移动机器人跟墙项目实施方案

一、项目实施背景

目前随着社会科技的不断发展，移动机器人已经广泛的应用到生产生活中去，替代人去执行一些重复，危险的工作，例如在发电站中或工厂生产中对线路的进行检测(如图 1，2 所示)，在火灾或者核辐射现场进行搜索救援。在移动机器人对未知环境的探索过程中，以固定距离跟随一侧的墙体是一种较为普遍的探索方式，同时在搜索过程中也可以生成一条全局的工作路径，以便移动机器人实现重复性探测的需求。

图 1 发电站中智能巡线检测

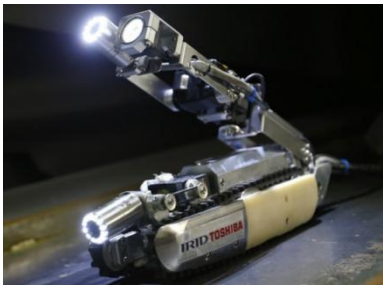

图 2 核辐射搜索救援

本项目就是此为背景进行设计方案和实施对，首先对移动机器人以固定距离跟随墙体进行数学建模（如图 3），移动机器人为图中蓝色的点，移动机器人的行驶方向为θ，在移动

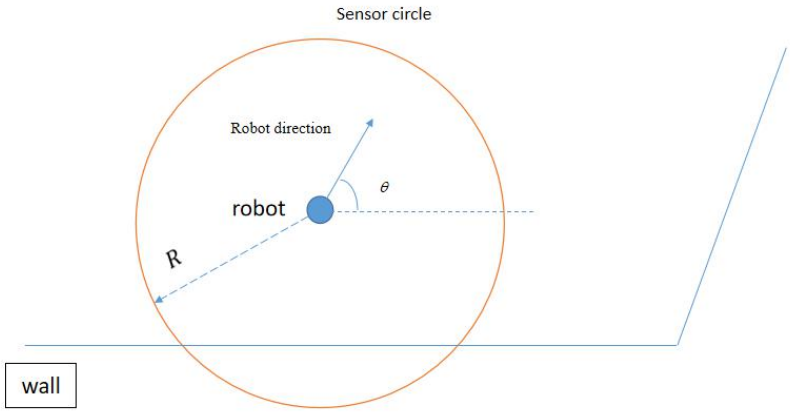

图 3 墙体跟随跟随示意图

机器人上搭载有一个 2D 激光雷达，激光雷达的测量半径为 R，激光雷达的最大测量角为$\phi_{\max}$，最小测量角为$\phi_{\min}$，角分辨率为ϕ_{reso}。图中蓝色的实线为需要跟随的墙体。现在要实现的目标为三个：

