Robust Parameter-free Coin Detector

杜宇 11121043 *answeror@gmail.com*
陈建青 21121215 *qingmarch@gmail.com*

1 目录

2 Environment & Library

1. Win7 x64
2. VS2010 SP1
3. CMake 2.8.5
4. OpenCV 2.3.1
5. Qt 4.7.2
6. Boost 1.47.0
7. Qwt 6.0.1
8. CML 1.0.3 (2D/3D 数学库)(http://cmldev.net/)
9. OvenToBoost (迭代器区间库)(https://github.com/faithandbrave/OvenToBoost)

3 Functionality

../common/*.*pp	各次作业公用的窗口部件，辅助类，函数等.
main.cpp	初始化 qt 环境，创建主窗口，启动 qt 消息循环.
mainwindow.*pp	qt 主窗口.
algo.*pp	边缘检测和找轮廓线，以及一些用霍夫变换做的实验代码.
fit_circle.*pp	线性最小二乘拟合圆.
ed.*pp	EDCircles 的实现和改进.
others	各种实验代码

4 Snapshot

1. 单击主窗口 Open，在弹出图片窗口单击 load 选择输入图片，再单击主窗口 Next. (图片窗口中的 save 用于保存图像; 1:1 用于查看原比例；下拉框用于调节尺寸变化模式)

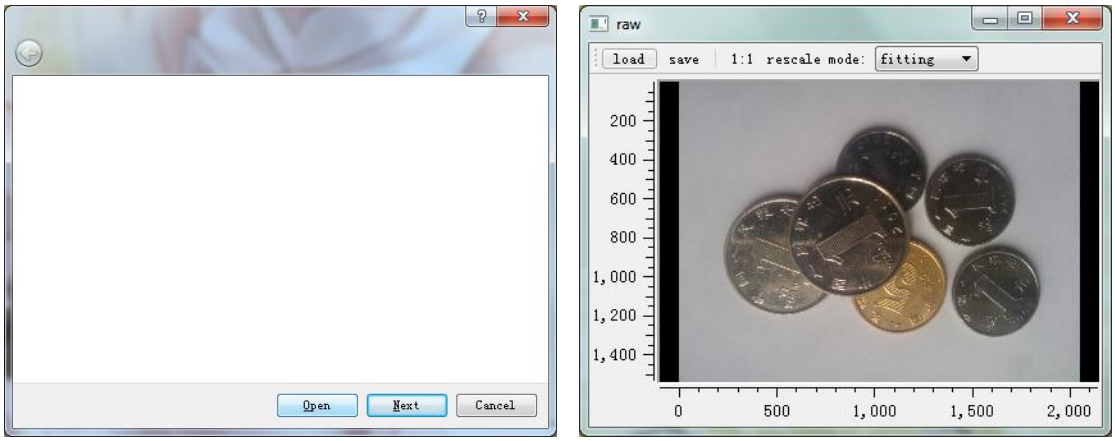

Figure 4-1 input

2. 在主窗口中调整可见圆阈值(绝大多数情况下 0.5 即可)，单击 detect，弹出三个图片窗口．其中 contour approximation 是对于边缘检测结果(白色)的轮廓线的多边形近似(蓝色); detected circles 是检测出的圆; overlapped 是以原图为背景的检测结果(红色).

