EX3 实验报告

15331159 李沁航

实验环境：Ubuntu 16.04 LTS 虚拟机

可执行文件：a.out

代码文件：canny.h canny.cpp HoughTransform.h HoughTransform.cpp

一、实验部分

3-1 找到图像的四个边缘，提取 A4 纸的角点，并且输出直线方程。

1. 定义一个结构体 hough，用于表示 hough space 极坐标中的点，以及一个 point 结构体，表示在二维坐标系中的点。

```
struct hough {
  hough(int i, int r) : i(i), r(r) {}
  hough() {}
  friend bool operator<(const hough& a, const hough& b) {
    return a.r < b.r || (a.r == b.r && a.i < b.i);
  }
  int i;
  int r;
};

struct point{
  int x, y;
  point(int a, int b) {
    x = a;
    y = b;
  }
};
```

2. 通过使用在上一次作业中 canny 算子得到的结果，对每一个值为 255 的像素点，进行 hough transform，并存在 map<hough, int>类型的变量 houghSpace 中。

```
  cimg_forXY(img, x, y) {
          if (x < 10 || x > width-10 || y < 10 || y > height-10) continue;
          if (img(x, y) == 255) {
              for (int i = 0; i < 180; i++) {
                  thete = PI*i/180;
                  r = (int)(x*cos(thete)+y*sin(thete));
                  if (r < -maxLength || r > maxLength)
                      continue;
                  if(r >=0 && r < maxLength)
                    houghGraph(i,r,0) = houghGraph(i,r)+1;
                  hough h(i, r);
                  houghSpace[h]++;
              }
          }
      }
      houghGraph.display();
```

在 houghSpace 中 houghSpace[h]的值就表示的是这个点上的计数值的大小。

3. 进行排序。

```
for (auto & x : houghSpace) {
    sortHough[x.second] = x.first;
}
```

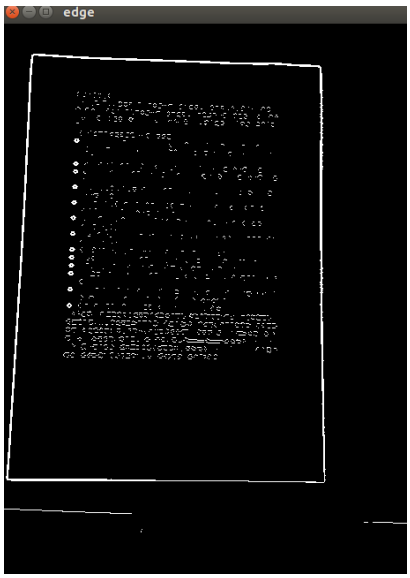
4. 对排序后的结果进行判断，如果判断为在原图像中的同一条直线，则去掉这个结果，否则将其存下来。

```
for (auto it = sortHough.rbegin(); it != sortHough.rend(); ++it) {
    bool duplicate = false;
    for (auto x : line) {
      //check whether the line is duplicated
      //if it's duplicated,we ignore this line
        if (abs(abs(it->second.r) - abs(x.r)) < 10 &&
        (abs(it->second.i - x.i) < 3 || abs(it->second.i + x.i - 180) < 3)) {
            duplicate = true;
            break;
        }
    }
    if (duplicate) continue;
//if it's not duplicated,we use the point in the polar system to draw the line
//on the origin picture
    line.insert(it->second);
```

