

KSim:KheperaIII完全Matlab模拟

张霖

指导教师

俞立 教授

浙江工业大学信息工程学院

最后修改日期

2014 年 10 月 12 日

Typeset by L^AT_EX 2_• at October 12, 2014
With package **CASthesis** v0.2 of CT_EX.ORG

目 录

目录	i
第一章 引言	1
1.1 系统要求	2
1.2 软件安装	2
1.3 系统组成	2
1.4 问题反馈	3
第二章 基础知识	5
2.1 KheperaIII机器人	7
2.2 面向对象的Matlab编程	8
2.3 S文件与Matlab类联合使用	9
2.4 Matlab的Embedded Coder工具箱	9
2.5 定制模拟环境	10
2.5.1 坐标系	10
2.5.2 定制障碍物	11
2.5.3 放置机器人	12
第三章 模拟研究	15
3.1 运动和编码器	15
3.1.1 轮子转速接口	16
3.1.2 轮子编码器接口	17
3.1.3 实例：多智能体编队	17
3.1.4 模拟与实际的区别	19
3.2 红外线传感器	19
3.2.1 红外线传感器接口	20

KSim:KheperaIII完全Matlab模拟

3.2.2 模拟与实际的区别	21
3.3 超声波传感器	21
3.3.1 超声波传感器接口	22
3.3.2 传感器噪声特性的获取	23
3.3.3 实例：利用EKF自定位	24
3.3.4 模拟与实际的区别	25
第四章 实验研究	27
4.1 遥控机器人方案	27
4.2 准备C语言库文件	27
4.3 将算法运行在机器人内	28
4.3.1 实例：Goto-Goal	30
参考文献	33
致谢	37

第一章 引言

KSim是一套旨在为移动机器人教学和科研提供便利的软件集合。

在Matlab环境下，KSim可以模拟KheperaIII^[5]机器人在平面运动的情况，包括轮子的驱动器，编码器，9个红外线接近传感器以及5个超声波传感器的工作（如图1.1）。向虚拟环境添加多个KheperaIII机器人以后，KSim可以模拟这些机器人之间的交互，包括物理碰撞和红外线传感器的测量。同时KSim还提供了将算法自动编译为能够在真实机器人上运行程序所必备的工具和方法。

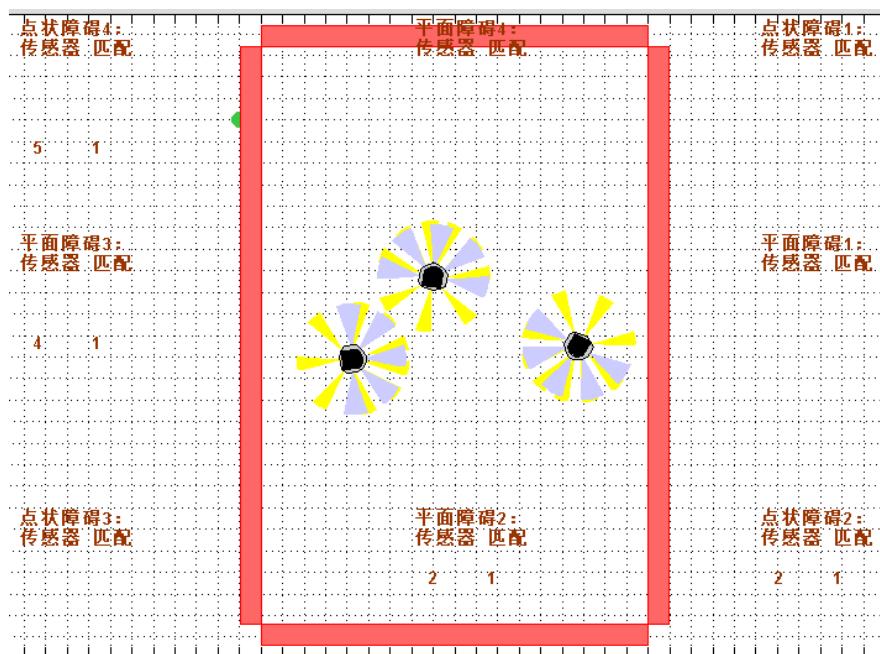


图 1.1: KSim的模拟环境

KSim再现了KheperaIII机器人超声波传感器的镜面反射特性和噪声特性，特别是其噪声特性参数由实验获得。利用显示功能组件，可以动态显示传感器对障碍物的侦测，以及任何内部参数变化情况。专门设计的代码区，不仅可以访问器人所有资源，而且能够在最少理KSim工作原理的情况下快速开展工作。在模拟环境当中完成算法测试以后，利用Embedded Coder工具箱以及“RealKteam”库文件，将算法直接编译为KheperaIII机器人嵌入式系统程序。

2 KSim:KheperaIII完全Matlab模拟

KSim开放所有代码，并与国外重点实验室软件包相互兼容。用户可以定值自己的KSim，包括模拟环境大小、障碍物的形状和位置，甚至添加其他型号的机器人。

衷心希望KSim可以为您的教学和科研工作带来便利。

1.1 系统要求

使用KSim的移动机器人模拟功能，需要2009a或更新版本的Matlab及Simulink。

使用“RealKteam”代码生成功能，则需要Matlab的Embedded Coder工具箱及KheperaIII机器人程序开发所需要的Linux环境。该环境的配置请参见[\[7\]](#)。

1.2 软件安装

类似Matlab工具箱的安装。将KSim软件包解压至安装路径，并将Matlab的工作目录导引至软件包内“simulink”文件夹下，执行该文件夹内的“SetPath.m”文件。

KSim在设置完成后，即启动示例程序，运行该程序以测试KSim是否正常工作。

1.3 系统组成

KSim由模拟器和“RealKteam”C语言运行库两部分组成。

可以将模拟器看作是一套能够调用Matlab类的Simulink程序集。用户在“simulink”文件夹下，利用模型库“KSim.mdl”内的模块创建自己的Simulink模型。模拟开始以后，模块内的S文件实例化“+simiam”文件夹内的Matlab类以构成模拟环境。该模拟环境与Simulink同步运行并与之交互信息，直到用户选择停止或者出现碰撞。通过对相应方法的调用，Matlab计算可能发生的事件（比如是否有机器人碰触到墙壁，超声波传感器探测到了哪些物体等），并将这些计算的结果转化为图像元素。通过不断重绘同一个Figure，用户看到了动态的机器人运动过程，即图[1.1](#)的用户界面。

“RealKteam”是一套专为KheperaIII设计的C语言运行库，用于在Linux环境下生成能够在真实机器人上执行的应用程序。

Matlab的Embedded Coder工具箱将两者联系在一起。利用KSim的模拟环境规范程序使用的函数及接口结构，使得在KSim设计的算法经过Embedded Coder编译为C语言以后，可以与“RealKteam”兼容。

1.4 问题反馈

在使用中遇到问题或者有任何建议，都可以和作者联系：

张霖 bluebird.house@163.com

欢迎大家反馈自己的使用情况，您的建议是我的进步的阶梯。

第二章 基础知识

如果使用KSim系统，就意味着理解其详细的工作原理，以及所有背景知识，那么这套软件的易用性就是存在质疑的。这一章首先给出KSim的研发历史，然后尽可能简略的介绍背景知识，并给出哪里能够得到有关这些知识的详细信息。一旦有需要可以查阅对应的参考文献。