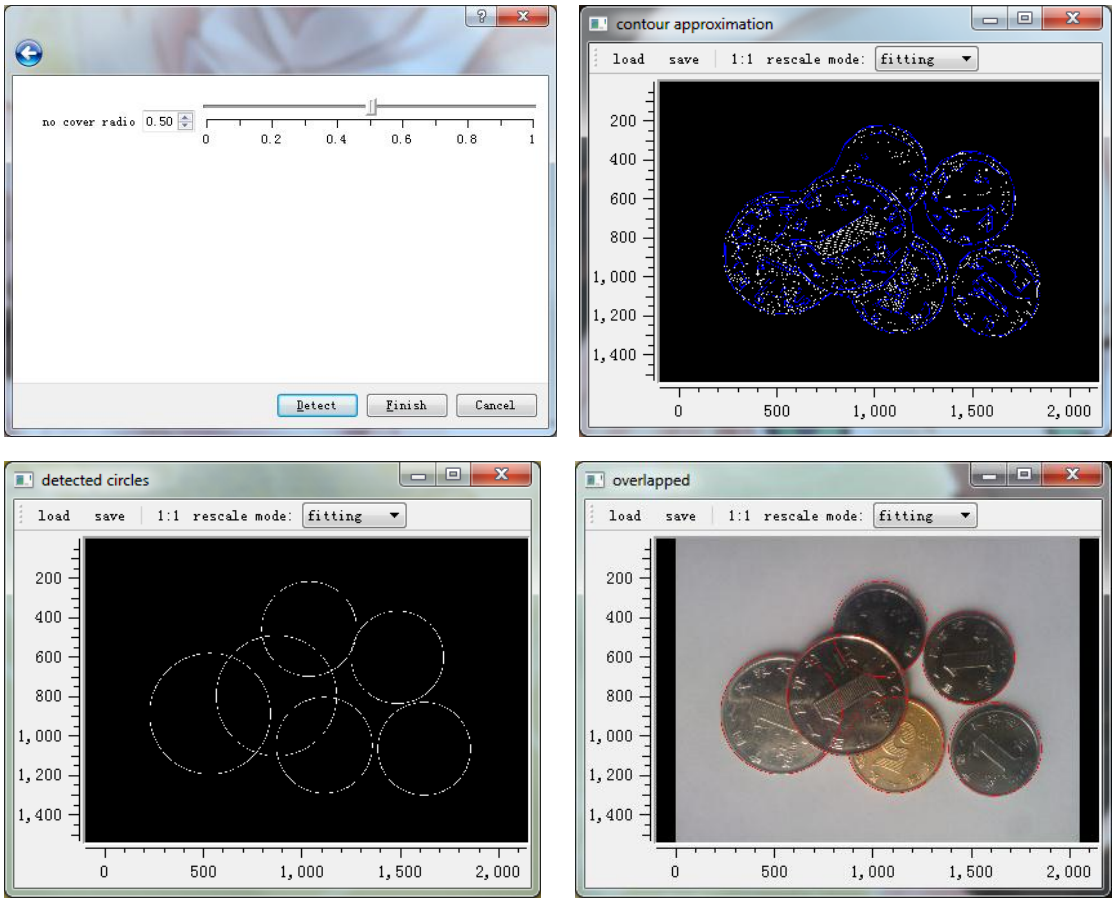

Figure 4-2 detect

5 Introduction

参考了{ http://ceng.anadolu.edu.tr/cv/EDCircles/ }上 Cuneyt Akinlar 的检测圆算法．该算法的好处正如页面标题" EDCircles: A Real-time Parameter-free Circle Detector"所讲的:

1. 快;
2. 对参数不敏感.

其网页上有 demo，对尺度，颜色，光照变化非常鲁棒，效果比 opencv 中自带的霍夫变换好很多．我们的实现比较粗糙，比 opencv 的霍夫变换略慢，但检测效果同样比霍夫变换好很多.

该文章在我们写这个程序的时候还没发表，网页上只有对算法的简略描述(已经发表在 ICASSP 2012)．我们按照自己的理解进行实现，并加入了几个他们没有提到的优化．他们

的边缘检测部分使用了他们 10 年 ICPR 上的文章[1]中的算法. 由于时间关系, 我们没有实现该边缘检测算法, 用了一个简单的近似, 但是效果很好.

6 Contributions

我们的总体思路是按照 Cuneyt Akinlar 的来的, 同样对于参数不敏感. 唯一需要调整的参数是可见圆阈值, 该参数在检测交叠的硬币时需要调整, 一般情况下设为 0.5 即可.

相对于他的实现, 我们的主要贡献在于:

1. 对图像尺寸不敏感(下文中有实验对比);
2. 边缘检测部分实现简单.