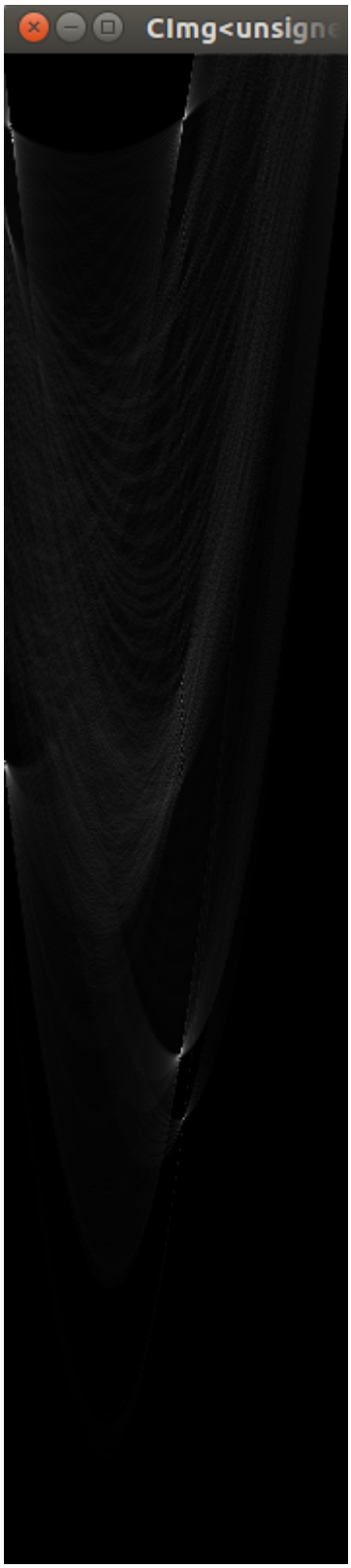
5. 取在极坐标中最“亮”的四个点作为结果，然后对图像中的每一个点，计算它在对应的 theta 值下映射得到的点，是否与这个四个点中的某一个重合，是的话这个点就作为边界点，然后标记为红色，作为标记的红色边界。并将角点标记出来。

```
    r = it->second.r;
    thete = PI*(it->second.i)/180;
//draw red lines for the edges
    cimg_forXY(lineOutput, x, y) {
            int cur_r = int(x*cos(thete) + y*sin(thete));
            if (abs(cur_r - r) < 2) {
              if(lineOutput(x, y, 0) == 255 &&lineOutput(x, y, 1) == 0 && lineOutput(x, y, 2) == 0) {
                intersetcions.push_back(point(x, y));
              }
                lineOutput(x, y, 0) = 255;
                lineOutput(x, y, 1) = 0;
                lineOutput(x, y, 2) = 0;;
            }
       }
```

