Vision Robotics Combat

视觉机器人实战

ElegantLaTeX Program

Victory won't come to us unless we go to it.

作者：周佳立 刘震 谢涛 李世龙 谢雁楠

时间：April 10, 2024

邮箱: elegantlatex2e@gmail.com

版本: 3.04

目　　录

1 概述 **1**

1.1 机器人的产生与发展 1

1.1.1 机器人的产生 1

1.1.2 机器人的发展 4

1.2 机器人的分类与应用 11

1.2.1 机器人分类 11

1.2.2 工业机器人 13

1.2.3 服务机器人 16

1.3 工业机器人应用 18

1.3.1 技术发展与产品应用 18

1.3.2 主要生产企业 21

2 快速入门 **26**

2.1 开箱 26

2.2 实战连线 26

2.2.1 电机线缆的连接（控制柜-机械手） 26

2.2.2 数据线的连接（控制柜-机械手） 28

2.2.3 分离式安全接口 X11 的连接（控制柜自连） 28

2.2.4 示教器线缆的连接（控制柜-smartPAD） 29

2.2.5 电源线缆的连接（控制柜-外接 220V 电源） 29

2.3 启动机器人控制系统（控制柜） 30

2.4 用示教器手动控制机器人 33

3 KUKA 机器人的基本操作 **35**

3.1 KUKA 机器人的结构 35

3.2 smartPAD 示教器 35

3.2.1 smartPAD 结构 36

3.2.2 smartPAD 正面 36

3.2.3 smartPAD 背面 37

3.3 KUKA 机器人基本操作与设定 38

3.3.1 KUKA 机器人的启动/关机和设定 38

3.3.2 smartPAD 示教器的热插拔 47

3.3.3 smartHMI 操作界面与信息显示 48
3.3.4 主菜单、显示语言、屏幕截图与用户等级 50
3.3.5 操作模式与安全防护 53
3.3.6 状态栏图标及操作 55
3.3.7 信息的在线文献和帮助 60
3.3.8 手动运行机器人 62

4 KUKA 机器人的零点标定 74
4.1 引言 74
4.1.1 零点标定的目的 74
4.1.2 零点标定方法 75
4.1.3 什么时候进行零点标定 75
4.1.4 哪些情况下必须进行零点标定 76
4.2 使用 MEMD 进行零点标定 77
4.2.1 利用零点标定标记将轴移入预零点标定位置 77
4.2.2 包含 SEMD 和 MEMD 的零点标定组件 80
4.2.3 零点标定 LED 指示灯 82
4.2.4 首次进行零点校正（用 MEMD） 85
4.3 手动删除轴的零点 89
4.3.1 为什么需要删除轴的零点 89
4.3.2 删除轴零点步骤 90
4.4 更改软件限位开关 92
4.4.1 为什么需要改变软件限位开关 92
4.4.2 改变软件限位开关操作方法 93

5 KUKA 机器人坐标系的测量 98
5.1 测量工具坐标系 100
5.1.1 测量工具坐标系的方法 100
5.1.2 工具坐标系测量的优点 101
5.1.3 工具的介绍 101
5.1.4 测量工具坐标系方法及操作步骤 104
5.2 测量基坐标 121
5.2.1 测量基坐标系的优点及方法 122
5.2.2 测量基坐标系 3 点法 124
5.2.3 工具、基坐标系改名 128

6 机器人的编程 130
6.1 KUKA 机器人编程基础 130

6.1.1 KUKA 导航器 130
6.1.2 新建文件夹 132
6.1.3 新建程序 133
6.1.4 文件或文件夹重命名 136
6.1.5 打开或选定程序 137
6.1.6 显示折叠夹 143
6.2 机器人的运动编程 144
6.2.1 运动编程概述 144
6.2.2 使用联机表格进行运动编程 146
6.3 使用联机表格进行逻辑编程 150
6.3.1 给等待时间编程-WAIT 150
6.3.2 对与信号有关的等待功能进行编程-WAITFOR 152
6.3.3 设置数字输出端-OUT 153
6.3.4 更改逻辑指令 155
6.4 执行程序 155
6.4.1 选择程序运动方式 155
6.4.2 语句指针 157
6.4.3 设定程序倍率 158
6.4.4 机器人解释器状态显示 159
6.4.5 启动正向运动程序（手动） 161
6.4.6 启动正向运行程序（自动） 163
6.4.7 语句选择 163
6.4.8 复位程序 165

7 视觉机器人积木抓取实操 166
7.1 手眼标定实操 166
7.1.1 KUKA 示教器切换中文 166
7.1.2 获取标定板坐标系 167
7.1.3 获取照片 179
7.1.4 matlab 工具箱相机标定 181
7.2 积木抓取模块硬件搭建 188
7.3 VisionMaster——机械臂与视觉控制器通信桥梁 199
7.4 视觉机械臂实战——通信 199
7.4.1 机械臂信息传输到视觉控制器 199
7.4.2 视觉控制器信息传输到机械臂 212
7.4.3 其他软件与 VisionMaster 的交互 216
7.5 VisionMaster——图像定位识别 221

7.5.1 VisionMaster——相机配置 221
7.5.2 VisionMaster——设置全局变量 226
7.5.3 VisionMaster——定位 227
7.6 VisionMaster 总流程搭建 230
7.6.1 积木识别定位模块 232
7.6.2 高处识别定位小球模块 237
7.6.3 低处识别定位小球模块 243
7.7 KUKA 运动控制搭建积木 244
7.8 上位机搭建 253
7.9 展示成果 253

8 机器视觉简介 254
8.1 机器视觉——模拟人眼获取光学信号 254
8.2 视觉机械臂的应用场景 256
8.3 摄像机几何 257
8.4 小孔成像模型 259
8.5 相机参数的定义 262
8.6 摄像机外参 266
8.7 摄像机标定 267
8.8 常见标定法简介 269
8.8.1 求解内参矩阵与外参矩阵的积 270
8.8.2 求解内参矩阵 271
8.8.3 求解外参矩阵 274
8.8.4 相机标定的畸变参数 274

9 机器视觉抓取实战 277
9.1 积木搭建 278
9.1.1 术语说明 278
9.1.2 搭建入门 281
9.1.3 积木分类 281
9.2 机械臂手眼标定 282
9.2.1 机械臂手眼标定步骤 284
9.2.2 旋转平移矩阵的检验 288
9.3 本次实验代码 294
9.4 机械臂-视觉系统实战之三维重建 300
9.5 基于 Context capture 三维重建 303
9.6 Context capture 软件 SDK 应用 307
9.6.1 代码解析 311

10 KUKA 机器人的视觉积木抓取实战 326
10.1 视觉系统构成 326
10.2 视觉系统软件部分——VisionMaster 328
10.3 VisionMaster 功能模块 329
10.3.1 VisionMaster——图像采集 329
10.3.2 VisionMaster——全局变量 333
10.3.3 VisionMaster——定位 334
10.4 轨道积木搭建难点 343
10.5 视觉机械臂实战——通信 345
10.5.1 机械臂信息传输到视觉控制器 345
10.5.2 视觉控制器信息传输到机械臂 364
10.5.3 其他软件与 VisionMaster 的交互 370
10.6 视觉机械臂积木抓取项目总流程 377
10.6.1 积木识别定位模块 379
10.6.2 高处识别定位小球模块 385
10.6.3 低处识别定位小球模块 391
10.7 KUKA 机器人代码详解 394
10.7.1 坐标定义 394
10.7.2 抓取积木 401
10.7.3 将小球放置顶部积木中心滚下 410
10.7.4 将积木放回原位 414
10.7.5 将弹珠全部捡起 416
10.8 上位机界面的制作 420
10.8.1 VisionMaster SDK 介绍 420
10.8.2 开发过程 425

第 1 章 概述

1.1 机器人的产生与发展

1.1.1 机器人的产生

1. 概念的出现

1959 年，世界上第一台工业机器人（Robot）问世，由于它能够协助、代替人类完成那些重复、频繁、单调、长时间的工作，或可进行危险、恶劣环境下的作业，因此其发展较为迅速。随着人们对机器人研究的不断深入，机器人技术逐渐发展成为一门新的综合性学科——机器人学（Robotics）。机器人技术与 PLC 技术、CAD 技术被称为现代工业自动化的三大支柱。

机器人（Robot）一词源自于捷克著名剧作家 Karel C˘apek（卡雷尔·恰佩克）1921 年创作的剧本 *Rossumovi univerza´lni´ roboti*（R.U.R，罗萨姆的万能机器人），由于 R.U.R 剧中的人造机器被取名为 Robota（捷克语，即奴隶、苦力)，因此，英文 Robot 一词开始代表机器人。

机器人概念一经出现，就引起了科幻小说作家的广泛关注。自 20 世纪 20 年代起，机器人就成了很多科幻小说、电影的重要角色，如《星球大战》中的 C-3PO 等。科幻小说作家的想象力是无限的。为了预防机器人给人类带来灾难，1942 年，美国科幻小说作家艾萨克·阿西莫夫（Isaac Asimov) 在 I，Robot 的第 4 个短篇“Runaround”中，首次提出了“机器人三原则”，它被称为“现代机器人学的基石”，这也是“机器人学（Robotics）”这个名词在人类历史上的首度亮相。

机器人三原则的主要内容如下。

原则 1：机器人不得伤害人类，或看到人类受到伤害而袖手旁观.

原则 2：机器人必须服从人类的命令，除非这条命令与第一条相矛盾。

原则 3：机器人必须保护自己，除非这种保护与以上两条相矛盾。

到了 1985 年，艾萨克·阿西莫夫在机器人系列的最后作品 *Robots and Empire* 中，又补充了凌驾于“机器人三原则”之上的“原则 0”。

原则 0：机器人必须保护人类的整体利益不受伤害，其他 3 条原则都必须在这一前提下才能成立。

继阿西莫夫之后，其他科幻作家也不断提出了对“机器人三原则”的补充、修正意见，但是，这些绝大多数是科幻小说作家对想象中机器人所施加的限制；实际上，“人类整体利益”等概念本身就是模糊的，甚至连人类自己都搞不明白，更不要说机器人了。因此，目前人类的认识和科学技术，实际上还远未达到制造科幻片

中的机器人的水平，制造出具有类似人类智慧、感情、思维的机器人，仍属于科学家的梦想和追求。

2. **机器人的产生**

现代机器人的研究起源于 20 世纪中期的美国，它从工业机器人的研究开始。第二次世界大战期间（1939-1945），由于军事、核工业的发展，原子能实验室需要机械代替人类进行放射性物质的处理。为此，美国的 Argonne National Laboratory（阿尔贡国家实验室）开发了一种遥控机械手（Teleoperator）。接着，在 1947 年，又开发出了一种伺服控制的主-从机械手（Master Slave Manipulator），这些都是工业机器人的雏形。

图 1.1: 工业机器人 Unimate

工业机器人的概念由美国发明家 George Devol（乔治 • 德沃尔）最早提出，他在 1954 年申请了专利，并在 1961 年获得授权。1957 年，美国著名的机器人专家 Joseph F.Engelberger（约瑟夫 • 恩格尔伯格）建立了 Unimation 公司，并利用 George Devol 的专利，于 1959 年研制出了世界上第一台真正意义上的工业机器人 Unimate，如图1.1所示，开创了机器人发展的新纪元。

Joseph Engelberger 对世界工业机器人的发展作出了杰出的贡献，被人们称为“工业机器人之父”。1983 年，就在工业机器人销售日渐增长的情况下，他又毅然地将 Unimation 公司出让给了美国 Westinghouse Electric Corporation（西屋电气公司，又译威斯汀豪斯），并创建了 TRC 公司，前瞻性地开始了服务机器人的研发工作。

从 1968 年起，Unimation 公司先后将机器人的制造技术转让给了日本 KAWASAKI（川崎）和英国 GKN 公司，机器人开始在日本和欧洲得到了快速发展。据有关方面的统计，目前世界上至少有 48 个国家在研发机器人，其中 25 个国家已在进行智能机器人研发工作，美国、日本、德国、法国等都是机器人的研发和制造大国，无论在基础研究还是产品研发、制造方面都居世界领先水平。

3. **国际标准化组织**

随着机器人技术的快速发展，在发达国家，机器人及其零部件的生产已逐步形

成产业，为了能够宣传、规范和引导机器人产业的发展，世界各国相继成立了相应的行业协会。目前，世界机器人生产与使用国的机器人行业协会主要有以下几个。

(1) International Federation of Robotics（IFR，国际机器人联合会）：该联合会成立于 1987 年，目前已有来自二十多个国家的九十多个成员。它是世界公认的机器人行业代表性组织，连接了全球范围内的机器人领域，已被联合国列为非政府正式组织。

(2) Japan Robot Association（JRA，日本机器人协会）：该协会原名为 Japan Industrial Robot Association（JIRA，日本工业机器人协会），也是全世界最早的机器人行业协会。JIRA 成立于 1971 年 3 月，最初称“工业机器人恳谈会”；1973 年 10 月成为正式法人团体；为了适应非工业机器人，该协会在 1994 年 6 月更名为 Japan Robot Association（JRA）。

(3) Robotics Industries Association（RIA，机器人工业协会）：该协会成立于 1974 年，是美国机器人行业的专业协会，旨在为北美的机器人工业服务。

(4) Verband Deutscher Maschinen Und Anlagebau（VDMA，德国机械设备制造业联合会）：VDMA 是拥有 3600 多家企业会员、500 余名专家的大型行业协会，其下设有 36 个专业协会和一系列跨专业的技术论坛、委员会及工作组，是欧洲目前最大的工业联合会，以及工业投资品领域中最大、最重要的组织机构。自 2000 年起，VDMA 设立了专业协会 Deutschen Gesellschaft Association fu¨r Robotik（DGR，德国机器人协会），专门进行机器人产业的规划和发展等相关工作。

(5) French Research Group in Robotics（FRGR，法国机器人研究协会）：该协会原名 Association Frencaise de Robotique Industrielle（AFRI，法国工业机器人协会），后来随着服务机器人的发展，在 2007 年更为现名。

(6) Korea Association of Robot Industry（KAR，韩国机器人工业协会）：亚洲较早的机器人协会之一，成立于 1999 年；2003 年，以 Korea Advanced Intelligent Robot Association（KAIRA，韩国先进智能机器人协会）成立了另一个与机器人产业有关的商业协会。为了提高机器人政策的效率和统一性，两家公司进行了横向合并，KAR 最终于 2008 年 5 月成立，从那时起，KAR 一直是代表韩国机器人产业的唯一商业组织。

4. **机器人的定义**

由于机器人的应用领域众多、发展速度快，加上它又涉及人类的有关概念，因此，对于机器人，世界各国标准化机构，甚至同一国家的不同标准化机构，至今尚未形成一个统一、准确、世界所公认的严格定义。

例如，欧美国家一般认为，机器人是一种“由计算机控制、可通过编程改变动作的多功能、自动化机械”。而日本作为机器人的生产大国，则将机器人分为“能够执行人体上肢（手和臂）类似动作”的工业机器人和“具有感觉和识别能力，并能够控制自身行为”的智能机器人两大类。

客观地说，欧美国家的机器人定义侧重其控制方式和功能，其定义和现行的工

业机器人定义较接近；而日本的机器人定义，关注的是机器人的结构和行为特性，且已经考虑到了现代智能机器人的发展需要，其定义更为准确。

作为参考，目前在相关资料中使用较多的机器人定义主要有以下几种。

(1) International Organization for Standardization（ISO，国际标准化组织）定义：机器人是一种“自动的、位置可控的、具有编程能力的多功能机械手，这种机械手具有几个轴，能够借助可编程序操作来处理各种材料、零件、工具和专用装置，执行各种任务”。

(2) Japan Robot Association（JRA，日本机器人协会）将机器人分为了工业机器人和智能机器人两大类，工业机器人是一种“能够执行人体上肢（手和臂）类似动作的多功能机器”；智能机器人是一种“具有感觉和识别能力，并能够控制自身行为的机器”。

(3) NBS（美国国家标准局）定义：机器人是一种“能够进行编程，并在自动控制下执行某些操作和移动作业任务的机械装置”。

(4) Robotics Industries Association（RIA，美国机器人工业协会）定义：机器人是一种“用于移动各种材料、零件、工具或专用装置的，通过程序动作来执行各种任务，并具有编程能力的多功能操作机”。

(5) 我国 GB/T12643 中定义：工业机器人是一种“能够自动定位控制，可重复编程的，多功能的、多自由度的操作机，能搬运材料、零件或操持工具，用于完成各种作业”。

以上标准化机构及专门组织对机器人的定义，都是在特定时期所得出的结论，多偏重于工业机器人。但未来的科学技术是无限开放的，当代智能机器人无论在外观，还是功能、智能化程度等方面，都已超出了传统工业机器人的范畴。机器人正在源源不断地向人类活动的各个领域渗透，它所涵盖的内容越来越丰富，其应用领域和发展空间正在不断延伸和扩大，这也是机器人与其他自动化设备的重要区别。

可以想象，未来的机器人不但可接受人类指挥、运行预先编制的程序，而且也可根据人工智能技术所制定的原则纲领，选择自身的行动；甚至可能像科幻片所描述的那样，脱离人们的意志而自主行动。

1.1.2 机器人的发展

1. **技术发展水平**

机器人最早用于工业领域，它主要用来协助人类完成重复、频繁、单调、长时间的工作，或进行高温、粉尘、有毒、有辐射、易燃、易爆等恶劣、危险环境下的作业。但是，随着社会进步、科学技术发展和智能化技术研究的深入，各式各样具有感知、决策、行动和交互能力，可适应不同领域特殊要求的智能机器人相继被研发，机器人已开始进入人们生产、生活的各个领域，并在某些领域逐步取代人类独立进行相关作业。

根据机器人现有的技术水平，人们一般将机器人产品分为以下 3 代。

****

添加一张自己拿着示教器操作机器人的图片

****

(1) 第一代机器人。第一代机器人一般是指能通过离线编程或示教操作生成程序，并再现动作的机器人。第一代机器人所使用的技术和数控机床十分相似，它既可通过离线编制的程序控制机器人的运动，也可通过手动示教操作（数控机床称为 Teach in 操作），记录运动过程并生成程序，并进行再现运行。

第一代机器人的全部行为完全由人控制，它没有分析和推理能力，不能改变程序动作，无智能性，其控制以示教、再现为主，故又称示教再现机器人。第一代机器人现已实用和普及，大多数工业机器人都属于第一代机器人，如图 1.1-2 所示。

探测机器人 LEMUR 3

人机协同作业机器人

图 1.2: 第二代机器人

(2) 第二代机器人。第二代机器人装备有一定数量的传感器，它能获取作业环境、操作对象等的简单信息，并通过计算机的分析与处理，作出简单的推理，并适当调整自身的动作和行为。

例如，图1.2(a)所示的探测机器人可通过所安装的摄像头及视觉传感系统来识别图像，判断和规划探测车的运动轨迹，它对外部环境具有了一定的适应能力。图1.2(b)所示的人机协同作业机器人安装有触觉传感系统，以防止人体碰撞，它可取消第一代机器人作业区间的安全栅栏，实现安全的人机协同作业。

第二代机器人已具备一定的感知和简单推理等能力，有一定程度上的智能，故又称感知机器人或低级智能机器人，当前使用的大多数服务机器人已经具备第二代机器人的特征。

(3) 第三代机器人。第三代机器人应具有高度的自适应能力，它有多种感知机能，可通过复杂的推理，作出判断和决策，自主决定机器人的行为，具有相当程度的智能，故称为智能机器人。第三代机器人目前主要用于家庭、个人服务及军事、

航天等行业，总体尚处于试验和研究阶段，目前还只有美国、日本、德国等少数发达国家能掌握和应用。

例如，美国 Tesla（特斯拉）公司研发的擎天柱（Optimus）机器人，如图1.3(a)所示，第二代“擎天柱”可以轻松准确地抓取和放下鸡蛋，整个过程比较连贯平稳，左手转右手的整个过程也颇为连贯，展示出其双手精巧的操控能力；同时，“擎天柱”采用与汽车一致的强大计算机视觉，拥有处理视觉数据、做出行动决策、支持通信交流的“大脑”，并搭载与特斯拉车辆同源的 FSD 电脑以及 Autopilot 相关神经网络技术。英国 Engineered Arts 公司制造的人形机器人 Ameca，如图1.3(b)所示，2024 年 2 月，在 Engineered Arts 最新视频中，机器人 Ameca 再次进化，由 GPT-4 提供支持，新增了视觉感知能力与声音克隆能力，能够对各种问题做出实时反应，它可以用马斯克的语气讲述火星科幻故事。

“擎天柱”机器人

Ameca 机器人

图 1.3: 第三代机器人

2. **主要生产国及产品水平**

机器人自问世以来，得到了世界各国的高度重视，美国、日本和德国为机器人研究、制造和应用大国，英国、法国、意大利、瑞士等国的机器人研发水平也居世界前列。目前，世界主要机器人生产制造国的研发、应用情况如下。

(1) 美国

美国是机器人的发源地，其机器人研究领域广泛、产品技术先进，机器人的研究实力和产品水平均居世界前列，Adept Technology、American Robot、Emerson Industrial Automation、S-T Robotics、iRobot、Remotec 等都是美国著名的机器人生产企业。

美国的机器人研究从最初的工业机器人开始，目前已更多地转向军用、医疗、家用服务及军事、场地等高层次智能机器人的研发。据统计，美国的智能机器人占据了全球约 60% 的市场，iRobot、Remotec 等都是全球著名的服务机器人生产企业。

美国的军事机器人（Military Robot）更是遥遥领先于其他国家，无论在基础技术研究、系统开发、生产配套方面，还是在技术转化、实战应用方面等都具有强大的优势，其产品研发与应用已涵盖陆、海、空、天等诸多兵种，美国是目前全世界

唯一具有综合开发、试验和实战应用能力的国家。

美国现有的军事机器人产品包括无人驾驶飞行器、无人地面车、机器人武装战车及多功能后勤保障机器人、机器人战士等。

图1.4(a)所示为 Boston Dynamics（波士顿动力）研制的多功能后勤保障机器人。其中，BigDog（大狗）系列机器人的军用产品 Legged Squad Support Systems（LS3），重达 1250 磅（约 570kg），这种机械狗的体型与大型犬相当，能够在战场上发挥非常重要的作用；它在搭载 400 磅（约 181kg）重物情况下，连续行走 20 英里（约 32km），并能穿过复杂地形、应答士官指令，为士兵运送弹药、食物和其他物品；它不但能够行走和奔跑，而且还可跨越一定高度的障碍物。

图1.4(b)所示为 Sand Flea（沙蚤）机器人，一个小型机器人，高 15 厘米，重 5 公斤，有 4 个轮子和一个强大的跳跃腿。它在平坦的地形上像遥控车一样行驶，但可以跳到空中 10 米处跳过障碍物，这足够可以跳过一个复合墙、一个房子的屋顶、一组楼梯或二层楼的窗户，而且它足够精确。图1.4(c)所示为 WildCat（野猫），是一种有四条腿的机器人，它能在各种地形上，以超过 25km/h 的速度跳跃和飞奔。

BigDog-LS3

Sand Flea

WildCat

Atlas

图 1.4: Boston Dynamics 研发的军事机器人

此外，Boston Dynamics 还研制出了类似科幻片中的“机器人战士”——人形机器人。如 Atlas（阿特拉斯）机器人专门用于移动操纵，非常擅长在广泛的地形上行走，包括雪地；它的四肢共拥有 28 个自由度，能够直立行走、攀爬、自动调整重心，具有精细动作技能的能力，其灵活性已接近于人类；它是电动和液压驱动，使用身体和腿部的传感器进行平衡，并在其头部使用光学雷达和立体传感器，以避免障碍物，评估地形，帮助导航和操纵对象。如图1.4(d)所示，最新版本的 Atlas 机器人大约 175 cm 高，82kg 重。

Spot

Handle

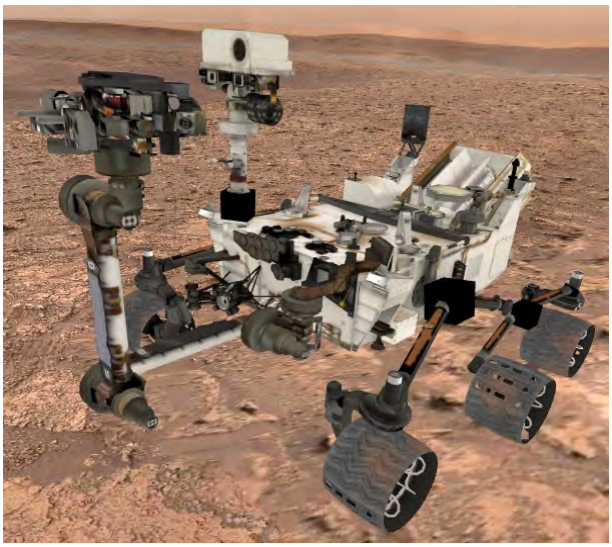
图 1.5: Boston Dynamics 研发的其它工业机器人

除军事机器人外，Boston Dynamics（波士顿动力）研制的 Spot、Handle 等机器人正与各行各业的数字化转型紧密结合。图1.5(a)所示为 Spot 机器人，重量仅 32.7kg，拥有自由奔跑、自主导航、后空翻、拉车等技能，还知道如何转动门把手来开门和用前腿别住门；Spot 通过连接各个资产和整个工厂运营，解锁了制造业的数字化转型。图1.5(b)所示为 Handle 机器人，是一款轮腿物流机器人，它的两个轮子上方安装有大型吸盘式抓器，可用于自动装载和卸载仓库中的货物箱，能搬动约 13.6 公斤的箱子，充电一次可行驶 24 公里左右；在大型仓库中，Handle 能够代替人类来回跑动、抓取货物箱体并整齐堆叠。

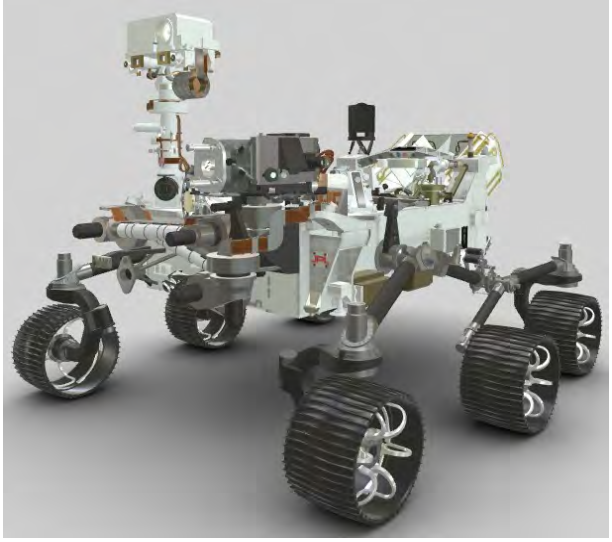
美国的场地机器人（Field Robots）研究水平同样令其他各国望尘莫及，其研究产品遍及陆地、水下，并已经被用于月球、火星等天体的探测。

早在 1967 年，National Aeronautics and Space Administration（NASA，美国国家航空航天局）所发射的“海盗”号火星探测器已着陆火星，并对火星土壤等进行了采集和分析，以寻找生命迹象；同年，还发射了“观察者”3 号月球探测器，该探测器也对月球土壤进行了分析和处理。在 2003 年，NASA 又接连发射了 Spirit MER-A（勇气号）和 Opportunity（机遇号）两个火星探测器，这两个火星探测器于 2004 年 1 月先后着陆火星表面，它们可在地面的遥控下，在火星上自由行走，通过它们对火星岩石和土壤的分析，科学家们收集到了表明火星上曾经有水流动的强有

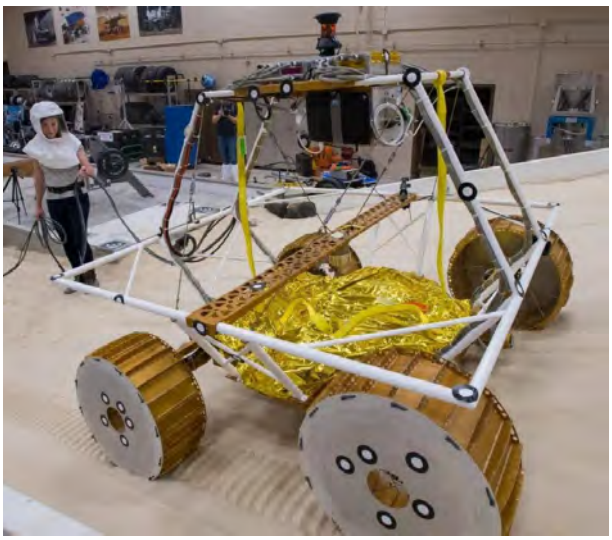
力证据，发现了形成于酸性湖泊的岩石、陨石等。2011 年 11 月，NASA 成功发射了 Curiosity（好奇号）核动力火星探测器，该探测器于 2012 年 8 月 6 日安全着陆火星，开启了人类探寻火星生命元素的历程，Curiosity 火星车如图1.6(a)所示。2020 年 7 月 30 日，NASA 又成功发射了 Perseverance（毅力号）核动力火星探测器，该探测器于 2021 年 2 月 19 日安全着陆火星，后在火星上成功收集到第 23 个岩石样本，成为天文学家探究火星古代环境的关键钥匙，Perseverance 火星车如图1.6(b)所示。图1.6(c)所示是 NASA 最新研发的新一代月球车原型——VIPER(极地探索漫游者) 月球车，预计于 2024 年 11 月 10 日发射前往月球南极，为未来“阿尔忒弥斯”任务寻找水、冰和其它物质。



Curiosity 火星车



Perseverance 火星车



VIPER 月球车

图 1.6: 美国的场地机器人

(2) 日本

日本是目前全球产量最大的机器人研发、生产和使用国，在工业机器人及家用服务、护理及医疗等智能机器人的研发上具有世界领先水平。日本在工业机器人的生产和应用居世界领先地位。20 世纪 90 年代，日本就开始普及第一代和第二代工业机器人，截至目前，它仍保持工业机器人产量世界第一的地位。据统计，日本的工业机器人产量约占全球的 50% 。

日本在工业机器人的主要零部件供给、研究等方面同样居世界领先地位，其主要零部件（精密减速机、伺服电动机、传感器等）占全球市场的 90% 以上。日本的哈默纳科公司（Harmonic Drive System，InC.）是全球最早的谐波减速器生产企业和目前全球最大、最著名的谐波减速器生产企业，其产品规格齐全，产量占全世界总量的 15% 左右。日本的 Nabtesco Corporation（纳博特斯克公司）是全球最大、技术最领先的 RV 减速器生产企业，其产品占据了全球 60% 以上的工业机器人 RV 减速器市场及日本 80% 以上的数控机床自动换刀装置（ATC）RV 减速器市场。世界著名的工业机器人绝大多数使用 Harmonic Drive System 生产的谐波减速器和 Nabtesco Corporation 生产的 RV 减速器。

日本在发展第三代智能机器人上，同样取得了举世瞩目的成就。为了攻克智能机器人的关键技术，自 2006 年起，政府每年都投入巨资用于服务机器人的研发，如 HONDA 公司 Asimo 机器人、Riken Institute 的 Robear 护理机器人、Toyota 公司的 Busboy 机器人等家用服务机器人的技术水平均居世界前列。

(3) 德国

德国的机器人研发稍晚于日本，但其发展十分迅速。在 20 世纪 70 年代中后期，德国政府在“改善劳动条件计划”中，强制规定了部分有危险、有毒、有害的工作岗位必须用机器人来代替人工，这为机器人的应用开辟了广大的市场。据 VDMA 统计，目前德国的工业机器人密度已是法国的 2 倍和英国的 4 倍以上，它是目前欧洲最大的工业机器人生产和使用国。

德国的工业机器人和军事机器人中的地面无人作战平台、水下无人航行体的研究和应用水平居世界领先地位。德国的 KUKA（库卡）、REIS（徕斯，现为 KUKA 成员）、Carl-Cloos（卡尔-克鲁斯）等都是全球著名的工业机器人生产企业；德国宇航中心、德国机器人技术商业集团、karcher 公司、Fraunhofer Institute for Manufacturing Engineering and Automatic（弗劳恩霍夫制造技术和自动化研究所) 及 STN 公司、HDW 公司等是有名的服务机器人及军事机器人研发企业。

德国在智能服务机器人的研究和应用上，同样具有世界公认的领先水平。例如，弗劳恩霍夫制造技术和自动化研究所最新研发的服务机器人 Care-O-Bot4，不但能够识别日常的生活用品，且还能听懂语音命令、看懂手势命令，按声控或手势的要求进行自我学习。

(4) 中国

我国航天航空、深海探测领域的场地机器人研究水平处于世界领先水平。从 1970 年发射第一颗人造卫星“东方红一号”、1999 年发射第一艘无人试验飞船“神

舟一号”、2013 年“嫦娥三号”登陆月球，到 2020 年 7 月“天问一号”火星探测器发射升空、2021 年 5 月 15 日成功着陆于火星乌托邦平原南部预选着陆区，我国首次火星探测任务着陆火星取得圆满成功，这是我国首次实现地外行星着陆，这使我国成为第二个成功着陆火星的国家；从 2012 年“蛟龙号”在马里亚纳海沟创造的 7062 米深潜记录，到 2020 年 5 月，“海斗一号”在马里亚纳海沟创造的 10907 米深潜记录，浙大机器鱼登上 Nature 封面，22cm 的机器鱼可在万米深海游动，相当于指尖承压 1 吨重量；这些均标志着我国在航天航空、深海探测领域的场地机器人研究水平已位于世界前列。未来，期待中国研发的机器人们能开辟宇宙探索的更高阵地，为人类遨游宇宙、探索星际注入中国智慧和中国担当！

我国的机器人研发起始于 20 世纪 70 年代初期，到了 20 世纪 90 年代，先后研制出了点焊、弧焊、装配、喷漆、切割、搬运、包装码垛等工业机器人，在工业机器人及零部件研发等方面取得了一定的成绩。但是总体而言，我国的工业机器人研发目前还处于初级阶段，产品以低档工业机器人为主，核心技术尚未掌握，关键部件绝大多数依赖进口，国产机器人的市场占有率十分有限。

中国是全世界工业机器人增长最快、销量最大的市场，总销量已经连续多年位居全球第一。2015 年的销量为 6.6 万台，占全球总销售量（24.7 万台）的 26.7%；2016 年的销量为 8.7 万台，占全球总销售量（29.4 万台）的 29.6%；2017 年的销量为 14.1 万台，占全球总销售量（38 万台）的 37.1%；2018 年的销量为 15.6 万台，占全球总销售量（42.2 万台）的 37%；2019 年的销量为 14.4 万台，占全球总销售量（37.3 万台）的 38.2%；2020 年的销量为 17 万台，占全球总销售量（39.7 万台）的 42.8%；2021 年的销量为 26.8 万台，占全球总销售量（51.7 万台）的 51.8%；2022 年的销量为 29 万台，占全球总销售量（55.3 万台）的 52.4%。

高端装备制造产业是国家重点支持的新兴产业之一，工业机器人作为高端装备制造业的重要组成部分，有望迎来快速发展期。

1.2 机器人的分类与应用

1.2.1 机器人分类

机器人的分类方法很多，但由于人们观察问题的角度有所不同，直到今天，还没有一种分类方法能够满意地对机器人进行世界所公认的分类。总体而言，通常的机器人分类方法主要有专业分类法和应用分类法两种。

1. **专业分类法**

专业分类法一般是机器人设计、制造和使用厂家技术人员所使用的分类方法，其专业性较强，行业外较少使用。目前，可按机器人控制系统的技术水平、机械结构形态和运动控制方式 3 种方式进行专业分类。

(1) 按控制系统的技术水平分类。根据机器人目前的控制系统技术水平，一般

可分为前述的示教再现机器人（第一代）、感知机器人（第二代）、智能机器人（第三代）3 类。第一代机器人已实用和普及，绝大多数工业机器人都属于第一代机器人，第二代机器人的技术已部分实用化，第三代机器人尚处于试验和研究阶段。

(2) 按机械结构形态分类。根据机器人现有的机械结构形态，有人将其分为圆柱坐标（Cylindrical Coordinate）、极坐标（Polar Coordinate）、直角坐标（Cartesian Coordinate）及关节型（Articulated）、并联型（Parallel）等，以关节型机器人较为常用。不同形态的机器人在外观、机械结构、控制要求、工作空间等方面均有较大的区别。例如，关节型机器人的动作类似人类手臂；而直角坐标及并联型机器人的外形和结构，则与数控机床十分类似。有关工业机器人的结构形态，将在第 2 章进行详细阐述。

(3) 按运动控制方式分类。根据机器人的控制方式，有人将其分为顺序控制型、轨迹控制型、远程控制型、智能控制型等。顺序控制型又称点位控制型，这种机器人只需要按照规定的次序和移动速度，运动到指定点进行定位，而不需要控制移动过程中的运动轨迹，它可以用于物品搬运等。轨迹控制型机器人需要同时控制移动轨迹、移动速度和运动终点，它可用于焊接、喷漆等连续移动作业。远程控制型机器人可实现无线遥控，故多用于特定的行业，如军事机器人、空间机器人、水下机器人等。智能控制型机器人就是前述的第三代机器人，多用于军事、场地、医疗等专门行业，智能型工业机器人目前尚未有实用化的产品。

2. **应用分类**

应用分类是根据机器人应用环境（用途）进行分类的大众分类方法，其定义通俗，易为公众所接受。例如，日本分为工业机器人和智能机器人两类；我国则分为工业机器人和特种机器人两类等。然而，由于对机器人的智能性判别尚缺乏严格、科学的标准，工业机器人和特种机器人的界线也较难划分。因此，本书参照国际机器人联合会（IFR）的相关定义，根据机器人的应用环境，将机器人分为工业机器人和服务机器人两类，前者用于环境已知的工业领域；后者用于环境未知的服务领域。如进一步细分，目前常用的机器人，基本上可分为以下几类，如图 1.2-1 所示。

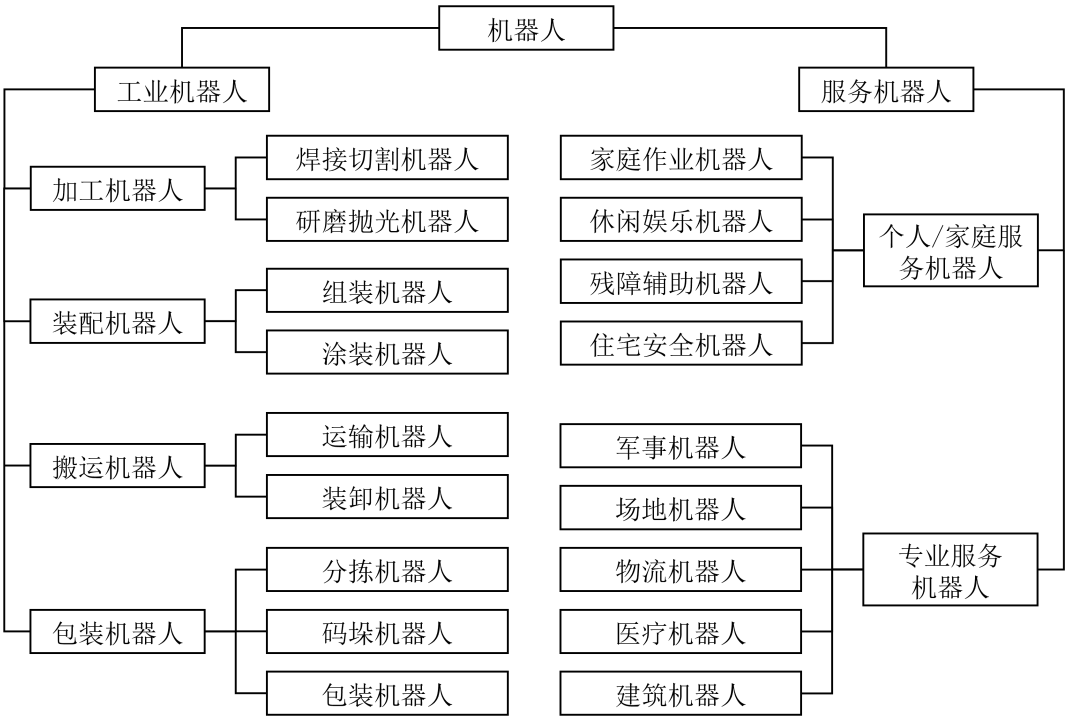
(1) 工业机器人。工业机器人（Industrial Robot，IR）是指在工业环境下应用的机器人，它是一种可编程的、多用途自动化设备。当前实用化的工业机器人以第一代示教再现机器人居多，但部分工业机器人（如焊接、装配等）已能通过图像的识别、判断，来规划或探测途径，对外部环境具有了一定的适应能力，初步具备了第二代感知机器人的一些功能。

工业机器人可根据其用途和功能，分为加工、装配、搬运、包装 4 大类；在此基础上，还可对每类进行细分。

(2) 服务机器人。服务机器人（Personal Robot，PR）是服务于人类非生产性活动的机器人总称，它在机器人中的比例高达 95%。根据 IFR 的定义，服务机器人是一种半自主或全自主工作的机械设备，它能完成有益于人类的服务工作，但不直接从事工业品的生产。

服务机器人的涵盖范围非常广，简言之，除工业生产用的机器人外，其他所有的机器人均属于服务机器人的范畴。因此，人们根据其用途，将服务机器人分为个人/家庭服务机器人（Personal/Domestic Robots）和专业服务机器人（Professional Service Robots）两类，在此基础上还可对每类进行细分。

以上两类产品研发、应用的简要情况如下。



1.2.2 工业机器人

工业机器人是用于工业生产环境的机器人总称。用工业机器人替代人工操作，不仅可保障人身安全、改善劳动环境、减轻劳动强度、提高劳动生产率，而且还能够起到提高产品质量、节约原材料消耗及降低生产成本等多方面作用，因此，它在工业生产各领域的应用也越来越广泛。根据工业机器人的功能与用途，其主要产品分为加工机器人、装配机器人、搬运机器人、包装机器人 4 大类机器人，如图1.7所示。

加工机器人

装配机器人

搬运机器人

包装机器人

图 1.7: 工业机器人的分类

1. **加工机器人**

加工机器人是直接用于工业产品加工作业的工业机器人，常用于金属材料焊接、切割、折弯、冲压、研磨、抛光等；此外，也有部分用于建筑、木材、石材、玻璃等行业的非金属材料切割、研磨、雕刻、抛光等加工作业。

焊接、切割、研磨、雕刻、抛光加工的环境通常较恶劣，加工时所产生的强弧光、高温、烟尘、飞溅、电磁干扰等都有害于人体健康。这些行业采用机器人自动作业，不仅可改善工作环境，避免人体伤害，而且还可自动连续工作，提高工作效率和改善加工质量。

焊接机器人（Welding Robot）是目前工业机器人中产量最大、应用最广的产品，被广泛用于汽车、铁路、航空航天、军工、冶金、电器等行业。自 1969 年美国 GM（通用汽车）公司在美国 Lordstown 汽车组装生产线上装备首台汽车点焊机器人以来，机器人焊接技术已日臻成熟，通过机器人的自动化焊接作业，可提高生产率、确保焊接质量、改善劳动环境，它是当前工业机器人应用的重要方向之一。

材料切割是工业生产不可缺少的加工方式，从传统的金属材料火焰切割、等离子切割，到可用于多种材料的激光切割加工都可通过机器人完成。目前，薄板类材料的切割大多采用数控火焰切割机、数控等离子切割机和数控激光切割机等数控机床加工；但异形、大型材料或船舶、车辆等大型废旧设备的切割已开始使用工业机器人。

研磨、雕刻、抛光机器人主要用于汽车、摩托车、工程机械、家具建材、电子电气、陶瓷卫浴等行业的表面处理。使用研磨、雕刻、抛光机器人不仅能使操作者远离高温、粉尘、有毒、易燃、易爆的工作环境，而且能够提高加工质量和生产效率。

2. **装配机器人**

装配机器人（Assembly Robot）是将不同的零件或材料组合成组件或成品的工业机器人，常用的有组装机器人和涂装机器人两大类。

计算机（Computer）、通信（Communication）和消费性电子（Consumer Electronic）行业（简称 3C 行业）是目前组装机器人最大的应用市场。3C 行业是典型的劳动密集型产业，采用人工装配，不仅需要使用大量的员工，而且操作工人的工作高度重复、频繁，劳动强度极大，人工难以承受；此外，随着电子产品不断向轻薄化、精细化方向发展，产品对零部件装配的精细程度在日益提高，部分作业已是人工无法完成的。

涂装机器人用于部件或成品的油漆、喷涂等表面处理，这类处理通常含有影响人体健康的有害、有毒气体，采用机器人自动作业后，不仅可改善工作环境，避免有害、有毒气体的危害，而且还可自动连续工作，提高工作效率和改善加工质量。

3. **搬运机器人**

搬运机器人是从事物体移动作业的工业机器人的总称，常用的主要有输送机器人（Transfer Robot）和装卸机器人（Handling Robot）两大类。

工业生产中的输送机器人以自动导引车（Automated Guided Vehicle，AGV）为主。AGV 具有自身的计算机控制系统和路径识别传感器，能够自动行走和定位停止，可广泛应用于机械、电子、纺织、卷烟、医疗、食品、造纸等行业的物品搬运和输送。在机械加工行业，AGV 大多用于无人化工厂、柔性制造系统（Flexible Manufacturing System，FMS）的工件、刀具的搬运和输送，它通常需要与自动化仓库、刀具中心及数控加工设备、柔性制造单元（Flexible Manufacturing Cell，FMC）的控制系统互连，以构成无人化工厂、柔性制造系统的自动化物流系统。

装卸机器人多用于机械加工设备的工件装卸（上下料），它通常和数控机床等自动化加工设备组合，构成柔性制造单元，成为无人化工厂、柔性制造系统的一部分。装卸机器人还经常用于冲剪、锻压、铸造等设备的上下料，以替代人工完成高风险、高温等恶劣环境下的危险作业或繁重作业。

4. **包装机器人**

包装机器人（Packaging Robot）是用于物品分类、成品包装、码垛的工业机器人，常用的主要有分拣机器人、包装机器人和码垛机器人三大类。

计算机、通信和消费电子行业（3C 行业）和化工、食品、饮料、药品工业是包装机器人的主要应用领域。3C 行业的产品产量大、周转速度快，成品包装任务繁重；化工、食品、饮料、药品包装由于行业特殊性，人工作业涉及安全、卫生、清洁、防水、防菌等方面的问题；因此，都需要利用装配机器人，来完成物品的分拣、

包装和码垛作业。

1.2.3 服务机器人

1. 基本情况

服务机器人是服务于人类非生产性活动的机器人总称。从控制要求、功能、特点等方面看，服务机器人与工业机器人的本质区别在于，工业机器人所处的工作环境在大多数情况下是已知的，因此，利用第一代机器人技术即可满足其要求；然而，服务机器人的工作环境在绝大多数情况下是未知的，故都需要使用第二代、第三代机器人技术。从行为方式上看，服务机器人一般没有固定的活动范围和规定的动作行为，它需要有良好的自主感知、自主规划、自主行动和自主协同等方面的能力，因此，服务机器人较多地采用仿人或生物、车辆等结构形态。

早在 1967 年，在日本举办的第一届机器人学术会议上，人们就提出了两种描述服务机器人特点的代表性意见。一种意见认为服务机器人是一种“具有自动性、个体性、智能性、通用性、半机械半人性、移动性、作业性、信息性、柔性、有限性等特征的自动化机器”；另一种意见认为服务机器人是具备以下 3 个条件的机器。

① 具有类似人类的脑、手、脚等功能要素。

② 具有非接触和接触传感器。

③ 具有平衡觉和固有觉的传感器。

当然，鉴于当时的情况，以上定义都强调了服务机器人的“类人”含义，突出了其由“脑”统一指挥、靠“手”进行作业、靠“脚”实现移动；通过非接触传感器和接触传感器，机器人能识别外界环境；利用平衡觉和固有觉等传感器感知本身状态等基本属性，这对服务机器人的研发具有参考价值。

服务机器人的出现虽然晚于工业机器人，但由于它与人类进步、社会发展、公共安全等诸多重大问题息息相关，应用领域众多，市场广阔，因此，其发展非常迅速、潜力巨大。有国外专家预测，在不久的将来，服务机器人产业可能成为继汽车、计算机后的另一新兴产业。国际机器人联合会（IFR）2013 年世界服务机器人统计报告等有关统计资料显示，目前已有 20 多个国家在进行服务型机器人的研发，有四十余种服务型机器人已进入商业化应用或试用阶段。2012 年全球服务机器人的总销量约为 301.6 万台，约为工业机器人（15.9 万台）的 20 倍；其中，个人/家用服务机器人的销量约为 300 万台，销售额约为 12 亿美元；专业服务机器人的销量为 1.6 万台，销售额为 34.2 亿美元。