7 Algorithm

我们的算法针对硬币检测作了一个假设: 待检测的圆的尺寸没有数量级的差异.

算法总体分为 4 个阶段:

1. 用折线表示边缘;
2. 用这些折线中计算出可能属于某个圆的弧及其对应的圆的参数;
3. 组合弧, 构成圆, 筛选出较好的圆.
4. 针对图像图像尺寸的后处理.

其中阶段 1 用 opencv 自带的函数实现即可, 非常简单; 阶段 2 按照 Cuneyt Akinlar 的描述来做, 其中某些参数的定义与其描述的不一样; 阶段 3 在 Cuneyt Akinlar 的描述中非常简略, 实际上需要实现的东西是最多的, 我们将其细化成两个步骤来做; 步骤 4 利用对于带检测圆的尺寸的假设, 根据第一次的检测结果调整图像尺寸, 然后重新检测, 其实现也非常简单.

7.1 Edge detection

首先进行高斯模糊, 然后 Canny 边缘检测, 再用形态学的 close 操作连接断掉的边缘, 最后用检测出边缘的轮廓线并用多边形进行近似.

下面的代码描述了所用的参数:

```
cv::Mat1b result;
cv::cvtColor(input, result, CV_BGR2GRAY);
cv::GaussianBlur(result, result, cv::Size(9, 9), 2, 2);
cv::Canny(result, result, 10, 30, 3);
auto elem = cv::getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(3, 3));
cv::morphologyEx(result, result, cv::MORPH_CLOSE, elem);
findContours( result, contours, hierarchy, CV_RETR_LIST, CV_CHAIN_APPROX_SIMPLE);
…
```

```
cv::approxPolyDP(contour, result, CONTOUR_APPROXIMATION_THRESHOLD, true);
```

其中在对轮廓线用多边形近似的时候需要把结果顶点序列的首点加入尾部，尾点加入首部，以方便后续对于转角的处理. CONTOUR_APPROXIMATION_THRESHOLD 取 2.

7.2 Extract arcs from edge segments

基本算法与 Cuneyt Akinlar 描述的一致，即从检测转角序中提取连续的一段同反向的，且角度大于 6 度小于 60 的子段作为弧.

但是我们发现这种简单的实现不能处理圆角矩形的情况，即虽然转角大小满足要求，但是其邻边非常长，导致误检. 我们的办法是在原有算法的基础上判断提取出的弧上对应的圆心是否到弧上某条边的距离小于半径的 0.866 倍(即在极端情况下，把六边形当作圆). 如果有，则在该边处将弧切断.

在计算弧对应的圆的参数时，我们使用线性最小二乘. 实现参考了
{ http://goo.gl/IXvAX }.

7.3 From arc to circle

基本步骤如下:

1. 具有相似圆心和半径的弧归为同一组，每条弧可属于多个组.
2. 对于每个组，寻找一个子集，使得该子集中的弧覆盖至少μ的弧长，且其对应的圆的 MSE(mean squared error)最小. μ为可见圆阈值.
3. 从具有相似圆心和半径的圆中筛选出 MSE 最小的.

下面针对 3 个步骤具体讨论.

7.3.1 Step 1

伪代码如下:

```
n = arc count
g = 0
for i in [0,n-1]
    if i marked
        continue
    mark i
    push i into group g
    for j in [i+1,n-1]
        if arc i near arc j
            mark j
            push j into group g
    ++g
```

判断弧是否相近的伪代码如下:

```
function near(arc a, arc b)
    minr = min(radius(a), radius(b))
    maxr = max(radius(a), radius(b))
    return
        minr > (1-COMBINE_THRESHOLD_RELATIVE)*maxr &&
```

```
distance(center(a)-center(b)) <= COMBINE_THRESHOLD_RELATIVE*maxr
```

其中 COMBINE_THRESHOLD_RELATIVE 取 0.3.

