上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR

论文题目：面向真实视觉感知的通用机械臂仿真平台

学生姓名：范裕达

学生学号：516030910280

专　　业：计算机科学与技术 (致远荣誉计划)

指导教师：卢策吾教授

学院 (系)：电子信息与电气工程学院、致远学院

面向真实视觉感知的通用机械臂仿真平台

摘　要

在本文中，我们提出了一个新的三维固定机器人仿真平台 CyberPanda。本平台以当今综合表现最为优异的四代虚幻引擎 (Unreal Engine 4) 为主体框架，并以 Bullet3 为物理学引擎。同时，在用户端利用远程过程调用的方法提供了 Python 的相关接口，为用户提供了简洁方便的构建场景、采集数据的方法，并且拥有准确的物理模拟和三维渲染效果。CyberPanda 拥有相比于传统模拟器更加逼真的仿真结果和更加便于调用的代码接口，使得研究人员可以轻松的定制自己的仿真数据集，从而减轻机器人视觉研究中仿真训练与实际应用的感知差别带来的负面影响。

关键词：机器人视觉，仿真环境，深度学习，三维图形渲染

CYBERPANDA：A UNIVERSAL ROBOTIC ARM SIMULATOR TOWARDS PHOTOREALISTIC VISUAL PERCEPTION

ABSTRACT

In this thesis, we propose a novel 3D universal stationary robot arm simulator towards realistic visual conception: CyberPanda. CyberPanda is built on the most powerful and outstanding real-time 3D creation platform: Unreal Engine 4, thus providing our simulator fantastic picture quality. Besides, we employed Bullet3 engine as our physics core to simulate the dynamics in the virtual reality. Meanwhile, we utilize the google remote procedure call framework, making the user of CyberPanda possible to use his Python script to launch and run CyberPanda. Our framework is capable of conducting efficient and precise simulation and has a simple and straightforward Python application programming interface, which makes it really friendly to the developers and researchers of the deep learning algorithm. The researchers are enabled by our framework to construct their own dataset without too much effort, and the training of the robot arm tasks will also benefit a lot from the high-quality synthetic data generated by CyberPanda.

Key words: Robotic Vision, Simulation, Deep Learning, 3D Rendering

目 录

第一章　绪论

1.1　研究背景及意义

尽管有着相当长的研究历史，机器人视觉中的许多任务对于如今的研究者依然极具挑战，例如机械臂的推动[1]、抓取[2] 等。近年来，伴随着深度学习技术[3][4] 的发展，许多相关的问题都得到的深入的研究和不错的解答。当然，这些都依赖于深度神经网络能够在庞大数据集中建立有效模型的强大的学习、泛化和推导能力。然而，这种数据驱动的学习方式依赖于大量精确的人工收集并标注的数据集。一方面，当今开源的、高质量的数据集并不能完全满足研究者们个性化的定制需求，因为机器人的物体感知、路径规划和动作决策均高度依赖于环境的变化，而不同的应用场景对于数据集的需求也是不尽相同的。另一方面，对一般的学术研究者来说，人工收集并标注一个完整的可用的数据集不仅耗时费力、价格不菲，甚至有可能是在现实世界中无法完成的。有鉴于此，越来越多的研究人员依赖于仿真环境来生成人工合成的数据。仿真环境不仅简化了数据生成和自动标注的过程，降低了研究开发的成本，而且方便了不同需求的研究人员自己定制相应的数据集。因此，仿真环境的研究与开发对于计算机视觉，尤其是机器人视觉领域，有着至关重要的意义，也愈发的得到人们的关注。

虽然利用仿真环境生成数据可以带来更为廉价、易得和个性化的数据，也可以极大的便利模型的训练，但是相比较于更为真实的实地采集和人工标注的数据，它有着一个先天的巨大的不足。研究人员们发现，在仿真环境中训练得到的深度神经网络的模型，在实际的应用场景中，往往不能取得他们在训练中所获得的优异表现，甚至有时有着巨大的差异。[5] 这种表现上的差异缘于现实场景与仿真环境在诸多特性中的差别，这一系列差别被研究人员们统称为“现实差”(Reality Gap)。现实差包括了许多方面的原因，例如场景种类的单一、物体纹理材质的有限，不灵活的相机位置、不准确的全局光照、不准确的图形渲染等因素，但总体上是仿真环境无法对真实场景进行准确模拟的种种表现。在许多情况下，现实差带来的分歧十分严重，以至于将仿真环境中训练得到的模型与算法部署到现实场景中成为了比建立、训练模型更加困难的任务。在这种情况下，如何解决这种仿真环境的现实差所带来的不利因素，成为了一个意义非凡的问题。这甚至将决定仿真环境下的学习能否真正的帮助到真实环境中的具体应用。

为了减轻乃至消弭这种现实差带来的影响，研究人员们提出了种种方法，其中有较大影响且对本工作有着相当启发意义的有二，分别是超现实算法 (Extreme Reality) 和域随机化算法 (Domain Randomization)。超现实算法是一类算法总和[6][7]，指的是使仿真的过程尽可能的贴合算法需要应用到的实际场景。具体的做法有：面向相机现实的渲染 (Photorealistic Rendering) 与精确物理 (Accurate Physics)。前者指的不是面向人眼的感知进行真实的渲染，而是去尽可能的模拟真实物理环境下相机镜头可能遭遇的失真问题，例如不同镜头特定的指纹，失真扭曲以及其它特定参数。后者指的是对相应的刚体或柔体的碰撞与运动做高保真的模拟，而不是在计算时使用大量的近似方法来进行加速以得到更好的性能。域随机化算法指的一类做法，其中心思想在于在模型训练时，其输入的训练集数据并不局限于之前生成完毕的合成数据，而是对仿真环境内的各个要素进行随机化的处理[8][9][10]，例如全局的光照、物体的材质、相机的视角、机械臂的摆放位置等等，提升训练时各个要素的方差，来让模型更广泛的识别和适应各个不同的场景，提高它的泛化程度，以期在现实场景中得

到更好的表现[11]。

在本课题中，我们将超现实算法与域随机化算法有机的结合起来部署在面向真实视觉感知的通用机械臂仿真平台中，以获得更好的仿真效果，并便利多种多样的机械臂任务的训练与部署。本平台以虚幻引擎 4(Unreal Engine 4)[1]为基础，采用了 Bullet3[2]作为物理引擎，虚幻引擎渲染管线为主体的图像引擎。另外，为了方便广大科研工作者的开发与使用，本平台在用户端使用了谷歌的开源远程过程调用系统 (google Remote Procedure Call)[3]，来实现 Python 用户端和平台服务端之间的通信与互动，从而方便了与许多深度学习框架技术的兼容。

本平台的创新点和特色主要有以下几个方面：一、面向深度学习开发者友好。它可以在用户端实现从场景构建、数据采集到模型训练与测试的一条龙服务，可以与许多深度学习框架兼容；二、对于 Python 开发者更加友好，高度的模块化和清晰的管线，使用户更加便捷的使用相关的内容；三、更为真实迅速的物理模拟，在虚幻引擎中嵌入使用 Bullet 物理引擎，可以自定义帧率等要素；四、丰富的场景捕获组件，不仅能在建立数据集的时候帮助采集数据，也可以在部署算法的时候帮助调试。

本文剩下的篇幅将会按照以下几个部分来组织：第二章将会介绍一些对于本平台的开发有启迪作用的相关工作，并分析他们各自的核心技术和优劣性；第三章将会介绍本平台的大体构造，并会着重介绍本平台的远程进程调用系统、物理引擎、渲染管线以及用户端的相关功能；第四章将会对本平台的工作进行总结，并分析当前的不足之处，给出相应的可能的改进方向。

1.2 CyberPanda 的定位与功能特性

CyberPanda，是面向真实视觉感知的通用机械臂仿真平台。通用机械臂，指的是丹麦通用机器人 (Universal Robots) 公司开发的合作机械臂系列产品，主要的型号有 UR3,UR5,UR10,UR16 等。除去新发售的 UR16 款机械臂，市场上较为常见、应用较为广泛的是 UR3,UR5,UR10 系列机械臂。这三款机械臂负重从 3 到 10 千克不等，有效工作半径从300 mm到1 300 mm不等。通用机械臂不同于移动机器人，属于固定的 (stationary) 机器人的一种，主要的运用环境也是室内，例如流水线传送带上物体的抓取、特定物品的分拣等任务。这个固定机械臂任务的特点主要有以下几个方面:

- 有效工作半径小，因此任务的范围都比较小，比较关注任务目标所在的区域，而对于远景质量的需求 (例如天气系统的模拟) 则比较有限
- 需要与环境内的物体进行连续的物理交互，而不是离散的语义层面的抽象交互
- 有高灵活的自由度，UR 系列机械臂有 6 个旋转关节，几乎可以覆盖工作区域内的所有抓取位置和姿势

针对这种情况，我们的 CyberPanda 专注于室内场景的相关任务，并给用户提供了以下功能:

- 用户可以导入自己的地形图作为背景，也可以使用 CyberPanda 预建的场景。
- 用户可以定义场景内的许多物理要素，例如全局光照，重力等
- 用户可以自主生成场景内的物体，对他的材质、光泽、形态、位置乃至面元的要素都可以定义，也可以通过 CyberPanda 的域随机化生成算法批量的生成数据。

[1]https://www.unrealengine.com/zh-CN/

[2]https://github.com/bulletphysics/bullet3

[3]https://grpc.io/

- 用户可以使用多种传感器模式来收集数据，例如 RGB 相机，深度相机等。
- 用户可以载入 UR 系列的通用机械臂，也可以通过机器人操作系统 (Robot Operating System) 所定义的统一机器人描述形式 (Unified Robot Description Format) 来载入自定义的机器人。

1.3 机械臂相关任务

本小节将介绍一些现在 CyberPanda 目标支持的任务，主要是对这些任务对于数据的不同需求做分析，以及简要的介绍一下有关的情况。

1.3.1 单目深度估计

尽管现在激光雷达的技术已经相当普及，但单目深度估计 (Monocular Depth Estimation/Recovery) 依然是几乎所有机器人视觉任务的基础，也是机器人感知环境、定位物体的关键一步。准确的单目深度估计的结果可以帮助机器人成功的重建三维场景，并对物体的姿态做出精确的估计。

单目深度估计的具体任务的经典形式是指给定一张二维 RGB 图片，对于每个像素估计其深度值 (在相机内参已知的情况下)。经典的单目深度估计的算法运用了信号处理的思想，利用马尔可夫随机场 (Markov Random Field) 对离散的深度信号进行建模[12]，利用最大似然估计的方式，将这个深度回归的问题转化为优化问题。

在使用深度学习方法解决单目深度估计的工作中，Eigen 等人的工作[13][14] 是较早的具有开拓性的工作。在他提出的模型中，运用了类似 Alexnet[15] 的卷积神经网络 (Convolutional Neural Network), 对图片进行逐层的特征提取，再在全连接层对图片进行恢复。在之后的工作中，基于全卷积网络[16] 的单目深度估计方法得到了充分的发展，这种新的网络结构抛弃了全连接层，而采用了加密-解密 (Encoder-Decoder) 的结构。在 Laina 等人的工作中[17], 使用了多种不同的预训练的固定网络作为加密者，而使用逆卷积的方法将特征图回复到输入的规模。

对于单目深度估计的任务，仿真环境需要提供机械臂摄像头位置的视角图片，并且需要场景中可见像素的深度作为真值数据 (Ground Truth Data)，这些都需要不同的传感器进行来实现。

