

# 反演变换

## 算法思想

给定反演中心  $O$  和反演半径  $r$ ，剩余点  $A$  的反演点  $A'$  满足  $|OA| \times |OA'| = R^2$

可以发现不过  $O$  的圆  $B$  其反演图形也是不过  $O$  的圆  $B'$

圆  $A$  半径为  $r_1$  其反演图形的半径为  $\frac{1}{2} \left( \frac{1}{|OA| - r_1} - \frac{1}{|OA| + r_1} \right) R^2$

## 代码实现

```

struct Inversion {
    Point o; //反演中心
    double r; //反演半径
    Inversion() {}
    Inversion(Point _o, double _r) {
        o = _o;
        r = _r;
    }
    //点的反演 flag为0获取失败1获取成功
    void getPointInv(Point a, Point &aa, int &flag) {
        if(a == o) {
            flag = 0;
            aa = a;
            return;
        }
        Point ptmp = a - o;
        double len = ptmp.len();
        ptmp = ptmp.trunc(r * r / len);
        aa = o + ptmp;
        flag = 1;
    }
    //圆的反演 flag为1变成圆-1变成直线
    void getCircleInv(circle c, Line &l, circle &cc, int &flag) {
        if(c.relation(o) ^ 1) {
            Point p1, p2, pp1, pp2;
            Line lt;
            flag = 1;
            if(c.p == o) {
                cc.p = o;
                cc.r = r * r / c.r;
                return;
            }
            lt = Line(c.p, o);
            int ii = c.pointcrossline(lt, p1, p2);

```

```
    int f;  
    getPointInv(p1,pp1,f);  
    getPointInv(p2,pp2,f);  
    Point pp=(pp1+pp2)/2;  
    cc.p=pp;  
    cc.r=pp1.distance(pp2)/2;  
    return;  
}  
flag=-1;  
Point ptmp=c.p*2-o,pptmp,p1,p2;  
int f;  
getPointInv(ptmp,pptmp,f);  
p1=o-pptmp;  
p1=p1.rotleft();  
p1=p1+pptmp;  
l=Line(pptmp,p1);  
}  
}iv;
```

From: <https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link: [https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal\\_string:%E7%8E%8B%E6%99%BA%E5%BD%AA:%E5%8F%8D%E6%BC%94&rev=1628082473](https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%8E%8B%E6%99%BA%E5%BD%AA:%E5%8F%8D%E6%BC%94&rev=1628082473)

Last update: 2021/08/04 21:07