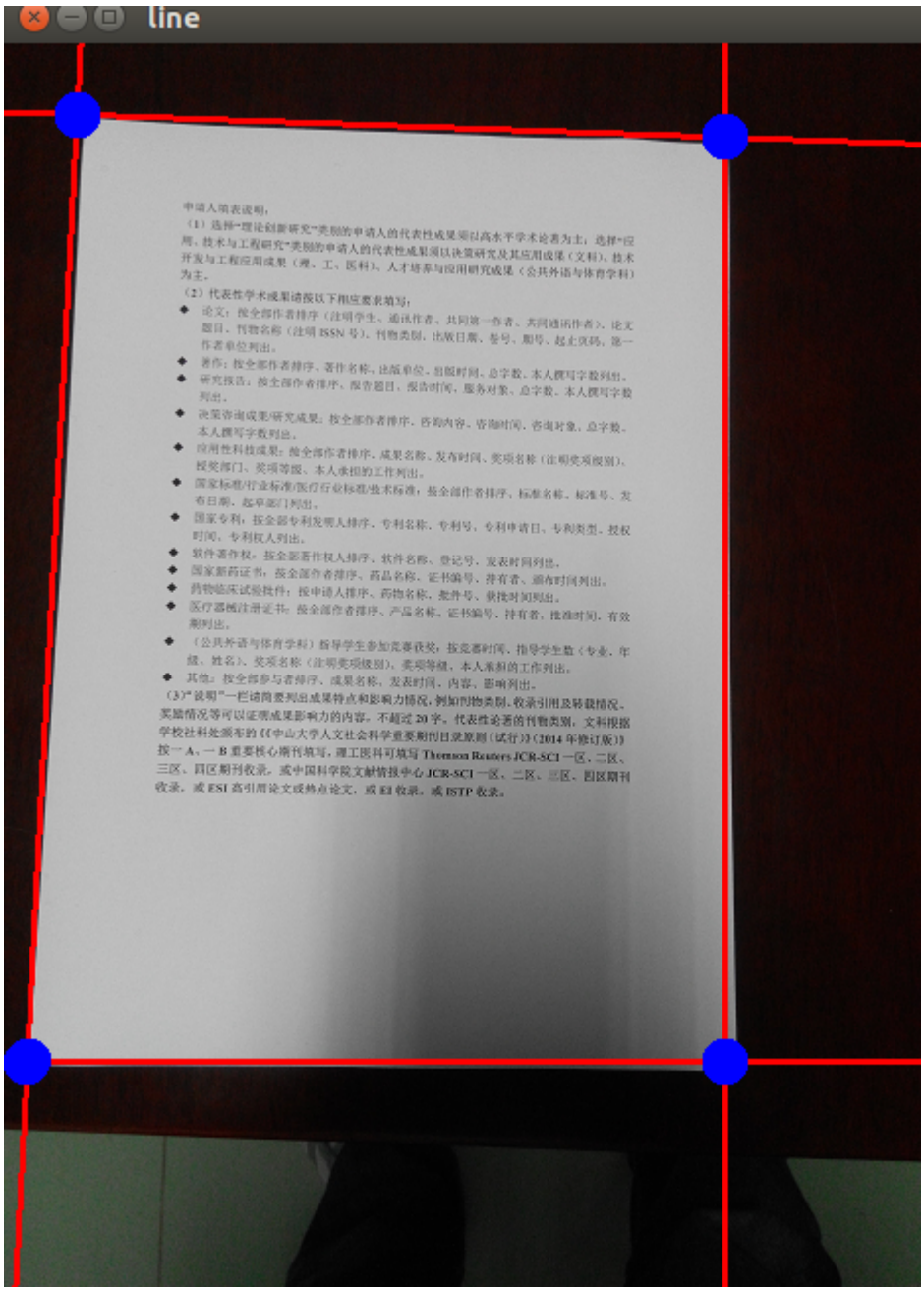
3-1 实验结果。



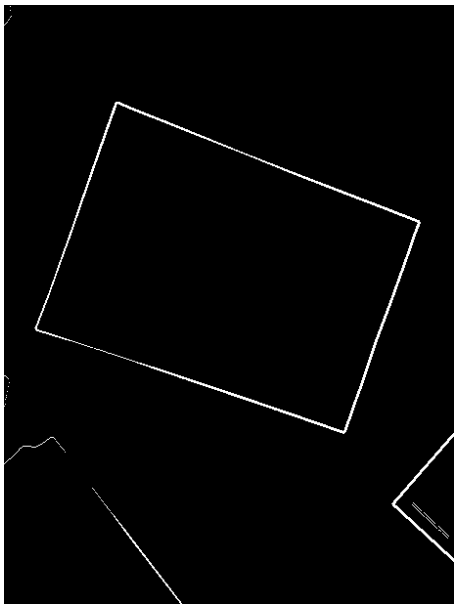
该图片对应的 hough space 的亮度图：



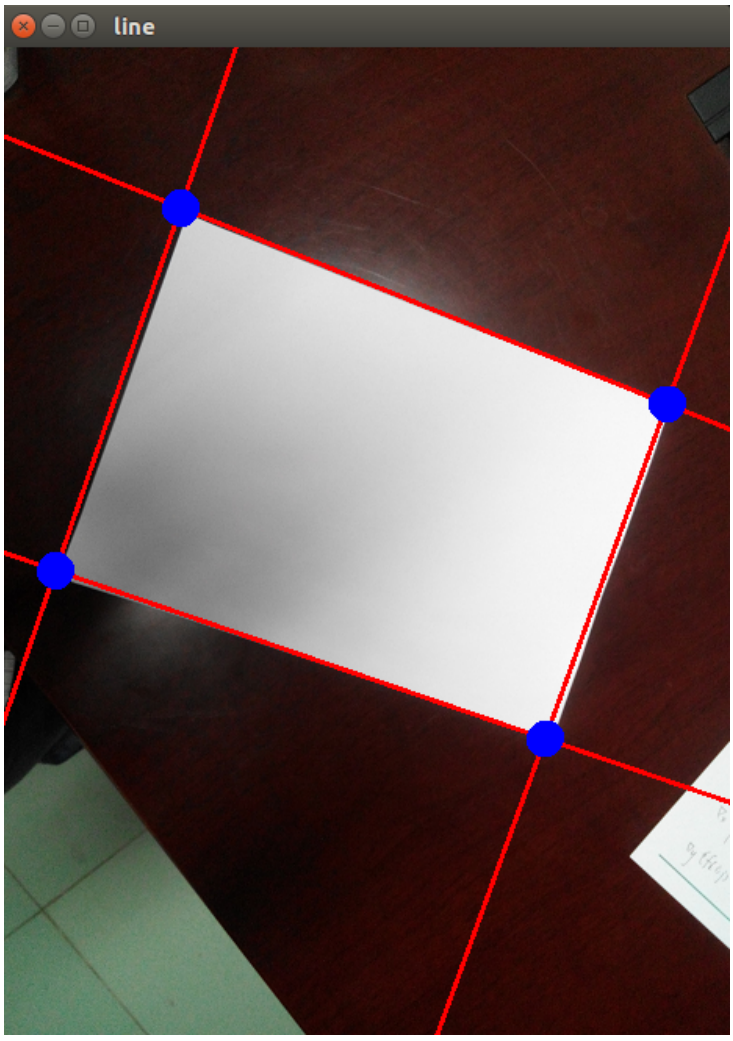
标记完成后的图片为：



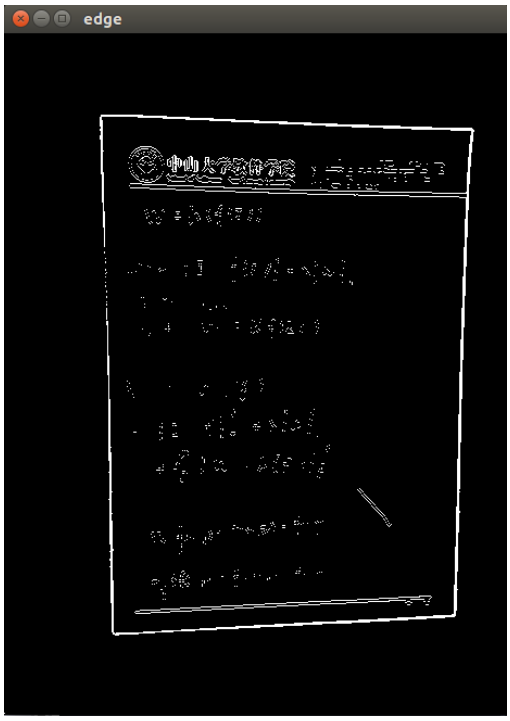
图片二：



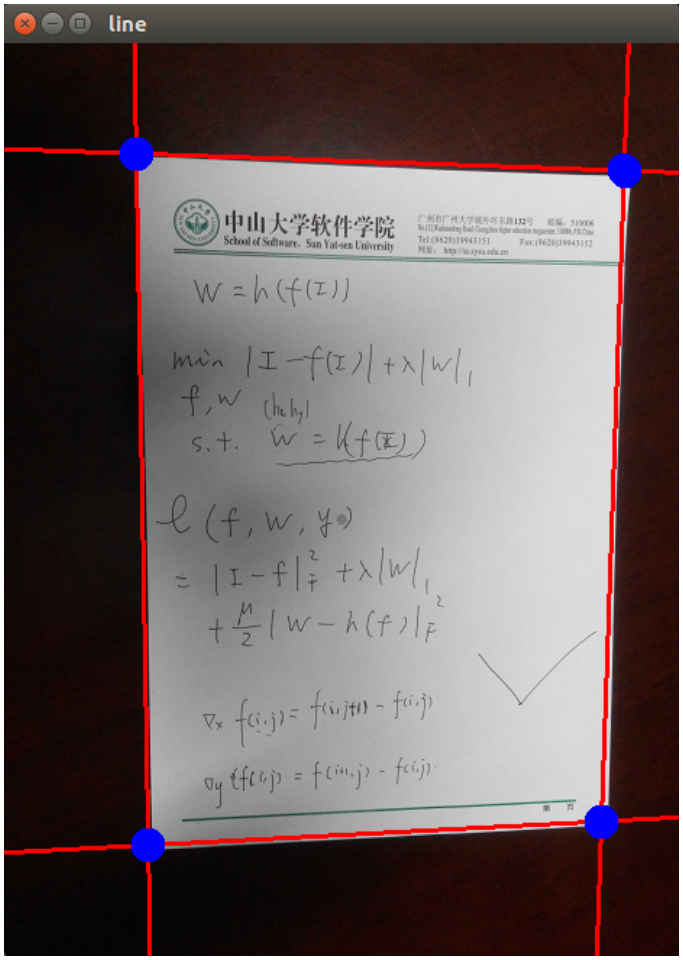

CImg<unsign


line

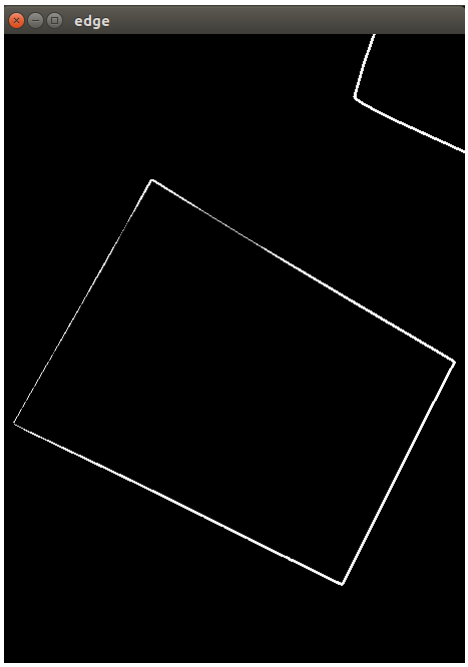
