Analytical Geometry

Sun Yaofeng, Huang Zhihuan, Ye Jianing

Peking University

February, 2019

Problem 1

- Given two straight lines in the 3D space, compute the distance between the two lines.

Problem 1

- Given two straight lines in the 3D space, compute the distance between the two lines.
- We need to find a segment which is perpendicular to both lines, and its length will be the answer.

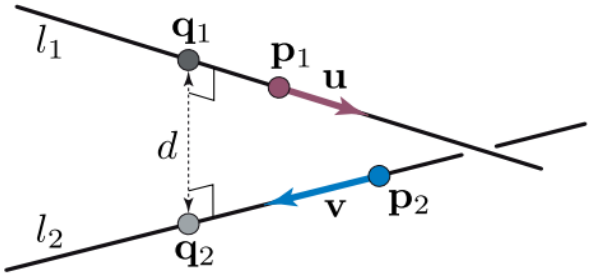


Figure 1.12: The distance between two lines in 3D.

Brute Force

- Brute force creates miracle

$$\begin{aligned}\langle \mathbf{p}_1 - \mathbf{p}_2 + s\mathbf{u} - t\mathbf{v}, \mathbf{u}\rangle &= 0 \\ \langle \mathbf{p}_1 - \mathbf{p}_2 + s\mathbf{u} - t\mathbf{v}, \mathbf{v}\rangle &= 0\end{aligned}$$

Intuition

- We will often try to use geometric understanding first, which will possibly simplify the problem, and then use the algebra.

Intuition

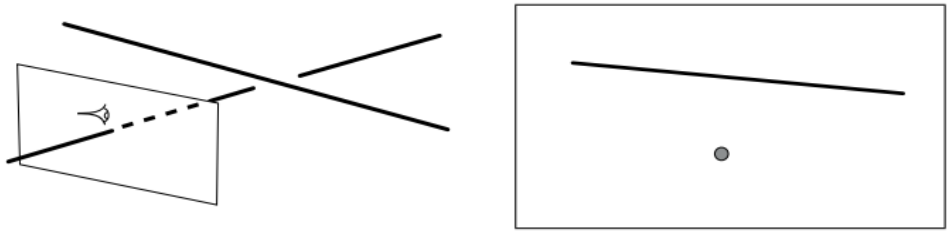


Figure 1.2: If we look at the scene in the direction of one line, that line becomes a single point.

Intuition

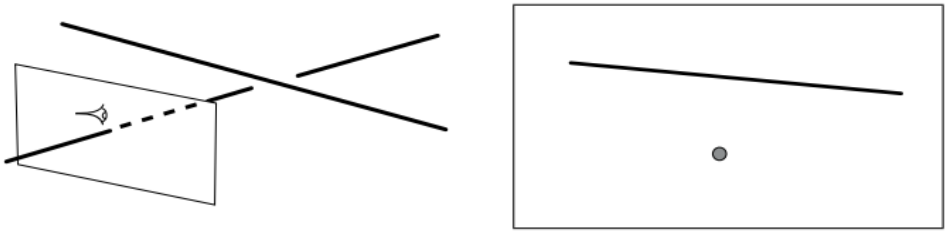


Figure 1.2: If we look at the scene in the direction of one line, that line becomes a single point.

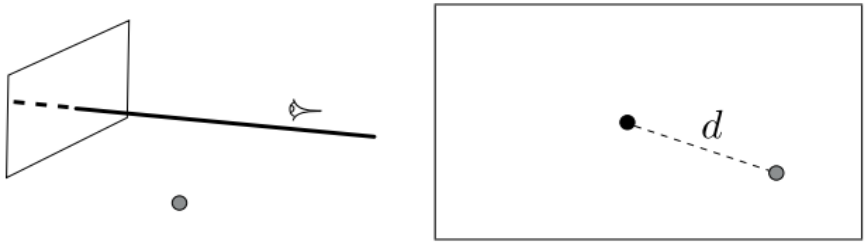


Figure 1.3: If we look at the scene in the direction of the line, that line becomes a single point.

