南開大學
Nankai University

南　　开　　大　　学

人 工 智 能 学 院

智能工程课程大作业

专　　业: 智能科学与技术
姓　　名: 石若川
学　　号: 2111381
指导教师: 孙雷
完成日期: 2024 年 5 月 29 日

目录

1 问题描述与运动学建模

一叉车式移动机器人有三个轮子组成，其中：两个随动的固定标准轮，轮 A 沿 X_R 轴正方向，且 $\alpha = -\pi/2, \beta = \pi, l = 1$, 轮 B 沿 X_R 轴正方向，且 $\alpha = \pi/2, \beta = 0, l = 1$；一个受控（带电机驱动、主动轮）的转向标准轮，$\alpha = 0, \beta = \pi/2, l = 2$，试求此机器人运动学方程。

ICR 方法

设驱动轮的半径为 r，车体瞬时旋转半径为 R，驱动轮的转速为 $\dot{\varphi}$。

由几何关系可得转弯半径 $R = \frac{L_2}{\cos\beta}$，由此可得

$$\dot{\theta}_R = -\frac{r\dot{\varphi}}{R} = -\frac{r\dot{\varphi}}{L_2}\cos\beta$$

由于随动轮存在滑动约束，所以 $\dot{y}_R = 0$ 由于车体是刚体，各点的角速度相同，有

$$\frac{\dot{x_R}}{R\sin\beta} = \frac{r\dot{\varphi}}{R}$$

得

$$\dot{x_R} = r\dot{\varphi}\sin\beta$$

所以运动学方程为

$$\dot{\xi}_R = \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = r\dot{\varphi}\begin{bmatrix} \sin\beta \\ 0 \\ -\frac{\cos\beta}{L_2} \end{bmatrix}$$

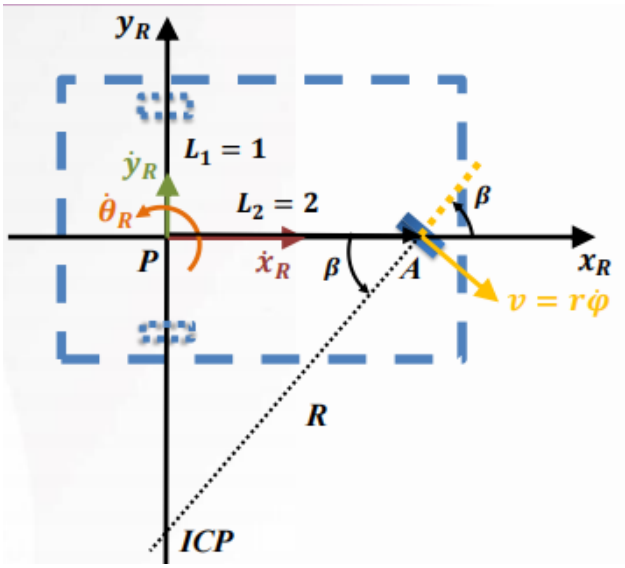


图 1.1: 叉车示意图

运动约束方法

对于一个驱动轮，其运动约束包括滚动和滑动约束。对于两个随动轮，由于两轮不独立，所以只需要一个滑动约束。

主动轮的滚动约束方程为:

$$[\sin(\alpha+\beta) \quad -\cos(\alpha+\beta) \quad -L_2\cos\beta]R(\theta)\dot{\xi}_I - r\dot{\varphi} = 0$$

代入 $\alpha = 0$ 可得：

$$[\sin\beta \quad -\cos\beta \quad -L_2\cos\beta]R(\theta)\dot{\xi}_I - r\dot{\varphi} = 0 \tag{1}$$

主动轮的滑动约束为：

$$[\cos(\alpha+\beta) \quad \sin(\alpha+\beta) \quad L_2\sin\beta]R(\theta)\dot{\xi}_I = 0$$

代入 $\alpha = 0$ 可得：

$$[\cos\beta \quad \sin\beta \quad L_2\sin\beta]R(\theta)\dot{\xi}_I = 0 \tag{2}$$

随动轮的滑动约束为：

$$[\cos(\alpha+\beta) \quad \sin(\alpha+\beta) \quad L_2\sin\beta]R(\theta)\dot{\xi}_I = 0$$

代入 $\alpha = \pi, \beta = 0$，可得：$[0 \quad 1 \quad 0]R(\theta)\dot{\xi}_I = 0$，因此有

$$\dot{y}_R = 0 \tag{3}$$

由式1、2、3可得以下线性方程组

$$\begin{bmatrix} \sin\beta & -\cos\beta & -L_2\cos\beta \\ \cos\beta & \sin\beta & L_2\sin\beta \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \dot{x_R} \\ \dot{y_R} \\ \dot{\theta_R} \end{bmatrix} = \begin{bmatrix} r\dot{\varphi} \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

由方程组4可得运行学方程

$$\dot{\xi}_R = \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = r\dot{\varphi} \begin{bmatrix} \sin\beta \\ 0 \\ -\frac{\cos\beta}{L_2} \end{bmatrix} \tag{5}$$

可以发现通过 ICR 方法和运动约束方法得到的运动学模型一致，如式5所示。

2 机器人定点控制

2.1 定点控制问题描述

定点控制的控制目标是设计一个控制器驱动移动机器人从当前位姿达到预定的目标位姿。

- 定义系统开环误差信号：令 $q_r = [x_r, y_r, \theta_r]^T \in \mathbb{R}^3$ 为参考状态，$\widetilde{q} = \left[\widetilde{x}, \widetilde{y}, \widetilde{\theta}\right]^T \in \mathbb{R}^3$ 为开环误差，其中

$$\widetilde{x} = x - x_r, \widetilde{y} = y - y_r, \widetilde{\theta} = \theta - \theta_r$$

- 定义机器人坐标下的误差信号：利用坐标系变换矩阵可得

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = R(\theta)\widetilde{q} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{\theta} \end{bmatrix}$$

- 设计控制阵 K

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \end{bmatrix}$$

其中 $k_{ij} = k(t, e)$

- 机器人的控制输入

$$\begin{bmatrix} v(t) \\ \omega(t) \end{bmatrix} = Ke = K \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}_R$$

- 最终实现

$$\lim_{t\to\infty} e(t) = 0$$

2.2 定点控制闭环系统误差模型

在设计控制器前，首先要求取闭环系统误差模型。

- 根据运动学方程5可知 $y_R = 0$，令 $v_1 = \dot{x}_R, v_2 = \dot{\theta}_R$
- 求取闭环系统误差模型

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = R(\theta)\widetilde{q}' + R'(\theta)\widetilde{q} = \begin{bmatrix} -\dot{\theta}\sin\theta & \dot{\theta}\cos\theta & 0 \\ -\dot{\theta}\cos\theta & -\dot{\theta}\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \widetilde{x} \\ \widetilde{y} \\ \widetilde{\theta} \end{bmatrix} + \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\widetilde{x}} \\ \dot{\widetilde{y}} \\ \dot{\widetilde{\theta}} \end{bmatrix}$$

其中

$$\begin{aligned} \dot{e_2}q &= -\dot{\theta}\sin\theta\widetilde{x} + \dot{\theta}\cos\theta\widetilde{y} + \cos\theta\dot{\widetilde{x}} + \sin\theta\dot{\widetilde{y}} \\ &= -\dot{\theta}\sin\theta\widetilde{x} + \dot{\theta}\cos\theta\widetilde{y} + \cos\theta\dot{x} + \sin\theta\dot{y} \\ &= \dot{\theta}e_2 + \dot{x_R} \\ &= v_1 + v_2 e_2 \end{aligned} \tag{6}$$

$$\begin{aligned} \dot{e_2} &= -\dot{\theta}\cos\theta\widetilde{x} - \dot{\theta}\sin\theta\widetilde{y} - \sin\theta\dot{\widetilde{x}} + \cos\theta\dot{\widetilde{y}} \\ &= -\dot{\theta}\cos\theta\widetilde{x} - \dot{\theta}\sin\theta\widetilde{y} - \sin\theta\dot{x} + \cos\theta\dot{y} \\ &= -\dot{\theta}e_1 + \dot{y}_R \\ &= -v_2 e_1 \end{aligned} \tag{7}$$

$$\dot{e_3} = \dot{\tilde{\theta}} = \dot{\theta} = v_2 \tag{8}$$

由式6、7、8可得闭环系统误差模型

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} v_1 + v_2 e_2 \\ -v_2 e_1 \\ v_2 \end{bmatrix} \tag{9}$$

2.3 非线性定点控制器稳定性判别

对于闭环系统误差模型9，设计如下非线性控制器

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} -k_1 e_1 \\ -k_2 e_3 + e_2^2 \sin(t) \end{bmatrix} \tag{10}$$

将非线性控制器10代入闭环系统误差模型9，可得

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} -k_1 e_1 + v_2 e_2 \\ -v_2 e_1 \\ -k_2 e_3 + e_2^2 \sin(t) \end{bmatrix} \tag{11}$$

2.4 非线性定点控制器稳定性判别

首先构造正定的李雅普诺夫函数 V_1

$$V_1 = \frac{1}{2} e_1^2 + \frac{1}{2} e_2^2$$

根据式11，该李雅普诺夫函数的导数为

$$\dot{V_1} = e_1 \dot{e_1} + e_2 \dot{e_2} = e_1(-k_1 e_1 + v_2 e_2) + e_2(-v_2 e_1) = -k_1 e_1^2$$