1.3.2 推动

推动任务是机械臂的最基本的任务之一，即使没有抓手 (Gripper) 的机械臂也能完成特定的推动任务。最早的推动任务的研究来自于 Lynch[18]。在那时，他将推动理解为物体发生错位或碰撞之后抓手与之接触并将其推动到指定位置对齐的一个准静态过程 (Quasi-Static Planar Motion)。在推动任务研究的早期，研究人员们所关注的还多是接触的模式 (Contact Mode): 是滑动接触 (Sliding Contact)、滚动接触 (Rolling Contact)，或是固定接触 (Fixed Contact)。早期的研究大多关注于准静态过程下平面的摩擦力对控制系统的影响，多是从动力学的角度除法解决控制模型的问题。在最近的研究中,[19] 研究了利用机器人的推动在障碍环境中将给定物体推动到指定位置的方法。当然，推动动作也有其它的运用方式，例如在机器人导航或者物体放置[20] 的任务中，推动动作可能用来移开物体腾出空间，但总之并没有脱离这个问题定义的范畴。

推动任务的普适性较大，对于抓手无法抓取的物体来说，依然可以利用推动来进行操作。对于推动任务的仿真来说，以下几个要素可能是作为智能体的输入的:

- 视觉传感器的信号
- 接触点的力学信号

另外仿真环境应该具有碰撞检测的能力，以判断路线是否合法，以及是否到达指定位置。

1.3.3 抓取

对于机械臂抓取任务的设计与研究历史悠久，最早研究的焦点集中在设计和利用弹性手指 (elastic fingers) 来执行收集的任务[21]。近年来随着深度学习的崛起，抓取任务的研究和解决方案也愈发的和三维视觉联系紧密。

对于已知三维模型的物体，其物理模型已经得到了充分的研究，包括在接触点进行相应的物理建模[22] 亦或是将固定式接触限值物体移动的思想方法迁移到抓取任务中来[23]。而抓取任务的控制设计，也有了较为成熟的流程：对于模型已知的物体进行抓取，会预先计算可能的最优抓取点[24]，而后在实际的场景中通过三维重建等方式对物体进行 6D 姿态的估计[25][26]，确定最优抓取点坐标变换之后的位置。这种方法比较适合于物体表面数据以及定位和姿态已知的情况，例如给定已经预扫描过的物体进行分拣收集等任务，而不能迁移到表面数据或者物理特性不明的物体上。

对于未知三维模型的物体的抓取，研究人员们依赖深度学习技术进行了深入了研究，并形成了许多行之有效的解决方案：例如在三维空间中对平面中的抓取点进行检测或者回归[27][28]，又或者利用核表示的方法对物体的表面几何进行编码，并基于相似性的估计进行抓取点的检测与搜索[29]。当然，这些检测与搜索的过程都可以使用深度神经网络进行学习与加速。

对于机械臂的抓取任务来说，最重要的莫过于要拥有正确的视觉感知，以得到精确的姿态估计；其次需要能够正确灵活的操作机械臂的各个关节，使抓手和物体接触点的物理能够得到正确的仿真。

第二章　相关工作

在本章我们将介绍一些对于本平台的开发和构建有启发意义的相关仿真环境，分析他们的优劣，并说明他们各自的创新点和核心思想。同时也将介绍一些本平台使用到的技术工具，对其核心机制加以阐释。

2.1　其它仿真环境

2.1.1　Actin SDK

Actin SDK[1]是由 ENERGID 公司开发的，面向广大机器人业界人士的，拥有集成开发环境的机器人控制与仿真软件。Actin 以 OpenGL 作为三维图像引擎，以嵌入的闭源软件作为物理引擎，并且能在绝大多数的平台与系统中部署。Actin 的最大特色在于提供了实时的机器人控制、规划与测试模块，能够满足工业开发、商业应用和学术研究的种种不同的要求，拥有相当强的普适性。在 Actin 中，机器人按照用户事先在计算机辅助设计 (CAD) 中规划好的路径来决策与行动，以避免与自身组件或者环境物体发生碰撞，或者满足一些其他的刚性条件。对于多个机器人智能体协同工作或者同时操纵多个机械臂的情况，Actin 也内置了实时的针对高自由度机器人的碰撞规避算法，以方便用户的开发。因此，Actin 的开发与使用，更适合标准化的工业场景，也在相关的场景中有着广泛的应用。Actin 的主要特色有以下几点：

- 提供了在线与离线多种的编程开发方式，便于 Actin 的使用
- 提供了对于多个机器人或者多机械臂的协同控制
- 提供了对外的传输控制协议 (Transmission Control Protocol)
- 建立了碰撞的检测和规避机制
- 提供了对于传感器的感知以及输入输出 (I/O) 支持
- 提供了利用 CAD 规划路径的方式
- 兼容了最多的机器人描述的方式

虽然 Actin 开发的历史悠久，而且在业界的应用广泛，但其专业性较强，在使用时用户需要拥有一定的工业辅助设计的基础才能驾驭。并且在其设计中，更加注重具体场景的机器人控制与仿真的功能，而不适合用来作为深度学习的数据采集的模块，因而他也更适合成为工业开发人员的帮手而不是深度学习用户所需的仿真环境。另外，Actin 并不是开源的软件，对于需要源码进行的开发来说也是一大阻碍。

2.1.2　V-REP/CoppeliaSim[30]

V-REP(现已更名为 CoppeliaSim), 全称是虚拟机器人实验平台（Virtual Robot Experimentation Platform），是由 Coppelia Robotics 开发的一款集成式的机器人仿真平台，在其中用户可以对场景的任何一个要素进行定制，例如机器人身上的机械臂，传感器，场景中的相机与灯光，这也极大的丰富了用户开发时的自由度。

V-REP 并不依赖于中心化的进程或者架构，而是采用了分布式的体系。在此体系中，V-REP 的进程们相对独立的运行，并且可以由用户自主决定是否开启相对应的进程。V-REP 使

[1] https://www.energid.com/actin

用了内部自建的三维图像引擎，使用了开源的 ODE/Bullet/Vortex/Newton 作为其物理引擎，并且本身也是在开源的协议下进行开发的，无疑对于开源社区的需求更加友好。V-REP 的整体框架相当的灵活，用户可以在每一个物体中插入自己的嵌入式脚本，或者利用远程过程调用的相关接口进行单独的控制。

V-REP 的场景要素相当的丰富，用户可以修改、设计节点 (Joint)，路径 (Path)，图形 (Shape)，相机 (Camera)，光照 (Light)，假体 (Dummies)，接近度传感器 (Proximity Sensor), 渲染传感器 (Rendering Sensor)，力学传感器 (Forces Sensor)，凸面 (Mills), 图像数据流 (Graph) 等要素，对场景进行高度个性化的搭建。

V-REP 内也使用了四种多样的物理引擎，可以快速准确的进行动力学的计算，对物体的碰撞进行迅速的检测与模拟，其中包括了以下几个重要的计算模块:

- 正向与反向的动量计算模块 该模块基于机器人逆动力学数值解法中的阻尼伪逆法 (Damped Least Squares/Levenberg-Marquardt Method) 进行求解，这种方法能够鲁棒地对于机械臂的位置进行准确的计算，尤其是在目标点接近奇异位置的时候也能得到正确的收敛值。
- 动力和物理模块 该模块采用了 Bullet 作为物理引擎，主要负责对刚体的机械运动做相应的模拟。
- 路径规划模块 对于完整约束或非完整约束的路径规划问题，尤其是汽车类的机器人载体，提供路径规划的算法。
- 碰撞检测模块 对于图形之间的碰撞约束做迅速的判断。
- 最短距离计算模块 对于图形之间的最短距离做计算。

V-REP 无疑为无数开发者们带来的许多的便利，其优势主要在于以下几个方面:

- 可以跨平台部署 代码完全开源，研究人员们可以方便的使用与开发。
- 模块化的结构以及良好的可扩展性 每个场景要素都可以嵌入脚本，保留了巨大的自定义空间。
- 极其强大的优化 在之前提到的计算模块中，V-REP 都使用了针对性的时序缓存策略，大大地提高了运行时的效率。

当然，作为一个如此全能的开发环境，它也有着明显的短板。V-REP 使用内建的三维渲染引擎，导致它的渲染效果较为一般，在颜色、纹理、光泽等方面也存在一定的失真，而这对于深度学习的数据采集的环节是比较不利的。

2.1.3 Gazebo[31]

Gazebo，是由 Open Source Robotics Foundation 所开发的一款三维机器人训练、仿真与测试环境，采用了面向对象的渲染引擎 (Object-Oriented Graphics Rendering Engine)。在物理引擎方面，它提供了一个调用的抽象层，从而可以兼容 ODE/Bullet/DART/Simbody 等多种物理引擎。Gazebo 的物理引擎由多种游戏引擎共同组成，拥有着相当强劲的模拟质量。

相对于 V-REP 寥寥数种传感器来说，Gazebo 提供了更加丰富的传感器以供选择，例如惯性测量单元 (Inertial Measurement Unit) 和全球导航系统单元 (Global Positioning System)。除此之外，用户还可以自主创建所需的传感器。为了使仿真环境中的传感器表现贴近于真实传感器，Gazebo 还使用了超现实算法，在仿真结果上加上了物理传感器的噪声。

Gazebo 的应用十分广泛，不同于其它许多引擎模拟的环境较为局限，Gazebo 对于不仅在室内环境有相当出色的表现，在室外场景的模拟中也维持了相当高的水准。Gazebo 中的

模拟机器人都使用了 ROS 的通信框架，因此被广泛的应用在了自动驾驶汽车，仓库机器人，四轴飞行器等种类繁多的物理机器人上。同时，Gazebo 还利用了虚拟现实的技术，为研究人员和仿真机器人进行直接的交互与控制提供了可能。

对于开发者来说，Gazebo 引人注目的特性主要有以下几点:

- 使用游戏引擎而不是传统的机械生物力学引擎作为物理引擎，获得了更好的仿真效果。
- 良好的可扩展性，对于 Gazebo 内容的进一步自定义开发都可以用插件的方式来实现。
- 命令行的支持，可以在命令行构建模拟世界并进行测试。
- 用户端和服务端分离的结构，并且在用户端用 QT 构建了较为美观和简洁的用户界面。
- 云仿真技术，Gazebo 的仿真过程可以在 Amazon,Softlayer 等服务器的云端进行，而不仅限于本地，彻底实现了用户端和服务端分离的特性。
- 对于机器人模型的广泛支持，例如 Actin 系统主要还是面向 UR 系列的机械臂所服务，而 Gazebo 已经可以包含了 PR2，Pioneer2 DX，iRobot Create 和 TurtleBot 等等不同的机器人模型。
- 良好的远程调用技术，得益于谷歌远程进程调用技术，Gazebo 可以在服务器上运行，而用户可以在任何使用 TCP/IP 协议的网络节点中进行远程的控制。

Gazebo 的对于开发者的最大便利在于真正的实现了用户端与服务端的分离，以及在云端仿真的功能，并且提供了在命令行进行调试的可能，极大的降低了远程开发的成本。Gazebo 的不足之处在于使用的 OGRE 引擎过分注重面向对象的软件结构设计，而相对的忽视了运行的效率与成本，导致近年来 OGRE 所代表的继承逻辑占主导的引擎被组合逻辑占主导的引擎所取代，其渲染的画面质量也逐渐居于下风。