Enlightenment

- There is no way to completely escape the math.
- However, the geometric approach provides us with a very convenient recipe that is easy to program.

Problem 2

- Compute the intersection between a ray and sphere.
- The sphere is of radius r and centered at c, and the ray emanates from point p_0 in direction v.

Problem 2

- Compute the intersection between a ray and sphere.
- The sphere is of radius r and centered at c, and the ray emanates from point p_0 in direction v.

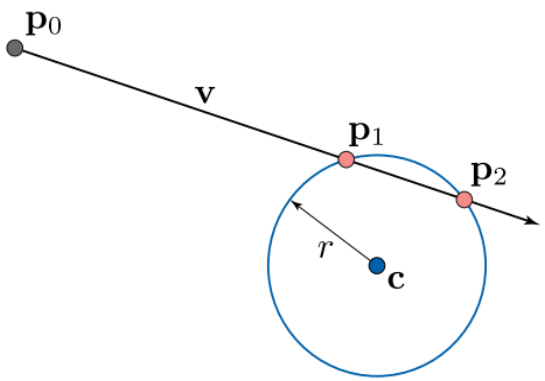


Figure 1.4: A ray intersecting a sphere at two points.

Problem 2

- Use projection!

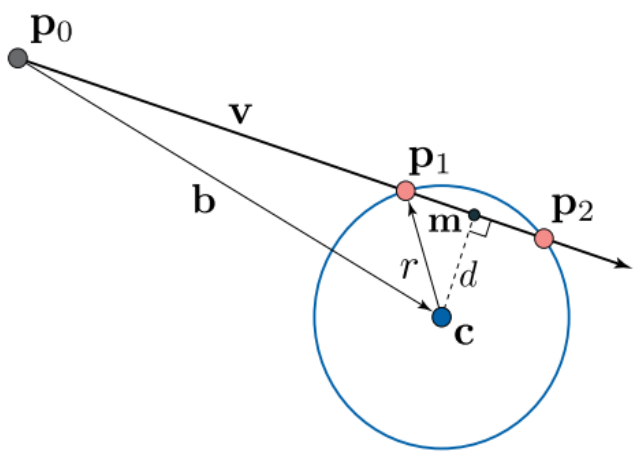


Figure 1.5: A ray intersecting a sphere at two points. The point $\mathbf{m}$ is the closest point along the ray to the center $\mathbf{c}$ of the sphere.

Enlightenment

- If we have the basic geometric routines implemented (like projections of lines), all we need to do is call these routines with the appropriate input.

Vector

Definition 1

Points locations in space. ($\mathbf{p}$, $\mathbf{q}$)
Vectors direction and magnitude. ($\mathbf{v}$, $\mathbf{w}$)
Operation +,-

Vector

Definition 1

Points locations in space. $(\mathbf{p}, \mathbf{q})$
Vectors direction and magnitude. $(\mathbf{v}, \mathbf{w})$
Operation +,-

```
struct poi{
    double x, y;
    poi(double x = 0, double y = 0): x(x), y(y) {}

    poi operator + (const poi&A){
        return poi(x + A.x, y + A.y);
    }
    poi operator - (const poi&A){
        return poi(x - A.x, y - A.y);
    }
    poi operator * (int t){
        return poi(x * t, y * t);
    }
};
```

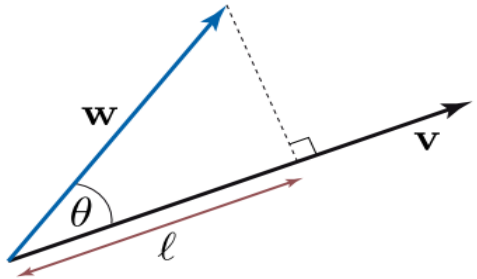
Scalar Product

Definition 2

Scalar product (dot product or inner product)
Given two vectors $\mathbf{v}$ and $\mathbf{w}$ and the angle θ between them,

$$\langle \mathbf{v}, \mathbf{w} \rangle = \|\mathbf{v}\| \cdot \|\mathbf{w}\| \cdot \cos\theta$$

Projection $l = \frac{\langle \mathbf{v}, \mathbf{w} \rangle}{\|\mathbf{v}\|}$ and $\cos\theta = \frac{l}{\|\mathbf{w}\|}$.