该李雅普诺夫函数的导数为负定的，因此该系统是渐进稳定的。

2.5 非线性定点控制器仿真结果

按照闭环系统误差模型和设计的控制器，在 Matlab Simulink 中搭建仿真模型，如图2.2所示。仿真中设置的参数如下

- 目标点：$x = 15, y = 10, \theta = 0.2$
- 控制器参数：$k1 = 8.2, k2 = 1.8$

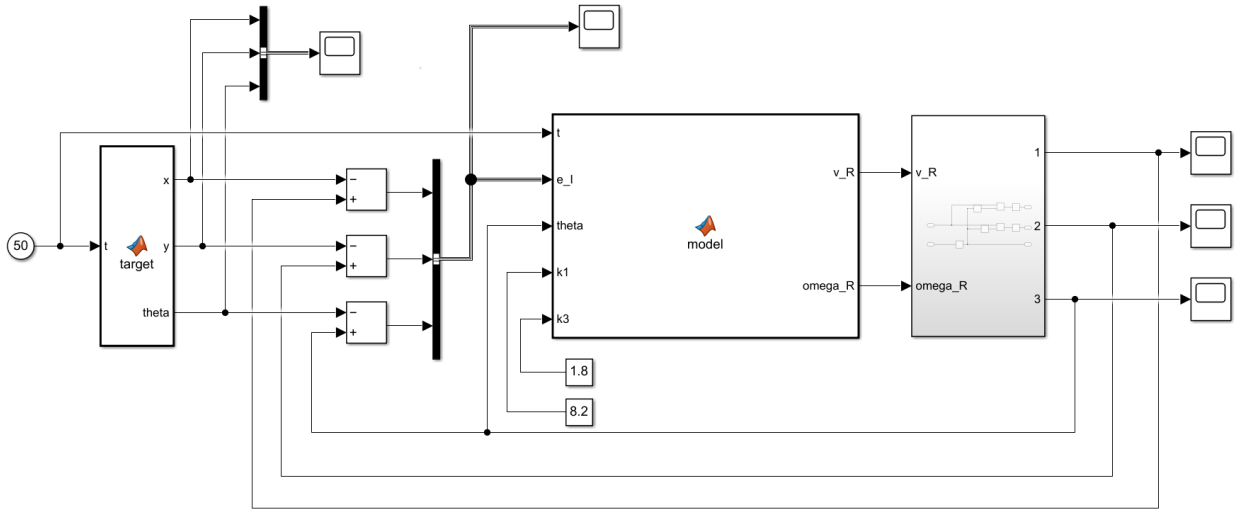


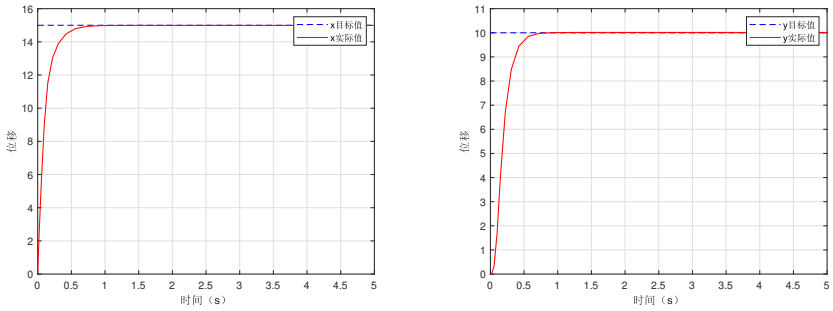
图 2.2: 定点控制仿真模型

模型中 target 模块用于生成目标点位姿，model 模块用于进行惯性坐标系到机器人坐标系的变换以及控制器搭建，代码如下所示。

```
function [x, y, theta] = target(t)

%定点控制
x=15;
y=10;
theta=0.2;
```

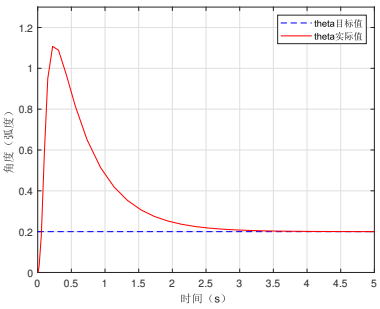
```
function [v_R,omega_R] = model(t,e_I,theta,k1,k3)

ex_I = e_I(1);
ey_I = e_I(2);
etheta_I = e_I(3);

%坐标系变换
e_R=[cos(theta),sin(theta),0;
    sin(theta),cos(theta),0;
    0, 0, 1]*[ex_I;ey_I;etheta_I];

%控制器
v_R=-k1*e_R(1);
omega_R=e_R(2)^2*sin(t)-k3*e_R(3);
```

图2.3为仿真实验结果，分别绘制了 X 和 Y 方向的位移、位姿的 θ 角度随时间变化的曲线，以及误差随时间变化的曲线。可以看出仿真实验中，叉车可以在 1.5s 左右基本到达设定的目标位姿，稳态误差很小。这说明所设计的非线性控制器具有较好的性能，能够完成叉车的定点控制。

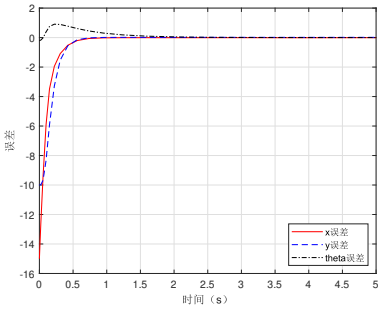


(a) X 方向

(b) Y 方向



(c) θ 方向



(d) 误差变化曲线

图 2.3: X、Y、θ 方向的仿真结果及误差曲线

图2.4展示了定点控制过程中，叉车从原点运动到目标点的轨迹，并绘制了最终小车的位置与姿态。

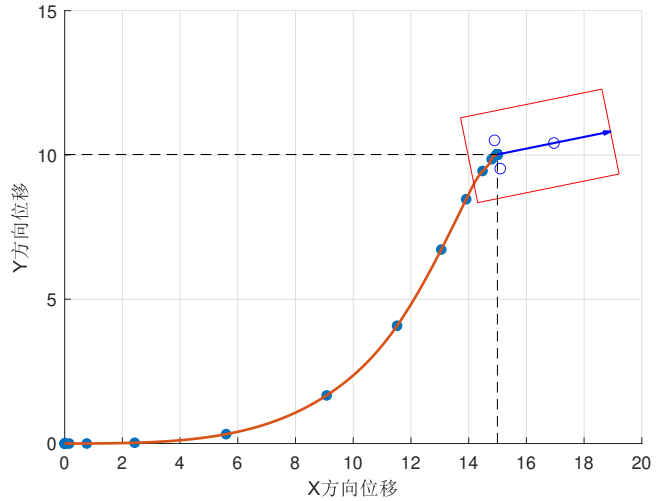


图 2.4: 叉车移动轨迹

3 机器人轨迹跟踪

3.1 轨迹跟踪问题描述

轨迹跟踪的控制目标是设计一个控制器驱动移动机器人从当前位姿跟踪一条给定的轨迹。

- 定义系统开环误差信号：令 $q_r(t)=[x_{rc}(t),y_{rc}(t),\theta_r(t)]^T\in\mathbb{R}^3$ 为参考状态，$\widetilde{q}=[\widetilde{x(t)},\widetilde{y(t)},\widetilde{\theta(t)}]^T\in\mathbb{R}^3$ 为开环误差

$$\widetilde{x(t)}=x(t)-x_{rc}(t),\widetilde{y(t)}=y(t)-y_{rc}(t),\widetilde{\theta(t)}=\theta(t)-\theta_r(t)$$

- 定义机器人坐标下的误差信号：利用坐标系变换矩阵可得

$$e=\begin{bmatrix}e_1\\e_2\\e_3\end{bmatrix}=R(\theta)\widetilde{q}=\begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}\widetilde{x}\\\widetilde{y}\\\widetilde{\theta}\end{bmatrix}$$

- 设计控制阵 K

$$K=\begin{bmatrix}k_{11}&k_{12}&k_{13}\\k_{21}&k_{22}&k_{23}\end{bmatrix}$$

其中 $k_{ij}=k(t,e)$

- 机器人的控制输入

$$\begin{bmatrix}v(t)\\\omega(t)\end{bmatrix}=Ke=K\begin{bmatrix}x\\y\\\theta\end{bmatrix}_R$$

- 最终实现

$$\lim_{n\to\infty}\widetilde{q}=0$$

3.2 轨迹跟踪闭环系统误差模型

与定点控制一样，在设计控制器前，首先要求取闭环系统误差模型。

- 根据运动学方程5可知 $y_R=0$，令 $v_1=\dot{x}_R,v_2=\dot{\theta}_R$
- 求取闭环系统误差模型

$$\begin{bmatrix}\dot{e}_1\\\dot{e}_2\\\dot{e}_3\end{bmatrix}=R(\theta)\widetilde{q}'+R'(\theta)\widetilde{q}=\begin{bmatrix}-\dot{\theta}\sin\theta&\dot{\theta}\cos\theta&0\\-\dot{\theta}\cos\theta&-\dot{\theta}\sin\theta&0\\0&0&0\end{bmatrix}\begin{bmatrix}\widetilde{x}\\\widetilde{y}\\\widetilde{\theta}\end{bmatrix}+\begin{bmatrix}\cos\theta&\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix}\begin{bmatrix}\dot{\widetilde{x}}\\\dot{\widetilde{y}}\\\dot{\widetilde{\theta}}\end{bmatrix}$$