7.3.2 Step 2

这是一个实数域的背包问题，收益是加入该弧后所覆盖的圆周长度的增量，花费是加入该弧后 MSE 的增量，并且这两个值都是依赖于已经选择的弧的，很难精确求解. 其近似算法的伪代码如下:

```
n = arc count
gmse = mse of all arcs
gcover = covered circumference of all arcs
for i in [0,n-1]
    cost[i] = gmse-(mse of all arcs except i)
    benefit[i] = gcover-(covered circumference of all arcs except i)
    index[i] = i
sort index so that benefit[index[i]]/cost[index[i]] decrease
for i in [0,n-1]
    if covered circumference of selected arcs >= cover threshold
        break
    select arc index[i]
```

7.3.3 Step 3

伪代码如下:

```
n = circle count
for i in [0,n-1]
    if i is marked
        continue
    for j in [i+1,n-1]
        if j is marked
            continue
        if near(circle[i],circle[j])
            if mse(circle[i]) < mse(circle[j])
                mark j
            else
                mark i
select all circles that are not marked
```

7.4 Postprocess

在第一遍检测出的圆中找出最大的半径 r，然后对原图缩放 50/r 倍，即理想圆的半径是 50. 然后重新检测.

实际上，在检测出的圆的半径与 50 相差没有数量级上的差异时，不做缩放也是可以的. 因为在从弧计算出圆的步骤中所用的阈值都是相对半径的相对阈值.

8 Experiments

以下所有实验仅仅调整可见圆阈值，在未加说明的情况下，该阈值均设为 0.5.

8.1 Compare with Cuneyt Akinlar's implementation

8.1.1 Best circle selection among neighboring circles

Figure 8-1 input 800x552

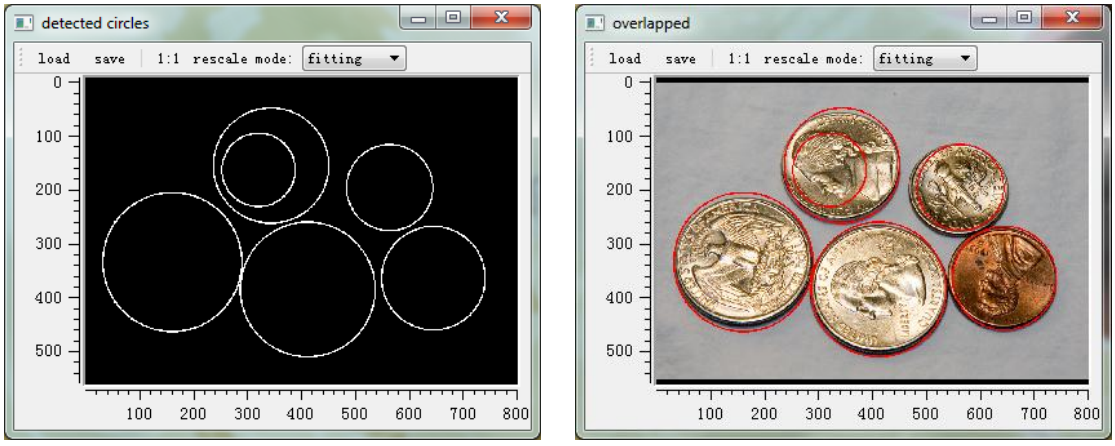

Figure 8-2 mine result

Figure 8-3 Cuneyt Akinlar's result

我们的后处理过程在临近的圆中选择了最优的一个.