2.1.4 Webots[32]

Webots 是由洛桑联邦理工学院学校 (EPFL) 最早出品，现在由 Cyberbotics Ltd. 维护开发的专业三维移动机器人的仿真软件包。它使用了开放动力引擎 (Open Dynamics Engine) 的一个分支作为物理引擎，而使用内建的渲染工具作为图像引擎。Webots 为用户提供了快速搭建仿真世界的方案，并且可以高自由度的定制每个元素的图形特性和物理特性，甚至包括了一些流体动力的特性。Webots 内置了默认的三维建模工具，同时也支持了 VRML97 的描述方式，使它可以和大部分建模软件 (如 AutoCAD，Blender 和 Inventor) 兼容，同时也可以导入复杂的地图和地形数据，这都是其它许多环境所不具备的特性。

在仿真机器人的建模方面，Webots 也有其独到之处，Webots 为仿真机器人提供了多样的传感器与执行器元件，例如驱动轮、马达、发射器和接收器等，极大的丰富了机器人的物理要素。Webots 机器人的控制程序，支持了许多主流编程语言，甚至可以简易的图形编程语言对 e-puck 等机器人进行编程。同时，为了方便移动机器人的原型制作，Webots 还提供了与真实的物理机器人的接口，使用户可以把控制程序部署到物理机器人上进行调试。

Webots 的优势和重点在于它和真实世界的交互，因此它也被广泛的应用于机器人运动研究和移动机器人原型制作中。但是很长一段时间内 Webots 都是作为需要专有许可的商业软件而不是免费的开源软件所出现的，因此其影响力在此期间不及其它开源的环境。

2.1.5 CHALET[33]

CHALET, 全称是康奈尔家居引擎学习环境 (Cornell House Agent Learning EnvironmenT), 是由康奈尔大学开发的三维室内场景机器人导航和控制环境。它基于 Unity3D 引擎进行开发, 而所采用的物理引擎也是 Unity 的内建物理引擎, 支持对家具物体和环境的修改和仿真, 同时也可以在众多的平台甚至云端的服务器部署。

与其它的仿真环境更倾向于让用户自主构建世界地形不同, CHALET 自己内建了 10 个房子, 由 58 个不同的房间相互连接而成。房间的种类十分丰富, 包括厨房、会客厅、起居室、盥洗室等, 内部的陈设也种类繁多, 包含有 70 种的家居物体。其中的一些甚至可以改变其状态, 例如可以打开烤箱的门, 橱柜的窗。同时, CHALET 还借鉴了域随机化的思想, 对物体的形状和材质进行了大量的组合, 生成了 330 余种不同的物体, 以合理的组合方式摆放在了不同的空间内。当然, 这些房屋和房间的构造并不是真实物理存在的, 而是 CHALET 开发者使用工业辅助设计的工具所绘制的, 因此房屋的陈设都相当规则, 而不会有现实中的杂乱物体。

CHALET 中机器人以第一人称视角进行操作和定位, 同时提供了连续和离散两种移动方式, 还包括了一系列与场景进行互动的动作。同时 CHALET 为更有挑战性的语言描述、路径规划的任务提供了测试的平台。CHALET 的每个房间内都会放置多种相同的物体, 甚至有可能摆放在相近的位置上, 这样仅靠语义级别的分割并不能让机器人定位到具体的某个物体, 而要提供其它方位、颜色等物理性质上的描述来让智能体进行分辨, 同时大大丰富了任务的自由度和描述语言的复杂度。

CHALET 的优势在于它自带了结构各异、陈设繁多的 58 个房间所组成的 10 个房屋, 更是有数百种不同的陈设物品, 让使用者免于自主构建室内场景的劳苦, 可以直接在仿真环境中部署测试自己的算法。但是遗憾的是, 一来这些房间都是 CHALET 开发者自己绘制而不是实地采集的, 所以物体的种类和摆放都相对的规则, 而实际的场景中多会有许多遮挡的杂物 (clutter), 在这方面将真实的场景做了一定的简化和近似。除此之外, CHALET 的劣势也相当明显: 一方面它较大的削弱了用户定制场景的自由度, 没有给用户静态场景建模的工具; 另一方面, 它的相机位置仅仅局限于第一视角, 并且机器人的模型面元 (Mesh) 也不在环境的考虑内。

2.1.6 HoME[34]

HoME, 全称是家模仿真环境 (Household Multimodal Environment), 是一个面向室内智能机器人学习的三维仿真环境。HoME 提供了多媒体的环境要素, 比如视觉信号, 听觉信号, 语义信号, 动力学信号, 与环境或者其它智能体的交互等。对于这些多种多样的要素, HoME 划分了明确的模块引擎进行管理: Panda3D 作为图像引擎进行图形的计算; EVERT 作为声学引擎, 利用射线追踪 (ray-tracing) 的技术来实现实时的声学模拟; 在语义引擎中, 对于每个物体提供了一段简短的文字描述作为语义信号, 包括了物体的种类、颜色、方位、材质、大小; Bullet3 作为物理引擎, 并对每个物体的物理我命提供了两种表示方式。

除此之外, HoME 还集成了超过 45000 种来自于 SUNGG 数据集的室内布局, 其丰富度远超之前所提到的 CHALET, 并且更为重要的是 SUNGG 数据集中的室内场景都来自于实地的扫描建模后再进行三维重建, 不同于 CHALET 由人工绘制的室内场景, 更加贴合真实的情况。除此之外, HoME 还带有与 OpenAI Gym 相兼容的 Python 框架, 更适合深度学习算法的开发者。

HoME 的显著优势在于它拥有着大规模的可定制的三维室内数据，并且要素十分丰富，除了视觉的反馈，还可以提供声学信号，物体的语义分割和语义描述等其它信息。但是，它的缺陷也十分明显：首先，HoME 并没有考虑镜头现实的渲染效果，因而他所得到的视觉信号与真正的物理传感器会有一定的偏差；其次，HoME 的建模中并没有机器人本身，因此也没有办法和环境中的物体做连续的物理交互，而只能局限于一些离散的抽象动作。

2.1.7 AI2-THOR[35]

AI2-THOR，全称是 The House of Interactions，是一个近似相机真实的智能体仿真学习框架。AI2-THOR 中包含了大量的三维室内场景，使得虚拟智能机器人能在其中执行各种任务。AI2-THOR 的任务种类十分丰富，适用人群也十分广泛，并不局限于普通的深度强化学习 (deep reinforcement learning)，也面向模仿学习 (imitation learning)，表示学习 (representation learning) 等多样化的学习框架和算法。

AI2-THOR 的整体构建基于 Unity3D 框架，使用 Unity3D 社区所提供的渲染管线，动力学模拟以及它三维的建模器 (3D modeller)。在用户端，AI2-THOR 提供了 Python 语言的接口，作为用户实现控制元件的方法，并且通过 HTTP 协议同 Unity3D 的游戏引擎进行交互。

同其它许多仿真环境所不同的是，AI2-THOR 更加关心同物体的交互。在 AI2-THOR 的场景中，大量的物体都是可操作的，并且用户的每个指令在执行后，AI2-THOR 都会自动的生成相应的 json 文件来描述每个物体的状态。除此之外，更有特色的是，AI2-THOR 中的室内场景，既不是由开发者直接手工设计的，也不是由实地场景采集数据后在仿真环境中重建的，而是由三维设计的从业者根据现实的场景做一定的加工后绘制的，相当于对采集的收据进行了手工的规整化 (refinement)，提高了场景的现实度也满足了一定的规则性。

AI2-THOR 的传感器也相当丰富，除了必须的视觉感知，AI2-THOR 也提供了压力和摩擦力的传感器。此外还支持用户自定义环境数据，部署多智体仿真任务等。AI2-THOR 的劣势在于对于室内机器人本身的模型缺乏建模，并且大多数与环境的交互依旧停留在离散状态的阶段。

2.1.8 MINOS[36]

MINOS，全称是多传感器模型室内仿真器 (Multimodal Indoor Simulator)，是一款针对多传感器模型机器人 (multisensory model robot) 在复杂室内环境的目标导向型任务训练的仿真环境。MINOS 利用了大量的室内场景数据来搭建，包括了之前 HoME 仿真平台所使用的 SUNGG 虚拟生成的数据集，以及 Matterport3D 这个由真实数据重建得到三维数据集，总计涵盖了超过 45000 个虚拟房屋和 90 个真实房屋，数据量十分充足，对于深度强化学习相关的任务也相当有挑战性。

MINOS 主要提供了三个方面的定制，包括环境的定制 (Environment Configuration)，控制模型的选择 (Agent Control Configuration)，以及对传感器数据流的定制 (Generic Sensor specification)。在 MINOS 中，机器人的建模由一个漂浮在半空中的简单圆柱体来表示，其相关的物理性质如质量、摩擦系数都可以由用户定义。MINOS 也提供了一系列控制的模式，包括压力控制和力矩控制等方式。同时对于机器人的动作，也定义了一些离散的抽象的动作。

同时，作为多传感器的室内仿真器，MINOS 提供了多种数据类型作为机器人智能体的输入，包括:

- 视觉输入 利用 WebGL 作为三维渲染引擎得到相应的画面效果

- 深度图 利用渲染管线中光栅化渲染过程中所使用的深度缓冲区 (depth buffer) 来得到相应的每个像素的景深
- 表面法线 利用物体的面元建模得到
- 接触点受力 利用物理引擎对三维形状的碰撞检测得到
- 语义分割 利用数据集中对应像素上的物体标注
- 其余各种传感器测量值 例如行进的速度，当前的位置等

MINOS 作为多传感器机器人的模拟器，其数据集相当全面，包含了生成环境和真实环境，同时感知的类型丰富，可以开展许多相关研究。美中不足的是对于机器人的建模过于简易，只用一个简单的圆柱体来代替。

2.2 CyberPanda 使用工具

在本节中，我们将介绍一些 CyberPanda 实现过程中所使用的工具，以及它们的基本功能。

2.2.1 虚幻引擎

虚幻引擎 (Unreal Engine)，是由 Epic Games 公司开发的一款游戏引擎，最早起源于 1998 年的第一人称视角射击游戏 Unreal。虽然虚幻引擎开始时是作为射击游戏开发工具出现的，但现在已经被广泛的应用于各种游戏的开发中。虚幻引擎的整体框架是由 C++ 语言所编写的，拥有很强的可移植性，能够被部署在大多数主流的平台上。在 CyberPanda 中，我们使用了虚幻引擎 4 以及相关的插件来实现我们定制的功能。

虚幻引擎的功能十分全面和强大，在 CyberPanda 中使用的主要功能有:

- 虚幻编辑器 这是一个集成式的开发环境，CyberPanda 的源代码在编译之后也可以在虚幻编辑器中测试。除此之外，使用者还可以凭借虚幻编辑器来构建世界场景、地形和地貌。
- 管道集成 虚幻引擎兼容多种标准支持媒体联通生产管道，并且对 Python 脚本实现了支持，可以实现工作流程的自动化，自动的进行数据准备和内容布局的相关流程。CyberPanda 传感器数据流的实现就是依赖于此。
- 动画编辑工具 虚幻引擎的动画编辑工具可以设计场景中要素的动作，并且具有精准的物理模拟，主要功能包括了混合空间、状态机、正向和逆向的动力学等。这些大多是通过虚幻引擎中嵌入的 PhysX 物理引擎的线程所实现的，当然在 CyberPanda 中我们对这一部分进行了重构。
- 渲染管线 虚幻引擎提供了强大的渲染功能，包括了逼真的延迟渲染，灵活的材质编辑器。在主体的渲染管线中，虚幻引擎采用了光栅化和光线追踪相结合的方式，并且拥有精确的全局光照算法。CyberPanda 中采用了相近的渲染算法。同时，虚幻引擎还能够提供丰富的后期处理效果 (Post Effect)，例如镜头眩光，虚光等效果。

