Artificial Intelligence and Robotics Research 人工智能与机器人研究, 2019, 8(3), 166-182
Published Online August 2019 in Hans. http://www.hanspub.org/journal/airr
https://doi.org/10.12677/airr.2019.83019

Hans汉斯

Research on Construction Method of Robot Motion Scene Based on Virtual Reality Interaction

Xuemei Liu, Yu Zhou, Langqin Meng, Chaoxing Zhang, Zhongyue Liu

Southwest Petroleum University, Chengdu Sichuan
Email: 406253978@qq.com

Received: Aug. 3rd, 2019; accepted: Aug. 21st, 2019; published: Aug. 28th, 2019

Abstract

In order to cope with the complex and ever-changing environment, this paper uses a combination of virtual and real interaction technology and robot control technology to design a virtual motion scene of the robot that matches the real scene, and realizes the motion control of the robot in the virtual scene and the real scene. That is, virtual and real interactive motion control system. The virtual motion scene of the robot is built by Unity 3D software, and various obstacles are set in the virtual scene. The Navgation navigation network module is used to calculate the optimal path from the starting point to the end point under the condition of avoiding obstacles, and the NavMesh plug-in is utilized to complete the robot's addressing obstacle avoidance motion control. The realistic scene adopts a two-legged six-degree-of-freedom robot. The robot will follow the robot synchronous motion in the virtual scene, and use the OpenMV3 camera to identify the image information of the obstacle in the real scene. After the image visual processing, the obstacle information is fed back to the Socket server. The scene extracts information from the server and reproduces the corresponding proportion in the virtual scene. At the same time, the virtual robot performs motion control to avoid obstacles, thereby achieving the purpose of virtual and real interactive motion control.

Keywords

Virtual and Real Interaction, Unity 3D, Motion Control, Socket Server, Obstacle Avoidance, DTW Algorithm, OpenMV

基于虚实交互的机器人运动场景构建方法的研究

刘雪梅，周　雨，孟茛钦，张朝兴，刘忠跃

西南石油大学，四川　成都

文章引用: 刘雪梅, 周雨, 孟茛钦, 张朝兴, 刘忠跃. 基于虚实交互的机器人运动场景构建方法的研究[J]. 人工智能与机器人研究, 2019, 8(3): 166-182. DOI: 10.12677/airr.2019.83019

Email: 406253978@qq.com

收稿日期：2019年8月3日；录用日期：2019年8月21日；发布日期：2019年8月28日

摘　要

为应对复杂多变的环境，本文采用虚实交互技术与机器人控制技术相结合的方法，设计了一种与真实场景相匹配的机器人虚拟运动场景，实现在虚拟场景与现实场景中对机器人的运动控制，即虚实交互运动控制系统。通过Unity 3D软件搭建机器人虚拟运动场景，并在虚拟场景中设置各类障碍物，用Navgation导航网络模块计算出在避开障碍物的条件下，从起点到终点的最优路径，并利用NavMesh插件完成机器人的寻址避障运动控制。现实场景采用双足六自由度机器人，机器人将跟随虚拟场景中的机器人同步运动，并用OpenMV3摄像头识别现实场景中障碍物的图像信息，通过图像视觉处理后将障碍物的信息反馈于Socket服务器，虚拟场景从服务器中提取信息，并在虚拟场景中对应比例的重现，同时虚拟机器人进行躲避障碍物的运动控制，以此来达到虚实交互运动控制的目的。

关键词

虚实交互，Unity 3D，运动控制，Socket服务器，避障，DTW算法，OpenMV

1. 引言

机器人技术在国外的自动化行业迅猛的增涨[1]，在研发中人们开始利用各类传感器让机器人具备环境感知能力，近年来机器人的研究趋势也逐渐成为热点并且更加紧密地与虚拟技术相结合，例如麦肯锡的具备视觉传感器系统视觉机器人控制系统与博尼的安装有压力传感器的“灵巧手”等。在国外[2]，日本致力于建立大规模 VR 知识库的研究，并应用于机器人之上。东京大学基于建立虚拟现实技术对物体三维结构的判定和三维图形的表示、静态图像的提取来辅助研究机器人。在国内，清华大学对虚拟现实和临场感方面进行了研究，在球面屏幕显示和图像随动、克服立体图闪烁的措施和深度感等方面都具有不少独特的方法。

本文研究的是基于虚实交互的机器人运动场景控制系统，通过虚与实的结合让机器人能应对各种复杂环境，在虚拟方面，利用 Unity3D 建立机器人模型和运动场景，且虚拟机器人能躲过障碍物走到终点，在实际方面，使用 OPNEMV 摄像头采集现实场景中障碍物的位置和大小信息，将该信息发送给上位机，并在虚拟场景中实现对应比例的实时重现。除此之外，通过蓝牙通信模块将虚拟机器人运动控制指令发送给下位机(机器人)单片机，让机器人完成相应的控制，最终完成虚实交互运动控制的任务。设计分为五个模块：现实障碍物图像数据采集模块(Python 编写)、虚拟场景搭建模块(Unity 3D 软件)、Socket 服务器(C#编写)、蓝牙通信模块(C++编写)、机器人控制模块(C++编写)。

2. 机器人正运动学分析

本文采用双足六自由度机器人[3]，如下图 1 所示，机器人共有六个舵机作为关节。其中有 2 个 270

度的舵机作为胯部活动大腿部分，另外 4 个 180 度的舵机分别用于膝盖和脚踝关节。六个舵机相互协调工作模拟人体动作仿生行走。

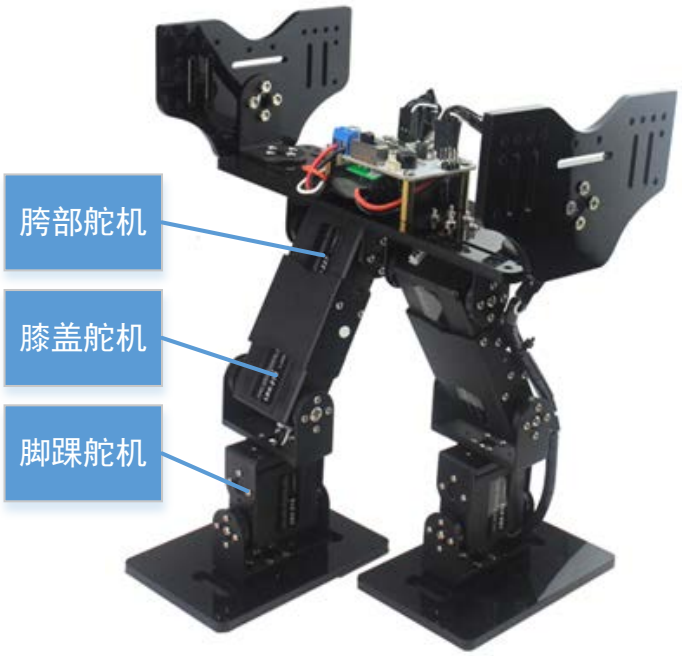

Figure 1. The physical appearance of a bipedal six-degree-of-freedom robot
图 1. 双足六自由度机器人实物外观图