8.1.2 Inner circle detection

Figure 8-4 input 1632x1224

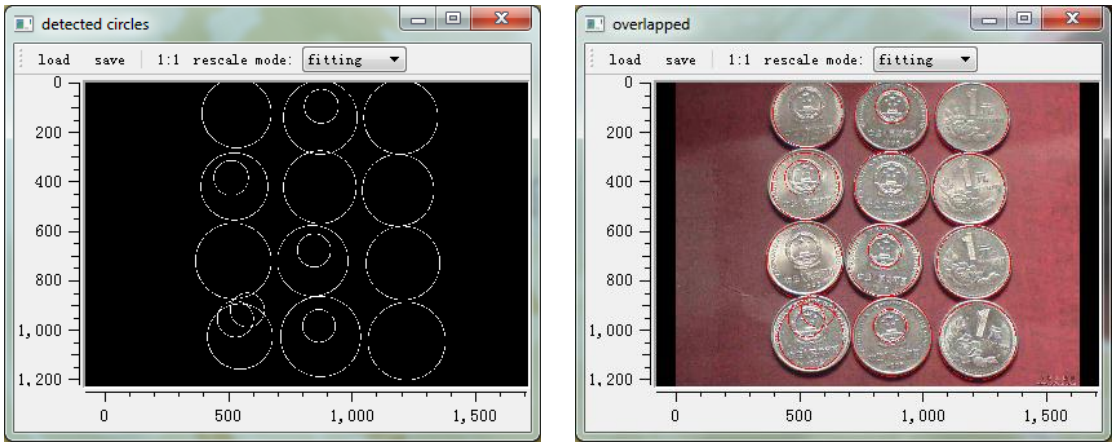

Figure 8-5 mine result

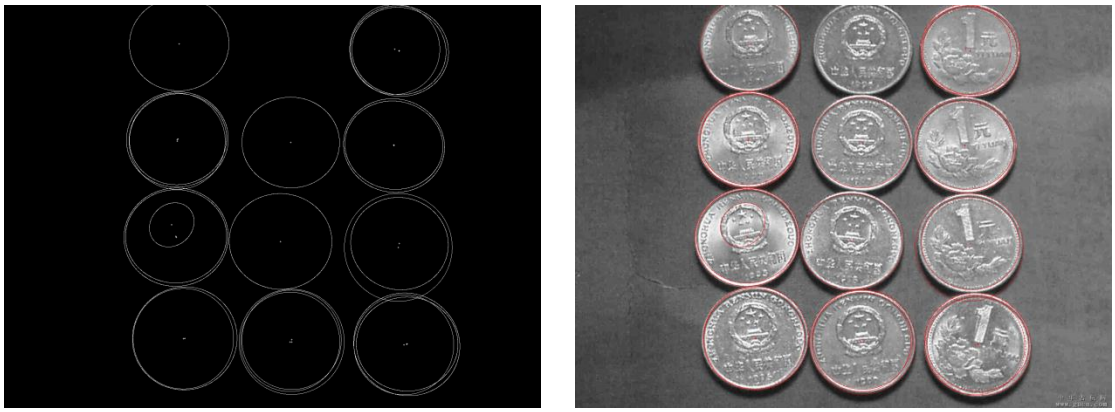

Figure 8-6 Cuneyt Akinlar's result

我们的算法准确地检测出了所有硬币，并且检测出了 8 个国徽中的 5 个(其中一个位置不太准确).