CyberPanda 是作为虚幻引擎的一个游戏关卡工程进行开发的，CyberPanda 中所有的实例 (instance) 都是作为虚幻引擎的 Actor 载入到世界地图中的，因此 CyberPanda 的源码也可以被虚幻引擎打包成二进制文件直接运行，用户想要修改调试 CyberPanda 的源码也只要从 uproject 文件利用本机的虚幻引擎生成相应的工程文件即可。

2.2.2 虚幻引擎插件

虚幻引擎具有良好的扩展性，因此许多功能都可以通过个人开发者所发布的插件来实现。在 CyberPanda 中，主要使用到三个相关的插件：

- InfraworldRuntime[1] 主要功能是将 gRPC 框架嵌入虚幻引擎中，使用户可以通过远程调用的方式来使用虚幻引擎的相关接口和 C++ 蓝图。当然，在 InfraworldRuntime 中整个引擎是作为客户端使用的，与 CyberPanda 的架构恰好相反，因此 CyberPanda 对通信的方式进行了重构，只是将这个插件作为嵌入 gRPC 的一个简洁的方式而已。
- RuntimeMeshLoader [2] 主要功能是支持运行时的物体模型载入，也可以帮助 CyberPanda 重建真实的场景。
- UnrealEnginePython [3] 尽管虚幻引擎提供了对 Python 脚本的支持，但是 CyberPanda 对虚幻引擎的物理核心部分进行了重写，并且考虑到实际运用的过程中也可能需要使用其他的 Python 软件包，因此可以利用 UnrealEnginePython 插件将 Python 的虚拟机 (Python Virtual Machine) 嵌入到虚幻引擎中去，从而运行所需的 Python 脚本。

2.2.3 物理引擎

物理模拟的引擎在传统上被分为两种，分别是传统的机械生物力学引擎和现代的游戏引擎。机械和生物力学的引擎的代表包括 MATLAB Robotics Toolbox，SD/FAST 以及 OpenSim 等，主要使用高效和精确的递归算法来进行计算。当然这种高效是建立在一定程度的近似下的，比如会适当的忽略接触面的动力学，或者使用简单的阻尼模型来进行接触点建模。游戏引擎的代表工作有 ODE，Bullet，PhysX，Havoc 等，这类大多采用较为现代接触点建模方法，将模型转化为逐帧求解的最优化问题。但是其主要的不足之处在于对于活动关节这种自由度的描述都是通过刚性的连接条件来实现的，从而失去了模拟弹性材质的可能。

在 CyberPanda 的初始版本中，选择了 Bullet 引擎作为其物理引擎。Bullet 拥有以下特点和优势:

- 三维建模能力 Bullet 支持多种图形，包括球体、圆柱体、圆锥体、以及任意形状的凸包甚至不凸的多面体。
- 碰撞检测能力 Bullet 支持离散检测和连续检测两种模式。
- 对于柔体的支持 除了刚体的碰撞模拟，Bullet 还支持类似绳子、衣物等可变形物体的物理模拟。

除此之外，Bullet 也包括了经典的逆向动力学模拟，这对于确定机器人关节的坐标，是非常有效的方法。

2.3 本章小结

在本章中，我们讨论了两部分的内容。一是介绍了之前相关的室内机器人的仿真环境，并分析了它们具体的优劣。这些工作的劣势大多集中在：欠佳渲染效果或者非相机真实的视觉效果，缺乏对于机器人本身的建模已经与环境的交互还停留在抽象的语义层面。二对 CyberPanda 中使用到的开发者工具进行了简略的介绍，介绍了所使用的主要的功能以及选择这些工具的原因。

[1] https://github.com/vizor-games/InfraworldRuntime

[2] https://github.com/GameInstitute/RuntimeMeshLoader

[3] https://github.com/20tab/UnrealEnginePython

第三章　CyberPanda 结构

在本章中，我们将对 CyberPanda 的框架结构做完整的介绍和解析，并对其中关键的部分如远程过程调用系统、渲染管线等做详细的介绍。

3.1　基本概念及定义

在本节中，我们将介绍虚拟场景主要的构成要素和它相应的定义。

- **场景**
  场景 (Scene)，在虚幻引擎中也被称为 Level，是一个世界中所有要素的总和，包括了世界地图，机械臂，其它的可操作物体，以及相对应的所有物理要素，也是用户能够统一操作的最大单元。
- **执行元件**
  执行元件 (Actor)，在虚幻引擎中为 AActor。Actor 是一种抽象的表示，它包括了一切可以被载入到场景中的要素，并且在需要的时候触发自己的功能。Actor 可以是具体的有碰撞体积的物体，比如场景中的机械臂或者被抓取的物体。Actor 也可以是虚拟的没有实体的控制元件，比如 CyberPanda 中有许多负责载入功能、接收远程过程调用的 Actor。
- **智体**
  智体 (Agent)，在 CyberPanda 指的是固定式的机械臂，CyberPanda 也支持多智体的操作，用户只要通过 CyberPanda 所提供的 RobotId 句柄就可以实现对每个智能体的逐一控制。
- **动作**
  动作 (Action)，指的是 CyberPanda 中能够控制机械臂所完成的动作，该部分的功能主要依靠 Bullet 引擎所提供的控制元件来完成，并提供了多种控制模式。
- **物体**
  物体 (Object)，指的是除了机械臂之外场景内的所有物理实体，在 CyberPanda 中，大多数物体都是可拾取的，但也支持静态的不可操作的物体。

以上便是 CyberPanda 中主要的实例类别，之后我们将着重介绍 CyberPanda 的整体框架结构。

3.2　框架结构

如图 3-1 所示为 CyberPanda 的大体框架结构，其中箭头的方向表示数据的流向。整个框架被划分成了三个大的部分，用不同的颜色进行标记：蓝色的是控制器，也就是用户与 CyberPanda 进行交互的部分，使用远程过程调用来实现进程间的通信；橙色部分是图像引擎，是整个仿真环境的核心部分，负责构建场景，并进行相应的渲染；绿色部分是嵌入的物理引擎，负责对场景中的机械运动进行模拟，并执行碰撞检测等任务。下面我们将逐一介绍这三个部分。

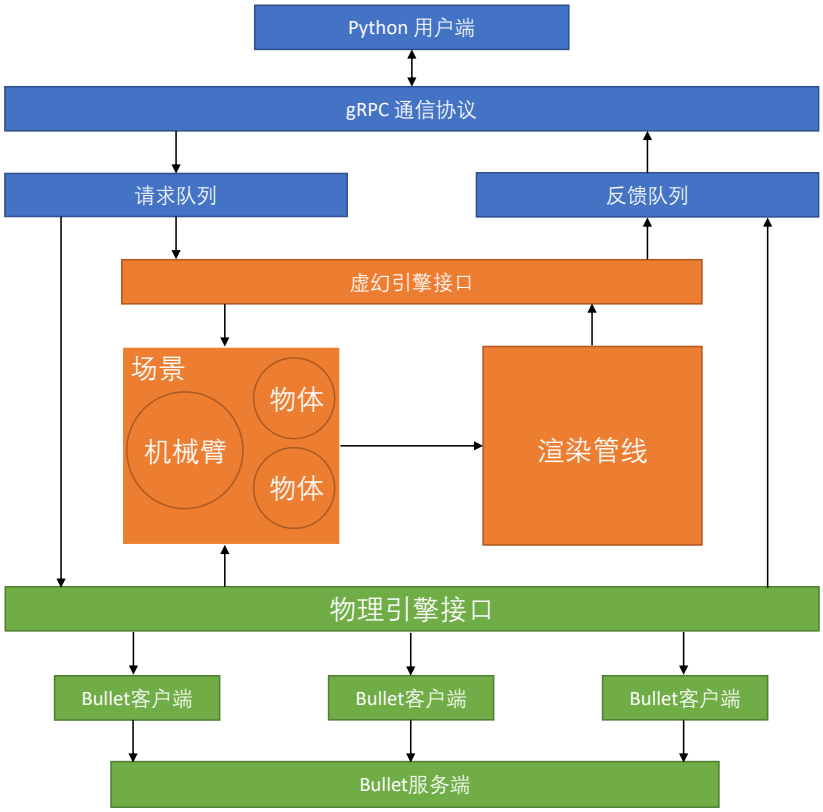

图 3–1 图示为 CyberPanda 的主要结构

3.2.1 控制器

与其它多数仿真环境不同，CyberPanda 在通讯方面的设计是与之相反的。其它的模拟器多采用的是事件驱动式的设计，例如基于 Unity3D 的 AI2-THOR 环境，便是将游戏引擎本身独立出来，作为用户端不断向控制脚本请求动作指令，并且将每个动作执行之后发生的数据、事件等发送给控制器，等待控制器给出下一步的动作命令。这样的过程通常是阻塞式的，即在下一个模拟指令发出之前，引擎都处于忙等待的状态。

而在 CyberPanda 中，我们采用的是指令驱动式的结构——服务端的功能 (包括物理引擎) 已经被集成在虚幻引擎中，并被打包成一个二进制文件从而隐去了具体的调用方式。在 CyberPanda 中，我们采用阻塞式同步调用的方式，即一次远程过程调用在用户端看来等同于一次简单的内部函数调用。

在 CyberPanda 中，我们使用 gRPC 作为远程过程调用的工具。在用户端与服务端之间，采用阻塞式的同步调用，而在用户端内部，则是采用异步的方式来加快速度。为了保证异步编程的不同线程之间的命令不会相互干扰，我们在服务器端使用了请求队列 (Request Queue) 和回复队列 (Response Queue) 来实现并发调用的串行化。请求队列是各个线程发出指令后放置请求的数据结构，而回复队列则放置了服务器给出的回复，这两个队列都是线程安全的。其处理的流程相当简单，各个客户端的线程将自己的请求内容和线程号同时放入请求队列中，同时在回复队列中忙等待，直到符合自己线程号的回复出现。而服务端的线程则利用信号量在请求队列上睡眠，等待新的请求的到来，之后再进行处理，并将所得的结果放入回复队列中。这个过程简明的代码如下。

```
void AsyncCall(String context)
```

```
{
        AsyncClientCall call = new AsyncClientCall(context, this);
        requestQueue.enqueue(call);
        while (responseQueue.head().thread != this);
        //当前的回复不是自己的，则忙等待
        Response response = responseQueue.dequeue().context;
        return response;
}

void AsyncResponse()
{
        this.sleep(requestQueue);
        //这里requestQueue中会有信号量来唤醒在它上面睡眠的服务端线程
        AsyncClientCall call = requestQueue.dequeue();
        Responce responce = Proceed(call);
        ResponseQueue.enqueue(response);
}
```

gRPC 具体的通讯协议，我们将在介绍 CyberPanda 具体功能时进行介绍。

3.2.2 场景

如图 3-1 所示，对于场景我们有两部分的内容要进行详细的介绍，分别是场景中的机械臂以及其他的物体。图 3-2 展示了 CyberPanda 中所有 Actor 的继承关系，机械臂在 CyberPanda 中对应的是 URDFAtor，而其他的物体则是 PrimitiveActor 的各个派生类，这两部分的物体 CyberPanda 将他们统一的抽象为了 PhysicsActor，继承了虚幻引擎中 AActor 的相关的虚拟方法。当然，除了 PhysicsActor 这种具体的物理 Actor，CyberPanda 中还有许多的抽象 Actor，比如 ServiceActor。Service Actor 是场景还没有生成其它物体之前最早启动的一个 Actor，也是整个 CyberPanda 的入口。所有的 Actor 都继承了 BeginPlay 和 EndPlay 的方法，在引擎启动或者指令生成相应的物理 Actor 时，BeginPlay 方法都会被调用，使相应的 Actor 被载入到场景中，而在引擎关闭或者指令重置场景时，EndPlay 方法都会被调用，使相应的 Actor 从场景中被移除。