对于机器人的正动力学问题是 “已知力求运动”，当给出了机器人各关节的驱动力、关节初始角度及初始角速度，求解各关节的角度、角速度和角加速度。本文通过采用拉格朗日方法求解正向动力学方程。拉格朗日方法具有完整约束的质点系动力学问题的普遍方法。它根据全部杆件的动能和势能求出拉格朗日函数，再带入拉格朗日方程中，从而导出机械系统的运动方程。此方法的主要特征是可以不考虑杆件相互的内部约束力[4]。具体步骤为：

设 q 表示步行系统的广义坐标，$\dot{q}$ 表示步行系统的广义速度，拉格朗日函数 $L(q,\dot{q})$ 定义为系统的总动能 K 与总势能 V 之差，即

$$L(q,\dot{q}) = K(q,\dot{q}) - V(q)$$

根据动力学普遍方程，系统的动力学方程可以用拉格朗日函数表示成

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i, i = 1,2,\cdots,N$$

式中，τ_i 为作用在第 i 个关节上的非保守广义力，它可以分解成：

$$\tau_i = u_i + E_i'(q)F_{ext}$$

其中，u_i 为作用在第 i 个关节上的驱动力矩；F_{ext} 为除关节驱动力矩之外的其他非保守力；$E_i(q)$ 为外力 F_{ext} 作用点的笛卡尔坐标对广义坐标的雅可比矩阵。

由于动力学方程的解析形式通常十分复杂，特别是是正向动力学拉格朗日方程中各矩阵的解析形式。为此采用MATLAB中的符号计算工具Symbolic来自动计算和生成拉格朗日方程中各矩阵项的解析表达式。

由本实验的双足六自由度的机器人，取 $q=(q_1,q_2,q_3)'$ 为步行系统的广义坐标，对于机器人身体上的每一个杆件的质心位置在绝对坐标系下的笛卡尔坐标为 $\left(p_i^x,p_i^y\right)$，并将其表示成广义的 q 的函数：

$$\begin{bmatrix} p_i^x \\ p_i^y \end{bmatrix}=\begin{bmatrix} p_i^x(q) \\ p_i^y(q) \end{bmatrix}$$

根据函数求导规则，杆件 i 的质心线速度和绝对线速度可表示成

$$\begin{bmatrix} \dot{p}_i^x \\ \dot{p}_i^y \end{bmatrix}=\begin{bmatrix} \dfrac{\partial p_i^x(q)}{\partial q} \\ \dfrac{\partial p_i^y(q)}{\partial q} \end{bmatrix}\dot{q}$$

则杆件 i 的动能为

$$K_i=\frac{1}{2}m_i\left(\left(p_i^x\right)^2+\left(p_i^y\right)^2\right)+\frac{1}{2}I_i\left(\dot{q}_i\right)^2=:\frac{1}{2}\dot{q}^{\mathrm{T}}D_i(q)\dot{q}$$

杆件 i 的势能为

$$V=m_igp_i^x=m_igp_i^y$$

系统总动能为

$$K=\sum_{i=1}^{3}K_i=\frac{1}{2}\dot{q}^{\mathrm{T}}D(q)\dot{q}$$

系统总势能为

$$V=\sum_{i=1}^{3}V_i$$

拉格朗日函数为

$$L=K-V$$

将拉格朗日函数代入动力学普遍方程就可以推导出拉格朗日方程。通过 MATLAB 仿真可以得到本文机器人的正运动动力学模型的拉格朗日方程解析方程矩阵：

$$\begin{pmatrix} 0 & \dfrac{M_1*(dq_1+3*dq_2+2*dq_3)}{2} & \dfrac{M_2*l_1*K_2*(dq_1+2*dq_2+3*dq_3)}{2} \\ -\dfrac{M_1*(3*dq_1+dq_2+2*dq_3)}{2} & 0 & \dfrac{M_2*l_2*K_3*(2*dq_1+dq_2+3*dq_3)}{2} \\ -\dfrac{M_2*l_1*K_2*(3*dq_1+2*dq_2+dq_3)}{2} & -\dfrac{(M_2*l_2*K_3*(2*dq_1+3*dq_2+dq_3)}{2} & 0 \end{pmatrix}$$

$K_1=\sin(q_1-q_2)$，$K_2=\sin(q_1-q_3)$，$K_3=\sin(q_2-q_3)$，$K_4=(a_2*m_2+l_2*m_3)$，$M_1=l_1*K_1*K_4$，$M_2=a_3*m_3$。

3. 虚拟场景构建与智能避障

采用基于 Unity3D 的物理引擎构建刚体组件[5]，碰撞系统等进而构建出虚拟场景，通过使用其中的 navigation 地图导航网格将机器人的行走表面勾选出来，即勾勒出静态避障的空间区域。在虚实交互区域，摄像头捕捉采集，当识别到物块时，会将识别到的物块像素发送到上位机，进而使上位机在虚拟场景中生成与场景成比例的虚拟物块，此时通过动态避障的脚本处理后机器人将会绕开物块进行避障。

3.1. 虚拟场景的构建

3.1.1. 刚体构建

为对象添加 Rigidbody 刚体组件[6]，可实现该对象在场景当中的物理交互。此时对象便可以接收外力与扭矩力，并且任何对象只有在添加 Rigidbody 组件之后才能受到重力的影响，让物体具有在现实世界的一般物体特性。Rigidbody 钢体组件如下图 2 所示：

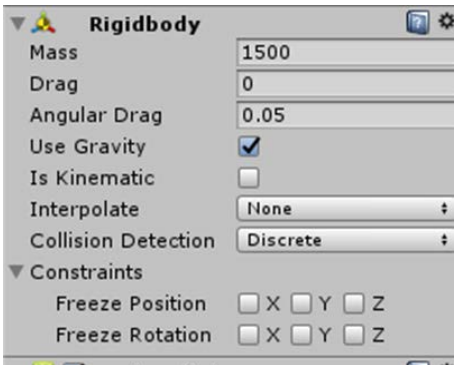

Figure 2. Rigidbody steel body assembly
图 2. Rigidbody 钢体组件

3.1.2. 碰撞体构建

碰撞体是物理组件中的一类，每个物理组件都有独立的碰撞体组件，它要与刚体一起添加到对象上才能触发碰撞。如果两个刚体相互撞在一起，只有两个对象是碰撞体时物理引擎才计算碰撞，在物理模拟中，没有碰撞体的刚体会彼此互相穿过。本文采用的是简单方便及耗内存少的 Box Collider 碰撞器，选用下图 3 所示的 Is Trigger 触发器选项，即可产生碰撞效果。Box Collider 碰撞器如下图 3 所示：