Scalar Product

```
double dot(const poi&a, const poi&b){
    return a.x * b.x + a.y * b.y;
}
```

Scalar Product

```
double dot(const poi&a, const poi&b){
    return a.x * b.x + a.y * b.y;
}
```

Definition 3

Perpendicular in 2D If $\langle \mathbf{v}, \mathbf{w} \rangle = 0$, then the vectors are perpendicular. Therefore, the perpendicular vector of $\mathbf{v} = (x_v, y_v)$ is $\mathbf{v}^{\perp} = (-y_v, x_v)$.

Cross product

Definition 4

Cross product : $\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \sin\alpha\ \hat{\mathbf{n}}$. where $\hat{\mathbf{n}}$ is a unit vector perpendicular to the plane containing $\mathbf{a}$ and $\mathbf{b}$. So the cross product of two vectors is also a vector!

Cross product

Definition 4

Cross product : $\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\|\|\mathbf{b}\| \sin\alpha\ \hat{\mathbf{n}}$. where $\hat{\mathbf{n}}$ is a unit vector perpendicular to the plane containing $\mathbf{a}$ and $\mathbf{b}$. So the cross product of two vectors is also a vector!

$$\mathbf{a} \times \mathbf{b} = \det \begin{bmatrix} \hat{\mathbf{i}} & \hat{\mathbf{j}} & \hat{\mathbf{k}} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{bmatrix}$$

where $\hat{\mathbf{i}}, \hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ are unit vectors forming an orthogonal righthanded coordinate system.

Line

Definition 5

Parametric Representation $l(t) = \mathbf{p} = \mathbf{p}_0 + t\mathbf{v}$

Line

Definition 5

Parametric Representation $l(t) = \mathbf{p} = \mathbf{p}_0 + t\mathbf{v}$

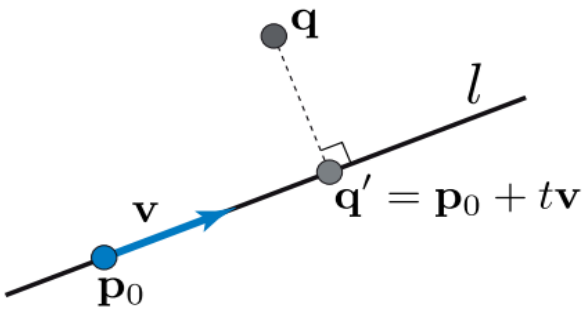
```
struct lin{
    poi p0, v;
    lin(poi p0 = poi(), poi v = poi()): p0(p0), v(v) {}

    poi loc(double t){
        return p0 + v * t;
    }
};
```

The distance between a point and a line

- **Distance between $\mathbf{q}$ and $l(t) = \mathbf{p}_0 + t\mathbf{v}$**

$$\langle \mathbf{q} - \mathbf{p}_0 + t\mathbf{v}, v \rangle = 0$$
$$\langle \mathbf{q} - \mathbf{p}_0, \mathbf{v} \rangle - t\langle \mathbf{v}, \mathbf{v} \rangle = 0$$

- Therefore, $t = \frac{\langle \mathbf{q}-\mathbf{p}_0, \mathbf{v} \rangle}{\|\mathbf{v}\|}$ and $dist^2(\mathbf{q}, l) = \|\mathbf{q} - \mathbf{p}_0\|^2 - t^2$.