ServiceActor 在整个 CyberPanda 中居于中心位置，它的主要功能有以下几点:

- 作为程序的唯一入口，唤醒所有的 CyberPanda 组件。
- 负责维护 gRPC 的通讯功能，作为服务端接收请求分发任务。
- 维护场景和物理引擎的对应关系。

在 3.2.2 节的剩下篇幅中，我们将着重介绍这些 PhysicsActor。

3.2.2.1 URDFActor

机械臂的建模是通过统一机器人描述规则 (Unified Robot Description Format，简称 URDF) 来进行的，该框架由机器人操作系统 (Robot Operating System) 所开发，提供了由 C++ 编写的解析器，并且在机器人视觉的相关问题中有着广泛应用。

URDF 的描述方式十分的简明扼要，它将所有的机器人抽象成一个有向的树，每一个可以活动的关节都作为一个树上的节点，而连接这些节点的机械臂则被抽象成树上的有向边，从父亲节点指向孩子节点。在 URDF 文件中，共有三种最为基本的数据类型:

base base 是整棵树的唯一的树根，在物理中一般也对应移动机器人的底座或者固定机器人的安装点。对于移动机器人来说，这个点可以被认为是三维空间中的自由点；对于

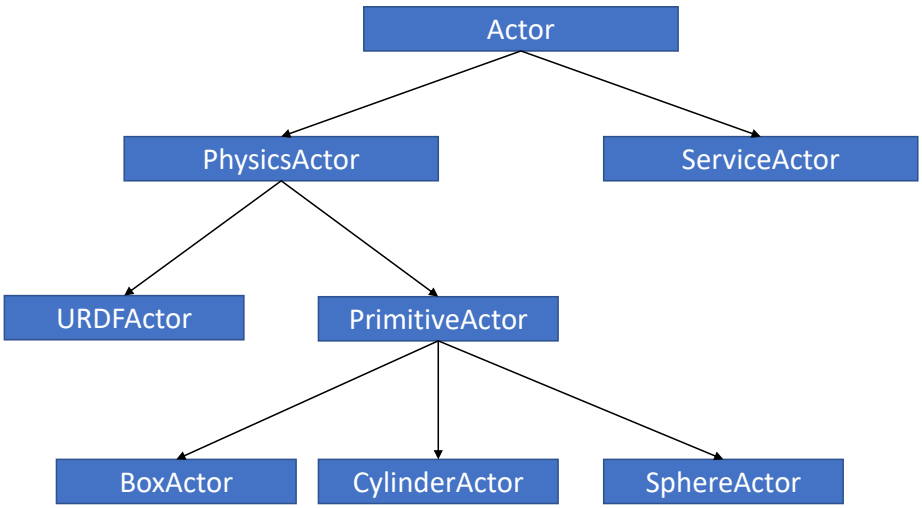

图 3–2 图中所示为 CyberPanda 中 Actor 的继承关系

固定的机器人来说，则需要提供 xyz 空间内的三维坐标。

joint 如图 3-3 所示，joint 是树中的所有非根节点，也就是机器人身上所有可以活动的关节，例如平面旋转关节或者立体旋转关节。对于每个 joint 节点，首先需要提供它相对于父亲节点的位移，其次对于平面旋转关节来说，还要提供它的旋转轴。此外，对于每个 joint 节点，必须有其两侧的 link 的信息。

link 如图 3-4 所示，link 是指的连接两个 joint 或者 joint 和 base 的机器人肢体。link 作为机器人的肢体，需要有相应的几何要素 (Geometry) 构成一定的碰撞体积，来得到正确的物理模拟。

虚幻引擎本身并没有提供载入 URDF 的方法，这个功能是由 Bullet 物理引擎所实现的。该部分功能的主要接口都集中在 URDFActor 中，其中 URDF 的对应数据类型大多数都在 CyberPanda 中得到了对应的实现。

在 CyberPanda 中，URDF Joint 对应的是 FRJoint，可以定制的要素有

- type 类型，指的是关节的种类，可以是绕轴旋转的关节，也可以是固定的关节 (没有任何自由度)、沿轴滑动的关节、在平面内运动的关节甚至是完全自由的关节。
- effort 受力上限，关节的应力超过这个阈值会对机械臂造成损伤，可以通过这个来判断规划的抓取方案是否在安全范围内。
- velocity 速度上限，可以是线速度也可以是角速度，和 effort 一样也是作为一个安全阈值所出现的。
- damping 关节运动时的阻尼值，适用于沿轴滑动型的关节。
- friction 关节的最大静摩擦系数。

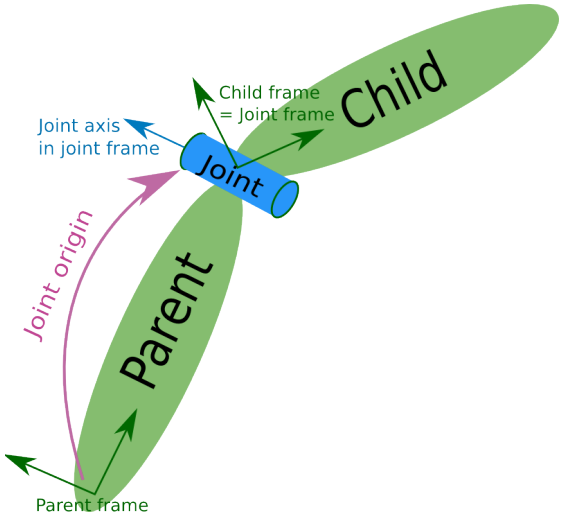

图 3–3 Joint 的图例，图中展示的是一个绕轴旋转的关节

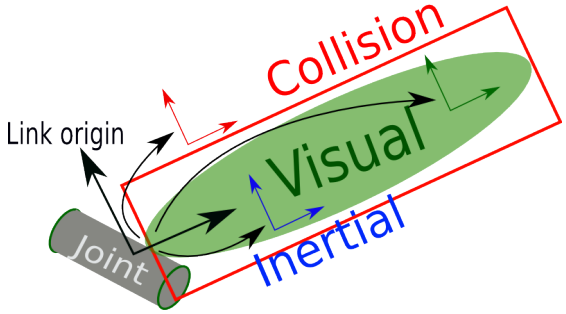

图 3–4 Link 的图例，图中展示的是一个碰撞模型为长方体的肢体

当然，除此之外，URDF 中的行为模仿参数 (mimic) 在这个版本的 CyberPanda 中还未能实现。

在 CyberPanda 中，URDF Link 对应的是 FRLink，其中可以定制的要素有

- Inertial 惯性，其中包括了它的姿态，质量，以及由三维对称矩阵来描述的转动惯量。
- Visual 视觉感知，包括了它的几何模型，颜色，材质等要素。
- Collision 碰撞模型，指肢体的三维物理模型。

Cyberpanda 利用 XML Parser 所生成的解析器，完成了自己的机器人模型构建器，利用了在抽象机器人树上深度优先遍历的方式实现了模型的增量构造。

3.2.2.2 PrimitiveActor

PrimitiveActor 是 PhysicsActor 的另外一个派生类，不同于机械臂的建模方式，他们的模型是基本确定的，只是几个内部参数需要确定而已。

PrimtiveActor 作为一个抽象类，具体的实例共有三种：长方体 (BoxActor)，圆柱体 (CylinderActor)，以及球体 (SphereActor)，与虚幻引擎中的 StaticMeshActor 相对应。除了他们的形状、位置、姿态等通用的参数之外，用户还可以定制他们的其它的物理特性，比如表面的材质。

如图 3-5 所示，在平面上所放置的 PrimitiveActor 从左到右分别是 CylinderActor，SphereActor 以及 BoxActor，可以看到他们除了大小不一之外，还使用了不同的材质，分别使用了

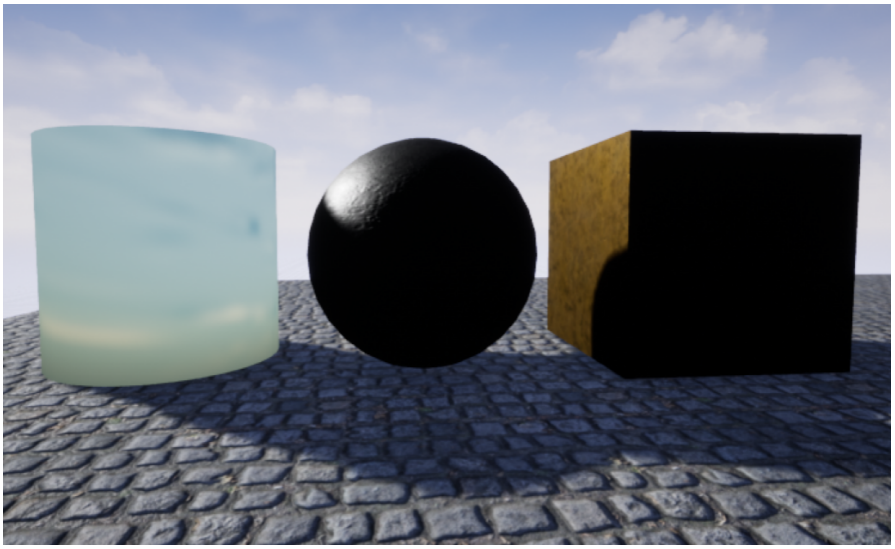

图 3–5 一个关于 PrimitiveActor 的图例

虚幻引擎所提供的天空、钢铁以及黄金的表面材质包。

3.2.3 渲染管线

所谓渲染，指的是将 CyberPanda 中构建的三维世界绘制到二维平面上去的过程。CyberPanda 使用了以虚幻引擎的渲染管线为主体的渲染算法，同时针对深度学习任务的需要，做了相应的调整。在 3.2.3 小节中，我们将介绍最为经典的两种渲染框架和算法，分别是光栅化渲染 (Rasterization) 算法以及光线追踪 (Ray Tracing)[37] 技术。

3.2.3.1 光栅化渲染

光栅化渲染指的是一系列将三维的模型投射到栅格化的平面上的操作。光栅化渲染有着相当悠久的历史，在 1974 年深度缓冲区的设计被提出之后光栅化渲染就逐渐成为了主流的渲染算法，直至今日大多数游戏引擎和显卡使用的还是光栅化渲染的框架，这也得益于光栅化渲染较高的性价比，只耗费较少的计算却能得到相当不错的渲染效果。

光栅化渲染的核心思想在于，不按照屏幕上像素的顺序来进行绘制，而是按照任意的物体的顺序在画面上并行的绘制投影。对于一个复杂的三维物体的模型来说，最为常见的一种刻画方式就是用多面体进行近似的建模，那么整个场景的最小单位就是一个个的三角形面元，而光栅化渲染考虑的就是把每个三角形面元依次投射到摄像机镜头前的屏幕网格中。

图 3-6 展示了一个光栅化算法的可视化模块，一个典型的光栅化渲染的流程是首先将三角形的顶点投射到对应的像素中，同时将对应像素的连边也进行栅格化，确定三角形在屏幕网络中所包括的区域，并在屏幕上填充相应的色彩。这样的一个完整的三角形投影绘制的过程，被称为一个图像计算单元的绘制调用 (Draw Call)，这也是 GPU 的最基本的工作内容之一。