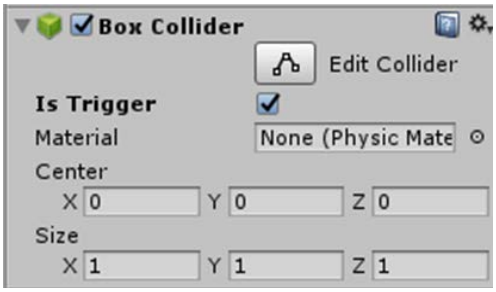

Figure 3. Box Collider Collider
图 3. Box Collider 碰撞器

3.1.3. 虚拟场景构建

在 Unity 3D 软件中通过 GameObject 窗口生成不同形状的 3D 物体，再对每个物体进行材质渲染，经过搭建便能得下虚拟运动场景如下图 4。

3.2. 虚拟场景动画添加

使用 Nav Mesh Agent (避障寻址插件)，让机器人在虚拟场景中进行位置移动，并设置好动画之间的转换时间、播放长度和动画播放速度，让机器人在运动控制场景按照一定规则动起来。虚拟场景中机器人模型如下图 5 所示：

Figure 4. Virtual motion scene graph
图 4. 虚拟运动场景图

Figure 5. Robot model built by virtual scene
图 5. 虚拟场景搭建的机器人模型

3.3. 虚拟场景避障

在虚拟场景中，机器人需要从设定的起点走到终点，并在这个过程中完成智能避障。我们使用 Unity 3D 中的 Navigation 地图导航网格将整个场景中行走表面勾选出来，再将障碍物选中，设置机器人参数(长宽高、可攀爬最大斜坡角度、可跨越最高台阶及跳跃的最大距离)，导航网格图如下图 6 所示。

在静态避障的基础下，机器人还需完成动态避障。现实场景中通过 OpenMV 摄像头识别障碍物的图像信息，并将物体的信息发送给虚拟场景，并在虚拟场景中实时的生成相应物块，机器人完成动态避障动作。动态避障流程如下图 7。

3.4. 虚实地图比例设计

本设计中的虚拟场景和现实场景为两个不同的场景，要达到虚实同步动作的要求，需要一个同步的标准，即现实场景的地图与虚拟场景中的地图所对应的比例，若是没有相应的比例则无法确定现实场景中的机器人所在的位置在虚拟场景中对应的关系。由于现实机器人是由舵机控制前进，其速度很难定量控制，故我们不采用直接设定地图比例的方法，而是通过在现实中与虚实场景对应进行多次匀速测试得到。测试方法是多次让现实场景与虚拟场景同步运动，通过多次测量并剔除偏差过大的某次测量值之后

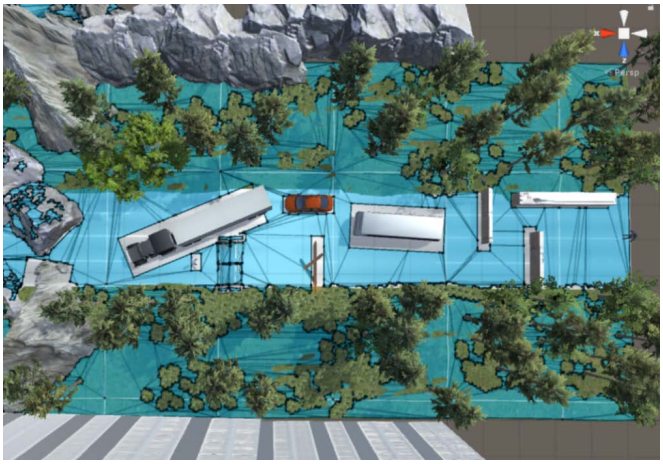

Figure 6. Navigation grid diagram
图 6. 导航网格图

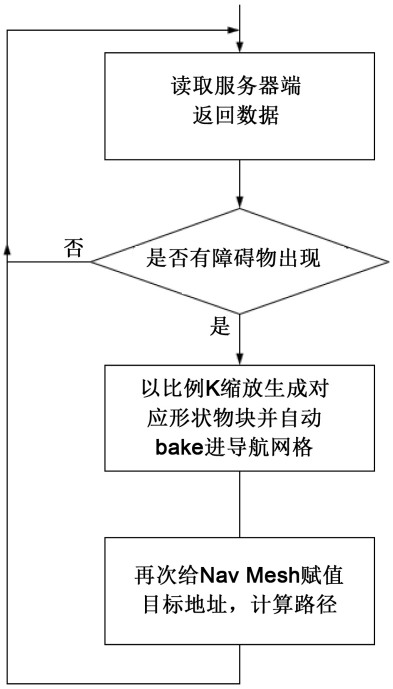

Figure 7. Dynamic obstacle avoidance flow chart
图 7. 动态避障流程图

再求取平均值，最后求取两个地图的比值 G。得到数据如下表 1：

Table 1. Virtual and real map scale measurement data table
表 1. 虚实地图比例测量数据表

次数	1	2	3	4	5	6	7	8	9	10
现实场景/m	3.154	3.212	3.611	3.491	3.773	3.125	3.649	3.412	3.225	3.621
虚拟场景/m	20	20	20	20	20	20	20	20	20	20

得出虚拟地图与现实地图之间的比例值 G 为：

$$G = \frac{\text{虚拟地图长度}}{\text{现实地图长度}} = 5.84$$

4. 虚实场景的交互

本文使用的是 OpenMV 摄像头，通过该摄像头来模拟真实情况下，机器人遇到突发状况后的运动特性，例如在机器人行走过程中，在它前方放置一块障碍物，摄像头识别到物体后能将物体的大小坐标信息传到 Unity3D 中，Unity3D 根据物体信息重现障碍物。

该模块带有 ARM Cortex M7 处理器，400 MHz，1 MB RAM 以及 2 MB flash。摄像头焦距为 2.8 mm。OpenMV 摄像头如下图 8 所示：