KSim基于两个软件集合开发而成。其模拟器的前端显示代码，来自美国乔治理工GRITS(Georgia Robotics and Intelligent Systems Laboratory)实验室的教学项目Sim.I.am^[8]，该项目旨在拉近控制理论教学与实践之间的距离。作为一款BSD^[13]授权的开源软件，Sim.I.am可以模拟两台KheperaIII机器人在位姿精确已知的情况下，于二维平面的运动状况。

KSim研制初期目的是配合浙江工业大学俞立教授关于滤波估计技术的教学需求，因而在Sim.I.am基础上的再开发过程中，增加了对KheperaIII机器人超声波传感器模拟支持，并可以支持多个机器人。受到《浙江省大学生科技创新活动计划（新苗人才计划）》资助以后，发展为功能完善的软件包。



图 2.1: KSim虚拟的实验平台实物

软件模拟环境真实再现了浙江工业大学信息工程学院嵌入式系统实验室的移动机器人实验平台（如图2.1）。KSim的模拟环境（图1.1）是该实验平台的比例缩小，

6 KSim:KheperaIII完全Matlab模拟

实验平台长: 2.680m, 宽: 1.791m。

每个坐标格代表实际距离10cm。

由于超声波传感器的镜面反射特性, 软件模拟环境中周围障碍物的超声波模型, 严格参考经典书籍[10]。KheperaIII机器人超声波传感器噪声特性在平台2.1上经实验获得。考虑到Sim.I.am本身也在不断完善中, KSim尽可能的保留了对Sim.I.am的最大支持。可以首先下载安装Sim.I.am, 然后将KSim覆盖在文件夹中, 以便获得到更多功能。

在教学实践中, 发现学生在Simulink环境下使用模块拼接控制算法的过程远快于学习面向对象的Matlab编程。KSim完全重写了Sim.I.am所带有的Simulink模块, 使其能够从S函数中访问机器人的所有属性和方法以方便学生搭建控制算法。但是对于高级研究者, 特别是需要资源较多的滤波估计算法研究, 还是推荐使用面向对象的Matlab语言。

在理论联系实际方面, Sim.I.am采用的是“遥控机器人方案”。原理是利用无线局域网络, 构建Sim.I.am与真实KheperaIII机器人之间的数据通路。所有的算法均在上位机中运算完成, 仅将运动指令发送至KheperaIII。在KSim中, 仍然可以使用这一功能。这一设计思路虽然被众多实验室采用, 但是由于不再考虑移动机器人本身的运算能力, 会鼓励研究者使用更复杂的计算方法。

KSim提供了将算法完全运行在真实的KheperaIII移动机器人系统内的所有工具和方法。其解决方案是综合利用Matlab的Embedded Coder工具箱以及“RealKteam”库文件, 这些库文件位于KSim安装目录的“RealKteam”文件夹中。首先利用Embedded Coder工具箱将相应的算法编译为嵌入式C代码, 然后与KheperaIII库文件联合编译。库文件的组织方式参考瑞士联邦理工大学Distributed Intelligent Systems and Algorithms Laboratory (DISAL) 实验室的Khepera III Toolbox^[14]软件包。该软件包为早期的Khepera III机器人设计, 由于生产厂商的技术升级已经无法继续使用, 且原有软件包的用户协议不允许在其基础上做修改。基于这种情况, 参考原有软件包的数据结构和工作原理, 利用Khepera III机器人生产厂商自带的KoreBot Library^[6]进行了重新编写。

“RealKteam”的特殊设计, 使得重新编写的库文件与Khepera III Toolbox的接口结构相互兼容, Khepera III Toolbox提供的工具软件^[17]可以运行在任何一个版本的Khepera III机器人上。如果需要使用这些工具软件, 请至参考文献指定的网址下载并利用“RealKteam”对其进行重新编译。

虽然整个KSim的工作同时利用了Matlab和Simulink两大平台，但对于研究控制算法的用户，仅需要关注自己创建的Simulink文件即可。推荐用户将控制算法写入“MATLAB Function”中，该模块可以对算法用到的函数加以筛选，从而在算法开发初期就避免使用那些不能被转换为C代码的函数。

滤波算法通常需要更多的资源，而这类实验通常预先设计机器人的运动路径以比较滤波算法的准确性。此时可以在Simulink的环境中规划机器人的运动，推荐使用KSim给出的示例程序，而将滤波算法加入：

“+simiam \ +robot \ Khepera3.m”

该文件描述KheperaIII的所有特征因而可以最大限度的为这些算法提供资源。在使用KSim的过程中，基本上只需要关注“Khepera3.m”一个文件即可。

KSim已经应用于浙江工业大学信息工程学院欧林林、张文安两位教授的研究生及本科生培养工作中。希望它也可以为您的教学和科研工作带来便利。

2.1 KheperaIII机器人



图 2.2: KheperaIII机器人

图2.2即为瑞士K-Team公司生产的移动机器人KheperaIII^[5]。该机器人高70mm，形状类似直径130mm的圆形，重690g，是典型的两差动轮机器人。安装有一块Intel XSCALE PXA-270@600MHz嵌入式系统板^[5]（称作Korebot 2）承载用户程序。环绕机身安装有9个红外线接近传感器，5个超声波距离传感器。所有传感器和电动机受内部DsPIC单片机控制，通过I²C总线与嵌入式系统板相连间接以便与用户程序通讯。

在KSim中，kheperaIII机器人绘制如图2.3的样子，机器人周围黄色三角形表示红外线传感器，淡蓝色三角形表示超声波传感器。

8 KSim:KheperaIII完全Matlab模拟

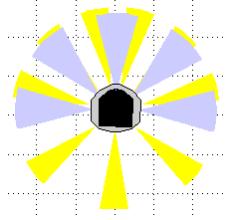


图 2.3: 模拟器中的KheperaIII机器人

2.2 面向对象的Matlab编程

Matlab自2004版以后具备了完善的面向对象编程功能，从而将数据与方法进行捆绑。配合Matlab强大的矩阵处理能力和丰富的工具箱，使得这一数学语言工具成为KSim开发的不二选择。如果您有VC++、VC #等任何一种面向对象语言编程经历，那么您不需要学习面向对象的Matlab编程即可平滑过渡。比如“Khepera3.m”文件中，Khepera3类是由simiam.robot.Robot类派生的，则写为：

```
classdef Khepera3 < simiam.robot.Robot
```

所以Khepera3的构成函数在初始化何类函数和方法以前，首先就要调用其超类的构成函数：

```
function obj = Khepera3(parent, pose)
obj = obj@simiam.robot.Robot(parent, pose);
```

其中的obj关键字表示这个类本身。

如果一个类是由handle类派生的，那么就是告诉Matlab这个类需要与核心进程同时工作以便及时处理事件。

类似的概念和规则出现在各种面向对象语言当中，可见Matlab面向对象编程支持是全面的和严格的。使用KSim并不一定需要知晓Matlab的方方面面，但是了解一二有助于更好的利用其功能。如果需要全面的Matlab面向对象编程的知识，可以参考文献[18]或者Matlab自身的帮助文件。在3.3.3节中，借助于面向对象语言的数据和方法捆绑特性，将扩展卡尔曼滤波器的繁杂过程精炼为几个方法，充分体现了这种技术的优势。

2.3 S文件与Matlab类联合使用

S文件可以通过间接的方式访问KSim当中的类实例，比如文件

```
"simulink \ msfun_khepera3_WithPosition.m"
```