第一：移动机器人以固定距离跟随墙体。

第二：在移动机器人跟随完依次之后生成一条可重复性跟随的全局路径。

第三：实现移动机器人对全局工作路径的稳定跟随。

二、项目实施方案

整个方案的结构框图如下所示：

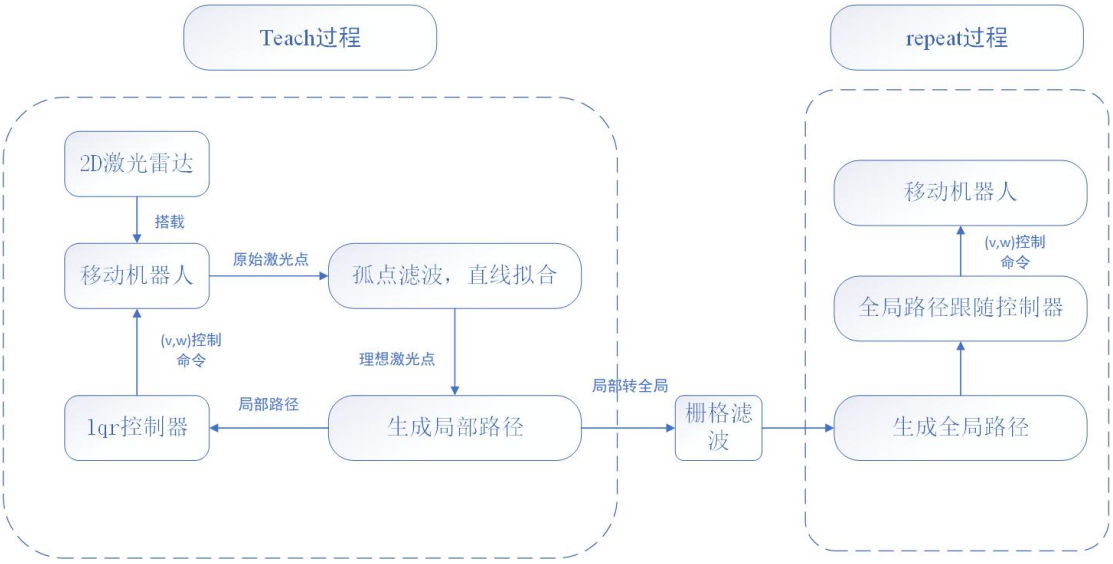

图 4 跟随方案结构框图

整个跟随方案按照 teach-repeat 的导航架构进行设计，teach 部分为移动机器人的教学部分，repeat 部分为移动机器人的可重复性自主导航部分。

移动机器人搭载一个 2D 激光雷达，设移动机器人在 teach 部分跟随右侧的墙体，所以获取右侧的原始激光点，由于墙体的不平滑或者激光传感器的测量误差，原始激光点数据无法直接用于生成局部路径和跟随，需要对原始激光点数据进行孤点滤波和直线拟合，从而完成原始激光点到理想激光点的转变。将理想激光点向移动机器人平移固定跟随距离，生成局部跟随路径，利用 lqr 控制器控制移动机器人对局部路径跟随，从而实现在 teach 阶段移动机器人以固定距离跟随墙体。

假设移动机器人定位准确的前提下，实时将生成局部路径转化成全局路径，并采用栅格滤波，直线拟合方式和移动机器人行驶路径相结合对全局路径进行优化，生成最终优化的全局路径。在 repeat 阶段，利用全局路径跟随控制器实现移动机器人对全局路径的跟随，从而实现对墙体的重复性跟随。

2.1 获取原始激光点

在实际应用中，2D 激光雷达的传输的 scan 消息返回的为激光传感器的最大测量角度，最小测量角度，角度分辨率，以及各个方向上的距离值，在进行孤点滤波之前需要将激光传感器返回的距离转化成激光点坐标。设激光传感器的最大测量角度为 $\phi_{\max}$，最小测量角度为 $\phi_{\min}$，角度分辨率为 ϕ_{reso}，测量半径为 R，由于移动机器人在跟随一侧墙体时容易受到另一侧墙体时干扰，所以距离转化激光点坐标的过程并不是将所有方向的距离都转化成激光点坐标，所以在此设置转化起始角 ϕ_s，转化终止角 ϕ_e，转化距离阈值 $disThres$,它们的关系图

如图 5 所示，它们的关系为：

$$\begin{cases} \phi_{\min} \le \phi_s < \phi_e \le \phi_{\max} \\ distThres <= R \end{cases} \quad (1)$$

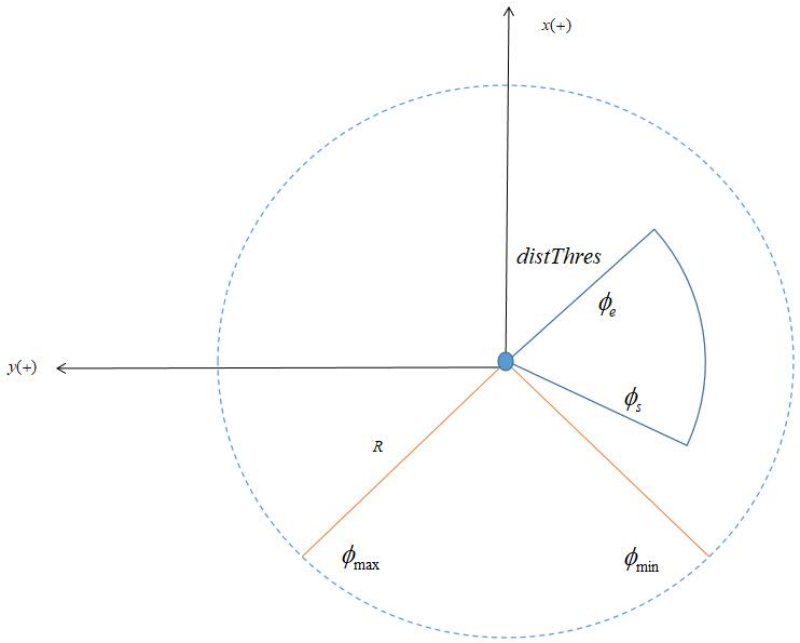

图 5 起始角度，终止角度，转化距离阈值，传感器测量最大角，测量最小角，测量半径关系示意图

当ϕ角包含在起始角度和终止角度所属的区间内，并且ϕ方向的所测距离小于转化距离阈值时，将距离转化成激光点坐标，转化公式为：

$$\begin{cases} x = d_\phi \cdot \cos(\phi) \\ y = d_\phi \cdot \sin(\phi) \end{cases} \quad (2)$$