8.1.3 Detection on large image

Figure 8-7 input 2048x1536

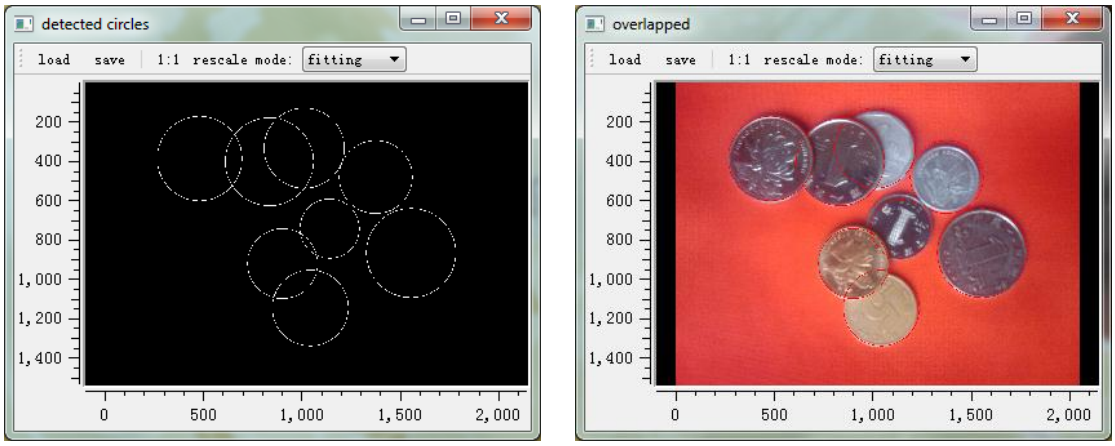

Figure 8-8 mine result

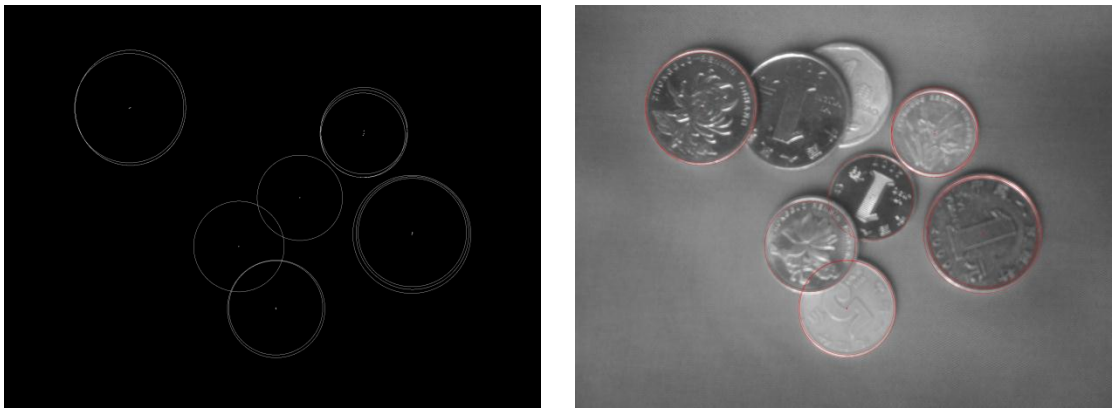

Figure 8-9 Cuneyt Akinlar's result

得益于使用了相对阈值和自动缩放的后处理过程，我们的算法在大尺寸图像中准确地检测出了所有硬币.

8.2 Detect in different environments

8.2.1 Detect among non-circle objects

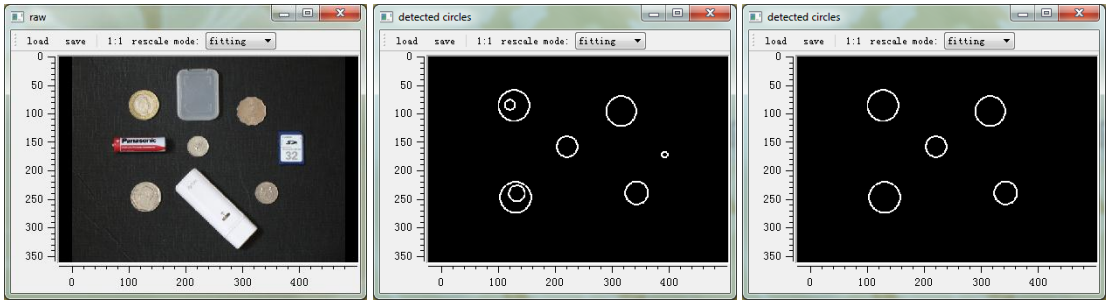

Figure 8-10 input(left), threshold 0.5(middle), threshold 0.7(right)

在可见圆阈值为 0.5 时，检测出了所有硬币，但是因为 SD 卡上的数字 3 非常像圆，所以被误检了．当阈值为 0.7 时，没有误检.