内的“robot”就是当前模块所模拟的KheperaIII机器人，及“Khepera3.m”的一个实例。需要注意的是，Matlab在运行以前，首先将所有类的方法和属性记录下来，然后在需要的时候执行其中对应的代码。如果修改类的方法或属性，则需要将这些信息清除掉。要做到这一点，可以使用指令：

```
clear classes
```

或者

```
clear java
```

如果这个属性或方法是静态的，那么需要关闭对应的Simulink窗口才能将其清除。

仅修改方法的代码就不需要清除任何信息。Matlab面向对象编程最好首先设计好属性、方法等数据结构，然后再向其中添加实现的代码。

2.4 Matlab的Embedded Coder工具箱

Embedded Coder工具箱可以将部分Matlab代码转换为用于嵌入式系统的C代码，这为算法的产品化提供了极大方便。美中不足的是它需要相对繁琐的配置才能够正常工作。

KSim从两方面解除用户的困扰。首先，KSim的模拟器鼓励用户将控制算法写入“MATLAB Function”中，该模块可以对算法用到的函数加以筛选，从而在算法开发初期就避免使用那些不能被转换为C代码的函数。模拟器和Simulink环境下的例程，共同起到了规范用户代码接口结构的作用，在应用Embedded Coder工具箱时得以简化配置操作。另一方面，KSim参考Khepera III Toolbox设计了自己的C语言编程库，这些库文件位于：

“RealKteam”

文件夹中，这使得控制算法不需要去考虑实际的硬件操作过程，Embedded Coder工具箱转换后的C代码可以直接被复制到KheperaIII的Linux开发环境当中加以利用。

¹⁰ KSim:KheperaIII完全Matlab模拟

2.5 定制模拟环境

与大多数Matlab工具箱一样，用户可以查看和修改KSim的所有代码，并且可以方便的定制模拟环境。这不但包括模拟环境大小、障碍物的形状和位置，甚至可以将其他型号的机器人添加其中。首先介绍KSim模拟环境的坐标系定义，然后给出定制模拟环境的方法。

2.5.1 坐标系

KSim的坐标系定义遵循一般移动机器人研究规范。运动平面上建立世界坐标系 $\{O\}$ ，位于整个模拟环境的中心。机器人上附着坐标系 $\{\text{Robot}\}$ ，其原点在两轴中心，x轴正方向指向两轴距中心和前部支撑点的连线方向，y轴正方向指向左轮。机器人绝对速度正方向为 $x_{\{\text{Robot}\}}$ ，大小用 v 表示，其转角定义为将 $x_{\{o\}}$ 正方依顺时针转到 $x_{\{\text{Robot}\}}$ 正方向所经过的角度 θ ，相应的角速度用 ω 表示（见图2.4）。

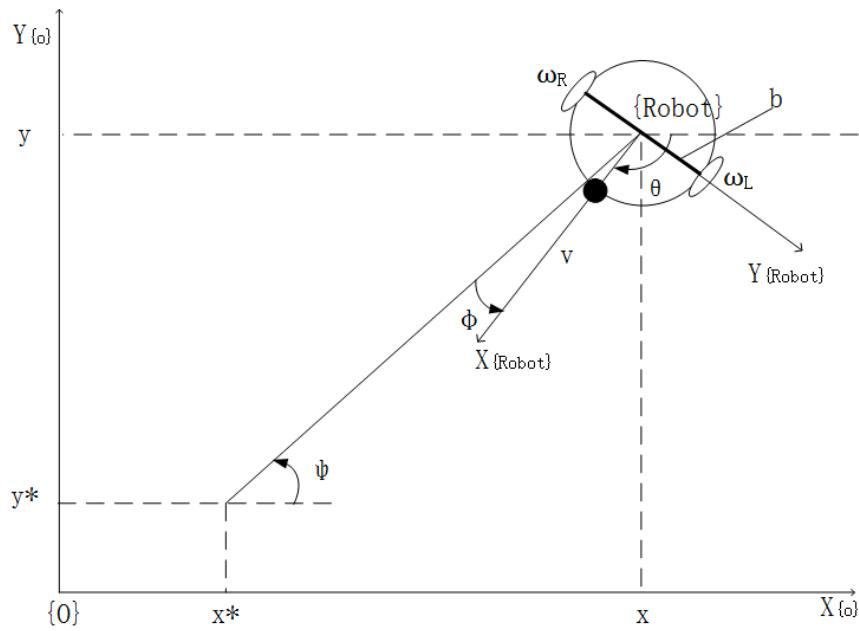


图 2.4: 机器人运动坐标系定义及各符号表示

给定模拟环境中任意一点 (x^*, y^*) 为机器人运动目标，定义 ϕ 为目标角，表示从机器人所在坐标到目标的射线按逆时针旋转到 $x_{\{\text{Robot}\}}$ 正方向所经过

的角度。在KSim的多智能体编队例程当中（见3.1.3节），需要将移动机器人的Unicycle模型近似变换为多智能体的单积分模型，将会用到目标角的定义。

2.5.2 定制障碍物

KSim本质上是不断重绘Figure窗体中的图形元素而形成动态用户界面的。障碍物描述信息保存在文件

“settings.xml”

当中，每个障碍物均使用xml文件的一个小节描述。每个小节包含世界坐标系 $\{O\}$ 的一个参考点以及四个形状描述参数。如需要描述图2.5当中的障碍物A、B，则需要图2.6的两个小结来描述。