转化完成之后，将原始的激光点坐标存储进 sensorPoint 列表中，用于之后的孤点滤除和直线拟合。

2.2 孤点滤除，滑动窗降噪和直线拟合

由于激光传感器误差等原因，sensorPoint 中存储的原始激光点坐标中会出现孤点，并且原始的激光点坐标会出现锯齿形波动(如图 6 所示)。

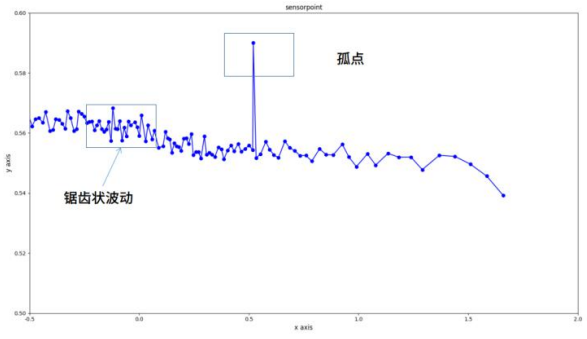

图 6 传感器噪声产生的孤点和锯齿状波动

(1)孤点滤波

针对孤点，采用圆形滤波器，设置滤波距离 *orphanDist* 和滤波阈值 *orphanNum* ，以每个点为圆心，滤波距离为半径(如图 7 所示)，由图 7 能够明显看出，孤点所在滤波圆内其他点的数量远远小于非孤点的数量，当圆内的其他点数量大于滤波阈值时，认为此点不是孤点，保留，否则认为次点是孤点，所以舍去，滤波完成之后见图 8。

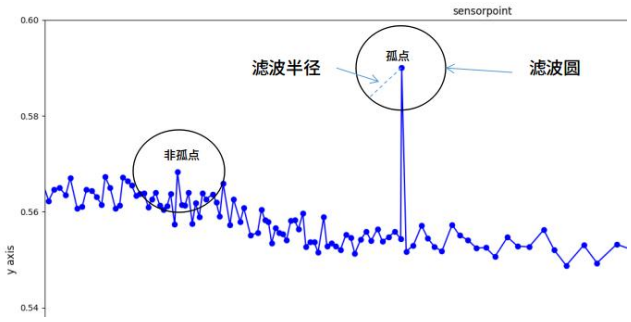

图 7 孤点滤波示意图

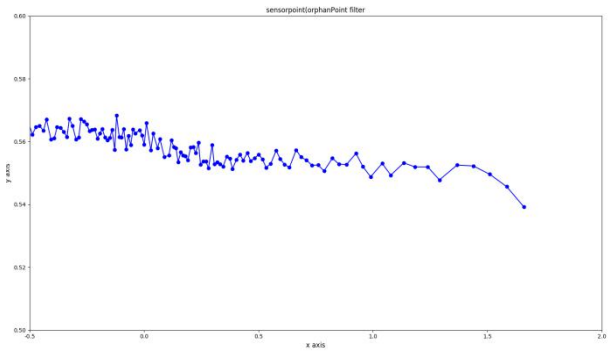

图 8 孤点滤波完成示意图

(2)滑动窗滤波

激光传感器的数据会伴随传感器本身携带噪声，导致激光点坐标出现锯齿状波动，在进行拟合之前，需要对带有噪声的激光点坐标进行降噪处理(如图 9,10 所示)，滑动窗降噪主要采用的是滑动平均法(moving average)又称移动平均法。在简单平均数法基础上，通过顺序逐期增减新旧数据求算移动平均值，借以降低噪声对数据的影响。滑动窗降噪算法执行流程如下所示：