在服务机器人中，个人/家用服务机器人（Personal/DomesticRobots）为大众化、低价位产品，其市场潜力最大。在专业服务机器人中，则以涉及公共安全的军事机器人（MilitaryRobot）、场地机器人（Field Robots）、医疗机器人的应用较广。

在服务机器人的研发领域，美国不但在军事、场地、医疗等高科技专业服务机器人的研究上遥遥领先于其他国家，而且在个人/家用服务机器人的研发上同样也

占有显著的优势，其服务机器人总量约占全球服务机器人市场的 60%。此外，日本的个人/家用服务机器人产量约占全球市场的 50%；欧洲的德国、法国也是服务机器人的研发和使用大国。我国在服务机器人领域的研发起步较晚，在 2005 年开始初具市场规模，总体水平与发达国家相比存在较大的差距；目前，我国的个人/家用服务机器人主要有吸尘、教育娱乐、保安、智能玩具等；专用服务机器人主要有医疗及部分军事、场地机器人等。

2. **个人/家用机器人**

个人/家用服务机器人（Personal/Domestic Robots）泛指为人们日常生活服务的机器人，包括家庭作业、娱乐休闲、残障辅助、住宅安全等。个人/家用服务机器人是被人们普遍看好的未来最具发展潜力的新兴产业之一。

在个人/家用服务机器人中，家庭作业机器人和娱乐休闲机器人的产量最大，两者占个人/家用服务机器人总量的 90% 以上；残障辅助机器人、住宅安全机器人的普及率目前还较低，但市场前景被人们普遍看好。

家用清洁机器人是家庭作业机器人中最早被实用化和最成熟的产品之一。早在 20 世纪 80 年代，美国已经开始进行吸尘机器人的研究，iRobot 等公司是目前家用服务机器人行业公认的领先企业，其产品技术先进、市场占有率为全球最大；德国的 KARCHER 公司也是著名的家庭作业机器人生产商，它在 2006 年研发的 Rc3000 家用清洁机器人是世界上第一台能够自行完成所有家庭地面清洁工作的家用清洁机器人。此外，美国的 Neato、Mint，日本的 SHINK、Panasonic（松下），韩国的 LG、三星等公司也都是全球较著名的家用清洁机器人研发、制造企业。

在我国，绝大多数家庭的作业服务目前还是由自己或家政服务人员承担，所使用的设备以传统工具和普通吸尘器、洗碗机等简单设备为主，家庭作业服务机器人的使用率非常低。

3. **专业服务机器人**

专业服务机器人（Professional Service Robots）的涵盖范围非常广，简言之，除工业生产用的工业机器人和为人们日常生活服务的个人/家用机器人外，其他所有的机器人均属于专业服务机器人。在专业服务机器人中，军事、场地和医疗机器人是应用最广的产品，3 类产品的概况如下。

(1) 军事机器人

军事机器人（Military Robot）是为了军事服务而研制的自主、半自主式或遥控的智能化装备，它可用来帮助或替代军人，完成特定的战术或战略任务。军事机器人具备全方位、全天候的作战能力和极强的战场生存能力，可在超过人类承受能力的恶劣环境，或在遭到毒气、冲击波、辐射等袭击时，继续进行工作；加上军事机器人也不存在人类的恐惧心理，可严格地服从命令、听从指挥，有利于指挥者对战局的掌控。

军事机器人的研发早在 20 世纪 60 年代就已经开始，产品已从第一代的遥控操作器，发展到了现在的第三代智能机器人。目前，世界各国的军事机器人已达上百

个品种，其应用涵盖侦察、排雷、防化、进攻、防御及后勤保障等各个方面。用于监视、勘察、获取危险领域信息的无人驾驶飞行器（UAV）和地面车（UGV）、具有强大运输功能和精密侦查设备的机器人武装战车（ARV）、在战斗中担任补充作战物资的多功能后勤保障机器人（MULE）是当前军事机器人的主要产品。

(2) 场地机器人

场地机器人（Field Robots）是除军事机器人外，其他可进行大范围作业的服务机器人的总称。场地机器人多用于科学研究和公共事业服务，如太空探测、水下作业、危险作业、消防救援、园林作业等。

美国的场地机器人研究始于 20 世纪 60 年代，其产品已遍及陆地、水下和太空，从 1967 年的海盗号火星探测器，到 2003 年的 Spirit MER-A（勇气号）和 Opportunity（机遇号）火星探测器、2011 年的 Curiosity（好奇号）核动力火星探测器和 2020 年的 Perseverance（毅力号）核动力火星探测器，都无一例外地代表了全球空间机器人研究的最高水平。此外，俄罗斯和欧盟在太空探测机器人等方面的研究和应用也居世界领先地位，如早期的空间站飞行器对接、燃料加注机器人等；德国于 1993 年研制、由哥伦比亚号航天飞机携带升空的 ROTEX 远距离遥控机器人等，也都代表了当时的空间机器人技术水平；我国在探月、水下机器人方面的研究也取得了较大的进展。

(3) 医疗机器人

医疗机器人是今后专业服务机器人的重点发展领域之一。医疗机器人主要用于伤病员的手术、救援、转运和康复，它包括诊断机器人、外科手术或手术辅助机器人、康复机器人等。例如，通过外科手术机器人，医生可利用其精准性和微创性，大面积减小手术伤口、迅速恢复正常生活等。据统计，目前全世界已有 30 个国家、近千家医院成功开展了数十万例机器人手术，手术种类涵盖泌尿外科、妇产科、心血管外科、胸外科、肝胆外科、耳鼻喉科等。

当前，医疗机器人的研发与应用大部分都集中于美国、欧洲、日本等发达国家，发展中国家的普及率还很低。美国的 Intuitive Surgical（直觉外科）公司是全球领先的医疗机器人研发、制造企业，该公司研发的达芬奇机器人是目前世界上最先进的手术机器人系统，它可模仿外科医生的手部动作，进行微创手术，目前已经成功用于普通外科、胸外科、泌尿外科、妇产科、头颈外科及心血管外科等手术。

1.3 工业机器人应用

1.3.1 技术发展与产品应用

1. **技术发展简史**

工业机器人自 1959 年问世以来，经过六十多年的发展，在性能和用途等方面都有了很大的变化；现代工业机器人的结构越来越合理、控制越来越先进、功能越

来越强大、应用越来越广泛。世界工业机器人的简要发展历程、重大事件和重要产品研制的简况如下。

(1) 1959 年，Joseph F.Engelberger（约瑟夫·恩格尔伯格）利用 George Devol（乔治·德沃尔）的专利技术，研制出了世界上第一台真正意义上的工业机器人 Unimate。该机器人具有水平回转、上下摆动和手臂伸缩 3 个自由度，可用于点对点搬运。

(2) 1961 年，美国 GM（通用汽车）公司首次将 Unimate 工业机器人应用于生产线，机器人承担了压铸件叠放等部分工序。

(3) 1968 年，美国斯坦福大学研制出了首台具有感知功能的第二代机器人 Shakey。同年，Unimation 公司将机器人的制造技术转让给了日本 KAWASAKI（川崎）公司，日本开始研制、生产机器人。次年，瑞典的 ASEA 公司（阿西亚公司，现为 ABB 集团）研制了首台喷涂机器人，并在挪威投入使用。

(4) 1972 年，日本 KAWASAKI（川崎）公司研制出了日本首台工业机器人“Kawasaki-Unimate2000”。次年，日本 HITACHI（日立）公司研制出了世界首台装备有动态视觉传感器的工业机器人；而德国 KUKA（库卡）公司则研制出了世界首台 6 轴工业机器人 Famulus。

(5) 1974 年，美国 Milacron（米拉克龙，著名的数控机床生产企业）公司研制出了首台微机控制的商用工业机器人 Tomorrow Tool（T3）；瑞典 ASEA 公司研制出了世界首台微机控制、全电气驱动的 5 轴涂装机器人 IRB6；全球最著名的数控系统（CNC）生产商、日本 FANUC 公司（发那科）开始研发、制造工业机器人。

(6) 1977 年，日本 YASKAWA（安川）公司开始工业机器人研发生产，并研制出了日本首台采用全电气驱动的机器人 MOTOMAN-L10(MOTOMAN 1 号)。次年，美国 Unimate 公司和 GM（通用汽车）公司联合研制出了用于汽车生产线的垂直串联型（Vertical Series）可编程通用装配机器人 PUMA（Programmable Universal Manipulator for Assembly）；日本山梨大学研制出了水平串联型（Horizontal Series）自动选料、装配机器人 SCARA（Selective Compliance Assembly Robot Arm）；德国 REIS（徕斯，现为 KUKA 成员）公司研制出了世界首台具有独立控制系统、用于压铸生产线的工件装卸的 6 轴机器人 RE15。

(7) 1983 年，日本 DAIHEN 公司（日本大阪变压器集团 Osaka Transformer Co.,Ltd 所属，国内称 OTC 或欧希地）研发了世界首台具有示教编程功能的焊接机器人。次年，美国 Adept Technology（娴熟技术）公司研制出了世界首台电机直接驱动、无传动齿轮和铰链的机器人 Adept One。

(8) 1985 年，德国 KUKA（库卡）公司研制出了世界首台具有 3 个平移自由度和 3 个转动自由度的 Z 型 6 自由度机器人。

(9) 1992 年，瑞士 Demaurex 公司研制出了世界首台采用 3 轴并联结构的包装机器人 Delta。

(10) 2005 年，日本 YASKAWA（安川）公司推出了新一代、双腕 7 轴工业机器

人。次年，意大利 COMAU（柯马，菲亚特成员、著名的数控机床生产企业）公司推出了首款 WiTP 无线示教器。

(11) 2008 年，日本 FANUC 公司（发那科）、YASKAWA（安川）公司的工业机器人累计销量相继突破 20 万台，成为全球工业机器人累计销量最大的企业。次年，ABB 公司研制出全球精度最高、速度最快的 6 轴小型机器人 IRB 120。

(12) 2013 年，谷歌公司开始大规模并购机器人公司，至今已相继并购了 Autofuss、Boston Dynamics（波士顿动力）、Bot & Dolly、DeepMind（英）、Holomni、Industrial Perception、Meka、Redwood Robotics、Schaft（日）、Nest Labs、Spree、Savioke 等多家公司。

(13) 2014 年，ABB 公司研制出世界上首台真正实现人机协作的机器人 YuMi。同年，德国 REIS（徕斯）公司并入 KUKA（库卡）公司。

2. **典型应用**

根据国际机器人联合会（IFR）等部门的最新统计，当前工业机器人的应用行业分布情况大致如图1.8所示。其中，汽车制造业、电子电气工业、金属制品及加工业是目前工业机器人的主要应用领域。

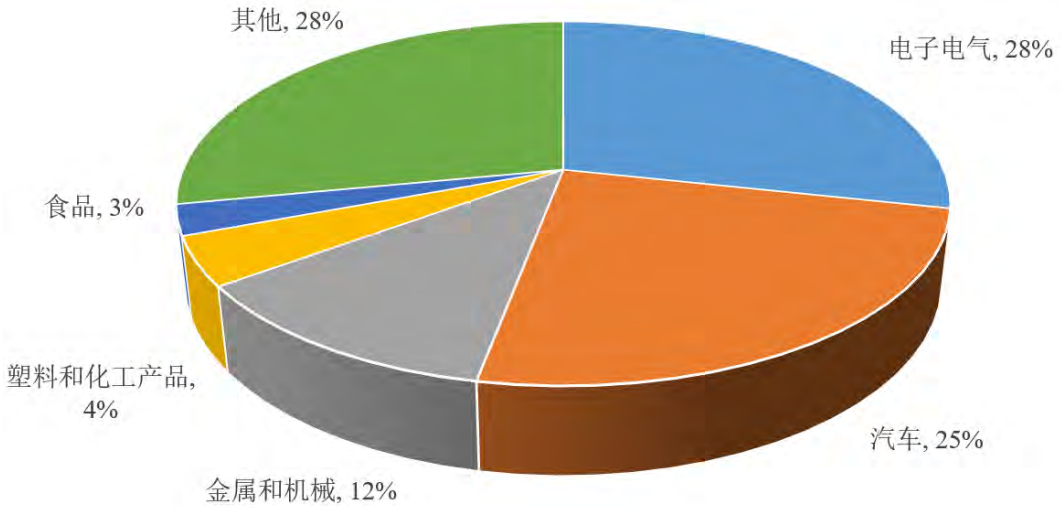


图 1.8: 工业机器人的应用

汽车及汽车零部件制造业历来是工业机器人用量最大的行业，其使用量长期保持在工业机器人总量的 40% 以上，使用的产品以加工、装配类机器人为主，是焊接、研磨、抛光及装配、涂装机器人的主要应用领域。

电子电气（包括计算机、通信、家电、仪器仪表等）是工业机器人应用的另一主要行业，其使用量也保持在工业机器人总量的 20% 以上，使用的主要产品为装配、包装类机器人。

金属制品及加工业的机器人用量大致在工业机器人总量的 10% 左右，使用的产品主要为搬运类的输送机器人和装卸机器人。

建筑、化工、橡胶、塑料和食品、饮料、药品等其他行业的机器人用量都在工

业机器人总量的 10% 以下，橡胶、塑料、化工、建筑行业使用的机器人种类较多；食品、饮料、药品行业使用的机器人通常以加工、包装类为主。

1.3.2 主要生产企业

目前，全球工业机器人的生产厂家主要集中于东亚和欧洲，例如，日本的 FANUC（发那科）、YASKAWA（安川）、KAWASAKI（川崎）、NACHI（不二越）、DAIHEN（OTC 或欧希地）、Panasonic（松下）及韩国的 HYUDAI（现代），瑞士与瑞典的 ABB，德国的 KUKA（库卡，现已被美的控股）、REIS（徕斯，现为 KUKA 成员）及意大利的 COMAU（柯马），奥地利的 IGM（艾捷莫）等。

以上企业从事工业机器人研发的时间基本可分为图 1.3-2 所示的 20 世纪 60 年代末、70 年代中、70 年代末 3 个时期，FANUC、YASKAWA、ABB、KUKA 是当前工业机器人产销量最大的代表性企业；KAWASAKI、NACHI 公司是全球最早从事工业机器人研发生产的企业；DAIHEN 焊接机器人是国际名牌，以上企业的产品在我国应用最为广泛，简介如下，KUKA 机器人的产品将在第 2 章详述。

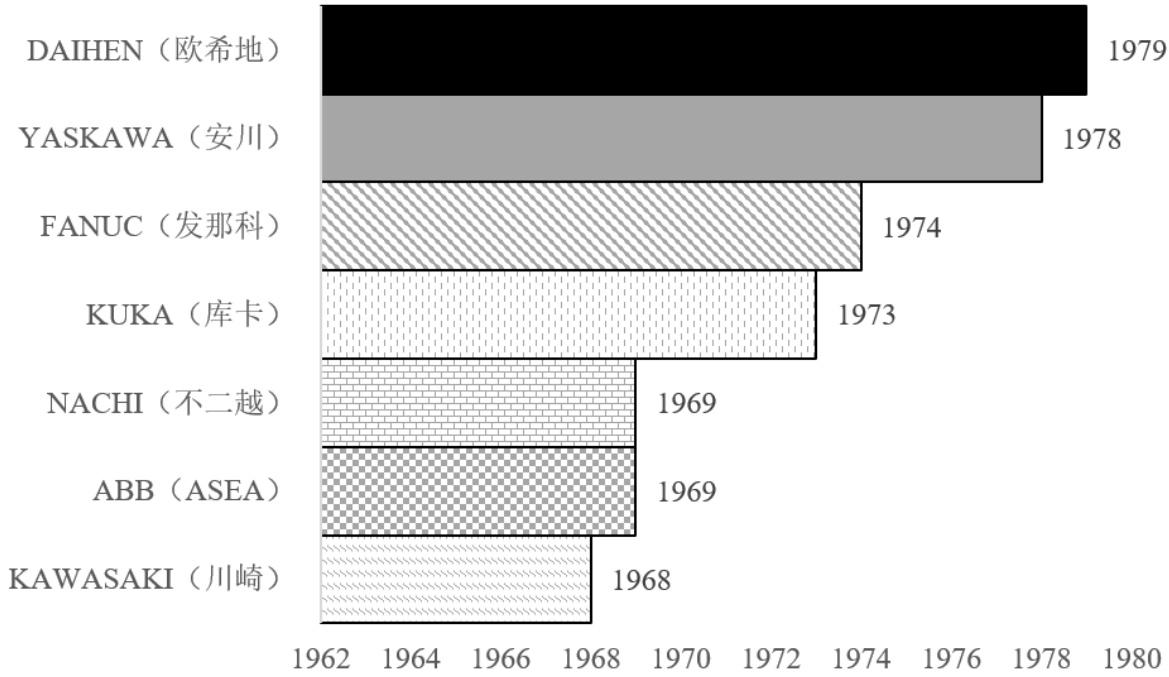


图 1.9: 工业机器人研发起始时间

1. **FANUC（发那科）**

FANUC（发那科）从 1956 年起就开始从事数控和伺服的民间研究，1972 年公司正式成立，1974 年开始进行工业机器人的研发与生产，是目前全球最大、最著名的数控系统（CNC）生产厂家和工业机器人生产厂家，工业机器人产销量自 2008 年至今一直位居世界第一。

FANUC 公司自 1977 年开始批量生产、销售工业机器人（ROBOT-MODEL1），1992 年在美国成立了 GE Fanue 机器人公司（GE Fanuc Robotics Corporation），1997

年和上海电气集团合资，成立了上海发那科（FANUC）机器人有限公司，是最早进入中国市场的国外工业机器人企业之一。2023 年，上海发那科智能工厂（三期）正式开业，占地约 431 亩，建筑面积 30 万平方米，总投资 15.8 亿元，建成集研发、制造、展示、销售、系统集成以及售后服务于一体的智能工厂，FANUC 努力地将物联网和人工智能技术应用于包括 FA、机器人和机器人设备在内的所有领域。

FANUC 工业机器人产品主要包括图1.10所示的垂直串联通用及专用（弧焊、涂装、食品药品等）工业机器人、并联 Delta 结构机器人和变位器等。

垂直串联通用及专用工业机器人　　并联 Delta 结构　　双轴变位器

图 1.10: FANUC 工业机器人产品

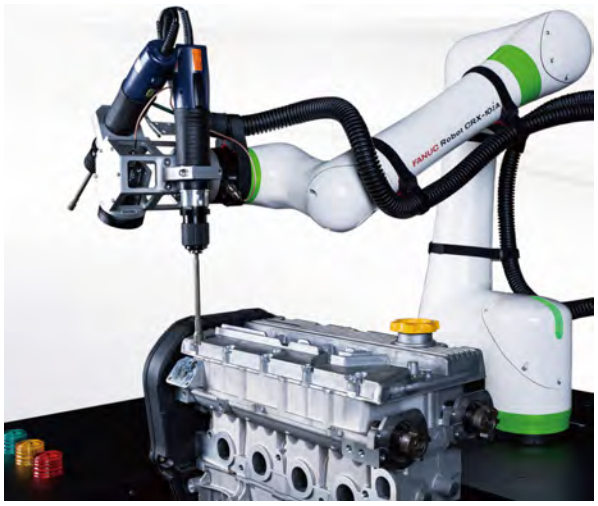


图 1.11: CRX 系列协作型机器人

图1.11所示的 CRX 系列协作型机器人是 FANUC 近期推出的新产品，它的出现，扫除了人机协作的障碍，它既能与人类并肩协同工作，又可确保周边区域安全无虞。无论是汽车轮胎组装、机床工件搬运，还是电子产品装配，它都能轻松拿下。

CRX 系列协作型机器人具有界面，可轻松连接夹爪、视觉等外围设备。同时，它还提供了专门用于外围设备的插件软件，便于快速启动和简便安装，即使是那些没有使用机器人经验的客户也可以直接开箱安装。CRX 针对小型部件的搬运、装

配等应用需求为用户提供精准、灵活、安全的人机协作解决方案，它还具有优秀的运动性能，其最高运动速度达到 1000mm/s。

2. **YASKAWA（安川）**

YASKAWA（安川）公司成立于 1915 年，是全球著名的伺服电机及驱动器、变频器和工业机器人生产厂家，2003-2008 年的工业机器人产销量为全球第一，目前产销仅次于 FANUC 位居世界第二。

YASKAWA 的工业机器人研发始于 1977 年，随后创立了 MOTOMAN 工业机器人品牌，1990 年正式成立 MOTOMAN 机器人中心，1996 年成立北京工业机器人合资公司，成为首家进入中国的工业机器人企业,2021 年 MOTOMAN 工业机器人累计出货达到 50 万台。

垂直串联通用及专用工业机器人　　并联 Delta 结构机器人　　水平串联 SCARA 机器人

图 1.12: YASKAWA 工业机器人产品

YASKAWA 工业机器人产品主要包括图1.12所示的垂直串联通用及专用（弧焊、涂装、食品药品等）工业机器人、并联 Delta 结构机器人、水平串联 SCARA 机器人等。

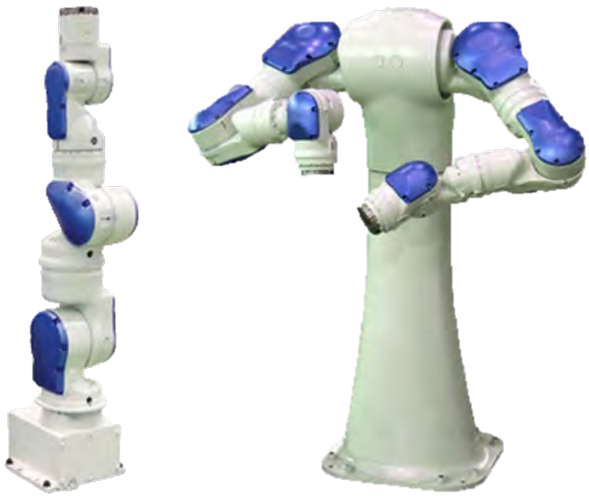


图 1.13: YASKAWA 手臂型机器人

图1.13所示的手臂型机器人（Arm Robot）是 YASKAWA 研发的第二代智能工

业机器人产品。手臂型机器人同样带有触觉传感器等智能检测器件，可感知人体接触并安全停止，实现人机协同安全作业。

安川手臂型机器人采用的是 7 轴垂直串联、类人手臂结构，其运动灵活、几乎不存在作业死区。安川手臂型机器人目前有图1.13所示的 SIA 系列 7 轴单臂（Single-arm）、SDA 系列 15 轴（2×7 单臂 + 基座回转）双臂（Dual-arm）两类，机器人可用于 3C、食品、药品等行业的人机协同作业。

3. **ABB**

ABB（Asea Brown Boveri）集团公司是由 ASEA（阿西亚，总部位于瑞典）和 Brown.Boveri & Co., Ltd（布朗勃法瑞，简称 BBC，总部位于瑞士）两个具有百年历史的著名电气公司于 1988 年合并而成，集团总部现在位于瑞士苏黎世。

ASEA 公司成立于 1890 年，是世界著名的电力设备制造企业；BBC 公司成立于 1891 年，是世界著名的电力设备、低压电器、电气传动设备生产企业，产品遍及工商业、民用建筑配电、各类自动化设备和大型基础设施工程。ASEA 公司在 1969 年研发出了全球第一台喷涂机器人，并开始进入工业机器人的研发制造领域，在 1974 年，又研发出了世界首台微机控制、全电气驱动的 5 轴涂装机器人 IRB6。组建 ABB 后，在 2009 年研制出了当时全球精度最高、速度最快的 6 轴小型工业机器人 IRB 120；在 2011 年研制出了当时全球最快的码垛机器人 IRB 460；2014 年研发了第二代人机协作机器人 YuMi。2005 年，在上海成立了 ABB 机器人研发中心；2010 年，中国机器人整车喷涂实验中心建成。

ABB 工业机器人产品主要包括图1.14所示的垂直串联通用及专用（弧焊、涂装、食品药品等）工业机器人、并联 Delta 结构机器人、水平串联 SCARA 机器人和变位器等。

垂直串联通用及专用工业机器人

并联 Delta 结构机器人

水平串联 SCARA 机器人

图 1.14: ABB 工业机器人产品

ABB 公司第二代智能工业机器人的代表性产品为图1.15所示的 YuMi 协作型机器人。YuMi 协作型机器人的结构和安川手臂型机器人基本相同，机器人同样有 7 轴单臂和 15 轴双臂两种，机器人带有触觉传感器等智能检测器件，可感知人体接触并安全停止，实现人机协同安全作业。

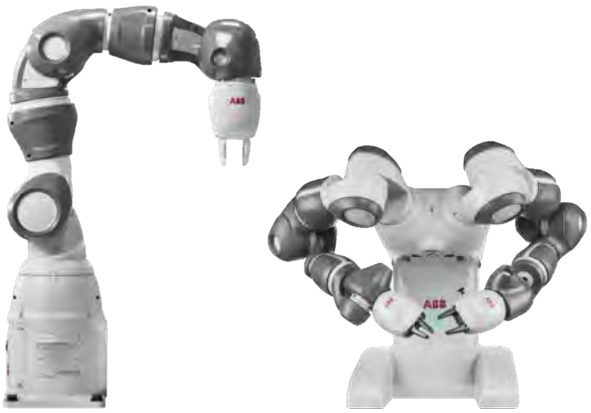


图 1.15: YuMi 协作型机器人

4. 其他

(1) KAWASAKI。KAWASAKI（川崎）公司成立于 1878 年，是具有悠久历史的日本著名大型企业集团，业务范围涵盖航空、航天、军事、电力、铁路、造船、工程机械、钢结构、发动机、摩托车、机器人等众多领域。其产品代表了日本科技的先进水平。

KAWASAKI（川崎）公司的工业机器人研发始于 1968 年，是日本最早研发、生产工业机器人的著名企业，在焊接机器人技术方面居世界领先水平，日本首台工业机器人“川崎-Unimation2000”、全球首台用于摩托车车身焊接的弧焊机器人均为该公司研发。

(2) NACHI。NACHI（不二越）是日本著名的机床企业集团，其主要产品有轴承、液压元件、刀具、机床、工业机器人等。NACHI 公司成立于 1928 年，1969 年开始研发生产机床和工业机器人，是日本最早研发生产和世界著名的工业机器人生产厂家之一，其焊接机器、搬运机器人技术处于世界领先水平。

不二越（NACHI）公司曾在 1979 年成功研制出了世界首台电机驱动多关节焊接机器人；2013 年成功研制出 300mm 往复时间达 0.31s 的世界最快的轻量机器人 MZ07；这些产品都代表了当时工业机器人在某一方面的最高技术水平。NACHI（不二越）公司的中国机器人商业中心成立于 2010 年，进入中国市场较晚。

(3) DAIHEN。DAIHEN 公司为日本大阪变压器集团（Osaka Transformer Co.,Ltd，OTC）所属企业，国内称为“欧希地（OTC）”公司。DAIHEN 公司是日本著名的焊接机器人生产企业，公司自 1979 年起开始从事焊接机器人生产；在 1983 年，研发了全世界首台具有示教编程功能的焊接机器人；在 1991 年，研发了全世界首个协同作业机器人焊接系统；这些产品的研发，都对工业机器人的技术进步和行业发展起到了重大的促进作用。

DAIHEN 公司自 2001 年起开始和 NACHI（不二越）合作研发工业机器人。自 2002 年起，该公司先后在我国成立了欧希地机电（上海）有限公司、欧希地机电（青岛）有限公司及欧希地机电（上海）有限公司广州、重庆、天津分公司，进行工业机器人产品的生产和销售。

第 2 章 快速入门

2.1 开箱

/

2.2 实战连线

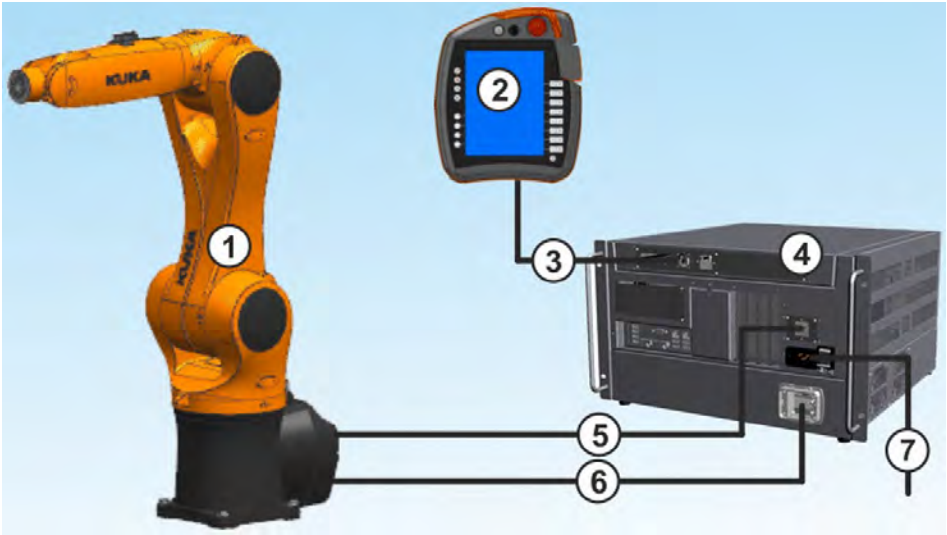


图 2.1: 工业机器人示例

① 机械手　② smartPAD 示教器
③ 连接线缆（控制柜-smartPAD）　④ 控制柜（机器人控制系统）
⑤ 数据线（控制柜-机械手）　⑥ 电机线缆（控制柜-机械手）
⑦ 外接设备连接线缆（控制柜-外接设备）

2.2.1 电机线缆的连接（控制柜-机械手）

使用图2.3的电机线缆，分别连接控制柜的 X20 接口（图2.5）与机械手的 X30 接口（图2.6）。直接把插头用力插上即可，插头可以自锁；若线缆正反错误，则安装不上。若发出咔擦声，则确定电机线缆插头在控制柜或机械手上已正确闭锁。电机线缆两侧的安装方式相同。

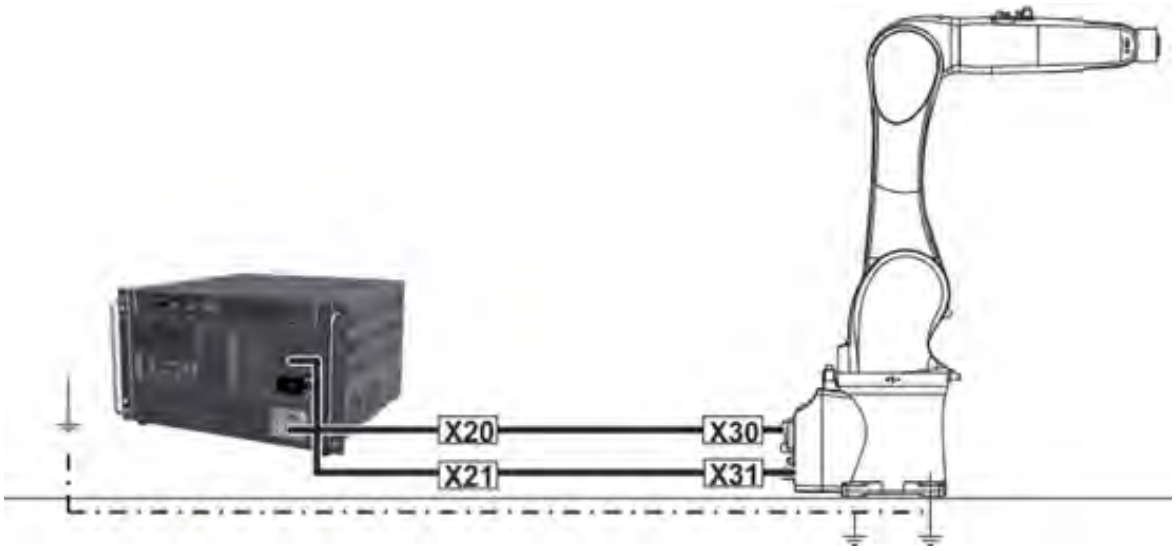


图 2.2: 电机线缆与数据线的连接（控制柜-机械手）

图 2.3: 电机线缆

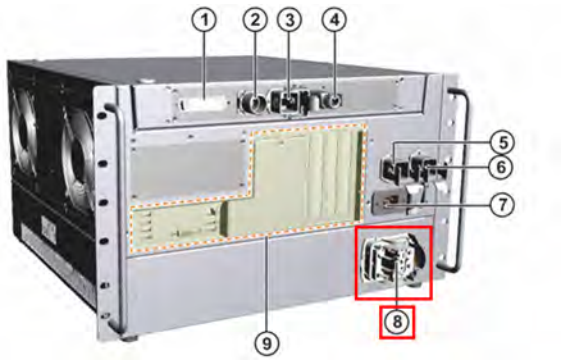


图 2.4: 电机线缆插口

图 2.5: 电机插头插入控制柜 X20 接口中

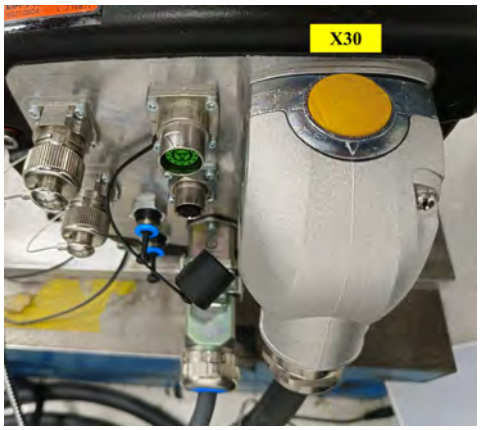


图 2.6: 电机插头插入机械手 X30 接口中

2.2.2 数据线的连接（控制柜-机械手）

图 2.7: 数据线

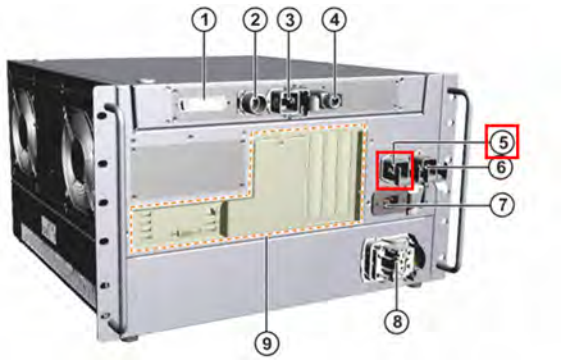


图 2.8: 数据线插口

图 2.9: 数据线插头插入控制柜 X21 接口中

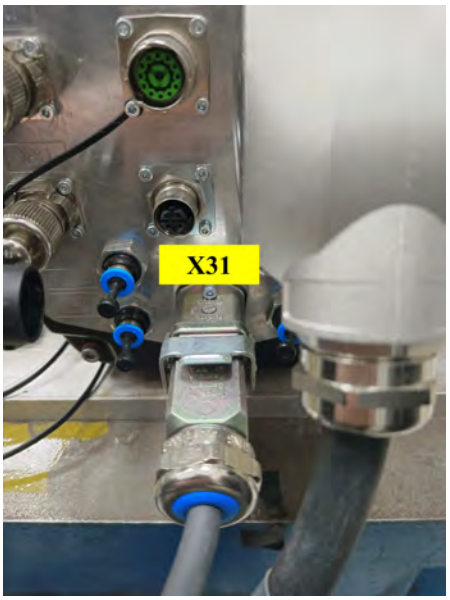


图 2.10: 数据线插头插入机械手 X31 接口中

使用图2.7的数据线，分别连接控制柜的 X21 接口（图2.9）与机械手的 X31 接口（图2.10）。数据线的每个头都有编号，将数据线插进接口，若发出咔擦声，则确定数据线插头在控制柜或机械手上已正确闭锁。

2.2.3 分离式安全接口 X11 的连接（控制柜自连）

如图2.11所示的分离式安全接口 X11，使用 RV 铜芯电子线（图2.12中红色的线）按照如下的方式连接 X11 的插口

(1) 急停：1-2，19-20；

(2) 操作人员安全：3-4，21-22；

(3) 确认操作员安全：5-6，23-24；

(4) 安全操作停止：7-8，25-26；

(5) 安全停止：9-10，27-28。

只有当机器人控制器已关断时，插头 X11 才能插入或拔出；如果插头 X11 在有电压时插入或拔出，会导致设备损伤。

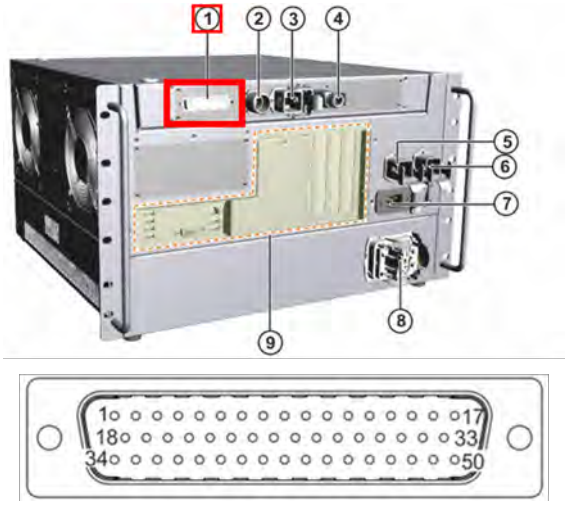


图 2.11: 分离式安全接口 X11

图 2.12: 连接好的 X11

2.2.4 示教器线缆的连接（控制柜-smartPAD）

使用图2.13的示教器线缆，连接控制柜的 X19 接口（图2.14）。开机前要将示教器线缆插头插入控制柜 X19 接口中，插入后会直接自锁，需旋转才可将其拔下。

图 2.13: 示教器线缆

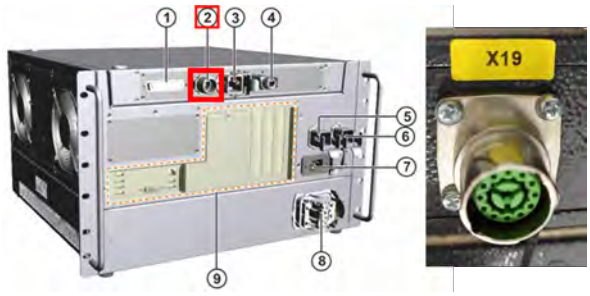


图 2.14: 控制柜 X19 接口

2.2.5 电源线缆的连接（控制柜-外接 220V 电源）

使用图2.15的电源线缆，将控制柜的 X1 接口（图2.16）与外接 220V 电源相连。

图 2.15: 电源线缆

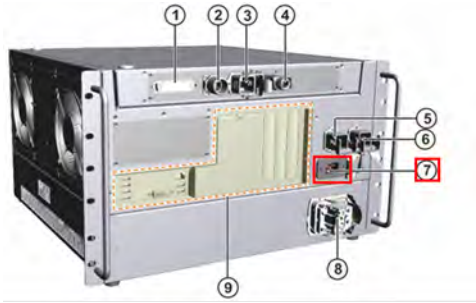


图 2.16: 电源插口 X1

2.3 启动机器人控制系统（控制柜）

1. 两步检查：

(1) 检查2.2节实战连线部分的所有连线是否准确无误；

(2) 检查控制柜的电源是否符合机器人使用说明书的规定，如额定输入电压、额定输入端功率。

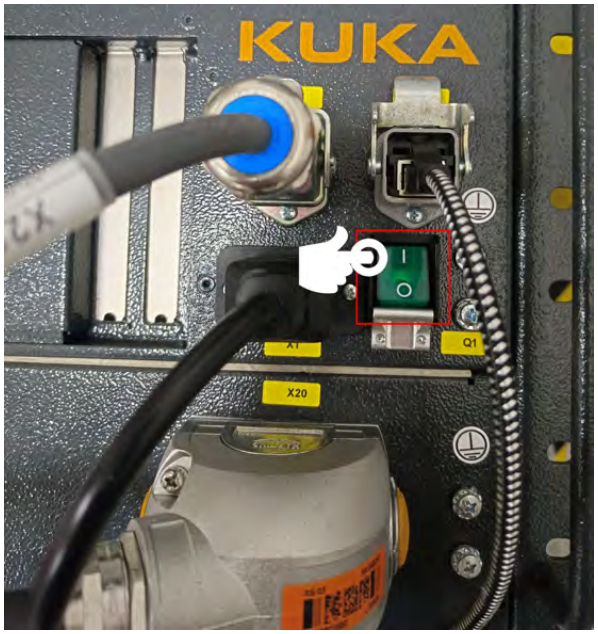


图 2.17: 电源主开关 ON（开）位置

2. 开启控制柜：

如图2.17，将机器人控制柜的电源主开关置于 I（ON）位置。通常情况下，电源主开关打开后，控制柜（控制系统）便可启动并自动安装 KUKA 系统软件（KUKA System Software，KSS），如图2.18。系统启动完成后，示教器（smartPAD）便可显示

图 **2.18:** 自启动 KSS 软件

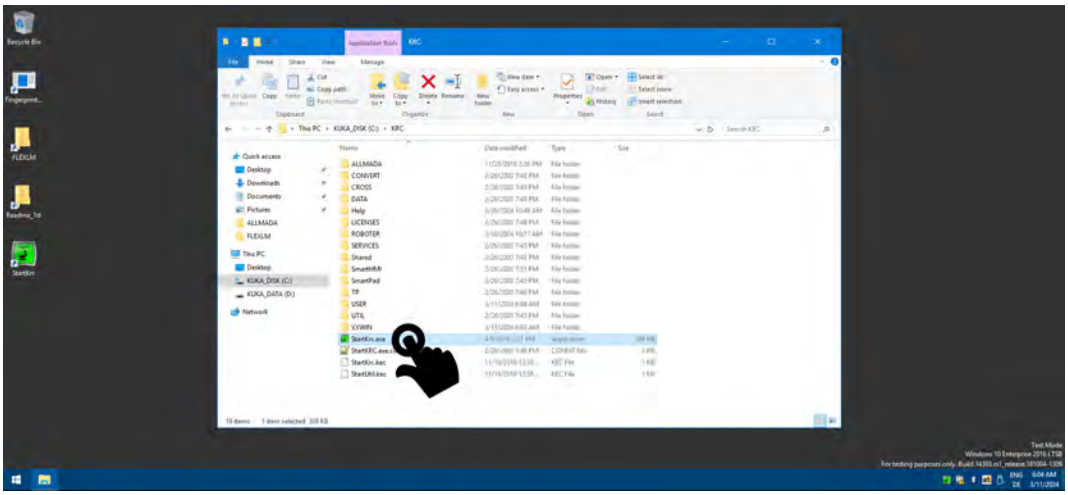
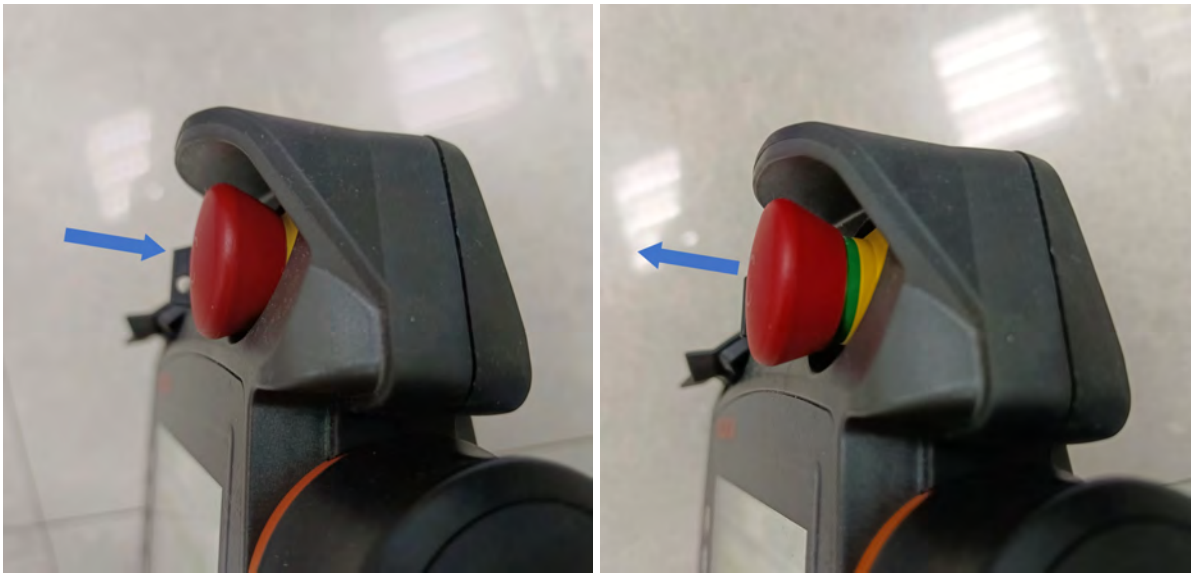


图 **2.19:** 双击 StartKRC.exe

正常操作界面。

注意：如果控制系统的自动启动功能被禁止，KSS 不能自动安装与启动，可选择路径 C:/KRC，如图2.19双击 KSS 启动程序 StartKRC.exe，便可安装、启动 KSS，若机器人控制系统登录在网络上，则启动过程会较长。



按下示教器急停按钮　　复位示教器急停按钮

图 2.20: 示教器急停按钮的使用

图 2.21: 单击 I 接通伺服驱动器状态主电源

3. 启动伺服驱动系统

(1) 复位示教器的急停按钮，如图2.20(b)，将按钮左右扭一下即可轻松拔出。

(2) 利用示教器（smartPAD）操作，如图2.21，单击 I，若红框处的 I 显示绿色，则表示伺服驱动器已启动并正常运行。启动伺服驱动系统后，便可按照规定的步骤对机

器人进行正常操作。

2.4 用示教器手动控制机器人

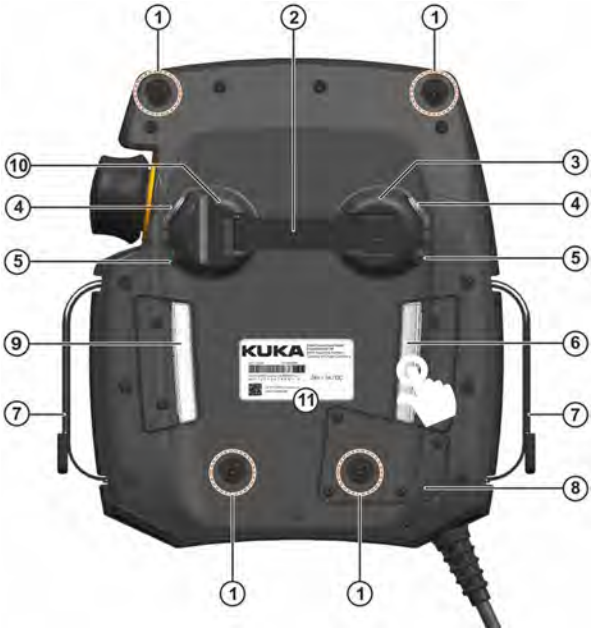


图 2.22: 按住示教器背面的手握确认开关

如图2.22，按住示教器背面的手握确认开关⑥或⑨（按到中间位置，不要按到底）。此时，驱动器通/断控制及启动条件显示框（图2.23红色框）显示为 S（绿色），且伺服驱动器状态显示图标（图2.23黄色框）显示为 I（绿色）。

图 2.23: 驱动器通/断控制显示与伺服驱动器状态显示

在按住示教器背面的手握确认开关下，可以按下示教器的前 6 行手动方向键（图2.24所示，6 个“-”和 6 个“+”），控制机器人的运动；第 7 行用于调节机器人程序自动运行的速度倍率；第 8 行用于调节机器人手动示教运行的速度倍率，机器人在此处的运行就是手动示教。