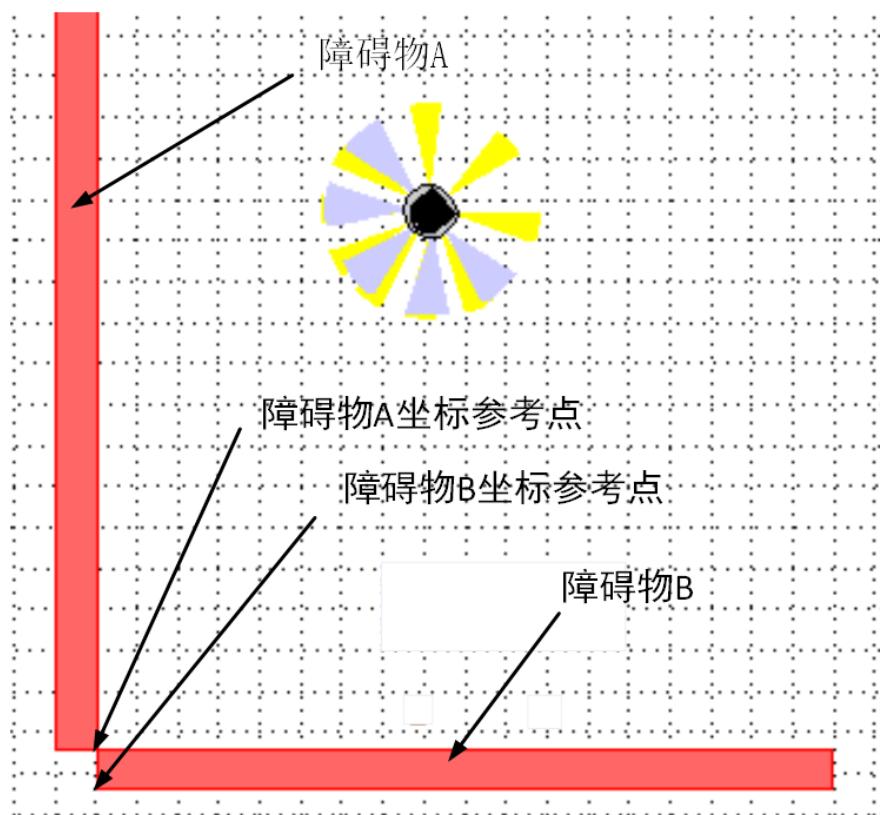


图 2.5: KSim的模拟环境中的障碍物

12 KSim:KheperaIII完全Matlab模拟

```
<obstacle><!-- 障碍物A描述小结 -->
  <pose x="-0.8955" y="-1.44" theta="0" /> <!-- 障碍物参考点 -->
  <geometry><!-- 几何形状描述 -->
    <point x="0" y="0" />
    <point x="1.791" y="0" />
    <point x="1.791" y="0.1" />
    <point x="0" y="0.1" />
  </geometry>
</obstacle>

<obstacle><!-- 障碍物B描述小结 -->
  <pose x="-0.8955" y="-1.34" theta="1.5708" /><!-- 障碍物参考点 -->
  <geometry><!-- 几何形状描述 -->
    <point x="0" y="0" />
    <point x="2.680" y="0" />
    <point x="2.680" y="0.1" />
    <point x="0" y="0.1" />
  </geometry>
</obstacle>
```

图 2.6: 障碍物描述小结

2.5.3 放置机器人

从KSim的模型库

“simulink \ KSim.mdl”

中向用户的Simulink窗体添加几个“BlueBird_Khepera3”模块则会在虚拟环境中出现几个KheperaIII机器人。在模拟开始以前，需要双击每一个模块，在如图2.7的窗体内为每个机器人设置不会重合的位姿。

如果模拟的机器人数量超过三个，则还需要将

“simulink \ +app \ ControlApp.m”

的构成函数中ArrayList设置为更大的数值。

```
obj.supervisors = simiam.containers.ArrayList(3);
```

KSim首先将需要绘制的图形元素压栈，然后再逐个处理。显然，如果设置过大的ArrayList数值，将会使得KSim申请大量无用的存储空间，进而影响系统的运行效率。



图 2.7: BlueBird_Khepera3模块设置界面

第三章 模拟研究

KheperaIII机器人融合了瑞士KTeam公司十年机器人研发经验，构造紧凑且传感器种类齐全，但缺点是价格高昂。一台KheperaIII机器人需要2600瑞士法郎，必须配置的组件Korebot II（即嵌入式系统板）也高达860瑞士法郎。这使得即便在学校KheperaIII机器人也成为稀缺设备。事实上，e-puck和乐高机器人较KheperaIII相对便宜，但其价格也不足以让教师允许学生带离实验室进行学习研究。

空洞的理论学习打击了学生对移动机器人的兴趣。Sim.I.am旨在解决这一问题，但是KSim解决的更好。它不但继承了Sim.I.am对于机器人平面运动和红外线传感器的模拟，而且可以模拟超声波传感器。有了KSim，学生们可以在教室、宿舍、甚至是草坪和操场看台等等任何他们愿意的地方学习移动机器人技术。当这些算法经过反复模拟验证以后，他们还可以利用四章的内容将算法直接转换为C代码放入机器人内部运行。下面我们将看到KSim如何逼真的将KheperaIII机器人带到学生面前的。

3.1 运动和编码器

KheperaIII是典型的两差动轮移动机器人，标准结构参数：

$$r_R = r_L = 20.5\text{mm}$$

$$b = 88.41\text{mm}$$

设 r_R 、 r_L 为机器人左右轮半径，则机器人的速度 (v, ω) 与两轮转速 (ω_R, ω_L) 的关系为^[1]：

$$\begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} = C \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \quad (3.1)$$

其中， C 为两差动轮机器人的结构参数矩阵，由式(3.2)定义。

$$C \triangleq \begin{bmatrix} \frac{r_R}{2} & \frac{r_L}{2} \\ \frac{r_R}{b} & -\frac{r_L}{b} \end{bmatrix} \in R^{2 \times 2} \quad (3.2)$$

16 KSim:KheperaIII完全Matlab模拟

驱动两侧轮子的直流电动机内部带有41：1的减速器，加之外部1.6：1的齿轮箱和软件4倍频，实际轮子转一圈可以得到

$$16 \times 41 \times 1.6 \times 4 = 4198.4$$

个脉冲。两侧轮子转速范围为：

$$\omega_{\min} = 0.44 \text{rad/s}, \quad \omega_{\max} = 9.56 \text{rad/s}.$$

两块DsPic单片机分别作为各自轮子的PID速度控制器。嵌入式系统板Korebot II（Intel XSCALE PXA-270@600MHz）通过*I²C*总线向PID速度控制器间接传送用户程序的速度指令。

3.1.1 轮子转速接口



图 3.1: KheperaIII的Simulink模块

在KSim中，KheperaIII对应Simulink模块为图3.1。该模块直接接受*rad/s*为单位的速度指令。考虑到移动机器人Unicycle模型的普遍应用，利用KSim的“BlueBird_uni_todiff”做模型之间的转换。需要注意的是，这种转换应用公式(3.1)，由于实际中机器人结构参数总会和标准参数有所差别，控制算法可能由于结构参数不准而造成模拟和实际结果相差甚远。获得相对准确的结构参数需要类似[1, 3, 9]的“结构参数校准实验”。针对移动机器人正常运行以前必做的功课，KSim给出了完善的实验工具和方法，请见4.2节。

在真实的KheperaIII机器人内，厂商向用户开放的速度控制接口需要输入一个长整数代表速度指令。这个长整数是实际的两轮转速指令(*rad/s*)乘以一个常量获得的^[5]。随着厂商对KheperaIII不断升级，这些常量随着机器人版本的不同而发生变化。

在KSim中，已经在“RealKteam”运行库中内置了所有相关信息。用户将控制器生成的以*rad/s*为单位的控制算法经Embedded Coder工具箱编译为C代码以后，直接将控制信号灌注至“RealKteam”运行库，运行库会自动处理这些信息。

3.1.2 轮子编码器接口

如图3.1，轮子的编码器信息从will_ticks端口输出。

在Simulink当中，编码器的脉冲数累加几乎可以不受约束的一直进行下去。但是真实的KheperaIII机器人编码器信息存储在一个32位整型当中，数值溢出后就会自动清零，这可能导航迹推算等底层算法错误。KSim的“RealKteam”运行库内置的航迹推算功能可以避免这一问题，并能够自动利用“结构参数校准实验”得到的信息。

3.1.3 实例：多智能体编队

移动机器人在二维环境下的编队总是能激发一批年轻学生的无限遐想！但是浩如烟海的多智能体论文却让他们望而却步。KSim提供了一个移动机器人二维编队的例程，在文件：

“simulink \ BlueBird_Forestation.slx”