步骤一：设置滑动窗口长度。

步骤二：从 sensorPoint 列表中的左侧开始，依次向右进行滑动，依次求取滑动窗内纵坐标的平均值。

步骤三：当一个激光坐标点的纵坐标有多个平均值时，进行再次平均。

步骤四：将滤波完成后的激光点坐标依次储存到 sensorPointfilter 列表中。

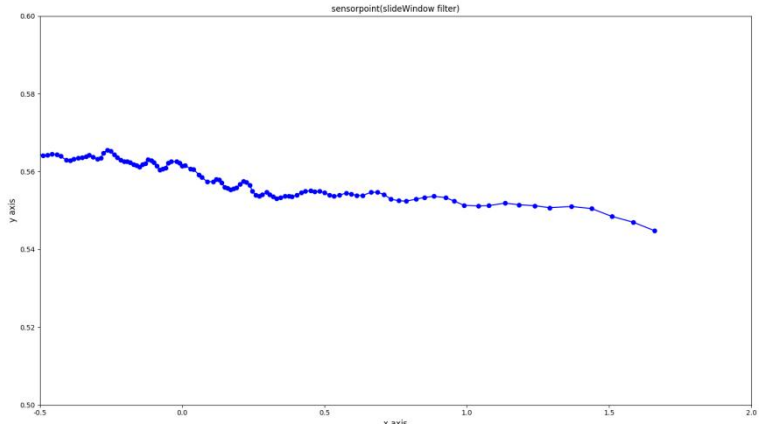

图 9 滑动窗降噪

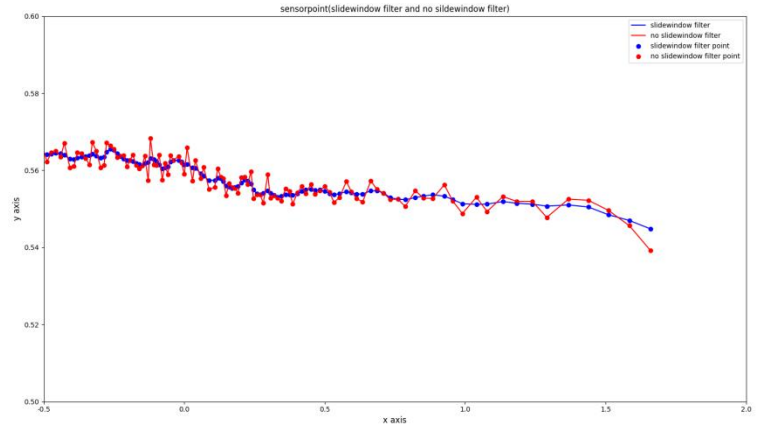

图 10 滑动窗降噪前后数据对比（前：红，后：蓝）

经过滑动窗降噪之后，因传感器噪声带来激光坐标点的数据的锯齿状波动幅度得到有效降低，减小传感器噪声对数据的影响。

(3)直线拟合

降噪完成之后，数据仍存在小幅度波动，为了使激光坐标数据达到理想状态，采用直线拟合对降噪数据进行拟合，消除激光点坐标的小幅度波动(如图 11,12 所示)，直线拟合算法执行流程如下所示：

步骤一：设置直线拟合阈值。

步骤二：连接坐标点列表的首尾两点，生成线段，求取其他点到线段的距离值，判断最大距离值是否小于阈值，如果最大距离值大于阈值，尾点变为最大距离值所对应的点，继续执行步骤二，如果最大距离值小于阈值，拟合成功。

步骤三：判断坐标点列表中所有点是否拟合完成，如果没有拟合完成，剩下的点继续执

行拟合。

步骤四：将拟合成功的激光点坐标存储到 sensorPointfit 列表中，每个激光点坐标所对应的切线斜率即为激光点所在拟合直线的斜率，将每个激光点所对应的切线斜率依次存储到 tangentslope 列表中。