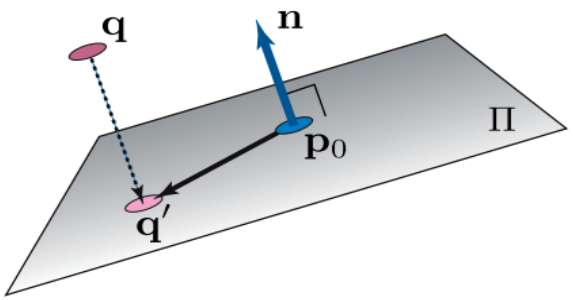


The distance between a point and a plane

- Since $\mathbf{q}$' - $\mathbf{q}$ is parallel to $\mathbf{n}$, $\mathbf{q}' = \mathbf{q} + \alpha\mathbf{n}$.

$$\begin{aligned} \langle \mathbf{q} + \alpha\mathbf{n} - \mathbf{p}_0, \mathbf{n} \rangle &= 0 \\ \langle \mathbf{q} - \mathbf{p}_0, n \rangle + \alpha \langle n, n \rangle &= 0 \\ \alpha &= \frac{\langle \mathbf{p}_0 - \mathbf{q}, n \rangle}{\|n\|^2} \end{aligned}$$

- Therefore, $dist^2(\mathbf{q}, \Pi) = \|\mathbf{q}' - \mathbf{q}\|^2 = \alpha^2 \|\mathbf{n}\|^2 = \frac{\langle \mathbf{q} - \mathbf{p}_0, \mathbf{n} \rangle^2}{\|\mathbf{n}\|^2}$.



The distance between two lines in 3D

- Let l_1 and l_2 be two given arbitrary lines in 3D:

$$l_1(s) = \mathbf{p}_1 + s\mathbf{u}$$
$$l_2(t) = \mathbf{p}_2 + t\mathbf{v}$$

- The distance is attained between two points $\mathbf{q}_1$ and $\mathbf{q}_2$ so that $(\mathbf{q}_1 - \mathbf{q}_2)\perp\mathbf{u}$ and $(\mathbf{q}_1 - \mathbf{q}_2)\perp\mathbf{v}$.

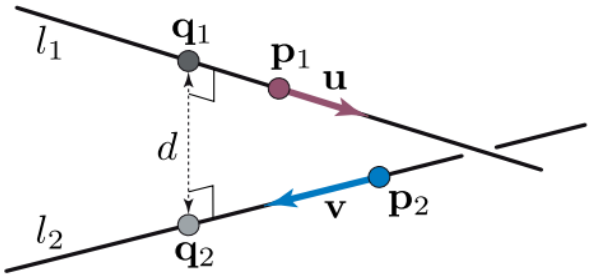


Figure 1.12: The distance between two lines in 3D.

The distance between two lines in 3D

- we want to figure out s and t:

$$\langle \mathbf{p}_1 - \mathbf{p}_2 + s\mathbf{u} - t\mathbf{v}, \mathbf{u} \rangle = 0$$
$$\langle \mathbf{p}_1 - \mathbf{p}_2 + s\mathbf{u} - t\mathbf{v}, \mathbf{v} \rangle = 0$$

- leading to:

$$\langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{u} \rangle + s\|\mathbf{u}\|^2 - t\langle \mathbf{v}, \mathbf{u} \rangle = 0$$
$$\langle \mathbf{p}_1 - \mathbf{p}_2, \mathbf{v} \rangle - t\|\mathbf{v}\|^2 + s\langle \mathbf{v}, \mathbf{u} \rangle = 0$$

The distance between two lines in 3D

- Cramer's Rule!

$$\tilde{s} = \frac{\beta\langle \mathbf{w}, \mathbf{v}\rangle - \|\mathbf{v}\|^2\langle \mathbf{w}, \mathbf{u}\rangle}{\|\mathbf{u}\|^2\|\mathbf{v}\|^2 - \beta^2}$$

$$\tilde{t} = \frac{\|\mathbf{u}\|^2\langle \mathbf{w}, \mathbf{v}\rangle - \beta\langle \mathbf{w}, \mathbf{u}\rangle}{\|\mathbf{u}\|^2\|\mathbf{v}\|^2 - \beta^2}$$

- where $\beta = \langle \mathbf{v}, \mathbf{u}\rangle$ and $\mathbf{w} = \mathbf{p}_1 - \mathbf{p}_2$.
- Finally, we have $dist(l_1, l_2) = \|l_1(\tilde{s}) - l_2(\tilde{t})\|$.