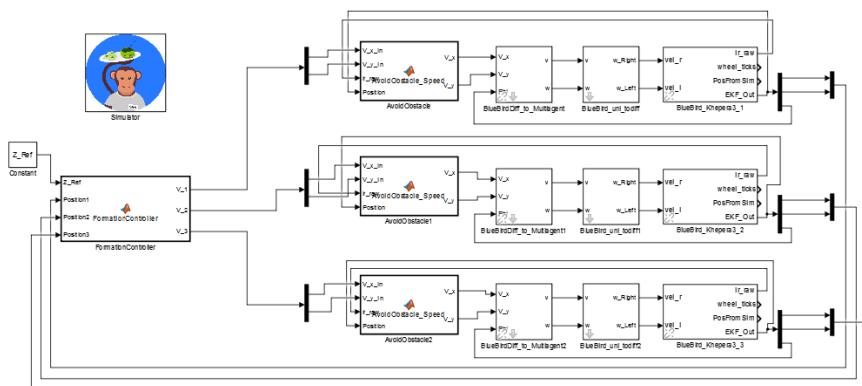


图 3.2: KSim多智能体编队例程

如图3.2所示，这个程序需要一个相对位置参考信息Z_Ref，可以输入：

$Z_Ref = [0.5 \ 0.5 \ -1 \ 0]$

为了保障例程精炼，程序利用多智能体控制算法编队的位姿信息来自于每个机器人利用超声波传感器的自定位过程。移动机器人受到噪声干扰而呈现

KSim:KheperaIII完全Matlab模拟

“飘动”的状态，如果不习惯这样，可以停止滤波器的工作。具体的方法请参考3.3.3节。

作为补充，我们录制了一段没有噪声干扰的KSim多机器人编队模拟的视频。

(视频：多智能体编队)

由于移动机器人Unicycle模型在x、y方向上的运动是耦合的，且不存在简单的线性化方法。KSim专门带有一个模块“BlueBirdDiff_to_Multiagent”解决这一问题（见图3.3），该模块可以将Unicycle模型近似变换为多智能体算法需要的单积分模型，原理是使用多层次控制。让移动机器人追踪二维平面上的一点 (x^*, y^*) ，将多智能体的单积分模型状态设置为这个点的坐标。通过多智能体协同算法控制这个点，也就间接控制了移动机器人。

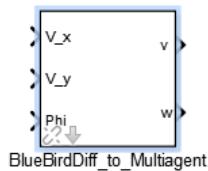


图 3.3: Unicycle模型向单积分模型转换模块

在此基础上，于x、y方向上分别应用一个“合作”协议^[12, 20]即可。算法简单，容易理解，学生还可以通过改变Z_Ref排出其他队形，激发学习兴趣。这种模型转换的思路虽然不同于很多对复杂个体模型“合作”协议^[21]的研究，但是却符合一般工程设计的思路。

3.1.4 模拟与实际的区别

KheperaIII机器人是两差动轮移动机器人，KSim的多智能体编队实验，需要首先将两差动轮机器人模型（根据式(3.1)）转换为Unicycle模型（使用模块“BlueBird.uni_todiff”），然后转换为多智能体单积分模型（使用模块“BlueBirdDiff_to_Multiagent”），这一系列转换需要准确的结构参数(3.2)。在模拟时，这当然不是问题，而实际当中则至关重要。KSim的“RealKteam”运行库，可以输出结构参数校准实验的结果，将这些信息重新加入模拟环境以检验控制算法的鲁棒性。

KheperaIII机器人两侧轮子驱动器作为物理器件，具有最高和最低速限制。当用户程序给出的控制指令高于最高转速时，KheperaIII的轮子转速也会高于[7]所规定的转速。这主要是因为KheperaIII采用了可以更换多种电动机的通用驱动器。一般情况下，当然不建议这样使用机器人。所以在工程上，当用户程序给出一个高于最高转速的指令后，仍然让驱动器工作于最高转速不变。这一功能已经加入到“RealKteam”当中。而在模拟时，也采取类似的方案。

用户程序给出低于最低转速的指令时，会造成KheperaIII转速不准确、抖动和突然的倒转。注意在很多情况下，需要移动机器人的驱动器能够工作在低于设计最低转速的条件，并且随机发生的状况不容易通过软件模拟。在设计控制算法时，需要将这一因素考虑在内。

3.2 红外线传感器

从红外线接近传感器获得准确的距离测量是比较困难的，如要做这方面的尝试可以参考文献[2, 11]。原因一方面是红外线接近传感器度数受到环境光线、角度、反射障碍物的颜色等多因素影响^[11]，另一方面是具有较强的随机噪声。KheperaIII机器人采用两次读取红外线传感器信息的方法，其中一次不发射测量光波。然后将两次测量的A\D转换数值相减，直接将这一数值输出给用户程序。图3.4显示了传感器9在暗室，有效距离范围内没有障碍物时的连续读数。

可见，即使在这样良好的条件下，仍然可以度到很强的噪声。基于这样的考虑，KSim沿用了Sim.I.am的距离信息获取方法。

图3.5截取自Sim.I.am的软件说明，作者采用一条确定的曲线近似传感器的测量特性。并且假设所有的传感器特性相同，这显然与实际情况有所区别。实

20 KSim:KheperaIII完全Matlab模拟

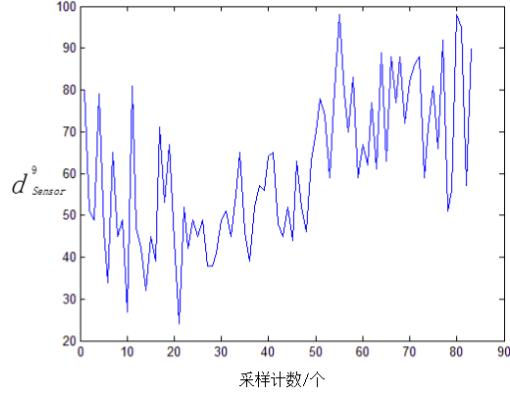


图 3.4: KheperaIII机器人红外线传感器9在暗室内的读数

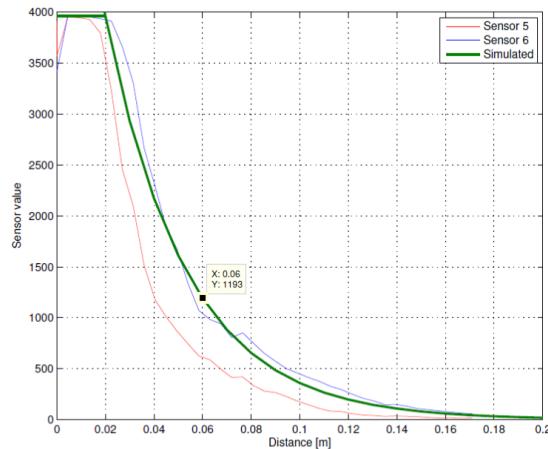


图 3.5: 红外线传感器”距离—读数“图

践显示，仅将接近传感器信息用于判断周围有无障碍物靠近，这种近似是简单和有效的。

3.2.1 红外线传感器接口

在模块图3.1的ir_raw端口上输出的传感器信息即利用图3.5生成。在KSim中，传感器编号与KheperaIII说明文档^[7]定义相同（见图3.6），实际中的红外线传感器位姿及坐标定义见图3.7。

KSim具有精确的KheperaIII机器人传感器位姿，这些信息存储在

“+simiam \ +robot \ Khepera3.m”