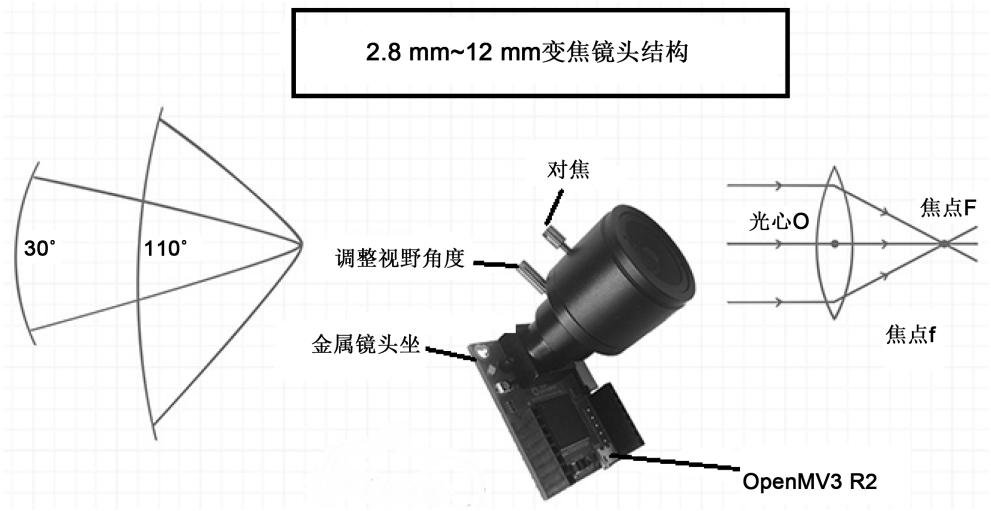

Figure 8. OpenMV camera
图 8. OpenMV 摄像头

在摄像头中，最重要的元件是感光元件，感光元件将光信息转换为电信号，再送到计算机中处理，OpenMV 使用的是 OV7725 感光元件，它能处理 640 × 480 8 位灰度图或者 640 × 480 16 位 RGB565 彩色图像。对于一张图片，有许多参数如分辨率，像素，色彩模式等[7]，其中色彩模式主要分为 RGB、CMYK、HSB、LAB，在本项目中对于色块的分辨运用的是 LAB 模式。通过调节 LAB 值来让摄像头找到我们所放置的障碍物。

4.1. 图像获取

我们使用 OpenMV 摄像头来识别红色障碍物体[8]，打开摄像头采集数据，截取其中一张图片进行阈值调节，我们需要的是红色，其他颜色需要排除在外，将红色区域调节成白色，其他区域调节成黑色，阈值设定如下图 9 所示：

调节完成后记录 LAB 值，把 LAB 值赋值到 red_threshold 数组中备用，通过

```
img=sensor.snapshot()
blobs=img.find_blobs{[red_threshold]}
```

这两行代码，摄像头就能准确获取红色的物块，随后通过

```
img.draw_rectangle(max_blob.rect())
```

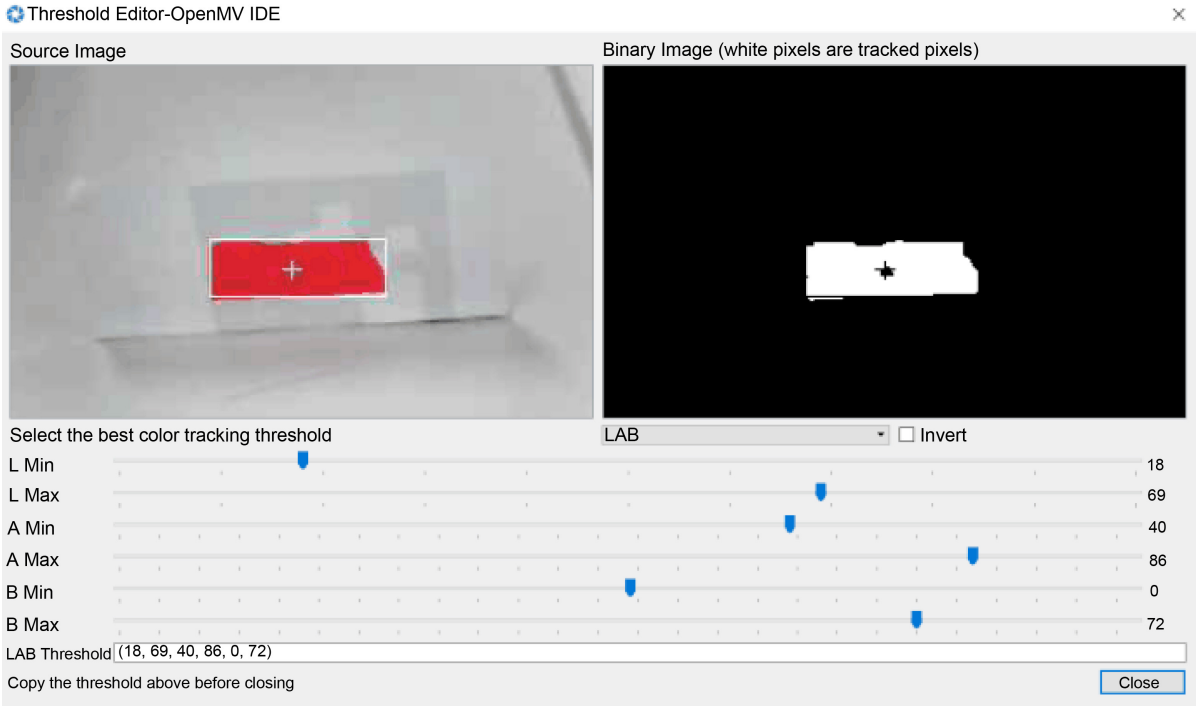

Figure 9. Threshold setting diagram
图 9. 阈值设定图

将红色物块框出来，方便计算它的长度和宽度。

摄像头获取到红色物块的信息储存在 blob 中，我们需要的是边框的 w,h 以及中心坐标，分别对应的是

blob.w　blob.h　blob.cx　blob.cy

最后将获取到的数据打包通过串口发送，再以比例 G 在 Unity3D 中重现障碍物。

4.2. 物体与摄像头距离测定

使用 OpenMV 摄像头进行物体距离测定主要有两种方法，其一是利用 Apriltag 对物体定位，Apriltag 是一个视觉基准系统，可用于各种任务包括 AR，机器人和相机校准，Apriltag 可以直接在 OpenMV IDE 中生成，通过 Apriltag 检测程序可以计算到物体相对于摄像头的 3D 位置，方向和 ID，Apriltag 如图 10 所示。

Figure 10. Apriltag code
图 10. Apriltag 码

Apriltag 主要分为以下几种：Tag16H5，Tag25H7，Tag25H9，Tag36H10Tag36h11 以及 ARTOOL-KIT，每一种称为一个家族，每个家族有各自的成员数量，如 Tag16H5 中有 0~29 号共 30 个标记，Tag25H7 中有 0~241 号共 242 个标记，不同家族之间除成员数量不同以外，每个家族的有效区域也有差别，如 Tag16H5

的有效区域是 4 × 4 个方块，Tag36H11 的有效区域为 6 × 6 个方块，有效区域大，校验的信息多，所以 Tag36H11 的出错率要比 Tag16H5 小。

Apriltag 具有 3D 定位功能，除了能测量到物体相对于摄像头的 X,Y,Z 坐标外，还能知道物体在每个轴上的旋转量。如图 11 所示， X_1,Y_1,Z_1 为空间的三个位置量， α,β,γ 为三个旋转量。