Generate a plane by three non-collinear points

- Three non-collinear points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$.
- A plane with an implicit equation $ax + by + cz + d = 0$ has a normal vector (a, b, c).

Generate a plane by three non-collinear points

- Three non-collinear points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$.
- A plane with an implicit equation $ax + by + cz + d = 0$ has a normal vector (a, b, c).
- The direction of this normal vector is also equal to the cross product of $(\mathbf{p}_1 - \mathbf{p}_2) \times (\mathbf{p}_1 - \mathbf{p}_3)$.
- d can be determined by plugging the value of one of the points into the plane equation.

Linear transformation of plane

- Given a transformation T and a plane $ax + by + cz + d = 0$. We can compute the equation of the transformed plane.

Linear transformation of plane

- Given a transformation T and a plane $ax + by + cz + d = 0$. We can compute the equation of the transformed plane.
- A direct application of the above is to recompute the plane of point $T(\mathbf{p}_1)$, $T(\mathbf{p}_2)$ and $T(\mathbf{p}_3)$.

Linear transformation of plane

- Given a transformation T and a plane $ax + by + cz + d = 0$. We can compute the equation of the transformed plane.
- A direct application of the above is to recompute the plane of point $T(\mathbf{p}_1)$, $T(\mathbf{p}_2)$ and $T(\mathbf{p}_3)$.
- Let us denote $\mathbf{a} = (a, b, c, d)$ and $\mathbf{x} = (x, y, z, 1)$, then we have $\mathbf{a}^\top \mathbf{x} = 0$.
- Similarly, the transformed plane is represented by $\mathbf{a}' = (a', b', c', d')$ and satisfies $\mathbf{a}'^\top T\mathbf{x} = 0$.

Linear transformation of plane

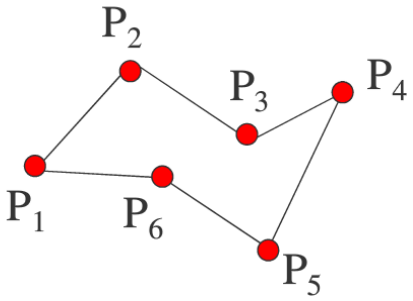
- Given a transformation T and a plane $ax + by + cz + d = 0$. We can compute the equation of the transformed plane.
- A direct application of the above is to recompute the plane of point $T(\mathbf{p}_1)$, $T(\mathbf{p}_2)$ and $T(\mathbf{p}_3)$.
- Let us denote $\mathbf{a} = (a, b, c, d)$ and $\mathbf{x} = (x, y, z, 1)$, then we have $\mathbf{a}^\top \mathbf{x} = 0$.
- Similarly, the transformed plane is represented by $\mathbf{a}' = (a', b', c', d')$ and satisfies $\mathbf{a}'^\top T\mathbf{x} = 0$.
- Since we can write $\mathbf{a}^\top x = 0$ as $\mathbf{a}^\top T^{-1} T\mathbf{x} = 0$, $\mathbf{a}'^\top = \mathbf{a}^\top T^{-1}$ or $\mathbf{a}' = T^{-\top}\mathbf{a}$.

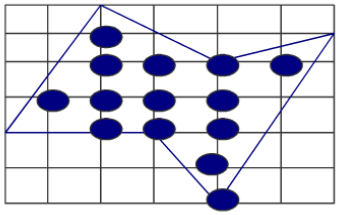
Linear transformation of plane

- Given a transformation T and a plane $ax + by + cz + d = 0$. We can compute the equation of the transformed plane.
- A direct application of the above is to recompute the plane of point $T(\mathbf{p}_1)$, $T(\mathbf{p}_2)$ and $T(\mathbf{p}_3)$.
- Let us denote $\mathbf{a} = (a, b, c, d)$ and $\mathbf{x} = (x, y, z, 1)$, then we have $\mathbf{a}^\top \mathbf{x} = 0$.
- Similarly, the transformed plane is represented by $\mathbf{a}' = (a', b', c', d')$ and satisfies $\mathbf{a}'^\top T\mathbf{x} = 0$.
- Since we can write $\mathbf{a}^\top x = 0$ as $\mathbf{a}^\top T^{-1} T\mathbf{x} = 0$, $\mathbf{a}'^\top = \mathbf{a}^\top T^{-1}$ or $\mathbf{a}' = T^{-\top}\mathbf{a}$.
- The first method's time complexity is $O(n^3)$, but the second one is $O(n^2)$.