8.2.2 Detect on complex texture

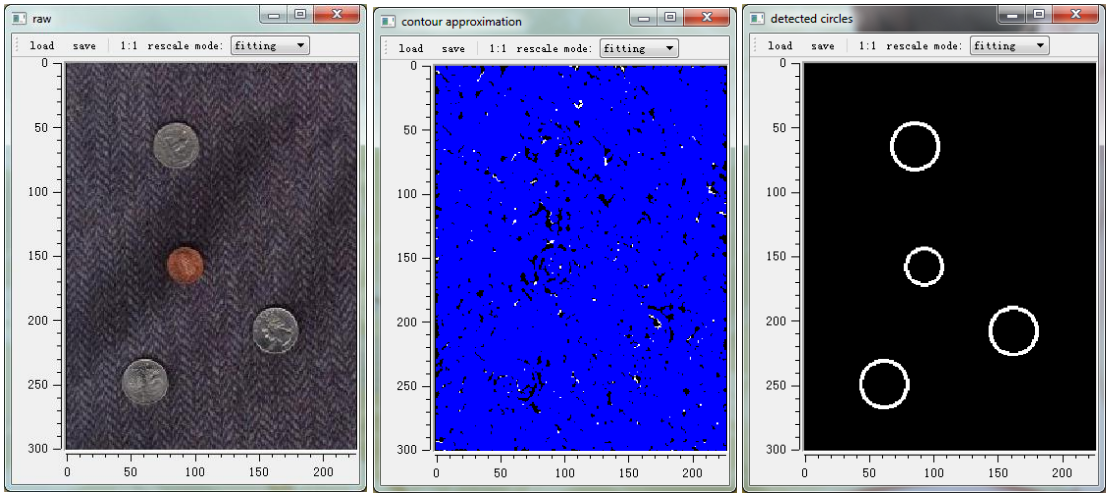

Figure 8-11 input(left), contour approximation(middle), result(right)

该图的背景纹理非常复杂. 从轮廓线近似图中，肉眼几乎无法识别出圆，但是我们的实现依然可以工作得很好. 最重要的是，不需要人为设定任何阈值.