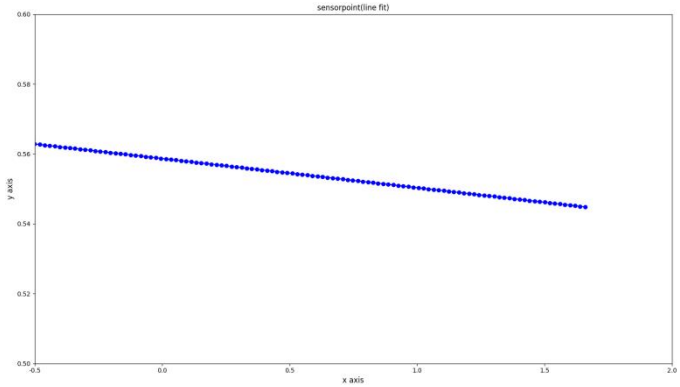

图 11 直线拟合效果图

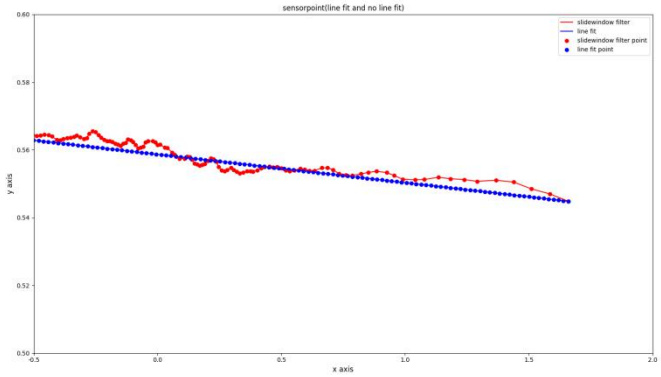

图 12 直线拟合前后数据对比

通过直线拟合，可以生成比较理想的激光点坐标数据，并且能够反映出激光点坐标的趋势。

2.3，生成局部路径

设第 i 个激光点为坐标为(x_i^t, y_i^t),第 i 个激光点所对应的切线斜率为k_i^t，对应的法线斜率为k_i^f，其中$k_i^f = -1/k_i^t$，法线的截距为$b_i^f = y_i^t - x_i^t \cdot k_i^f$。法线点为法线上距激光点为固定跟随距离的点，法线点通过联立以下两个方程求解：

$$\begin{cases} k_i^f \cdot x_i^f + b_i^f = y_i^f \\ (x_i^f - x_i^t)^2 + (y_i^f - y_i^t) = followdist^2 \end{cases} \quad (3)$$

联立得：

$$Ax_i^{f\,2} + Bx_i^f + C = 0 \quad (4)$$

其中：

$$\begin{cases} A = 1 + {k_i^f}^2 \\ B = 2k_i^f b_i^f - 2x_i^t - 2k_i^f y_i^t \\ C = {x_i^t}^2 + {b_i^f}^2 - 2b_i^f y_i^t + {y_i^t}^2 - followdist^2 \end{cases} \tag{5}$$

解得：

$$\begin{cases} x_{i,1}^f, x_{i,2}^f = \dfrac{-B - \sqrt{B^2 - 4AC}}{2A}, \dfrac{-B + \sqrt{B^2 - 4AC}}{2A} \\ y_{i,1}^f, y_{i,2}^f = k_i^f x_{i,1}^f + b_i^f, k_i^f x_{i,2}^f + b_i^f \end{cases} \tag{6}$$

选取到移动机器人距离近的一组坐标点，舍弃较远的一组坐标点。将选取的法线点依次存储到 normalPoint 列表中。但是存在选取的法线点到拟合直线的最短距离小于固定跟随距离(如图 13 所示)，需要滤除掉。