其中

$$\begin{aligned}\dot{e_1}&=-\dot{\theta}\sin\theta\widetilde{x}+\dot{\theta}\cos\theta\widetilde{y}+\cos\theta\dot{\widetilde{x}}+\sin\theta\dot{\widetilde{y}}\\&=\dot{\theta}e_2+\cos\theta(\dot{x}-\dot{x_r})+\sin\theta(\dot{y}-\dot{y_r})\\&=\dot{\theta}e_2+(\cos\theta\dot{x}+\sin\theta\dot{y})-(\cos\theta\dot{x_r}+\sin\theta\dot{y_r})\\&=\dot{\theta}e_2+\dot{x_R}-(v_{1r}\cos\theta\cos\theta_r+v_{1r}\sin\theta\sin\theta_r)\\&=\dot{\theta}e_2+\dot{x_R}-v_{1r}\cos(\theta-\theta_r)\\&=v_1+v_2e_2-v_{1r}\cos e_3\end{aligned}\tag{12}$$

$$\begin{aligned}\dot{e_2} &= -\dot{\theta}\cos\theta\widetilde{x} - \dot{\theta}\sin\theta\widetilde{y} - \sin\theta\dot{\widetilde{x}} + \cos\theta\dot{\widetilde{y}} \\ &= -\dot{\theta}\cos\theta\widetilde{x} - \dot{\theta}\sin\theta\widetilde{y} - \sin\theta(\dot{x} - \dot{x_r}) + \cos\theta(\dot{y} - \dot{y_r}) \\ &= -\dot{\theta}e_1 + (\cos\theta\dot{y} - \sin\theta\dot{x}) + (\sin\theta\dot{x_r} - \cos\theta\dot{y_r}) \\ &= -\dot{\theta}e_1 + \dot{y}_R + (v_{1r}\sin\theta\cos\theta_r - v_{1r}\cos\theta\sin\theta_r) \\ &= -\dot{\theta}e_1 + \dot{y}_R + v_{1r}\sin(\theta - \theta_r) \\ &= -v_2e_1 + v_{1r}\sin e_3 \end{aligned} \tag{13}$$

$$\dot{e_3} = \dot{\widetilde{\theta}} = \dot{\theta} - \dot{\theta_r} = v_2 - v_{2r} \tag{14}$$

由式12、13、14可得闭环系统误差模型

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} v_1 + v_2e_2 - v_{1r}\cos e_3 \\ -v_2e_1 + v_{1r}\sin e_3 \\ v_2 - v_{2r} \end{bmatrix} \tag{15}$$

3.3 非线性轨迹跟踪控制器设计

对于误差模型15，设计如下非线性控制器

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} -k_1e_1 + v_{1r}\cos e_3 \\ -v_{1r}\frac{\sin e_3}{e_3}e_2 - k_2e_3 + v_{2r} \end{bmatrix} \tag{16}$$

将非线性控制器16代入闭环误差模型15可得

$$\begin{bmatrix} \dot{e}_1 \\ \dot{e}_2 \\ \dot{e}_3 \end{bmatrix} = \begin{bmatrix} -k_1e_1 + v_2e_2 \\ -v_2e_1 + v_{1r}\sin e_3 \\ -k_2e_3 - v_{1r}\frac{\sin e_3}{e_3}e_2 \end{bmatrix} \tag{17}$$

3.4 非线性轨迹跟踪控制器稳定性判别

首先构造正定的李雅普诺夫函数 V_1

$$V_1 = \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 + \frac{1}{2}e_3^2$$

根据式17，该李雅普诺夫函数的导数为

$$\begin{aligned}\dot{V_1} &= e_1\dot{e_1} + e_2\dot{e_2} + e_3\dot{e_3} \\ &= e_1(-k_1e_1 + v_2e_2) + e_2(-v_2e_1 + v_{1r}\sin e_3) + e_3(-k_2e_3 - v_{1r}\frac{\sin e_3}{e_3}e_2) \\ &= -k_1e_1^2 - k_2e_3^2 \end{aligned}$$

该李雅普诺夫函数的导数为负定的，因此该系统是渐进稳定的。

3.5 非线性轨迹跟踪控制器仿真结果

按照闭环系统误差模型和设计的控制器，在 Matlab Simulink 中搭建仿真模型，如图3.5所示。仿真中设置的参数如下

- 目标轨迹：实验中设定了一条正弦函数的轨迹，如下所示。

$$\begin{cases} x(t) = t \\ y(t) = \sin t \\ \theta(t) = \arctan(\cos t) \end{cases}$$

其中 θ 的确定是根据 x 与 y 轨迹的斜率确定的。

- 控制器参数：$k_1 = 25, k_2 = 20$

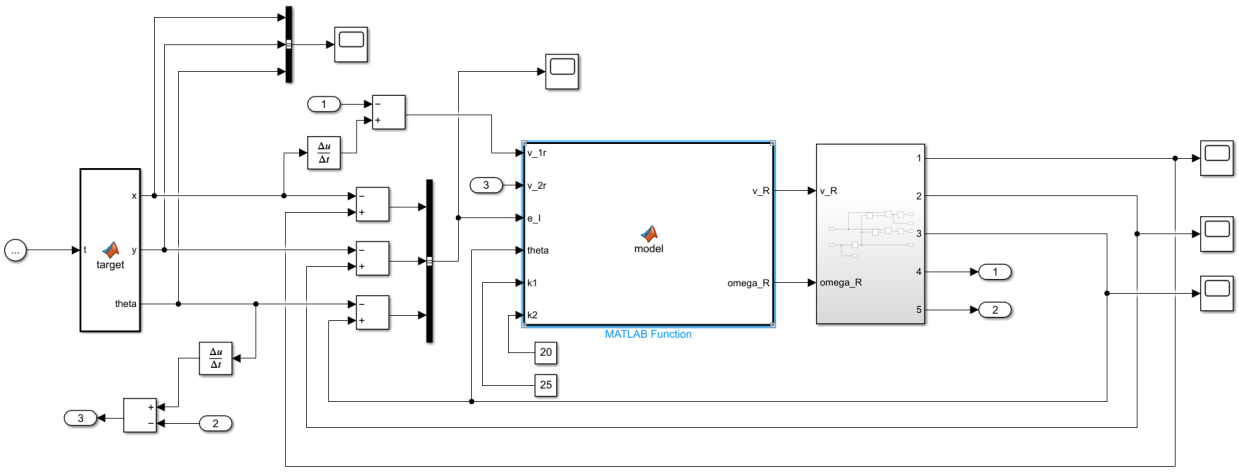


图 3.5: 轨迹跟踪仿真模型

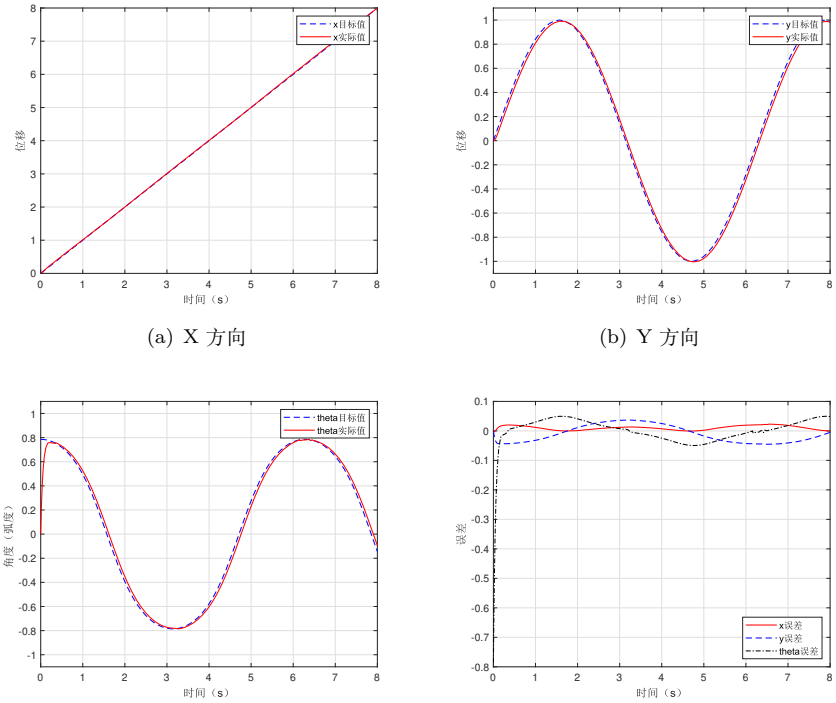
模型中 target 模块用于生成目标点位姿，model 模块用于进行惯性坐标系到机器人坐标系的变换以及控制器搭建，代码如下所示。

```
function [x, y, theta] = target(t)

% 轨迹跟踪
x = t;
y = sin(t);
theta = atan(cos(t));
```

```
function [v_R, omega_R] = model(v_1r, v_2r, e_I,theta,k1,k2)

ex_I = e_I(1);
ey_I = e_I(2);
etheta_I = e_I(3);

```

```
%坐标系变换
e_R=[cos(theta),sin(theta),0;
    -sin(theta),cos(theta),0;
    0, 0, 1]*[ex_I;ey_I;etheta_I];

%控制器
v_R=-k1*e_R(1)+v_1r*cos(e_R(3));
omega_R=-v_1r * e_R(2) * sin(e_R(3)) / (e_R(3)+10^(-6))- k2 * e_R(3) + v_2r;
```

图3.6为仿真实验结果，分别绘制了 X 和 Y 方向的位移、位姿的 θ 角度随时间变化的曲线，以及误差随时间变化的曲线。可以看出仿真实验中，系统能在 0.2s 内迅速跟踪上目标轨迹，系统的 x, y, θ 的稳态误差均在 0.1 以下。这说明所设计的非线性控制器具有较好的性能，能够完成叉车的轨迹跟踪。



(a) X 方向 (b) Y 方向

(c) θ 方向 (d) 误差变化曲线

图 3.6: X、Y、θ 方向的仿真结果及误差曲线

图3.7展示了轨迹跟踪过程中，叉车从原点开始跟踪目标轨迹的实际轨迹点，并对实际轨迹点进行了拟合。可以发现实际轨迹与目标轨迹几乎重合，这说明设计的非线性控制器控制效果较好。实际轨迹与目标轨迹相比有一定的相位滞后，这是由于控制器基于误差工作，本身就具有一定的滞后性。

