基于 PID 控制的 NAO 机器人循线行走技术研究

秦 娇,王向华
(山东科技大学 电气与自动化工程学院,山东 青岛 266590)

摘 要:智能机器人的循线行走技术在工业安全领域具有重要意义。选用 NAO 机器人作为研究平台,创新点在于将其在平面上的运动学动态建模为线性定常离散系统,并将摄像头获取到的图像转化成参考输入信号,由此将循线行走问题转化成信号跟踪问题。与传统的渐进跟踪方法不同,基于改进后的数字增量型 PID 算法设计机器人行走的控制器,借用 Simulink 仿真平台通过凑试法确定控制器参数,最后用 Python 语言编程实现了控制 NAO 在白色地板上循黑线行走的目标,Webots 软件仿真证实了该算法有效。

关键词:循线行走;信号跟踪控制;NAO 机器人;PID 控制

中图分类号:TP242.6 **文献标志码**:A **文章编号**:1672-3767(2017)04-0087-09
DOI:10.16452/j.cnki.sdkjzk.2017.04.013

Research on Line-tracking Walking Technique of NAO Robot Based on PID Control

QIN Jiao, WANG Xianghua
(College of Electrical Engineering and Automation, Shandong University of Science and Technology, Qingdao, Shandong 266590, China)

Abstract: Line-tracking walking technique is an essential part of intelligent robot technology. This paper selected NAO robot as the research platform to study the line-tracking walking problem. Firstly, the kinematics of the robot in the plane was dynamically modeled as a linear constant discrete system, and then the image obtained by the camera was transformed into a reference input signal so that the problem of line-tracking walking was transformed into a signal tracking problem. Different from the traditional incremental tracking method, this paper designed the controller based on the improved digital incremental PID(proportion integration differentiation) algorithm and determined the controller parameters through the concise test method on Simulink. Finally, it was realized by programming in Python language. The simulation results of Webots software show that the proposed algorithm in this paper is effective.

Key words: line-tracking walking; signal tracking control; NAO robot; PID control

智能机器人是一种在思想和行为等方面全面模拟人类的可编程自动化装置,具备一定的环境认知能力以及做出相应决策的能力。现代化的工程技术系统正朝着大规模、复杂化的方向发展,这类系统一旦发生事故就可能造成人员和财产的巨大损失[1]。而具有循线行走功能的智能机器人可以在一些环境极其恶劣的场合,帮助或代替人类完成一些高精密度、高工作量和高危险性的巡查工作,发现设备异常时能够发出报警信号或进行简单的故障处理,避免进一步的损失,因此研究智能机器人的循线行走技术在工业安全领域具有重要的现实意义。

收稿日期:2017-04-05
基金项目:青岛市博士后应用研究项目(2015179);山东省博士后创新项目
作者简介:秦 娇(1994—),女,山东济宁人,硕士研究生,主要从事故障诊断与容错控制研究.
王向华(1986—),女,山东威海人,博士后,主要从事制导与控制研究,本文通信作者.
E-mail:xianghuaw@pku.edu.cn

目前关于循线机器人的研究可大致分为两类，一是基于各项智能车大赛的四轮小车研究为主，如文献[2]设计了基于 PID 控制器的行走驱动算法，实现了对依靠双轮直流电机驱动的机器人平台的循线行走控制；二是致力于对高压输电线上的悬挂式巡线机器人的研究，如文献[3]针对无人值守变电站巡检机器人导航定位的问题，制定了一种基于引导线的包括循线行走和定点检测两部分的单目视觉导航方案。随着人形智能机器人逐渐进入人们的日常生活，实现人形智能机器人的循线行走控制也就提上了日程。目前智能机器人对外界环境的感知仍主要依靠摄像头来实现，所以图像检测处理技术和目标识别与跟踪技术是进一步发展智能机器人循线行走技术的基础。由法国 Aldebaran Robotics 公司研制并生产的 NAO 机器人可以实现音响定位、探测视觉图像、感知障碍物等功能[4]，但目前的相关研究成果大多只涉及到如何实现 NAO 机器人对固定或移动的单个目标的检测与跟踪，如文献[5]提出一种改良后的检测方法降低了光照的影响，从而缩短了机器人的反应时间；文献[6]提出一种通过将 NAO 摄像头获取的图像和声纳传感器提供的信息进行融合来寻找运动路径的方法；文献[7]提出一种机器学习与特征匹配相结合的方法以提高 NAO 机器人目标识别的正确率，在室内光线无遮挡的情况下取得了较好的跟踪效果。

理论上讲，控制 NAO 机器人循线行走可视作一种跟踪问题，然而要实现系统的渐进跟踪，现有研究大多需要基于确定性的参考输入信号，而由于导引线事先未知，传统方法便不再适用。本研究主要工作内容包括：①将 NAO 机器人在平面上的运动学动态建模为线性定常离散系统，并将循线行走问题转化成信号跟踪问题；②由机器人自带的摄像头每隔 30 s 采集一次图像，通过一些技术和技巧将图像信息转化成参考输入信号；③将机器人位置与参考输入信号的差作为控制器输入构成输出反馈控制器；④采用改进后的数字增量型 PID 算法设计控制器，并通过 Simulink 环境下的仿真完成参数整定；⑤用 Python 语言编写程序，在 Webots 软件中完成 NAO 机器人循线行走仿真实验。

1 系统模型

以往研究通常只对NAO机器人的行走控制系统进行建模，本研究忽略NAO机器人高度上的改变，只考虑在平面上的运动，可将其运动学动态过程近似建模为一个线性的定常离散系统。但由于 NAO 机器人的摄像头视野范围有限，无法一次性获取全部的导引线信息，所以需要每隔一段时间 T_0 重新获取图像。将 NAO 机器人看作质点，若将导引线按照时间周期 T_0 划分成若干段，则每一段都能够以 NAO 机器人所在位置为原点建立如图 1 所示的二维平面直角坐标系。