图片 3：

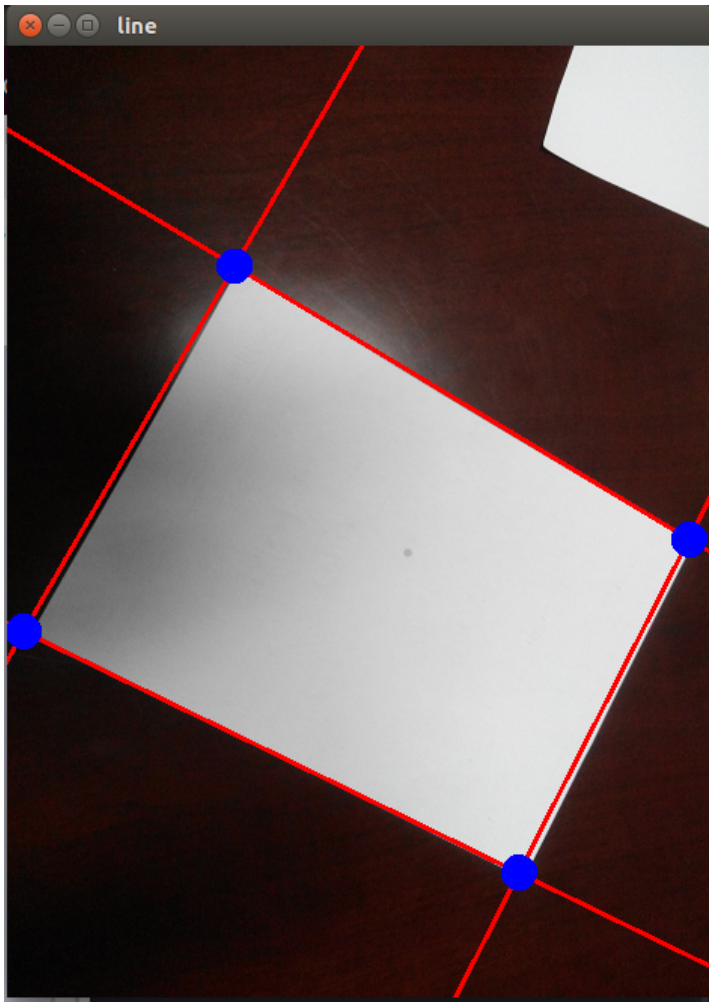

edge


CImg<unsign


line
中山大学软件学院
School of Software, Sun Yat-sen University

图片四：





其余的结果已经存在 dataSet1 中，命名为 answer_i.bmp i 为图片对应的编号。

思考：

加快运速度的方法可以使用 down sampling 的方式，将图片的大小减小，从而能够减少运算次数，加快运行的速度。

3-2 输出图像的边缘，将图像中的边缘拟合成圆，并且输出图片数量。

首先定义结构体 circle。

```
struct circle {
  circle(int a, int b, int r):a(a), b(b), r(r) {}
  circle() {}
  friend bool operator < (const circle& c1, const circle& c2){
    return c1.r < c2.r || (c1.r == c2.r && c1.a < c2.a) ||
    (c1.r == c2.r && c1.a == c2.a && c1.b < c2.b);
  }
  int a;
  int b;
  int r;
};
```