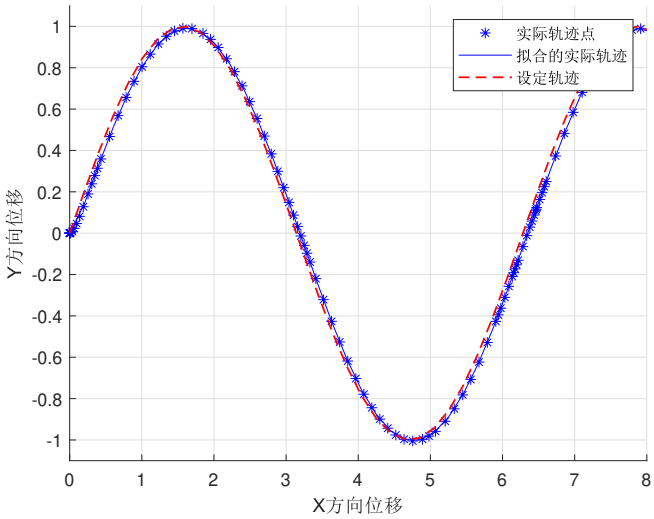


图 3.7: 叉车移动轨迹

4 机器人里程计模型

4.1 里程计模型建模

里程计（odometer）是一种用于测量车辆行驶里程的装置或系统，利用车辆的轮速传感器来监测车轮的转动，并将转动次数转换为行驶里程。

由式5所建立的叉车运动学模型，可得码盘读数为

$$\begin{bmatrix}\Delta s\\ \Delta\theta\end{bmatrix}=\begin{bmatrix}\Delta\varphi r\sin\beta\\ -\Delta\varphi\frac{r\cos\beta}{L_2}\end{bmatrix} \tag{18}$$

利用二阶 Runge-Kutta 法对里程计模型进行求解可得

$$\begin{cases}x_{k+1}=x_k+v_kT_scos\left(\theta_k+\frac{\omega_kT_s}{2}\right)\\ y_{k+1}=y_k+v_kT_ssin\left(\theta_k+\frac{\omega_kT_s}{2}\right)\\ \theta_{k+1}=\theta_k+\omega_kT_s\end{cases} \tag{19}$$

其中 $\omega_kT_s=\Delta\theta$，$v_kT_s=\Delta s$。因此，将码盘读数18代入里程计模型19可得

$$\begin{cases}x_{k+1}=x_k+\Delta\varphi r\sin\beta cos\left(\theta_k-\frac{\Delta\varphi r\cos\beta}{2L_2}\right)\\ y_{k+1}=y_k+\Delta\varphi r\sin\beta sin\left(\theta_k-\frac{\Delta\varphi r\cos\beta}{2L_2}\right)\\ \theta_{k+1}=\theta_k-\frac{\Delta\varphi r\cos\beta}{L_2}\end{cases} \tag{20}$$

4.2 误差传导模型建模

里程计在进行测量移动距离和角度时存在误差，误差的主要来源包括：

- 积分期间分辨率有限（时间增量、测量分辨率）

- 车轮位置的偏移
- 车轮的直径不相等
- 车轮接触点发生变化
- 车轮与地面的接触不均匀，如地面不平整、车轮打滑

令 $p' = [x', y', \theta']$ 为里程计更新后的位姿，则可以得到：

$$p' = f(x, y, \theta, \Delta\phi_R, \Delta\phi_L) = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} \Delta\varphi r \sin\beta cos\left(\theta_k - \frac{\Delta\varphi r\cos\beta}{2L_2}\right) \\ \Delta\varphi r \sin\beta sin\left(\theta_k - \frac{\Delta\varphi r\cos\beta}{2L_2}\right) \\ -\frac{\Delta\varphi r\cos\beta}{L_2} \end{bmatrix} \tag{21}$$

假设：

- 在起始点初始协方差阵已知，即 $\sum_p$ 已知，实验中设定起始点处 $\sum_p = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- 各个驱动轮的两个误差是独立的
- 轮速误差的方差正比与轮速增量，转角误差的方差为常数，即

$$\sum_{\Delta} = covar(\Delta\varphi, \beta) = \begin{bmatrix} k_\varphi \|\Delta\varphi\| & 0 \\ 0 & k_\beta \end{bmatrix}$$

则误差传导函数包括两部分，第一部分为里程计读数导致 $[x, y, \theta]$ 的误差，第二部分为车轮导致 $[\Delta\varphi, \beta]$ 的误差。

$$\sum_{p'} = \nabla_p f \cdot \sum_{p} \cdot \nabla_p f^T + \nabla_\varphi f \cdot \sum_{\Delta} \cdot \nabla_\varphi f^T \tag{22}$$

其中 $\nabla_p f$ 为函数 f 对 $[x, y, \theta]$ 的偏导，$\nabla_\varphi f$ 为函数 f 对 $[\Delta\varphi, \beta]$ 的偏导。

令 $F_p = \nabla_p f$ 有

$$F_p = \nabla_p f = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial \theta} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial \theta} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial \theta} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -r\Delta\varphi \sin\beta \sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) \\ 0 & 1 & r\Delta\varphi \sin\beta \cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) \\ 0 & 0 & 1 \end{bmatrix}$$

代入式18即可得到

$$F_p = \nabla_\varphi f = \begin{bmatrix} 1 & 0 & -\Delta s \sin\left(\theta_k + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & \Delta s \cos\left(\theta_k + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{23}$$

令 $F_\Delta = \nabla_\varphi f$ 有

$$F_\Delta = \nabla_\varphi f = \begin{bmatrix} \frac{\partial f_1}{\partial \Delta\varphi} & \frac{\partial f_1}{\partial \beta} \\ \frac{\partial f_2}{\partial \Delta\varphi} & \frac{\partial f_2}{\partial \beta} \\ \frac{\partial f_3}{\partial \Delta\varphi} & \frac{\partial f_3}{\partial \beta} \end{bmatrix} \tag{24}$$

其中，

$$\begin{bmatrix} \frac{\partial f_1}{\partial \Delta\varphi} \\ \frac{\partial f_2}{\partial \Delta\varphi} \\ \frac{\partial f_3}{\partial \Delta\varphi} \end{bmatrix} = \begin{bmatrix} rsin\beta\cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) + r\Delta\varphi\sin\beta\frac{rcos\beta}{2L_2}\sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) \\ r\sin\beta\sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) - r\Delta\varphi\sin\beta\frac{r\cos\beta}{2L_2}\cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) \\ -\frac{r\cos\beta}{2} \end{bmatrix}$$

$$= \begin{bmatrix} r\sin\beta\left[\cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) + \frac{r\cos\beta}{2L_2}\Delta\varphi\sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right)\right] \\ r\sin\beta\left[\sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) - \frac{r\cos\beta}{2L_2}\Delta\varphi\cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right)\right] \\ -\frac{r\cos\beta}{2} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial f_1}{\partial \beta} \\ \frac{\partial f_2}{\partial \beta} \\ \frac{\partial f_3}{\partial \beta} \end{bmatrix} = \begin{bmatrix} \Delta\varphi r\cos\beta\cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) - r\Delta\varphi\sin\beta\frac{r\Delta\varphi}{2L_2}\sin\beta\sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) \\ \Delta\varphi rcos\beta\sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) + r\Delta\varphi\sin\beta\frac{r\Delta\varphi}{2L_2}\sin\beta\cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) \\ \frac{r\Delta\varphi\sin\beta}{2} \end{bmatrix}$$

$$= \begin{bmatrix} r\Delta\varphi\left[\cos\beta\cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) - (\sin\beta)^2\frac{r\Delta\varphi}{2L_2}\sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right)\right] \\ r\Delta\varphi\left[\cos\beta\sin\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right) + (\sin\beta)^2\frac{r\Delta\varphi}{2L_2}\cos\left(\theta_k - \frac{r\Delta\varphi\cos\beta}{2L_2}\right)\right] \\ \frac{r\Delta\varphi\sin\beta}{2} \end{bmatrix}$$

4.3 误差传导模型仿真

实验中分别使用直线轨迹和圆弧轨迹在 Matlab 中进行仿真，验证误差传导模型（式4.8）。仿真设定的参数如下

- 移动轨迹：直线轨迹方程如下

$$\begin{cases} x = t \\ y = 0 \\ \theta = 0 \end{cases}$$

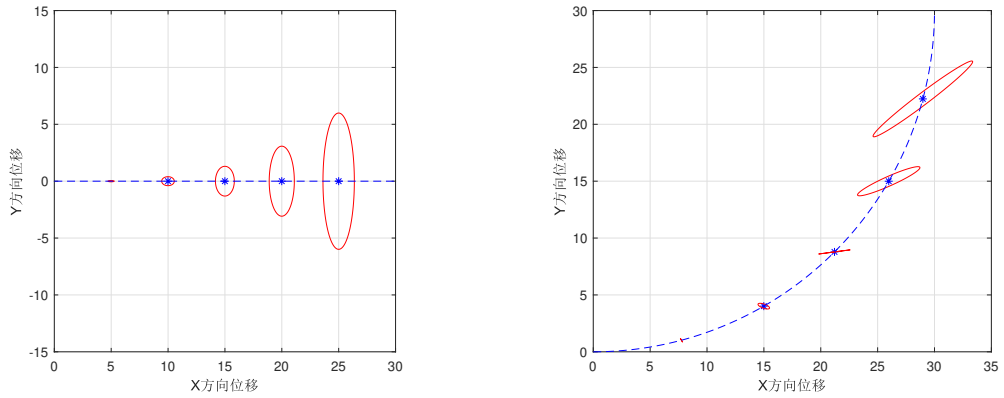
圆弧轨迹方程如下

$$\begin{cases} x = 30\cos\theta \\ y = 30\sin\theta \\ \theta = -\dfrac{\pi t}{60} \end{cases}$$