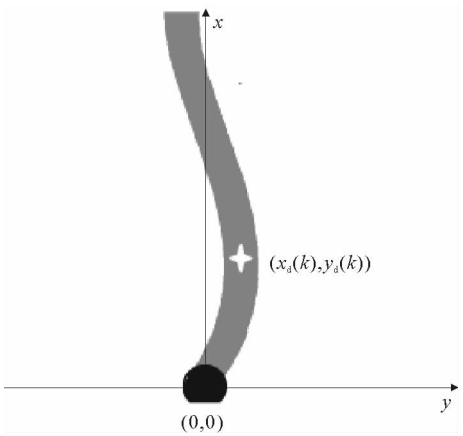


图 1 平面直角坐标系

Fig. 1 Rectangular coordinate system

图 1 中(0,0)点为 NAO 机器人的初始位置，初始面向为 x 轴正方向，$(x_d(k),y_d(k))$ 为 NAO 在 k 时刻期望达到的目标点。设计目标就是将 NAO 机器人的实际行走运动分解为 x 轴方向和 y 轴方向两个单独的运动，控制其从初始位置出发后不断跟踪目标点从而实现循线行走的功能。

由图 1 所示，设 NAO 机器人在 k 时刻的真实位置坐标为 $(x(k),y(k))$，采样周期为 T。设 k 时刻 NAO 机器人在 x 轴方向和 y 轴方向的速度和加速度分别为：$v_x(k)$，$a_x(k)$ 和 $v_y(k)$，$a_y(k)$，则有如下运动学方程：

$$\begin{cases} x(k+1)=x(k)+T\cdot v_x(k)+\dfrac{T^2}{2}\cdot a_x(k) \\ v_x(k+1)=v_x(k)+T\cdot a_x(k) \end{cases},0<T<T_0; \tag{1}$$

$$\begin{cases} y(k+1)=y(k)+T\cdot v_y(k)+\dfrac{T^2}{2}\cdot a_y(k) \\ v_y(k+1)=v_y(k)+T\cdot a_y(k) \end{cases},0<T<T_0。 \tag{2}$$

选取系统控制向量 $\boldsymbol{u}(k)=[a_x(k),\ a_y(k)]^{\mathrm{T}}$，状态向量 $\boldsymbol{\Phi}(k)=[x(k),\ v_x(k),\ y(k),\ v_y(k)]^{\mathrm{T}}$，输出向量 $\boldsymbol{\Psi}(k)=[x(k),\ y(k)]^{\mathrm{T}}$，即可建立如下线性离散系统模型：

$$\begin{cases}\boldsymbol{\Phi}(k+1)=\boldsymbol{A}\boldsymbol{\Phi}(k)+\boldsymbol{B}\boldsymbol{u}(k)\\ \boldsymbol{\Psi}(k)=\boldsymbol{C}\boldsymbol{\Phi}(k)\end{cases}。\tag{3}$$

其中，$\boldsymbol{A}=\begin{bmatrix}1 & T & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & T\\ 0 & 0 & 0 & 1\end{bmatrix}$，$\boldsymbol{B}=\begin{bmatrix}T^2/2 & 0\\ T & 0\\ 0 & T^2/2\\ 0 & T\end{bmatrix}$，$\boldsymbol{C}=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}$。

显然系统(3)是可控可观的。设系统参考输入 $\boldsymbol{r}(k)=[x_{\mathrm{d}}(k),y_{\mathrm{d}}(k)]^{\mathrm{T}}$，令 $\boldsymbol{e}(k)=\boldsymbol{r}(k)-\boldsymbol{\Psi}(k)$，则 NAO 机器人的循线行走控制问题就转化为跟踪问题，即设计控制器使 $\boldsymbol{e}(k)\to 0$。当然事实上，由于要受到 NAO 机器人实际步长范围限制以及各种干扰因素的影响，不可能实现机器人每一步的误差 $\boldsymbol{e}(k)$ 都为 0，只要尽可能使每一步的 $\boldsymbol{e}(k)$ 最小即可保证机器人能够循线行走。

另外，由于 NAO 机器人默认的行走每一步的最大步长限制为 x 轴方向 4 cm，y 轴方向 14 cm，即 $a_x(k)$，$a_y(k)$ 具有上下界限制，所以需要对 $\boldsymbol{u}(k)$ 加入饱和环节 $\boldsymbol{g}[\boldsymbol{u}(k)]$ 如下式所示。

$$\boldsymbol{g}[\boldsymbol{u}(k)]=\begin{cases}[-0.08\quad -0.18]^{\mathrm{T}}, & [u_x(k)\quad u_y(k)]^{\mathrm{T}}<[-0.08\quad -0.18]^{\mathrm{T}}\\ [\ 0.08\quad\ 0.18]^{\mathrm{T}}, & [u_x(k)\quad u_y(k)]^{\mathrm{T}}>[-0.08\quad -0.18]^{\mathrm{T}}\\ \boldsymbol{u}(k),\text{其他情况}\end{cases}。\tag{4}$$

2　具体实现

2.1　图像处理

NAO 机器人拥有两个高清摄像头，具备 920 万有效像素，更新速度达每秒 30 帧，可以用于识别跟踪不同的物体，以及产生并存储图像或视频文件。NAO 机器人不能同时使用两个摄像头，本设计中主要采用其位于嘴部的摄像头来获取地面信息，摄像头的角度参数如图 2 所示。