多边形的扫描转换

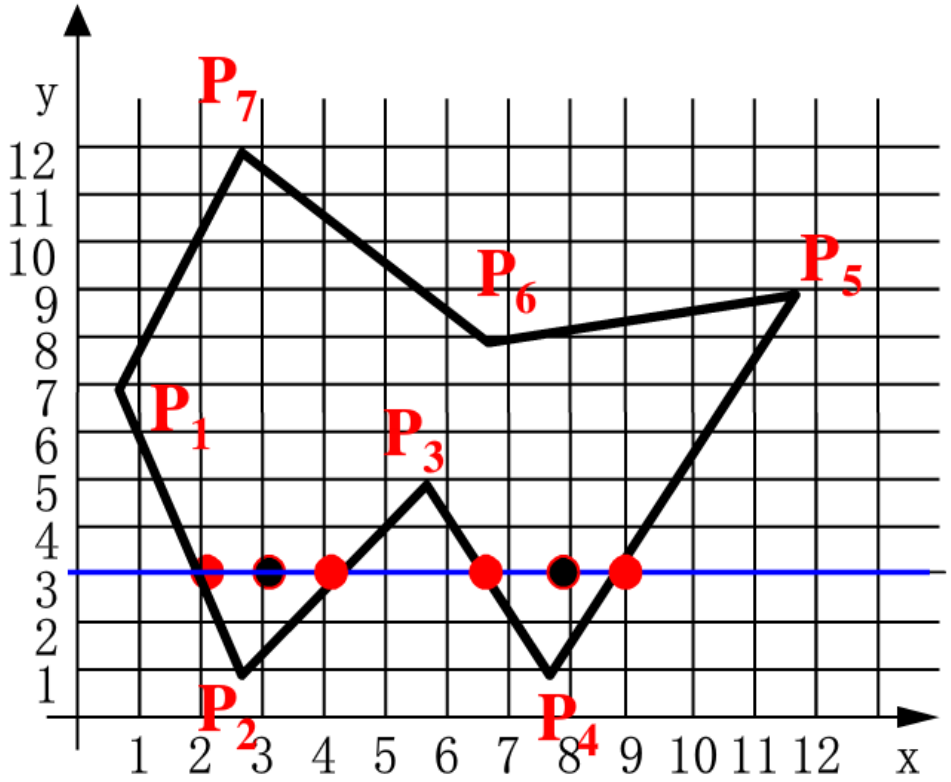
- 在光栅显示屏中，我们需要将一个连续的二维图形转换为离散的点阵形式：





扫描线

- 设屏幕大小为$n \times n$。
- 枚举$y = 1, .., n$，计算多边形与直线$y = i$有交的区间，并染色之。



扫描线

- 设多边形边数为m;
- 计算直线$y = i$与多边形的交，时间复杂度为$O(m)$;
- 得到交的区间后，需要$O(n)$染色;
- 总复杂度为$O(n(n+m))$;

扫描线

- 设多边形边数为m;
- 计算直线$y = i$与多边形的交，时间复杂度为$O(m)$;
- 得到交的区间后，需要$O(n)$染色;
- 总复杂度为$O(n(n+m))$;