常用的坐标系一般有两种。

(1) 如图2.25左图，在默认情况下选择的是轴坐标系，可通过前 6 行手动方向键控制机器人向A_1、A_2、A_3、A_4、A_5、A_6 方向运动；

(2) 如图2.25右图，也可切换为工具坐标系，其坐标中心在法兰坐标中心，可通过 12 个手动方向键控制机器人向 x、y、z、A、B、C 方向运动。

图 2.24: 示教器手动方向键

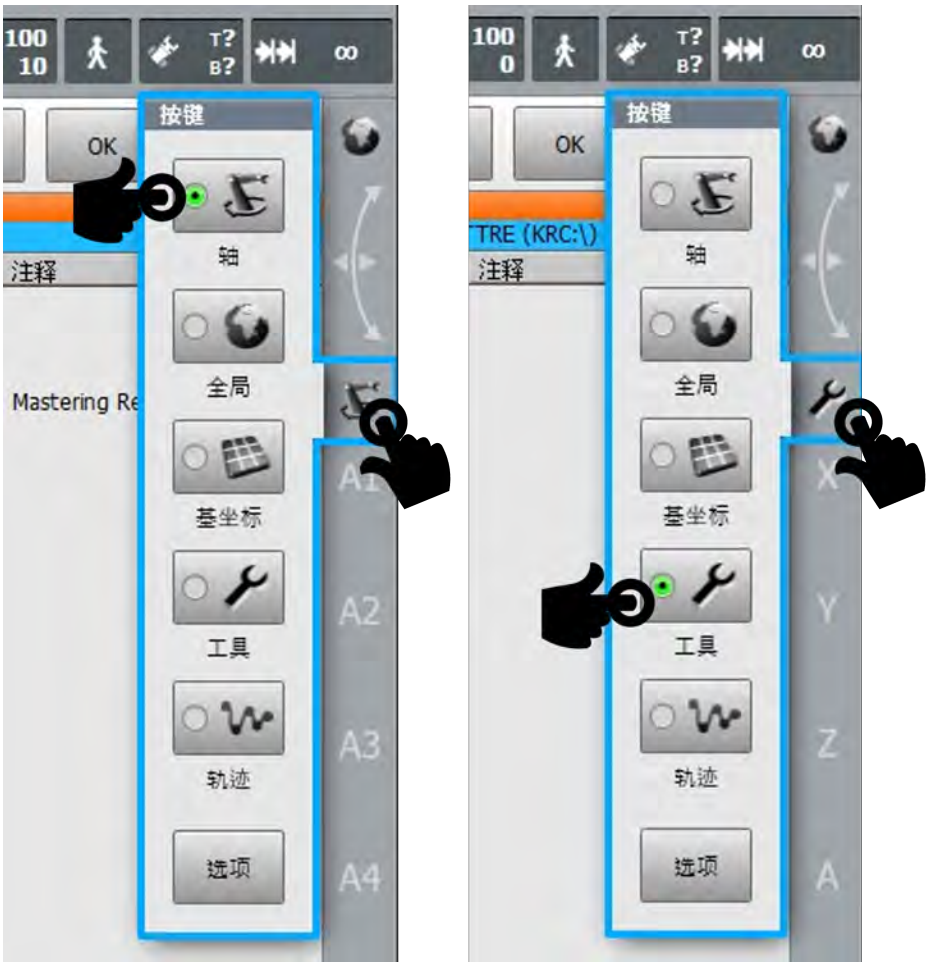


图 2.25: 坐标系的选择与切换

第 3 章 KUKA 机器人的基本操作

3.1 KUKA 机器人的结构

KUKA 机器人由下列部件构成：

- 机械手
- 机器人控制系统
- smartPAD 示教器
- 连接电缆
- 选项，附件

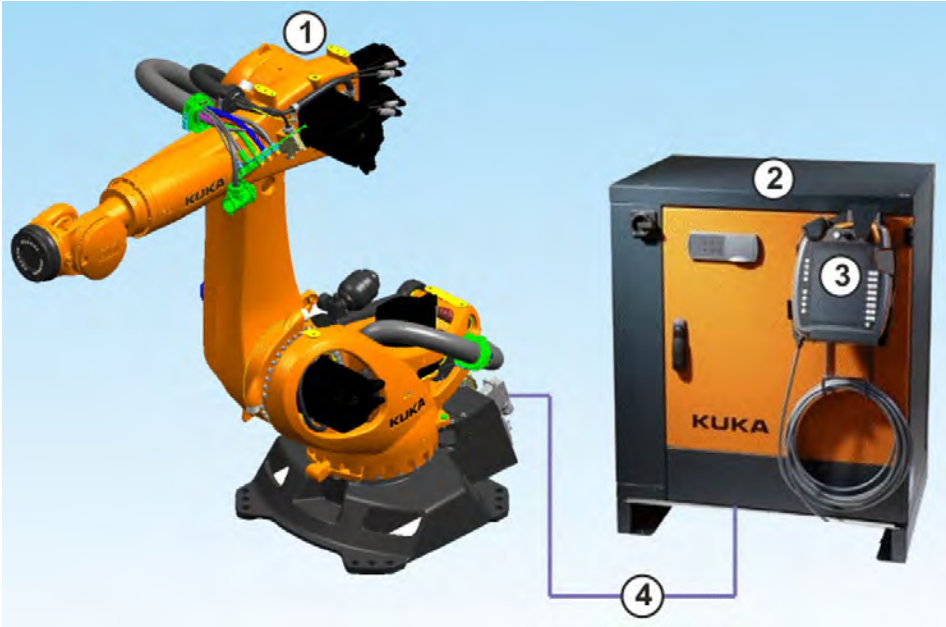


图 3.1: KUKA 工业机器人示例图

1	机械手	3	KUKA smartPAD
2	机器人控制系统	4	连接电缆

3.2 smartPAD 示教器

smartPAD 是一种用于工业机器人的手持编程器，也叫示教器，是 KUKA 机器人最主要的操作部件，本书 KUKA 机器人控制系统配套的示教器如图3.2所示。

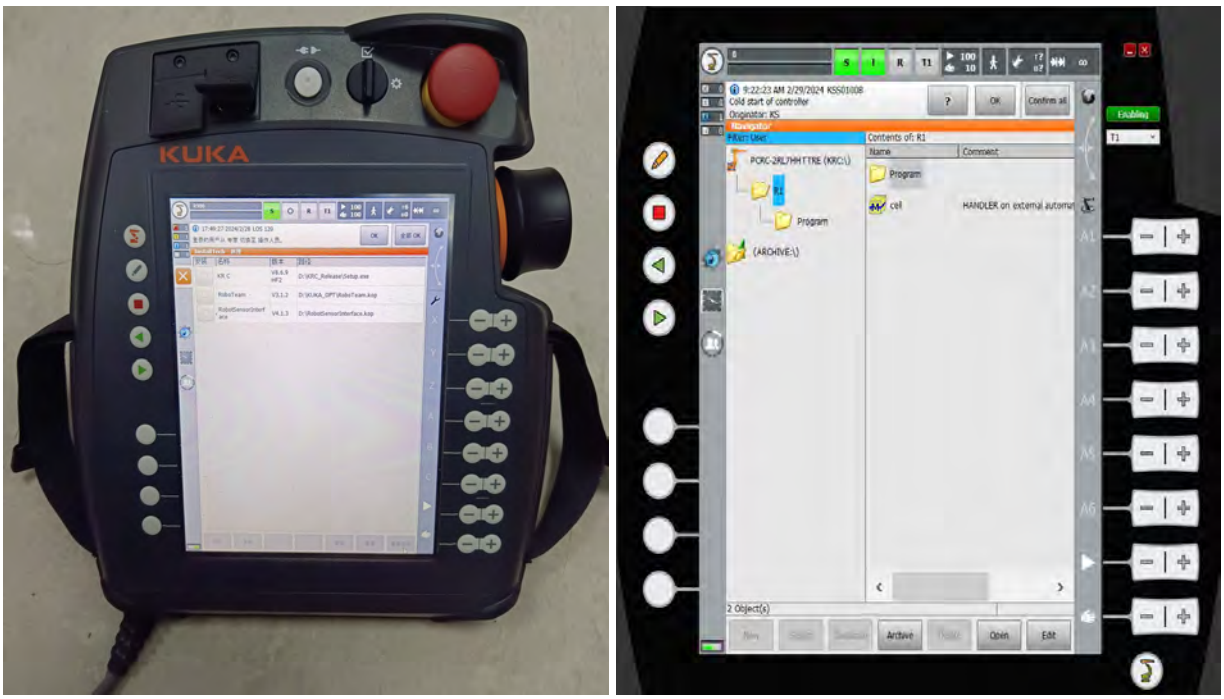


图 3.2: smartPAD 示教器

3.2.1 smartPAD 结构

smartPAD 具有工业机器人操作和编程所需的各种操作和显示功能，其配备的 smartHMI 触摸屏可用手指或指示笔进行操作，无需使用外部鼠标和键盘。

3.2.2 smartPAD 正面

smartPAD 示教器正面的按键与开关如图3.3所示，其基本功能如下。

① 2 个有盖子的 USB2.0 接口：可用于存档，以及插入 NTFS 和 FAT32 格式化的 U 盘。

② 热插拔 smartPAD：可在机器人系统接通时取下 smartPAD。

③ 运行方式选择开关（机器人解锁钥匙）：通过该选择开关可以调用连接管理器，切换机器人操作模式。

④ 急停按钮：用于在危险情况下关停机器人，急停按钮在被按下时将自行闭锁。

⑤ 空间鼠标（6D 鼠标）：用于手动移动机器人。

⑥ 移动键：用于手动操作机器人时，选择坐标轴和运动方向。

⑦ 有尼龙搭扣的手带：用于手动提起示教器，如果不使用手带，则手带可能完全被拉入。

⑧ 程序速度倍率调节键：用于调节机器人程序自动运行时的程序速度倍率，可选择的速度倍率为 1%、3%、10%、30%、50%、75%、100%。

图 3.3: smartPAD 示教器正面

⑨ 手动速度倍率调节键：用于调节机器人程序自动运行时的手动速度倍率，可选择的速度倍率为 1%、3%、10%、30%、50%、75%、100%。

⑩ 连接线。

⑪ 状态键：主要用于设定机器人备选软件包中的参数，其具体功能取决于所安装的备选软件包。

⑫ 启动键：用于启动机器人程序的运行。

⑬ 逆向启动键：用于逆向启动机器人程序。

⑭ 停止键：用于暂停正在运行的机器人程序。

⑮ 键盘按键：用于显示键盘，由于 smartHMI 可识别需要通过键盘输入的情况并自动显示键盘，故不必特地将键盘显示出来。

⑯ 主菜单键：用于在 smartHMI 上显示和隐藏主菜单，也可通过它创建屏幕截图。

3.2.3 smartPAD 背面

smartPAD 示教器背面的按键与开关如图3.4所示，其基本功能如下。

① 用于固定背带的按钮。

② 拱顶座支撑带。

③ 左侧拱顶座：用右手握 smartPAD。

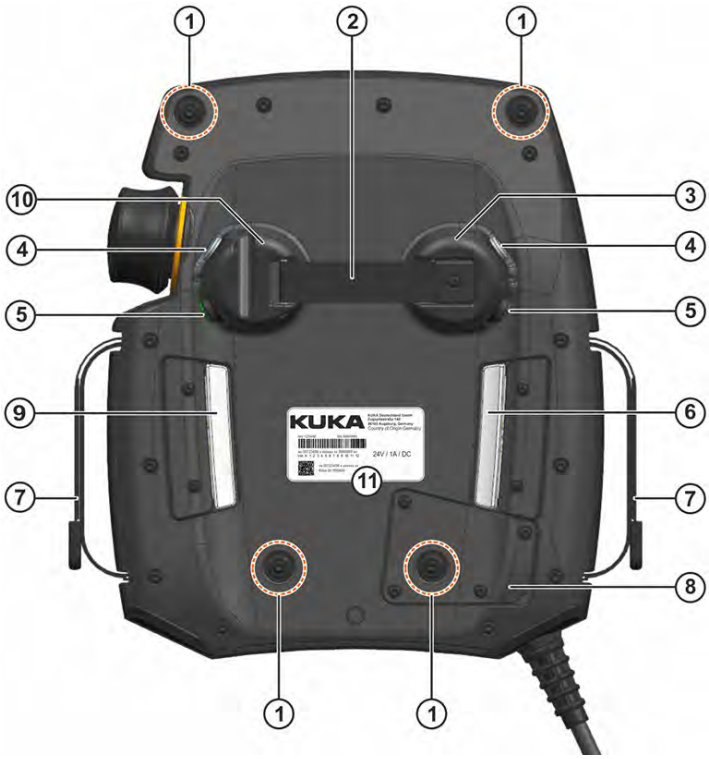


图 **3.4:** smartPAD 示教器背面

④ 操作确认开关：确认开关具有 3 个位置，分别为未按下、中间位置和完全按下（紧急位置）；在运行方式 T1（手动慢速运行方式）和 T2（手动快速运行方式）下，只有当确认开关保持在中间位置时，方可运行机械手；在采用自动运行模式和外部自动运行模式时，确认开关不起作用。

⑤ 启动键（绿色）。通过启动按键，可启动一个程序。

⑥ 、⑨ 手握确认开关：与操作确认开关的功能一致。

⑦ 有尼龙搭扣的手带：如果不使用手带，则手带可能完全被拉入。

⑧ 盖板（连接电缆的盖板）。

⑩ 右侧拱顶座：用左手握 smartPAD。

⑪ 型号铭牌。

3.3 KUKA 机器人基本操作与设定

3.3.1 KUKA 机器人的启动/关机和设定

1. KUKA 机器人控制系统的正常启动步骤

(1) 两步检查：

① 检查控制系统连接是否准确无误，如控制柜的电源输入电缆、示教器连接电缆、机器人连接电缆、保护接地线及其他部件的电气连接。

② 检查控制系统电源是否符合机器人使用说明书的规定，如电源输入容量。

图 3.5: 电源主开关 ON（开）位置

图 3.6: 自启动 KSS 软件

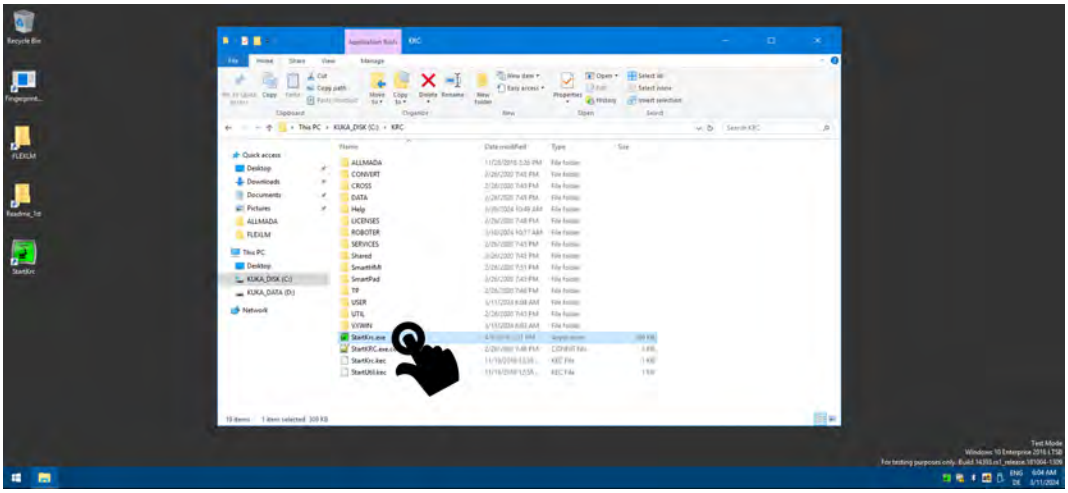
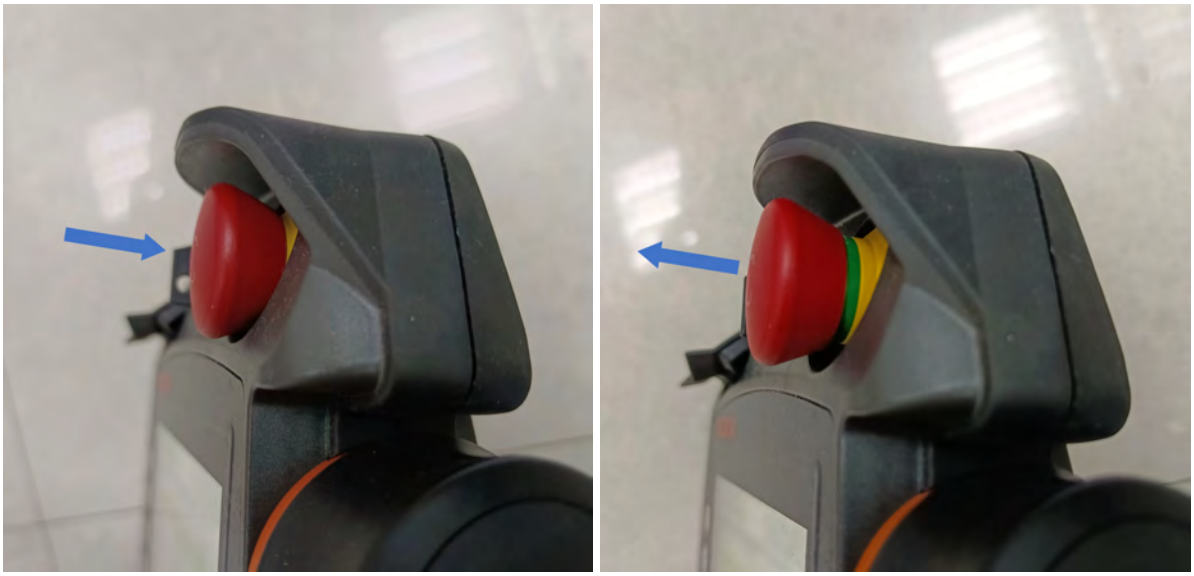


图 3.7: 双击 StartKRC.exe

(2) 开启机器人控制系统：

如图3.5，将机器人控制柜的电源主开关置于 ON（开）位置。在通常情况下，电源主开关 ON 后，控制系统便可启动并自动安装 KUKA 系统软件（KUKA System Software，KSS），如图3.6。如果控制系统的自动启动功能被禁止，KSS 不能自动安装与启动，可选择路径 C:/KRC，如图3.7双击 KSS 启动程序 StartKRC.exe，便可安装、启动 KSS，若机器人控制系统登录在网络上，则启动过程会较长。系统启动完成后，示教器（smartPAD）便可显示正常操作界面。



按下示教器急停按钮　　复位示教器急停按钮

图 3.8: 示教器急停按钮的使用

图 3.9: 单击 I 接通伺服驱动器状态主电源

(3) 启动伺服驱动系统

① 复位示教器的急停按钮，如图3.8(b)，将按钮左右扭一下即可轻松拔出。

② 利用示教器（smartPAD）操作，如图3.9，单击 I，若红框处的 I 显示绿色，则表示伺服驱动器已启动并正常运行。

启动伺服驱动系统后，便可按照规定的步骤对机器人进行正常操作。系统启动后，控制系统的初始状态与系统启动方式有关 (冷启动、休眠启动或重新启动控制系统 PC)，有关内容详见后述。

2. KUKA 机器人控制系统的正常关机步骤

图 3.10: 程序运行完成

(1) 两步确认：

① 确认程序已执行完成，如图3.10，R 显示为黑色即表示程序已运行完成。

② 确认机器人、作业工具、辅助部件等可动部件的运动均已停止、停止位置合适。

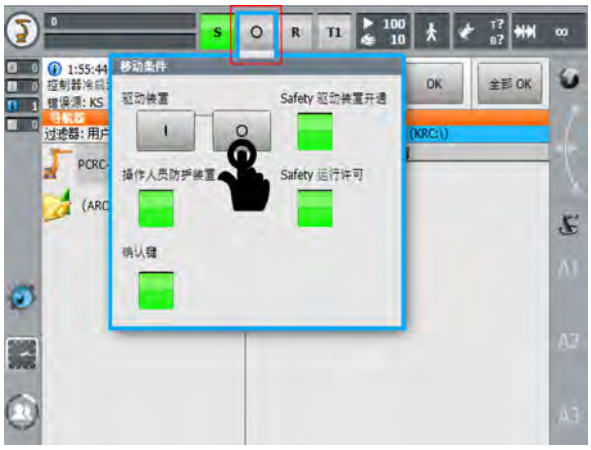


图 3.11: 单击 O 断开伺服驱动器状态主电源

图 3.12: 电源主开关 OFF（关）位置

(2) 利用示教器（smartPAD）操作，单击 O，若红框处的 O 显示灰色，则表示驱动器主电源断开，关闭伺服驱动系统。

(3) 将系统控制柜的电源主开关置于 OFF 位置，如图3.12。

注意：如果机器人控制系统在系统启动方式设定中选择了“重新启动控制系统 PC”操作，且重新启动尚未结束，则不得按主开关，必须等待机器人控制器重新启动完成，否则会损坏系统文件。

3. KUKA 机器人控制系统的启动方式总结

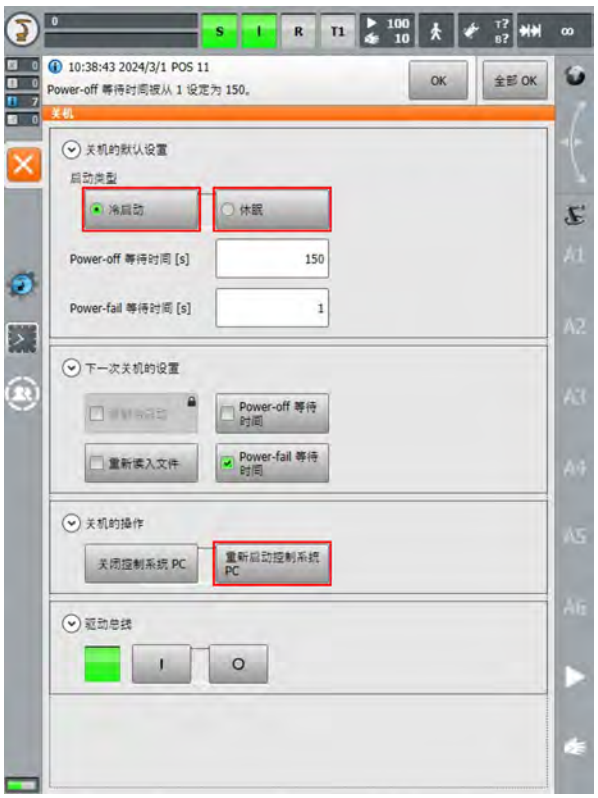


图 3.13: 系统启动方式设定

KUKA 机器人控制系统常用的开机方式有冷启动、休眠启动、重新启动控制系

统 PC、初始化启动 4 类，而具体要求控制系统以哪种方式启动则与如图3.13的设定有关。这 4 类启动方式的作用与操作步骤如下。

(1) 冷启动。冷启动（Cold start）是控制系统最常用的正常开机启动方式，是我们一般选择的启动方式，也是初始的启动方式，可直接利用接通电源主开关启动。控制系统以冷启动方式开机时，通常进行如下处理。

① 控制系统的全部输出（DO、AO 等）都被设置为 OFF（FALSE）状态。
② 程序自动“结束”运行，程序运行状态全部复位。
③ 速度倍率、坐标系等恢复为系统默认的初始值。
④ 冷启动完成后，示教器显示系统默认初始页面。

(2) 休眠启动。休眠启动方式的开机与个人电脑休眠状态的恢复类似，这是连续作业机器人的开机启动方式，也可直接利用接通电源主开关启动。控制系统以休眠启动方式开机时，将进行如下处理。

① 控制系统的全部输出（DO、AO 等）都恢复为电源断开时刻的状态。
② 如果电源断开，程序处于自动运行状态，则恢复断电时刻的程序状态，并自动生效程序暂停功能，程序自动运行可通过程序启动键继续。
③ 速度倍率、坐标系等设定均恢复为电源断开时刻的状态。
④ 休眠启动完成后，示教器可恢复断电时刻的显示页面。

(3) 重新启动控制系统 PC。该启动方式通常用于生效控制系统参数、清除不明原因的软件故障等。使用该方式进行重启时，机器人控制系统将重新安装 KSS 系统软件，然后以冷启动方式重新启动。

(4) 初始化启动。系统初始化启动将重新安装操作系统、恢复所有出厂设定，格式化用户存储器、清除全部用户数据，系统通常需要重新调试才能恢复正常工作，因此，操作一般需要由专业调试维修人员在更换硬件、重装操作系统时才能进行。

4. 系统启动方式设定操作与内容

系统启动方式设定的操作步骤如下：

(1) 如图3.14，按 smartHMI 状态显示栏的“用户组”图标，smartPAD 显示切换用户组的界面，选择“专家”或更高权限用户，输入密码 (如 kuka)，登录系统。

(2) 如图3.15，按 smartPAD 主菜单键或单击 smartHMI 状态显示栏的主菜单图标，smartPAD 显示图3.16所示的主菜单。单击主菜单“关机”，smartPAD 可显示图3.17所示的系统启动方式设定框。

(3) 根据需要选择设定项，完成系统启动方式设定。

KUKA 机器人系统启动方式设定内容：

KUKA 机器人控制系统的启动方式设定需要“专家”或更高权限的用户操作，可进行的设定如图3.17所示，设定项含义如下。

图 3.14: 更改用户组

图 3.15: 示教器主菜单键

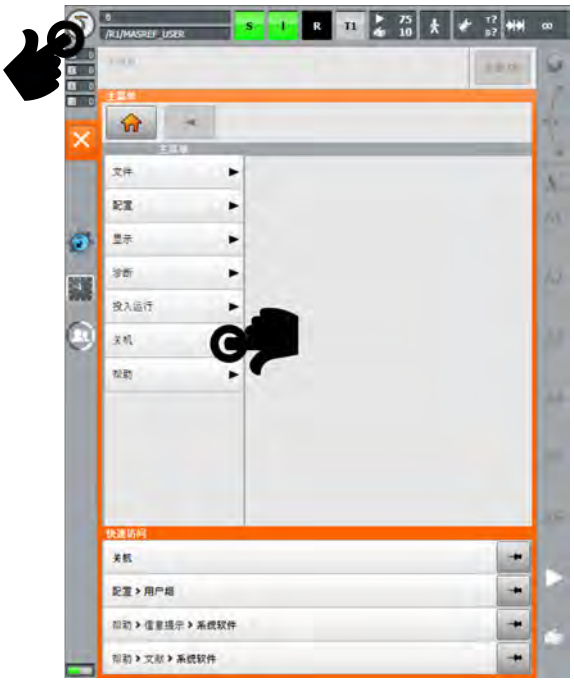


图 3.16: 示教器主菜单

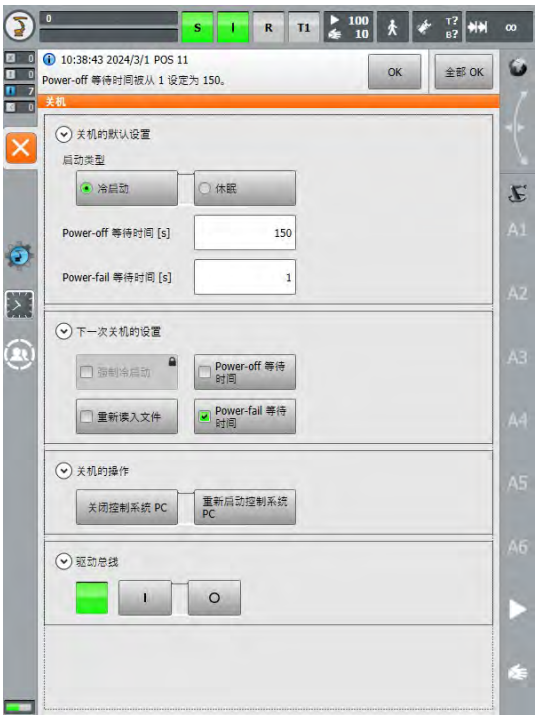


图 3.17: 系统启动方式设定

图 3.18: 关机的默认设置

(1) “关机的默认设置”栏用于启动方式及系统关闭延迟设定，其设定状态可一直保持有效, 直至再次被设定。设定项的含义如下。

① 冷启动：单击选择后，可使系统冷启动方式生效。

在下面的情况下，机器人控制系统始终执行初始的冷启动，这不取决于选择了哪个启动类型：

- ► 在安装或更新 KSS 后
- ► 如果机器人控制系统在关闭时已发现了错误

② 休眠：单击选择后，可使系统休眠启动方式生效。

③ Power-off 等待时间：如果机器人控制系统通过电源主开关关断，则系统的关闭延迟，在 Power-off 等待时间结束后才关闭；在此期间，机器人控制系统由蓄电池供电，用于电网的瞬时断电保护；Power-off 等待时间可通过软键盘输入调节，最大设定值为 240s。

注意：

- ► 只有当通过主开关断电时，Power-off 等待时间才有效；针对真正的断电情况，以 Power-fail 等待时间为准。
- ► 针对“KR C4 compact”控制器型式：Power-off 等待时间不起作用！通过主开关关断时，同样以 Power-fail 等待时间为准。

④ Power-fail 等待时间：断电时，机器人停止。然而机器人控制系统并不立即关闭，而是在 Power-fail 等待时间过后才关闭；因此短暂的断电可借助于这一等待时间被桥接，之后仅须确认故障信息并且程序可继续进行；在 Power-fail 等待时间过程中，机器人控制系统由蓄电池供电；Power-fail 等待时间对无可靠电源的设备尤为重要，等待时间最长可达 240 秒，调节方式与 Power-off 一致。

机器人控制系统	Power-fail 等待时间
“KR C4 compact”型	1 秒
KR C4 的所有其他类型	3 秒

图 3.19: 下一次关机的设置

(2) “下一次关机的设置”栏是只用于系统下次启动的一次性设定，设定状态只对设定完成后的系统第一次启动有效。设定项的含义如下。

① “强制冷启动”：若当前的启动方式选择了“休眠”，单击选择后，系统下次启动强制以“冷启动”方式启动；若当前启动方式为“冷启动”，则此项无效。

② “重新读入文件”：如果当前的启动方式为“冷启动”，或者在“休眠”方式下选择了“强制冷启动”，单击选择“重新读入文件”可将下一次启动方式更改为“初始化启动”。

③ “Power-off 等待时间”：单击可激活/取消下次关机时的 Power-off 等待时间。

④ “Power-fail 等待时间”：单击可激活/取消下次关机时的 Power-fail 等待时间。

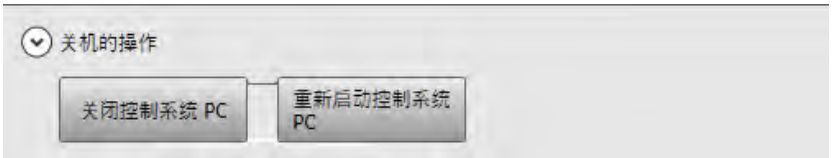


图 3.20: 关机的操作

(3) “关机的操作”栏仅对示教操作模式 T1、测试运行模式 T2 有效，所选操作可立即执行。设定项的含义如下。

① “关闭控制系统 PC”：机器人控制系统关机。

② “重新启动控制系统 PC”：机器人控制系统关机，然后以冷启动方式重新启动；选择“重新启动控制系统 PC”操作时，必须等待机器人控制系统重新启动完成后，才能关断电源主开关。

图 3.21: 驱动总线状态显示图标

(4) 驱动总线状态显示图标：绿色为驱动总线接通，红色为驱动总线关闭，灰色为驱动总线状态未知。

① 驱动总线“I”图标：单击可启动驱动总线通信。

② 驱动总线“O”图标：单击可中断驱动总线通信。

3.3.2 smartPAD 示教器的热插拔

1. 在系统通电状态取下 smartPAD

   (1) 按用来拔下 smartPAD 的按钮，如图3.22所示。smartHMI 上会显示热插拔提示信息和一个计时器。计时器会计时 30 秒，在此时间内可从机器人控制器上拔下 smartPAD。

   (2) 在 30 秒倒计时结束前，将示教器电缆连接器从系统控制柜中拔出、取下示教器。如果在计时器计时期间没有拔下 smartPAD，则此次计时失效。可任意多次按下用于拔下的按钮，以再次显示计时器。

   注意：如果在计数器未运行的情况下取下 smartPAD，会触发紧急停止。只有重新插入 smartPAD 才能取消紧急停止。

图 3.22: 按下 smartPAD 热插拔按钮

2. 在系统通电状态插入 smartPAD

   ① 确认 smartPAD 型号与规格准确。

   ② 将 smartPAD 连接电缆插入系统控制柜的连接器上。将 smartPAD 插入 30 秒后，紧急停止和确认开关再次恢复功能，且自动重新显示 smartHMI。（可能需要 30 秒以上）插入的 smartPAD 会应用机器人控制器的当前运行方式。

   注意：当前运行方式并不总是与拔出 smartPAD 之前相同：如果是一个 RoboTeam 的机器人控制系统，则运行方式可能在拔出之后发生变化，例如通过主设备（Master）。

   警告：将 smartPAD 插在机器人控制器上的用户，之后至少必须在 smartPAD 旁停留 30 秒，就是说直到紧急停止和确认开关再次恢复正常功能。这样就可避免出现另一操作人员在紧急情况下使用紧急停止装置暂时无效的情况。

3.3.3 smartHMI 操作界面与信息显示

1. smartHMI 操作界面

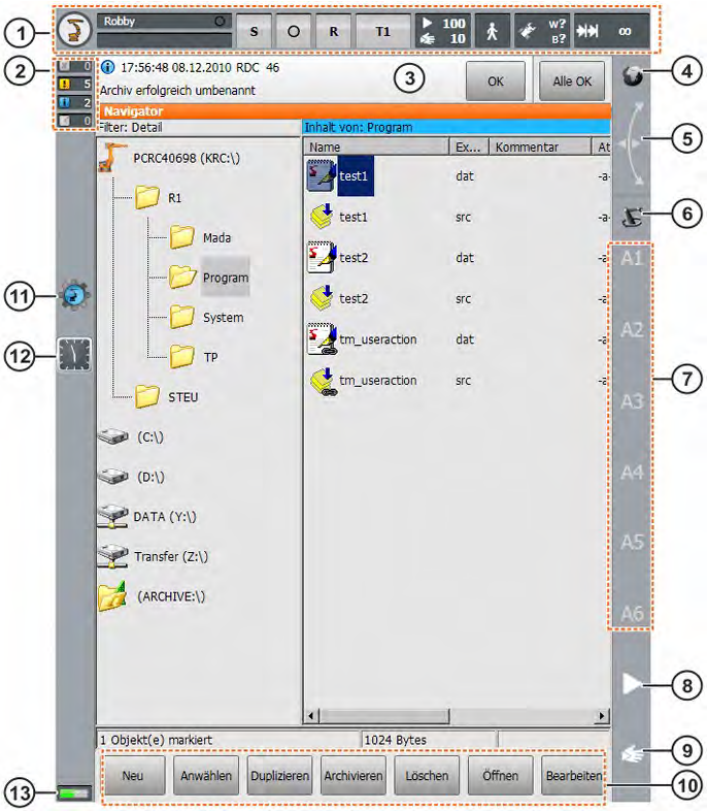


图 3.23: KUKA smartHMI 操作界面

① ——状态栏。可显示机器人名称、作业程序名称，以及控制系统、伺服驱动器、机器人工作状态显示和操作的图标与按键。

② ——信息提示计数器。可显示控制系统未处理的各类系统信息及数量，单击图标可显示“现有信息”显示窗，显示各类信息的名称以及数量，触摸提示信息计数器可放大显示。

③ ——信息窗口。根据默认设置将只显示最后一条提示信息，触摸提示信息窗口可放大该窗口并显示所有待处理的提示信息；可以被确认的信息可用 OK 键确认，所有可以被确认的信息可用全部确认键一次性全部确认。

④ ——空间鼠标状态显示。其会显示用空间鼠标手动移动的当前坐标系，触摸该显示就可以显示所有坐标系并可以选择另一个坐标系。

⑤ ——空间鼠标定位显示。触摸该显示会打开一个显示空间鼠标当前定位的窗口，在窗口中可以修改定位。

⑥ ——运行键状态显示。可显示用运行键手动移动的当前坐标系，触摸该显示就可以显示所有坐标系并可以选择另一个坐标系。

⑦ ——运行键标记。如果选择了与轴相关的移动，这里将显示轴号（A1、A2 等）；如果选择了笛卡尔式移动，这里将显示坐标系的方向（*X*、*Y*、*Z*、*A*、*B*、*C*）；触

摸则会显示选择了哪种运动系统组。

⑧ ——编程速度倍率调节。单击图标，可打开/关闭机器人程序自动运行的编程速度倍率（POV）微调按钮，以 1% 的增量微调速度倍率；单击显示区以外的区域，可关闭倍率微调按钮、速度倍率生效；编程速度倍率也可通过编程速度倍率调节键（smartPAD 辅助操作键，见前述）选择 100%、75%、50%、30%、10%、3%、1%。

⑨ ——手动速度倍率调节。单击图标，可打开/关闭机器人手动操作的速度倍率（HOV）微调按钮，以 1% 的增量微调速度倍率；单击显示区以外的区域，可关闭倍率微调按钮、速度倍率生效；手动速度倍率也可通过手动速度倍率调节键（smartPAD 辅助操作键，见前述）选择 100%、75%、50%、30%、10%、3%、1%。

⑩ ——按键栏。这些按钮自动进行动态变化，并总是针对 smartHMI 上当前激活的窗口，用于对当前页面进行输入、编辑和显示等操作；最右侧是按钮编辑，用这个按钮可以调用导航器的多个命令。

⑪ ——WorkVisual 图标。通过触摸图标可至窗口项目管理。

⑫ ——时钟。时钟显示系统时间，触摸时钟就会以数码形式显示系统时间以及当前日期。

⑬ ——显示存在信号。如果左侧和右侧小灯交替发绿光，且交替缓慢（约 3 秒）而均匀闪烁，则表示 smartHMI 激活。

2. 信息显示

(1) 信息提示是控制系统自动生成的操作提醒，单击示教器的“信息提示”图标，可显示图3.24所示的“现有信息”显示窗，显示当前各类信息的名称及数量。选定信息类别，可在信息显示区显示信息文本及操作确认键。

图 3.24: “现有信息”显示窗 1

KUKA 机器人“现有信息”显示窗的显示图标、信息类别、信息性质及需要操作者进行的操作如表3.1所示。

选定“现有信息”显示窗的信息类别，示教器的信息显示窗可显示图3.25所示的最近一条系统信息，并进行相关处理。单击信息显示空白区，可展开显示其他未处理的信息，单击第一行，可重新收拢显示区。

(2) 信息显示窗右侧的图标用于信息详情显示与确认，如图3.25。

① 〖?〗：可打开当前显示信息的帮助文件、显示详情，有关内容可参见后述。

② 〖OK〗：当前信息确认按钮，单击可确以当前信息提示。

表 3.1: 信息显示图标、信息类别、信息性质及需要操作者进行的操作

图标		信息类别	信息性质	需要操作者进行的操作
形状	颜色			
	红	确认信息	中断操作，导致机器人停止，并禁止机器人启动	需要操作者利用正确的操作进行确认
	黄	状态信息	系统状态显示	需要改变系统的工作状态解除
	蓝	操作提示信息	对操作者的操作提示	可按提示要求进行操作
	绿	等待信息	系统等待内容显示	等待条件满足，或利用“模拟”键解除

图 3.25: “现有信息”显示窗 2

③〖全部 OK〗：全部信息确认按钮，可一次性确认全部信息提示。

3.3.4 主菜单、显示语言、屏幕截图与用户等级

1. 主菜单

smartPAD 采用触摸屏菜单操作方式，大多数功能需要使用菜单选择、打开及操作。机器人控制系统启动、smartHMI 操作界面正常显示后，按图3.26所示的 smartPAD 主菜单键或单击 smartHMI 状态显示栏的主菜单图标，示教器便可显示图3.27所示的 smartPAD 操作主菜单。主菜单显示页可显示的内容与操作图标如下。

① 主菜单关闭。单击可关闭主菜单显示页面。

② HOME 操作。单击可显示所有已打开的下级菜单。

③ 返回。单击可返回上级菜单。

④ 子菜单打开。单击可打开子菜单。

⑤ 下级菜单打开。单击可继续打开下一级子菜单（如存在）。

⑥ 已打开的下级菜单（最大 6 项），单击可直接显示已打开的下级菜单。

2. 设定操作界面的语言

smartHMI 操作界面可根据需要，选择德文、英文、中文、法文、西班牙文等 20 多种语言显示与操作。

操作步骤：

图 3.26: 主菜单键的选择

图 3.27: 系统操作主菜单

(1) 如图3.26，按 smartPAD 主菜单键或单击 smartHMI 状态显示栏的主菜单图标，smartPAD 显示图3.27。

(2) 单击主菜单中的配置 » 其它 » 语种。

(3) 单击“语种”，选中所需的语言。

(4) 单击“OK”键确认，即可将操作界面切换为所选择的语言。

3. 屏幕截图

在 smartPAD 上创建屏幕截图的操作步骤如下：

■ 按下 smartPAD 左上方的主菜单键 2 次，如图3.29，将屏幕截图保存到文件夹 C:/KUKA/Screenshot 中。

图 3.28: 双击主菜单键

4. 用户等级

KUKA 系统软件（KSS）针对不同的操作者（即用户组），设定相应的操作权限，机器人安装调试完成后，调试人员可根据操作者的层次，设定相应的密码和分配系统操作权限。KUKA 机器人出厂默认所有用户的密码均为“kuka”。

(1) KSS 的操作权限可由低到高设定 6 级，操作人员、用户为系统默认的操作权限，可直接操作。专家及以上级用户需要通过密码登录，高级用户可覆盖低级用户的全部权限，不同用户组具有不同的可用功能。共有下列用户组：

① 操作人员：此为默认用户组，只能进行最基本的操作和编程。

② 用户：机器人使用厂家的一般操作人员，默认情况下操作人员和用户的操作权限是一样的。

③ 专家：编程人员用户组，是机器人程序设计人员，可进行高层次的操作与编程；此用户组有密码保护，需要输入正确的专家密码才能登录系统。

④ 安全维护人员：机器人调试、维修人员，该用户组可以激活和配置机器人的安全配置，在专家级的基础上增加了机器人安全保护设定功能；该用户组有密码保护，需要输入正确的安全维护密码才能登录系统。

⑤ 安全调试员：机器人设计人员，在安全维护级的基础上增加了机器人安全操作（KUKA.Safe Operation）、安全范围监控（KUKA.Safe Range Monitoring）功能，只在进行上述安全操作时，该用户组才相关；该用户组有密码保护，需要输入正确的安全调试密码才能登录系统。

⑥ 管理员：机器人设计人员，功能与专家用户组一样；另外可以将插件 (Plug-Ins) 集成到机器人控制系统中；该用户组有密码保护。默认密码为“kuka”。

(2) 更改用户组的操作步骤：

① 按 smartPAD 主菜单键或单击 smartHMI 状态显示栏的主菜单图标，smart-PAD 显示图3.27。

② 单击主菜单中的配置 » 用户组，smartPAD 可显示当前用户级别。

③ 若欲切换至默认用户组，则单击“登录…”图标，可显示用户组选择框、选择用户等级。单击“标准”图标，可恢复系统默认的用户等级；若欲切换至其它用户组，则单击“登录…”图标，选定所需的用户组。

④ 如果需要，输入密码并确认，所选的用户等级将生效。

图 3.29: 双击主菜单键

(3) 切换为默认用户组的几种情况：

① 重启时将选择默认用户组。

② 如果要切换至 AUT （自动）运行模式或 AUT EXT（外部自动运行）运行模式，则机器人控制系统将出于安全原因切换至默认用户组；如果希望选择另外一个用户组，则须此后进行切换。

③ 如果在一段固定时间内未在操作界面进行任何操作，则机器人控制系统将出于安全原因切换至默认用户组。此时间段默认设置为 300 秒。

(4) 更改用户组密码的操作步骤：

① 单击主菜单中的配置 » 用户组，smartPAD 可显示当前用户级别。

② 单击“登录…”图标，选中要更改其密码的用户组。

③ 单击“密码…”，输入旧密码，并输入新密码两次；出于安全考虑，输入内容均会加密显示；密码区分大小写。

④ 单击“OK”键，新密码立刻生效。

3.3.5 操作模式与安全防护

1. 操作模式

机电一体化设备的运行控制方法称为“操作模式”。工业机器人的操作模式通

常有示教 1（T1）、示教 2（T2）、自动（AUT）、外部自动（EXT AUT）4 种，但在不同公司生产的机器人上，操作模式的功能稍有区别，KUKA 机器人的操作模式与功能如下。

(1) 示教 1（T1）。T1 模式称为手动低速运行模式，可用于机器人手动操作、示教编程及程序低速试运行。选择 T1 模式时, 手动操作或程序自动运行的机器人 TCP 运动速度均被限制在 250mm/s 以下。

(2) 示教 2（T2）。T2 模式又称手动快速运行模式，KUKA 机器人的 T2 模式只能用于程序高速试运行，不能用于机器人手动操作和示教编程。选择 T2 模式时，程序可按编程速度运行，机器人 TCP 可达到最大移动速度。

(3) 自动（AUT）。AUT 模式用于机器人作业程序的再现（Play）自动运行，不能用于机器人手动操作和示教编程。KUKA 机器人的 AUT 模式属于“本地运行”，自动运行程序需要通过示教器选择、启动。

(4) 外部自动（EXT AUT）。EXT AUT 模式用于上级控制器控制的程序自动运行，又称“远程运行（REMOTE）”。EXT AUT 模式同样不能用于机器人手动操作和示教编程，自动运行的程序选择、启动需要由控制系统的 DI 信号控制。

2. 安全防护

工业机器人是一种多自由度的自动化设备，为了确保操作人员人身及设备安全，需要根据不同的操作模式，设置相应的安全防护装置。

工业机器人的安全防护装置主要有急停按钮、手握确认开关（包括操作确认按钮）、安全栅栏等。急停按钮、手握确认开关是任何机器人必备的基本安全防护装置，至少需要在示教器上安装，smartPAD 示教器的手动操作保护开关安装可参见（控制柜部分的介绍，X11）。

不同操作模式下的 KUKA 机器人安全保护装置如表3.2所示，安全保护功能如下。

表 3.2: KUKA 机器人的安全保护装置

保护装置	安全保护动作	机器人操作模式			
		T1	T2	AUT	EXT AUT
急停按钮	机器人急停、驱动器主电源断开	有效	有效	有效	有效
手握确认开关（中间）	机器人减速停止、伺服启动信号撤销	有效	有效	无效	无效
手握确认开关（按下）	机器人急停、驱动器主电源断开	有效	有效	无效	无效

(1) T1 模式。安全栅栏防护门开关无效，驱动器的伺服启动（伺服 ON）信号由示教器背面的手握确认开关或操作确认按钮控制（参见3.2节、图3.4），手握确认开关握至中间位置并保持，系统可输出伺服启动信号、控制机器人运动，松开手握确认开关，伺服启动信号撤销、机器人减速停止；按下手握确认开关，机器人急停、驱动器主电源断开。

(2) T2 模式。安全栅栏防护门开关无效，机器人同样由示教器的手握确认开关或

操作确认按钮控制伺服启动、机器人急停。但是，T2 模式的机器人需要以最大速度高速运动，存在碰撞和干涉的危险，因此，应由专业操作人员在确保人身设备安全的前提下实施。

(3) AUT 模式。示教器手握确认开关无效，机器人原则上应设置安全栅栏，利用防护门开关控制机器人自动运行、紧急停止。AUT 模式的机器人同样需要以最大速度高速运动，对于未安装安全栅栏的机器人，必须由专业操作人员在确保人身设备安全的前提下实施。

(4) EXT AUT 模式。示教器手握确认开关无效，机器人必须设置安全栅栏，利用防护门开关控制机器人自动运行、紧急停止。

3. 操作模式选择

采用 smartPAD 示教器的 KRC4 工业机器人控制系统无专门的操作模式选择开关，机器人操作模式切换需要在机器人解锁后，利用示教器操作实现，其操作步骤如下。

(1) 确认机器人的自动运行已经结束，可动部件均已停止运动。

(2) 如图3.30(a)所示，将机器人解锁钥匙插入 smartPAD 示教器的操作模式切换旋钮上，并将旋钮旋转至“解锁”位置。此时，smartPAD 示教器可显示图3.30(b)所示的机器人操作模式选择框。

(3) 在操作模式选择框中，单击选定所需要的操作模式。

(4) 将机器人解锁钥匙旋回至“闭锁”位置、取下钥匙。

(5) 在图3.30(c)所示的状态栏显示图标上，确认操作模式已生效。

3.3.6 状态栏图标及操作

1. 状态显示

smartHMI 状态显示栏的显示如图3.31所示。

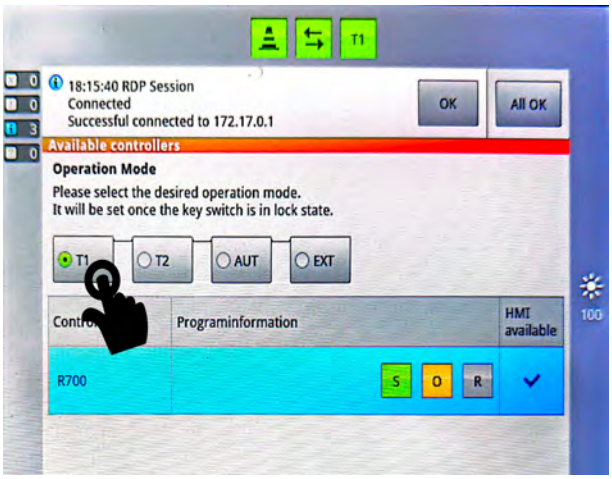
图 3.31: smartHMI 状态显示栏

状态显示栏左起第 1 个图标为 smartPAD 主菜单显示图标，作用与 smartPAD 主菜单按键相同，单击可打开图3.27所示的 smartPAD 主菜单。