NAO 机器人获取的图像信息可以通过格式转化以数组的形式存储，由于彩色图像得到的是三维数组，处理起来比较复杂，在假设 NAO 机器人行走的地面为白色，导引线为黑色的情况下，实际上只需要关心黑色导引线的信息，所以可对图像进行灰度化处理，最终得到一个 240×320 大小的一维数组，数组中的每一个元素值即为图像对应像素点的亮度，压缩了数据量，简化了计算过程。

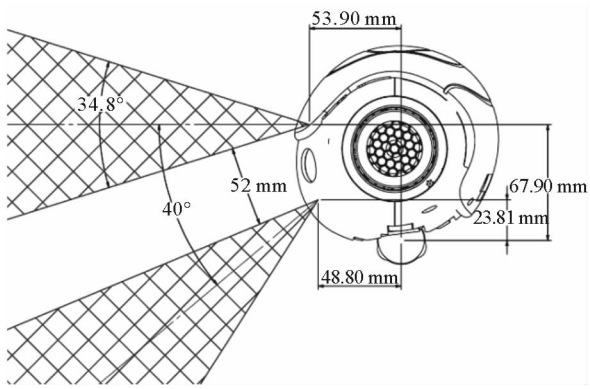


图 2　NAO 机器人摄像头角度参数

Fig. 2　The camera angle parameters of NAO

理想情况下，NAO 机器人通过摄像头获取到的图像包含且只包含一定视线范围内的地面导引线信息。但在实际情况中，图像的采集过程总是或多或少地受到各类干扰，比如系统内部电路、机器人的机械运动以及外部电磁波等都会产生噪声，进而影响图像品质。另外，机器人视野内难免会有其他物体出现并影响导引线边界信息的提取。针对第一种情况，本研究采用高斯滤波对噪声进行抑制和消除；而对于后一种情况，本研究将采用阈值分割的方式对图像进行二值化处理，其中的阈值通过 Otsu 算法选取。Otsu 算法[7]是日本的大津展之在 1980 年提出的一种非参数和无监督自动选取阈值的全局阈值法，其基本思想是设阈值将图像中的每一个像素点按照其灰度等级分割成两类，其中一类对应目标物体，另一类对应背景，使得这两类灰度等级的类内方差最小，类间方差最大的阈值即为最佳阈值。

另外在图像处理的最后部分，为了便于提取导引线的形状特征，本研究对图像反色后进行了先腐蚀再膨胀的形态学处理。

2.2 参考输入

已知型号为 H25 的 NAO 机器人嘴部摄像头距离地面高度约为 52.42 cm，配合图 2 中的摄像头角度参数可得图 3。

设定 NAO 机器人的行走频率为每 1 s 行进一步，由于 NAO 机器人 x 轴方向的单步最大步长为 4 cm，所以理论上 NAO 机器人走完每幅图像的长度最少需要 24 s。为了在 NAO 机器人有可能偏离导引线行走时尽快给予更正，理论上对于图像采集周期 T_0 的选取应越小越好。极限情况下若将 T_0 取为 1 s，则 NAO 机器人仅依靠实时获取的图像信息即可实现循线行走，而无需设计控制器。但图像的更新频率越快，其处理过程调用越频繁，占用内存就越大。所以本设计建立在假设机器人无法实时获取图像的基础上，最终综合考虑各项因素选取 $T_0=30$ s。

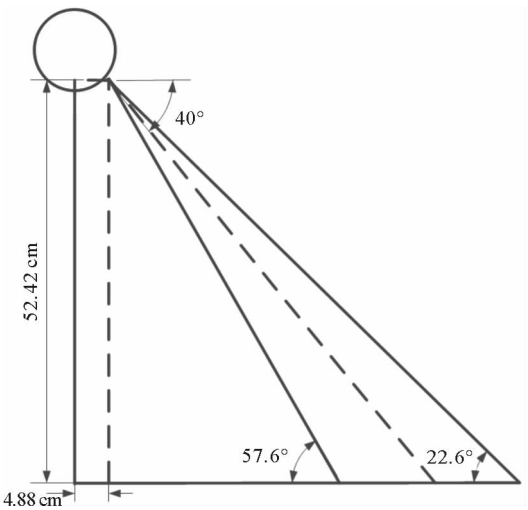


图 3 NAO 嘴部摄像头 x 轴上的视野范围

Fig. 3 Range of vision on the x axis of NAO's mouth camera

图像采集与处理过程结束后，对储存在数组中的图像信息每隔 8 行（约 3 cm）沿 y 轴正方向扫描，当扫描到导引线边缘即白色的像素点（对应数组内数值为 0）时将此点的列坐标加 1（原数组坐标编号是 0 到 319，即第 3 列的列坐标值为 2，为便于后续讨论，此处将坐标值+1）记为 hy，并跳出这一行的扫描过程，如果某一行没有扫描到白色像素点，则记录 hy 值为 0。

由于图像扫描顺序为从左到右，而图 1 所示坐标系的 y 轴位于图像中心，设导引线宽度为 18 个像素，则 $(hy+8-320/2)$ 的值就是导引线中点在 y 轴上的坐标。又由于图像扫描顺序为从上到下，所以需要将得到的坐标值重新倒序排列后才能作为系统在 y 轴方向上的参考输入 $y_d(k)$ 使用。由于 $y_d(k)$ 的值是每隔 8 行获取一次，所以 x 轴方向上的参考输入可设定为 $x_d(k)=0.03k$。另外，由图 3 可知 NAO 机器人身前存在 x 轴上长度为 38.15 cm 的盲区，也就是说 NAO 机器人实际上需要在 10 s 后才能真正抵达图像所示范围内，所以需要将上一次获取到的图像的相应位置的信息复制到这次的参考输入中用以填补盲区。而对于第一次图像获取时的盲区，默认其参考输入为 $x_d(k)=0.03k$，$y_d(k)=0$。