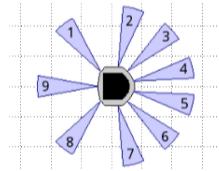


图 3.6: KSim 中的红外线传感器表示

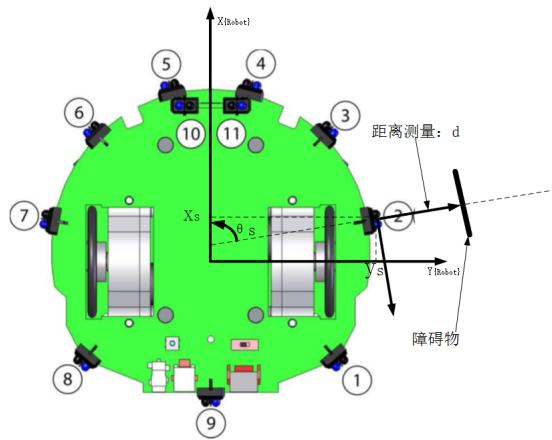


图 3.7: KheperaIII 机器人红外线传感器位姿

中，以方便使用。

3.2.2 模拟与实际的区别

KSim对红外线接近传感器的模拟虽然简单但是粗糙，它没有将光线，角度，反射障碍物的颜色等因素考虑在内，更为精确的红外线接近传感器特性请参见[2, 11]。

3.3 超声波传感器

KheperaIII具有5个固定于机身的测量范围为20~400cm的超声波传感器。每个传感器由一个400ST100型超声波发生器和一个400SR100型接收器组成（如图3.8），经由必要的信号调理电路统一连接到一颗DsPIC单片机A/D接口上。在单片机内计算TOF（time-of-flight）时间并换算为距离信息。返回的测量数据包括收到的回声个数，每个回声代表的距离（cm）、强度、以及探测到回声的时间（时间戳），传感器分辨率为1cm。

22 KSim:KheperaIII完全Matlab模拟



图 3.8: KheperaIII的超声波传感器

3.3.1 超声波传感器接口

超声波传感器类由

+simiam \ +robot \ +sensor \ UltrasonicSensor.m

定义，其属性location_List记录了传感器的精确位姿。与红外线接近传感器不同的是，声波在碰到墙壁等比空气密度明显偏大的障碍物以后，会发生镜面反射。这样，只有特殊形状的障碍物^[10]才具有将声波原路返回从而被传感器侦测到的特性。KSim真实再现了浙江工业大学信息工程学院嵌入式系统实验室的移动机器人实验平台，如图2.1所示其周围的矩形障碍物在超声波下呈现4个平面障碍和4个内拐角障碍，在“UltrasonicSensor.m”内具有针对两种障碍物的观测方程。

显然，障碍物的可见性还取决于传感器的位姿。在KSim中，这一功能可以利用

+simiam \ +robot \ +Filters \ EKF.m

的Obstacle_Visible()方法轻松获得。为了能够清晰的显示障碍物的可见性，可以利用ShowObject()方法，该方法在模拟界面上打印障碍物的可见状态。

3.3.2 传感器噪声特性的获取

精确模拟超声波传感器还需要获取可靠的传感器噪声特性参数。噪声特性参数获取实验环境如图3.9所示。激活KheperaIII机器人正前方的3号传感器，并将其放置于运动平面上，上面设置有水平仪保证声波水平发射。平面周围有木质围栏作为被测障碍物，围栏外侧有起固定作用的铁质框架。

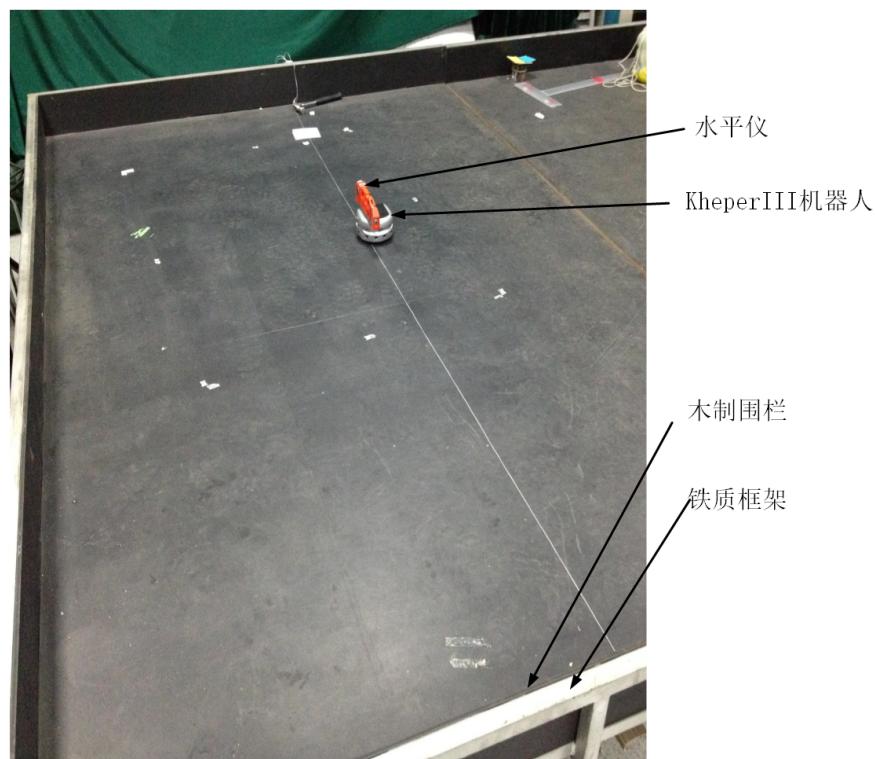


图 3.9: 超声波传感器噪声参数估计实验环境

KheperaIII机器人超声波传感器连续发射测量声波，并将得到的数据传回计算机。使用文献[19]的乘性噪声模型和最小二乘法估计噪声特性参数。KSim具有专门的类

```
+simiam \ +robot \ +sensor \ +noise \ Khepera3_US_Noise.m
```

保存这些信息。

24 KSim:KheperaIII完全Matlab模拟

3.3.3 实例：利用EKF自定位

滤波估计算法通常需要诸如控制信息、机器人当前位姿，障碍物可见情况等相对比控制算法更多的资源，KSim鼓励用户使用面向对象的Matlab语言来编写这些算法，并给出了使用扩展卡尔曼滤波自定位的例程。该例程位于文件

simulink \ BlueBird_RobotLocation.slx

用于移动机器人自定位的扩展卡尔曼滤波算法并不特殊，由于超声波镜面反射特性，需要不断的根据障碍物的可见情况变换观测方程且对得到的数据与地图进行匹配。针对KSim默认模拟环境的地图存储在

+simiam \ +robot \ +map \ Map_Rectangle.m

地图的数据格式遵守文献[10]。如果重新定制了模拟环境，需要重新绘制地图。在某一特殊位姿，所有的超声波传感器可能均无法测量到障碍物，滤波器只能将预测信息作为滤波的结果。滤波器的输出在图3.1的“EKF_Out”端口上，以便直接在Simulink环境下分析结果。有关扩展卡尔曼滤波器详细原理请参考[4]，针对移动机器人的滤波估计研究请参考[10]。

下面的视屏显示了KSim环境中，一台KheperaIII机器人利用超声波传感器确定位姿，同时利用红外线传感器避免与墙壁撞击的情况。

(视频：移动机器人自定位)

为了分析滤波器效果，通常需要比较实际位姿与估计值。“BlueBird_Khepera3”模块“PosFromSim”端口输出的机器人真实角度会随着机器人的旋转逐渐累加。而滤波算法通常输出 $[\pi, -\pi]$ 的角度值，不便于直接比较。