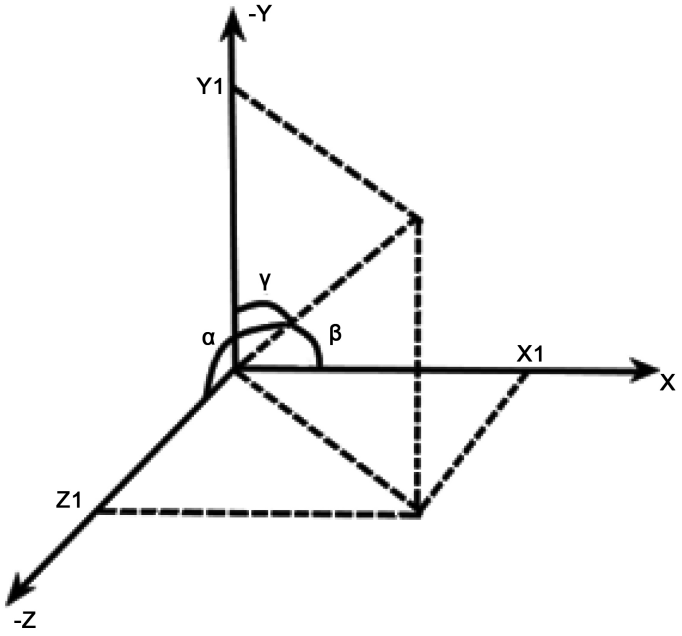

Figure 11. 3D data from Apriltag
图 11. 检测 Apriltag 得到的 3D 数据

但用这种方法需要在每个物体上贴 Apriltag，使用起来比较麻烦，所以我们采用第二种方法来测量距离，其原理如下图 12。

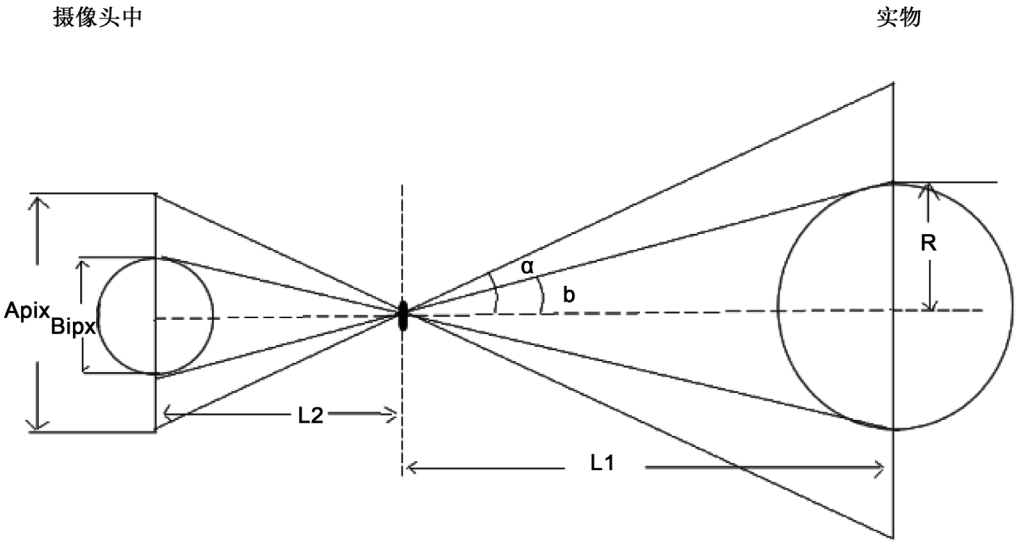

Figure 12. Camera measurement object distance schematic
图 12. 摄像头测量物体距离原理图

Apix 为摄像头画面固定像素，*Bpix* 为物体在摄像头画面中所占像素，*R* 为物体半径(以球为例)，α 为摄像头视角的一半。左边摄像头的几何关系为：

$$\tan(\alpha)=\frac{Apix}{2L_2}$$

$$\tan(b)=\frac{Bpix}{2L_2}$$

所以有：$\frac{\tan(\alpha)}{\tan(b)}=\frac{Apix}{Bpix}$

由右边实物环境中几何关系可得：$\tan(b)=\frac{R}{L_1}$

综合上述式子可得：

$$L_1 * Bpix=\frac{R * Apix}{\tan(\alpha)}$$

所以 $L_1=\frac{K}{Bpix}$，即，距离 $=\frac{\text{常数}}{\text{直径的像素}}$。

在知道这个常数之后，就可以测出物体与摄像头的距离，首先用一个参照物，放在离摄像头 *A* cm 处，打印出物体在摄像头中直径的像素，再将两个数据相乘，即可得到 *K* 值。

4.3. 动态时间归整算法

为了衡量虚拟主角运动轨迹和实体机器人轨迹的相似度，本文使用了动态时间归整算法，即(Dynamic Time Warping, DTW)，该算法最初运用在语音识别领域，解决说话语速不一致的问题。DTW 算法主要是基于动态规划的思想，通过动态规整来对齐时间轴，使每个数据点有最好的对齐方式，进而消除不齐造成的误差，获取时间序列间最好的匹配方式[9]。两条时间序列各个数据点对齐如下图 13 所示，在经过时间轴拉伸之后，得到如下图 14 所示，最终发现两条相似的时间序列。