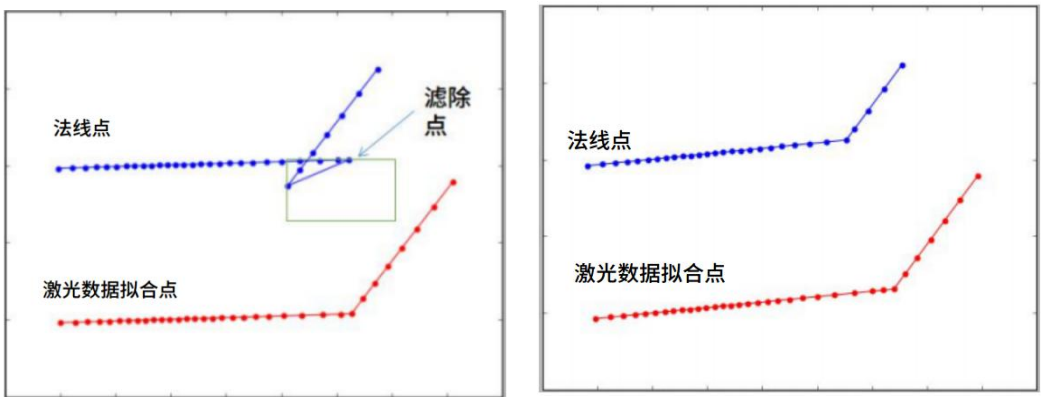

图 13 法线点滤波前（左），法线点滤波后（右）

2.5 lqr 控制器控制

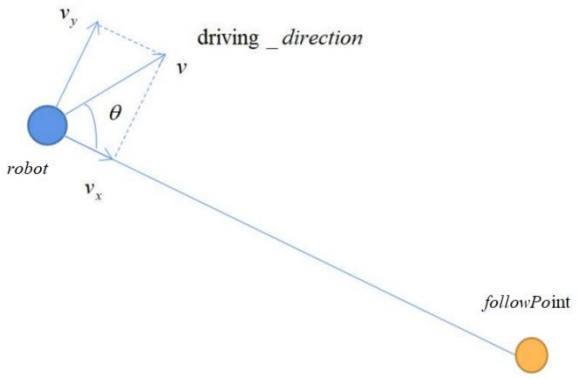

图 14 lqr 控制器示意图

图 14 中蓝色的为移动机器人，橙色的为要跟随的目标点，移动机器人的行驶方向如图所示，

以移动机器人与目标点连线为 x 轴，垂直于 x 轴为 y 轴，移动机器人的速度分别向 x 轴和 y 轴做投影，分别为v_x, v_y，令移动机器人到 x 轴的距离为S，则$\dot{S} = v \cdot \sin\theta = v_y$，$w = \dot{\theta}$，其中$v$为移动机器人的行驶速度，$\theta$为移动机器人行驶方向与 x 轴的夹角,$w$为移动机器人的角速度。系统的状态方程为：

$$\begin{bmatrix} \dot{S} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} S \\ \theta \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w \quad (7)$$

其中：

$$\begin{cases} A = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \\ B = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ x = \begin{bmatrix} S \\ \theta \end{bmatrix} \end{cases} \quad (8)$$

式 7 就转化为：

$$\dot{x} = Ax + Bw \quad (9)$$

令$w = -kx$,其中$k = [k_1, k_2]$,则式 9 就转化为：

$$\dot{x} = (A - Bk)x \quad (10)$$

所以需要得到合适的k,使得$A - Bk$的矩阵特征值为负，所得到的控制量w就会使θ趋向于 0，从而使移动机器人朝着目标点前进。关于k的获取，采用代价函数$J = \int_0^\infty x^T Qx + u^T Rudt$输入 Q，R 矩阵参数，选取产生最小代价函数值得$k$，在具体的程序中，采用 python 语言的 control 工具包中 lqr 函数进行求解k。

所以在移动机器人跟随由法线点组成的局部路径时，采用上述的 lqr 控制器进行控制。示意图如下所示：

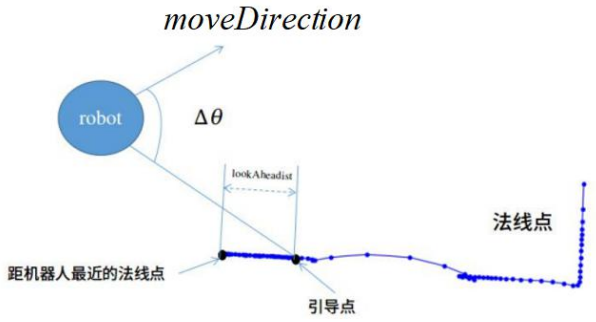

图 15 移动机器人 lqr 跟随示意图

移动机器人选取距移动机器人最近法线点为$lookAhead$dis 的法线点为目标点，采用上述的

lqr 控制器，通过 control 工具包中的 lqr 函数，获取 k 值，从而获取角速度控制量 $w=-kx$，使得移动机器人朝着目标点前进。

关于线速度控制量，根据实际试验结果发现,为了使移动机器人能够稳定的，高精度的以固定距离跟随墙体，移动机器人的线速度的控制命令表达式为 $u=u_d/e^{k_v|\Delta\theta|}$，当移动机器人遇到直角墙面时，应当减速，从而提高移动机器人的跟随精度。

2.6 局部路径转化为全局路径

移动机器人在利用 lqr 控制器以固定跟随距离跟随墙面时，需要实时将全局路径转化为局部路径，用于 repeat 阶段的全局路径跟随，假设移动机器人的定位实时有效，移动机器人的全局位置坐标为 $p=(x_r,y_r,\theta_r)$，局部路径点坐标为 $q=(x_l,y_l)$，根据转化公式所得的全局路径点坐标为：

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix}=\begin{bmatrix} \cos\theta_r & -\sin\theta_r \\ \sin\theta_r & \cos\theta_r \end{bmatrix}\begin{bmatrix} x_l \\ y_l \end{bmatrix}+\begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{11}$$

将生成的全局路径存储到 globalpath 列表中，进行栅格滤波，将栅格地图覆盖到转化的全局路径上面，通过保留含有全局路径坐标点多的栅格，滤除掉包含全局路径点少的栅格，栅格滤波的步骤分为三步，如下所示：

步骤 1，设置栅格大小，根据全局路径点的横纵坐标最大最小值生成栅格地图。

步骤 2，通过查表方式，依次查询每个栅格所包含全局路径点的个数。

步骤 3，设置滤波阈值，当栅格包含的全局路径点的个数大于滤波阈值，保留此栅格，将中心坐标，保存至 path 中，否则滤除此栅格。

栅格滤波完成之后，借助移动机器人 teach 阶段的行驶路径与直线拟合相结合，优化出最优路径，对 path 内的点进行直线拟合，最终生成全局路径，存储到 pathfit,并通过 topic 发布出去。