当然，不可避免地这些三角形面元的投影会有相互覆盖的情况，也就会导致同一个像素被渲染多次，这种现象被称为 Overdraw。因此光栅化的渲染必须保证每个像素点所对应的都是离相机最近的面元，而深度缓冲区 (G Buffer) 的技术则针对性的解决了这个问题。深度缓冲区对于每个像素都维护了当前绘制的点对应的深度，每次需要重新绘制时，则用新

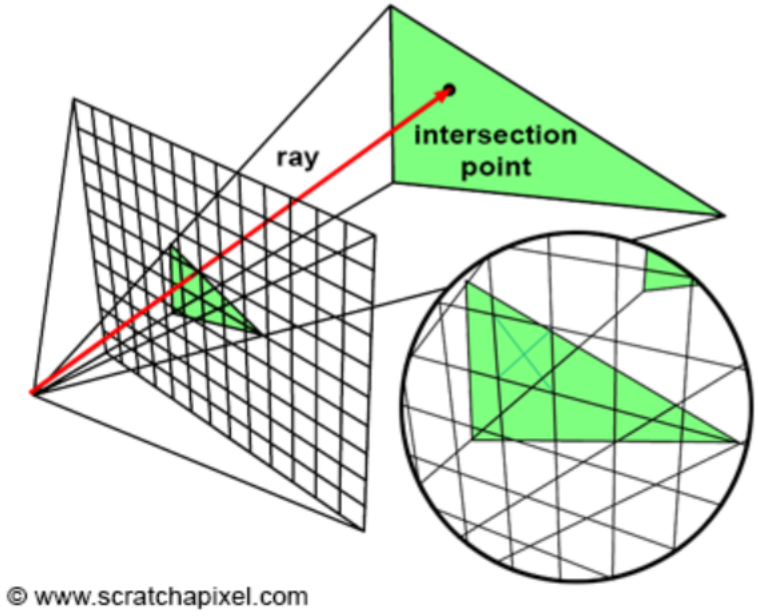

图 3–6 该图是一个三角形面元光栅化的过程

面元的深度和深度缓冲区中当前像素的深度进行比较，从而决定是否覆盖该像素或是放弃这个像素的绘制。在这种技术的帮助下，面元绘制的顺序并不会影响最终的结果，因此每个面元的 Draw Call 都可以并行的进行，大大加速了绘制的过程。

相较于逐像素渲染的算法而言，光栅化渲染的算法要在效率上高出许多，主要的原因有二：一是在大多数的场景中，物体都比较稀疏，导致很多像素并没有被前景覆盖到而都是背景的颜色，这样对物体的面元进行逐个的渲染就会自动地忽略这些不需要考虑的像素点，无形之中提高了效率；二是这种按照面元连续绘制地方式也是符合缓存的空间连续性 (cache coherency) 的，因此可以获得更好的缓存命中率，并且降低缺页错误 (Page Fault) 的概率。

一个典型的光栅化渲染的管线主要包括三个阶段。第一阶段是应用程序阶段，CyberPanda 也处于这个阶段，这个阶段的所有操作均在 CPU 的计算单元中实现。包括对世界的建模、物理碰撞的模拟、整个环境的仿真等任务，从而得到相机视角下的世界窗口的三维数据。第二阶段是 GPU Draw Call 的阶段，这个阶段也可以细分为两个部分：第一部分的主要功能是对顶点和多边形进行相应的几何变换，包括将世界坐标系变换到相机坐标系，对顶点进行着色等；第二部分则是核心的光栅化渲染的阶段，实现的就是前文所描述的光栅化的过程，得到画面图像。第三个阶段则回到了 CPU 中，应用程序会根据其需求对渲染的结果进行后期处理，在 CyberPanda 中，我们就进行了一系列面向相机真实的处理方法，比如为照片添加相机的噪声、指纹等。

3.2.3.2 光线追踪算法

光线追踪 (Ray-Tracing) 算法是光线投射 (Ray Casting) 算法的进一步扩展，采用了与光栅化渲染对偶的思想，对相机视野平面中的所有像素进行逐一的渲染。光线追踪的提出可以追溯到 1980 年贝尔实验室的相关工作，拥有比较久远的研究历史。尽管提出的时间并不算晚，但是在实际生产中采用光线追踪算法的引擎并不多，主要原因是其计算量太大，当时

的硬件性能很难负担。而近年来，随着英伟达推出带有光线追踪功能的 RTX 系列显卡，光线追踪技术优秀的画面质量使得它愈发的受到重视。

光线追踪采用与光栅化渲染的相反的计算流程，考虑将相机和屏幕网络中的每个像素都连接射线，对于每个射线都递归的计算它的光路，模拟它在场景中的反射、折射等物理行为。然而如何刻画光子的运动模型一直是光线追踪算法的一大瓶颈之一，直到 1986 年渲染方程 (Rendering Equation)[38] 的提出。在渲染方程中，作者将求解全局光照的问题转化为了求解每个点的光子辐射度的问题，即从可见的每个点射向相机位置的光线的强度，作者对光的辐射模型做了如下的刻画:

$$L_{\mathrm{o}}(\mathbf{x},\omega_{\mathrm{o}},\lambda,t)=L_{\mathrm{e}}(\mathbf{x},\omega_{\mathrm{o}},\lambda,t)\ +\int_{\Omega}f_{\mathrm{r}}(\mathbf{x},\omega_{\mathrm{i}},\omega_{\mathrm{o}},\lambda,t)L_{\mathrm{i}}(\mathbf{x},\omega_{\mathrm{i}},\lambda,t)(\omega_{\mathrm{i}}\cdot\mathbf{n})\,\mathrm{d}\,\omega_{\mathrm{i}}$$

其中 $\mathbf{x}$ 是所求解的辐射点，λ 是光线的波长，t 是时间，都是确定的参数。$L_{\mathrm{o}}(\mathbf{x},\omega_{\mathrm{o}},\lambda,t)$ 表示在 t 时刻从 $\mathbf{x}$ 点沿着 ω_{o} 方向射出的波长为 λ 的光的总辐射度，也就是观察位置所看到的该点的亮度。$L_{\mathrm{e}}(\mathbf{x},\omega_{\mathrm{o}},\lambda,t)$ 则表示 $\mathbf{x}$ 作为光源按照类似 L_{o} 的定义发射光的辐射度，$f_{\mathrm{r}}(\mathbf{x},\omega_{\mathrm{i}},\omega_{\mathrm{o}},\lambda,t)$ 则表示光子由 ω_{i} 反方向入射，散射至 ω_{o} 方向的双向反射率分布函数，$L_{\mathrm{i}}(\mathbf{x},\omega_{\mathrm{i}},\lambda,t)$ 表示沿着 ω_{i} 反方向入射的光的强度，$(\omega_{\mathrm{i}}\cdot\mathbf{n})$ 表示 ω_{i} 向量与该点的外向法向量的内积，而 Ω 则是表示以该点法向量为中心轴的外向半球面。

当然，在复杂的真实环境中，物体表面的双向反射率分布函数 (Bidirectional Reflectance) 的形式都很复杂，甚至于没有解析的形式。因此，渲染方程在通常情况下也不会有解析解，而要依赖数值解法来得到结果，这其中有两种解法取得了较大影响。利用有限元算法的计算路线进行求解的衍生出了光能传递算法 (radiosity algorithm)，利用蒙特卡洛方法进行无偏估计的计算路线衍生出了光子照射 (Photon Mapping)、路径追踪 (Path Tracing) 等一系列算法，而虚幻引擎的 Lightmass 模块就使用了 Photon Mapping 的算法。

3.2.3.3 CyberPanda 渲染管线

CyberPanda 的渲染管线以虚幻引擎的 FDeferredShadingSceneRenderer 为主体，并定制了一些辅助的功能，以增加渲染的真实感。在本段落中，我们将对 CyberPanda 的延迟渲染管线做简要的介绍:

- Preprocessing 渲染开始之前会进行一定的预处理，将一些不会出现在视野中的物体排除掉 (例如离传感器距离过远的物体)，进行可见性删除，降低渲染过程的计算量。
- Initialize 进行真正的可见性计算，同时计算出深度图，将完全被遮挡的物体去除。
- Base Pass 进行初步的渲染计算，将不透明的物体或者具体材质表面的物体的渲染结果输出到 G Buffer 中。同时，每个物体预先计算的光图 (Light Map) 和天空的直接光照也会参与颜色的计算。
- Grouping Stage 为了避免 Quad Shading(因为 GPU 单次渲染的最小单元是四个像素会带来大量的重叠渲染) 带来的问题，对于包围框比较接近的模型，合并到一起进行绘制，减少 Draw Call 的调用，也提升渲染的效率。
- Global Illumination 进行全局光照的计算，这里虚幻引擎采用了一种光栅化渲染和光线追踪复合式的算法。
- Lit Translucency 对于半透明物体自己内部的阴影进行处理。
- Post Processing 添加各种后期效果，使得渲染的画面更加的真实美观。

图 3-7 展示了一个作为样例的室内场景，一个封闭的屋子内载入了机械臂和其他的简单元件。可以看到这个场景中的 Actor 并不是简单的白模，而是包裹了我们提供的材质。同时

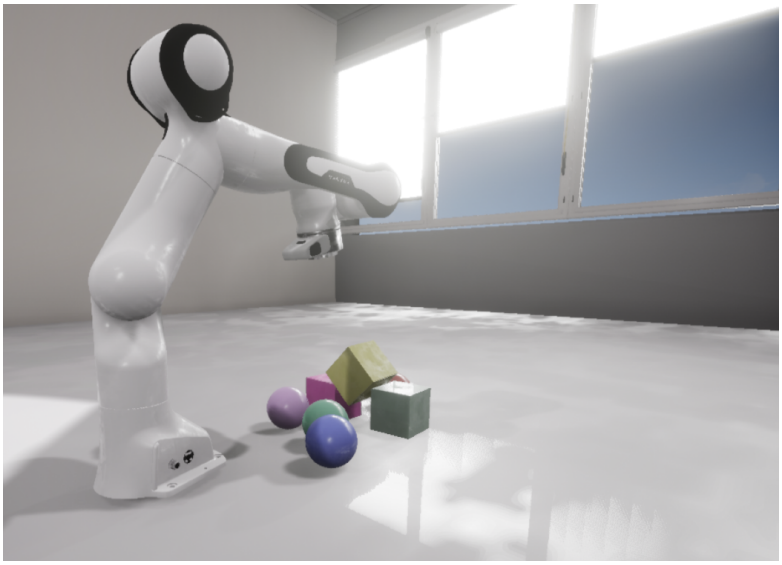

图 3–7 一个室内的样例场景

不难发现这个场景中光影的效果非常的好，所有的物体阴影都十分清晰，而且窗外的强光甚至在窗框边产生了高光的效果，这也是所添加的后期效果之一。

3.2.4 物理引擎

关于 Bullet 引擎的相关特性与优势，我们在前文中已经有了简要的介绍，在这个小节中我们将主要介绍其在 CyberPanda 中的工作方式。

首先，Bullet 自身的工作逻辑就使用了服务端与客户端分离的方式，真正的计算服务被安排在服务端进行，而每个客户端需要先连接到一个服务端才能正常的进行函数调用，这个连接是由 Bullet 本身进行调度的。CyberPanda 在设计上希望支持多线程的工作方式，也即用户可以使用多个 Python 用户端同时构建多个场景，而 CyberPanda 作为唯一的服务端要对他们的请求分别做出回应。