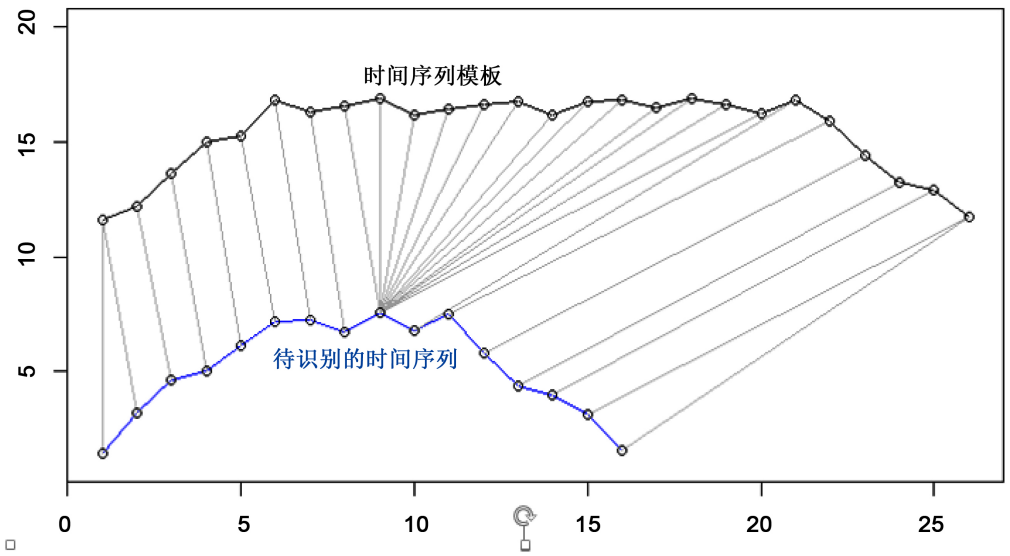

Figure 13. Data point alignment of time series
图 13. 时间序列的数据点对齐

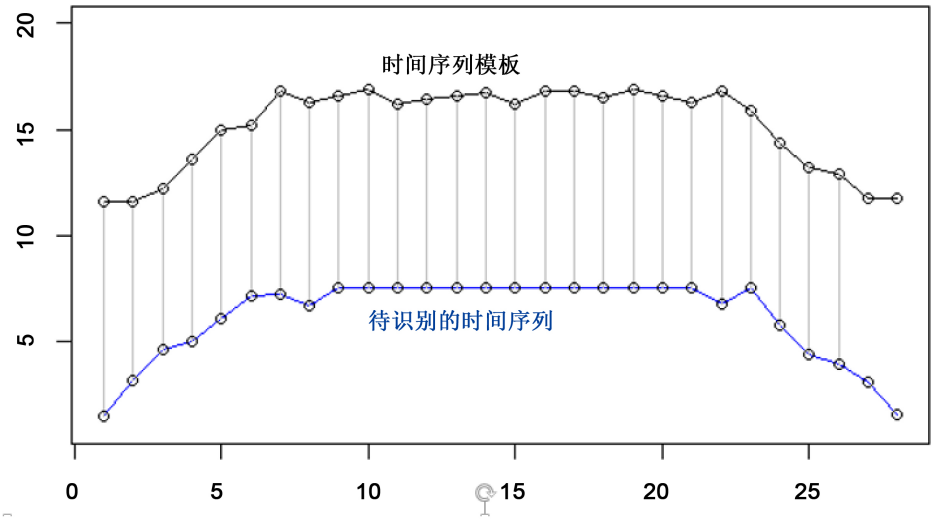

Figure 14. Time series after dynamic rounding
图 14. 经过动态归整后的时间序列

计算参考模板与测试模板之间的直接距离可以得出它们之间的相似度。两个模板之间的距离越短，模板之间的相似度就越高。在使用 DTW 算法进行相似判断前，需要先使这两组时间序列在时间轴上对齐。

首先，构造一个 $M*N$ 的矩阵 D，矩阵元素 $d(i,j)$ 表示 $\overrightarrow{R(i)}$ 和 $\overrightarrow{T(j)}$ 两点之间的距离(也是参考模板中的每条时间序列和测试模板每条时间序列之间相似度)，其中 $d(i,j)=\left(\overrightarrow{R(i)}-\overrightarrow{T(j)}\right)^2$，根据 DTW 的边界条件，最短距离也就是 $d(0,0)$ 到 $d(m,n)$ 之间的最短路径如图 15。再结合 DTW 算法的单调性和连续性要求，设定局部路径约束条件为：当从一个方格 $d(i-1,j-1)$、$d(i-1,j)$ 或者 $d(i,j-1)$ 中到下一个方格 $d(i,j)$，横向或竖向的距离为 $d(i,j)$，斜着对角线过来的则是 $2d(i,j)$。下面的数学(1)可以解释这种约束条件，其中 $g(i,j)$ 表示 2 个模板都从 $g(0,0))$点到 $g(m,n)$ 逐次匹配时两个模板之间的距离，并且要在上一次匹配的基础上加 $d(i,j)$ 或者 $2d(i,j)$，然后取最小值。

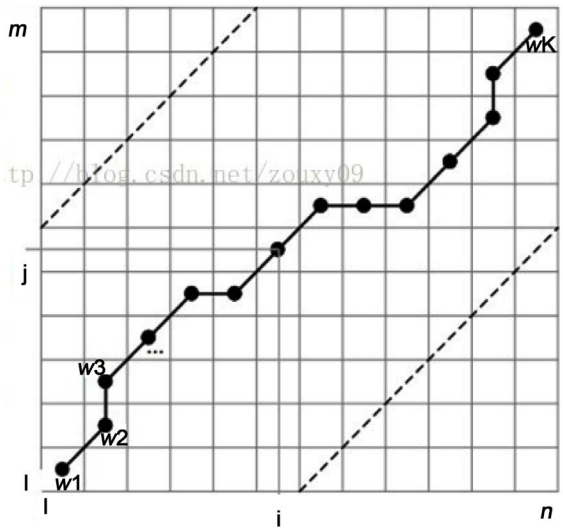

Figure 15. Finding the shortest path between two time series
图 15. 找到两个时间序列间的最短路径

$$g(ck)=g(i_k,j_k)=g(i,j)=\min\begin{cases}g(i-1,j)+d(i,j)\\ g(i,j-1)+2d(i,j)\\ g(i,j-1)+d(i,j)\end{cases} \tag{1}$$

5. 通信桥梁——Socket 服务器

5.1. Socket 服务器与上位机通信

Socket 服务器[10]作为联系上位机和下位机、虚拟场景和现实的关键通信媒介。Unity3D 虚拟场景客户端与 Socket 服务器通过在 Visual Studio 上配置需要的引用文件和对应命名的空间。其具体步骤为首先应在 VS 中配置 Socket 服务器的 I/O 地址和端 SS 口号，用于客户端访问时访问的唯一标识。当客户端初始化 Socket 并和 Socket 服务器连接成功后，服务器就会监听配置地址端口上的服务信息，当配置地址有数据传输过来，便会立刻将数据保存下来。此时 Socket 服务器既可以向客户端发送数据，客户端接收到数据后又可以将数据缓存到对应的寄存器中以便下次使用。在本实验中，Unity3D Socket 服务器中使用运动刚体函数 Velocity，获取物体运动中的速度向量等信息从而读取虚拟界面机器人的运动状态。

5.2. Socket 与下位机通信

串口是连接 Socket 服务器和蓝牙模块的通信媒介[11]，通过 Socket 服务器定义的端口号接入一个 USB 转 TTL 的连接器，并将两个转接器的 TX/RX 反接，即 RX 接 TX，TX 接 RX。其中一个串口通过串口调试助手用于对服务器的数据传输实施实时监听，而另一个串口为 Socket 服务器的数据发送接收端，将客户端发出的运动角度信息发送出来，发送到蓝牙模块。

6. 实验

6.1. 摄像头识别精度

6.1.1. 不同标定下距离测量精度

根据实验，红色小球在 20 cm 时，在摄像头中的像素为 70，那么 $K=1400$，下表 2 是小球在 15~25 cm 时，摄像头测出的摄像头到小球的距离数据。