2.7 全局路径跟随算法

2.7.1 建立 *Serret - Frenet* 坐标系

全局路径跟随算法目前采用基于 *Serret - Frenet* 坐标系下的路径跟随算法，*Serret – Frenet* 坐标系时分析曲线的最重要的工具之一，在全局路径跟随算法中，*Serret - Frenet* 坐标系以全局路径上的目标点为原点，目标点所对应的切线为横轴，垂直于切线的法线为纵轴，横轴的正半轴与移动机器人跟随的方向相关，例如移动机器人从左向右跟随一条全局路径，那么横轴的正半轴在切线的右侧，纵轴的正半轴为横轴正半轴的左侧。随着目标点的不断移动， *Serret - Frenet* 坐标系也在不断的移动。

对于二维曲线，在某一处的曲率大小等于其密切圆半径的倒数，曲率度量了二维曲线偏离密切平面上一条直线的程度，对于曲线 $y=f(x)$，其曲率为 c_c 为：

$$c_c=\frac{\left[1+f'^2(x)\right]^{1/2}f''(x)}{\left[1+f'^2(x)\right]^2} \tag{12}$$

曲率的倒数，称为曲率半径。任意直线的曲率是 0，半径为 r 的圆弧的曲率为常数(等于 $1/r$)。

在明确 *Serret - Frenet* 坐标系定义之后，以给定路径上的参考目标点为原点，引入一个 *Serret - Frenet* 坐标系，如图 11 所示：

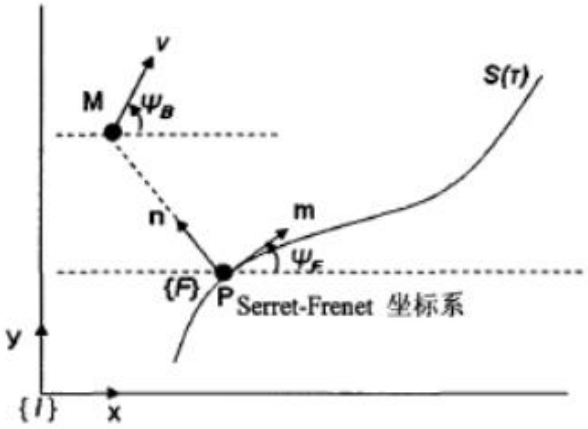

图 16 *Serret - Frenet* 坐标系

在图 16 中有两个坐标系，其中$\{I\}$表示为全局坐标系，$\{F\}$为*Serret – Frenet* 坐标系，移动机器人采用质点 M 表示，位姿信息为(x, y, ψ_B)，目标点采用 P 表示，其位姿信息为(x_r, y_r, ψ_F)，控制目标使移动机器人最终到达目标点的位置，并且时时刻刻跟随目标点，移动机器人转化在 *Serret – Frenet* 坐标系的误差坐标的具体表达式为：

$$\begin{bmatrix} x_e \\ y_e \\ \psi_e \end{bmatrix} = \begin{bmatrix} \cos\psi_F & \sin\psi_F & 0 \\ -\sin_F & \cos\psi_F & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \psi_B - \psi_F \end{bmatrix} \tag{13}$$

如果移动机器人的速度向量为$q = (u, w)^T$，u 为其线速度，w 为其角速度。作为目标参考点 P 的速度向量$q_r = (u_r, w_r)^T$。得到的误差动力学如下：

$$\begin{pmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\psi}_e \end{pmatrix} = \begin{pmatrix} w_r y_e - u_r + u\cos\psi_e \\ -w_r x_e + u\sin\psi_e \\ w - w_r \end{pmatrix} \tag{14}$$

定义有符号的沿曲线横坐标(也称为沿路径的广义弧长)为s，则路径可视为被 s 参数化；定义 *Serret – Frenet* 坐标系原点在曲线上的曲率为$c_c(s)$，坐标系原点相对于 s 的曲率微分为$g_c(s)$，即$\dot{c}_c(s) = g_c(s)\dot{s}$。则有：

(1) 根据 *Serret – Frenet* 坐标系原点瞬时速度对应关系，有参考目标点的运动速度$u_r = \dot{s}$；

(2) 根据曲率的几何定义，$c_c(s)$ 与坐标原点方位角 ψ_r 之间的关系为

$w_r = \dot{\psi}_r = c_c(s)\dot{s}$，因此式可以重写为

$$\begin{pmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\psi}_e \end{pmatrix} = \begin{pmatrix} c_c(s)\dot{s}\, y_e - \dot{s} + u\cos\psi_e \\ -c_c(s)\dot{s}\, x_e + u\sin\psi_e \\ w - c_c(s)\dot{s} \end{pmatrix} \tag{15}$$

2.7.2 设计趋近角

建立好 *Serret – Frenet* 坐标系之后，控制策略主要有两个：

第一：控制移动机器人使得 x_e, y_e 趋向 0.