图 3.10: 角度修正模块

KSim准备了图3.10的角度修正模块，可以将角度值调节至 $[\pi, -\pi]$ 的范围内。KSim建议通过直接修改“EKF.m”文件中的代码来编写自己的滤波估计算法。图3.11显示了面向对象的Matlab语言呈现的扩展卡尔曼滤波器代码块，可见其简洁明了。

```
%下面加入滤波器运算逻辑
%首先生成滤波器需要的控制信号：
EKF_u = obj.Filter.Prediction_u_Maker(dt, vel_r, vel_l); %产生滤波器需要使用的驱动信息
obj.Filter.Prediction_Pose_EKF(EKF_u); %预测位姿
obj.Filter.Prediction_Variance_EKF(); %预测方差
obj.Filter.Obstacle_Visible(pose); %利用预测查询地图，观测那些障碍物可见
obj.Filter.Pseudo_observation(pose); %根据预测的可见障碍物，产生伪观测信息
obj.Filter.Guess_Data_MeanVariance(); %根据伪观测信息，产生应有的噪声信息（包括模拟需要的噪声）
obj.Filter.InnovationMaker(true); %true: 产生用于模拟的新息
obj.Filter.Match(true); %做匹配操作
obj.Filter.Estimate(); %生成估计

obj.Filter.Temp_ObservationShow(); %显示观测数据库信息
obj.Filter.ShowObject(); %在屏幕上打印观测到的障碍物
obj.Filter.CleanAllObstacleData(); %清楚掉ObstacleData
```

图 3.11: 使用面向对象语言编写的滤波器代码

3.3.4 模拟与实际的区别

KSim虽然使用了乘性噪声模型模拟传感器噪声特性，但是真实的KheperaIII机器人超声波传感器可能具有色噪声，并且随着反射障碍物密度的不同，其噪声特性也相应地发生改变。这些因素在实际中会恶化滤波效果。

第四章 实验研究

再没有比将理论的东西变为现实更激动人心的了。KSim提供了完善的将算法变为嵌入式代码的解决方案，不需要重新编写代码，就能将算法运行于KheperaIII机器人内。

注意，如果一种控制或滤波算法在模拟环境下都无法正常工作的话，那么即使将其转换为C代码也不大可能奏效。KSim推荐操作真实的KheperaIII机器人以前，首先在模拟环境下对算法进行彻底的测试。

4.1 遥控机器人方案

将KheperaIII机器人当作遥控玩具，所有算法运行在上位机的确是一种可行的解决方案。乔治理工大学的GRITS实验室，中国的浙江大学，均采用这种方案。甚至是KTeam公司本身也推出了一款名为Kh3interface^[5]的软件集合。配合Matlab的Instrument Control Toolbox可以像玩具一样通过WIFI遥控KheperaIII机器人。

这样做，缺陷也是明显的。上位机的运算能力可以非常强大，这鼓励研究者越来越多的采用运算相对复杂的算法，而不用担心被控对象的承载能力。ios、Android等优秀的嵌入式系统成功经验告诉我们，时间或空间复杂度低而性能可靠的计算方法才是真正需要的。有些时候，移动机器人需要完全停止运动以等待某些算法计算完毕，这些行为明显不能够在基于“遥控机器人方案”的实验平台上开发。

4.2 准备C语言库文件

Embedded Coder工具箱生成的C代码必须与KSim的“RealKteam”运行库结合才能在Linux环境中生成合适的KheperaIII应用程序。首先应参考[6, 7]配置好KheperaIII在Linux中的开发环境，然后将整个“RealKteam”文件夹内的文件全部引入该环境中。“RealKteam”运行库数据结构及函数接口请参见[15]。由于“RealKteam”依赖“libkorebot”，所以用于直接操纵硬件的“i2cal”、“i2c stream”、“nmea”不再使用。尝试编译

KSim:KheperaIII完全Matlab模拟

“RealKteam \ Odometry_Calibration”

文件夹下的所有C代码，编译成功之后首先运行“UMBmark_Test_Odometry”，在没有校准结构参数以前，KheperaIII应该能够做视频内的动作。

(视频：结构参数校准实验)

可以利用[16]的简略方法进行结构参数校准，如果KheperaIII机器人的固件版本高于3.0，则使用

“RealKteam \ Odometry_Calibration \ calibration.pl”

内的脚本文件进行校准实验。更精确的校准方法请参考[1, 3, 9]。

4.3 将算法运行在机器人内

直接生成C代码而避免对算法的重新编写，Matlab很早就提供了Embedded Coder工具箱来满足这类需求。缺点是，Embedded Coder工具箱需要周全的配置才能有效运作。在汽车软件开发行业，甚至需要第三方公司在Embedded Coder工具箱基础上做二次开发，以提供某一型号汽车的专用C代码转换工具。

KSim解决这一问题的方法首先是规范使用的Matlab函数，鼓励用户将算法输入至“MATLAB Function”中。该模块可以对算法用到的函数加以筛选，从而在算法开发初期就避免使用那些不能被转换为C代码的函数。“MATLAB Function”内也可以使用面向对象的编程，只要所有涉及的

函数均是Embedded Coder允许的，即可以做C代码转换。如果使用了无法转换的函数，比如“figure”，KSim会在模拟的时候就提示出错，而不是等到编译C代码的时候才提示用户。

注意到图3.2带有一个避障模块“AvoidObstacle”，该模块在“MATLAB Function”中编写，KSim的模拟已经证明它是有效的。作为示例，将“AvoidObstacle”转换为C代码。

Step1：将这个模块复制到一个空白的Simulink窗体中。

Step2：在Simulink—>Model Configuration Parameters，选择“Solver—>Solver options”，类型选择“Fixed-step”。

Step3：在“Code Generation”内选择“System target file”为“ert.tlc”。

Step4：在“Code Generation”内选择“Report”，选择“Create code generation report”和“Open report automatically”。

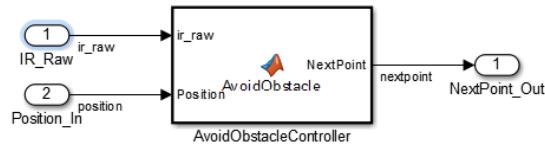


图 4.1：为每一个输入输出端口命名

Step5：为这个模块的每一个端口添加“Inport”、“Outport”，并在每个连接线上命名，如图4.1。

Step6：打开Model Explorer，选中“Base Workspace”，在菜单内选择“Add”，然后选“Add custom”。添加“mpt.Signal”，每一个“mpt.Signal”的名称与Step5中的命名相同。其属性根据实际端口属性填写，比如“ir_raw”的属性应如图4.2填写。

Step7：右键单击信号线，选择“Properties”，选择“Signal name must resolve to Simulink signal object”。每一条信号线均这样选择。Simulink窗口应该为图4.3。

Step8：重新打来Simulink—>Model Configuration Parameters，选择“Code Generation”，“Interface”，“Configure Model Functions”。在弹出的对话框中设置熟悉的C语言函数结构，如图4.4所示。