2.3 控制器设计

2.3.1 数字 PID 控制算法

由问题描述可知系统整体结构如图 4 所示，其中的控制器设计通过计算机编程实现。

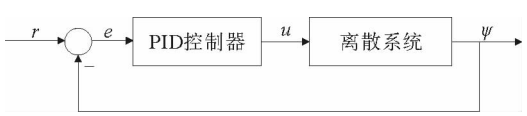


图 4 系统整体结构框图

Fig. 4 General structure of system

数字控制器的设计方法主要有直接和间接两种，直接设计法中最常用的是最少拍控制器，然而考虑到最少拍控制器对不同输入信号的适应性较差，所以选用间接设计中的 PID 控制作为系统控制器。在模拟控制系统中，PID 控制器[8]为：

$$\boldsymbol{u}(t)=\boldsymbol{K}_{\mathrm{p}}\left[\boldsymbol{e}(t)+\frac{1}{T_{\mathrm{i}}}\int_0^T\boldsymbol{e}(t)\,\mathrm{d}t+T_{\mathrm{d}}\,\frac{\mathrm{d}\boldsymbol{e}(t)}{\mathrm{d}t}\right]+\boldsymbol{u}_0\ 。\tag{5}$$

其中 $\boldsymbol{u}(t)$ 和 $\boldsymbol{e}(t)$ 为 2×1 的列向量，$\boldsymbol{u}_0$ 是 $\boldsymbol{e}(t)=0$ 时的基准控制量，本设计中 $\boldsymbol{u}_0=[0,0]^{\mathrm{T}}$ 由于计算机控制只

能根据采样时刻的偏差值计算控制量，式(5)中的积分和微分项都不能直接计算，只能通过数值计算的方式逼近。当采样周期 T 足够小时，可以用求和代替积分，用后向差分代替微分，得到数字 PID 位置型控制算法如下：

$$\boldsymbol{u}(k)=\boldsymbol{K}_{\mathrm{p}}\left[\boldsymbol{e}(k)+\frac{T}{T_{\mathrm{i}}}\sum_{j=0}^{k}\boldsymbol{e}(j)+T_{\mathrm{d}}\,\frac{\boldsymbol{e}(k)-\boldsymbol{e}(k-1)}{T}\right]。\tag{6}$$

由式(6)可以看出，这种位置型控制算法需要对偏差 $\boldsymbol{e}(j)$ 进行不断累加，导致内存占用较大，从而造成控制器实时性差。因此可考虑增量型控制算法，令 $\Delta\boldsymbol{u}(k)=\boldsymbol{u}(k)-\boldsymbol{u}(k-1)$，可得：

$$\Delta\boldsymbol{u}(k)=\boldsymbol{K}_{\mathrm{p}}[\boldsymbol{e}(k)-\boldsymbol{e}(k-1)]+\boldsymbol{K}_{\mathrm{p}}\frac{T}{T_{\mathrm{i}}}\boldsymbol{e}(k)+\boldsymbol{K}_{\mathrm{p}}\frac{T_{\mathrm{d}}}{T}[\boldsymbol{e}(k)-2\boldsymbol{e}(k-1)+\boldsymbol{e}(k-2)]。\tag{7}$$

可以看出式(7)所示的增量型控制算法只需用到距今 3 个时刻的偏差值 $\boldsymbol{e}(k)$，$\boldsymbol{e}(k-1)$ 和 $\boldsymbol{e}(k-2)$，大大节省了计算量。且由于不需要做累加运算，也减小了计算误差及精度对控制量的影响。另外，由于式(7)输出的是控制量的增量，所以即便在某些特殊情况下限制了控制器输出或者控制器失效时输出为 0，也不会对系统安全产生较大的影响。

2.3.2　算法改进

在实际应用中，由于在每个采样时间间隔内控制器的输出不变，系统有可能出现暂时失控状态，所以上述数字 PID 控制器的实际控制效果并不是很理想，还需对其进行适当改进。

对于积分项，由于原积分系数为常数，所以控制过程中积分速率保持恒定，但我们希望系统的积分作用能在偏差较大时减弱，在偏差较小时加强，即要求积分速度具备自适应能力[8]。为实现这一目标可采用变速积分法，引入关于偏差 $\boldsymbol{e}(k)$ 的函数 $\boldsymbol{f}[\boldsymbol{e}(k)]$ 作为积分项中 $\boldsymbol{e}(k)$ 的系数，则原 PID 控制算法可以改写成变速积分 PID 的形式：

$$\boldsymbol{u}(k)=\boldsymbol{K}_{\mathrm{p}}\left\{\boldsymbol{e}(k)+\frac{T}{T_{\mathrm{i}}}\left[\sum_{j=0}^{k-1}\boldsymbol{e}(j)+\boldsymbol{f}[\boldsymbol{e}(k)]\boldsymbol{e}(k)\right]+T_{\mathrm{d}}\,\frac{\boldsymbol{e}(k)-\boldsymbol{e}(k-1)}{T}\right\}。\tag{8}$$

其中函数 $\boldsymbol{f}[\boldsymbol{e}(k)]$ 的具体形式可人为设定，其函数值与 $[|e_x(k)|,|e_y(k)|]^{\mathrm{T}}$ 的大小成反比。