其中，$t \in [0, 30]$。

- 轮子转速 $\varphi = 0.11$，轮子半径 $r = 0.05$，驱动轮与从动轮的水平距离 $L_2 = 2$。

图4.8为仿真实验结果，分别绘制了直线轨迹和圆弧轨迹下的误差传导情况。其中蓝色虚线为移动轨迹，红色椭圆的长轴与短轴的长度分别对应式22特征值中 x 的部分与 y 的部分，方向对应特征向量中 x 的部分与 y 的部分。



(a) 直线轨迹 (b) 圆弧轨迹

图 4.8: 里程计的误差传导

图4.8(a) 说明在移动初期 x 方向的误差增长速度略快于 y 方向，在移动的中后期 y 方向的误差增长速度显著快于 x 方向，误差椭圆垂直于直线轨迹。图4.8(b) 说明曲线运动时，误差椭圆并不垂直于轨迹。出现这一现象的原因是，直线轨迹中 x 与 y 方向的位移是相互独立的，因此误差的增长也是相互独立的；而在圆弧轨迹中 x 与 y 方向的位移存在关联（均与 θ 相关），是非独立的，因此误差的增长也是非独立的，出现误差椭圆与轨迹不垂直的现象。

5 基于 SVD 的机器人定位算法

5.1 问题描述

已知两组点集，这两组点集是匹配的：在世界坐标系下的点集 $P \in R^{2\times n}$，在传感器坐标系下的点集 $Q \in R^{2\times n}$：

$$P = \{p_1, p_2, \cdots, p_n\}$$
$$Q = \{q_1, q_2, \cdots, q_n\}$$

要求计算出它们之间的刚体变换，即 R 和 t。由于传感器存在误差，要求所求的 R 和 t 满足

$$(R,t) = \arg\min \sum_{i=1}^{n} w_i \|(Rp_i + t) - q_i\|^2 \tag{25}$$

其中 w_i 为每个点对之间的权重。由于距离传感器的距离越远误差越大，所以越远的点应当赋予越小的权重。令 $w_i = 1/\sigma_i(\rho_i)$，即权重与距离成反比。

5.2 算法原理

首先式25对 t 求偏导得

$$\frac{\partial F}{\partial t} = 2t\left(\sum_{i=1}^{n} w_i\right) + 2R\left(\sum_{i=1}^{n} w_i p_i\right) - 2\sum_{i=1}^{n} w_i q_i = 0 \tag{26}$$

引入点集合 P 的加权中心点 $\hat{p}$ 和点集合 Q 的加权中心点 $\hat{q}$, 分别为:

$$\hat{p} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i} \quad \hat{q} = \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i} \tag{27}$$

式26两边同时除以 $\sum_{i=1}^{n} w_i$，得到

$$t = \widehat{q} - R\widehat{p} \tag{28}$$

将式28代入式25得

$$\begin{aligned} \sum_{i=1}^{n} w_i \|(Rp_i + t) - q_i\|^2 &= \sum_{i=1}^{n} w_i \|Rp_i + \hat{q} - R\hat{p} - q_i\|^2 \\ &= \sum_{i=1}^{n} w_i \|R(p_i - \hat{p}) - (q_i - \hat{q})\|^2 \end{aligned} \tag{29}$$

引入用集合 X 和集合 Y 表示 $p_i - \hat{p}$ 和 $q_i - \hat{q}$, 用 x_i 和 y_i 分别表示去中心化后的的点。

$$x_i = p_i - \widehat{p} \quad y_i = q_i - \widehat{q} \tag{30}$$

将式30代入式29，此时式25等价于

$$R = \arg\min \sum_{i=1}^{n} w_i \|Rx_i - y_i\|^2 \tag{31}$$

其中

$$\begin{aligned} \|Rx_i - y_i\|^2 &= (Rx_i - y_i)^T (Rx_i - y_i) \\ &= x_i^T R^T R x_i - y_i^T R x_i - x_i^T R^T y_i + y_i^T y_i \end{aligned} \tag{32}$$

由于旋转矩阵 R 为单位正交阵,所以 $R^T R = I$。由于 $x_i^T R^T y_i$ 为标量,所以 $x_i^T R^T y_i = (x_i^T R^T y_i)^T = y_i^T R x_i$。因此有

$$\|Rx_i - y_i\|^2 = x_i^T x_i - 2y_i^T R x_i + y_i^T y_i \tag{33}$$

将式33代入式31中，可得

$$\begin{aligned} \arg\min \sum_{i=1}^{n} w_i \|Rx_i - y_i\|^2 &= \arg\min \sum_{i=1}^{n} w_i \left(x_i^T x_i - 2y_i^T R x_i + y_i^T y_i\right) \\ &= \arg\min \left(\sum_{i=1}^{n} w_i x_i^T x_i - 2 \sum_{i=1}^{n} w_i y_i^T R x_i + \sum_{i=1}^{n} w_i y_i^T y_i \right) \\ &= \arg\min \left(-2 \sum_{i=1}^{n} w_i y_i^T R x_i \right) \\ &= \arg\max \left(\sum_{i=1}^{n} w_i y_i^T R x_i \right) \end{aligned} \tag{34}$$

对式34中 $\sum_{i=1}^{n} w_i y_i^T R x_i$ 进行变换

$$\begin{aligned}
\sum_{i=1}^{n} w_i y_i^T R x_i &= \mathrm{tr}\left(\begin{bmatrix} w_1 y_1^T R x_1 & & & \\ & w_2 y_2^T R x_2 & & \\ & & \ddots & \\ & & & w_n y_n^T R x_n \end{bmatrix}\right) \\
&= \mathrm{tr}\left(\begin{bmatrix} w_1 y_1^T \\ w_2 y_2^T \\ \vdots \\ w_n y_n^T \end{bmatrix} \begin{bmatrix} R x_1 & R x_2 & \cdots & R x_n \end{bmatrix}\right) \\
&= \mathrm{t}\left(\begin{bmatrix} w_1 & & & \\ & w_2 & & \\ & & \ddots & \\ & & & w_n \end{bmatrix} \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix} R \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}\right)
\end{aligned} \tag{35}$$

令 $W = \mathrm{diag}(w_1, \cdots, w_n), X = [x_1, x_2, \cdots, x_n], Y = [y_1, y_2, \cdots, y_n]$，优化目标可写为

$$\arg\max\left(\sum_{i=1}^{n} w_i y_i^T R x_i\right) = \arg\max\left(\mathrm{tr}\left(W Y^T R X\right)\right) \tag{36}$$

由于矩阵的迹满足 $\mathrm{tr}(AB) = \mathrm{tr}(BA)$，所以

$$\sum_{i=1}^{n} w_i y_i^T R x_i = \mathrm{tr}(W Y^T R X) = \mathrm{tr}((RX)(W Y^T)) = \mathrm{tr}(RX W Y^T) \tag{37}$$

令 $S = X W Y^T$, 由 SVD 原理，可知：

$$S = U \Sigma V^T \tag{38}$$

其中，U, V 为单位正交阵，Σ 为对角阵 $\mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_n)$。

将式38带入式37可得：

$$\sum_{i=1}^{n} w_i y_i^T R x_i = \mathrm{tr}(RXW Y^T) = \mathrm{tr}(RU\Sigma V^T) \overset{\mathrm{tr}(AB)=\mathrm{tr}(BA)}{=} \mathrm{tr}(\Sigma V^T R U) \tag{39}$$

令 $M = V^T R U$, 由于 U, V, R 均为单位正交阵，则 M 同样也是正交矩阵。这也意味在矩阵 M 中，每

一列的向量 $m_j, m_j^T m_j = 1$，式39可转换为

$$\begin{aligned}\sum_{i=1}^{n} w_i y_i^T R x_i &= \mathrm{tr}(\Sigma V^T R U) = \mathrm{tr}(\Sigma M) \\ &= \mathrm{tr}\left(\begin{bmatrix}\sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n\end{bmatrix}\begin{bmatrix}m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn}\end{bmatrix}\right) \\ &= \sum_{i=1}^{n} \sigma_i m_{ii} \le \sum_{i=1}^{n} \sigma_i\end{aligned} \tag{40}$$

当且仅当 $M = I$ 时，等号成立，式40取最大值，则

$$V^T R U = I \Rightarrow R = V U^T \tag{41}$$

将式41代入式28即可求得 t。最小匹配误差为

$$\arg\min \sum_{i=1}^{n} w_i \|(R p_i + t) - q_i\|^2 = \sum_{i=1}^{n} w_i x_i^T x_i + \sum_{i=1}^{n} w_i y_i^T y_i - 2\sum_{i=1}^{n} \sigma_i$$

5.3 算法仿真

实验中在 Python 下对 SVD 定位算法进行仿真。首先设定传感器坐标系相对于世界坐标系的旋转矩阵与平移向量。在世界坐标系下，随机选取 n 个点（$n \le 3$），并利用旋转矩阵与平移向量求得这些点在传感器坐标系下的对应坐标。由于传感器具有一定的误差，所以实验中对传感器坐标下的点坐标增加了服从正态分布（均值为 0，标准差为 0.01）的噪声。利用在传感器坐标系下的点坐标反推旋转矩阵和平移向量。图5.9展示了在世界坐标系下随机选取的点集（红色）与传感器坐标系（蓝色）。