状态显示栏左起第 2 个位置的第 1 行为机器人名称显示。机器人名称可通过主菜单“投入运行”→ 子菜单“机器人数据”，在机器人数据设定页面进行设定与修改。

状态显示栏左起第 2 个位置的第 2 行为当前程序名称显示。选择程序编辑、自动运行等操作时，此区域可显示当前选择的编辑、自动运行程序名称。

解锁



选择模式

确认状态

图 3.30: 操作模式选择

状态显示栏右侧区域为控制系统状态显示与操作图标，其内容如下。

2. 后台处理状态显示与操作

KUKA 机器人控制系统具有前后台程序并行处理功能，前台处理用于机器人运动、I/O 信号输入输出处理，KUKA 说明书译作“机器人解释器”；后台处理用于变量读写、I/O 缓冲器状态读写非同步附加轴运动等辅助指令处理，KUKA 说明书中译为“提交解释器”“SUBMIT 解释器”“控制解释器”等。

系统后台处理的状态，在 smartPAD 上以带 S 的图标表示，单击图标可打开图3.32所示的后台处理操作选项，选择、启动或停止、取消后台处理功能。

后台处理操作选项也可通过图3.27所示的 smartPAD 主菜单“配置”→“SUBMIT 解释器”打开与选择。

后台处理状态显示图标的颜色含义如下。

① 绿色：已选择并启动后台处理功能（正常状态）。

② 红色: 后台处理已停止，需要重新启动。

③ 黄色：已选择程序，但后台处理未启动，执行指针位于程序的起始位置。

④ 灰色：未选择后台处理功能。

3. 驱动器状态显示与操作

控制系统的伺服驱动器状态以带 I 或 O 的图标表示，单击图标可打开图3.33所示的驱动器通/断控制及启动条件显示框（KUKA 说明书中译为“移动条件”），并进行驱动器主电源接通或断开操作。

(1) 伺服驱动器状态显示图标的含义如下。

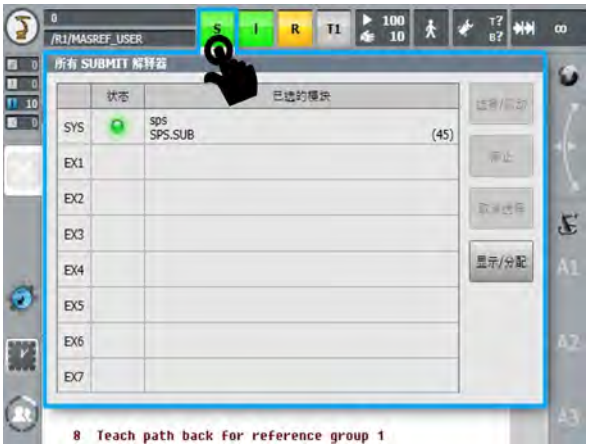


图 3.32: 后台处理状态显示与操作

图 3.33: 伺服驱动器状态显示与操作

① I（绿色）：伺服驱动器已启动并正常运行。驱动器主电源已接通、直流母线充电完成，伺服启动（伺服 ON）信号已接通（处于中间位置），系统无报警，机器人可正常移动。

② I（灰色）：伺服驱动器主电源接通、直流母线充电完成，但是伺服启动（伺服 ON）信号未接通，例如，机器人手动操作时，手握确认开关或操作确认按钮不在中间位置，或者系统存在报警，伺服启动信号被断开、机器人

不能运动。

③ O（灰色）：驱动器主电源断开或直流母线充电未完成，机器人不能运动。

(2) 驱动器通/断控制及启动条件显示框的显示内容如下。

① 驱动装置：单击“I”可接通伺服驱动器主电源；单击“O”可断开伺服驱动器主电源。

② Safety 驱动装置开通：驱动器急停信号状态显示，绿色代表急停信号正常接通、允许接通驱动器主电源；灰色代表急停信号断开、系统处于 STOP 0 或 STOP 1 状态，禁止接通驱动器主电源。

③ Safety 运行许可：驱动器伺服使能状态指示，绿色代表驱动器的伺服使能信号正常、机器人可以正常运动；灰色代表伺服启动信号断开、系统处于 STOP 1 或 STOP 2 状态，机器人不能运动。

④ 操作员防护装置：安全栅栏等机器人安全保护信号状态指示，绿色代表机器人手动操作（模式 T1、T2）时，已接通手握确认开关（或操作确认按钮）的伺服启动信号，或者程序自动运行时（模式 AUT）安全栅栏等防护装置已关闭，允许机器人运动；灰色代表以上条件不具备、不允许机器人运动。

⑤ 确认键：手握确认开关（或操作确认按钮）信号状态指示，绿色代表手握确认开关（或操作确认按钮）的伺服启动信号已接通（处于中间位置）；灰色代表手握确认开关（或操作确认按钮）尚未接通，或者当前操作无须接通手握确认开关（或操作确认按钮）。

4. 前台处理状态显示与操作

控制系统的前台处理状态以带 R 的图标表示（KUKA 说明书中译为“机器人解释器”），单击图标可打开图3.34所示的程序选择（或取消程序选择）、程序复位操作选项，用于选择、取消或复位 KRL 程序, 程序选择的方法详见（问李世龙谢涛：8.4.1）节。

图 3.34: 程序执行状态显示与操作

前台处理状态显示图标的颜色含义如下。

① 黄色：程序已选择，程序执行指针（光标）位于程序开始位置。

② 绿色：已选择程序，并启动前台处理。

③ 红色：程序已选择，前台处理处于停止状态。

④ 黑色：程序已选择，但程序执行指针（光标）位于程序结束位置。

⑤ 灰色：程序未选择。

5. 操作模式显示

控制系统当前操作模式以带代号的图标显示，T1 为手动示教，T2 为程序试运行，AUT 为程序自动运行，EXT 为外部自动运行。操作模式的切换方法可参见前述。

6. 速度倍率显示与调节

控制系统当前的速度倍率以倍率图标的形式显示，单击图标可打开图3.35所示的速度倍率调节框。单击“+”“-”键或移动滑移调节图标，可改变编程速度倍率（程序调节量）、手动速度倍率（手动调节量）的值。

单击图3.35中的“选项”图标，可进一步打开机器人手动移动选项设定框，进行手动操作条件设定，有关内容详见后述的机器人手动操作说明。

7. 程序运行方式显示与操作

控制系统当前的程序运行方式以图标的形式显示，单击图标可打开图3.36所示的程序运行方式选择框，改变程序运行方式，

图 3.35: 速度倍率调节框

图 3.36: 程序运行方式显示与选择

程序运行方式显示图标及含义如下。

：连续执行，正向连续执行程序。

：机器人单步，正向、单步执行机器人移动指令，到达程序点自动暂停。

：指令单步，正向、单步执行全部指令 (需要有专家级以上操作权限)。

单击“选项”图标，同样可打开机器人手动移动选项设定框，进行机器人手动操作条件的综合设定，有关内容详见后述的机器人手动操作说明。

8. 坐标系显示与选择

控制系统当前生效的坐标系以图标的形式显示, 单击图标可打开图3.37所示的坐标系选择框，改变工具坐标系、工件坐标系及工具安装方式。

工具坐标系、工件坐标系可通过单击输入选择键选择，如果工具坐标系、工件坐标系未设定，坐标系名称显示“未知”、编号显示“[?]”。工具安装方式可通过单击“法兰”“外部工具”选定，选择“法兰”时，工具安装方式为机器人移动工具作业，选择“外部工具”时，工具安装方式为机器人移动工件作业。单击“选项”

图标，同样可打开机器人手动移动选项设定框，进行机器人手动操作条件的设定，有关内容详见后述的机器人手动操作说明。

9. 手动增量显示与设定

机器人手动增量移动以图3.38所示的图标显示，单击图标可打开手动增量操作设定框，改变移动方式和手动增量值。

选择“持续的”时，机器人为手动连续移动（JOG），选择 0.1mm/0.005°、1mm/1°、10mm/3°、100mm/10°（角度用于关节坐标系、工具定向增量移动）时，机器人为手动增量操作。

图 3.37: 坐标系显示与选择

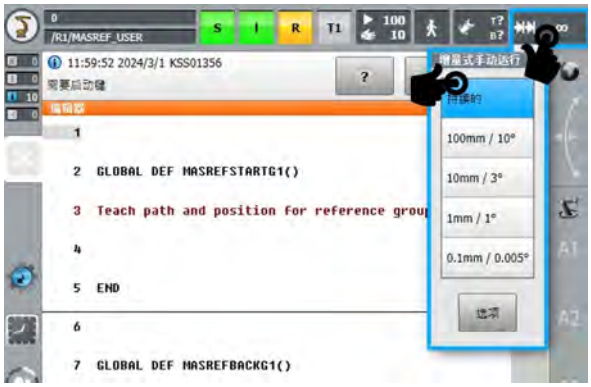


图 3.38: 手动增量显示与设定

3.3.7 信息的在线文献和帮助

1. 调出在线文献

KUKA 机器人的在线文献是可以在机器人控制系统上显示系统软件的文献，有些工艺数据包也同样具有可在机器人控制系统上显示的文献。调出在线文献的操作步骤如下：

(1) 按 smartPAD 主菜单键或单击 smartHMI 状态显示栏的主菜单图标，smartPAD 显示图3.27。

(2) 单击主菜单中的帮助 » 文献 » 系统软件/EthernetKRL。

(3) 单击所需类别，如单击“系统软件”，smartPAD 即显示图3.39所示的帮助文献，可通过页面、章节选择按钮进行选择和阅读相关内容。

2. 调出信息的帮助

KUKA 机器人的信息提示帮助文件可通过主菜单打开，或者通过信息显示窗直接选择、打开当前显示信息的帮助文件。

利用主菜单打开、阅读信息提示文件的操作步骤如下：

(1) 按 smartPAD 主菜单键或单击 smartHMI 状态显示栏的主菜单图标，smartPAD 显示图3.27。

(2) 单击主菜单中的帮助 » 信息提示 » 系统软件/EthernetKRL。

(3) 单击所需类别，如单击“系统软件”，smartPAD 即显示该类提示信息的目录。

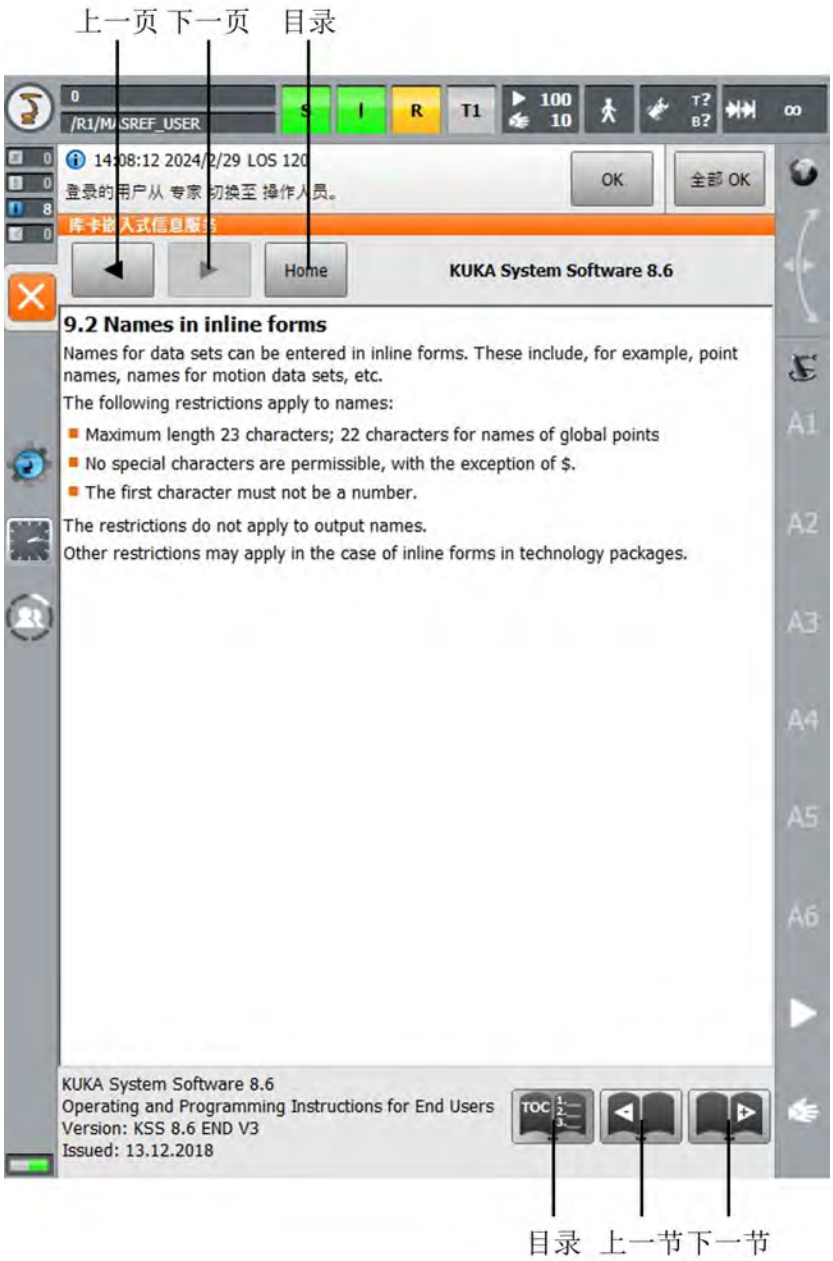


图 **3.39:** KUKA 机器人帮助文献

信息提示帮助文件　　详细说明文件

图 3.40: 信息提示详细说明文件

(4) 单击提示信息的目录，smartPAD 即可显示如图3.40(a)所示的信息提示帮助文件，检查该提示信息对系统运行的影响、类别及产生原因、解决办法。

(5) 单击信息提示帮助文件显示页的页面选择、目录图标，可切换显示内容；单击“详细显示”图标，可继续显示图3.40(b)所示的详细说明文件，查看提示信息产生的原因与解决办法的详细说明。

3.3.8 手动运行机器人

机器人手动操作又称 JOG 操作，它是利用示教器手动控制机器人、外部轴移动的一种方式。机器人手动操作可在关节坐标系、笛卡儿坐标系（工件、工具等）中进行，但外部轴移动一般只能在关节坐标系中进行。机器人手动操作前，首先应通过示教器操作，完成机器人手动移动选项设定，然后选择手动模式、启动伺服、移动机器人。手动操作的步骤参见后述，手动移动选项设定方法如下。

1. 手动移动选项设定显示

KUKA 机器人的手动移动选项设定用于机器人手动操作参数的综合设定，其中的部分参数也可直接通过对应的 Smart PAD 状态显示栏设定（见3.3.5节），两者作用相同。

手动移动选项设定可通过单击打开图3.41(a)所示的状态显示栏图标（S、I、R

除外）后，再单击“选项”图标打开，其基本显示如图3.41(b)所示。

手动移动选项设定的显示区 ① 可显示手动设定的项目标签, 显示区 ② 可显示所选项目的设定项，并进行相关设定。

smartPAD 手动移动选项打开后，可自动显示图3.41(b)所示的基本设定页面，并进行速度倍率、程序运行方式的显示与设定，显示页也可通过单击“概述”标签打开，其设定内容可参见3.3.5节。其他设定定标签的设定内容如下。

选择　　显示

图 3.41: 手动移动选项设定显示

2. 手动方向键设定

手动移动选项设定标签“按键”用于如图3.42(a)所示的 smartPAD 示教器手动方向键的功能设定。单击手动移动选项设定标签“按键”，可显示图3.42(b)所示的设定选项，并进行如下设定。

(1) 激活按键。单击图标可生效/取消（接通/关闭）smartPAD 示教器手动方向键。对于机器人本体，当手动方向键有效时，机器人可通过图3.42(a)所示的手动方向键进行手动移动，手动方向键取消时，机器人只能通过后述的 3D 鼠标进行手动移动。如果手动方向键和 3D 鼠标同时有效，则在鼠标操作和机器人移动停止后，机器人可利用方向键移动，方向键操作和机人移动停止后，机器人可利用 3D 鼠标移动。

变位器等附加轴的手动移动只能通过方向键操作，不能用 3D 鼠标手动移动，因此，附加轴动必须生效手动方向键。

(2) 增量式手动移动。可选择方向键操作时的机器人移动方式及增量移动的距离，单击输入选项图标 (显示区右侧向下箭头)，可显示如下移动方式及增量移动距离输入选项，并单击选定。选项也可直接单击“手动增量显示与设定”图标打开、设定 (参见3.3.5节)。

① “持续的”：方向键用于机器人手动连续移动 (JOG) 运动轴及方向选择。

② 0.1mm/0.005°、1mm/1°、10mm/3°、100mm/10°: 方向键用于机器人手动增量移动操作，设定按一次方向键的手动增量移动距离 (角度用于关节坐标

手动方向键

设定选项

图 3.42: 手动方向键设定

系、工具定向)。

(3) 运动系统组。用于方向键操作的控制轴组选择（参见（问李世龙谢涛：6.1.1）节）。可选择的控制轴组有“机器人轴”“附加轴”及“外部运动系统组”“用户定义的运动系统组”。方向键用于机器人手动操作时，应选择“机器人轴”；用于变位器等附加轴（E1 E6 轴）控制时，应选择“附加轴”；“外部运动统组”“用户定义的运动系统组”一般只用于特殊机器人系统操作。

(4) 坐标系统。用于机器人手动方向键操作的坐标系选择，选择“轴”“全局”“基坐标”“工具”图标，分别为关节（Joint）、大地（World）、工件（Base）、工具（Tool）坐标系移动。图标“同步”用于手动方向键、3D 鼠标坐标系的同步控制，未选择该图标时，方向键操作和 3D 鼠标可以选择不同的坐标系；选择该图标时，方向键操作和 3D 鼠标的手动操作坐标系自动同步，更改手动方向键坐标系时，3D 鼠标坐标系也将被同时修改。

3. 3D 鼠标设定

手动移动选项设定标签“鼠标”用于图3.43(a)所示的 smartPAD 示教器 3D 鼠标设定。单击手动移动选项的设定标签“鼠标”，示教器可显示图3.43(b)所示的设定框，并进行如下设定。

3D 鼠标

设定显示

图 3.43: 3D 鼠标设定

(1) 激活鼠标。单击图标可生效/取消 (接通/关闭)smartPAD 示教器的 3D 鼠标，参见上述“手动方向键设定”说明。

(2) 鼠标设置。用于 3D 鼠标的操作功能设定，详见后述。

(3) 坐标系统。用于机器人 3D 鼠标手动操作的坐标系选择，含义与手动方向键相同，参见上述“手动方向键设定”说明。

4. 示教器操作方位设定

手动移动选项设定标签“Kcp 项号”用于示教器操作方位设定，使得 3D 鼠标的操作方向与机器人实际移动方向一致，有关内容详见后述。

5. 工件/工具坐标系设定

手动移动选项设定标签“激活的基坐标/工具”用于机器人手动操作的工件坐标系、工具坐标系选择，单击标签“激活的基坐标/工具”，可显示图3.44所示的工件坐标系、工具坐标系的设定选项，并进行如下设定。

(1) 工具选择。单击工具选择下拉列表中的选项，可显示系统现有的工具坐标系 (STOOL) 名称、编号，并单击选定。系统出厂默认的工具坐标系为“无效 ($NULLFRAME)”、坐标系编号为 [0] 即工具坐标系与手腕基准坐标系重合。如果编号显示“[?]”，表明工具坐标系尚未被设定。

(2) 基坐标选择。单击“基坐标选择”下拉列表中的选项, 可显示系统现有的工件坐标系 ($BASE) 名称、编号，并单击选定。系统出厂默认的工件坐标系为“无效 ($NULLFRAME)”、坐标系编号为 [0]，即工件坐标系与机器人基座坐标系重合。如果编号显示“[?]”代表工件坐标系未设定。

(3) Ipo 模式选择。用于工具安装方式（插补模式）设定，选择图标“法兰”时，为机器人移动工具作业 (SIPO_MODE=$BASE)，选择图标“外部工具”工具安装方式为机器人移动工件作业 ($IPO MODE=$TCP)。

图 3.44: 工件坐标系、工具坐标系设定

6. 3D 鼠标操作设定

(1) 3D 鼠标操作

KUKA 示教器的 3D 鼠标可进行图3.45(a)所示的前后、上下、内外 3 个方向的直线移动，以及图3.45(b)所示回绕中心点的 3 个方向偏转或倾斜，因此，

3D 鼠标最大可以用于 6 轴手动操作。

移动　　偏转

图 3.45: 3D 鼠标操作

3D 鼠标操作与机器人 TCP 笛卡儿坐标系运动轴、方向的对应关系，以及鼠标功能，需要通过示教器操作方位、鼠标功能设定，其设定内容、操作步骤如下。

(2) 示教器操作方位设定

示教器操作方位设定的目的是在机器人进行 TCP 笛卡儿坐标系手动移动时，可以使 3D 鼠标的操作与机器人 TCP 的运动方向一致。

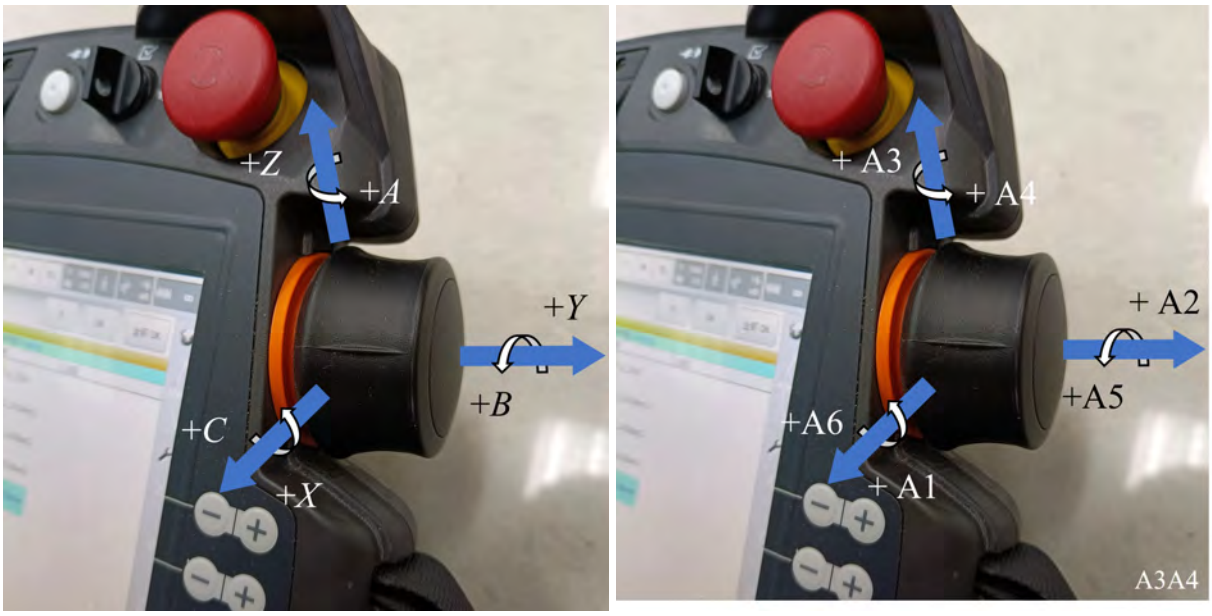
带 3D 鼠标的 smartPAD 或 KCP 示教器的操作方位设定操作，可通过单击手动移动选项的设定标签“Kcp 项号”选择，设定页面的显示如图3.46所示。移动圆周上的示教器图标，便可改变示教器与机器人的相对位置，使 3D 鼠标操作与机器人 TCP 的实际运动方向统一。

在 0° 方位，控制机器人 TCP 及工具定向正向运动时，3D 鼠标的操作方向如图3.47(a)所示。鼠标反向操作时，可使对应轴反向运动。例如，将 3D 鼠标拉出（向右），机器人 TCP 可进行基座坐标系 $+Y$ 方向的直线移动; 将 3D 鼠标压入（向左），机器人 TCP 则可进行基座坐标系$-Y$ 方向的直线移动; 逆时针偏转 3D 鼠标，即可使工具进行绕基座坐标系 $+Y$ 轴回转的 $+B$ 轴工具定向运动。

3D 鼠标也可以用于关节坐标系手动操作（关节手动），此时，机器人的运动与示教器方位无关，关节轴正向回转所对应的 3D 鼠标操作始终如图3.47(b)所示。反向操作鼠标时，同样可进行关节轴的反向回转运动。

(3) 3D 鼠标功能设定

图 **3.46:** 示教器设定页面显示



TCP 移动　　关节轴移动

图 **3.47:** 3D 鼠标操作方向

smartPAD 示教器的 3D 鼠标的操作功能，可通过图3.48所示的手动移动选项“鼠标”设定页的“鼠标设置”项选择，单击对应的图标，可选择如下操作功能。

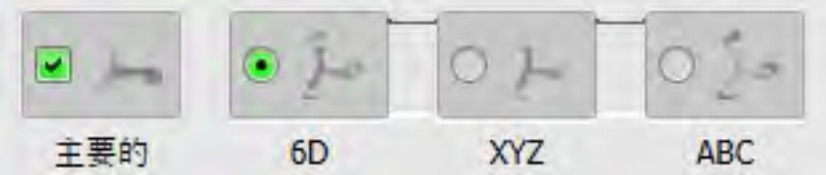


图 3.48: 3D 鼠标功能设定

① 主要的：主导轴操作。单击图标选定后，3D 鼠标只能用于“主导轴”移动控制，其他方向的控制均无效。“主导轴”可以在图标选定后，通过操作鼠标确定。

例如，对于图3.47所示的 3D 鼠标，如果向外拉动 3D 鼠标，机器人 TCP 沿基座坐标系的 *Y* 轴运动，此轴便成为“主导轴”，3D 鼠标只能用于 *Y* 轴移动控制等。如果在确定主导轴时，鼠标同时产生了多个方向的移动，系统将以 3D 鼠标移动量最大的轴作为主导轴。

② 6D：3D 鼠标 6 个方向的运动控制均生效。

③ *XYZ*：3D 鼠标的转动、偏摆操作无效，鼠标只能通过左右、前后、上下推拉，控制机器人 TCP 的笛卡儿坐标系 *X* 轴、*Y* 轴、*Z* 轴正反向直线移动，或者控制关节轴 A1、A2、A3 的正反向回转。

④ *ABC*：3D 鼠标的左右、前后、上下推拉操作无效，鼠标只能通过转动、偏摆，控制作业工具的笛卡儿坐标系 *A* 轴、*B* 轴、*C* 轴正反向定向回转，或者控制关节轴 A4、A5、A6 的正反向回转。

7. 机器人手动操作

(1) 基本说明

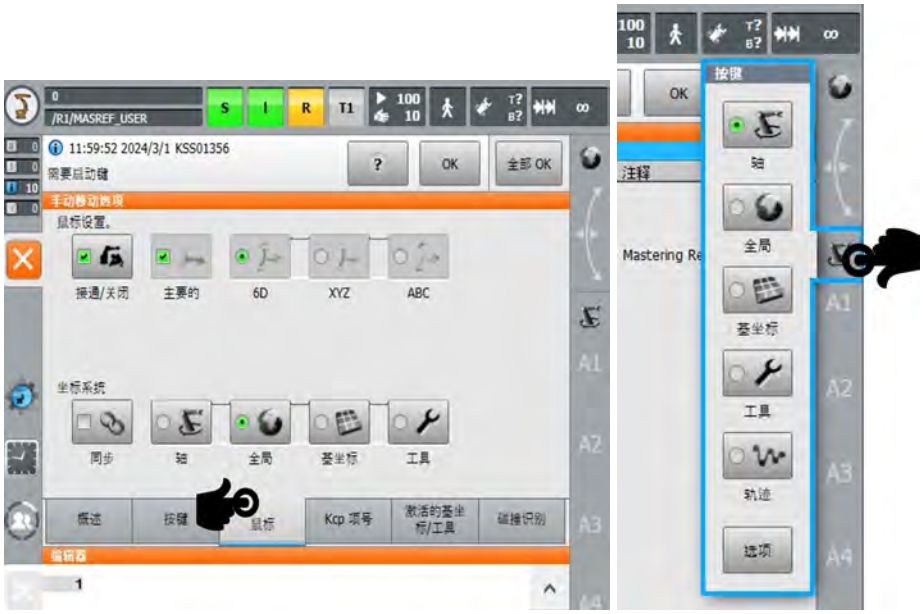
工业机器人的手动操作是利用示教器按键、3D 鼠标，对机器人及附加部件进行的手动操作，KUKA 机器人手动操作的移动方式分以下几种。

根据操作对象，工业机器人的手动移动方式通常有机器人本体及附加轴的关节坐标系手动移动（以下简称关节手动），以及笛卡儿坐标系的机器人手动移动与工具定向（以下简称机器人手动）2 类。关节手动可用于机器人本体关节（机器人轴）、变位器等附件（附加轴）的位置调整，一般以单轴移动为主，机器人手动通常用于作业工具的位置调整、示教编程，其运动为多个关节轴合成的三维空间运动。

根据运动方式，工业机器人的手动移动方式通常有手动连续移动（JOG，亦称点动）和手动增量移动（INC，亦称手动单步）两种。选择 JOG 操作时，可通过 smartPAD 示教器按键、3D 鼠标的操作，控制机器人进行指定方向的移动，松开按键、3D 鼠标，运动即停止。选择手动增量移动时，运动轴及方

向需要通过示教器按键选择（3D 鼠标无效），每操作一次操作方向键，机器人可以在指定方向移动指定距离，运动到位后，无论是否松开方向键，机器人均将停止运动，需要继续运动时，必须在松开方向键后，进行再次操作。

KUKA 机器人手动操作的移动方式、移动速度、坐标系及操作控制部件（方向键、3D 鼠标）等条件需要在机器人移动前予以设定。KUKA 机器人的手动操作条件可通过“手动移动选项”、单击状态栏图标、操作 smartPAD 右侧图标等多种方式打开、设定，其效果相同。例如，对于 3D 鼠标操作的坐标系，既可通过图3.49(a)所示的手动移动选项设定，也可以单击打开 smartPAD 右侧 3D 鼠标坐标系图标选定等，对此不再一一说明。以下以“手动移动选项”设定为例进行说明。



手动移动选项选择　　坐标系图标选择

图 3.49: 3D 鼠标坐标系选择

(2) 关节手动操作

关节手动操作可用于图3.50所示的机器人本体或外部轴的关节坐标系手动移动。由于 KUKA 机器人关节轴的运动方向与 FANUC、安川、ABB 等公司的机器人有所不同，手动操作时需要注意运动方向选择。

KUKA 机器人关节手动的基本操作步骤分别如下。

① 检查机器人、变位器（外部轴）等运动部件均处于安全、可自由运动的位置。

② 接通总电源、启动机器人控制系统。

③ 复位示教器及外部操作部件（如存在）上的急停按钮，并确认控制系统工作正常、无报警。

④ 单击状态显示区的驱动器状态显示图标，接通伺服驱动器主电源（参见3.3.5节）。

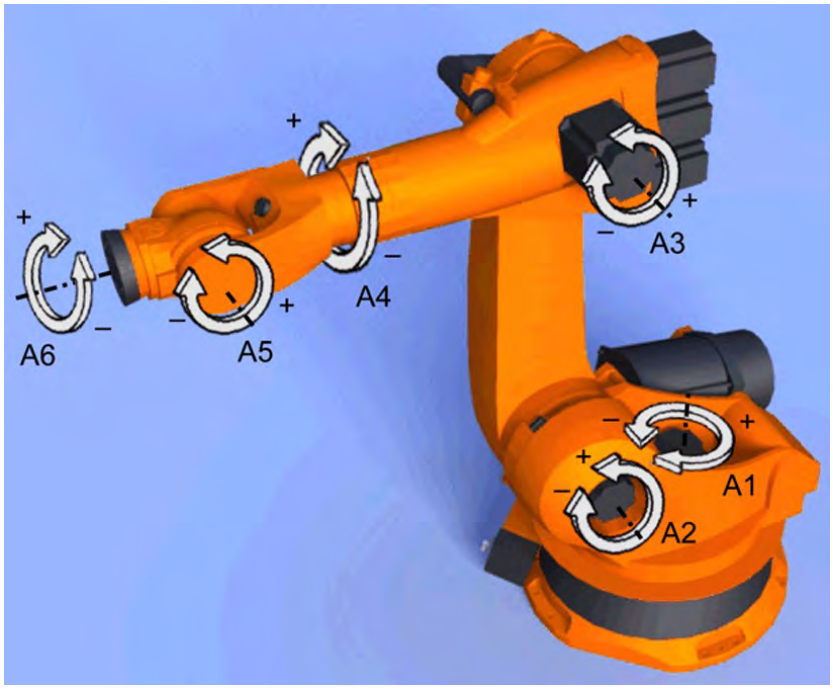


图 3.50: KUKA 机器人关节手动

⑤ 在 smartPAD 示教器操作模式切换旋钮上插入钥匙，并旋转旋钮至“解锁”位置，然后，在示教器显示的操作模式选择框中选择 T1 模式 (手动与示教操作模式)。操作模式选定后，再将旋钮旋回“闭锁”位置，取下钥匙，并确认状态显示区的操作模式图标为 T1（参见3.3.5节）。

⑥ 单击状态显示区的速度倍率显示图标（或其他图标），在打开的显示框中单击“选项”，打开手动移动选项设定框后，选择项目标签，完成如下设定。

- “概述”项：利用“手动调节量”的“+”“一”键或滑移调节图标，以倍率的形式，设定手动移动速度。T1 模式的机器人 TCP 最大移动速度为 250mm/s，对应的倍率为 100%。
- “按键”项：根据实际操作需要，完成如下设定。

  单击“激活按键”图标，生效/取消 smartPAD 示教器手动方向键。方向键一旦被取消，机器人手动操作只能利用 3D 鼠标进行，因此，机器人将不能进行手动增量移动。

  单击“增量式手动移动”输入选择图标，选择手动移动方式及增量距离。进行手动连续移动操作时选择“持续的”，进行手动增量移动时，选定增量移动距离（0.1mm/0.005° 或 1mm/1°、10mm/3°、100mm/10°），并确认方向键有效。

  单击“运动系统组”输入选择图标，选择控制轴组（机器人轴或附加轴）。

  单击“坐标系统”选项，选定方向键操作的机器人坐标系，使用

关节坐标系手动操作时，必须选定“轴”（参见 6.1.4 节）。

► “鼠标”项：根据实际需要，完成如下设定。

单击“激活鼠标”图标，生效/取消 Smart PAD 示教器 3D 鼠标。

单击“鼠标设置”选项，设定 3D 鼠标操作功能（主导轴操作、3 轴操作或 6 轴操作）。

单击“坐标系统”选项，选定 3D 操作的机器人坐标系，关节手动时，同样必须选定“轴”，或者单击生效“同步”选项，自动选择与方向键操作相同的坐标系（关节坐标系）。

► “Kcp 项号”项：使用关节坐标系手动操作时，3D 鼠标与关节轴对应关系保持不变，无须设定示教器操作方位。

► “激活的基坐标/工具”项：关节手动操作与工件坐标系、工具坐标系等笛卡儿坐标系无关，无须设定工件坐标系、工具坐标系。

⑦ 将 smartPAD 示教器的手握确认开关握至中间位置并保持，启动伺服。

⑧ 利用 smartPAD 示教器的手动方向键、3D 鼠标选择移动轴及运动方向，机器人关节即可按要求移动。选择手动增量移动时，3D 鼠标无效，关节只能利用 smartPAD 示教器的方向键选择运动轴与方向。

(3) 机器人手动操作

利用机器人手动操作，可使机器人 TCP 在所选的笛卡儿坐标系中进行 *X* 轴、*Y* 轴、*Z* 轴直线移动及 *A* 轴（绕 *Z* 轴回转）、B 轴（绕 *Y* 轴回转）、C 轴（绕 *X* 轴回转）工具定向回转。

****

8.3.11 图需要建模？

****

例如，当手动操作坐标系为基座坐标系时（系统默认设定大地坐标系、基座坐标系、工件坐标系重合），机器人可进行图 8.3-11(a) 所示的 *X* 轴、*Y* 轴、*Z* 轴直线移动及图 8.3-11(b) 所示的工具定向回转。

KUKA 机器人手动的基本操作步骤分别如下。

步骤 (1) (5) 同关节手动操作。

步骤 (6)——单击打开状态显示区的速度倍率显示（或其他图标），选择“选项”，打开手动移动选项设定框，并利用项目标签完成如下设定。

① “概述”项：选定手动移动速度倍率（同关节手动操作）。

② “按键”项：与关节手动操作一样，生效/取消 smartPAD 示教器手动方向键、选择手动移动方式及增量距离、设定控制轴组（机器人轴或附加轴）。然后，在“坐标系统”选项中选择方向键操作的笛卡儿坐标系 (参见 6.1.4 节)，机器人手动的坐标系必须为“全局（大地坐标系）”或“基坐标（工件坐标系）”“工具（工具坐标系）”之一，不能选择“轴（关节坐标系）”。

③ “鼠标”项：与关节手动操作一样，生效/取消 smartPAD 示教器 3D 鼠标、

在“鼠标设置”选项中选定鼠标操作功能。然后，在“坐标系统”选项中选择 3D 鼠标操作的笛卡儿坐标系，机器人手动的坐标系同样必须为“全局（大地坐标系）”或“基坐标（工件坐标系）”“工具（工具坐标系）”之一，或者单击生效“同步”选项，自动选择方向键操作相同的笛卡儿坐标系。

④ “Kcp 项号”项：根据实际需要，调整示教器操作方位，将 3D 鼠标操作方向和机器人实际运动方向调成一致。

⑤ “激活的基坐标/工具”项：选定机器人的工件坐标系、工具坐标系编号，对于无作业工具、进行机器人基座坐标系移动的手动操作，工件坐标系、工具坐标系及编号可选择 $NULLFRAME、[0]；然后，在“Ipo 模式选择”选项中选定工具安装形式。

(4) 将示教器手握确认开关握至中间位置并保持，启动伺服。

(5) 利用 SmartPAD 示教器手动方向键、3D 鼠标，选择移动轴及方向，机器人 TCP、作业工具即可按要求移动、定向。选择手动增量移动时，3D 鼠标，无效，机器人只能利用 smartPAD 示教器手动方向键选择运动轴与方向。

第 4 章 KUKA 机器人的零点标定

零点标定是指确定机器人各关节的初始位置或零位点，这是机器人定位和运动控制的基础。通过零点标定，可以确保机器人的运动轨迹与预定的轨迹相吻合，从而提高操作的精度和效率。此外，零点标定还有助于诊断和校正机器人的偏差，保障机器人系统的长期稳定性和可靠性。

4.1 引言

4.1.1 零点标定的目的

零点标定的主要目是为了确保机器人在执行任务时能够准确地定位和移动到指定位置，其是机器人系统中一个关键的校准过程。具体来讲，为了达到以下几个目的：

1. 精确定位：通过零点标定，可以确保机器人的关节和执行器在空间中的位置与控制系统中的坐标一致，从而实现精确的定位。
2. 提高重复性：标定过程可以减少机器人在重复执行相同任务时的位置偏差，提高操作的重复性和一致性。
3. 校正偏差：零点标定有助于识别和校正机器人系统中可能存在的机械偏差或磨损，保持机器人的长期稳定性和可靠性。
4. 优化性能：准确的零点标定可以优化机器人的运动轨迹和速度，提高操作效率和生产效率。
5. 降低风险：通过确保机器人的精确控制，零点标定有助于降低操作过程中的安全风险和潜在的损坏风险。

机器人在标定之后方可进行正确的笛卡尔运动并移至编程位置。机器人的机械位置和电子位置会在调整过程中协调一致。为此需要将机器人置于一个已经定义的机械位置，即零点标定位置。然后，每个轴的传感器值均被储存下来。所有机器人的零点标定位置都相似，但不完全相同。精确位置在同一机器人型号的不同机器人之间也会有所不同。

总的来说，零点标定是确保机器人或自动化系统能够准确测量和响应的重要步骤，有助于提高系统的性能和稳定性。

4.1.2 零点标定方法

机器臂工具的零点校准是确保工具相对于机器臂坐标系的准确位置和姿态的过程。以下是一些常见的机器臂工具零点校准方法：

1. 手动零点校准：手动零点校准是一种简单的方法，通过手动调整工具或末端执行器的位置和姿态，使其与机器臂坐标系对齐。操作员可以通过观察和调整来确保工具的零点位置准确，例如使用千分表进行零点标定。
2. 基于视觉的零点校准：利用视觉传感器或摄像头进行工具零点校准是一种常见的方法。通过在工具上安装视觉标记或特征点，并使用机器视觉算法检测和识别这些标记，可以实现工具相对于机器臂坐标系的准确位置和姿态校准。
3. 基于传感器的零点校准：如果工具上配备有传感器，如编码器或其他位置传感器，可以利用这些传感器的反馈信息进行工具零点校准。通过分析传感器输出数据，可以确定工具相对于机器臂坐标系的零点位置。
4. 自动化零点校准程序：一些机器人系统具有自动化的工具零点校准程序，可以通过预先定义的校准任务或程序来实现工具零点校准。这种方法通常需要进行一次手动校准以建立参考点，然后通过自动化程序来调整工具位置和姿态使其与机器臂坐标系对齐。

选择合适的零点校准方法取决于机器人系统的具体要求和工具特性。确保定期进行工具零点校准可以帮助保持机器人系统的精度和可靠性。

在本书中，主要使用 KUKA 自动化零点校准程序以及使用千分表来进行校准，确保每个读者按照操作步骤即可完成 KUKA 机器人的零点校准。

4.1.3 什么时候进行零点标定

零点标定是一个重要的校准步骤，通常在以下情况下需要进行零点标定：

1. 系统组装后：当机器人系统组装完成后，特别是在安装和调试阶段，通常需要进行零点标定，以确保系统的准确性和稳定性。
2. 定期维护：在进行维护操作之后，如更换了点击或者 RDC，机器人的零点值丢失。定期进行零点标定是保持机器人系统性能的关键步骤。随着时间的推移，机器人系统中的传感器和执行器可能会出现漂移或误差累积，因此定期进行零点标定可以确保系统保持高精度。
3. 更换关键部件：如果更换了机器人系统中的关键部件，如传动装置、执行器或控制器，通常需要进行零点标定，以适应新组件并确保系统的正常运行。
4. 碰撞：发生一次碰撞之后，或以高于 250 mm/s 的速度撞到一个终止挡上之后。

   校准建议：根据制造商的建议或机器人系统的使用手册，可能会提供关于零点标定的最佳实践和频率建议。遵循制造商的建议进行零点标定可以确保系统在最佳状态下运行。

   总的来说，零点标定应该作为机器人系统维护和校准的重要组成部分，并根据

实际情况和制造商建议进行定期进行，以确保系统的准确性、稳定性和性能。

4.1.4 哪些情况下必须进行零点标定

在以下情况必须对机器人进行零点标定：

1. 在调试时；
2. 在进行维护操作之后，如更换了电机或者 RDC，机器人的调整值丢失；
3. 若机器人在无机器人控制系统操控的情况下运动（例如借助自由旋转装置）；
4. 更换传动装置后；
5. 以高于 250mm/s 的速度上行移至一个终端止挡之后；
6. 在碰撞后。

如图4.1所示，当示教器上显示该信息时，必须执行零点校正方可正常使用。

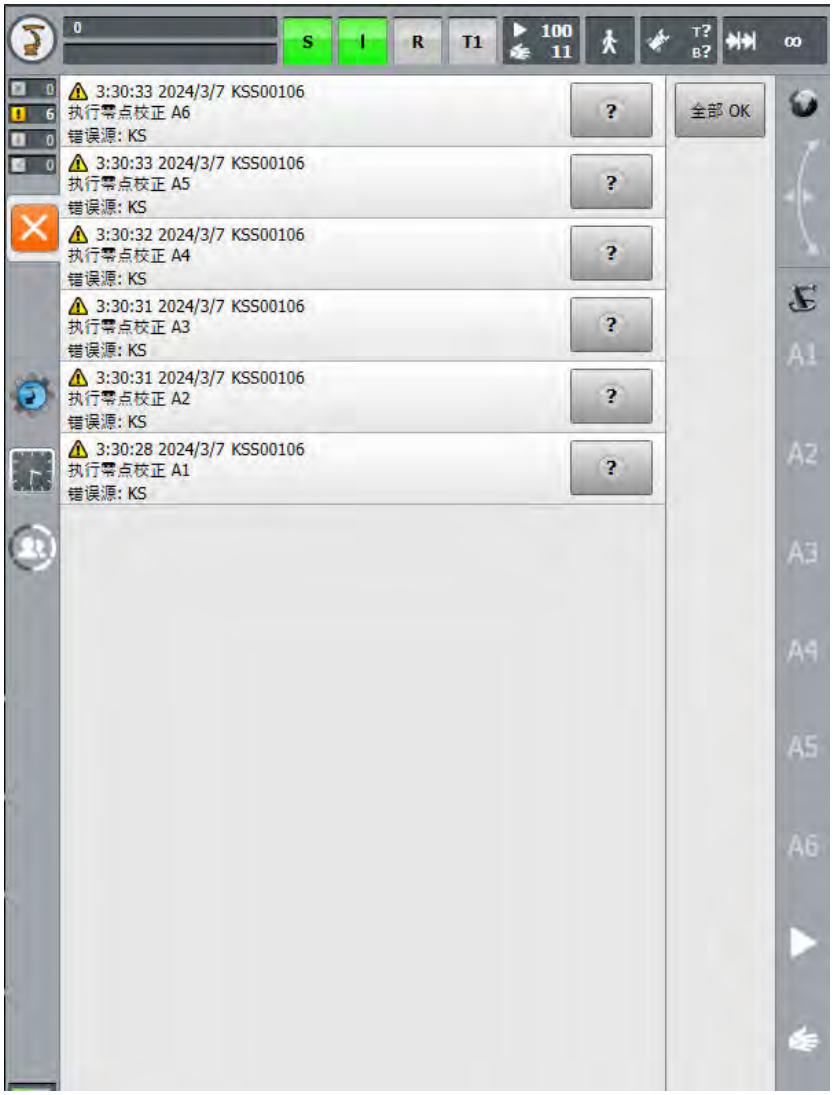


图 4.1: 必须执行零点校正信息

在进行新的零点标定之前必须删除旧的零点标定数据！通过手动删除轴的零点来删除零点标定数据，该项内容操作见后续。

4.2 使用 MEMD 进行零点标定

4.2.1 利用零点标定标记将轴移入预零点标定位置

在每次零点标定之前我们需要将轴移至预零点标定位置（如图4.3），逐一每一关节轴，使零点标定标记尽量重叠，不一定要特别准确，但要存在一些重叠。

注意：对于第六轴而言，我们需要将 A6 移至零点标定位置。（所指的是在总零点标定过程之前，而不是直接在 A6 自身的零点标定前）。我们可以观察到 A6 的金属上刻有很精细的线条，如图4.2所示。为了将 A6 移至零点标定位置上，这些线条要精确地相互对齐。在驶向零点标定位置时，须从前方正对着朝固定的线条看，这一点尤其重要。如果从侧面朝固定的线条看，则可能无法精确地将运动的线条对齐。后果是没有正确地标定零点。