第二：控制移动机器人的方向误差 ψ_e 趋向于趋近角 δ，趋近角 δ 实际上是方向误差角 ψ_e 的期望角度。

趋近角应满足如下条件：

1、$\delta = \delta(y_e, u)$, 当 $y_e \to 0$ 时，$\delta(y_e, u) \to 0$。

2、$y_e u \sin\delta(y_e, u) \le 0, \forall y_e \in R, \forall u \in R$。

这里，令趋近角 δ 表达为一个挤压函数的形式：

$$\delta(y_e, u) = -\mathrm{sgn}(u)\psi_e \frac{e^{2k_\delta y_e} - 1}{e^{2k_\delta y_e} + 1} \tag{16}$$

其中常数 $k_\delta > 0, \psi_e \in (0, \pi/2)$. 符号函数 $\mathrm{sgn}(u) = \begin{cases} -1 : u < 0 \\ 0 : u > 0 \\ +1 : u > 0 \end{cases}$

由式可以看出，机器人与路径上的参考点之间具有较大纵向误差 y_e，将带来较大趋向角 δ，即给机器人带来较大地指向路径的运动趋势，如果机器人偏离在路径前向方向的左边，趋近角将引导机器人向，从而趋紧路径，法制，趋近角将引导机器人将向左转动从而趋近路径。当机器人运动到路径之上，y_e=0 从而 $\delta(0, u) = 0$，即趋近角也递减为 0，保证机器人不会偏离路径，将只有前向运动而无转向运动。

2.7.3 设计李雅普诺夫函数

在引入趋近角的前提下，设计线性二次型 Lyapunov 函数如下：

$$V = \frac{1}{2}[(x_e^{\,2} + y_e^{\,2}) + \frac{1}{\lambda_e}(\psi_e - \delta)^2] \tag{17}$$

对 Lypunov 函数微分，可得到

$$\dot{V} = (x_e\dot{x}_e + y_e\dot{y}_e) + \frac{1}{\lambda_e}(\psi_e - \delta)(\dot{\psi}_e - \dot{\delta}) \tag{18}$$

将 $Serret-Frenet$ 坐标系下的跟踪误差动力学方程带入上式，

$$\dot{V}=x_e(u\cos\psi_e-\dot{s})+y_e u\sin\delta+\frac{1}{\lambda_e}(\psi_e-\delta)(\dot{\psi}_e-\dot{\delta}+\lambda_e y_e u\frac{\sin\psi_e-\sin\delta}{\psi_e-\delta} \tag{19}$$

若选取如下控制律：

$$\begin{cases}\dot{s}=u\cos\psi_e+k_1x_e\\ \dot{\psi}_e=\dot{\delta}-\lambda_e y_e u\dfrac{\sin\psi_e-\sin\delta}{\psi_e-\delta}-k_2(\psi_e-\delta)\end{cases} \tag{20}$$

其中 k_1,k_2,λ_e 为正常数。则

$$\dot{V}=-k_1{x_e}^2+y_e u\sin\delta-\frac{k_2}{\lambda_e}(\psi_e-\delta)^2\le 0 \tag{21}$$

由于 $\dot{\psi}_e=\dot{\psi}_B-\dot{\psi}_F=r-c_c(s)\dot{s}$，令 u_d 为期望的机器人跟踪速度，推演得到的路径跟踪运动学控制器可描述为：

$$\begin{cases}u=u_d\\ \dot{s}=k_1x_e+u\cos\psi_e\\ w=\dot{\delta}-\lambda_e y_e u\dfrac{\sin\psi_e-\sin\delta}{\psi_e-\delta}-k_2(\psi_e-\delta)+c_c(s)\dot{s}\end{cases} \tag{22}$$

第一项表示运动学阶段假定机器人能“完美”跟踪预期的线速度要求，第二项是在路径上移动的虚拟目标点根据机器人速度，第三项通过调整角速度，减小移动机器人的纵向误差 y_e，至此，我们得到了非完整性移动机器人的运动学路径跟踪控制律。