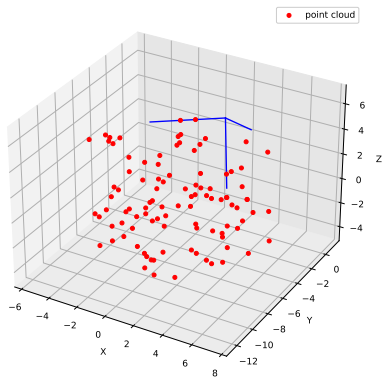


图 5.9: SVD 定位仿真

实验中选取 $n = 100$，利用矩阵的 Frobenius 范数衡量所求结果与设定值的误差，仿真结果如图5.10所示。可以发现，添加噪声后求得到旋转向量和平移矩阵，与设定值的误差较小，证明了 SVD 定位算法的有效性。实验中还发现随着 n 的增长，算法的误差越小。

```
设定旋转矩阵
 [[-0.69492056 -0.71352099  0.08929286]
 [ 0.19200697 -0.30378504 -0.93319235]
 [ 0.69297817 -0.6313497   0.34810748]]

设定平移向量
 [0.5 1.  3. ]
```

(a) 设定值

```
求得旋转矩阵
 [[-0.69463487 -0.71383938  0.08897039]
 [ 0.19230118 -0.30344374 -0.93324282]
 [ 0.69318298 -0.63115389  0.34805476]]

求得平移向量
 [0.49680081 0.99788593 3.00272421]
```

(b) SVD 定位算法的结果

```
旋转矩阵误差
 0.0007587077272521313

平移向量误差
 0.004703768586500226
```

(c) SVD 定位算法的误差

图 5.10: 设定的机器人位姿与 SVD 定位算法所求位姿

6 基于 ICP 的点云匹配算法

6.1 问题描述

ICP（Iterative Closest Point，迭代最近点）是一种点云配准算法，用于将两个或多个点云之间的位置和姿态进行匹配，使它们在空间中尽可能地重合。在机器人定位中，ICP 算法可以用于将传感器信息与已知的地图信息进行点云匹配，进而利用 SVD 算法求得机器人的位姿。

6.2 算法原理

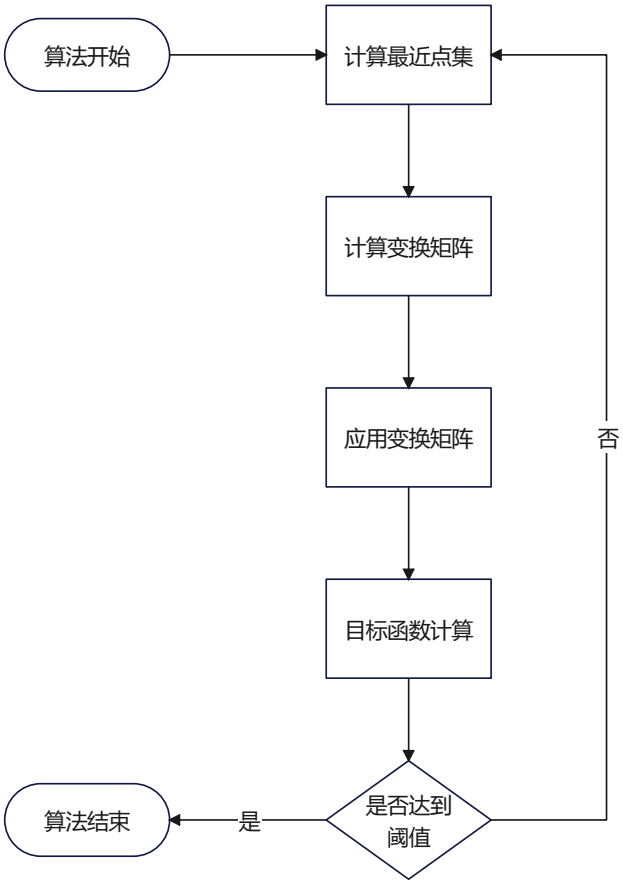


图 6.11: ICP 算法流程

ICP 算法的基本流程如图6.11所示，主要包括：

- Step 1：计算最近点集。

  点集 P(传感器检测数据点集) 中有 N_p 个数据点

$$P = \{\vec{p}_i, i = 1, \cdots, N_p\}$$

  点集 $X_{\text{(目标点集) 中有}N_x}$ 个数据点

$$X = \{\vec{x}_i, i = 1, \cdots, N_x\}$$

  d 为某个数据点 $\vec{p}_i \in P$ 和点集 X 的几何距离

$$d(\vec{p}_i, X) = \min_{\vec{x} \in X} \|\vec{x} - \vec{p}_i\|$$

  令 $\vec{y}_i \in X$ 为最近点，满足

$$d(\vec{p}_i, \vec{y}_i) = d(\vec{p}_i, X)$$

  求取最近点集 Y

$$V = C(P, X) = \{(\vec{p}_i, \vec{y}_i) | d(\vec{p}_i, \vec{y}_i) = d(\vec{p}_i, X), \vec{p}_i \in P, \vec{y}_i \in X, i = 1, \cdots, N_p\}$$

- Step 2：计算变换矩阵：利用上一章所述的 SVD 算法计算最近点集的旋转矩阵 R 和平移向量 t。
- Step 3：应用变换矩阵：对最近点集应用求得的 R 和 t，更新点集的坐标值。
- Step 4：计算目标函数并判断是否小于阈值，若小于阈值则停止迭代；若大于阈值，则继续迭代。

6.3 算法仿真

实验中使用了斯坦福大学开源的 3D 测试模型 Stanford Bunny 作为点云数据，如图6.12所示。由于原模型 bun000.ply 的点云数据过多，仿真中只选取了原模型四分之一的点云信息（10065 个点）。

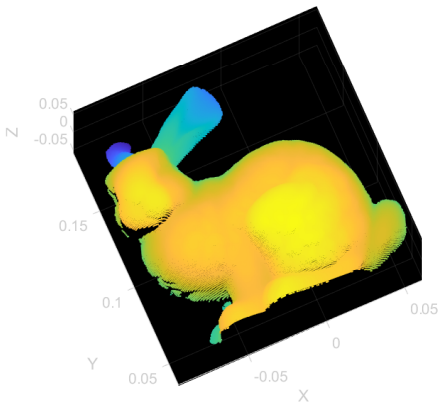


图 6.12: Stanford Bunny 模型

实验中设定阈值为 10^{-6}。仿真中首先设定旋转矩阵和平移向量，利用原始数据得到传感器下的坐标值。利用 ICP 算法进行迭代，得到误差随迭代次数的变换曲线，如图6.13所示。可以发现误差随迭代次数的增加而减小，迭代至 30 轮时误差小于设定阈值。图6.14展示了迭代过程中点云匹配的过程。