8.2.3 Detect overlapped coins

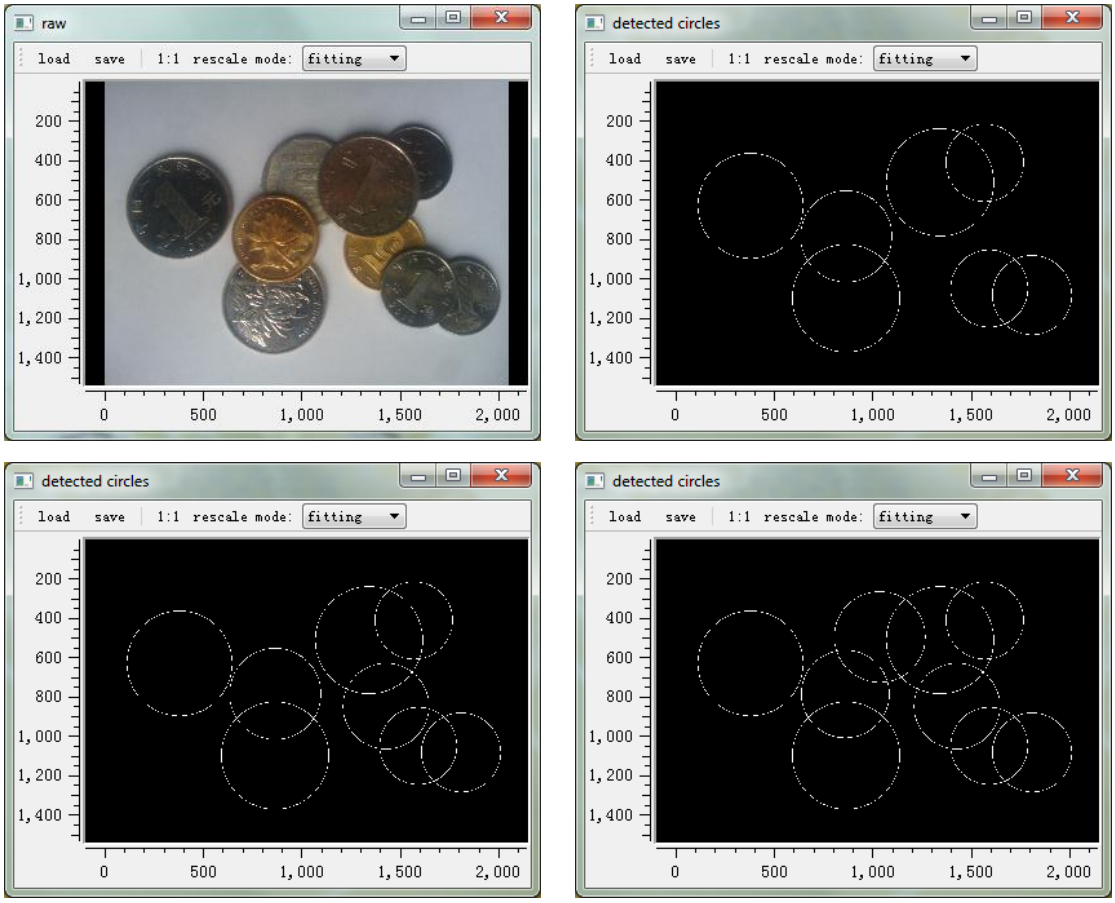

Figure 8-12 threshold 0.5 (right-top), threshold 0.4 (left-bottom), threshold 0.35 (right-bottom)

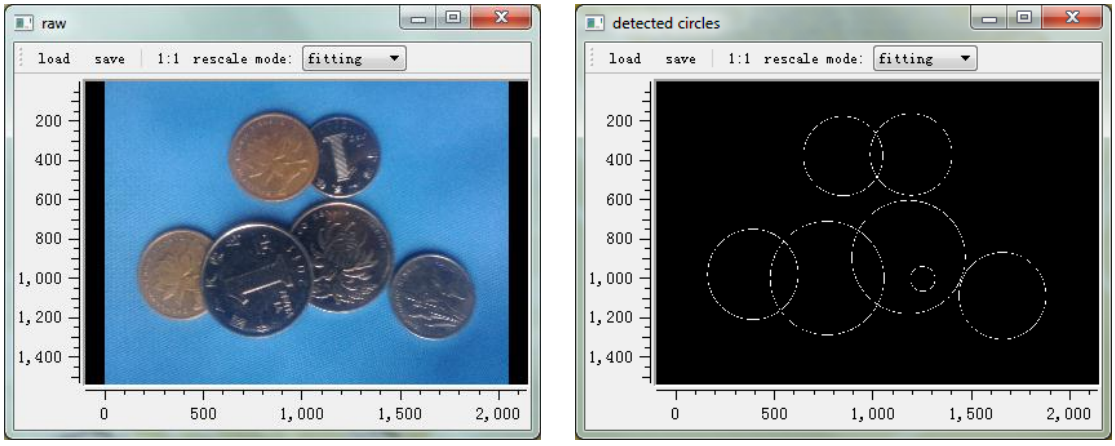

Figure 8-13 threshold 0.5

从 Figure 8-12 可以看出，在硬币有交叠的情况下，改变可见圆阈值即可准确检测出被覆盖的硬币．但是在 Figure 8-13 中，使用阈值 0.5，有一个小圆被误检出来了．对于硬币检测，可以用一个后处理过程(比如判断相对最大圆的半径是否过小，圆之间是否存在包含关系)来去除它们.

9 Future work

限于时间关系，没有做币种的识别，打算用 SIFT 来做.

bin/coin/mine 目录下有很多我们拍的硬币照片，可以做一个识别率的统计，跟{ http://ceng.anadolu.edu.tr/CV/EDCircles/demo.aspx }上的 demo 程序比一比，但是文件很多，而且那个网页的访问速度也不快，手动做的话太慢了，需要写个脚本来弄.

参考文献

[1] C. Topal, C. Akinlar, and Y. Genc, “Edge Drawing: A Heuristic Approach to Robust Real-Time Edge Detection,” in *Proceedings of the 2010 20th International Conference on Pattern Recognition*, 2010, pp. 2424-2427.