用途与 3-1 步骤中的类似。

在进行 hough transform 之前，对图片进行 down sampling，可以加快运算的速度。 在找圆的部分，由于不知道圆形的半径，所以需要假定一个半径之后，进行 hough transform。所以需要对每一个可能的半径都进行一次运算。最后找出最合适的点。

```
cimg_forXY(img,x, y) {
  // cout << x << " " << y << endl;
  if (x < 15 || x > width-15 || y < 15 || y > height-15) continue;
  if(img(x,y) == 255) {
    for(int radius = 25; radius < maxRadius; radius++) { //try each radius
      for(int angle = 0; angle <= 360; angle++) {
        double theta = angle * PI /180;
        int temp1 = x - radius * cos(theta);
        int temp2 = y - radius * sin(theta);
        circle c(temp1, temp2, radius);
        circleHoughSpace[c]++;
      }
    }
  }
}
```

在试验中，圆形半径的区间为[25, height>width?(heigh/2-10):(width/2-10)，对每一个可能的半径进行尝试并且进行映射。

对得到的结果进行排序之后，设定一个 threshold，在超过在 threshold 的情况下，如果不存在 duplicate 的圆，就认定它为一个圆。

```
for(auto it = sortCircleHoughSpace.rbegin(); it != sortCircleHoughSpace.rend(); ++it) {
  if(it->first >= threshold) {
    bool duplicate = false;
    for(auto x : originCircle) {
      if(abs(x.a-it->second.a) <= 35 &&
      abs(x.b - it->second.b) <= 35 ) {
        if(abs(x.r - it->second.r) > 35) {
          duplicate = false;
          break;
        }
        duplicate = true;
        break;
      }
    }
    if(duplicate) {
      continue;
    } else {
      originCircle.insert(it->second);
    }
  }
}
```