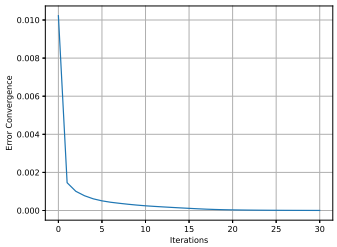


图 6.13: 误差随迭代次数的曲线图

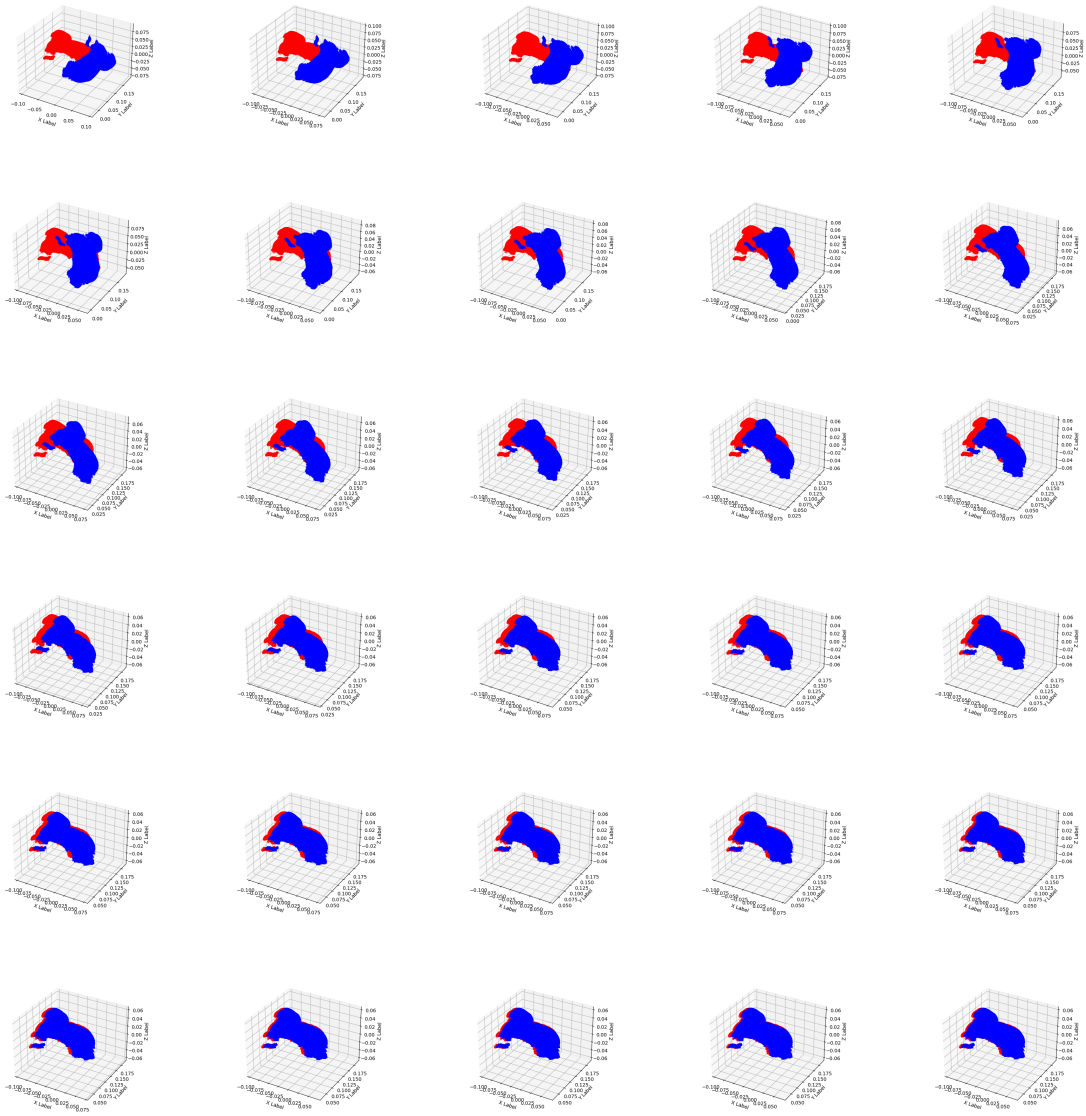


图 6.14: 点云匹配过程

7 基于 Hough 变换的直线提取算法

7.1 问题描述

由于原始点云信息数据量庞大，利用原始点云进行机器人定位的过程中，计算速度较慢。另外因为原始点云特征存在误差，在匹配过程中会存在匹配失败的问题。因此，在原始点云信息的基础上提取特征信息，利用特征信息进行匹配，将提高计算速度和匹配精度。

在传感器获得的信息中，直线是一种常见且容易提取的信息特征。提取直线需要解决三个问题:

- 图像中有多少直线?
- 哪些点属于哪条直线?
- 如何对直线进行拟合?

常用的直线提取算法包括 Split-and-merge、Linear regression、RANSAC、Hough 变换。本报告中使用 Hough 变换进行直线提取。

7.2 算法原理

Hough 变换是一种在图像处理和计算机视觉领域中广泛应用的技术，主要用于检测图像中的几何形状，特别是直线和圆。在 Hough 变换中，图像被转换到参数空间，从而使得原始图像中的几何形状（如直线、圆）在参数空间中表现为单个点。这种变换使得在原始图像中检测几何形状变得更加容易。

对于检测直线，Hough 变换的工作原理如下。由于常用的表示直线的斜截式方程 $y = ax + b$ 无法表示垂直于 X 轴的直线，所以使用极坐标表示的直线方程

$$x\cos\theta + y\sin\theta = \rho \tag{42}$$

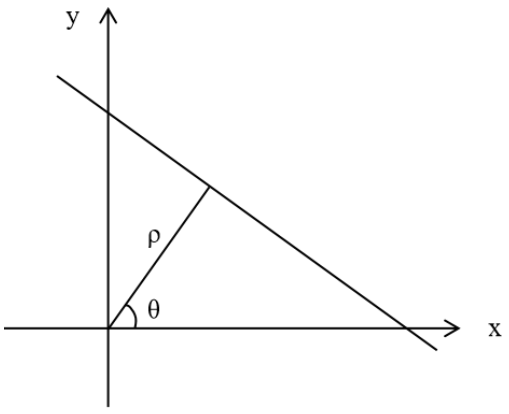


图 7.15: 直线的极坐标表示

如图7.16所示，对于 $x-y$ 空间中的一个点 (x_0, y_0)，转换到 $\rho-\theta$ 空间中对应的为一条曲线 $\rho = x_0\cos\theta + y_0\sin\theta$；对于 $x-y$ 空间中的一条直线 $x\cos\theta + y\sin\theta = \rho$，对应于 $\rho-\theta$ 空间中的一个点 (ρ_0, θ_0)。因此，可以设置一个 Hough 参数空间累加矩阵，对于 $x-y$ 空间中的每个点，遍历 $\theta \in [0, 180]$，在累加矩阵对应的 (ρ, θ) 中加 1。遍历所有点后，累加矩阵中最大值对应的 (ρ, θ) 所产生的直线 $x\cos\theta + y\sin\theta = \rho$，则是图像中最有可能的直线。另外，若设定一个阈值，累加矩阵中大于该阈值对应的 (ρ, θ) 均认为为直线，就可以得到图像中的多条可能直线。

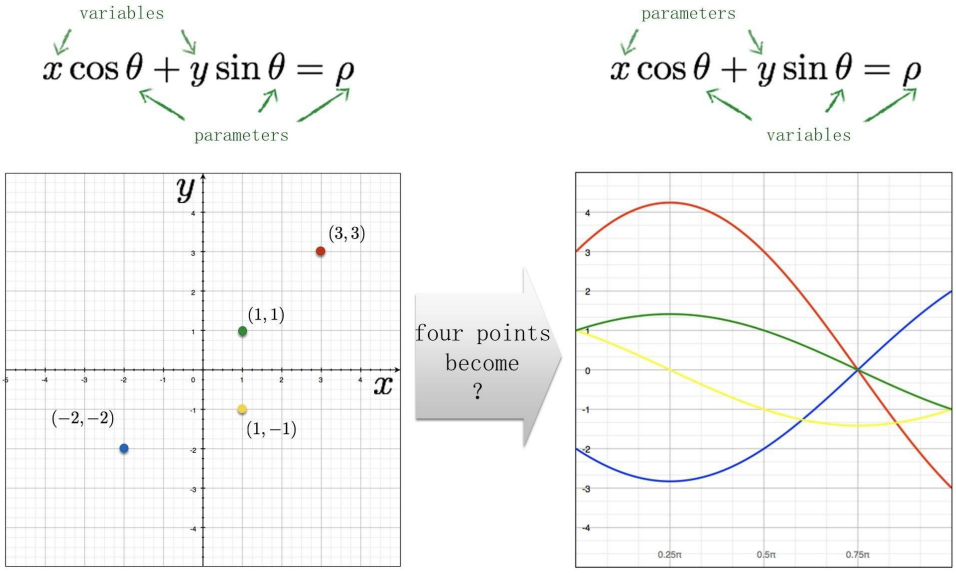
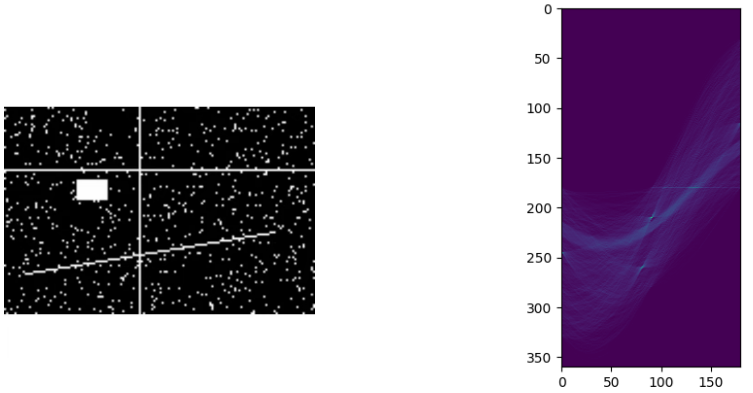


图 7.16: 空间变换

7.3 算法仿真

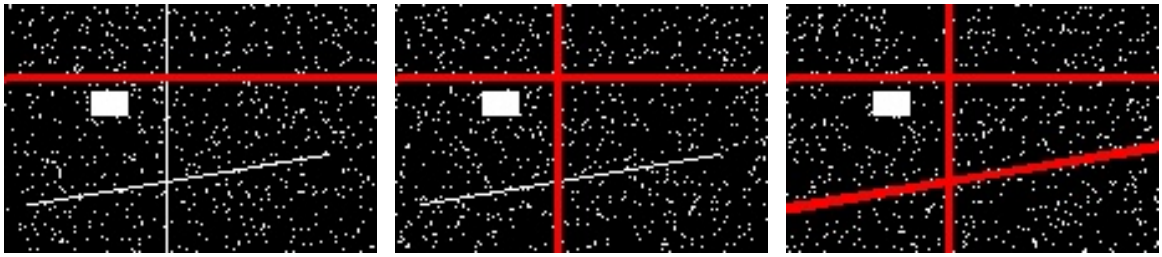
实验中利用 Python 进行仿真实验。首先生成一张 100*150 的图片，在图片中生成三条直线，并在图像中随机生成一些点，如图7.17(a) 所示。对该图像进行 Hough 变换，将 $x-y$ 空间转换为 $\rho-\theta$ 空间，得到的 Hough 参数空间累加矩阵，如图7.17(b) 所示。可以发现，在累加矩阵中有若干较亮的点，说明该点可能对应一条 $x-y$ 空间的直线。



(a) 原图 (b) Hough 空间累加矩阵

图 7.17: 测试图像与经 Hough 变换的结果

由于实验中设置的图像中含有多条直线，所以不能单纯选取累加矩阵的最大值，而是需要设定一定阈值。实验中分别设置了 0.9、0.6 倍和 0.5 倍的累加矩阵中的最大值，认为累加矩阵中大于该阈值的 ρ,θ) 表示一条直线，结果如图7.18所示。可以发现，随着阈值降低图像中被识别为直线的数量增加。因此在使用 Hough 变换算法提取直线时，需要设定一个合适的阈值，才能提取到图像理想的直线。



(a) $threshold = 0.9 \times max$ (b) $threshold = 0.6 \times max$ (c) $threshold = 0.5 \times max$