30 KSim:KheperaIII完全Matlab模拟

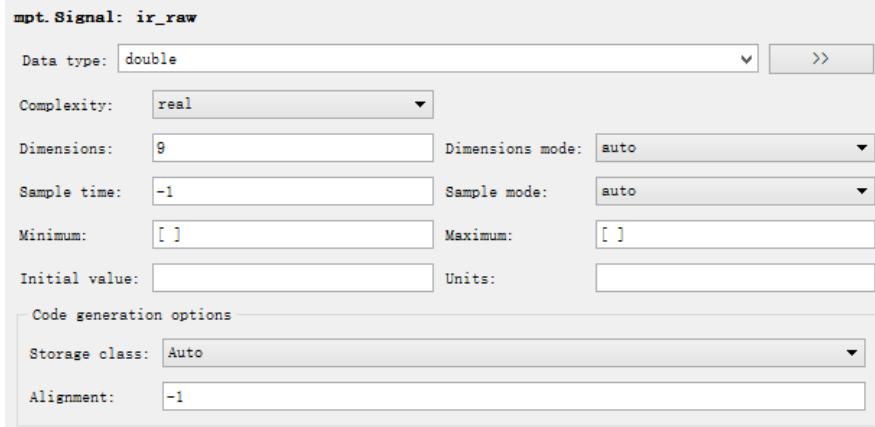


图 4.2: ir_raw的属性

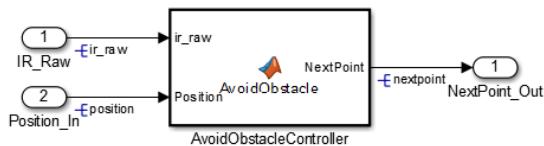


图 4.3: 信号名与信号对象配对

Step9: 点击“Ctrl+B”快捷键，等待编辑过程完成，就会出现“Code Generation Report”，这份Matlab自动生成的报告中已经详细说明了如何使用生成的C代码。

将KSim的“RealKteam”内的文件与生成的C代码一同引入的Linux编程环境。这些库文件的入口函数结构可以与生成的C代码兼容，编写C代码建立两者关联。

“RealKteam”内的库文件不但可以适用于任何版本的Khepera III机器人，而且可以兼容Khepera III Toolbox^[17]内的所有程序。但是这些程序在使用以前，必须重新利用“RealKteam”库文件进行编译。

4.3.1 实例：Goto-Goal

利用航迹推算引导机器人运动到平面内指定的一点，及Goto-Goal实验，是移动机器人学习的经典课题。KSim带有一个基于能量函数的Goto-Goal算法，它不但可以引导机器人到达目标点，且能够避免受到KheperaIII机器人驱动器最小转速限制的影响。作为练习，可将这个算法从“Simulink”文件夹

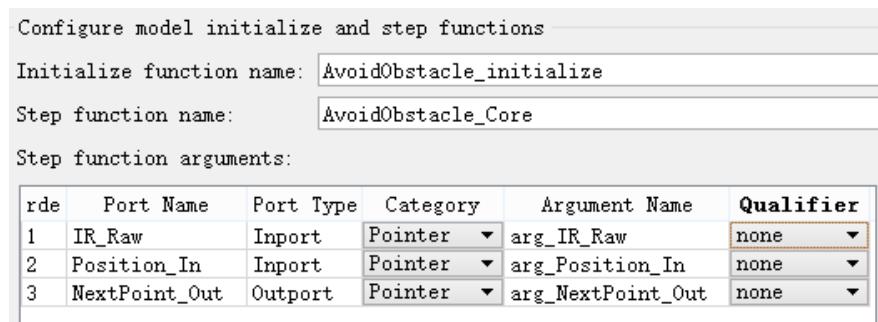


图 4.4: 为每一个输入输出端口命名

中复制出来，利用KSim的模拟环境加以验证，利用Embedded Coder工具箱转换为C代码，最后在Linux下利用“RealKteam”库文件编译。该算法的运行效果见下面的视屏。

(视频: Goto-Goal)

参考文献

- [1] G Antonelli, S Chiaverini, and G Fusco. A calibration method for odometry of mobile robots based on the least-squares technique: theory and experimental validation. *Robotics, IEEE Transactions on*, 21(5):994–1004, 2005.
- [2] Gines Benet, Francisco Blanes, José E Simó, and Pascual Pérez. Using infrared sensors for distance measurement in mobile robots. *Robotics and Autonomous Systems*, 40(4):255–266, 2002.
- [3] J Borenstein and Feng Liqiang. Measurement and correction of systematic odometry errors in mobile robots. *Robotics and Automation, IEEE Transactions on*, 12(6):869–880, 1996.
- [4] C K Chui and G Chen. *Kalman Filtering: With Real-Time Applications*. Springer, 2009.
- [5] K-Team Corporation. Khepera III. <http://www.k-team.com/mobile-robotics-products/khepera-iii/specifications>.
- [6] K-Team Corporation. KoreBot Library Documentation. <http://ftp.k-team.com/KorebotII/software/common/libkorebot/api/html/index.html>, 2012.
- [7] J.Tharin F.Lambercy. *KheperaIII user manual*, volume 3.5. K-Team S.A., Switzerland, 2013.
- [8] Jean Pierre de la Croix. Sim.I.am. <http://jpdelacroix.com/simiam/>, 2013.
- [9] A Kelly. Fast and easy systematic and stochastic odometry calibration. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3188–3194 vol.4, 2004.

34 KSim:KheperaIII完全Matlab模拟

- [10] John J Leonard and Hugh F Durrant-Whyte. *Directed sonar sensing for mobile robot navigation*, volume 448. Kluwer Academic Publishers Dordrecht, 1992.
- [11] Paul M Novotny and Nicola J Ferrier. Using infrared sensors and the Phong illumination model to measure distances. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1644–1649. IEEE, 1999.
- [12] Reza Olfati-Saber and Richard M Murray. Consensus problems in networks of agents with switching topology and time-delays. *Automatic Control, IEEE Transactions on*, 49(9):1520–1533, 2004.
- [13] Opensource.org. The BSD 3-Clause License. <http://opensource.org/licenses/BSD-3-Clause>.
- [14] Bahr A Prorok A, Arfire A. Khepera III Toolbox. http://en.m.wikibooks.org/wiki/Khepera_III_Toolbox, 2012.
- [15] Bahr A Prorok A, Arfire A. Khepera III Toolbox Modules. http://en.m.wikibooks.org/wiki/Khepera_III_Toolbox/The_Toolbox/Modules, 2012.
- [16] Bahr A Prorok A, Arfire A. Khepera III Toolbox odometry calibration. http://en.m.wikibooks.org/wiki/Khepera_III_Toolbox/Examples/odometry_calibration, 2012.
- [17] Bahr A Prorok A, Arfire A. Khepera III Toolbox Programs. http://en.m.wikibooks.org/wiki/Khepera_III_Toolbox/The_Toolbox/Programs, 2012.
- [18] Andy H Register. *A guide to Matlab object-oriented programming*. Chapman & Hall/CRC, 2007.
- [19] Wang Xingbo, Fu Minyue, and Zhang Huansui. Target Tracking in Wireless Sensor Networks Based on the Combination of KF and MLE Using Distance

- Measurements. *Mobile Computing, IEEE Transactions on*, 11(4):567–576, 2012.
- [20] Daniel Zelazo, Amirreza Rahmani, and Mehran Mesbahi. Agreement via the edge laplacian. In *Decision and Control, 2007 46th IEEE Conference on*, pages 2309–2314. IEEE, 2007.
- [21] Lijun Zhu and Zhiyong Chen. Robust homogenization and consensus of nonlinear multi-agent systems. *Systems & Control Letters*, 65:50–55, 2014.

致 谢

感谢浙江工业大学信息工程学院俞立、王宏霞和欧林林老师对该项目给予的支持。感谢陈博同学的支持。

王宏霞老师在KheperaIII移动机器人超声波传感器噪声特性估计实验过程中给予了热情的鼓励和技术性指导。

欧林林老师在KSim的“RealKteam”库文件开发过程中提供了一台KheperaIII移动机器人。

陈博同学在滤波估计技术基础知识学习过程中提供了资料。

KSim系统受《浙江省大学生科技创新活动计划（新苗人才计划）》资助。

.....

谨把本文献给我最敬爱的父母！