另外由于式(6)中的微分项对具有高频扰动的生产过程响应过于灵敏，容易引起控制过程振荡，进而降低调节品质。为弥补这一不足，可以在输出端串联一个一阶惯性环节，使控制器变成一个不完全微分的 PID 控制器，则改进后的位置型 PID 控制算法如下：

$$\boldsymbol{u}(k)=\frac{\boldsymbol{T}_{\mathrm{f}}}{\boldsymbol{T}_{\mathrm{f}}+\boldsymbol{T}_{\mathrm{e}}}\boldsymbol{u}(k-1)+\frac{\boldsymbol{T}_{\mathrm{e}}}{\boldsymbol{T}_{\mathrm{f}}+\boldsymbol{T}_{\mathrm{e}}}K_{\mathrm{p}}\left\{\boldsymbol{e}(k)+\frac{T}{T_{\mathrm{i}}}\left[\sum_{j=0}^{k-1}\boldsymbol{e}(j)+\boldsymbol{f}[\boldsymbol{e}(k)]\boldsymbol{e}(k)\right]+T_{\mathrm{d}}\,\frac{\boldsymbol{e}(k)-\boldsymbol{e}(k-1)}{T}\right\}。\tag{9}$$

由式(9)可得改进后的增量型 PID 控制算法为：

$$\begin{aligned}\Delta\boldsymbol{u}(k)=&\boldsymbol{\alpha}\Delta\boldsymbol{u}(k-1)+\boldsymbol{\beta}\{[\boldsymbol{K}_{\mathrm{p}}+\boldsymbol{K}_{\mathrm{i}}\boldsymbol{f}[\boldsymbol{e}(k)]+\boldsymbol{K}_{\mathrm{d}}][\boldsymbol{e}(k)-\boldsymbol{e}(k-1)]\\&+(\boldsymbol{K}_{\mathrm{i}}-\boldsymbol{K}_{\mathrm{d}})[\boldsymbol{e}(k-1)-\boldsymbol{e}(k-2)]+\boldsymbol{K}_{\mathrm{i}}\boldsymbol{e}(k-2)\}\\=&\boldsymbol{\alpha}\Delta\boldsymbol{u}(k-1)+\boldsymbol{\beta}\{[\boldsymbol{K}_{\mathrm{p}}+\boldsymbol{K}_{\mathrm{i}}\boldsymbol{f}[\boldsymbol{e}(k)]+\boldsymbol{K}_{\mathrm{d}}]\boldsymbol{e}(k)\\&+\{-\boldsymbol{K}_{\mathrm{p}}+\boldsymbol{K}_{\mathrm{i}}\{1-\boldsymbol{f}[\boldsymbol{e}(k-1)]\}-2\boldsymbol{K}_{\mathrm{d}}\}\boldsymbol{e}(k-1)+\boldsymbol{K}_{\mathrm{d}}\boldsymbol{e}(k-2)\}。\end{aligned}\tag{10}$$

其中

$$\boldsymbol{\alpha}=\frac{\boldsymbol{T}_{\mathrm{f}}}{\boldsymbol{T}_{\mathrm{f}}+\boldsymbol{T}_{\mathrm{e}}},\boldsymbol{\beta}=\frac{\boldsymbol{T}_{\mathrm{e}}}{\boldsymbol{T}_{\mathrm{f}}+\boldsymbol{T}_{\mathrm{e}}},\Delta\boldsymbol{u}(k-1)=\boldsymbol{u}(k-1)-\boldsymbol{u}(k-2)，$$

$$\boldsymbol{K}_{\mathrm{p}}=\begin{bmatrix}K_{\mathrm{px}}&0\\0&K_{\mathrm{py}}\end{bmatrix},\boldsymbol{K}_{\mathrm{i}}=\boldsymbol{K}_{\mathrm{p}}\frac{T}{T_{\mathrm{i}}}=\begin{bmatrix}K_{\mathrm{ix}}&0\\0&K_{\mathrm{iy}}\end{bmatrix},\boldsymbol{K}_{\mathrm{d}}=\boldsymbol{K}_{\mathrm{p}}\frac{T_{\mathrm{d}}}{T}=\begin{bmatrix}K_{\mathrm{dx}}&0\\0&K_{\mathrm{dy}}\end{bmatrix}。$$

2.3.3　控制器参数整定

由式(10)可知，控制器需要整定的参数有 $\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{f}[\boldsymbol{e}(k)],\boldsymbol{K}_{\mathrm{p}},\boldsymbol{K}_{\mathrm{i}}$ 和 $\boldsymbol{K}_{\mathrm{d}}$。首先由 $T=1$ s，得 $\boldsymbol{T}_{\mathrm{e}}=\begin{bmatrix}T&0\\0&T\end{bmatrix}=\begin{bmatrix}1&0\\0&1\end{bmatrix}$，选取 $\boldsymbol{T}_{\mathrm{f}}=\begin{bmatrix}1&0\\0&9\end{bmatrix}$，可得 $\boldsymbol{\alpha}=\begin{bmatrix}0.5&0\\0&0.9\end{bmatrix}$，$\boldsymbol{\beta}=\begin{bmatrix}0.5&0\\0&0.1\end{bmatrix}$。其次，$\boldsymbol{f}[\boldsymbol{e}(k)]$ 可选取为关于 $\boldsymbol{e}(k)$ 的线性或者高阶函数：

$$\boldsymbol{f}[\boldsymbol{e}(k)]=\begin{cases}[1 \quad 1]^{\mathrm{T}},[\,|e_x(k)| \quad |e_y(k)|\,]^{\mathrm{T}}\leqslant \boldsymbol{W}\\ \dfrac{2\boldsymbol{W}-[\,|e_x(k)| \quad |e_y(k)|\,]^{\mathrm{T}}}{\boldsymbol{W}},\boldsymbol{W}<[\,|e_x(k)| \quad |e_y(k)|\,]^{\mathrm{T}}\leqslant 2\boldsymbol{W}\\ [0 \quad 0]^{\mathrm{T}},[\,|e_x(k)| \quad |e_y(k)|\,]^{\mathrm{T}}>2\boldsymbol{W}\end{cases}。\tag{11}$$

其中 $\boldsymbol{W}=[0.15,\ 0.1]^{\mathrm{T}}$。最后通过在 Simulink 仿真环境中使用凑试法反复测试，最终选取参数