在多场景的资源管理上，我们对于每个场景 (Scene/Level) 都新建了一个专门的 Bullet 服务端对其中的要素进行维护。同时，对于所有 PhysicsActor，我们都实现了两个统一的方法：一是 GetPhysicsEngine()，用来得到它所在场景的所对应的物理引擎，从而进行相应的模拟，这个功能是通过 ServiceActor 所居中维护的信息实现的；二是 Tick()，表示根据给定的时间，进行一定时间的物理模拟，更新场景中物体的状态。

除了进行基础的碰撞检测，Bullet 可以帮助用户定制一些场景中的物理常数，例如用户可以设置重力的方向与大小。图 3-8 给出了这样的一个例子，在这里重力加速度被设置为地球表面的重力加速度，半径为 1 的球体在半空中的随机位置生成做自由落体运动，图 3-8 给出了一段时间后的模拟结果，可以看到球体落到地表后因为弹性碰撞纷纷向外滚去，甚至坠落到平面外。

3.2.5 通信协议与用户功能

在本小节中，我们将对 CyberPanda 中的 gRPC 通信协议使用的较为重要的 message 和 request-response 做简要的介绍，并且对用户端的功能做一定的总结。

图 3–8 一个测试物理引擎碰撞检测和重力作用的样例

名称	内容	用途
void	空	进行过程调用时所传空参数
status	Boolean, Error Message	过程调用是否成功及错误信息
PhysicsActorHandle	Integer	场景中 Actor 的唯一标识
BoxInfo	Parameters	对 BoxActor 参数的描述
CylinderInfo	Parameters	对 CylinderActor 参数的描述
SphereInfo	Parameters	对 SphereActor 参数的描述
URDFInfo	String	描述 URDF 文件的路径
PictureInfo	String, Shape	分别描述图片的像素内容和维度

表 3–1 CyberPanda 中一部分的 message 设计

首先我们将介绍 message 的种类及构成，这些 message 将会在 gRPC 的通讯网络中被跨进程传送。为了使说明更加简洁，我们以下介绍的都是的单个用户端的情况，事实上多个用户端只要在各个 message 中都加入客户端的唯一标识来进行区分即可。

表 3-1 给出了 CyberPanda 中一部分的 message 的设计与用途，表 3-2 则列举了一些用户在生成数据方面需要的一些接口，图 3-9，3-10 则分别为我们展示了同一视角下色彩通道和深度通道的结果。

当然，现在 CyberPanda 的功能还很不全面，许多控制请求还不能通过 gRPC 来实现，这也是将来进一步的改进方向。

3.3 本章小结

本章我们深入而全面的讨论了 CyberPanda 的整体架构以及各个部分的功能，并对其中关键的设计和技术进行了详细的阐释。这其中包括了控制器与渲染引擎、物理引擎之间的通信方法，场景中物理实体的载入和定制，三维渲染的主流算法以及 CyberPanda 的渲染管

名称	参数	返回值	作用
StartEngine	void	status	启动引擎
StopEngine	void	status	关闭引擎
SpawnBoxActor	BoxInfo	PhysicsActorHandle	生成 BoxActor
SpawnCylinderActor	CylinderInfo	PhysicsActorHandle	生成 CylinderActor
SpawnSphereActor	SphereInfo	PhysicsActorHandle	生成 SphereActor
LoadURDFActor	URDFInfo	PhysicsActorHandle	载入 URDF 文件
CaptureScene	void	PictureInfo	获取世界窗口图片
CaptureDepth	void	PictureInfo	获取世界窗口景深

表 3–2 CyberPanda 中一部分的 request-response 设计

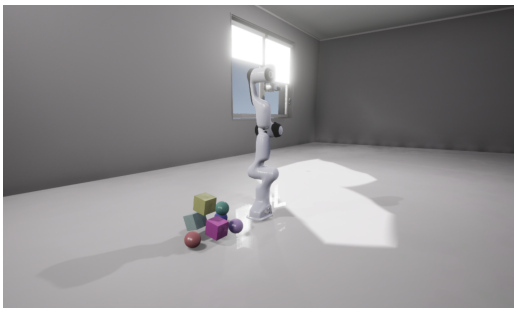

图 3–9 RGB 视图

图 3–10 深度视图

线等内容，最后介绍了 CyberPanda 在用户端的功能。对于每个模块，我们都在文中展示了样例的效果，并分析了当前功能的不足之处。

全文总结

在本文中，我们提出了全新的面向真实视觉感知的通用机械臂仿真平台——CyberPanda，并对研究的背景、相关工作、使用的工具和方法、最终的结果进行了全面的展示，同时也表明 CyberPanda 是一款拥有不俗潜力的深度学习开发软件。

我们首先分析了当前进行机械臂仿真平台开发的必要性和意义。之后，我们也指出了利用仿真环境生成的数据来进行训练的弊端——仿真环境与现实物理环境的“现实差”会导致训练得到的算法并不能成功的迁移到现实环境中，而产生这种“现实差”的一个重要原因便是各种仿真环境中不真实的视觉感知。有鉴于此，开发面向真实视觉感知的模拟器也能有助于消除“现实差”带来的负面影响，提高相关任务的训练与测试的表现。

在第二章中，我们广泛的探讨之前的机器人仿真环境，并详细的分析了他们的优势与特性、缺陷与不足。此外，我们也介绍了我们所使用的虚幻图像引擎、Bullet 物理引擎以及虚幻引擎中的插件，简要阐明了选择这些工具进行开发的原因。

讨论问相关工作之后，我们详细的介绍了 CyberPanda 的体系结构，包括其中每个模块的实现方法和具体功能。在 gRPC 的异步编程中，我们使用了请求和回复队列的维护方式；在介绍场景的章节中中，我们阐述了 CyberPanda 中物理对象的继承关系，并且对 PrimitiveActor 和 URDFActor 的载入和定制给出了详细的说明，并且给出了相应的效果图；在渲染管线的章节中，我们以虚幻引擎的渲染管线为主体，得到了不俗的画面质量；在物理引擎的篇幅中，我们展示了重力和碰撞测试的结果；除此之外，我们展示了一部分已经完成的 Python 接口调用方法，并且展示了深度取景的例子。

最后，我们也指出了当前 CyberPanda 的不足之处和未来的改进方向。

参考文献

[1] LYNCH K M, MASON M T. Stable Pushing: Mechanics, Controllability, and Planning.[J/OL]. I. J. Robotic Res., 1996, 15(6): 533-556. http://dblp.uni-trier.de/db/journals/ijrr/ijrr15.html#LynchM96.

[2] Asada haruhiko H. Studies on prehension and handling by robot hands with elastic fingers[J]., 1979.

[3] SIMONYAN K, ZISSERMAN A. Very Deep Convolutional Networks for Large-Scale Image Recognition[J/OL]. CoRR, 2014, abs/1409.1556. http://arxiv.org/abs/1409.1556.

[4] HE K, ZHANG X, REN S, et al. Deep Residual Learning for Image Recognition[J/OL]. CoRR, 2015, abs/1512.03385. arXiv: 1512.03385. http://arxiv.org/abs/1512.03385.

[5] MARTINEZ-GONZALEZ P, OPREA S, GARCIA-GARCIA A, et al. UnrealROX: An eXtremely Photorealistic Virtual Reality Environment for Robotics Simulations and Synthetic Data Generation[J/OL]. CoRR, 2018, abs/1810.06936. arXiv: 1810.06936. http://arxiv.org/abs/1810.06936.

[6] GAIDON A, WANG Q, CABON Y, et al. Virtual Worlds as Proxy for Multi-Object Tracking Analysis[J/OL]. CoRR, 2016, abs/1605.06457. arXiv: 1605.06457. http://arxiv.org/abs/1605.06457.

[7] MCCORMAC J, HANDA A, LEUTENEGGER S, et al. SceneNet RGB-D: 5M Photorealistic Images of Synthetic Indoor Trajectories with Ground Truth[J/OL]. CoRR, 2016, abs/1612.05079. arXiv: 1612.05079. http://arxiv.org/abs/1612.05079.

[8] BOUSMALIS K, IRPAN A, WOHLHART P, et al. Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping[J/OL]. CoRR, 2017, abs/1709.07857. arXiv: 1709.07857. http://arxiv.org/abs/1709.07857.

[9] TOBIN J, FONG R, RAY A, et al. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World[J/OL]. CoRR, 2017, abs/1703.06907. arXiv: 1703.06907. http://arxiv.org/abs/1703.06907.

[10] TOBIN J, ZAREMBA W, ABBEEL P. Domain Randomization and Generative Models for Robotic Grasping[J/OL]. CoRR, 2017, abs/1710.06425. arXiv: 1710.06425. http://arxiv.org/abs/1710.06425.

[11] TREMBLAY J, PRAKASH A, ACUNA D, et al. Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization[J/OL]. CoRR, 2018, abs/1804.06516. arXiv: 1804.06516. http://arxiv.org/abs/1804.06516.

[12] RAJAGOPALAN A N, CHAUDHURI S. An MRF model-based approach to simultaneous recovery of depth and restoration from defocused images[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999, 21(7): 577-589.

[13] EIGEN D, PUHRSCH C, FERGUS R. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network[J/OL]. CoRR, 2014, abs/1406.2283. arXiv: 1406.2283. http://arxiv.org/abs/1406.2283.

[14] EIGEN D, FERGUS R. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture[J/OL]. CoRR, 2014, abs/1411.4734. arXiv: 1411.4734. http://arxiv.org/abs/1411.4734.

[15] KRIZHEVSKY A, SUTSKEVER I, HINTON G. ImageNet Classification with Deep Convolutional Neural Networks[J]. Neural Information Processing Systems, 2012, 25. DOI: 10.1145/3065386.

[16] LONG J, SHELHAMER E, DARRELL T. Fully Convolutional Networks for Semantic Segmentation[J/OL]. CoRR, 2014, abs/1411.4038. arXiv: 1411.4038. http://arxiv.org/abs/1411.4038.

[17] LAINA I, RUPPRECHT C, BELAGIANNIS V, et al. Deeper Depth Prediction with Fully Convolutional Residual Networks[J/OL]. CoRR, 2016, abs/1606.00373. arXiv: 1606.00373. http://arxiv.org/abs/1606.00373.

[18] MASON M T. Mechanics and Planning of Manipulator Pushing Operations[J/OL]. The International Journal of Robotics Research, 1986, 5(3): 53-71. eprint: https://doi.org/10.1177/027836498600500303. https://doi.org/10.1177/027836498600500303. DOI: 10.1177/027836498600500303.

[19] KRIVIC S, UGUR E, PIATER J. A robust pushing skill for object delivery between obstacles[C]//2016 IEEE International Conference on Automation Science and Engineering (CASE). [S.l. : s.n.], 2016: 1184-1189.

[20] COSGUN A, HERMANS T, EMELI V, et al. Push planning for object placement on cluttered table surfaces[C]//2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. [S.l. : s.n.], 2011: 4627-4632.

[21] HANAFUSA H, ASADA H. A Robot Hand with Elastic Fingers and Its Application to Assembly Process[J]. IFAC Proceedings Volumes, 1977, 10: 127-138. DOI: 10.1016/S1474-6670(17)66592-3.

[22] WEISZ J, ALLEN P K. Pose error robust grasping from contact wrench space metrics[C]//2012 IEEE International Conference on Robotics and Automation. [S.l. : s.n.], 2012: 557-562.

[23] RODRIGUEZ A, MASON M T, FERRY S. From caging to grasping[J/OL]. The International Journal of Robotics Research, 2012, 31(7): 886-900. eprint: https://doi.org/10.1177/0278364912442972. https://doi.org/10.1177/0278364912442972. DOI: 10.1177/0278364912442972.

