic','loose')); %求旋转后的图像

%imshow(uint8(imgRotated));

[row, col] = size(imgRotated);

%%%判断最小外接矩形的边界

for i = 1 : row

if sum(imgRotated(i, :)) > col

break;

end

end

yMinTest = i;

for i = row : -1 : 1

if sum(imgRotated(i, :)) > col

break;

end

end

yMaxTest = i;

for i = 1 : col

if sum(imgRotated(:, i)) > row

break;

end

end

xMinTest = i;

for i = col : -1 : 1

if sum(imgRotated(:, i)) > row

break;

end

end

xMaxTest = i;

%%%判断最小外接矩形的边界

%%%计算面积

XLU = xMinTest \* cosd(angle) - yMinTest \* sind(angle);

YLU = xMinTest \* sind(angle) + yMinTest \* cosd(angle);

XLD = xMinTest \* cosd(angle) - yMaxTest \* sind(angle);

YLD = xMinTest \* sind(angle) + yMaxTest \* cosd(angle);

XRU = xMaxTest \* cosd(angle) - yMinTest \* sind(angle);

YRU = xMaxTest \* sind(angle) + yMinTest \* cosd(angle);

XRD = xMaxTest \* cosd(angle) - yMaxTest \* sind(angle);

YRD = xMaxTest \* sind(angle) + yMaxTest \* cosd(angle);

l1 = sqrt((XLU - XRU) ^ 2 + (YLU - YRU) ^ 2);

l2 = sqrt((XLU - XLD) ^ 2 + (YLU - YLD) ^ 2);

nowSize = l1 \* l2;

%%%保存当前求得的MER

if angle == 0 || nowSize < typicalSize

xMin = xMinTest;

yMin = yMinTest;

xMax = xMaxTest;

yMax = yMaxTest;

typicalSize = nowSize;

typicalAngle = angle;

typicalImg = imgRotated;

end

%%%保存当前求得的MER

lastSize = nowSize;

end

%%%重现

XLU = xMin \* cosd(typicalAngle) - yMin \* sind(typicalAngle);

YLU = xMin \* sind(typicalAngle) + yMin \* cosd(typicalAngle);

XLD = xMin \* cosd(typicalAngle) - yMax \* sind(typicalAngle);

YLD = xMin \* sind(typicalAngle) + yMax \* cosd(typicalAngle);

XRU = xMax \* cosd(typicalAngle) - yMin \* sind(typicalAngle);

YRU = xMax \* sind(typicalAngle) + yMin \* cosd(typicalAngle);

XRD = xMax \* cosd(typicalAngle) - yMax \* sind(typicalAngle);

YRD = xMax \* sind(typicalAngle) + yMax \* cosd(typicalAngle);

%%%重现

out1 = imrotate(imgTh,typicalAngle,'bicubic','loose');

rectx = [xMin, xMax, xMax, xMin, xMin];

recty = [yMax, yMax, yMin, yMin, yMax];

subplot(2,1,2);

imshow(out1);

title(['旋转角度为',num2str(typicalAngle),'°']);

line(rectx(:),recty(:),'color','r');

subplot(2,1,1);

imshow(uint8(imgTh));

title('原图像');

% out2 = imrotate(Figure1,-typicalAngle,'bicubic','loose');

% imshow(out2);

% rectx = [XLU, XLD, XRD, XRU, XLU];

% recty = [YLU, YLD, YRD, YRU, YLU];

% imshow(uint8(imgTh));

% line(rectx(:),recty(:),'color','r');

%%%旋转图像求其MER