图 4.2: 六轴零点位置

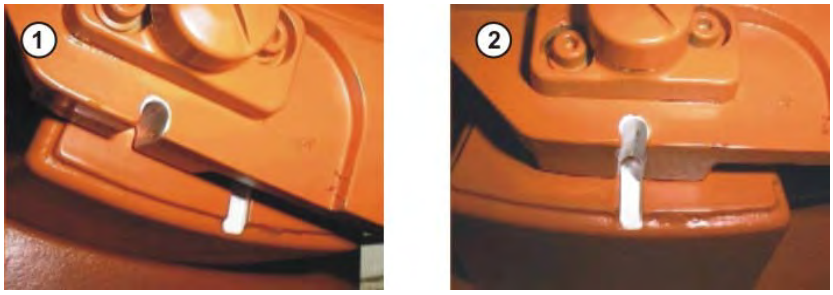


图 4.3: 机器人预零点位置

下图4.4显示零点标定标记位于机器人上的位置。由于机器人的型号不同，位置会与插图稍有差异。

操作步骤

1. 运行方式为 T1，运行模式“运行键”已激活。

   注意：在 A4 和 A6 进入预零点标定位置前，必须确保供能系统（如果有的话）应处在正确位置，不得翻转 360°。

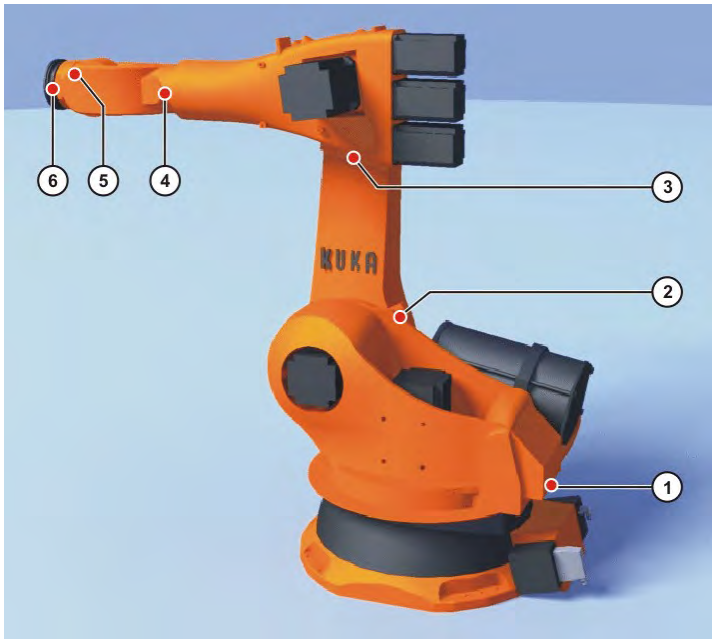


图 4.4: 零点标记位置

2. 如图4.5所示，在库卡 smartpadHMI 上选择轴作为运动键的坐标系。

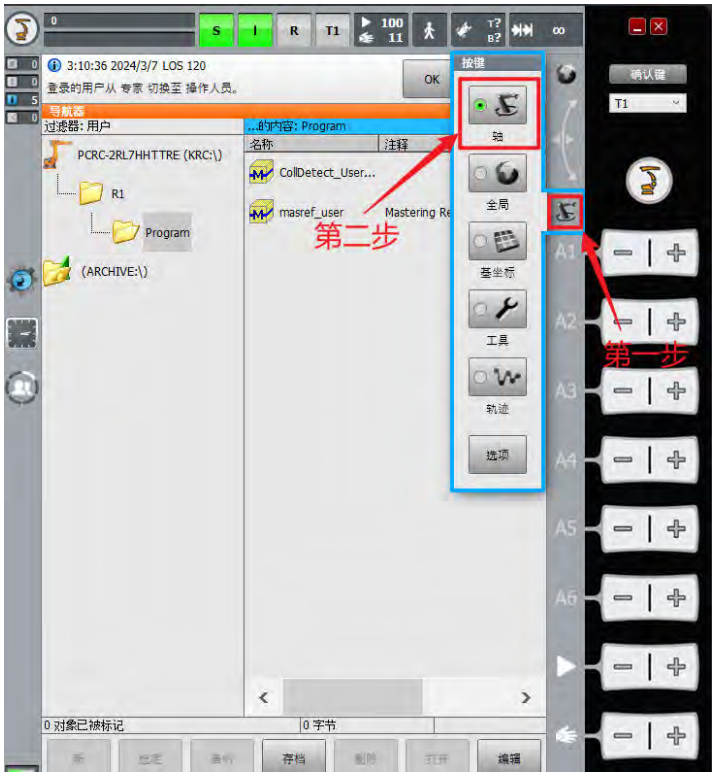


图 4.5: 选择轴坐标系

3. 如图4.6所示，按住库卡背部白色确认开关。在移动键旁边将显示轴 A1 至轴 A6。

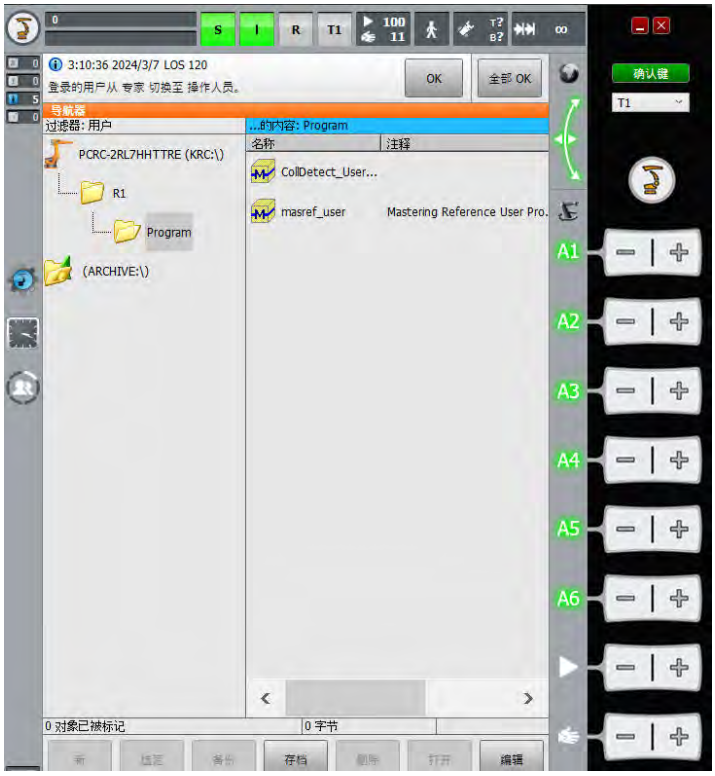


图 4.6: 按住确认开关

4. 按下正或负运动键，以使轴朝正方向或反方向运动。

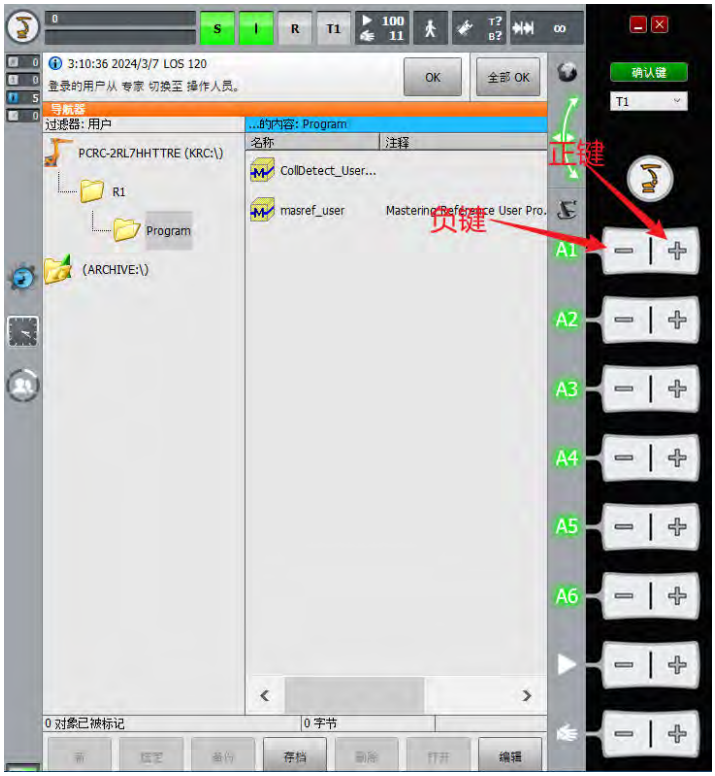


图 4.7: 正负键

5. 从 A1 开始逐一移动各个轴，使零点标定标记相互重叠。（在借助标记线对轴进

行零点标定的机器人上的 A6 除外。）如图4.4所示。

4.2.2 包含 SEMD 和 MEMD 的零点标定组件

1. **零点标定工具箱**

我们首先介绍一下我们所需要的零点标定工具，零点标定工具箱是由 KUKA 公司所提供的，我们在拥有一台 KUKA 机器人时，可以选择向厂商进行购买或者租借，一般称之为零点标定工具箱，工具箱中包含 SEMD 和 MEMD 的零点标定组件。如图4.8所示，主要包括①-零点标定盒、②-用户 MEMD 的螺丝刀、③-MEMD、④-SEMD、⑤-电缆。

图 4.8: 包含 SEMD 和 MEMD 的零点标定组件

对于 KUKA 手动零点标定，一般有 SEMD 和 MEMD 两种工具以供我们选择，哪个零点标定工具用于我们的机器人，则取决于机器人配备了哪种类型的测量筒，各种类型在防护盖的尺寸方面有所区别。不同的零点标定应用不同的测量筒，不同的测量筒其防护盖的尺寸有所不同。用于 SEMD 的测量筒 (Standard Electronic Mastering Device) 防护盖配 M20 的细螺纹；用于 MEMD 的测量筒 (Mikro Electronic Mastering Device) 防护盖配 M8 的细螺纹，我们可以通过观察或者游标卡尺进行测量来确认。一般地，SEMD 传感器用于大型校准套筒，MEMD 传感器用于小型校准套筒。由于我们的机器人为 KR6 R700 类型机器人，需要使用 MEMD 进行校准，因此主要对其进行讲解，使用 SEMD 校准方法及操作步骤基本相同。

2. **零点标定工具箱中线缆的连接**

如图4.8所示的零点标定组件，其中包含两条电缆，细电缆是测量电缆，将 SEMD 和 MEMD 与零点标定盒相连接，这里我们将 MEMD 与零点标定盒的一端相连。粗电缆是 EtherCAT 电缆，也需要将其的一端与零点标定盒进行连接。

图 4.9: 将零点标定盒连接

其次，还需将 EtherCAT 的另一端与机械手尾部的 X32 接口进行连接，在机械手尾部众多接口中，我们可以找到旁边印有 X32 的接口（如图4.10所示）取下接口 X32 上的盖子，并将 EtherCAT 线缆另一端与机器人上的 X32 连接起来（如图4.11所示），自锁型按钮，用力插入即可。

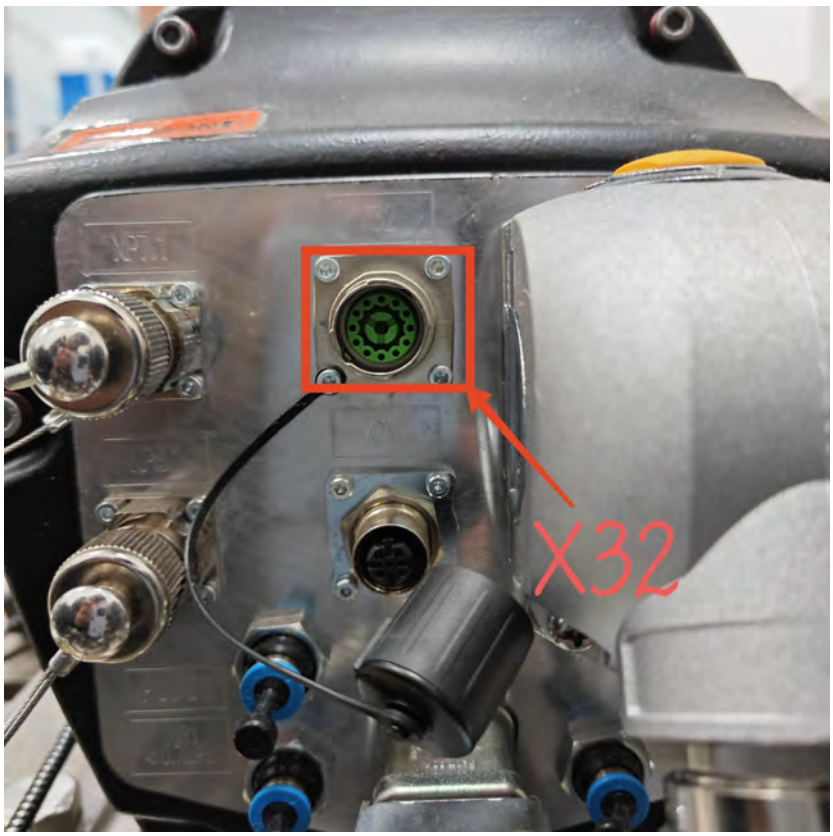


图 4.10: X32 位置

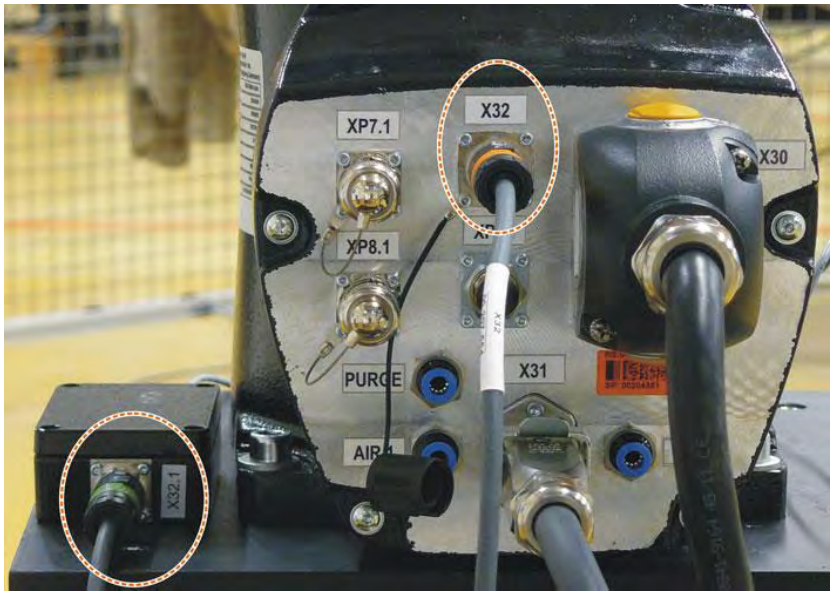


图 **4.11:** 将 EtherCAT 电缆连接到 X32 上

注意: 让测量电缆插在零点标定盒上，并且要尽可能少地拔下。传感器插头 M8 的可插拔性是有限的。经常插拔可能会损坏插头。

下列情况针对测量导线没有安装牢固的测头：始终将设备不带测量导线拧到测量筒上。然后方可将导线接到设备上。否则导线会被损坏。同样在拆除设备时也始终先拆下设备的测量导线。然后才将设备从测量筒上拆下。

在零点标定之后，将 EtherCAT 电缆从接口 X32 上取下。否则会出现干扰信号或导致损坏。

4.2.3 零点标定 LED 指示灯

为了方便，我们首先介绍一下零点 LED 指示灯不同颜色表达的意思，以便读者可在下个步骤操作中遇到这些情况时可以处理。

在大多数零点标定过程中，smartHMI 显示一个含有轴的列表。两个 LED 指示灯位于列表右侧，如图4.14所示为零点标定 LED 指示灯, 下面简要介绍一下零点标定 LED 指示灯。

将用户组调整为专家用户组，如图4.12和4.13所示, 在主菜单中选择“投入运行”→“调整”→“EMD”→“标准”→“执行零点校正”。

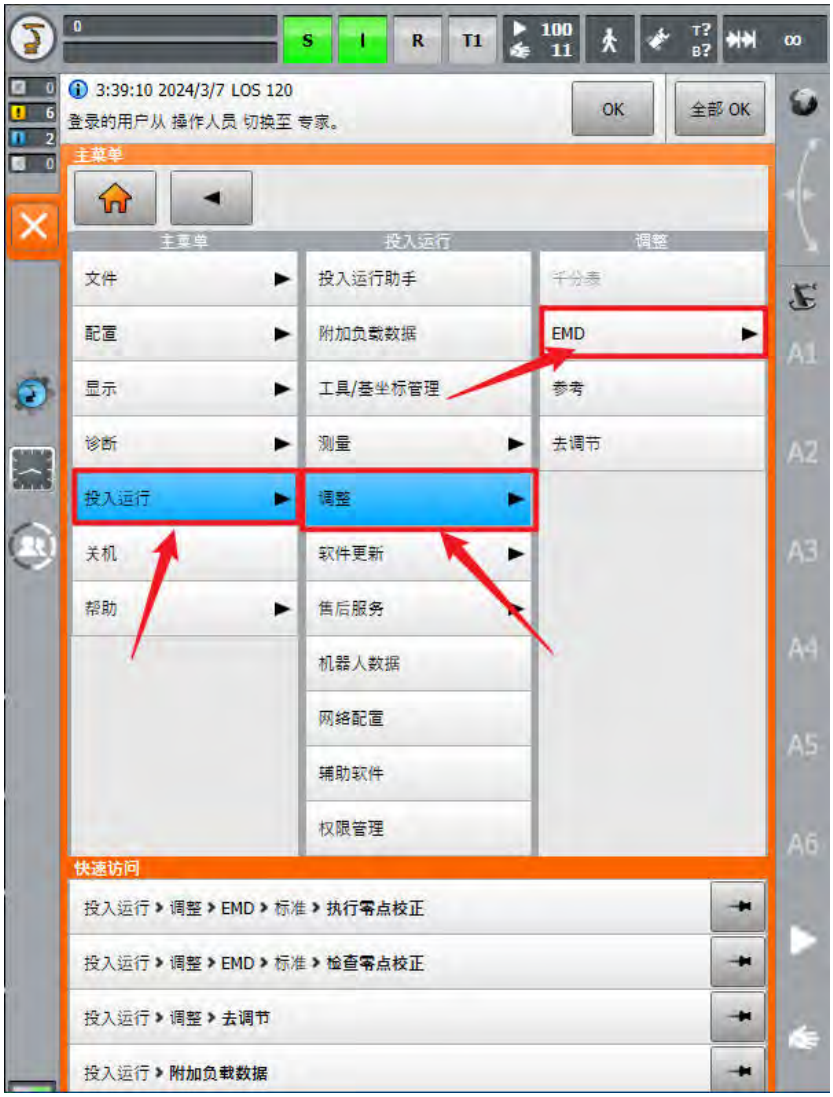


图 **4.12:** 找到 LED 灯 1

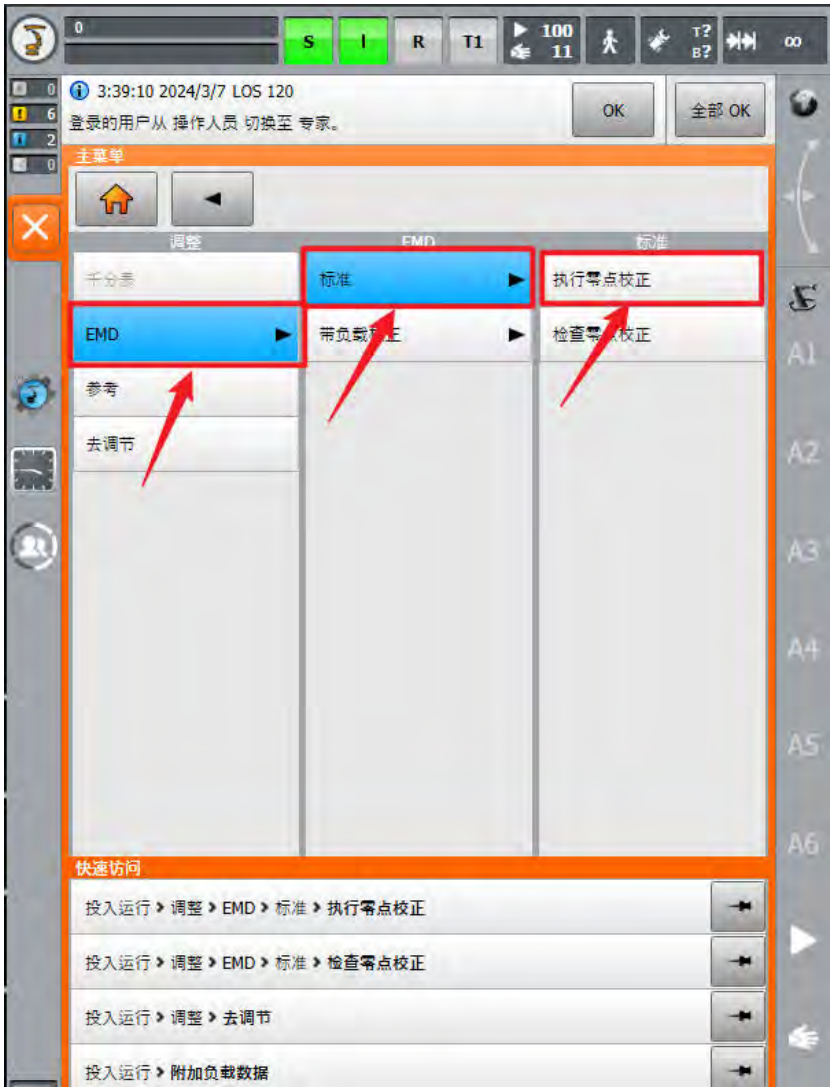


图 **4.13:** 找到 LED 灯 2

如图4.14所示，此时我们可以看到零点标定 LED 指示灯。

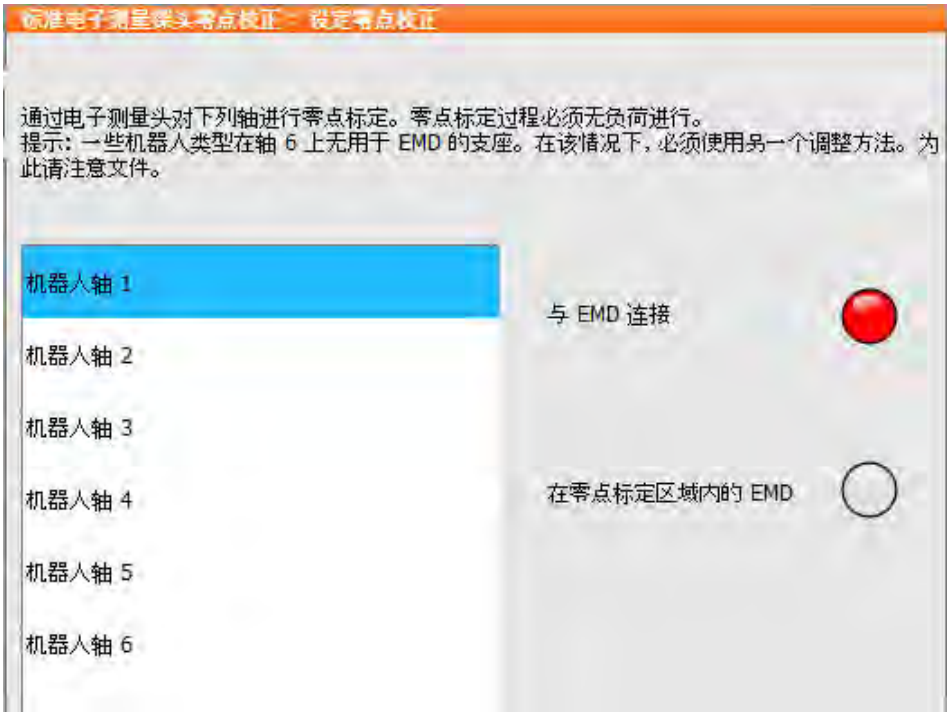


图 4.14: LED 指示灯

“与 EMD 连接”指示灯显示：

红色：测头没有与接口 X32 相连接。

绿色：测头与接口 X32 相连接。

如果该 LED 显示红色，则在零点标定区域内的 EMD 的 LED 显示为灰色。

“在零点标定区域内的 EMD”指示灯显示：

灰色：测头没有与接口 X32 相连接。

红色：测头位于无法进行零点标定的位置上。

绿色：测头直接位于用于零点标定的槽口旁或在凹口中。

4.2.4 首次进行零点校正（用 MEMD）

1. 使用 MEMD 进行首次零点校正前提条件
   (1) 机器人没有负载。也就是说，没有安装工具或工件和附加负载。
   (2) 所有轴都处于预调位置。
   (3) 没有选择程序。
   (4) 运行方式 T1。

   注意：始终将 NEMD 不带测量导线拧到测量筒上。然后方可将导线接到 MEMD 上。否则导线会被损坏。

   同样在拆除 MEMD 时也必须先拆下 MEMD 的测量导线。然后才将 MEMD 从测量筒上拆下。

在零点标定之后，将 EtherCAT 电缆从接口 X32 上取下。否则会出现干扰信号或导致损坏。

实际所用的 MEMD 不一定与图中所述的模型精确对应。二者用途相同。

2. 操作步骤

(1) 如图4.15所示，从窗口中选定的轴上取下测量筒的防护盖，一般我们轴的选择从小到大。（翻转过来的 MEMD 可用作螺丝刀）。

图 4.15: 取下测量筒的防护盖

(2) 将 MEMD 插入测量筒之中。

图 4.16: 将 MEMD 拧到测量筒上

(3) 在主菜单中选择“投入运行”→“调整”→“EMD”→“标准”→“执行零点校正”，一个窗口自动打开。所有待零点标定轴都显示出来，编号最小的轴“机器人轴 1”已被选定。

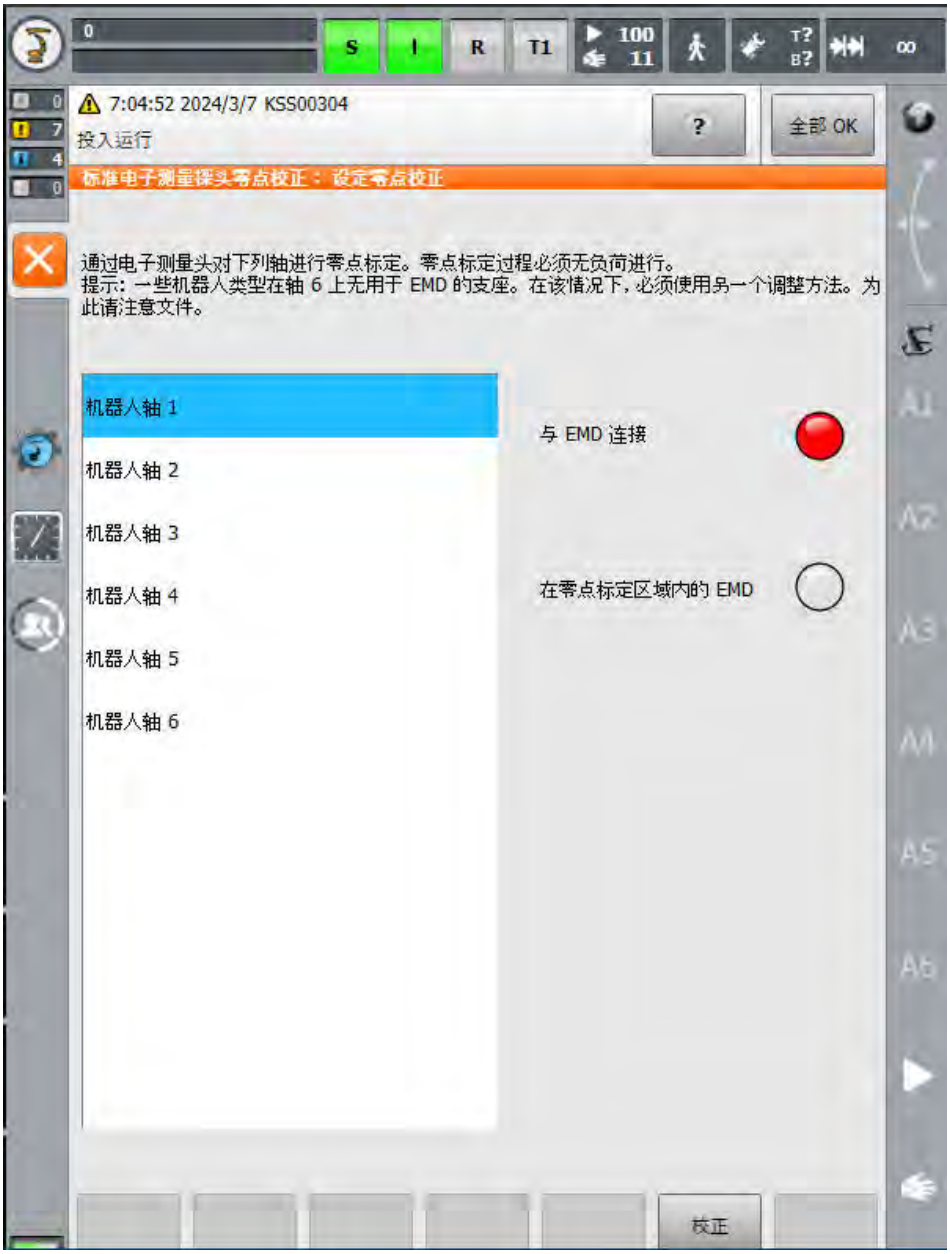


图 4.17: 零点校正页面

(4) 如图4.18所示, 点击校正，按下确认开关和启动键。

图 4.18: 点击校正

(5) 如果 SEMD 已经通过了测量切口，则零点标定位置将被计算。机器人自动停止运行。数值被保存，该轴在窗口中消失（如图4.19所示）。

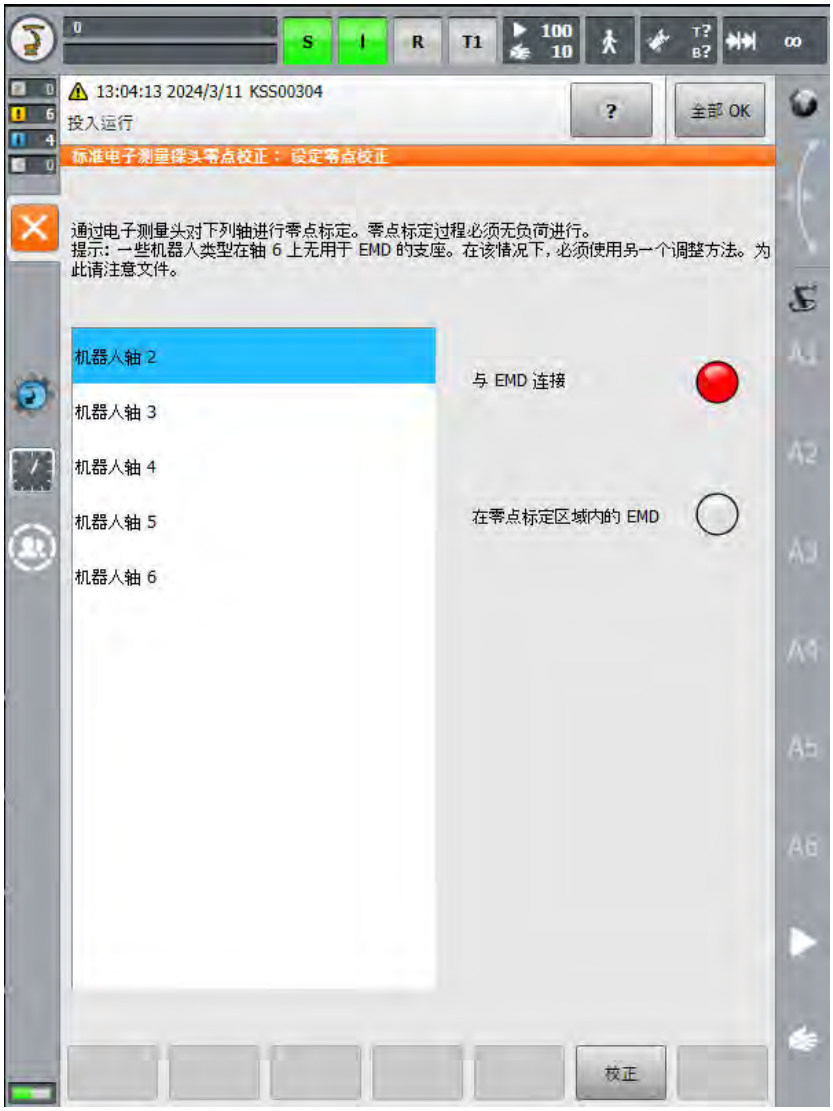


图 4.19: 机器人轴 1 已校正

(6) 将测量导线从 MEMD 上取下。

(7) 从测量筒上取下 MEMD，并将防护盖重新装好。

(8) 对所有待零点标定的轴重复步骤 (3) 至 (4)。

(9) 关闭窗口，将 EtherCAT 电缆从接口 X32 和零点标定盒上取下。

注意：让测量电缆插在零点标定盒上，并且要尽可能少地拔下。传感器插头 M8 的可插拔性是有限的。经常插拔可能会损坏插头。

4.3 手动删除轴的零点

4.3.1 为什么需要删除轴的零点

通常，需要在以下情况下删除或重新进行机器人的零点标定：

1. 更换或维修部件：如果机器人的任何部件被更换或维修，特别是涉及运动或定

位的部件（如关节、驱动器或传感器），可能需要重新进行零点标定。

2. 机器人迁移：如果机器人被移动到一个新的位置或安装在一个新的环境中，可能需要重新进行零点标定以确保其在新环境中的准确性。
3. 长时间未使用：如果机器人长时间未使用，可能会出现微小的机械变化或漂移，这可能需要重新进行零点标定以确保其准确性。
4. 检测到定位偏差：在机器人的正常运行过程中，如果检测到定位偏差或运动不准确的情况，可能需要重新进行零点标定。
5. 定期维护：作为定期维护的一部分，可能会定期重新进行零点标定，以确保机器人的长期准确性和可靠性。

在必要时，可将各个轴的零点标定值删除。删除轴的零点时轴不动。轴 A4、A5 和 A6 以机械方式相连，当轴 A4 数值被删除时，轴 A5 和 A6 的数值也被删除，当轴 A5 数值被删除时，A6 的数值也被删除。注意：对于已去调节的机器人，软件限位开关已关闭。机器人可能会驶向极限卡位的缓冲器，由此可能使其受损，以至必须更换。尽可能不运行已去调节的机器人，或尽量减少手动倍率。

4.3.2 删除轴零点步骤

1. **前提条件**
   (1) 没有选定任何程序。
   (2) 运行方式为 T1。
2. **操作步骤**
   (1) 如图4.20所示，在主菜单中选择投入运行 → 调整 → 取消调整。一个窗口打开。

图 **4.20:** 取消零点校正

(2) 标记需进行取消调节的轴。

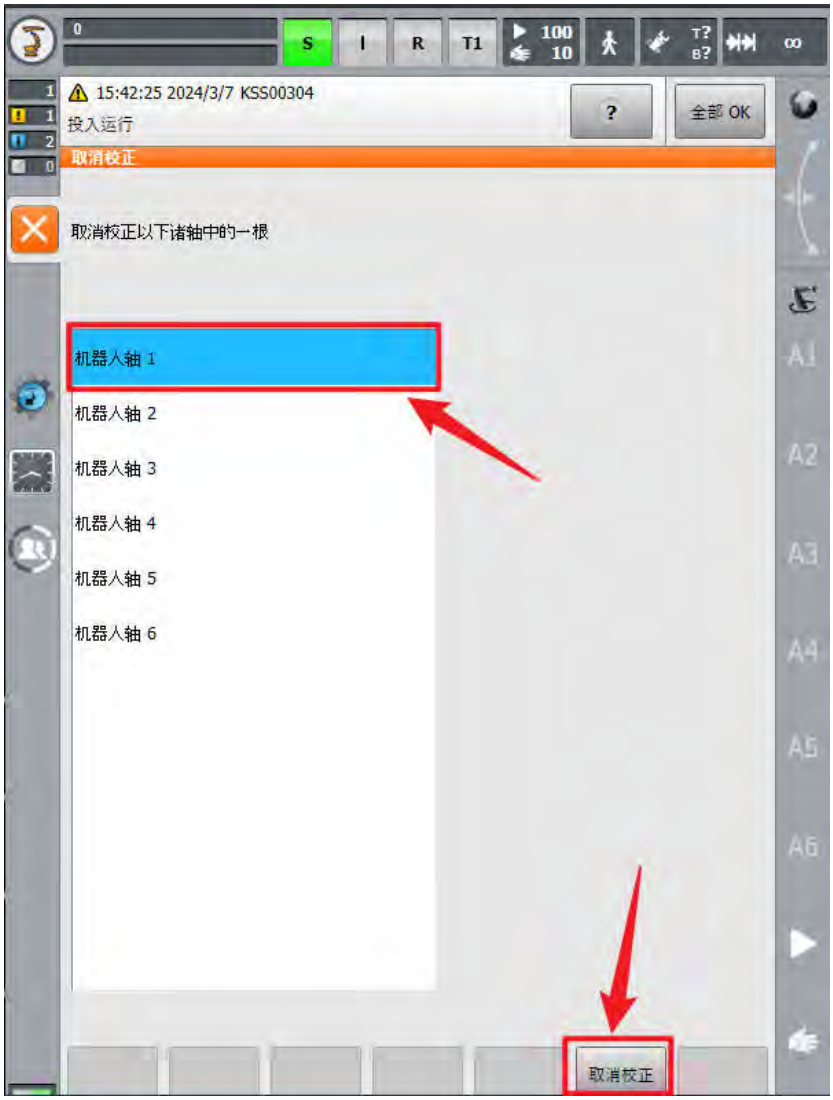


图 4.21: 取消校正

(3) 请按下取消调节。轴的调整数据被删除。

(4) 对于所有需要取消调整的轴重复第 (2) 和第 (3) 步骤。

(5) 关闭窗口。

4.4 更改软件限位开关

4.4.1 为什么需要改变软件限位开关

软件限位开关（也称为软件限位器或虚拟限位器）是用于防止机器人运动超出预设范围的一种安全措施。它们在机器人的控制软件中设置，用以限制机器人关节或轴的运动范围，以防止碰撞、损坏或其他安全问题。有几个原因可能需要更改机器人的软件限位开关：

1. 改变工作范围：如果机器人的任务或工作环境发生变化，可能需要调整其运动

范围。在这种情况下，需要更改软件限位开关以适应新的工作范围。

2. 防止碰撞：如果在机器人的工作区域中增加了新的设备或障碍物，可能需要调整软件限位开关以防止机器人与这些物体发生碰撞。
3. 优化性能：通过调整软件限位开关，可以对机器人的运动范围进行微调，以提高效率或减少不必要的运动。
4. 维护和修理：在维护或修理机器人时，可能需要临时更改软件限位开关以允许访问某些部件或执行特定的维护任务。
5. 安全原因：为了提高机器人系统的安全性，可能需要根据安全评估或事故调查的结果调整软件限位开关。

4.4.2 改变软件限位开关操作方法

有两种更改软件限位开关的方法：要么手动输入所需的数值，要么限位开关自动与一个或多个程序适配。在此过程中，机器人控制系统确定在程序中出现的最小和最大轴位置。得出的这些数值可以作为软件限位开关来使用。

1. **前提条件**

   (1) 权限为专家用户组。

   (2) 运行方式为 T1、T2 或 AUT。

2. **操作步骤**

   (1) **手动更改软件限位开关：**

      i. 如图4.22所示，在主菜单中选择“投入运行”→“售后服务”→“软件限位开关”。窗口软件限位开关自动打开。

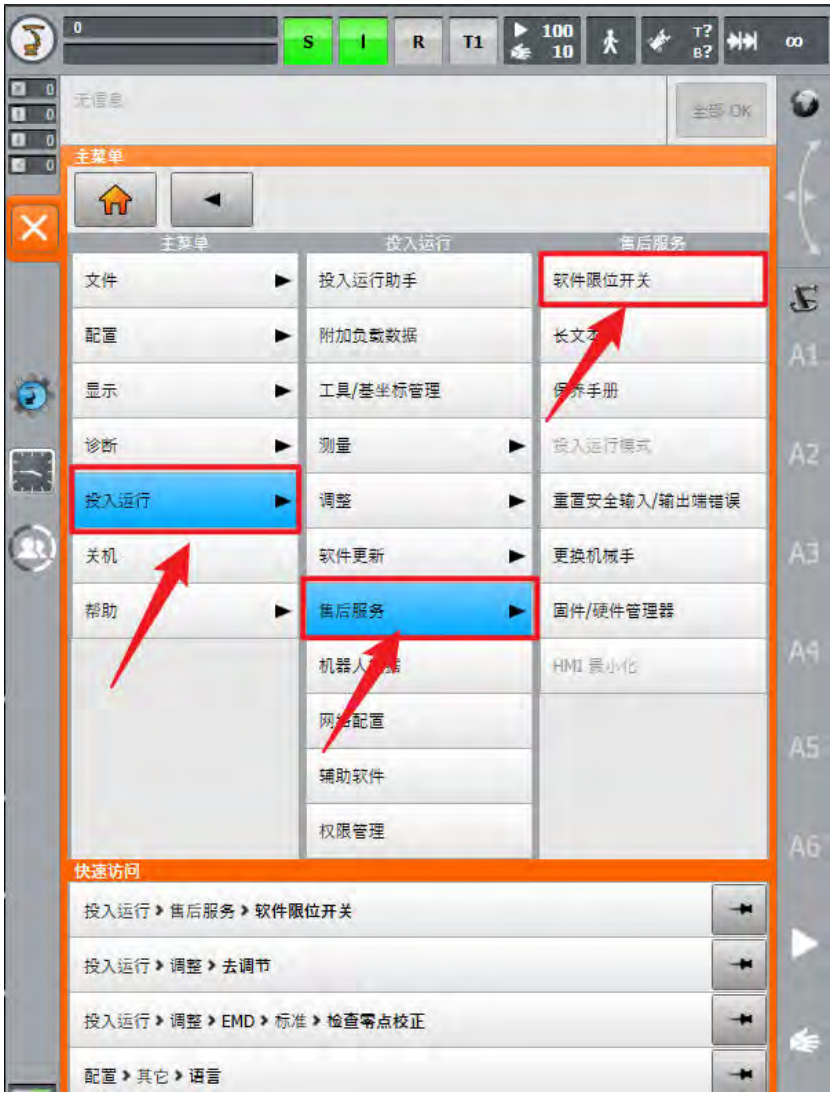


图 **4.22:** 软件限位开关

ii. 在负和正两列中按需要更改限位开关。

图 **4.23:** 更改软件限位开关

iii. 用保存键保存更改。

图 **4.24:** 限位开关保存成功

3. **将软件限位开关与程序相适配：**

   i. 在主菜单中选择“投入运行”→“售后服务”→“软件限位开关”。窗口软件限位开关自动打开, 如图4.25所示。

图 4.25: 在自动确定前

①——当前的负向限位开关; ②——轴的当前位置; ③——当前的正向限位开关

ii. 按下自动计算。显示以下提示信息：自动获取进行中, 如图4.26所示。

iii. 启动限位开关应与之相适配的程序。让程序完整运行一遍，然后取消选择。在窗口软件限位开关中即显示每个轴所达到的最大和最小位置。

图 4.26: 在自动确定期间

④——自启动计算以来，相应轴所具有的最小位置;
⑤——自启动计算以来，相应轴所具有的最大位置

iv. 为该软件限位开关应与之相适配的所有程序重复步骤③。在窗口软件限位开关中即显示每个轴所达到的最大和最小位置，而且以执行的所有程序为基准。
v. 如果所有需要的程序都执行过了，则在窗口软件限位开关中按下结束。
vi. 按下保存，以便将确定的数值用作软件限位开关。
vii. 需要时还可以手动更改自动确定的数值。建议：将确定的最小值减小 5°。将确定的最大值增加 5°。在程序运行期间，这一缓冲区可防止轴达到限位开关，以避免触发停止。
viii. 用保存键保存更改。

第 5 章 KUKA 机器人坐标系的测量

工业机器人的坐标系在机器人运动和控制中起着关键作用，如图5.1所示，在库卡机器人控制系统中定义了世界坐标系、机器人坐标系、基坐标系和工具坐标系，库卡机器人的运动也总是要围绕这些坐标系来讨论，理解这些坐标系以及学会测量这些坐标系是进行机器人运动的基础。

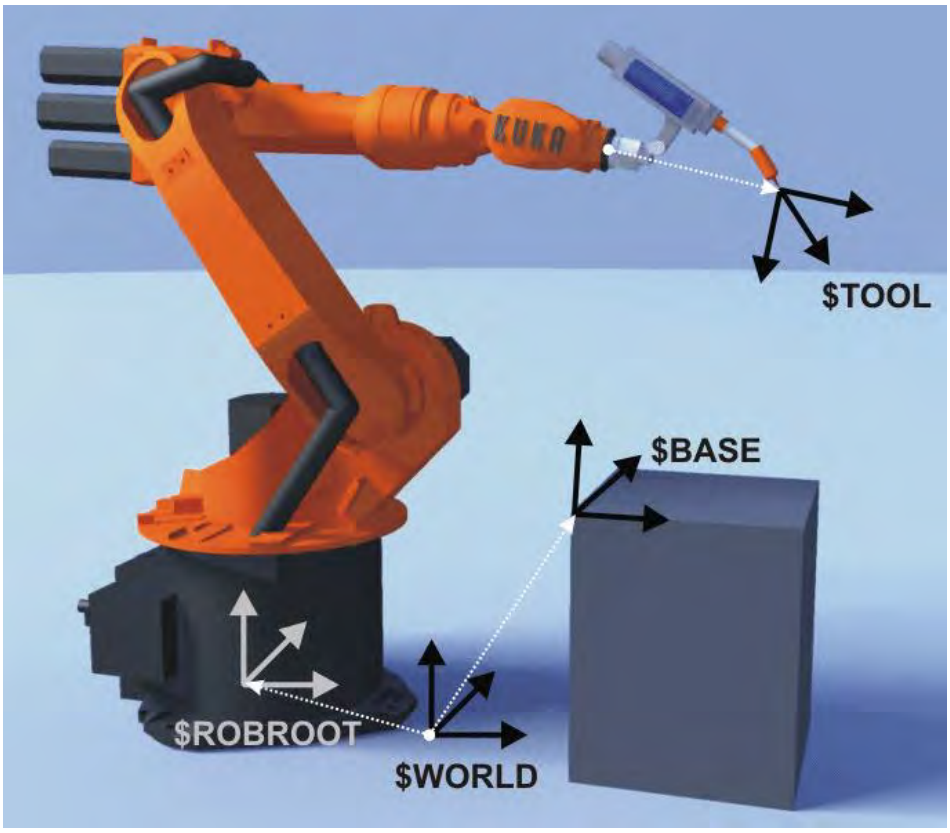


图 5.1: 机器人坐标系

工业机器人常用的几种坐标系如下所示：

1. (ROBROTT) 机器人坐标系：
   机器人坐标系是以机器人安装基座为基准的笛卡尔坐标系，它固定位于机器人足部，用来描述机器人本体运动的直角坐标系。它遵循右手准则，其中 X 轴为机器人标准位置前方，Y 轴为机器人侧向，Z 轴为机器人底座向上。
2. (WORLD) 世界坐标系：以工作环境中的固定参考点或参考物体作为基准来定义的坐标系，它通常是一个惯性坐标系，不随机器人底座的运动而改变，在多数情况下，世界坐标系和基坐标系是一致的（特殊场合除外）。在协同工作的机器人中特别方便，可让多个机器人定位到车间里的同一个位置点，例如一个机器人

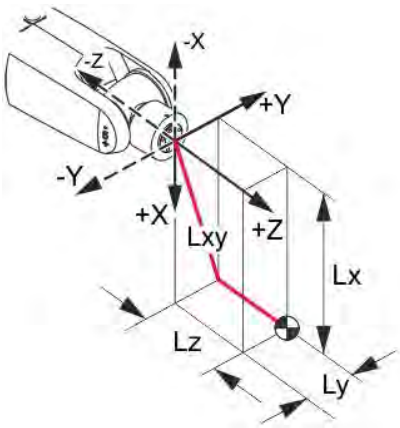
抓取另一个机器人焊接好的部件。

3. (TOOL) 工具坐标系：

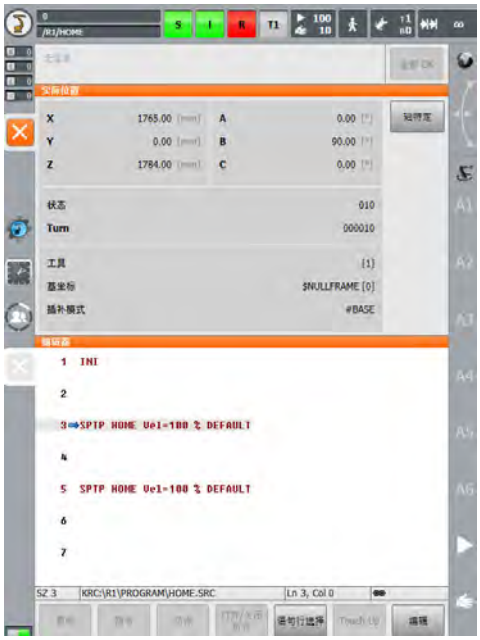
工具坐标系是一个笛卡尔坐标系，它固定在机器人工具（如焊枪、吸盘等）上，位于工具的工作点。它相对于机械手法兰中心不变，用于描述工具的位置。工具坐标系的原点是机械手运动中心点（TCP）。

在默认情况下,工具坐标系的原点在法兰中心上,因而被称作法兰坐标系 (FLANGE)。在示教器中可以依次点击“菜单”→“显示”→“实际位置”来查看当前法兰中心位置。