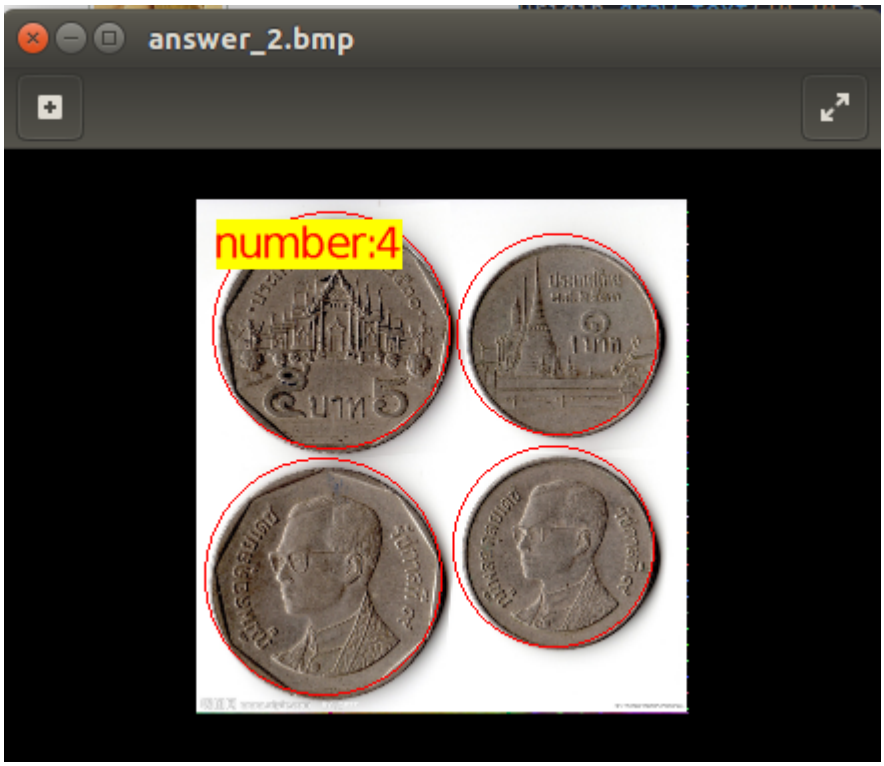
对所有的点检测了之后，在图片上画出标记的圆形。

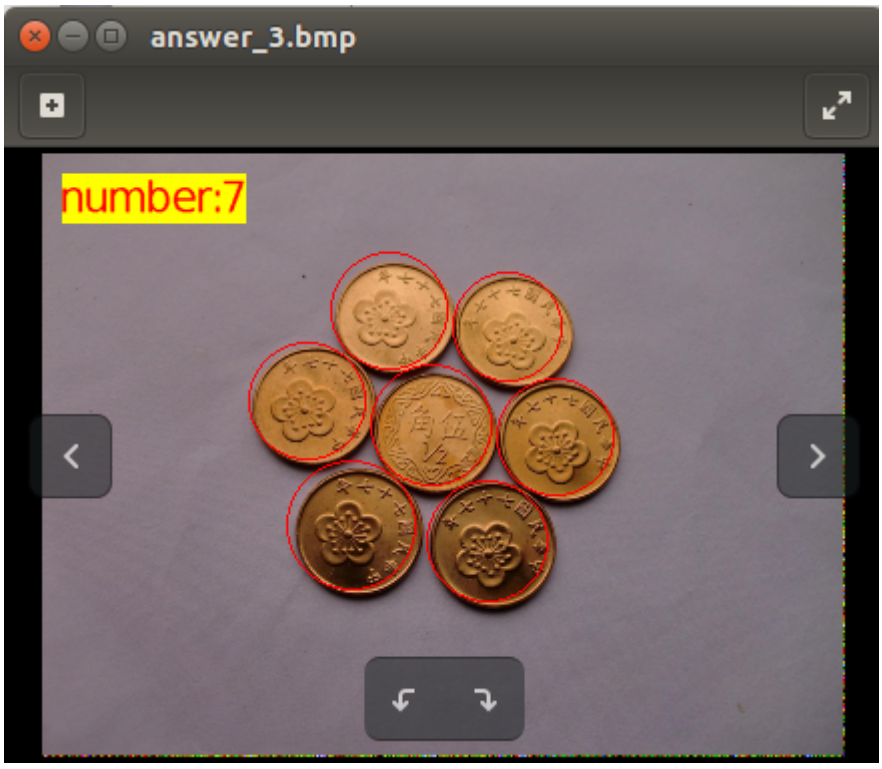
```
for(auto x : originCircle) {
  smallOrigin.draw_circle(x.a, x.b, x.r, red, 1, 5);
}
```

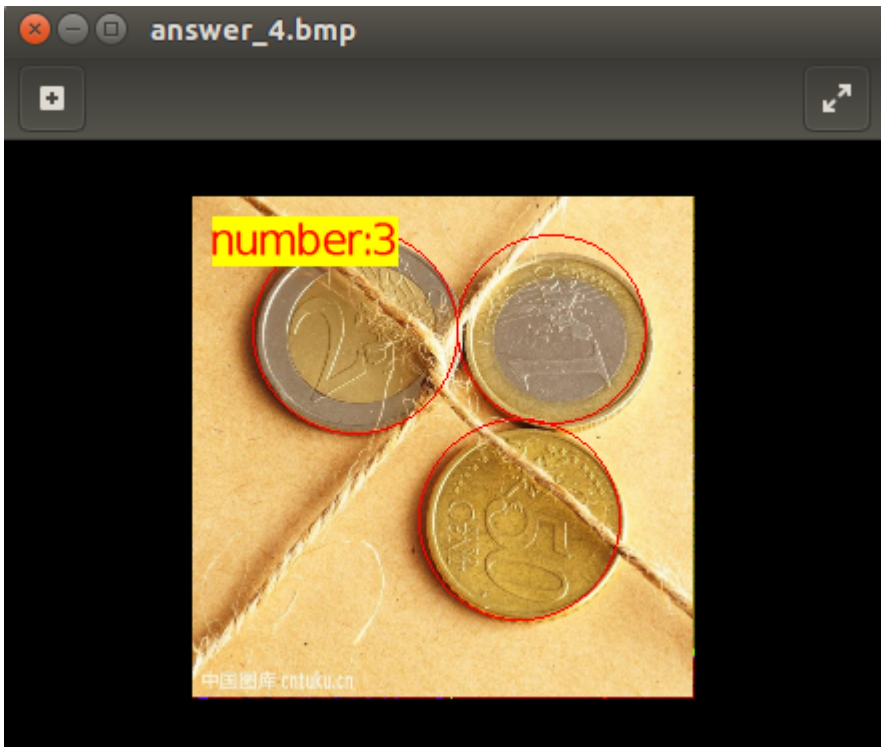
并且标记出找到的个数：

```
char num;
num='0'+originCircle.size();
a = a+num;
smallOrigin.draw_text(10,10,a.c_str(),red,yellow,1,25);
```

得到的实验结果如图：


answer_2.bmp
number:4


answer_3.bmp
number:7



得到的结果已经储存在 dataset2/answer_i.bmp 中，其中 i 为图片的编号。