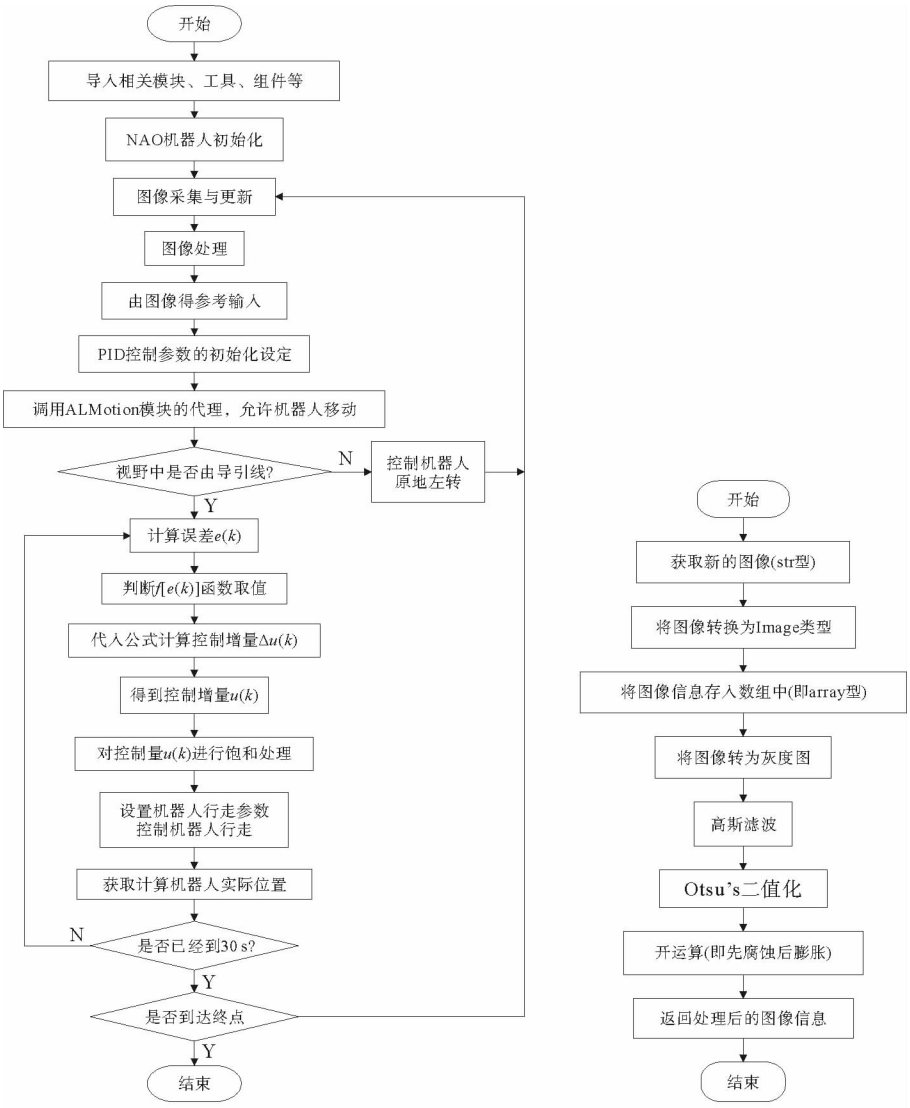
$$\boldsymbol{K}_{\mathrm{p}}=\begin{bmatrix}0.6 & 0\\ 0 & 0.47\end{bmatrix},\boldsymbol{K}_{\mathrm{i}}=\begin{bmatrix}0.018 & 0\\ 0 & 0.043\end{bmatrix},\boldsymbol{K}_{\mathrm{d}}=\begin{bmatrix}6 & 0\\ 0 & 1.88\end{bmatrix}。\tag{12}$$

为后续编程及仿真方便，记 $\boldsymbol{K}_{\mathrm{p}}+\boldsymbol{K}_{\mathrm{d}}=\boldsymbol{q}_0$，$-\boldsymbol{K}_{\mathrm{p}}+\boldsymbol{K}_{\mathrm{i}}-2\boldsymbol{K}_{\mathrm{d}}=\boldsymbol{q}_1$，$\boldsymbol{K}_{\mathrm{d}}=\boldsymbol{q}_2$，则改进后算法可整理成：

$$\Delta\boldsymbol{u}(k)=\boldsymbol{\alpha}\Delta\boldsymbol{u}(k-1)+\boldsymbol{\beta}\{\{\boldsymbol{q}_0+\boldsymbol{K}_{\mathrm{i}}\boldsymbol{f}[\boldsymbol{e}(k)]\}\boldsymbol{e}(k)+\{\boldsymbol{q}_1-\boldsymbol{K}_{\mathrm{i}}\boldsymbol{f}[\boldsymbol{e}(k-1)]\}\boldsymbol{e}(k-1)+\boldsymbol{q}_2\boldsymbol{e}(k-2)\}。\tag{13}$$

2.4 程序编写

本设计所用仿真程序使用 Python 语言编写。作为一种不受局限、跨平台的开源编程语言，Python 语言相比其他语言来说更简单明了，且更容易移植，由于其可以把其他语言编程的模块结合在一起，所以又被人称作胶水语言。本设计整体程序流程如图5(a)所示。其中的图像处理部分用到了代码完全开源的跨平台