通过查阅机器人相关文档可知，在我们进行机械手的零点标定后，其 X 轴正向即为法兰向下方向，垂直法兰向外为 Z 轴正向，再次通过右手准则，即可判断出 Y 轴的正向。



法兰坐标系



实际位置

图 5.2: 法兰及法兰坐标

4. (BASE) 基坐标系：

基坐标系是一个笛卡尔坐标系，用来说明工件的位置，有时也称工件坐标系，工件坐标系被人为所定义，并通过机器人测量得到坐标系的精确位置，一般情况下，得到的是工件坐标系相对于机器人坐标系的位置关系。

这些坐标系在机器人控制和运动中起着重要作用，确保机器人能够准确执行任务。本章首先介绍机器人常用的几个坐标系，教大家认识机器人的坐标系以及如何定义与产生，各个坐标系之间有什么关系。其中，最为关键的是工具坐标系，工具坐标系的确定是我们测量一切物体坐标的基础，基于此可以知道空间中任何一点的坐标，进而可以测量空间中任何一个坐标系，除此之外，本书中用工件坐标系描述工件的位置，有了工件坐标系，它可以让机器人沿着工件的方向直接移动。

因此，坐标系的测量是后续章节实战部分的基础，读者应该认真掌握，动手操作以确保可以掌握其中要领。

5.1 测量工具坐标系

机器人工具坐标的测量是确定机器人末端执行器（通常是工具或末端执行器）相对于机器人基坐标系的位置和姿态的过程。这个过程至关重要，因为机器人的任务通常需要将末端执行器精确放置在目标位置，并以特定的姿态执行任务。首先介绍一下测量机器人工具坐标的方法及优点，并着重讲解如何测量一个工具坐标系。

5.1.1 测量工具坐标系的方法

1. 手动测量：
   使用测量工具（如卷尺、角度量具等），手动测量末端执行器相对于机器人基坐标系的位置和姿态。这种方法需要人工操作，并且可能不够精确，但对于简单的机器人或快速的测量任务可能是足够的。
2. 基于标定板的视觉测量：
   使用特定的标定板或标志物，将其固定在已知位置，并使用机器人配备的相机或视觉系统拍摄标定板的图像。通过识别标定板上的特征点或图案，并与已知的标定板位置进行比较，可以确定机器人末端执行器相对于基坐标系的位置和姿态。
3. 基于激光测量：
   使用激光跟踪系统，将激光测量设备安装在末端执行器上，并将机器人移动到已知的位置和姿态。激光测量设备可以测量末端执行器到参考点的距离和方向，从而确定末端执行器相对于基坐标系的位置和姿态。
4. 基于机器人自身功能的测量：
   某些高级机器人系统可能配备了自我定位和标定功能，可以通过执行特定的校准程序来测量末端执行器的位置和姿态。这种方法通常依赖于机器人系统内置的传感器和算法，可以实现较高的测量精度和自动化程度。
5. 手眼标定：
   对于配备视觉系统的机器人，通常需要进行手眼标定，以确定机器人末端执行器相对于相机坐标系的位置和姿态。手眼标定通常涉及使用特定的标定物体或模式，通过拍摄多个位置和方向的图像，并分析相机与末端执行器之间的关系来确定标定参数。

   无论使用哪种方法，测量机器人工具坐标都需要精确的操作和分析，以确保获得准确的结果。在进行测量之前，通常需要进行适当的准备工作，例如清理工作区域、校准传感器或相机，以及执行任何必要的校准程序。

5.1.2 工具坐标系测量的优点

1. 精确的工具定位：通过测量工具坐标系，可以精确确定工具在机器人末端执行器上的位置和姿态，从而实现对工具的准确控制和定位。
2. 提高工作精度：正确标定工具坐标系可以确保机器人末端执行器的运动和操作在工作空间内保持准确、精确，提高了机器人工作的精度和可靠性。
3. 灵活的工作环境：通过工具坐标系的测量和标定，可以使机器人在不同的工作环境中使用相同的工具，并且能够自动适应不同的工件或任务，提高了机器人的灵活性和适应性。
4. 简化编程：准确的工具坐标系测量可以简化机器人编程过程，减少编程难度和复杂性。编程人员可以直接使用标定好的工具坐标系进行任务规划和路径生成，而无需考虑工具的位置和姿态。
5. 减少生产成本：通过准确测量和标定工具坐标系，可以避免因工具位置误差引起的生产缺陷和浪费，从而降低生产成本并提高生产效率。
6. 增强安全性：正确标定工具坐标系可以确保机器人在执行任务时工具的位置和姿态与预期一致，减少意外碰撞或损坏工件的风险，提高了工作安全性。
7. 可重复性：一旦工具坐标系标定完成，可以反复使用，保持机器人在不同工作周期内的稳定性和可重复性。

总的来说，机器人工具坐标系测量的优点在于提高了机器人工作的精度、灵活性和可靠性，减少了生产成本和风险，并简化了机器人编程和操作过程。

5.1.3 工具的介绍

对于具体的任务，选择合适的工具至关重要，而工具必须固定到法兰末端才可以工作，因此必须了解法兰的机械结构，如图5.3所示，为本书所用机器人法兰的机械结构示意图，不同的机器人由不同的法兰结构组成，本节仅对库卡 KR6 R700 sixx 型号机械手进行讲解，对于其余型号读者应查阅自身机器人相关文档。

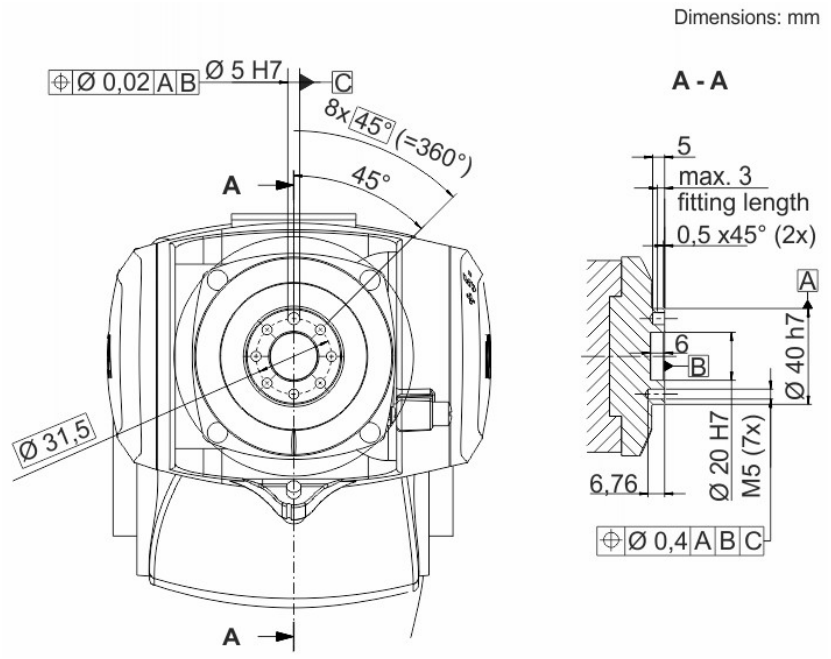
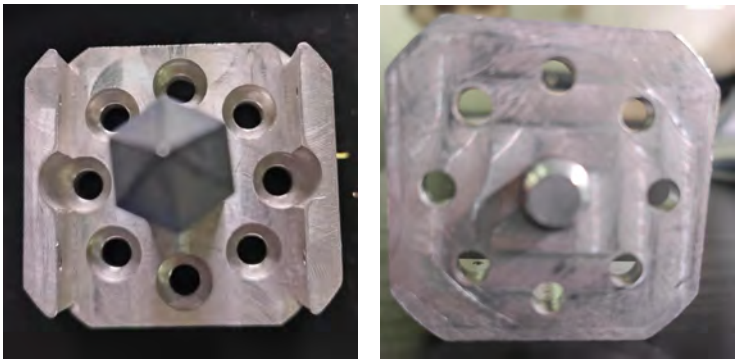


图 5.3: 法兰机械结构

查阅官方文档可知，连接法兰螺纹孔中心一圈直径为 31.5mm，其所需紧固螺纹孔的数量为 7 个（其中一个为零点位置），螺栓尺寸为 M5，拧入深度最小为 5.5mm，最大为 7mm。根据这些信息我们通过三维设计建模软件设计出如图所示的机械结构，即为本次将要测量的工具（如下图5.4所示）。



测量工具正视图　　测量工具底部视图

图 5.4: 待测工具的两侧视图

工具的制作首先要保证其可以牢牢地固定在法兰盘上，所设计出的工具需要按照法兰盘的规格，将工具螺丝孔与法兰螺纹孔对其。如图所示，本次所使用的工具与法兰盘通过 7 颗内六角螺丝进行固定，，确保两者在多次拆卸仍然可以保持良好的精度。

测量工具左侧视图　　测量工具右侧视图

图 5.5: 待测工具正视图和底视图

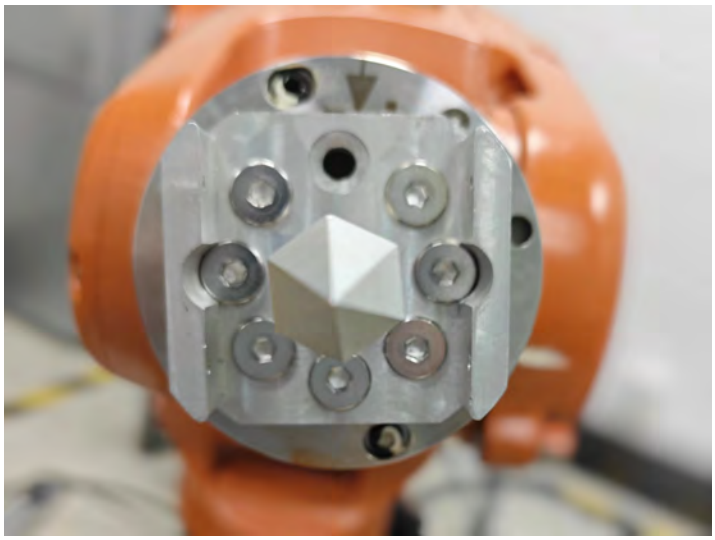
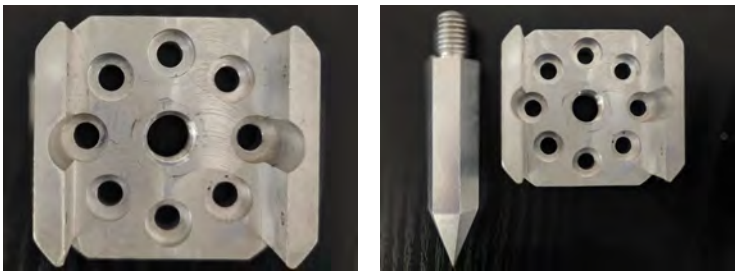


图 5.6: 已安装在法兰末端的工具

同时，为了兼顾后续相机的手眼标定，如图所示，我们将中央尖锥部分做了可拆卸处理，其可以通过扭动下方尖端进行拆卸，进而将相机安装固定上去，方便我们进行后续章节的手眼标定部分。



拆卸后底座　　拆卸后尖锥

图 5.7: 拆卸后的工具

该工具通过工程设计软件设计出后，经过钢结构打印成型，可重复使用次数多，且不易形变，是优秀的测量工具，读者可以根据自己需求自行设计加工，如有需要可以联系我们获取相关模型。

5.1.4 测量工具坐标系方法及操作步骤

在进行工具测量时，即给安装在法兰连接处的工具分配一套笛卡尔坐标系。工具坐标系以用户设定的一个点作为其原点。此点称做 TCP（Tool CenterPoint，工具中心点）。可以测量多少个工具坐标系取决于 WorkVisual 中的配置（WorkVisual 介绍见后续章节）。默认：16 个 TOOL 坐标系。变量：TOOL_DATA[1 …16]。

工具的测定由两步组成，如图5.8所示，首先要确定工具坐标系的原点，一般通过 XYZ 四点法或 XYZ 参照法进行测量，此时可以得到工具相对于法兰的 XYZ 左边。此外，还要确定工具坐标系的姿态，即欧拉角 ABC，此时可以通过 ABC2 点法或 ABC 世界坐标系法。如果测量数据已知，则可将其直接输入。

在本书中，由于工具类型更加使用于 XYZ 四点法以及 ABC2 点法，因而仅对这两个方法进行操作演示，其余方法读者可自行查阅相关书籍。

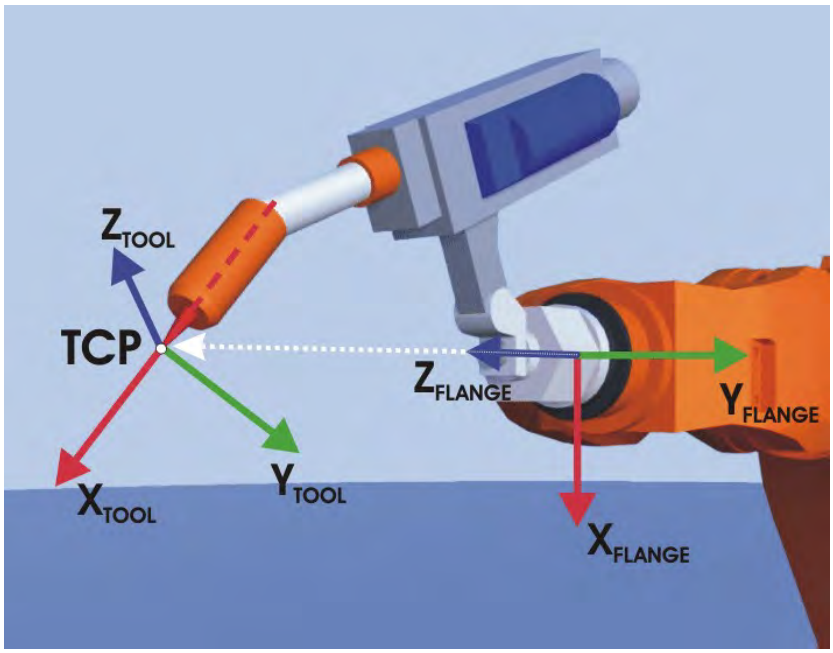


图 5.8: TCP 测量原理

1. **测量工具坐标系的原点：XYZ 4 点法**

   如图5.9所示，将待测量工具的 TCP 从 4 个不同方向移向一个参照点。参照点可以任意选择。机器人控制系统从不同的法兰位置值中计算出 TCP。移至参照点的 4 个法兰位置，彼此必须间隔足够远，并且不得位于同一平面内。具体操作步骤如下所示。

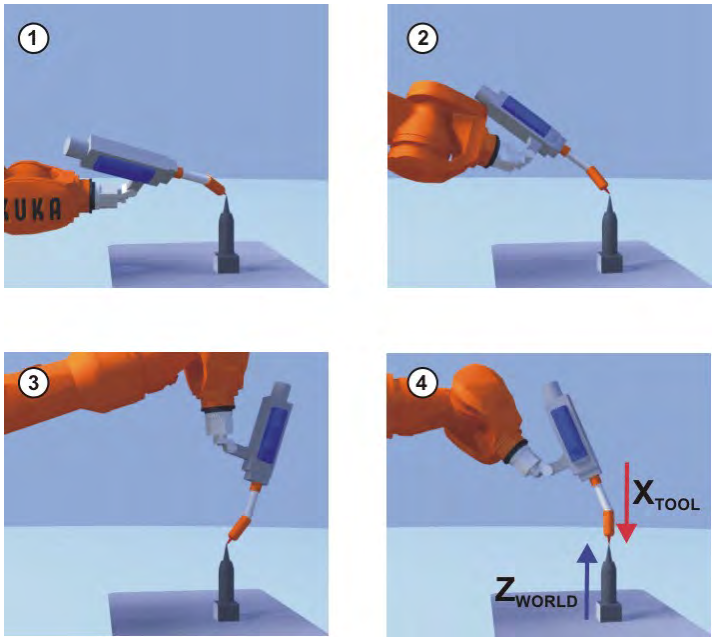


图 5.9: XYZ4 点法

(1) 连接法兰处已经安装了待测定的工具，如图5.10所示，运行方式调整为 T1。

图 5.10: 待测工具

(2) 在主菜单中选择“投入运行”→“工具/基坐标管理”。

图 5.11: 工具菜单

(3) 在当前页面，注意要选择下方的工具工件栏，点击“添加”，添加一个工具坐标系。

图 5.12: 添加工具坐标

(4) 当前页面即为我们的工具/基坐标系管理页面，我们可以对该工具坐标系命名，并且在名称左侧的选项中选择“工具”。

图 5.13: 工具/基坐标系管理界面

(5) 单击下方测量按钮，在下拉页面中选择“XYZ4 点法”。

图 **5.14:** 单击 XYZ4 点法

(6) 在当前页面选择测量点 1。

图 5.15: 选择测量点 1

(7) 使用示教器，将法兰末端工具移动到我们所做的参考点，如图5.16所示。

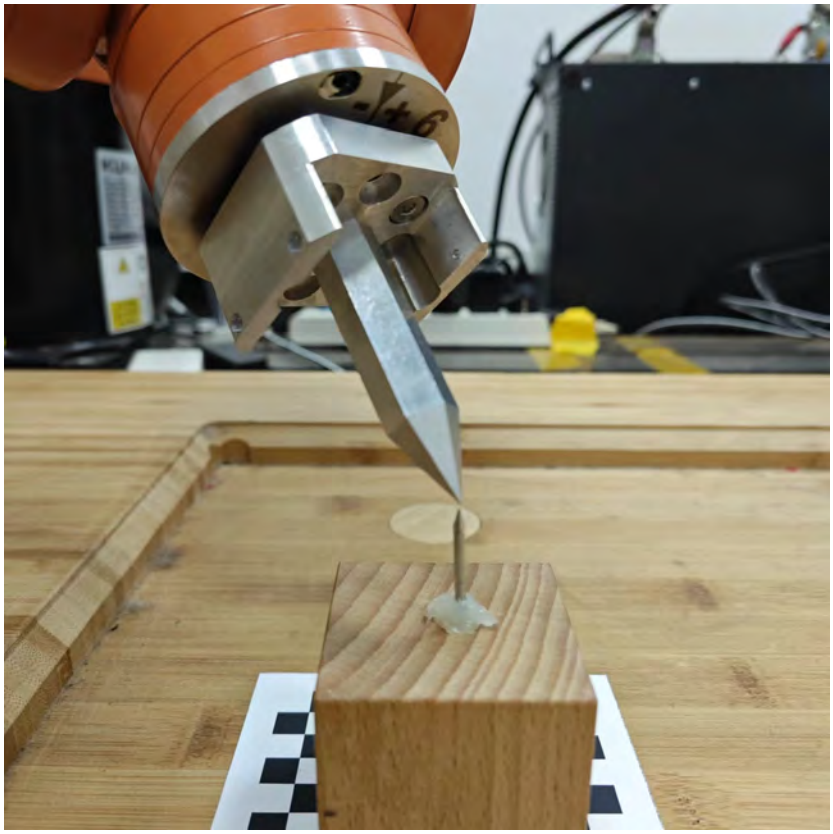


图 5.16: 首次将末端工具驶至参考点

(8) 单击下方按钮“Touch-Up”。

图 5.17: 单击 Touch-Up

(9) 如图所示，将末端工具按其他方向驶至参考点，重复三次步骤⑥和⑦，注意最后一次要尽量保持垂直驶至参考点。

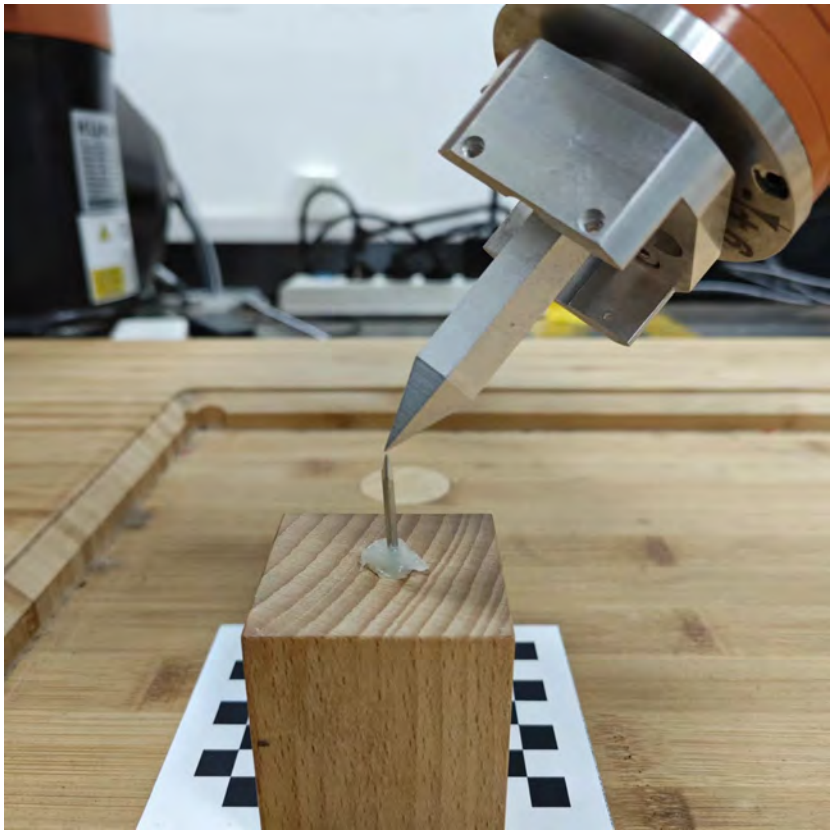


图 5.18: 再次驶至参考点

图 5.19: 第三次驶至参考点

图 5.20: 最后驶至参考点

(10) 得到 XYZ 测量结果，单击“保存”，完成步骤。

图 5.21: XYZ 测量结果

2. **确定姿态：ABC 世界坐标法**

用户将工具坐标系的轴调整为与世界坐标系的轴平行，机器人控制器从而得知工具坐标系的取向，如图所示。

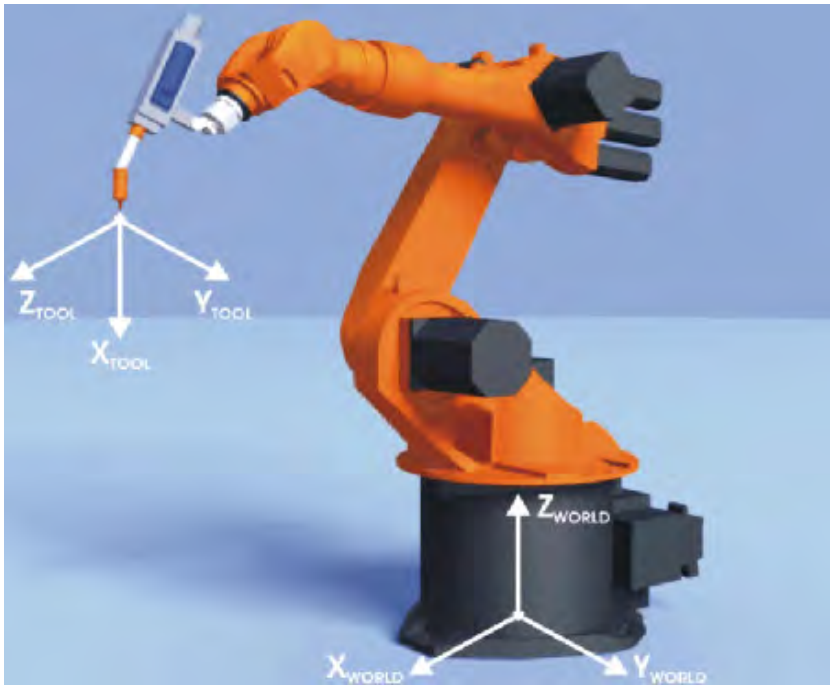


图 5.22: ABC 世界坐标法

此方法有两种方式：

a)5D：用户将工具的作业方向告知机器人控制系统。该作业方向被默认为 X 轴。其它轴的姿态将由系统确定，用户无法改变。系统总是为其它轴确定相同的姿态。如果之后必须对工具重新进行测定，比如在发生碰撞后，仅需要重新确定作业方向。而无需考虑作业方向的转度。应用范围：例如：MIG/MAG 焊接，激光切割或水射流切割等。

b)6D：用户将所有 3 根轴的方向告知机器人控制系统。应用范围：例如：焊钳、抓爪或粘胶喷嘴等。在本书中，由于我们的工件类型，我们选择 5D 方式来进行测量姿态，6D 方法具体操作步骤与该操作类似。

(1) 连接法兰处已经安装了待测定的工具，工具坐标的 XYZ 已测量，运行模式为 T1。
(2) 在主菜单中选择“投入运行”→“工具/基坐标管理”。
(3) 选择要测量的工具编号。单击“编辑”。

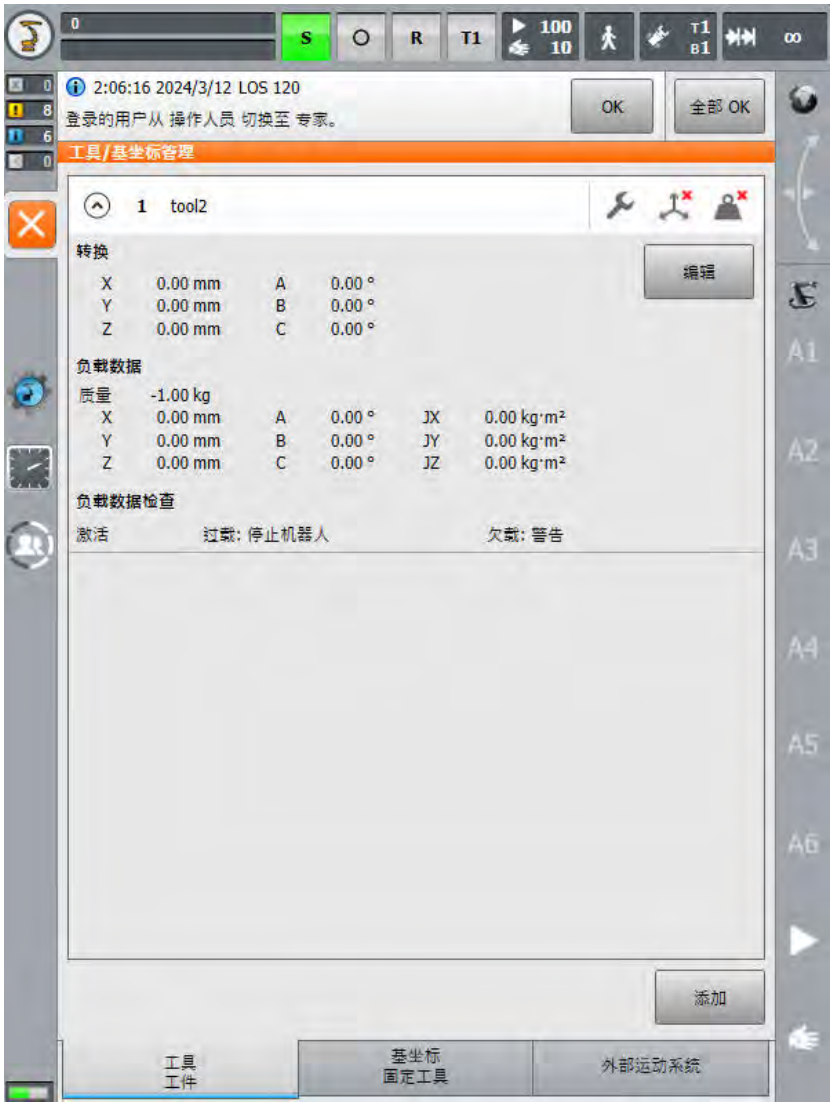


图 5.23: 选择要测量的工具

(4) 点击“测量”→“ABC 世界法”。

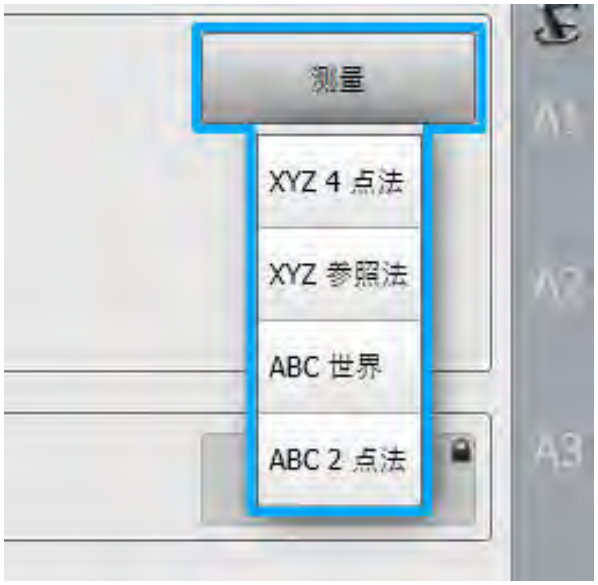


图 5.24: 点击 ABC 世界法

(5) 如图所示，选择 5D。

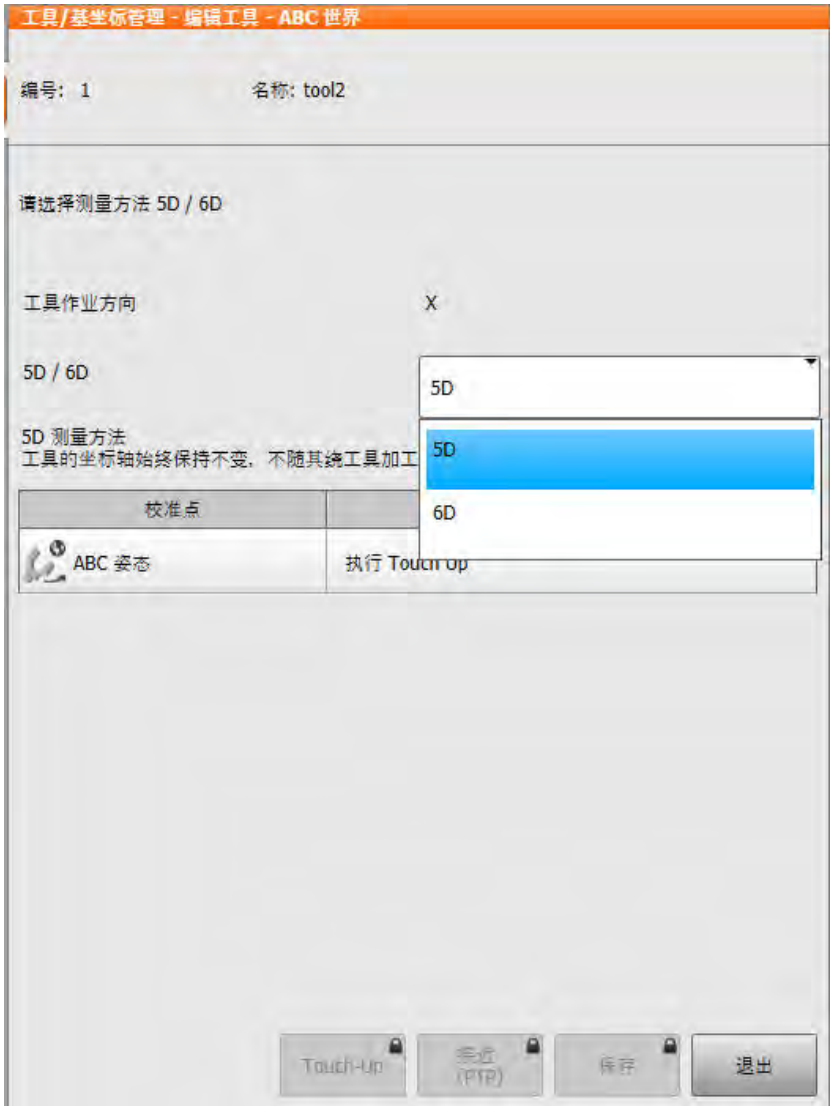


图 **5.25:** 选择 5D

(6) 使用示教器，移动机器人的法兰末端，使得工具的 X 轴正向与世界的 Z 轴负向平行对齐。+XTOOL 与-ZWORLD 平行对齐。（+XTOOL = 加工方向）

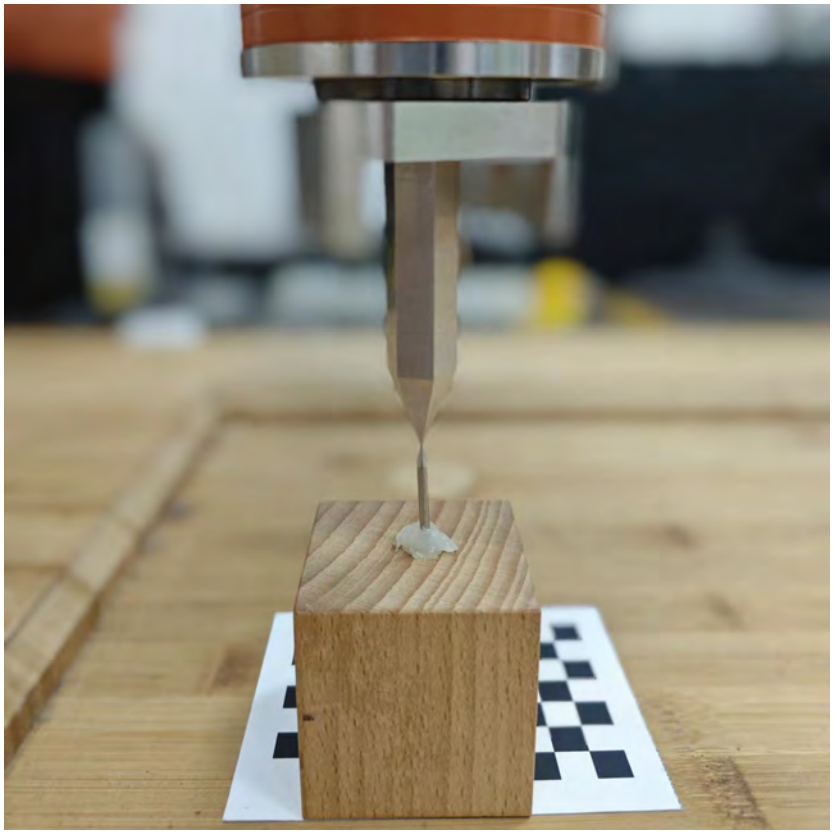


图 5.26: 工具的 X 轴与世界 Z 轴平行

(7) 点击“Touch-Up”, 再点击“保存”即可。

3. **数字输入**

如若事先清楚工具数据，可以通过手动输入。可能的数据源有 CAD、外部测量的工具、工具制造商的数据，卸码垛机器人有 4 根轴，必须在工具数据中以数字形式输入。不能使用不同的 XYZ 和 ABC 法，因为在这类机器人上要变换方向有一定限制。

(1) 下列数据已知：

a) 相对于法兰坐标系的 X、Y、Z。

b) 相对于法兰坐标系的 A、B、C。

(2) 运行方式为 T1。

(3) 在主菜单中选择“投入运行”→“工具/基坐标管理”。

(4) 在下方点击“添加”按钮，这两步骤与 XYZ4 点法步骤相同，图示略。

(5) 输入工具数据。用按键继续确认。

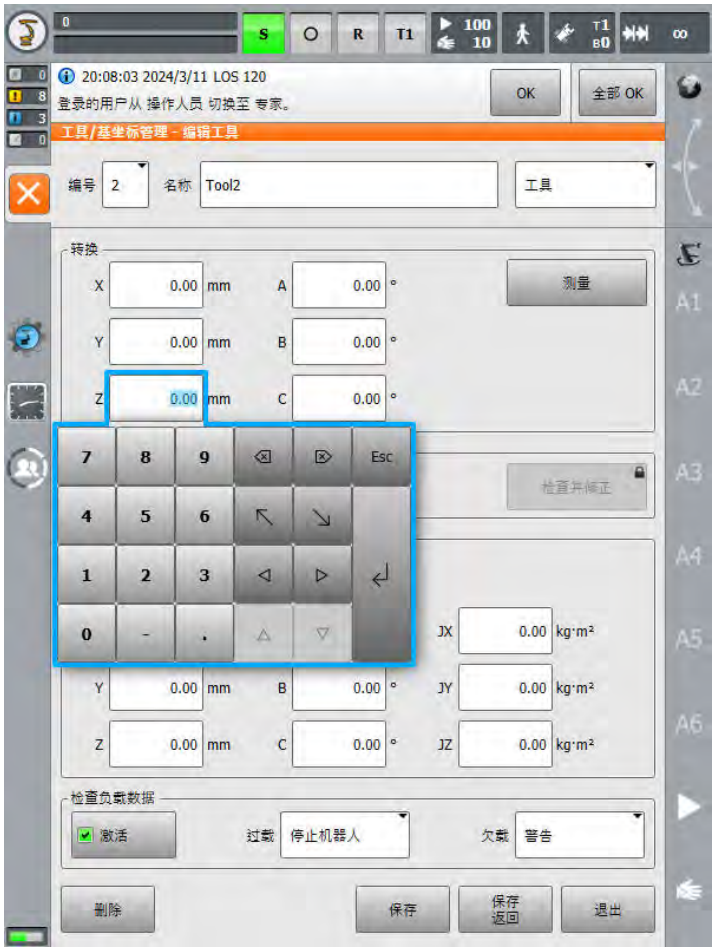


图 5.27: 手动输入结果

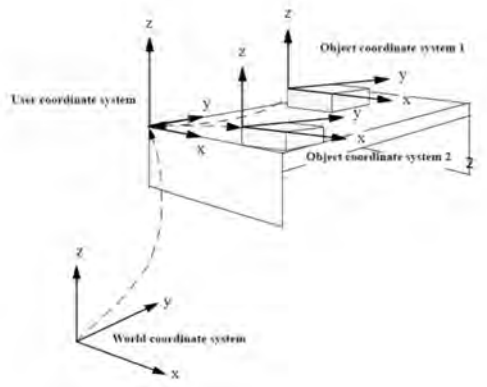
(6) 输入负载数据。（如果要单独输入负载数据，则可以跳过该步骤。）

(7) 点击保存。

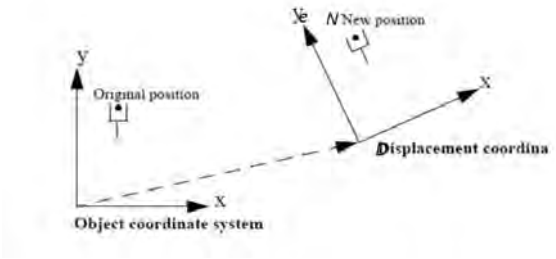
5.2 测量基坐标

基坐标系测量表示根据世界坐标系在机器人周围的某一个位置创建坐标系。其目的是使机器人的运动以及编程设定的位置均以该坐标系为参考。

在机器人线性运行时，按照用户自己定义的坐标系的方向在工件平面上作线性运动，而不拘泥于系统提供的基座坐标系和世界坐标系这几种固定的坐标系。此外，当工作台面与机器人之间的位置发生相对移动时，只需要更新工件坐标系，即不需要重新示教机器人轨迹，从而很方便的实现轨迹的纠正。



工件坐标系的工件



工件坐标系移动

图 5.28: 工件坐标系

因此，对于特定的工作场景，利用基坐标系可以大大减少我们的工作量。

5.2.1 测量基坐标系的优点及方法

1. 测量基坐标系的优点：

   1) TCP 可以沿着工作面或工件的边缘手动运行。

   2) 测量基坐标系可以为机器人提供准确的参考坐标系，从而可以进行坐标转换，确保机器人能够准确执行各种运动和操作。。

   3) 通过测量基坐标系，机器人可以准确识别工作区域内的工件位置和姿态，从而可以执行精准的抓取、放置或加工操作。4) 可以参照基坐标对这些点示教。如果必须移动基坐标，例如由于工作面被移动，这些点也随之移动，但不必重新进行示教。

2. 测量基坐标系的方法

   1) 激光跟踪法：使用激光跟踪系统，通过激光测量工具和参考点的位置，确定机器人末端执行器的位置和姿态，并从而确定基坐标系。

   2) 机械示教法：通过手动操作机器人，将末端执行器移动到已知位置，并使用示教器或手动操纵器记录这些位置，然后计算基坐标系的位置和姿态。

   3) 视觉定位法：使用摄像头和视觉算法，识别工作区域内的标志物或特征点，通过计算这些点的位置和姿态，确定机器人的基坐标系。

3. KUKA 机器人常用测量基坐标方法

   1) 三点法。

   如图5.29所示，移至新基础坐标系的原点和其它 2 个点。这 3 个点定义了新的基础系。

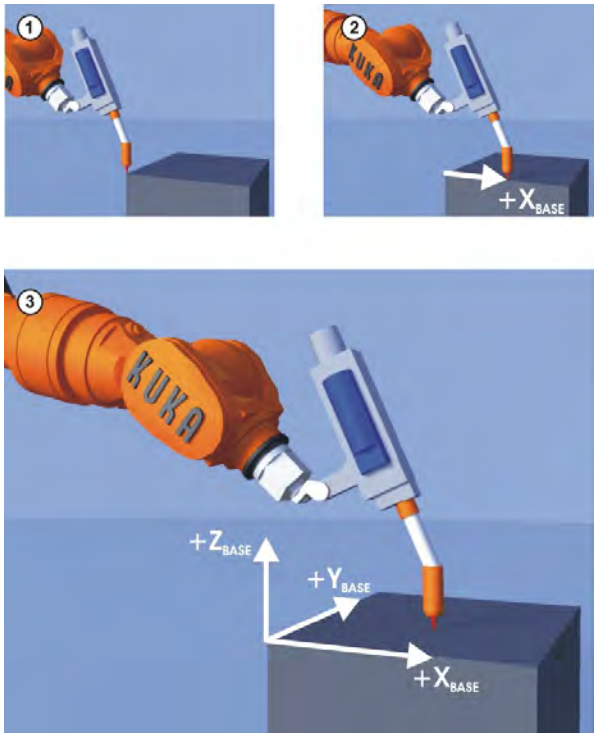


图 5.29: 3 点法简要步骤

2) 间接法。

如图所示，当无法移入基准原点时，例如，由于该点位于工件内部，或位于机器人作业空间之外时，须采用间接的方法。此时须移至基准的 4 个点，其坐标值必须已知。机器人控制系统将以这些点为基础对基准进行计算。

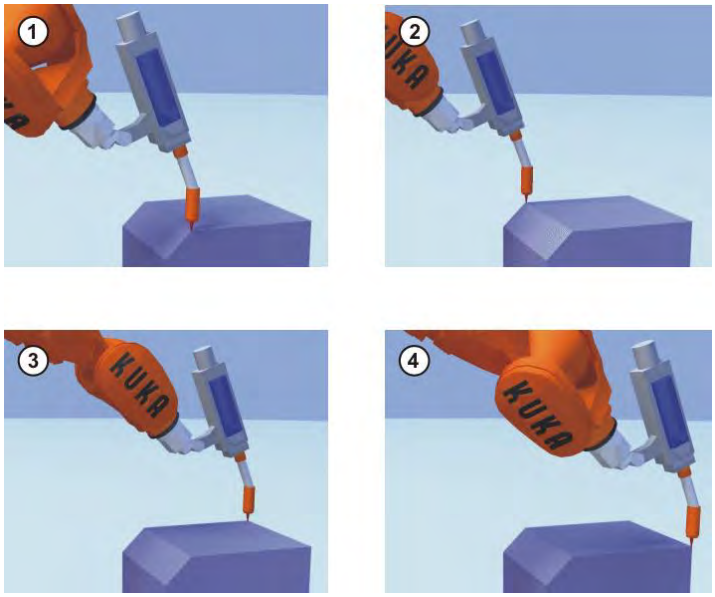


图 5.30: 间接法

在本书中，由于 3 点法的适用范围更广，我们仅对 3 点法进行详细说明，间接法的使用与该方法类似。

5.2.2 测量基坐标系 3 点法

1. 首先必须在连接法兰上装有一个已测量过的工具。
2. 运行方式为 T1。
3. 在主菜单中选择“投入运行”→“工具/基坐标系管理”。
4. 下方选项栏中选择“基坐标固定工具”栏，并且点击右下角“添加”。