- 如果有T个不相交的多边形（凹凸均可）需要染色;
- 时间复杂度为$O(Tn(n+m))$。

点定位

- 如何判断一个点是否在凸多边形内？

点定位

- 如何判断一个点是否在凸多边形内？
  - 取出凸多边形中相邻的边；
  - 用 **叉积** 判断该点是否在这条边的左侧。

点定位

- 如何判断一个点是否在凸多边形内？
  - 取出凸多边形中相邻的边；
  - 用 **叉积** 判断该点是否在这条边的左侧。
- 如何判断一个点是否在凹多边形内？

点定位

- 如何判断一个点是否在凸多边形内？
  - 取出凸多边形中相邻的边；
  - 用 **叉积** 判断该点是否在这条边的左侧。
- 如何判断一个点是否在凹多边形内？
  - 从该点向上做一条射线；
  - 数这条射线穿过凹多边形的多少条边；
  - 奇数条在内部，偶数条在外部。

点定位

- 如何判断一个点是否在凸多边形内？
  - 取出凸多边形中相邻的边；
  - 用 **叉积** 判断该点是否在这条边的左侧。
- 如何判断一个点是否在凹多边形内？
  - 从该点向上做一条射线；
  - 数这条射线穿过凹多边形的多少条边；
  - 奇数条在内部，偶数条在外部。
- 如何判断一个点是否在T个不相交的多边形内？

点定位

- 如何判断一个点是否在凸多边形内？
  - 取出凸多边形中相邻的边；
  - 用 **叉积** 判断该点是否在这条边的左侧。
- 如何判断一个点是否在凹多边形内？
  - 从该点向上做一条射线；
  - 数这条射线穿过凹多边形的多少条边；
  - 奇数条在内部，偶数条在外部。
- 如何判断一个点是否在T个不相交的多边形内？
  - 仍然用射线法！

点定位

- 枚举屏幕上$n \times n$个格点；
- 用射线法判断其是否在T个不相交的多边形内。

点定位

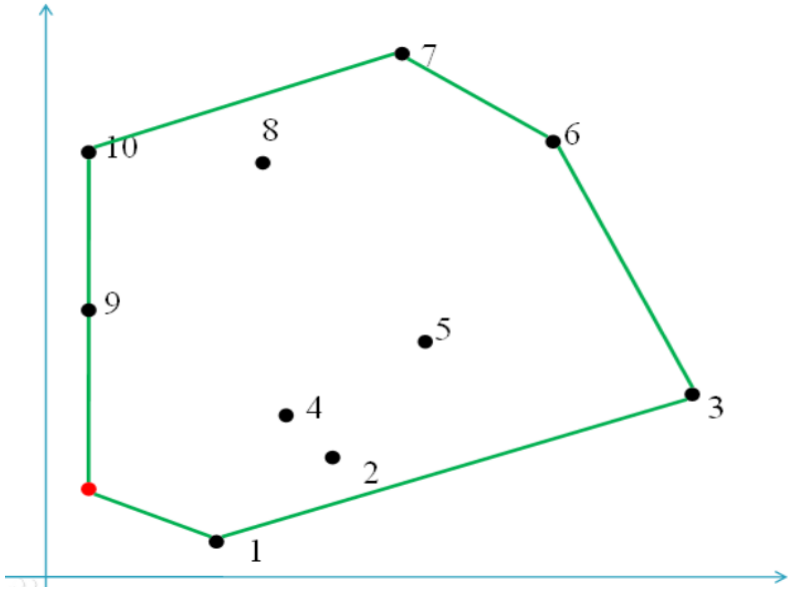
- 枚举屏幕上$n \times n$个格点；
- 用射线法判断其是否在T个不相交的多边形内。
- 可以用线段树实现！
  - 对x轴建立线段树，线段树上每个节点是一个vector；
  - 将T个多边形相邻的边，插入到线段树对应的vector里；
  - 将vector里的线段按照y排序；
  - 询问时，找到对应的vector，二分出该点上方有多少个点。

点定位

- 枚举屏幕上$n \times n$个格点；
- 用射线法判断其是否在T个不相交的多边形内。
- 可以用线段树实现！
  - 对x轴建立线段树，线段树上每个节点是一个vector；
  - 将T个多边形相邻的边，插入到线段树对应的vector里；
  - 将vector里的线段按照y排序；
  - 询问时，找到对应的vector，二分出该点上方有多少个点。
- 总复杂度为$O(n^2 \log^2(Tm))$；
- 两种算法在不同情况下各有千秋！