(a) 程序整体流程图　　(b) 图像处理流程图

图 5　流程图

Fig. 5　Flow chart

计算机视觉库 OpenCV，该库几乎涵盖了目前所有较成熟的图像处理和计算机视觉方面的通用算法[10]。本设计中图像处理部分的具体流程图如图 5(b)所示。

3　仿真验证

3.1　Simulink 仿真验证

根据图 4 所示系统结构及式(13)可搭建 Simulink 仿真结构框图，如图 6 所示。

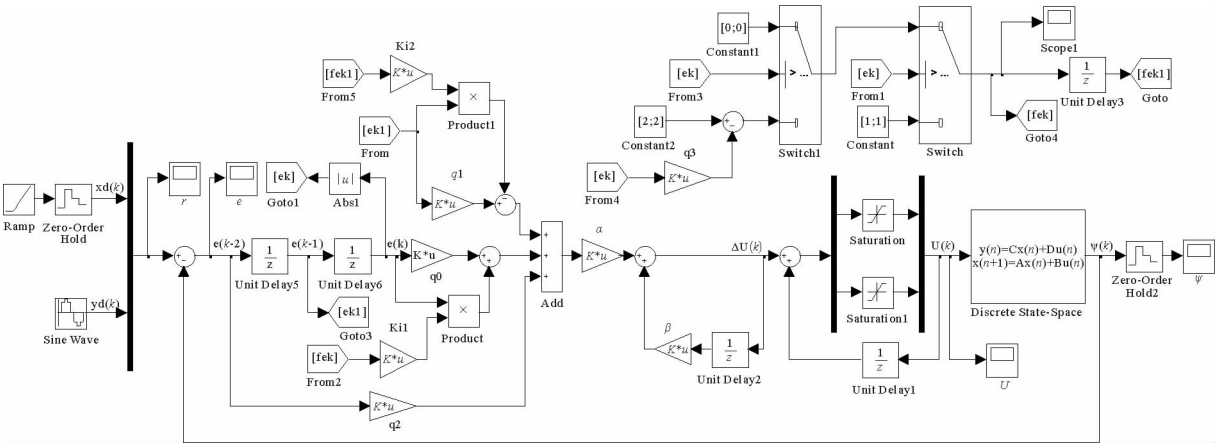


图 6　Simulink 仿真结构框图

Fig. 6　Simulation structure diagram by Simulink

代入整定好的各项参数，为模拟真实的导引线信息，取斜率为 0.03 的斜坡信号作为 $x_d(k)$，幅值为 0.05 频率为 0.15 的正弦信号为 $y_d(k)$，设置仿真时间为 30 s，运行后得到仿真结果如下：

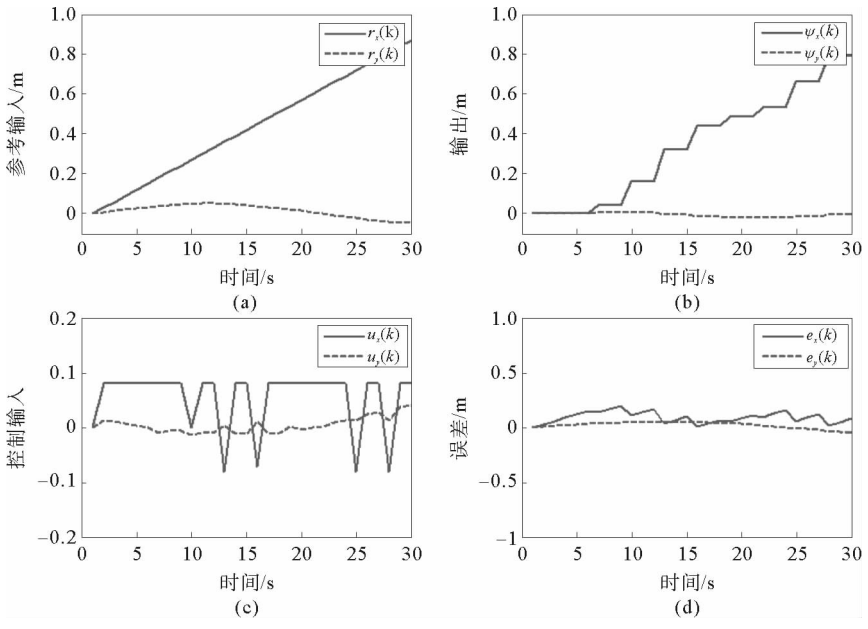


图 7　Simulink 仿真结果波形图

Fig. 7　Waveform of simulation results by Simulink

从图 7 可以看出，在 30 s 的仿真时间内，输出一直跟随给定的系统参考输入的变化而变化，系统误差总维持在 0.2 m 以内。也就是说，该 PID 控制器可以实现控制 NAO 机器人循线行走的目的。

3.2 Webots 仿真验证

Webots 是由 Cyberbotics 公司出品的一款用于移动机器人建模、编程和仿真的开发环境软件，软件内建的 3D 编辑器能构建出 3D 机器人模型，支持包括 C，C＋＋，JAVA，Python 和 Matlab 在内的多种编程语言，通过编辑程序可模拟机器人的动作。在 Webots 仿真软件中建立 9 m×8 m 的空白场地，加入形状如图 8 所示的黑色导引线，图中星形代表 NAO 机器人初始位置。