Table 2. Camera measurement data of the ball under the 20 cm standard
表 2. 小球在 20 cm 标准下的摄像头测量数据

实际距离/cm	15.00	16.00	17.00	18.00	19.00	20.00	21.00	22.00	23.00	24.00	25.00
测量距离/cm	15.38	16.47	17.28	18.18	19.17	20.00	21.05	22.04	22.95	23.93	24.77

小球在 30 cm 时，在摄像头中的像素为 49，那么 $K=1470$，下表 3 是小球在 25~35 cm 时，摄像头测出的摄像头到小球的距离数据。

Table 3. Camera measurement data of the ball under the 30 cm standard
表 3. 小球在 30 cm 标准下的摄像头测量数据

实际距离/cm	25.00	26.00	27.00	28.00	29.00	30.00	31.00	32.00	33.00	34.00	35.00
测量距离/cm	25.12	26.25	26.97	28.00	29.10	30.00	30.94	31.95	33.03	34.18	35.00

小球在不同位置时，对应的 K 值也有差异，下表 4 是小球在 10 个不同位置处对应的 K 值。

Table 4. *K* value of the ball at different distances
表 4. 小球在不同距离下的 *K* 值

距离/cm	10	15	20	25	30	35	40	45	50	55
K 值	1350	1372.5	1400	1475	1470	1470	1480	1507.5	1500	1512.5

通过表 4 看出，小球在 15、20 cm 处的 *K* 值差别较大，所以通过在 20 cm 处的标定值来测量 15 cm 处的物体误差较大，同理 20 cm 与 25 cm 处的 *K* 值差别较大，用 20 cm 处的 *K* 值来测量 25 cm 处的物体误差较大。小球在 30 cm，35 cm 处的 *K* 值相同，那么理论上用 30 cm 处的 *K* 值来测量 35 cm 处的物体，测量结果应该很准确，通过表 3 看出确实如此。

小球距离摄像头太近，会造成像在摄像头中比较模糊，如图 16 为小球在距离摄像头 15 cm 处的成像，可以看到成像比较模糊，那么对于像素的计算就不准确，计算出的 *K* 值就有较大差别。

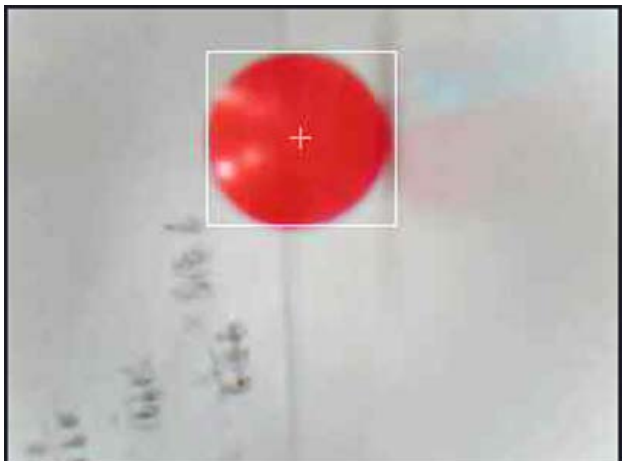

Figure 16. Small ball imaging at 15 cm
图 16. 15 cm 处小球成像

而在 30 cm 处定标，小球在摄像头中较为清晰(图 17)，对于物体像素的计算较为准确，测出的距离接近实际距离。

Figure 17. Small ball imaging at 30 cm
图 17. 30 cm 处小球成像

6.1.2. 不同物体大小精度

在摄像头测距的基础上，同样可以测得物体的大小，首先用一个已知直径 A cm 的小球，放在距离摄像头 B cm 处进行定标，该小球在图像中的像素数为 α，那么可以得到另一个比例关系 $k_2 = A/\alpha$，通过这一比例关系可以测得其他物体在同样距离处的大小。

根据实验，一个直径 2 cm 的小球分别在距离摄像头 15 cm，20 cm，30 cm，40 cm 处，在图像中的像素数分别为 91，71，49，36.5，那么 k_2 的值分别为 0.0219，0.02817，0.04082，0.05479，以下是通过摄像头测得的不同边长的正方形。

由表 5~8 可以看出，在距离摄像头 15 cm 处测量物体长度的精度相对较差，在 30 cm 处测量物体的大小精度相对较高。

Table 5. Measurement data of square size at 15 cm
表 5. 15 cm 处正方形大小的测量数据

实际长宽/cm	1.00	1.50	2.00	2.50	3.00	3.50	4.00
测量长/cm	1.05	1.62	2.04	2.65	3.25	3.75	4.26
测量宽/cm	1.01	1.56	1.95	2.63	3.14	3.73	4.21

Table 6. Measurement data of square size at 20 cm
表 6. 20 cm 处正方形大小的测量数据

实际长宽/cm	1.00	1.50	2.00	2.50	3.00	3.50	4.00
测量长/cm	1.04	1.57	2.11	2.61	3.11	3.69	4.25
测量宽/cm	1.01	1.57	2.08	2.63	3.10	3.69	4.22

Table 7. Measurement data of square size at 30 cm
表 7. 30 cm 处正方形大小的测量数据

实际长宽/cm	1.00	1.50	2.00	2.50	3.00	3.50	4.00
测量长/cm	1.02	1.55	2	2.61	3.18	3.71	4.24
测量宽/cm	0.97	1.51	1.96	2.57	3.06	3.71	4.2

Table 8. Measurement data of square size at 40 cm
表 8. 40 cm 处正方形大小的测量数据

实际长宽/cm	1.00	1.50	2.00	2.50	3.00	3.50	4.00
测量长/cm	1.09	1.58	2.02	2.62	3.23	3.78	4.32
测量宽/cm	0.93	1.53	1.91	2.68	3.12	3.72	4.27

6.1.3. DTW 算法轨迹相似度比较

用 DTW 算法将虚与实场景中对应的数据点进行时间序列相似度比较。虚拟场景中的角度信息是机器人正前方与右肩所在坐标系的角度，该角度信息在脚本中计算完成后通过 InvokeRepeating (“yan”,7f,0.8f)；将角度信息每隔 0.8 秒打印在 Unity3D 的控制台上。现实场景中机器人的角度信息是通过陀螺仪测量得到后发送给上位机，在 VS 中把蓝牙接收到的角度信息打印在 VS 的控制台中。将多次实验的数据进行数据处理，通过 Matlab 仿真中，得到的仿真图如图 18 和图 19：

