

作者： 夏敏
日期： 2025-10-20
邮箱： xiamin58@qq.com
电话：

湖南长沙

算法分享 - 计算点与直线的距离（3 种算法）

摘 要：此案例，是笔者在做一个项目时遇到的案例。核心就是使用光纤传感器扫描产品边缘任意 2 个点位，生成一条直线函数。再根据数学公式，计算出 XY 伺服模组到达边缘的最短距离以及最近点的坐标。具体工艺保密由于保密需要要求就不分享出来。

这里还想分享一点，就是研究算法时可以多使用工具去计算，随着人工智能的发展，电气工程师的编程不再局限于苦研算法。例如当计算出现难点时，可以前往 DeepSeek 进行搜索，在想验证计算结果时，可以使用 Mathematica、CAD 或者 SolidWorks 等软件。要让这些工具物尽其用，让编程更加得心应手。

关键字：数学算法、人工智能、CAD

免责声明：我们已对本文档描述的内容做测试。但是差错在所难免，无法保证绝对正确并完全满足您的使用需求。本文档的内容可能随时更新，也欢迎您提出改进建议。

附件：

1、源代码见文件夹：FB_PointToLineDistance

更新记录：

1、 无

目录

1	使用直线的一般式方程	2
2	使用直线的斜截式方程	2
3	使用直线上的两个点（简单一点）	3
4	调用方法.....	3

1 使用直线的一般式方程

```

FUNCTION F_General : LREAL // 使用直线的一般式方程
VAR_INPUT
    A, B, C: LREAL; // 直线方程系数  $Ax + By + C = 0$ 
    x0, y0: LREAL; // 点坐标
END_VAR
VAR
    Numerator: LREAL; // 分子
    Denominator: LREAL; // 分母
END_VAR

// 计算分子:  $|Ax_0 + By_0 + C|$ 
Numerator := ABS(A * x0 + B * y0 + C);

// 计算分母:  $\sqrt{A^2 + B^2}$ 
Denominator := Sqrt(A * A + B * B);

// 避免除零错误
IF Denominator = 0 THEN
    F_General := 0.0;
ELSE
    F_General := Numerator / denominator;
END_IF;
    
```

2 使用直线的斜截式方程

```

1  FUNCTION F_SlopeIntercept : LREAL // 使用直线的斜截式方程
2  VAR_INPUT
3      k: LREAL; // 斜率
4      b: LREAL; // 截距
5      x0, y0: LREAL; // 点坐标
6  END_VAR
7  VAR
8      Numerator: LREAL; // 分子
9      Denominator: LREAL; // 分母
10 END_VAR

1 // 转换为一般式:  $y = kx + b \rightarrow kx - y + b = 0$ 
2
3 // 计算分子:  $|k*x_0 - y_0 + b|$ 
4 Numerator := ABS(k * x0 - y0 + b);
5
6 // 计算分母:  $\sqrt{k^2 + 1}$ 
7 Denominator := Sqrt(k * k + 1);
8
9 // 避免除零错误
10 IF Denominator = 0 THEN
11     F_SlopeIntercept := 0.0;
12 ELSE
13     F_SlopeIntercept := Numerator / Denominator;
14 END_IF
    
```

3 使用直线上的两个点（简单一点）

```

1  FUNCTION F_TwoPoints : LREAL // 使用直线上的两个点
2  VAR_INPUT
3      x1, y1:      LREAL; // 直线上的点A坐标
4      x2, y2:      LREAL; // 直线上的点B坐标
5      x0, y0:      LREAL; // 点P坐标
6  END_VAR
7  VAR
8      crossProduct: LREAL; // 向量积
9      abLength:     LREAL; // AB向量的长度
10     dx, dy:        LREAL;
11 END_VAR

1 // 计算向量分量
2 dx := x2 - x1;
3 dy := y2 - y1;
4
5 // 计算叉乘的绝对值: (x2-x1)*(y0-y1) - (y2-y1)*(x0-x1)
6 CrossProduct := ABS(dx * (y0 - y1) - dy * (x0 - x1));
7
8 // 计算AB向量的长度
9 abLength := SQRT(dx * dx + dy * dy);
10
11 // 避免除零错误
12 IF abLength = 0 THEN
13     F_TwoPoints := 0.0;
14 ELSE
15     F_TwoPoints := CrossProduct / abLength;
16 END_IF;
    
```

4 调用方法