图 5.31: 基坐标固定工具栏

5. 再编辑基坐标界面，我们可以为其选择编号和名称，之后点击右方的“测量”按钮，选择“3 点”。

图 **5.32:** 编辑基坐标界面

图 5.33: 选择 3 点

6. 此时，在校准点下方，选择原点一行，用 TCP 驶至新基座的原点。点击“Touch-Up”。

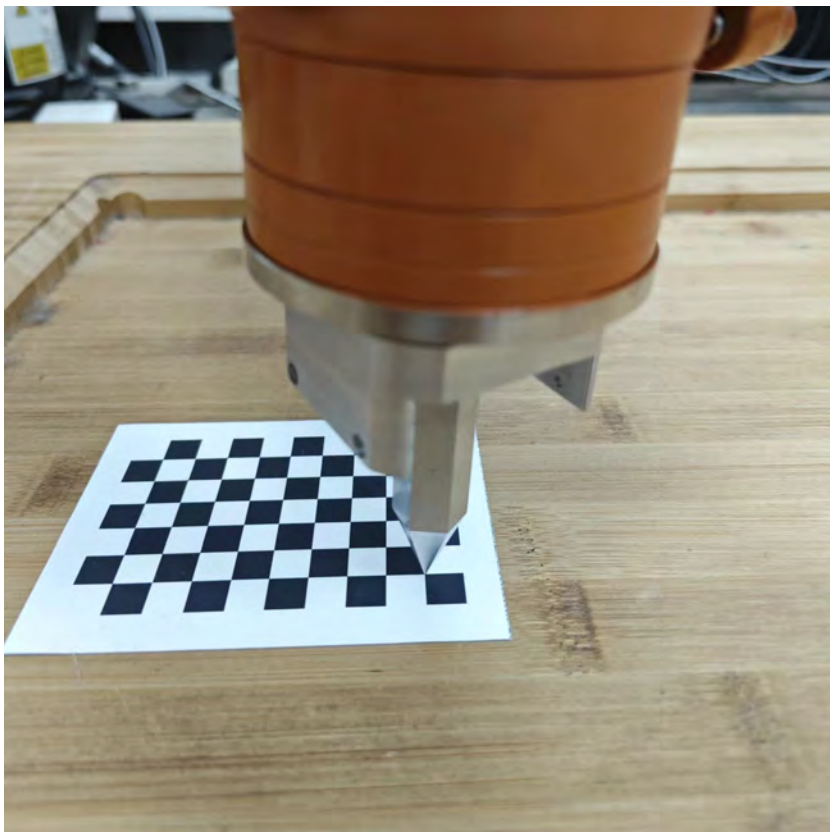


图 5.34: 移到新坐标系原点

7. 此时在将 TCP 移至新基座正向 X 轴上的一个点，点击“Touch-Up”。

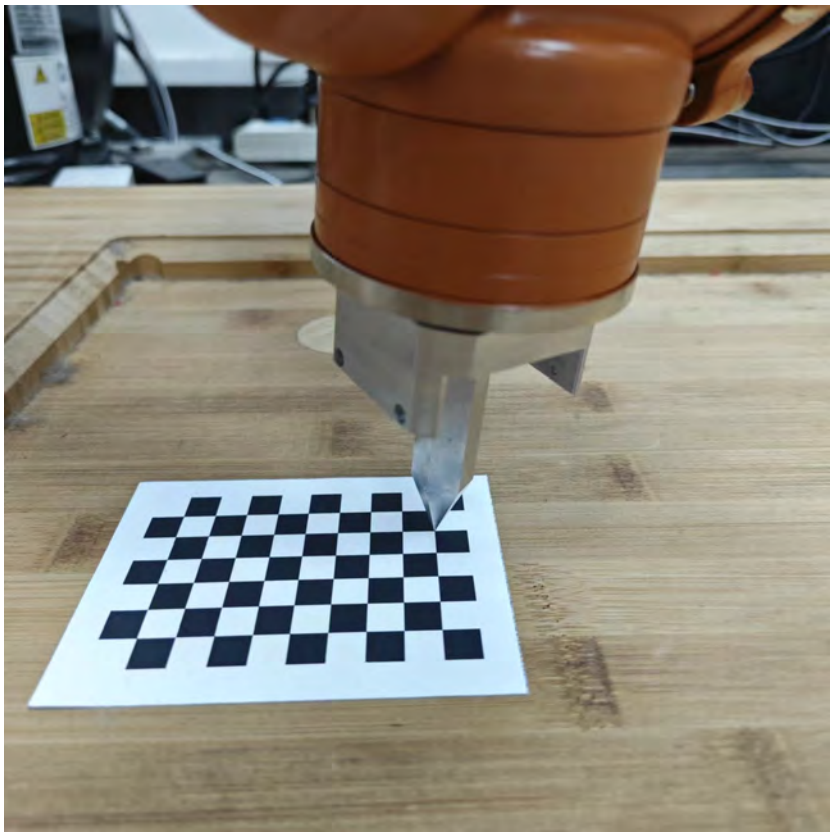


图 5.35: 移到 X 轴上一点

8. 用 TCP 在 XY 平面上接近带正 Y 值的一点。点击“Touch-Up”。

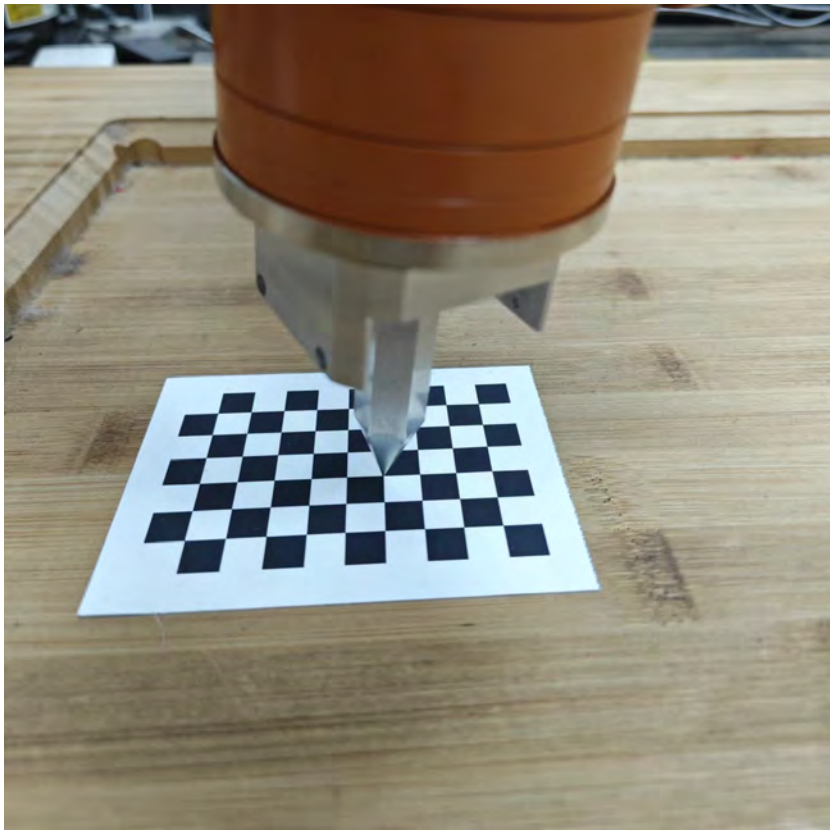


图 5.36: 移至 XY 平面接近带正 Y 值的一点

9. 点击“保存”。

5.2.3 工具、基坐标系改名

有时需要对我们已经测量的工具或基坐标系进行更换名称，为此有必要对其进行操作讲解，温习上面所学内容，读者可对其进行尝试，加深操作熟练度。

1. 运行方式切换为 T1。
2. 在主菜单中选择“投入运行”→“工具/极坐标管理”。
3. 单击选择要更改的坐标系，右侧“编辑”按钮。

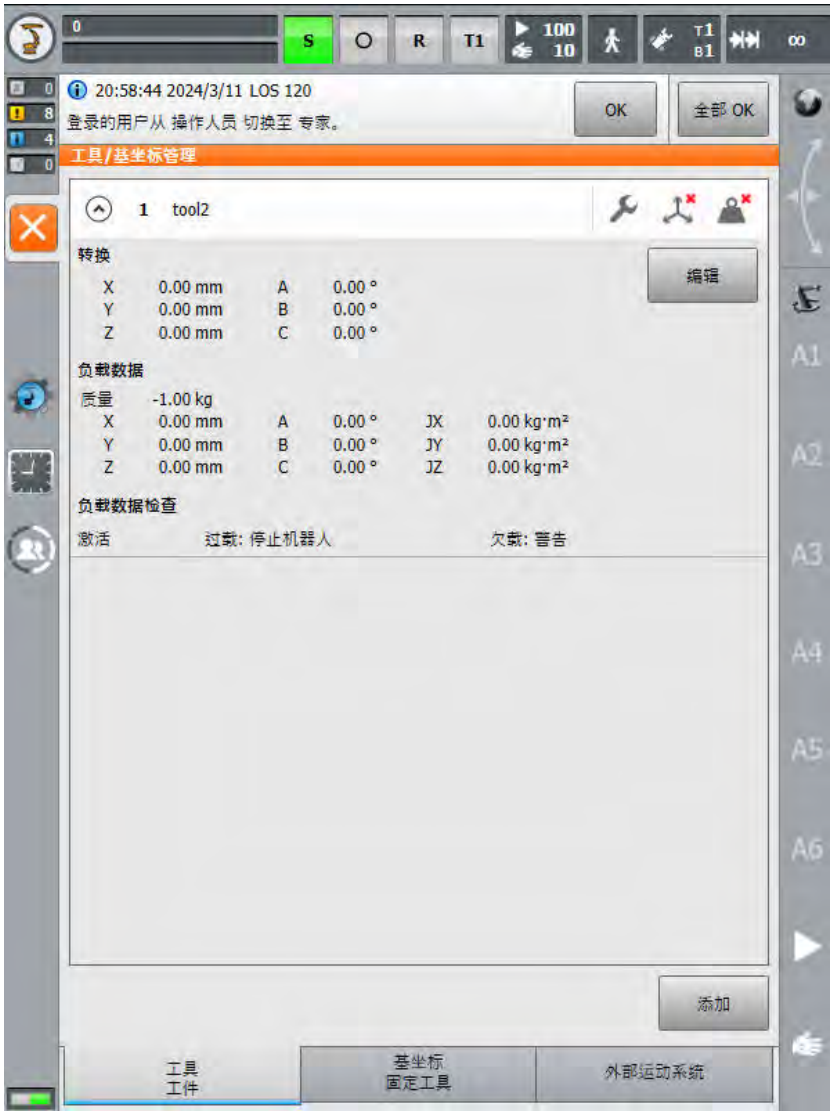


图 5.37: 更改名称界面

4. 此时可以选择更改名称，或者对该坐标系重新测量。

第 6 章 机器人的编程

通过前面章节的学习，读者应已经初步了解机器人的大致使用，一些初级的操作，然而我们仅仅只是对 KUKA 机器人有了个大概的了解。进入本章，我们将学习并掌握机器人的一些简单编程，读者可以创建自己的程序，让机器人按照程序所预定的轨迹进行运动。

6.1 KUKA 机器人编程基础

在本节内容，主要给各位读者讲解如何新建程序，文件，选择、打开以及编辑程序部分，难度较低，对于刚刚上手 KUKA 机器人的读者来说，应按照操作步骤完全尝试一遍。

6.1.1 KUKA 导航器

当时示教器打开后，映入眼帘的就是 KUKA 文件导航器界面（如图6.1所示），用户可在导航器中管理程序及所有系统相关文件。

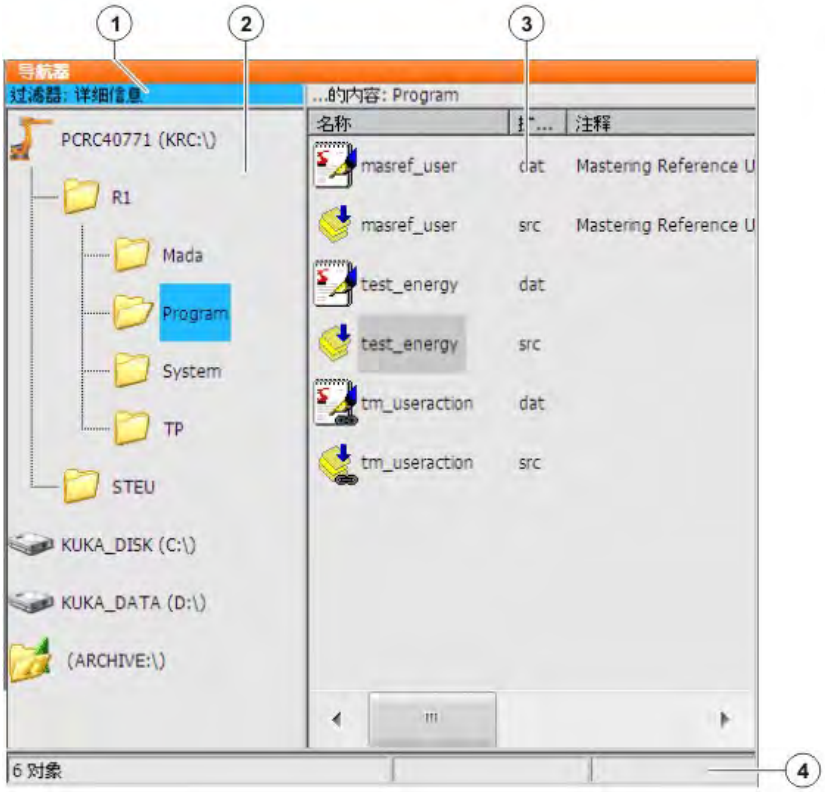


图 **6.1:** 导航器

①——标题行，其左侧区域显示所选的过滤器，右侧区域显示哪些目录和驱动器，取决于用户组及配置。

②——目录结构，目录和驱动器概览。显示哪些目录和驱动器，取决于用户组及配置。

③——文件清单，显示在目录结构中选中了的目录或驱动器内容。以何种形式显示程序，则取决于所选择的过滤器。文件列表分为以下几列：

a) 名称：目录名称或文件名称。

b) 拓展名：文件拓展名，此列在应用人员用户组中不显示。

c) 注释：注释作用。

d) 属性：操作系统和基本系统属性，此列在应用人员用户组中不显示。

e) 规格：文件大小，单位为千字节，此列在应用人员用户组中不显示。

f) # ：文件上的更改数量。

g) 修改时间：最后一次修改的日期和时间。

h) 创建时间：创建的日期和时间，此列在应用人员用户组中不显示。

④——状态行，状态行可显示下列信息：

a) 选中的对象

b) 进行中的动作

c) 用户对话

d) 要求用户输入

e) 安全询问

6.1.2 新建文件夹

编程时往往需要将一个项目中的程序放在同一个文件夹中，接下来我们将带着大家创建一个文件夹，从而学会如何使用。

1. 根据要显示的文件夹，可能至少需要专家用户组。
2. 在目录结构中选定要在其中创建新文件夹的文件夹，例如 R1/Program 下，光标放在文件夹上，点击左下角“新”。

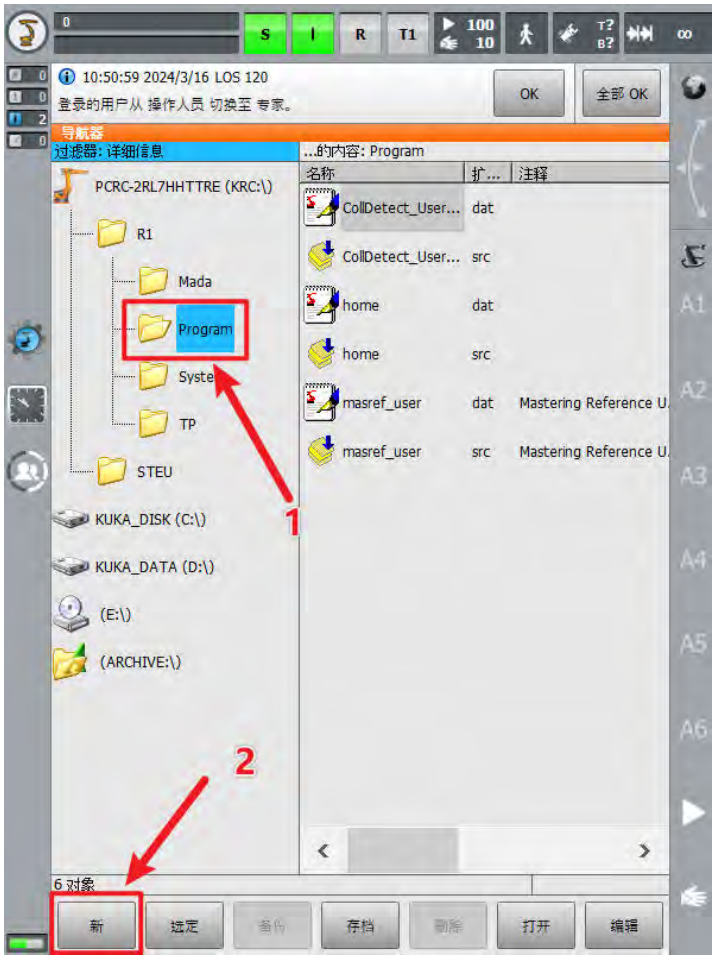


图 6.2: 新建文件夹

3. 为新文件夹命名，并点击“OK”。

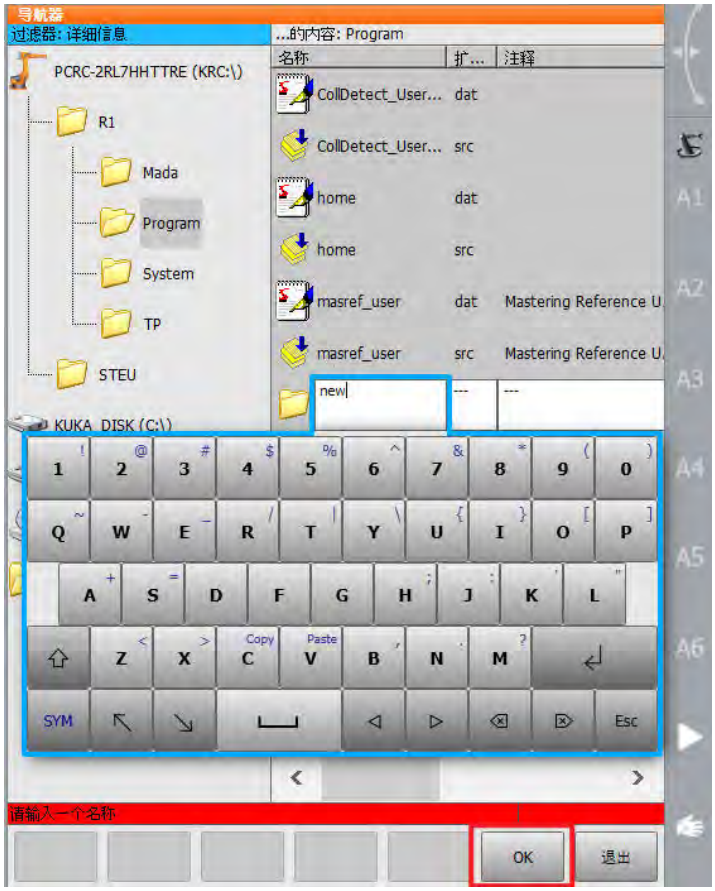


图 6.3: 为其命名并确定

4. 完成文件夹的创建。

6.1.3 新建程序

在 KUKA 文件格式中，需要在已经创建的程序模板中进行编程，不同程序模板有不同的作用，常用的模板为 Modul 模块，我们以此为例进行操作演示。

1. 用户组切换为“专家用户组”。
2. 在当前目录下，点击左下角“新”。

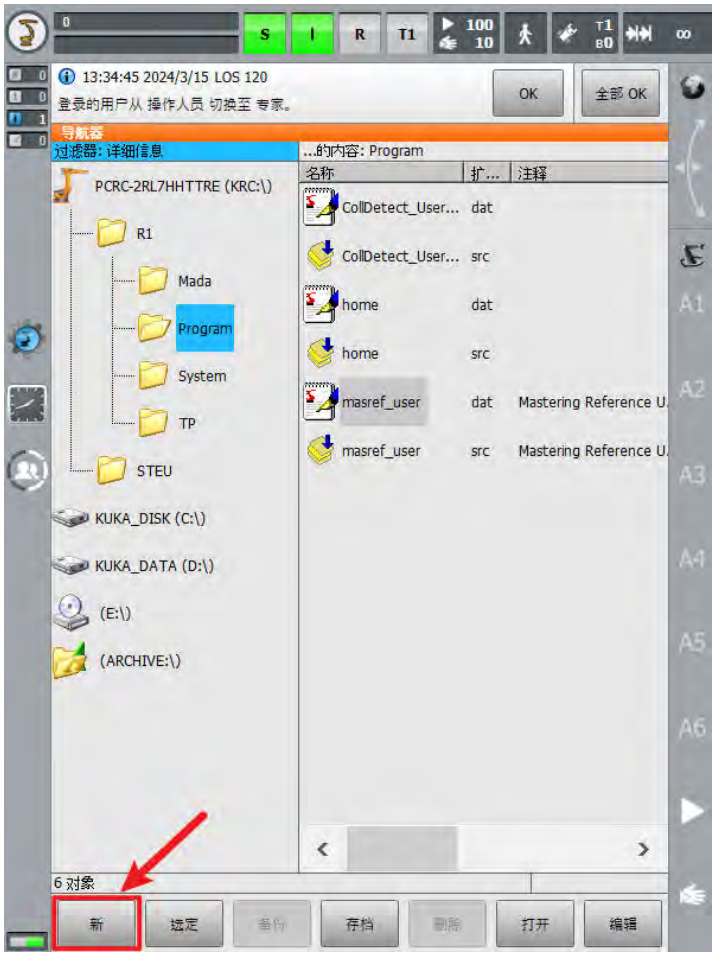


图 6.4: 点击左下角“新”按钮

3. 在该界面左侧选择我们要创建的模板，这里选择“Modul 模块”，点击右下角“OK”。

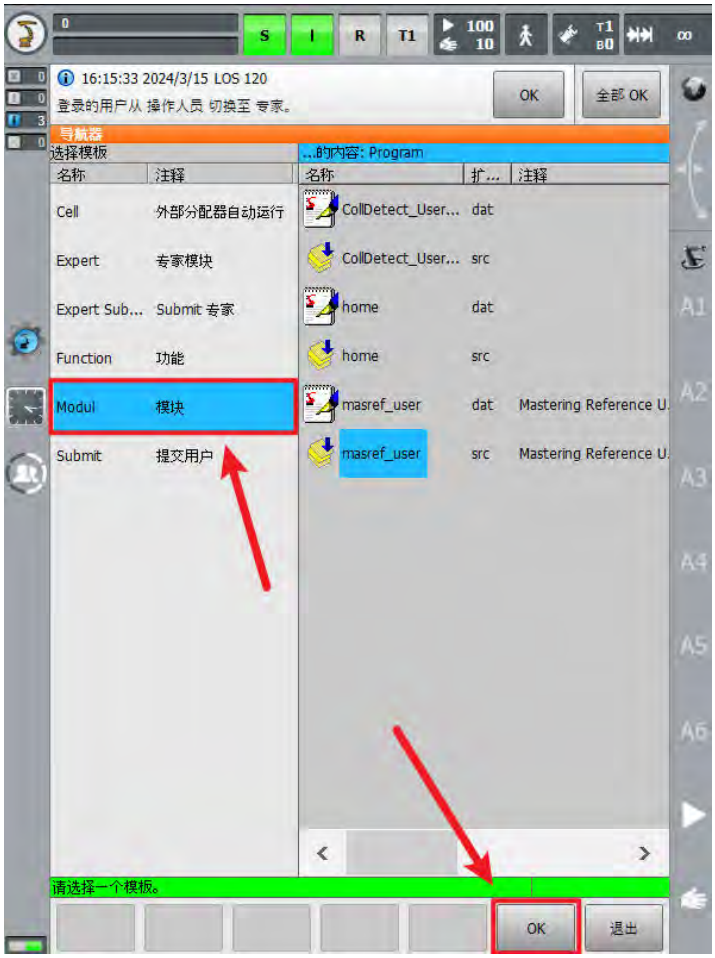


图 **6.5:** 选择模板并确定

4. 为该程序命名，之后点击“OK”。

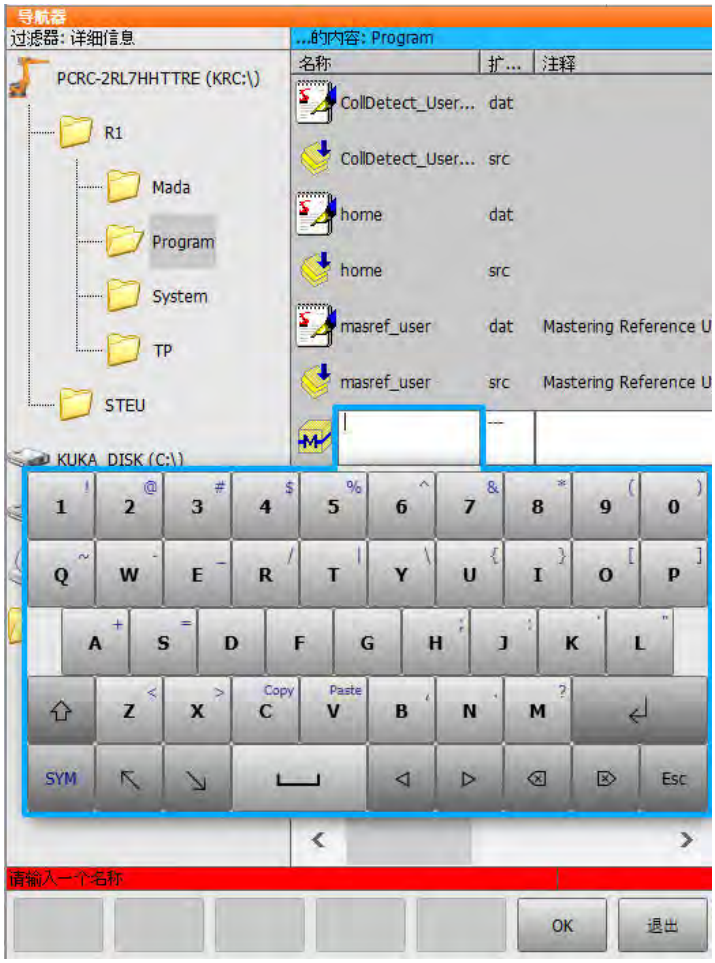


图 **6.6:** 为程序命名

5. 创建成功，在该目录即可查看。

6.1.4 文件或文件夹重命名

有时会对已创建的文件或文件夹重新命名，下方为具体操作步骤。

1. 用户权限至少为专家。
2. 选择我们要改名的文件或文件夹，在右下方点击“编辑”，选择“改名”。

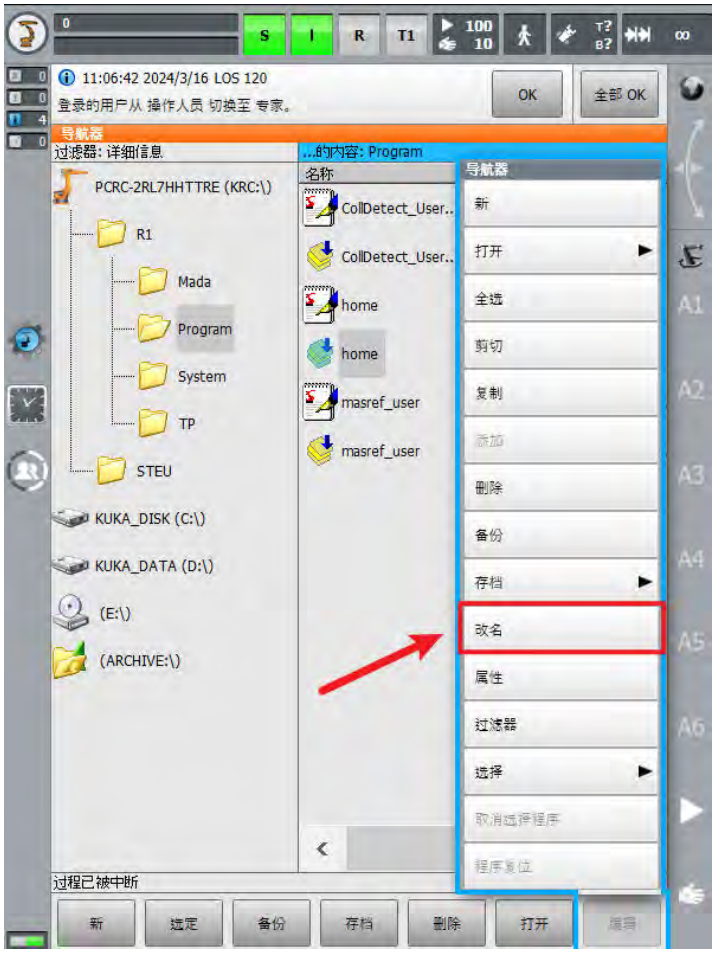


图 6.7: 为文件或文件夹改名

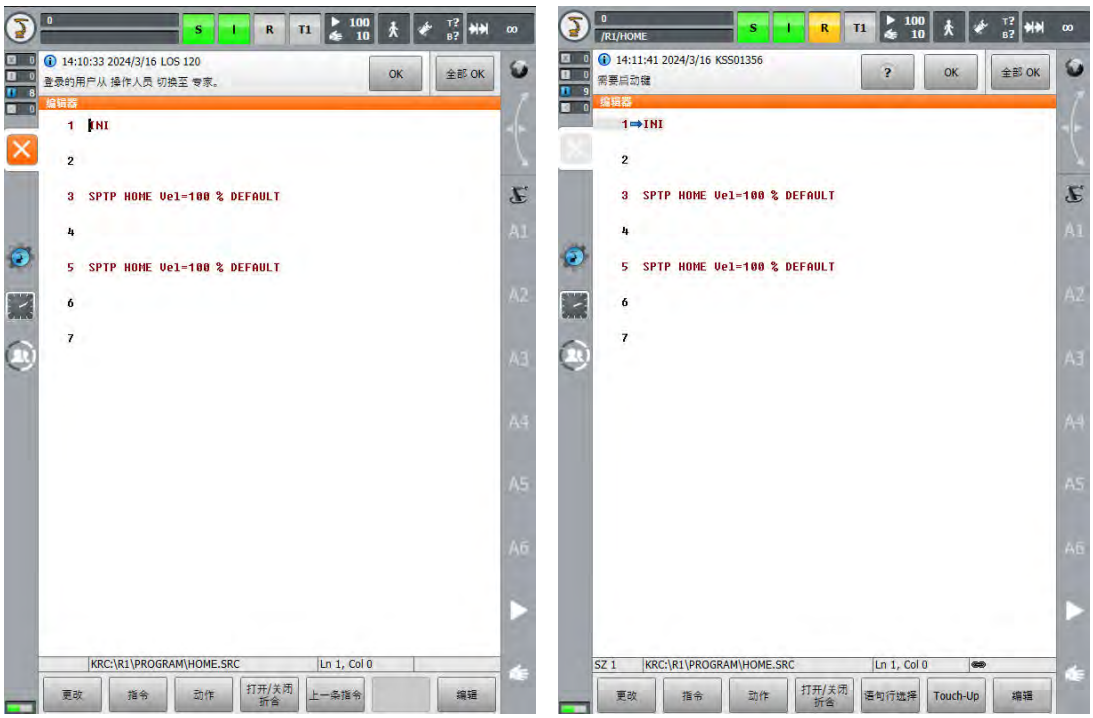
3. 编辑一个新的名称，点击“OK”，即更改完毕。

6.1.5 打开或选定程序

对于所创建的一个程序而言，有两种方式可以打开，要么打开一个程序，要么选定一个程序，打开程序模式仅仅能对程序进行修改，无法运行，只有在选定程序情况下才可运行程序。

在打开程序后，示教器界面如图所示，左侧可以点击叉号关闭，存在一处光标闪烁，往往在该界面对程序进行修改。在选定程序后，示教器界面如图所示，在某行代码左侧有蓝色箭头，上方解释器 R 为黄色, 左侧叉号灰色，只有在该界面才可以进行程序的调试与运行。在程序显示和导航器之间可以来回切换。

我们在下面具体讲打开程序与选定程序的区别。



打开程序　　选定程序

图 6.8: 打开和选定程序界面

1. 区别
   1) 程序已选定
      i. 程序指针将被显示。
      ii. 程序可以启动。
      iii. 可以有限地对程序进行编辑。选定的程序尤其适用于应用人员用户组进行编辑的情况。例如：不允许使用多行的 KRL 指令（例如 LOOP … ENDLOOP）。
      iv. 从在取消选择时，无需回答安全提问即可应用更改。如果对不允许的更改进行了编程，则会显示出一则故障信息。
   2) 程序已打开
      i. 程序不能启动。
      ii. 程序可以编辑。打开的程序尤其适用于专家用户组进行编辑的情况。
      iii. 关闭时会弹出一个安全询问。可以应用或取消更改。
2. 选择和取消选择程序

   如果在专家用户组中对一个选定程序进行了编辑，则在编辑完成后必须将光标从被编辑行移开至另外一行中。只有这样才能保证在程序被取消选择时可以保存编辑内容。
   1) 运行方式为 T1、T2、AUT 均可。
   2) 在导航器中选定程序并按选择。在编辑器中显示该程序。至于选定的是一

个模块、还是一个 SRC 文件或一个 DAT 文件则无关紧要。编辑器中始终显示 SRC 文件。

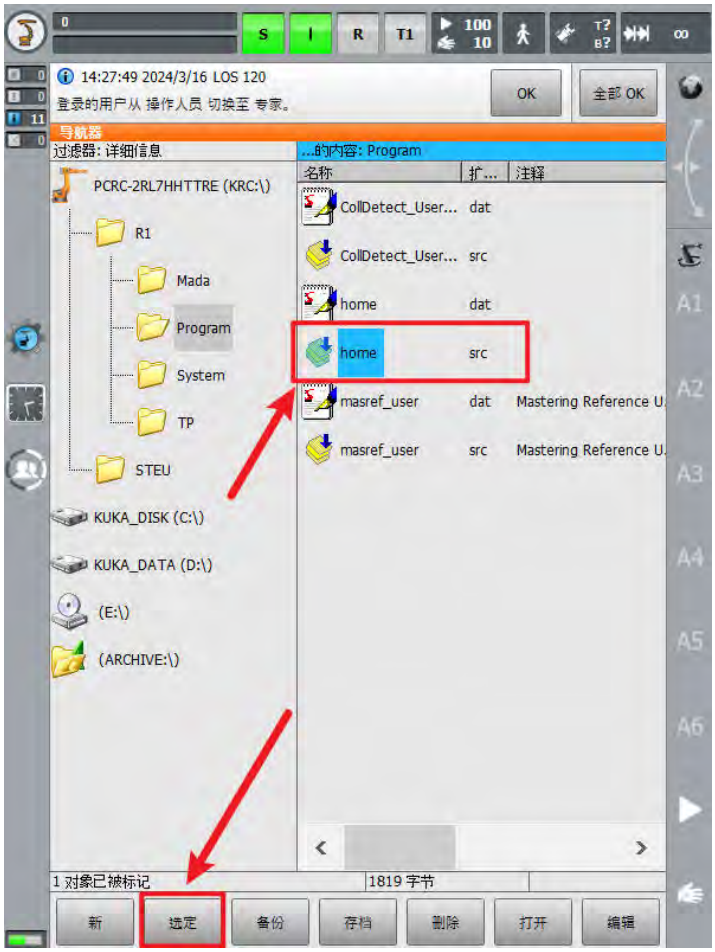
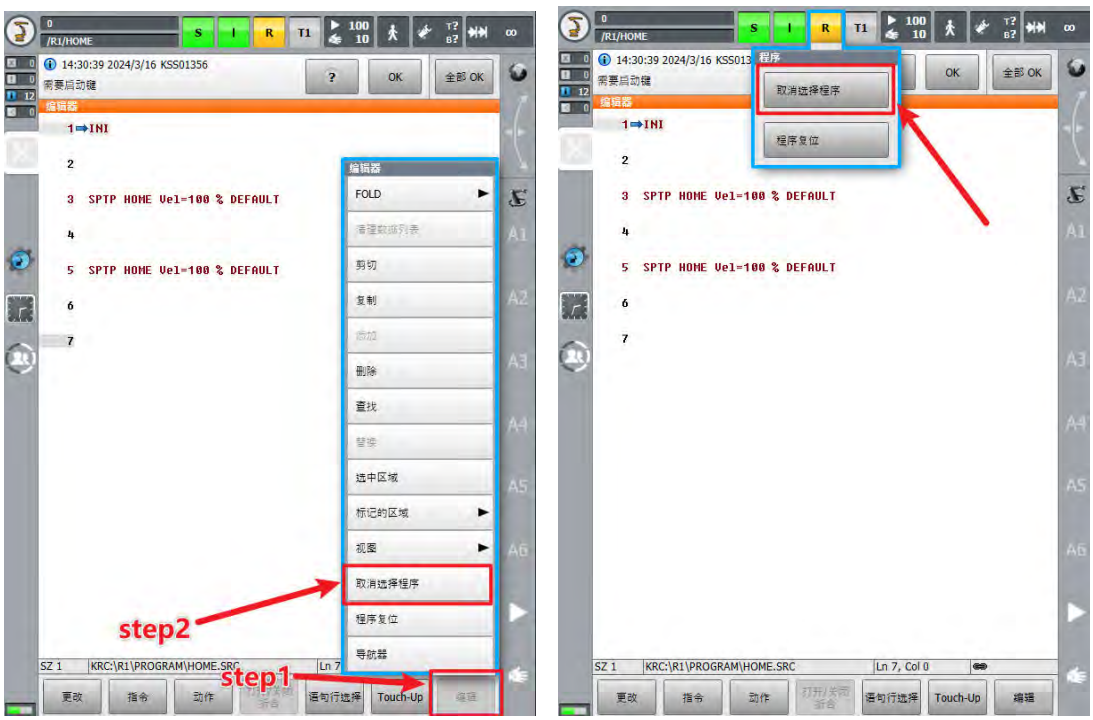


图 **6.9:** 选定 home

3) 根据需要启动或编辑程序。
4) 重新取消选择程序：选择编辑 → 取消选择程序。或者：在状态栏中触摸状态显示机器人解释器。一个窗口打开。选择取消程序。如果程序正在运行，则在取消程序选择前必须将程序停止。如果已选定了一个程序，则状态显示机器人解释器会显示该程序。



方式一　　方式二

图 6.10: 取消选择程序

5) 如图6.11所示，在程序被选定时，①表示语句指针，当前要执行的程序位置，②为鼠标光标，③为程序的路径和文件名，④表示程序中光标当前的位置，⑤表示该图标显示程序已被选定。

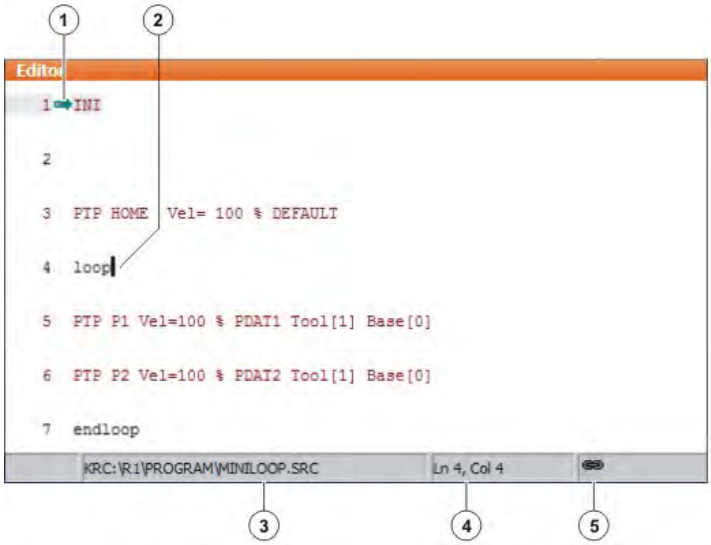


图 6.11: 程序已选定

3. 打开程序

1) 运行方式可以为 T1、T2 或 AUT，然而在 AUT 模式下不能对其进行编辑。

2) 在导航器中选定程序并按打开，当时示例为打开 home 文件。编辑器中将显

示该程序。如果选定了一个模块，SRC 文件将显示在编辑器中。如果选定了一个 SRC 或 DAT 文件，则相应的文件会显示在编辑器中。

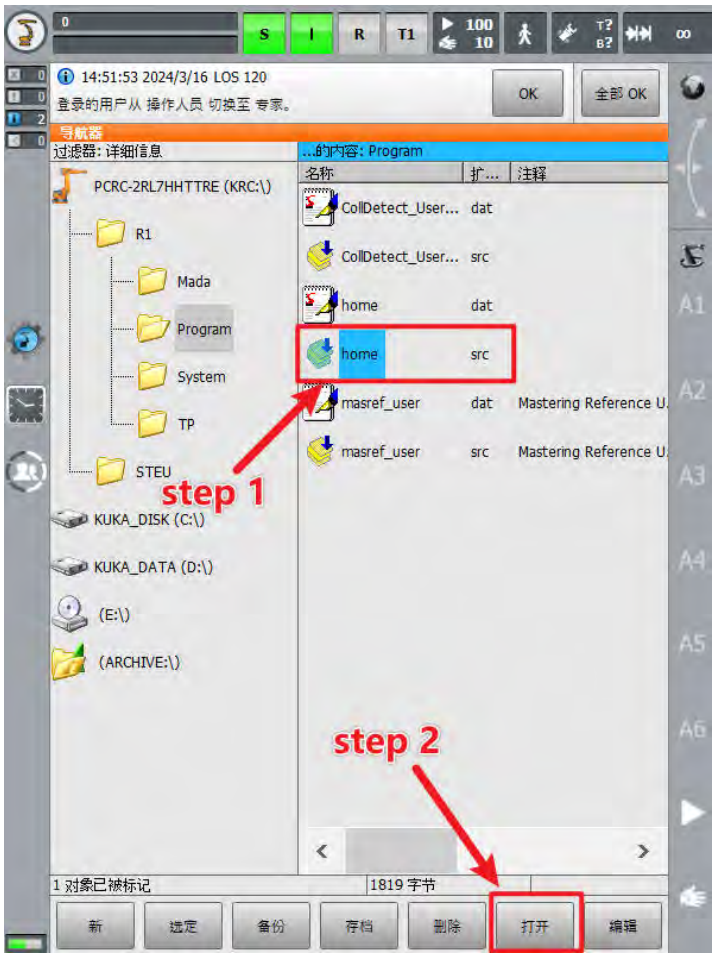
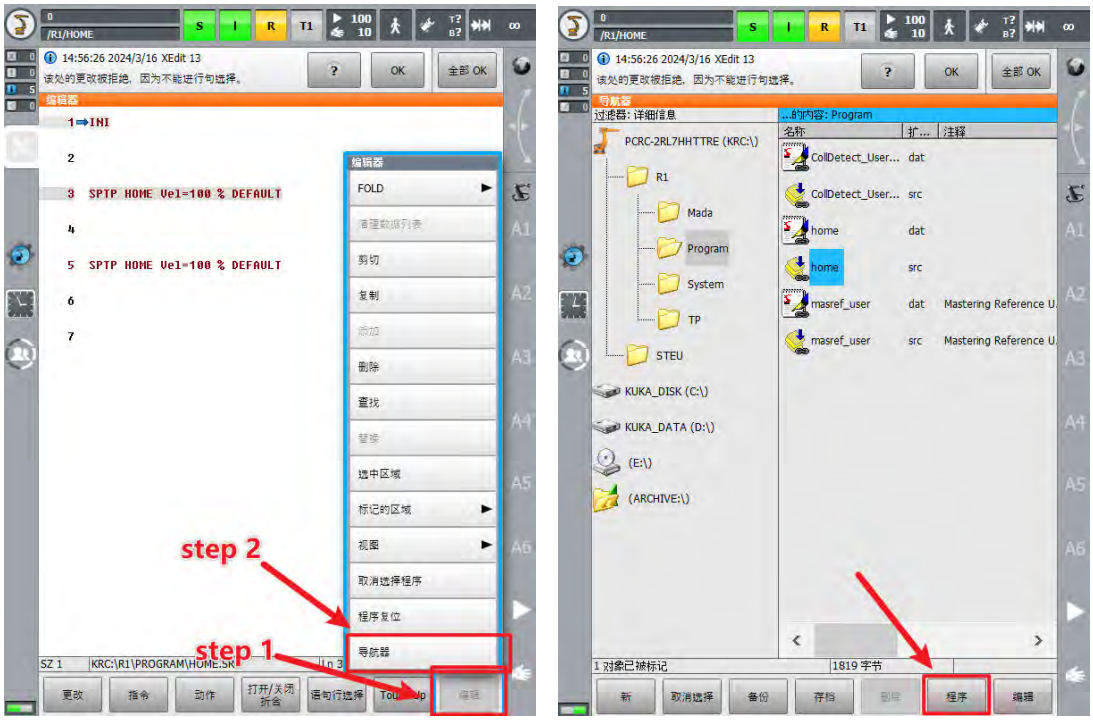


图 6.12: 打开 home 文件

3) 编辑程序。
4) 关闭程序。
5) 为应用更改，点击是回答安全询问。

4. 在导航器和程序之间切换如果已选定或打开了一个程序，则仍可以重新显示导航器，而不必取消选择程序或关闭程序。然后可以重新返回程序。
   1) 程序已选定：
      i. 从程序切换到导航器：选择菜单序列编辑 → 导航器。
      ii. 从导航器切换到程序：点击程序。



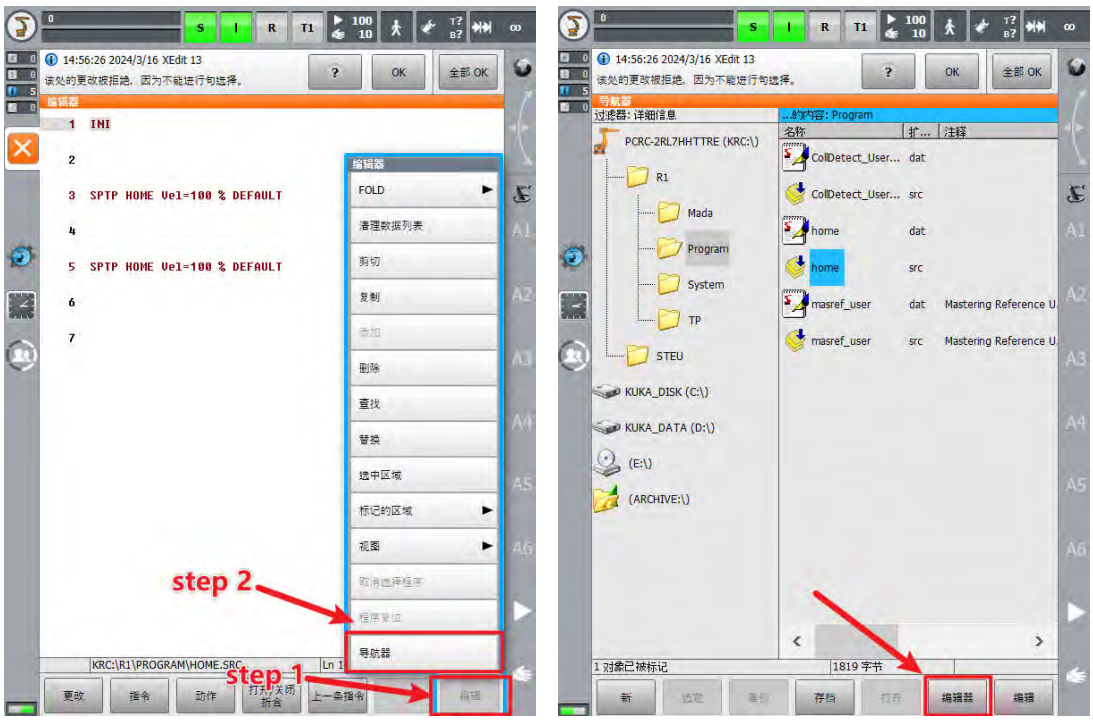
从程序切换到导航器　　从导航器切换到程序

图 6.13: 选定程序时：导航器和程序的切换

2) 程序已打开：

i. 从程序切换到导航器：选择菜单序列编辑 → 导航器。

ii. 从导航器切换到程序：点击编辑器。



从程序切换到导航器　　从导航器切换到程序

图 6.14: 打开程序时：导航器和程序的切换

6.1.6 显示折叠夹

为了程序显示方便，往往需要在折叠夹（Fold）中将隐藏程序段。折叠夹以此使程序更加条理分明。隐藏的程序段在程序运行时有着和正常程序段一样的处理进程。在应用人员用户群中，这些折叠夹始终关闭。即折叠夹的内容不可见，无法编辑。在专家用户群中，这些折叠夹默认关闭。它们可以打开并进行编辑。可以创建新的折叠夹。当一个程序被挑选出或者关闭时，所有折叠夹将自动关闭。

1. 已选定程序，且权限为专家用户组。
2. 选定含有折叠夹的行，点击“打开 / 关闭折合”。折叠自动打开。

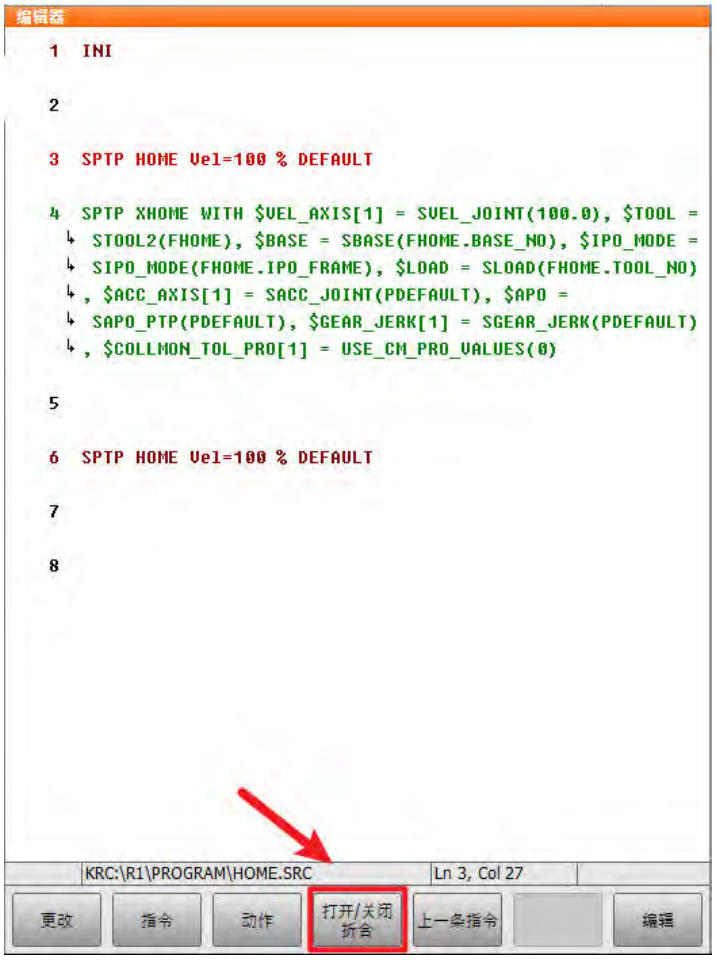


图 **6.15:** 打开或关闭折叠

3. 为了闭合折叠夹，重新按下“打开 / 关闭折合”即可。

6.2 机器人的运动编程

6.2.1 运动编程概述

机器人的运动编程大致分为三种，分别为点到点运动（PTP）、线性运动（LIN）、圆周运动（CIRC），本节主要讲解点到点运动（PTP），另外两个运动，由于篇幅原因，只做简单介绍，后面我们均以 PTP 运动为例进行讲解，其余两者类似，不再细致讲解。

1. 点到点运动（PTP）

   机器人沿最快的轨道将 TCP 引至目标点，如图6.16所示。PTP 移动可以快速抵达目标位置，这是最快也是时间最优化的移动方式。一般情况下最快的轨道并不是最短的轨道，也就是说并非直线。因为机器人轴进行回转运动，所以曲线轨道比直线轨道进行更快。

   由于此轨迹无法精确预知，所以在调试以及运行时，应该在阻挡物体附近降低速度测试机器人的移动特性。如果不进行这项工作，则可能发生对撞并且由此

造成部件、工具或机器人损伤的结果。

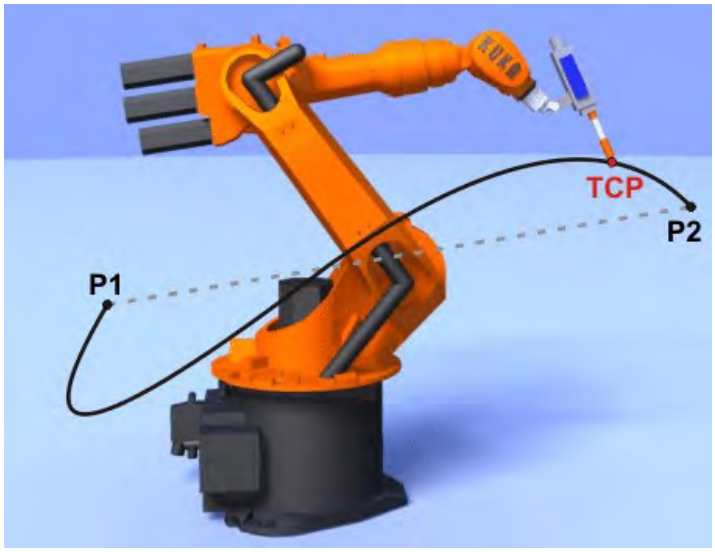


图 6.16: 点到点运动（PTP）

2. 线性运动（LIN）

机器人沿一条直线以定义的速度将 TCP 引至目标点。如图6.17所示。

在做 LIN 移动时，工具尖端从起点到目标点做直线运行。因为两点确定一条直线，所以只要给出目标点就可以了。此时，只有工具的尖端精确地沿着定义的轨迹运行，而工具本身的去向则在移动过程中发生变化，此变化与程序设定的取向有关。