图 7.18: 不同阈值下的直线提取结果

8 基于卡尔曼滤波的位置跟踪算法

8.1 问题描述

一阶线性离散系统可用以下方程进行描述

$$x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1} \tag{43}$$

$$z_k = Hx_k + v_k \tag{44}$$

其中 $x_k \in R^n$ 为系统状态，$z_k \in R^m$ 为测量输出，$u_k \in R^l$ 为系统输入。$w_k \in R^n$ 为过程噪声 (Process Noise), $v_k \in R^m$ 为测量噪声 (Measurement Noise) 为白噪音

$$p(w) = N(0, Q) \quad p(v) = N(0, R)$$

目标： 求取通过系统状态转移方程预测的系统状态 x_k 和通过测量方程观测到的系统状态 z_k 下最优的状态估计。

8.2 算法原理

首先定义先验估计、先验估计误差和先验估计方差，以及后验估计、后验估计误差和后验估计方差。

- 先验估计：定义 $\hat{x}_k^- \in R^n$ 为状态 k 处的先验状态估计，由前一状态 x_{k-1} 和式43得到。
- 先验估计误差：定义 $e_k^- \equiv x_k - \hat{x}_k^-$ 为状态 k 处的先验估计误差。
- 先验估计方差：定义 $P_k^- = E[e_k^- e_k^{-T}]$ 为状态 k 处的先验估计方差。
- 后验估计：定义 $\hat{x}_k \in R^n$ 为状态 k 处的后验状态估计，由式44的测量结果和先验误差 $\hat{x}_k^-$ 得到。
- 后验估计误差：定义 $e_k \equiv x_k - \hat{x}_k$ 为状态 k 处的后验估计误差。
- 后验估计方差：定义 $P_k = E[e_k e_k^T]$ 为状态 k 处的后验估计方差。

在卡尔曼滤波中，利用先验估计 $\hat{x}_k^- \in R^n$ 和实际测量值 z_k 来对当前状态进行估计，如下式所示

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-) \tag{45}$$

其中，$K \in R^{n\times m}$ 为卡尔曼增益。具体而言，式45表示将先验估计 $\hat{x}_k^- \in R^n$ 和实际测量值与预测测量值的差 $z_k - H\hat{x}_k^-$ 进行加权。当 $K=0$ 时，表示只利用先验估计进行状态更新；当 $K=H^{-1}$ 时表示只利用实际观测值进行状态更新。在实际使用中，可以根据先验估计与实际测量的准确程度，调节 K 值，从而更加准确地更新状态值。

卡尔曼滤波算法的基本思想是通过融合先验估计和实际测量值，得到更加准确的状态估计值。从优化策略角度出发，卡尔曼滤波算法的目标是找到后验估计方差最小的融合策略，即找到使得下式最小的 K 值

$$\arg\min_K P_k \tag{46}$$

对 P_K 进行展开

$$\begin{aligned}
P_k &= E[e_k e_k^T] = E[(x_k-\hat{x}_k)(x_k-\hat{x}_k)^T] \\
&= E[\{x_k - [\hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right)]\}\left\{x_k - [\hat{x}_k^- + K\left(z_k - H\hat{x}_k^-\right)]\right\}^T] \\
&= E[((x_k-\hat{x}_k^-) - K(H(x_k-\hat{x}_k^-)+v_k))\left((x_k-\hat{x}_k^-\right) \\
&= E[(e_k - K(He_k^- + v_k))(e_k^- - K(He_k^- + v_k))^T] \\
&= E[((I-KH)e_k^- - Kv_k)((I-KH)e_k^- - Kv_k)^T] \\
&- (I-KH)e_k^- v_k^T K^T] \\
&= (I-KH)\underbrace{E[e_k^- e_k^{-T}]}_{P_k^-}(I-KH)^T + K\underbrace{E[v_k v_k^T]}_{R}K^T - K\underbrace{E[v_k e_k^{-T}]}_{0}(I-KH)^T \\
&- (I-KH)\underbrace{E[e_k^- v_k^T]}_{0}K^T \\
&= (I-KH)P_k^-(I-KH)^T + KRK^T
\end{aligned} \tag{47}$$

需要求式47的最小值。对 K 求偏导，并使其为 0

$$\frac{\partial P_k}{\partial K} = -2P_k^- H^T + 2KHP_k^- H^T + 2KR = 0$$

可以得到

$$K = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{48}$$

根据式48可以发现，当 $R\to 0$，即测量噪声完全准确时，有

$$\lim_{R\to 0} K = H^{-1}$$

当 $R_k^- \to 0$，即先验估计完全准确时，有

$$\lim_{P_k^-\to 0} K = 0$$

这样的结果符合上文所述的：“当 $K=0$ 时，表示只利用先验估计进行状态更新；当 $K=H^{-1}$ 时表示只利用实际观测值进行状态更新”。

卡尔曼滤波的算法流程如图8.19所示，包括预测和修正两个部分，具体步骤如下：

1. 初始化：设定初始状态和初始后验估计方差。

2. 预测：

(a) 根据式43得到先验的状态估计 $\hat{x}_k^-$

(b) 根据式49得到先验估计协方差 $\hat{P}_k^-$

$$\begin{aligned} e_k^- &= x_k - \widehat{x}_k^- \\ &= (Ax_{k-1} + Bu_{k-1} + w_{k-1}) - (Ax_{k-1}^- + Bu_{k-1}) \\ &= A(x_{k-1} - \hat{x}_{k-1}^-) + w_{k-1} \\ &= Ae_{k-1} + w_{k-1} \end{aligned}$$

$$\begin{aligned} \hat{P}_k^- &= E[e_{k-1}^- {e_{k-1}^-}^T] \\ &= E[(Ae_{k-1} + w_{k-1})(Ae_{k-1} + w_{k-1})^T] \\ &= E[(Ae_{k-1})(Ae_{k-1})^T] + E[w_{k-1}w_{k-1}^T] \\ &= AP_{k-1}A^T + Q \end{aligned} \tag{49}$$

3. 修正：

(a) 根据式48计算最优卡尔曼增益 K

(b) 根据式45计算卡尔曼滤波算法得到的当前状态估计 $\hat{x}_k$

(c) 根据式50计算后验估计的协方差矩阵 P_k

$$\begin{aligned} P_k &= P_k^- - KHP_k^- - P_k^- H^T K^T + K(HP_k^- H^T + R)K^T \\ &= P_k^- - KHP_k^- - P_k^- H^T K^T + P_k H^T (HP_k^- H^T + R)^{-1}(HP_k^- H^T + R)K^T \\ &= P_k^- - KHP_k^- - P_k^- H^T k^T + P_k H^T K^T \\ &= (I - KH)P_k^- \end{aligned} \tag{50}$$

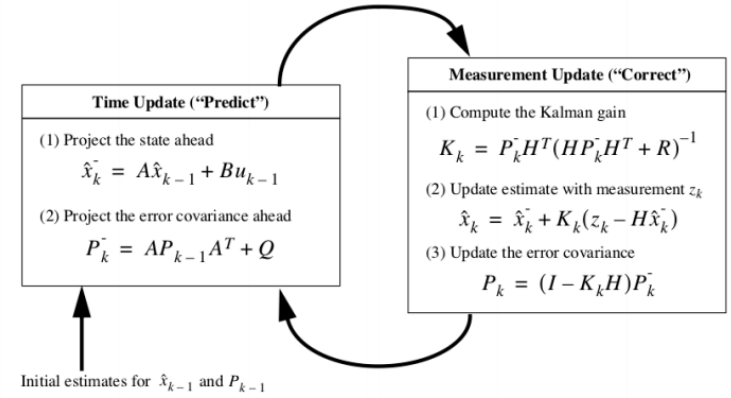


图 8.19: 卡尔曼滤波的算法流程

8.3 算法仿真

实验中利用 Matlab 进行仿真实验。首先设定一阶线性离散系统方程：

$$\begin{cases} \begin{bmatrix} x_k \\ y_k \end{bmatrix} = A \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} + Bu + w_{k-1} \\ z_k = H \begin{bmatrix} x_k \\ y_k \end{bmatrix} + v_k \end{cases} \tag{51}$$

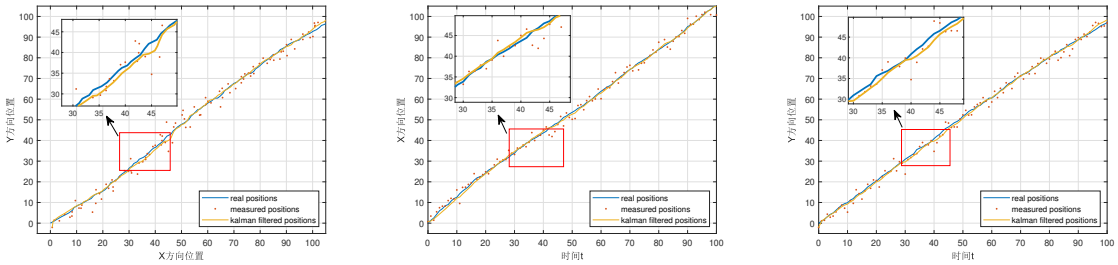
其中，

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad u = 1$$

仿真中认为系统方程的较准确，方差较小；测量方程的较不准确，方差较大。过程噪声的协方差矩阵 Q 和测量噪声的协方差矩阵 R 分别为：

$$Q = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \quad R = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} \tag{52}$$

仿真中设定系统的初始状态 $\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$，初始后验估计协方差矩阵 $P_0 = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}$。仿真结果如图8.20所示，分别展示了 X-Y 上的位置跟踪结果和对 X、Y 的位置跟踪结果。可以看出，系统方程的轨迹为一条直线。在测量方差较大的情况下，卡尔曼滤波也可以对真实位置进行较好的跟踪。



(a) X-Y 上的位置跟踪 (b) 对 X 的位置跟踪 (c) 对 Y 的位置跟踪

图 8.20: 卡尔曼滤波仿真结果