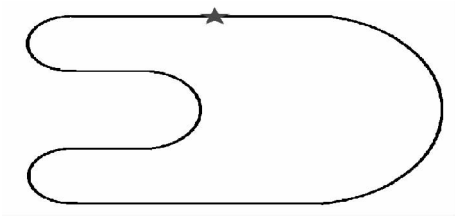
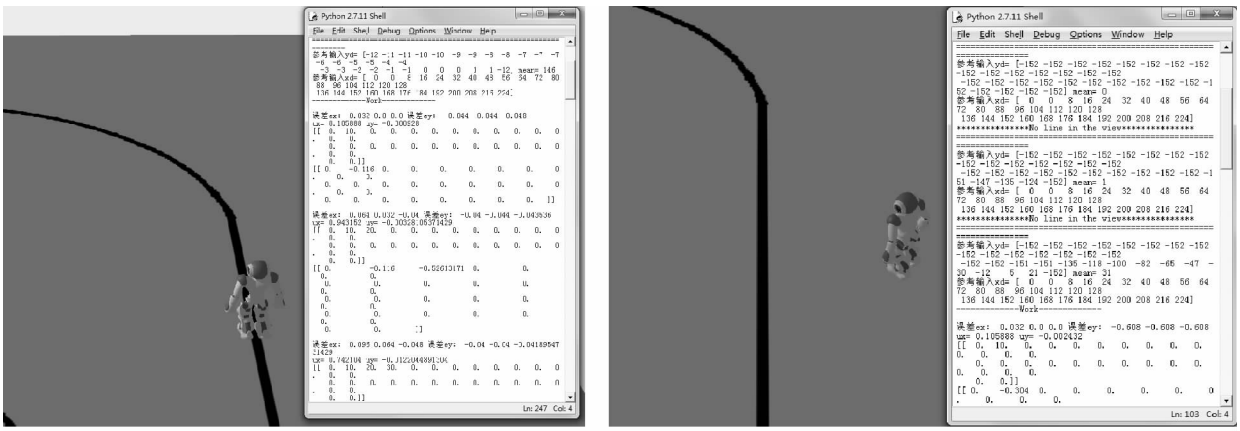


图 8 Webots 仿真实验中的导引线

Fig. 8 Lead wire in Webots simulation experiment

进行仿真试验时为了方便程序的观察与调试，在原程序中适当的增加了一些显示函数，用以即时显示 NAO 机器人运行状态，正常行走状态下的仿真实验结果如图 9(a)所示，而对于视野中没有导引线时的仿真实验结果如图 9(b)所示。



(a) NAO正常行走状态　　(b) NAO视野中没有导引线时

图 9 NAO 机器人的 Webots 仿真实验结果

Fig. 9 Simulation results of NAO robot by Webots

4 结论

对 NAO 机器人循线行走的控制问题进行了研究，首先将问题归结成信号跟踪问题并建立线性定常离散系统模型和适当的坐标系，对机器人每 30 s 获取到的图像进行一系列处理得到参考输入信号。采用改进后的数字增量型 PID 算法设计控制器，选取机器人实际位置与参考输入的差作为控制器的输入，并使用凑试法在 Simulink 仿真平台下完成参数整定。最后用 Python 语言编写程序，Webots 软件仿真。结果表明，对任意给定的导引线，NAO 机器人都能在全程无人干预的情况下自主寻找并沿线行走。即在 NAO 事先未知导引线信息的情况下，通过图像的实时获取和 PID 控制实现循线行走的功能。

参考文献：

[1]张生. NAO 机器人的目标识别与定位研究[D]. 合肥：安徽大学，2013.

[2]赵坤. 变电站智能巡检机器人视觉导航方法研究[D]. 保定：华北电力大学，2014.

[3]孟宪龙. RoboCup 中红球识别追踪及自定位研究[D]. 合肥：安徽大学，2014.

[4]柏雪峰，杨斌. 基于 NAO 机器人目标识别与定位算法[J]. 成都信息工程学院学报，2014，29(6)：625-629.

BAI Xuefeng，YANG Bin. Target recognition and localization algorithm based on NAO robot[J]. Journal of Chengdu Univer-

sity of Information Technology,2014,29(6):625-629.
[5]宗鹏程.基于NAO机器人的视觉目标检测与跟踪[D].保定:华北电力大学,2015.
[6]孙翔侃,白宝兴.基于机器学习的NAO机器人检测跟踪[J].长春理工大学学报(自然科学版),2016,39(2):116-119.
SUN Xiangkan,BAI Baoxing. Detection and tracking of NAO based on machine learning[J]. Journal of Changchun University of Science and Technology(Natural Science Edition),2016,39(2):116-119.
[7]NOBUYUKI O. A threshold selection method from gray-level histograms[J]. IEEE Transaction on System,Man,and Cybernetics,1979,9(1):62-66.
[8]徐文尚.计算机控制系统[M].2版.北京:北京大学出版社,2014:113-132.
[9]BILL L. Python语言及其应用[M].北京:人民邮电出版社,2016.
[10]王艳红.基于OpenCV的运动目标检测与跟踪算法的研究[D].杭州:杭州电子科技大学,2013.
[11]BECCARI G,CASELLI S,ZANICHELLI F,et al. Vision-based line tracking and navigation in structured environments[C]//IEEE International Symposium on Computational Intelligence in Robotics and Automation. IEEE Computer Society,1997:406-411.

(责任编辑:吕海亮)

(上接第79页)

[17]闻新,李新,张兴旺,等.应用MATLAB实现神经网络[M].北京:国防工业出版社,2015:95-118.
[18]YU S,ZHAO D,CHEN W,et al. Oil-immersed power transformer internal fault diagnosis research based on probabilistic neural network[J]. Procedia Computer Science,2016,83:1327-1331.
[19]张龙,陈宸,韩宁,等.压缩感知理论中的建筑电气系统故障诊断[J].智能系统学报,2014,9(2):204-209.
ZHANG Long,CHEN Cheng,HAN Ning,et al. Fault diagnosis of electrical system in buildings based on compressed sensing[J]. CAAI Transaction on Intelligent System,2014,9(2):204-209.
[20]王鑫,于洪亮,段树林,等.贝叶斯与遗传神经网络相融合的柴油机故障诊断研究[J].船舶工程.2012,34(1):32-35.
WANG Xin,YU Hongliang,DUAN Shulin,et al. Study on diesel engine fault diagnosis based on integration of Bayesian and genetic netword method[J]. Ship Engineering,2012,34(1):32-35.

(责任编辑:高丽华)