[24] GOLDFEDER C, CIOCARLIE M, Hao Dang, et al. The Columbia grasp database[C]//2009 IEEE International Conference on Robotics and Automation. [S.l. : s.n.], 2009: 1710-1716.

[25] XIANG Y, SCHMIDT T, NARAYANAN V, et al. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes[J/OL]. CoRR, 2017, abs/1711.00199. arXiv: 1711.00199. http://arxiv.org/abs/1711.00199.

[26] BILLINGS G, JOHNSON-ROBERSON M. SilhoNet: An RGB Method for 3D Object Pose Estimation and Grasp Planning[J/OL]. CoRR, 2018, abs/1809.06893. arXiv: 1809.06893. http://arxiv.org/abs/1809.06893.

[27] Yun Jiang, MOSESON S, SAXENA A. Efficient grasping from RGBD images: Learning using a new rectangle representation[C]//2011 IEEE International Conference on Robotics and Automation. [S.l. : s.n.], 2011: 3304-3311.

[28] REDMON J, ANGELOVA A. Real-Time Grasp Detection Using Convolutional Neural Networks[J/OL]. CoRR, 2014, abs/1412.3128. arXiv: 1412.3128. http://arxiv.org/abs/1412.3128.

[29] DETRY R, EK C H, MADRY M, et al. Learning a Dictionary of Prototypical Grasp-predicting Parts from Grasping Experience[C]//. [S.l. : s.n.], 2013. DOI: 10.1109/ICRA.2013.6630635.

[30] FREESE M, SINGH S, OZAKI F, et al. Virtual Robot Experimentation Platform V-REP: A Versatile 3D Robot Simulator[C]//: vol. 6472. [S.l. : s.n.], 2010: 51-62. DOI: 10.1007/978-3-642-17319-6_8.

[31] KOENIG N, HOWARD A. Design and use paradigms for Gazebo, an open-source multi-robot simulator[C]//2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566): vol. 3. [S.l. : s.n.], 2004: 2149-2154 vol.3.

[32] MICHEL O. Webots: Symbiosis Between Virtual and Real Mobile Robots[C]//VW '98: Proceedings of the First International Conference on Virtual Worlds. Berlin, Heidelberg: Springer-Verlag, 1998: 254-263.

[33] YAN C, MISRA D K, BENNETT A, et al. CHALET: Cornell House Agent Learning Environment[J/OL]. CoRR, 2018, abs/1801.07357. arXiv: 1801.07357. http://arxiv.org/abs/1801.07357.

[34] BRODEUR S, PEREZ E, ANAND A, et al. HoME: a Household Multimodal Environment[J/OL]. CoRR, 2017, abs/1711.11017. arXiv: 1711.11017. http://arxiv.org/abs/1711.11017.

[35] KOLVE E, MOTTAGHI R, GORDON D, et al. AI2-THOR: An Interactive 3D Environment for Visual AI[J/OL]. CoRR, 2017, abs/1712.05474. arXiv: 1712.05474. http://arxiv.org/abs/1712.05474.

[36] SAVVA M, CHANG A X, DOSOVITSKIY A, et al. MINOS: Multimodal Indoor Simulator for Navigation in Complex Environments[J/OL]. CoRR, 2017, abs/1712.03931. arXiv: 1712.03931. http://arxiv.org/abs/1712.03931.

[37] WHITTED T. An Improved Illumination Model for Shaded Display[J/OL]. Commun. ACM, 1980, 23(6): 343-349. https://doi.org/10.1145/358876.358882. DOI: 10.1145/358876.358882.

[38] KAJIYA J T. The Rendering Equation[C/OL]//SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques. New York, NY, USA: Association for Computing Machinery, 1986: 143-150. https://doi.org/10.1145/15922.15902. DOI: 10.1145/15922.15902.

致　谢

在这里我要感谢许多人，没有他们我不能顺利的完成本篇论文，也不能经历如此丰富多彩的大学四年。

感谢卢策吾老师对于本课题的指导！

感谢俞勇老师、严骏驰老师、张弛豪老师、卢宏涛老师、李超老师平日的指点！

感谢蒋健巍学长、方浩树学长的鼎力相助！他们在本工作的编程实践方面给予了我许多宝贵的建议。

感谢虚幻引擎社区、英伟达社区的众多开发者和用户们！他们坚实的工作和丰富的经验为本工作奠定了基础。

感谢父母、老师、同学们的陪伴与帮助！没有他们我不能度过这忙碌而充实的本科生涯。

感谢致远学院、上海交通大学对我的栽培！

CYBERPANDA：A UNIVERSAL ROBOTIC ARM SIMULATOR TOWARDS PHOTOREALISTIC VISUAL PERCEPTION

In this thesis, we propose a novel universal robotic arm simulator focusing on the indoor environment robotic vision task, which is called CyberPanda. The purpose of our framework is to give researchers who devote efforts to robotic vision problems easy access to the synthetic generated data and bridge the gap between the virtual environment and the real world.

Having been thoroughly investigated for decades, some of the robotic vision tasks still remain challenging, likewise pulling and grasping tasks. The past few years have witnessed the popularity of the deep neural network, which leads to the huge advancement of the solution to these robotic vision problems. However, the fantastic performance of and the giant leap made by the deep neural networks highly depend on the big scale dataset with full annotations. On one hand, although various kinds of dataset have been built up for different tasks, there exist many individual researchers whose diverse demands can not be satisfied with limited kinds of dataset. One the other hand, it is too expensive for individual researchers to afford the cost to set up their own dataset to launch their projects. Sometimes their requirements may even not be met in the real world. In such a case, the researchers in the field of robotic vision increasingly rely on the simulators to generate synthetic data. Not only the simulator greatly simplify and shorten the process of building up the dataset, but also is it able to produce the annotations and labels automatically instead of manually collecting the data. The simulator saves researchers from tremendous cost and heavy labor, partially addressing the needs of researchers to customize their own specific dataset. Therefore, the development of a simulator is of great significance to the field of computer vision, especially the area of robotic vision. According to this, more and more attention has been paid to the simulators since several years ago.

Although the synthetic data produced by the simulator is of lower price and is more available to researchers, from which the development of deep neural networks benefits a lot, the synthetic still has fatal disadvantages compared to the genuine data. According to the knowledge of the prior research, it is well-known that the deep neural networks which are trained with the synthetic data can not be deployed in the real world as successful as in the virtual environment. That is, the performance of the algorithm in the real world can not be mentioned in the same breath to the performance during its training time, even overwhelmed by a huge gap between them. For example, the success rate of a grasping algorithm that is trained in the simulator will likely encounter a considerable drop when transferred to the real world. This kind of difference in performance is usually caused by the divergence between the data distribution of the synthetic world and the real world. The special divergence between them is named reality gap by the researchers. Many reasons play a role in the occurrence and formation of the reality gap, like the invariant simulator environment, the limited kinds of object materials and textures, the rigid position and orientation of the camera, the inaccurate global illumination calculation, and the imprecise rendering results. In summary, these reasons listed above can be concluded as an insufficient and inaccurate simulation of the real world. Sometimes,

the divergence caused by the reality gap is so considerable that deployed the trained model in reality is even more difficult than training the model in the virtual environment. Subsequently, how to address the problem given rise to by the reality gap becomes another valuable and meaningful task. Whether we can solve this problem or not will even determine the future evolving path of all these simulators.

Aiming to mitigate or even eliminate the negative influence brought by the reality gap, many researchers have paid their own efforts. Among all these numerous tries, two kinds of methods have been proved to be effective: extreme realism and domain randomization. Extreme realism is a set of algorithms whose core idea is trying to make the process of simulation as close as possible to the real environment that the trained models or robots are likely to be deployed in. The set of extreme realism algorithm contains various and diverse techniques. For instance, the famous one of them is photorealistic rendering. Photorealistic rendering does not merely refer to make the pipeline of rendering produce the realistic visual perception, likewise realistic geometry, global illumination, materials, textures, etc. Photorealistic rendering also takes the noise of visual sensors into consideration, including simulating the specific noise and the different distortion of the camera lens and other pose effects like lens flare. Apart from the photorealistic rendering, extreme realism algorithms may also refer to the high-fidelity physics simulation. Commonly, when most of the traditional physics engines deal with the contact force, they may rely on the earlier spring-damper approach which is obviously outdated or even just leaves out the contact dynamics. Some physics engines adopt this strategy because this kind of approximation can lead to many efficient and accurate recursive algorithms in joint coordinates. On the contrary, the high-fidelity physics focus on simulating the dynamics and contact force as accurate as possible instead of sacrificing the simulation quality for real-time efficiency. Domain randomization also represents a series of techniques. The novel idea of domain randomization algorithms is to expose the model to a giant range of elements on the training stage instead of just training the model in a single synthetic world. The strategy of domain randomization is to randomize as many as possible elements in the simulation environments to increase to variation of the simulation process, expecting the robot to encounter much more different situations than simulating in the static data. For example, the researchers may put the camera on a random position and pose to a random orientation, or they may randomly substitute the materials and the textures of the object. Besides, they may also randomly generate the skylight to change the illumination condition of the whole environment. All these varied means have the same purpose. That is to drive the model to generalize more during the training time and improve its ability to cognate different situations or even learn to handle with the unseen situations.

In our framework CyberPanda, we combine the strategy of extreme realism and domain randomization, trying to bridge or narrow the reality gap between the simulator environment and the real world, thus making the training, the evaluation, and the testing of the robotic vision algorithms more convenient. Our platform is built on the base of the most powerful and outstanding real-time 3D creation software Unreal Engine 4. Unreal Engine 4 is a versatile development tool and its applications range from design visualizations to high-quality games across different platforms. CyberPanda employs the rendering pipeline of the Unreal Engine 4 as our rendering core, thus providing our simulator with fantastic and high-quality visual perception. Besides, CyberPanda utilizes the Bullet3 Physics Engine as our physics core to simulate the gravity, motion, collision, and contact in

the virtual environment. Unlike other traditional physics engine who leave out the contact dynamics, Bullet3 found the contact force by solving an optimization problem at each time step. It also includes the inverse dynamics calculation module which can robustly locate the joint of the robots. The Bullet3 not only offers CyberPanda accurate and efficient dynamics simulation, but also empowers CyberPanda to conduct continuous interaction between robots and non-static objects instead of remaining on the level of discrete interaction. Apart from all the above, with the help of google remote procedure call framework, CyberPanda is able to provide our users with a Python application programming interface. According to this feature, users can launch CyberPanda with a simple Python script without too much domain knowledge in computer graphics. Therefore, CyberPanda is quite friendly to deep learning researchers.

The contribution of CyberPanda lies in the following two aspects. For one thing, the framework provides a simple Python API which can help deep learning researchers easily generate synthetic data. For another, the framework also produces more accurate and photorealistic rendering results compared to some of the other simulators, which may narrow the gap between the simulator environment and the real world.

The thesis is organized in the following three parts. In the first chapter we make a thorough introduction to the background of the simulator and related robotic vision tasks. After that, we make an overall discussion of related work, focusing on their advantages and drawbacks. Apart from other simulators, we will also make a brief introduction to the tools we utilized in the development of CyberPanda. In chapter three, we comprehensively introduce our platform's architecture, describe the implementation of each module, and demonstrate the ability of CyberPanda. In this chapter, we will focus on the remote procedure call system, the rendering pipeline, and the physics engine. In the final chapter, we draw a conclusion of the whole thesis and point out the potential improvement of CyberPanda.