二维凸包

- 给定平面上n个点A_i，要求从中选出m个点S_i，若将S_i顺次连接为**凸多边形**，使得任意点A_i都在该凸多边形中，则称S_i构成的凸多边形为凸包。
- 凸多边形：相邻角小于180°。



$O(n\log n)$ 算法

- 设已经选出来的点为S_i，共m个点。
- 将A_i按照x轴从小到大排序，从1到n枚举A_i；
- 如果S_{m-1},S_m,A_i大于等于180°，则将S_m删去；
- 重复此过程……
- 当S_{m-1},S_m,A_i小于180°时，将A_i加入S，令其为S_{m+1}。

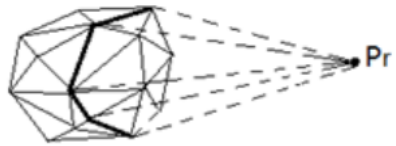
代码

```
    sort(A, A+n, Cmp);
    for (int i = 0; i < n; i ++){
        for (; m >= 2 && cross(A[i] - S[m - 1], S[m - 2] - S[m - 1]) <= 0; m --);
        S[m ++] = A[i];
    }

    for (int i = n - 1, mm = m; i >= 0; i --){
        for (; m > mm && cross(A[i] - S[m - 1], S[m - 2] - S[m - 1]) <= 0; m --);
        S[m ++] = A[i];
    }
    S[m] = S[0];

    double ans = 0;
    for (int i = 1; i <= m; i ++) ans += cross(S[i-1], S[i]);

    cout << ans * 0.5 << endl;
```

增量法求三维凸包

情况1：新点在当前凸包内部，只需简单地忽略该点，如图1所示。

情况2：新点在当前凸包外部，需要计算并的凸包，在这种情况下，首先需要计算原凸包相对于Pr的水平面，即Pr可以看到的封闭区域，如图2所示。



CH(P_{r-1})相对于P_r的水平面

图1

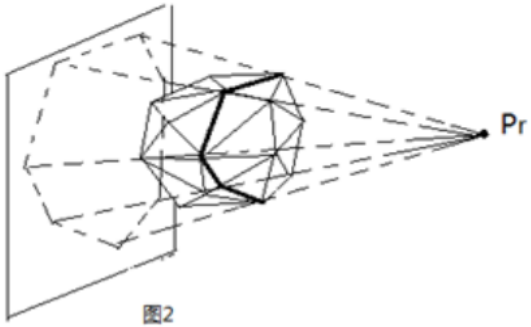
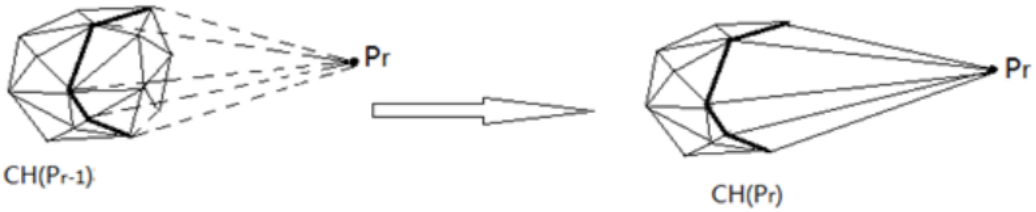


图2



Homework 1

```
struct poi{
    private:
        int x,y;
        // abs(x),abs(y)<=10000
    public:
        poi(int x = 0, int y = 0):x(x),y(y){}
        poi operator - (const poi&A){
            // vector subtraction
            return poi(x-A.x, y-A.y);
        }
        int operator ^ (const poi&A){
            // outer product
            return x*A.y-y*A.x;
        }
};

bool inside(poi A, poi B, poi C, poi D){
    // fill the body of the function
    // determine if D lies in triangle ABC strictly.
}
```