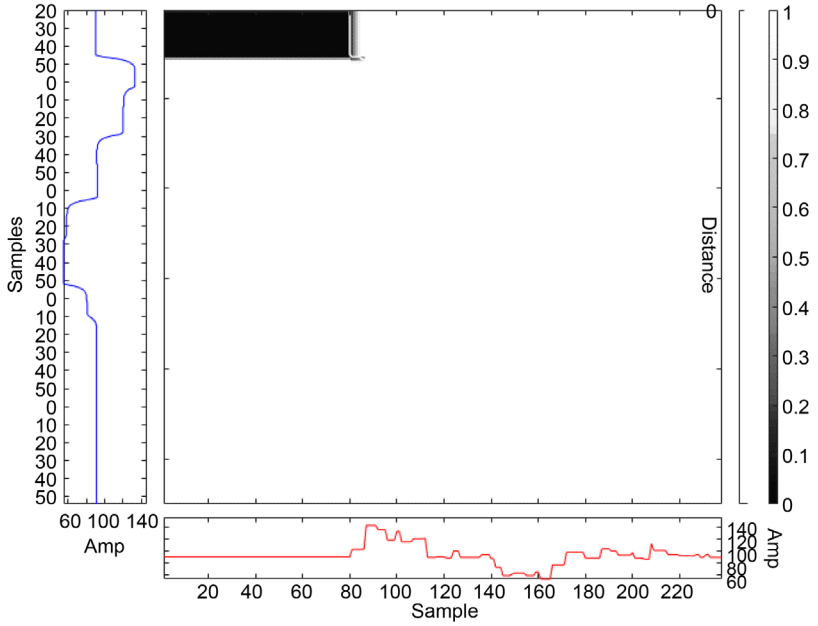

Figure 18. DTW cumulative distance matrix and optimal path (Similarity)
图 18. DTW 累积距离矩阵和最优路径(相似度)

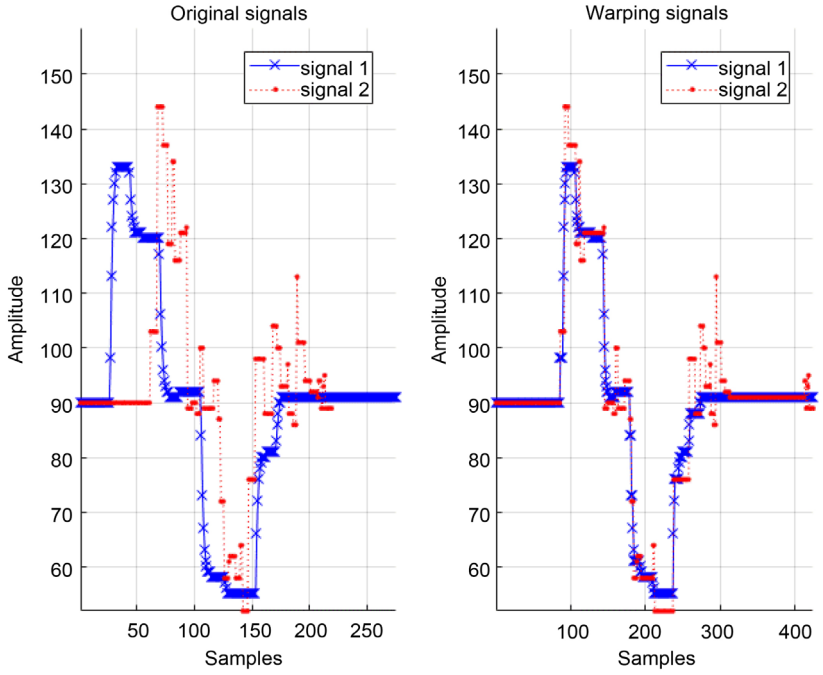

Figure 19. Rounding time series signal diagram
图 19. 归整时间序列信号图

7. 结论

1) 通过建立 Unity 3D 软件搭建起了机器人虚拟运动场景，并能够在虚拟场景中设置各类障碍物，可以采用 Navgation 导航网络模块计算出在避开障碍物的条件下，从起点到终点之间整个地图的最优路径，并利用 NavMesh 插件完成机器人的寻址避障运动控制。

2) 现实场景采用双足六自由度机器人，机器人将跟随虚拟场景中的机器人同步运动，并用 OpenMV3 摄像头识别现实场景中的障碍物图像信息，通过图像视觉处理后将障碍物的信息反馈于 Socket 服务器，虚拟场景从服务器中提取障碍物信息，并在虚拟场景中对应比例的重现，同时虚拟机器人进行躲避障碍物的运动控制，以此来达到虚实交互运动控制的目的。

3) 通过摄像头精度识别实验，从物体距摄像头的距离不同的测量数据，得出摄像头在距离为 15 cm 处精度相对较低，在距离为 30 cm 处精度相对较高。

4) 通过该摄像头来模拟真实情况下，将虚拟场景中机器人行走路径与现实场景中机器人行走路径按时间序列进行记录，通过 DTW 算法对两个路径进行相似图分析，通过 MATLAB 仿真得出参考模板与测试模板之间相似度较高，其相似度为 0.918 完成了虚实机器人与现实机器人同步运动控制，并说明了机器人遇到突发状况后的运动特性。

参考文献

[1] 陈启愉, 吴智恒. 全球工业机器人发展史简评[J]. 机械制造, 2017, 57(7): 7-10+25.

[2] 何敏. 虚实结合的远程教学控制实验系统研究[D]: [硕士学位论文]. 武汉: 华中科技大学, 2012.

[3] 马飞. 六自由度机器人虚拟实验室系统关键技术的研究[D]: [博士学位论文]. 北京: 中国矿业大学, 2018.

[4] 陈恳, 付成龙. 仿人机器人理论与技术[M]. 北京: 清华大学出版社, 2010.

[5] Lu, G.P., Xue, G.H. and Chen, Z. (2011) Design and Implementation of Virtual Interactive Scene Based on Unity 3D. *Advanced Materials Research*, **317-319**, 2162-2167. https://doi.org/10.4028/www.scientific.net/AMR.317-319.2162

[6] 季铮, 谢予星, 王玥. 基于 Unity 3D 的虚拟测量实验系统[J]. 测绘通报, 2016(10): 97-100.

[7] 古晶. RGB 到 Lab 颜色模式转换[J]. 广东印刷, 2009(5): 11-14.

[8] 梅妍玭, 傅荣. 基于 OpenMV 的 3D 定位识别系统[J]. 新技术新工艺, 2018(2): 50-53.

[9] 胡诗琴, 袁海云, 邹金池, 石俞磊, 刘雪梅, 孟莨钦. 基于时间序列相似性的体感机械手臂控制系统研究[J]. 动力系统与控制, 2018, 7(4): 287-297.

[10] 杨静, 梁浩峰, 高鑫. 用 Socket 实现服务器/客户机的数据交换[J]. 甘肃科技, 2002(7): 19.

[11] 张群, 杨絮, 张正言. 蓝牙模块串口通信的设计与实现[J]. 实验室研究与探索, 2012, 31(3): 79-82.