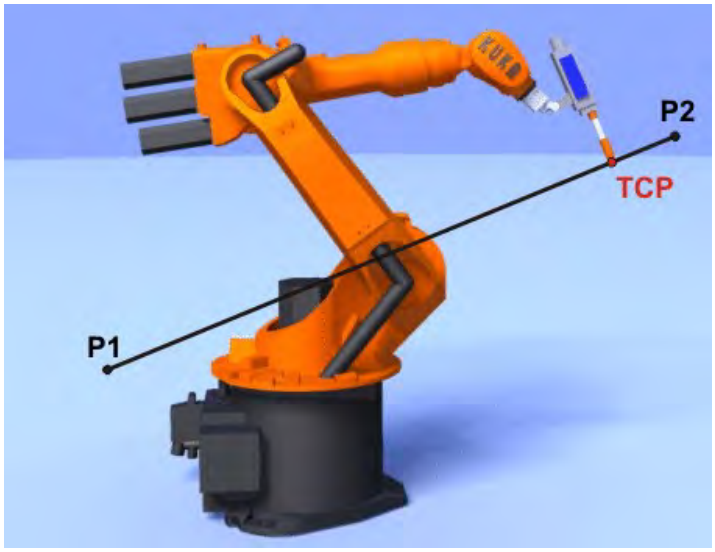


图 6.17: 线性运动（LIN）

3. 圆周运动（CIRC）

机器人沿圆形轨道以定义的速度将 TCP 移动至目标点，如图 3-3-15 所示。

在做 CIRC 移动时，工具尖端从起点到目标点沿着圆周轨迹运动。此圆周轨迹由三个点来确定，这三点为起始点、辅助点和结束点。起始点是上一个指令的精

确定位点。辅助点是指圆周所经历的中间点，对于辅助点来说，只是坐标 X、Y 和 Z 起决定作用。在移动过程中，工具尖端取向的变化顺应于持续的移动轨迹。

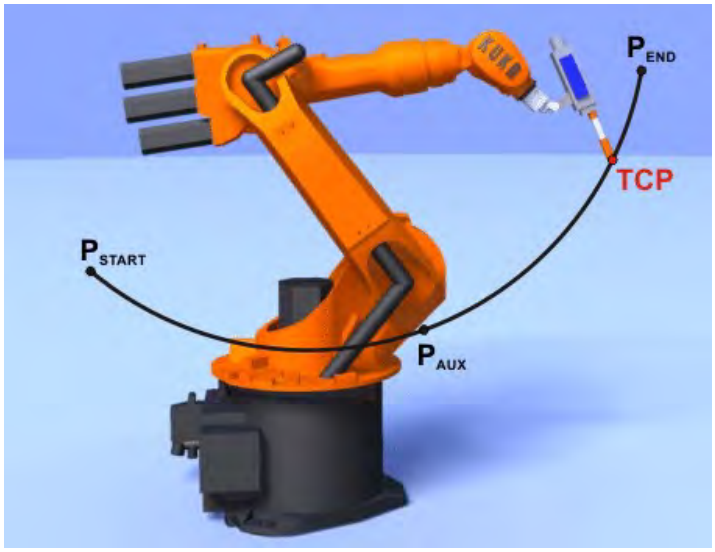


图 6.18: 圆周运动（CIRC）

4. PTP 运动的轨道逼近

轨迹逼近的意思是：将不会精确移至程序编定的点。圆滑过渡是一个选项，可在进行运动编程时选择。当在运动指令之后跟着一个触发预进停止的指令时，无法进行圆滑过渡。

TCP 离开可以准确到达目标点的轨道，采用另一条更快的轨道。进行运动编程时将确定至目标点的距离，TCP 最早允许在此距离处离开其原有轨道。

当发生轨迹逼近的 PTP 运动时，轨迹曲线不可预见。而且，滑过点在轨道的哪一侧经过也无法预测。

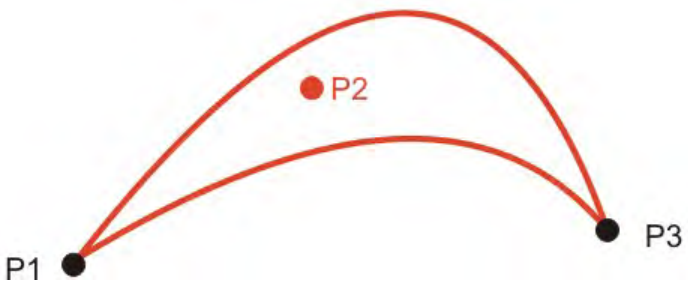


图 6.19: PTP 运动，P2 已滑过

6.2.2 使用联机表格进行运动编程

在 KUKA 中，存在一个以模板为形式的指令，称其为联机表格，联机表格中的每个可选的参数格所弹出的内容，我们称之为“帧选择窗口”，本节我们以运动编程为例，向大家介绍这一种编程方式。

1. 联机表单中的名称在联机表单中可以输入数据组名称。例如点名称、运动数据组名称等。但名称最长为 23 个字符，第一位不允许是数字，且不允许使用除 $ 之外的特殊字符。

   此限制不适用于输出端名称。对于应用程序包中的联机表单可能有另外的限制。

2. 对 PTP 运动进行编程

   (1) 程序已被选定且运行方式为 T1。

   (2) 使用示教器移动机械臂各个关节轴，将 TCP 移向应被设为目标点的位置。

   (3) 将光标置于其后应添加运动指令的那一行中。

   (4) 如图所示，在下方选择菜单中依次选择“命令”→“运动”→“PTP”。

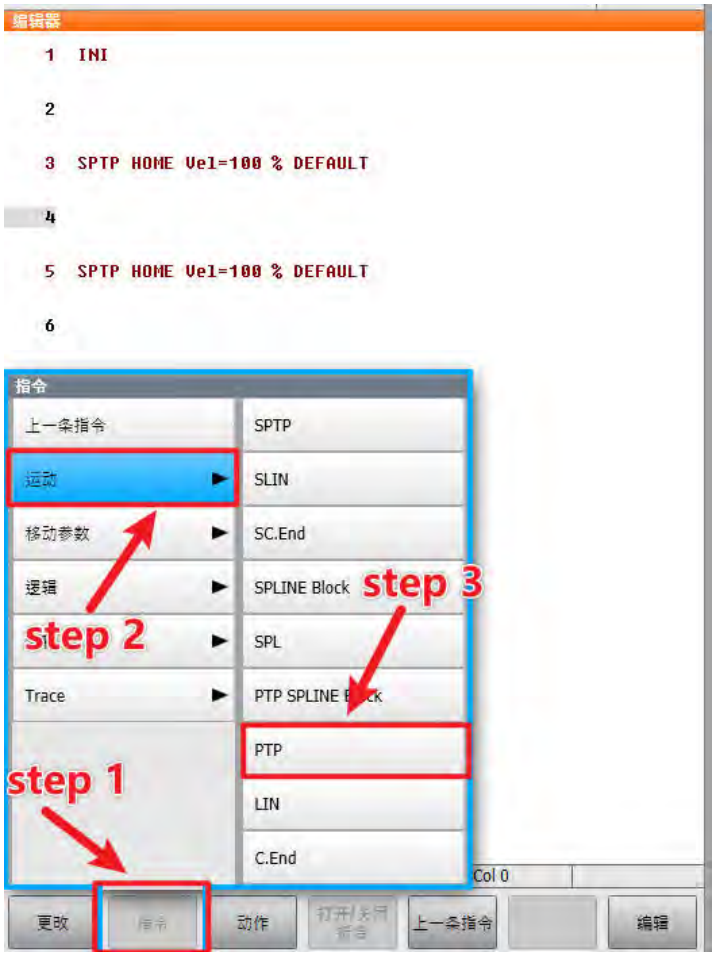


图 **6.20:** a89PTP 运动

   (5) 此时，出现联机表格，在行指令中设置参数，选择右下角“指令 OK”完成。

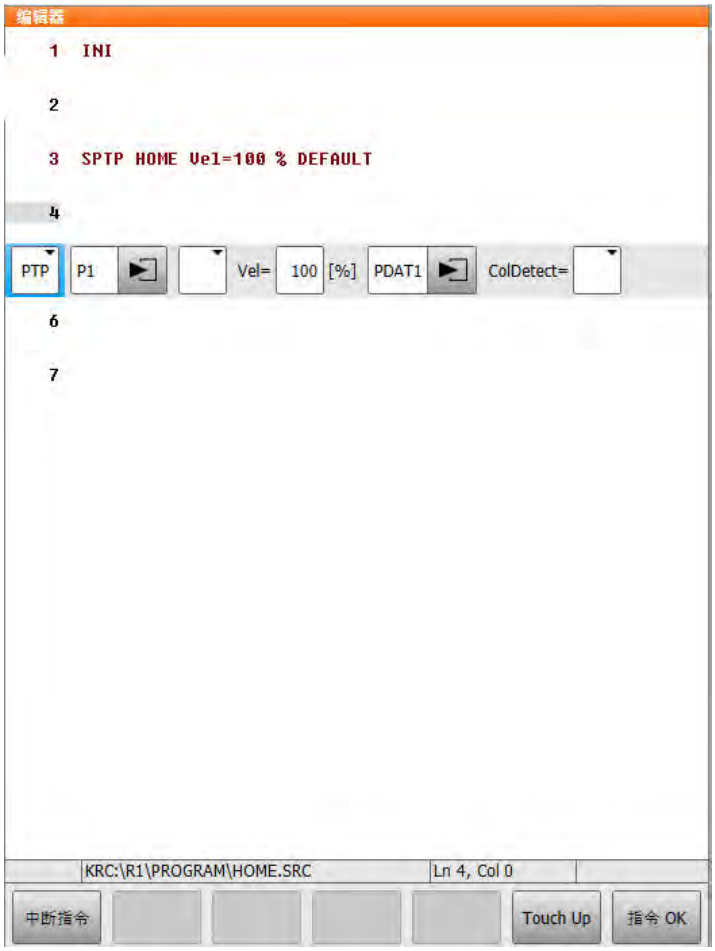


图 6.21: PTP 联机表格

(6) 下面对联机表格各个参数进行讲解。

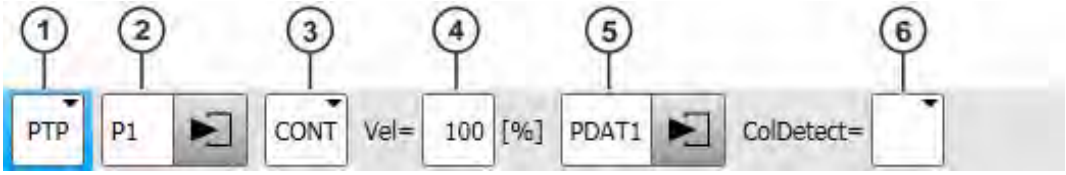


图 6.22: PTP 运动的联机表格

图6.22说明：

①——运动方式 PTP

②——目标点名称：系统自动赋予一个名称。名称可以被改写。需要编辑点数据时请触摸箭头，相关选项窗口即自动打开（“选项窗口坐标系”页面见后续）。

③——CONT: 目标点被轨迹逼近。[空]：将精确地移至目标点。

④——速度：1%-100%。

⑤——运动数据组的名称：系统自动赋予一个名称。名称可以被改写。需要编辑点数据时请触摸箭头。相关选项窗口即自动打开（“选项窗口移动参

数”页面见后续）。

⑥——该运动的碰撞识别：[空]：该运动没有自己的碰撞识别设置。如果接通了通用碰撞识别，则其值适用。如果未接通，则碰撞识别关闭。CDSet_Set[编号]：碰撞识别已开启，为了识别将使用数据组编号中的数值。如果同时激活了通用碰撞识别，则不为该运动考虑其值。

3. 选项窗口坐标系

图 6.23: “选项窗口坐标系”界面

①——选择法兰上的工具或工件。

②——选择基座或固定工具。

③——规定插补模式。若一个工具在法兰上，基坐标为基座上，则不勾选；若一个工件在法兰上，基坐标为固定工具，则勾选。

4. 选项窗口移动参数

图 6.24: “选项窗口移动参数”界面

①——加速度以机器数据中给出的最大值为基准。此最大值与机器人类型和所设定的运行模式有关。

②——只有在行指令中选择了该点应该被轨迹逼近，则此栏目才显示。在目标点之前最早开始执行轨迹逼近的距离此距离最大可为起始点至目标点距离的一半。

如果在此处输入了一个更大数值，则此值将被忽略而采用最大值。

③——仅在 LIN 和 CIRC 运动时才显示该栏，目的为选择姿态。

6.3 使用联机表格进行逻辑编程

KUKA 机器人往往需要接受以及发送数据，故逻辑编程无论在通信还是算法上，都占据举足轻重的地位，这一节主要对常用的逻辑编程进行介绍，其余更多的知识读者可以自行查阅相关文档。

6.3.1 给等待时间编程-WAIT

WAIT 命令，为等待命令，程序会在该处进行堵塞，不会向下执行，与运动编程结合会有出乎意料的效果。当代码运行到运动指令那行后，程序往往不会等待机械手运动到指定位置再向下运行，此时合理地插入 WAIT 命令，赋予代码更多的鲁棒性。

1. 程序已选定，运行方式为 T1。
2. 将光标放到其后应插入逻辑指令的一行上。
3. 选择菜单序列指令 → “逻辑” → “WAIT”。

图 6.25: 等待时间编程

4. 在联机表格中设置参数。

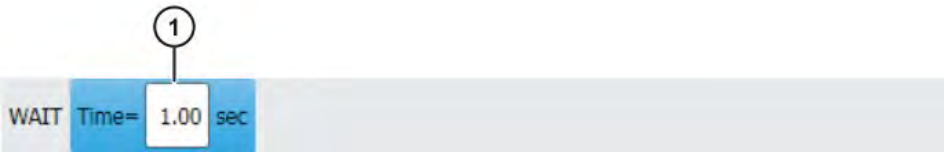


图 6.26: 设置联机表格参数

①——设置等待时间（至少为 0）。

5. 点击右下角“指令 OK”存储指令。

6.3.2 对与信号有关的等待功能进行编程-WAITFOR

WAITFOR 指令为等待某信号时使用，在 KUKA 机器人通信中有重要作用，它可以实时检验输入/输出信号的变化，从而做出想要的结果。当所期待的指令没有发生时，程序会一直在此堵塞，直到所期待命令改变，其一般用来监听外部命令。

1. 程序已选定，运行方式为 T1。
2. 将光标放到其后应插入逻辑指令的一行上。
3. 选择菜单序列指令 → “逻辑” → “WAITFOR”。

图 6.27: 与信号有关的等待编程

4. 在联机表格中设置参数。

指令设定了一个与信号有关的等候功能。需要时可将多个信号（最多 12 个）按

逻辑连接。如果添加了一个逻辑连接，则联机表格中会出现用于附加信号和其它逻辑连接的栏。

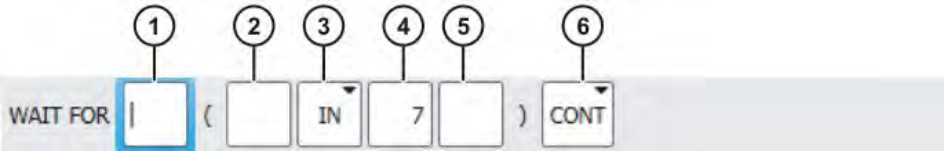


图 6.28: 设置等待信号联机表格参数

①——加外部连接。运算符位于加括号的表达式之间，AND、OR、EXOR。添加 NOT，NOT、[空]，用相应的按键插入所需的运算符。

②——添加内部连接。运算符位于一个加括号的表达式内，AND、OR、EXOR。添加 NOT，NOT、[空]，用相应的按键插入所需的运算符。

③——等待的信号。IN、OUT、CYCFLAG、TIMER、FLAG。

④——信号的编号。

⑤——如果信号已有名称则会显示出来。专家用户组以上，通过点击长文本可输入名称。名称可以自由选择。

⑥——CONT：在预进过程中加工，[空]：带预进停止的加工。

5. 点击右下角“指令 OK”存储指令。

6.3.3 设置数字输出端-OUT

1. 程序已选定，运行方式为 T1。
2. 将光标放到其后应插入逻辑指令的一行上。
3. 选择菜单序列指令 → 逻辑 → OUT → OUT。

图 6.29: 设置输出端

4. 在联机表格中设置参数。

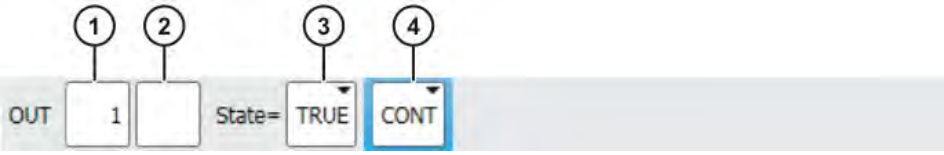


图 6.30: 在联机表格中设置参数

①——输出端编号

②——如果输出端已有名称则会显示出来。从专家用户组起：通过点击长文本可输入名称。名称可以自由选择。

③——输出端被切换成的状态：TRUE/FALSE。

④——CONT：在预进过程中加工，[空]：带预进停止的加工。

5. 点击右下角“指令 OK”存储指令。

6.3.4 更改逻辑指令

有时需要对已经写好的指令进行更改，而不必删除后重新示教编程。

1. 程序已选定且运行方式为 T1。
2. 将光标放在须更改的指令行里。
3. 点击更改。指令相关的联机表格自动打开。
4. 更改该参数。
5. 用指令 OK 保存更改。

6.4 执行程序

不同于其他程序的执行，需要一个 main 函数作为程序执行口，在 KUKA 程序运行中，没有所谓的主程序，所有程序都是相同等级的，用户只需要将需要执行的程序“选定”即可。程序执行有不同的方式，在不同的场景需要切换不同的方式，下面对执行程序进行讲解。

6.4.1 选择程序运动方式

1. 不同程序运行方式说明：

名称	状态显示	说明
Go #GO		程序不停顿地运行，直至程序结尾。
动作 #MSTEP		程序运行过程中在每个点上暂停，包括在辅助点和样条段点上暂停。对每一个点都必须重新按下启动键。程序没有预进就开始运行。
单个步骤 #ISTEP		程序在每一程序行后暂停。在不可见的程序行和空行后也要暂停。对每一个行都必须重新按下启动键。程序没有预进就开始运行。 **单个步骤**仅供专家用户组使用。
返回 #BSTEP		如果按下启动反向键，则会自动选择这种程序运行方式。不得通过其他方式选择。 特性与**动作**时相同，有以下例外情况：CIRC 运动回退，与最后的前行情况相同。也就是说，如果前行时在辅助点没有停止，则回退时在那里也不会停止。 这种例外情况不适用于 SCIRC 运动。在这种运动中，反向运行时始终在辅助点上暂停。

图 6.31: 程序运行方式

2. 选择程序运行方式

(1) 如图所示，在左上角触摸状态显示“程序运行方式”。窗口“程序运行方式”自动打开。

图 6.32: 选择程序运行方式

(2) 选择所需的程序运行方式。窗口关闭并将应用选定的程序运行方式。

6.4.2 语句指针

程序运行时，语句指针可以显示我们所需要的各种信息：机器人正在执行或结束了哪项运动、是否移到一个辅助点或目标点上、机器人执行程序的方向，下图为一些指针符号所对应表达的含义。

指针	方向	说明
	向前	移至目标点。
	逆向	
	向前	已以精确暂停到达目标点。
	逆向	
	向前	移至辅助点。
	逆向	
	向前	已以精确暂停到达辅助点。
	逆向	

图 **6.33:** 语句指针讲解

1. 正向运动的示例

```
5  PTP P3 Vel=100 % PDAT1 Tool[1] Base[0]
6  PTP P4 Vel=100 % PDAT2 Tool[1] Base[0]
7  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
```

图 **6.34:** 机器人从 P3 运动到 P4

```
5  PTP P3 Vel=100 % PDAT1 Tool[1] Base[0]
6  PTP P4 Vel=100 % PDAT2 Tool[1] Base[0]
7  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]
```

图 **6.35:** 机器人已精确暂停达到 P4

2. 反向运动的示例

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]

7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]

8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

图 **6.36:** 机器人从 P8 反向运动到 P7

```
6  PTP P5 Vel=100 % PDAT3 Tool[1] Base[0]

7  CIRC P6 P7 Vel=2 m/s CPDAT1 Tool[1] Base[0]

8  PTP P8 Vel=100 % PDAT16 Tool[1] Base[0]
```

图 **6.37:** 机器人已精确暂停达到 P7

6.4.3 设定程序倍率

程序调节量是程序进程中机器人的速度。程序倍率以百分比形式表示，以已编程的速度为基准。在运行方式 T1 中，最大速度为 250 mm/s。

1. 在示教器中，点击所示图标，即可打开程序倍率调节。

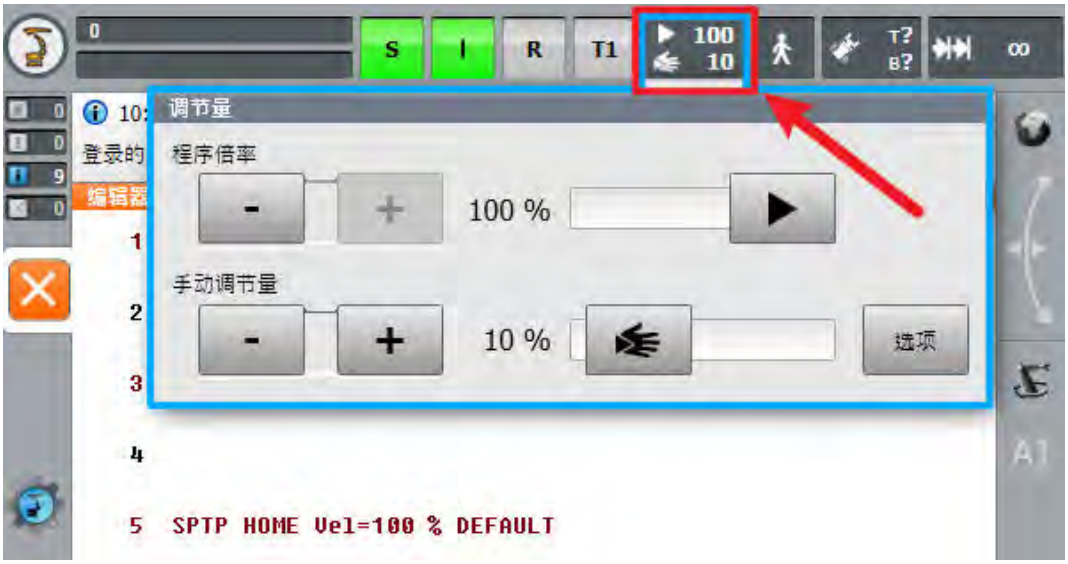


图 **6.38:** 设置程序倍率

2. 设定所希望的程序倍率。可通过正负键或通过调节器进行设定。正负键只能以 100%、75%、50%、30%、10%、3%、1% 步距为单位进行设定。调节器：倍率可以以 1 % 步距为单位进行更改。

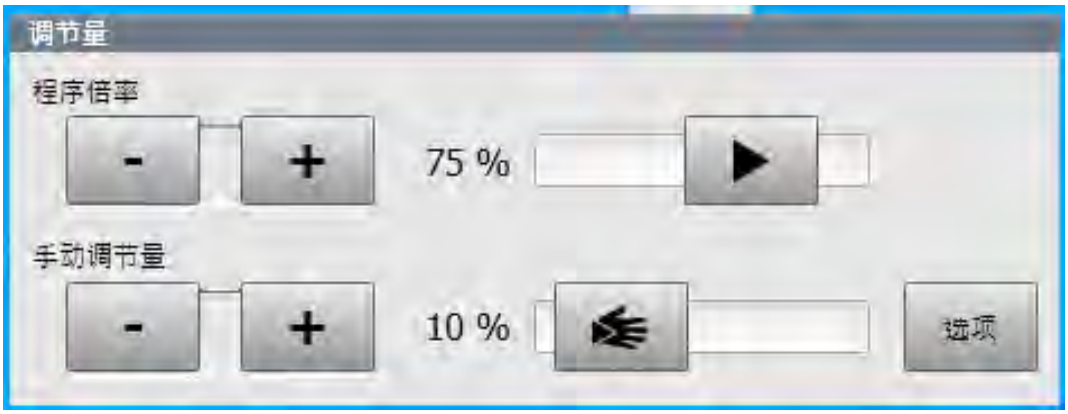


图 6.39: 速率调节窗口

3. 重新触摸状态显示，窗口关闭并应用所需的倍率。在窗口倍率中可通过选项打开窗口手动移动选项。也可以使用 smartPAD 右侧的正负按键来设定倍率。可以以 100%、75%、50%、30%、10%、3%、1% 步距为单位进行设定。

6.4.4 机器人解释器状态显示

如图6.40所示，该图标即表示机器人解释器状态，下面对指示灯各个状态进行解释。

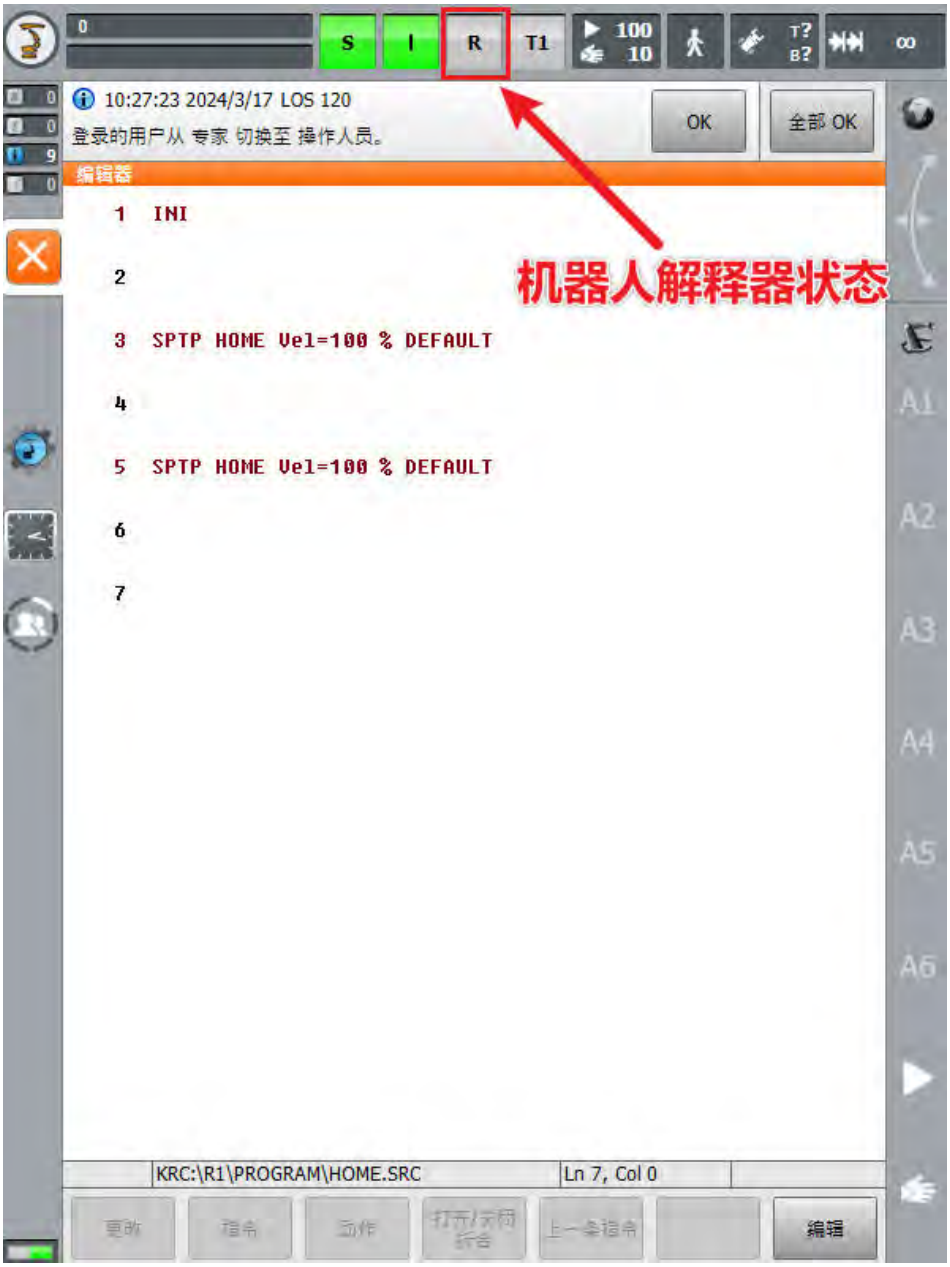


图 **6.40:** 机器人解释器状态

符号	颜色	说明
	灰色	未选定程序。
	黄色	语句指针位于所选程序的首行。
	绿色	已经选择程序，而且程序正在运行。
	红色	选定并启动的程序被暂停。
	黑色	语句指针位于所选程序最后。

图 6.41: 机器人指示灯状态说明

6.4.5 启动正向运动程序（手动）

1. 启动前程序需要已被选定。
2. 选择程序运行方式。
3. 按住确认开关，直至状态栏显示“驱动器已准备就绪”：

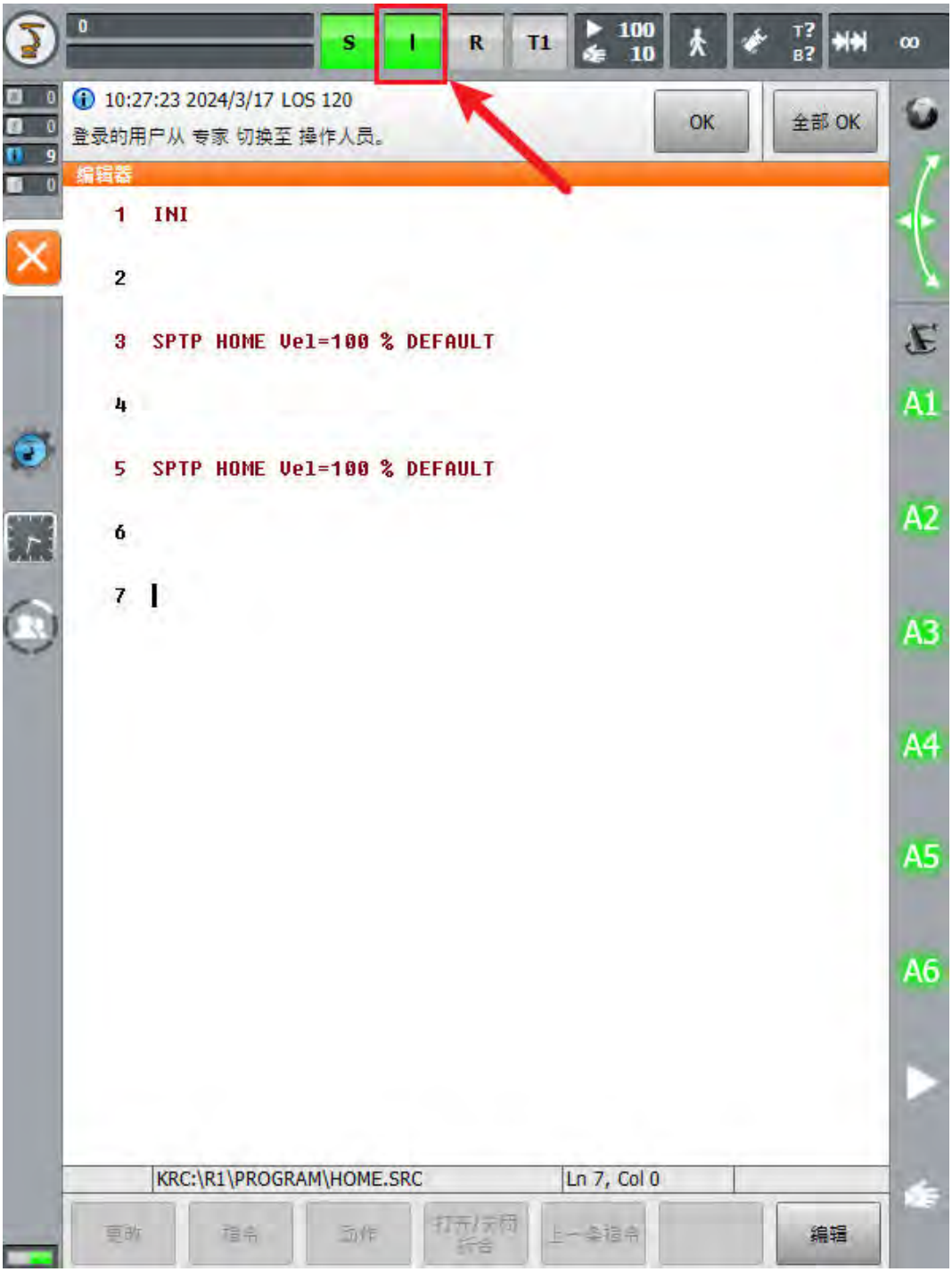


图 **6.42:** 驱动器已准备就绪

4. 执行 SAK 运动：按住启动按键，直至提示信息窗显示“已到达 BCO ”。机器人停下。BCO 运行必须作为 LIN 或 PTP 运动从实际位置移动到目标位置，速度已自动下降。无法确保预见运动过程。在 BCO 运行时观察运动，以便有发生碰撞的危险时机器人可及时停止。
5. 按下启动键并按住。程序开始运行，根据程序运行方式带暂停或不带暂停。如果要停止一个手动启动的程序，可松开启动键。

6.4.6 启动正向运行程序（自动）

1. 启动前程序需要已被选定，扭动钥匙旋钮，切换为 AUT 模式（如图所示）。

图 6.43: 状态栏显示程序为 AUT 模式

2. 选择程序运行方式 Go。
3. 按住启动键，程序即可自动运行，启动 AUT 模式前，最好先使用手动模式进行调试，以确保安全。
4. 为了停止在自动运行中启动的程序，按下停止键。

6.4.7 语句选择

使用语句选择可使一个程序直接略过中间行，跳转到指定行运行，中间略过的行均不会执行，此时程序的任何变量不会发生改变，仍然为选择之前的数值。

1. 选定程序，运行方式调整为 T1 或 T2。
2. 选择程序运行方式。
3. 将鼠标光标点在应启动程序的运动语句。

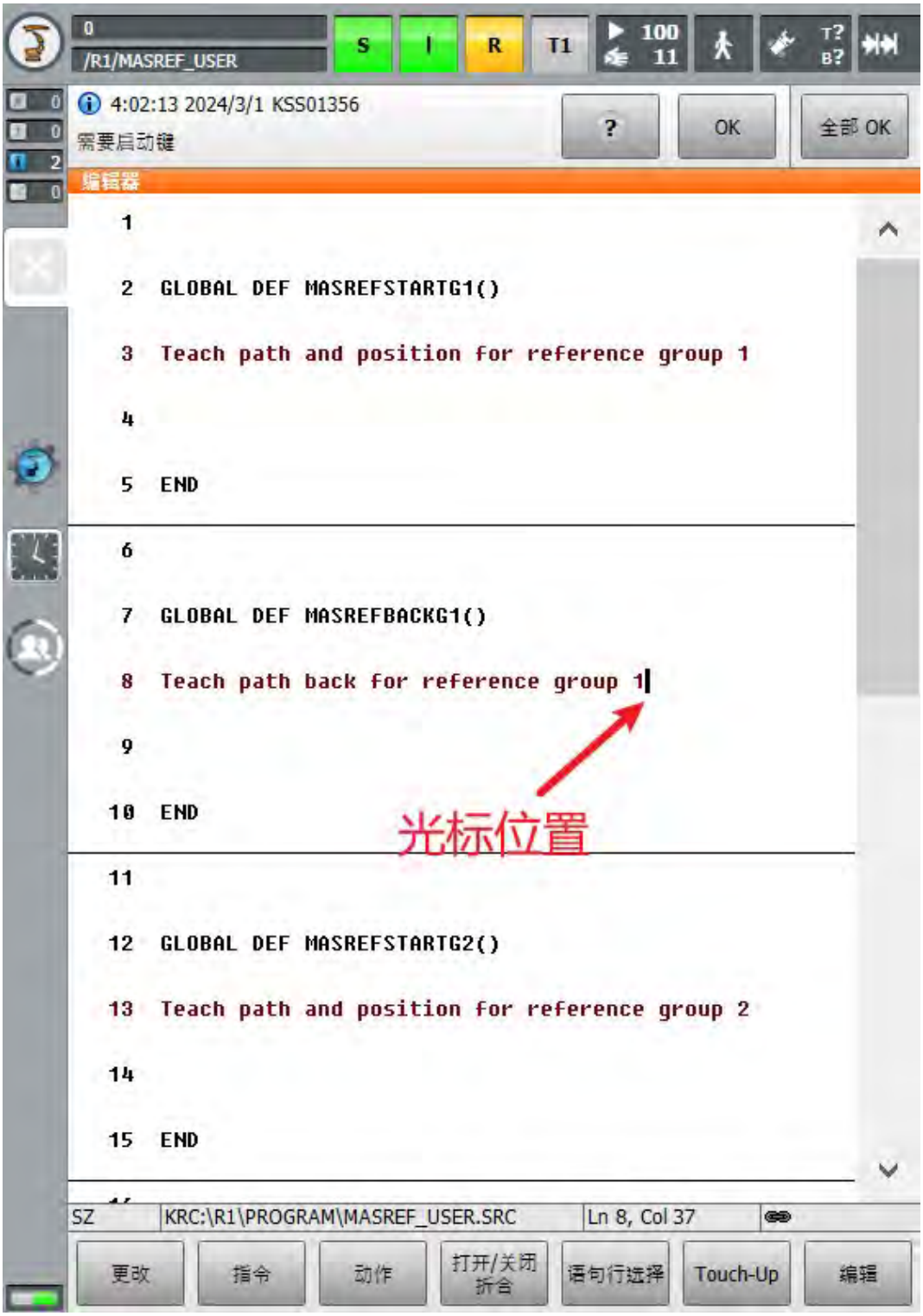


图 **6.44:** 将光标移至所选行

4. 点击下方的“语句行选择”指令按钮，语句指针指在动作语句上，此时该行语句左侧出现向前指针。
5. 按住确认开关，点击启动键启动程序即可。

6.4.8 复位程序

如果要从头重新开始一个中断的程序，则必须将其复位。这样可使程序回到起始状态。

选定程序后，有两种方法可以复位程序。在页面下方菜单列选择“编辑”→“程序复位”，或者在状态栏中触摸状态显示机器人解释器。一个窗口自动打开。选择程序复位。

程序复位方法一　　程序复位方法二

图 6.45: 程序复位方法

第 7 章 视觉机器人积木抓取实操

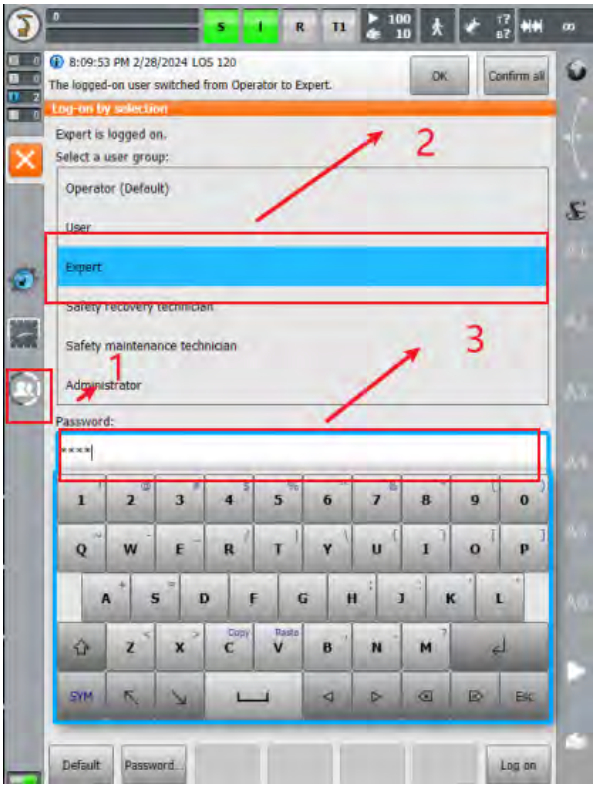
7.1 手眼标定实操

7.1.1 KUKA 示教器切换中文

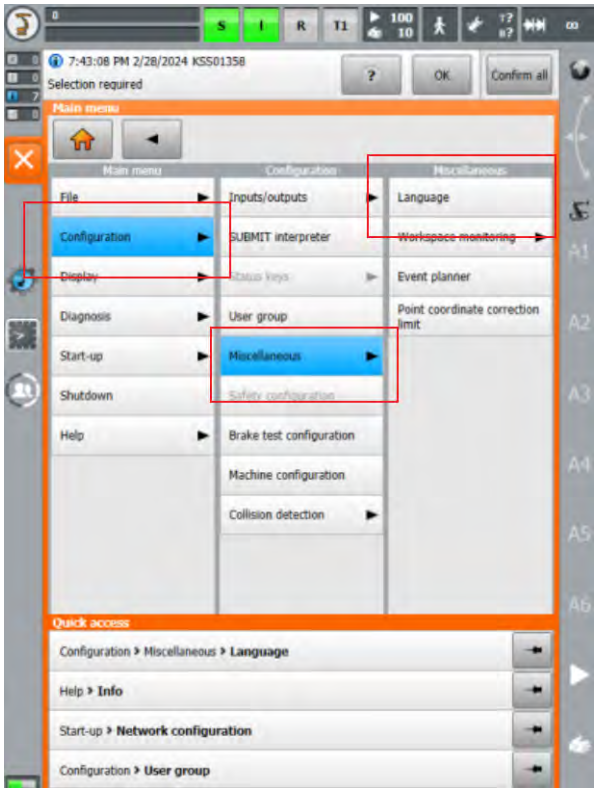
初始的示教器页面是英文显示的，为了叙述方便现将其转为中文。

首先要将用户组切换为专家及以上的等级。

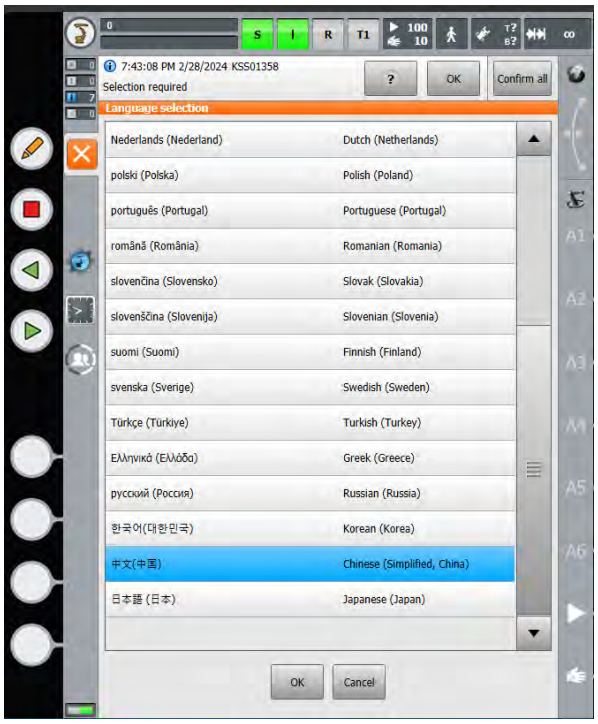
权限密码为 kuka



点击示教器主界面图标，选择 Configuration->Miscellaneous->Language



下拉找到中文，然后点击 OK，语言界面就切换完成了。



7.1.2 获取标定板坐标系

在现实生活想要知道物体的具体位置，前提必须要知道参照物的位置和物体与参照物之间的相对位置关系。

在我们的 eye-in-hand 视觉机械臂模型里也是一样，已知的条件是法兰坐标系与机械臂 Base 坐标系的相对关系，即旋转平移矩阵，当固定好相机之后，相机坐标系相对于法兰坐标系的位置是未知的，因此相机坐标系相对于机械臂 Base 坐标系的位置也是未知的。

由于法兰坐标系相对于机械臂 Base 坐标系的位置关系可以直接从示教器里读取出来，手眼标定的目的就是要获取相机坐标系相对于机械臂 Base 坐标系的绝对位置(此位置要求能够实时获取)。

我们知道对于普通的相机标定模型，只能计算得到相机之间的相对位置关系 (计算时间较长)，而手眼标定之后，每拍一张照片，其拍摄的相机位置都能快速获取到，相当于有一把刻度尺能实时测量相机坐标系的位置，大大简化了之后视觉处理定位的流程。

手眼标定的第一个步骤就是要获取标定板的坐标系，只需测量标定板上不在同一直线三个点在机械臂 Base 坐标系上的点即可。我们规定测量点 X，Y，原点如下图所示。

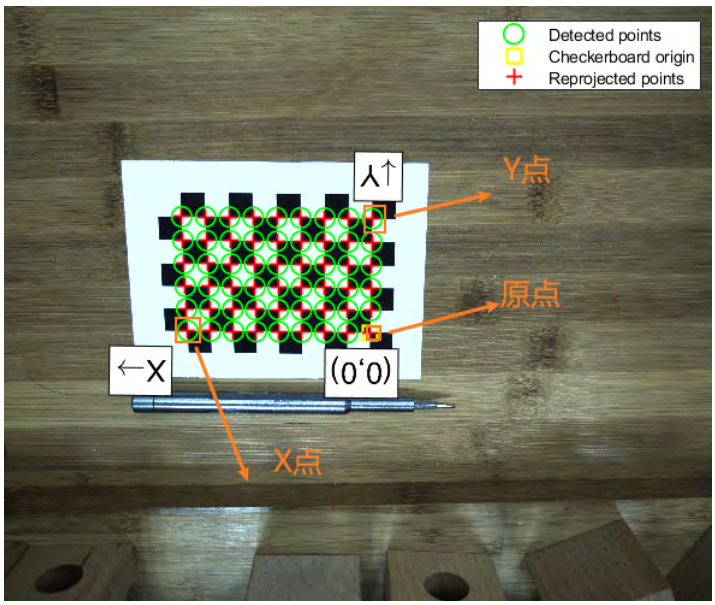


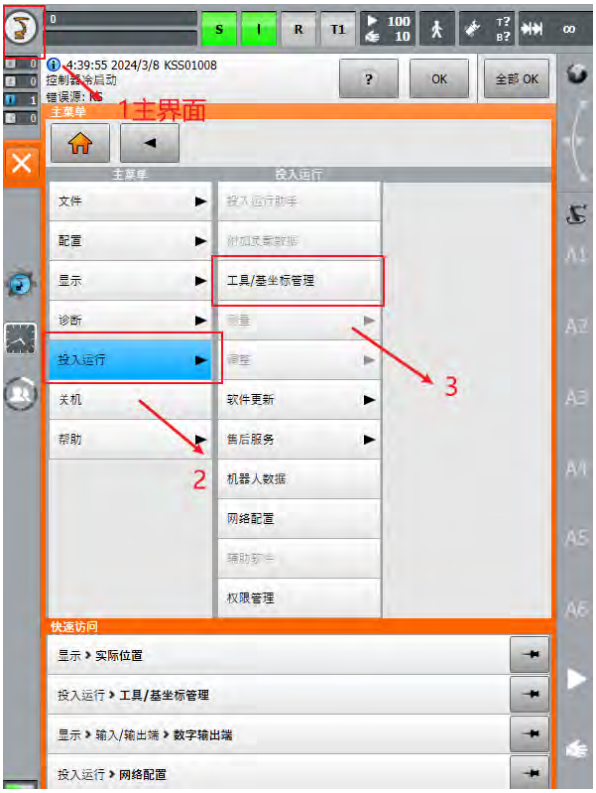
图 7.1: 标定板测量点

要测量这三个点在机械臂 Base 坐标系下的坐标，我们就必须要依托尖的合金头手动示教去触碰这三个点，然后在机械臂上读取到点的坐标。下面是具体操作步骤：

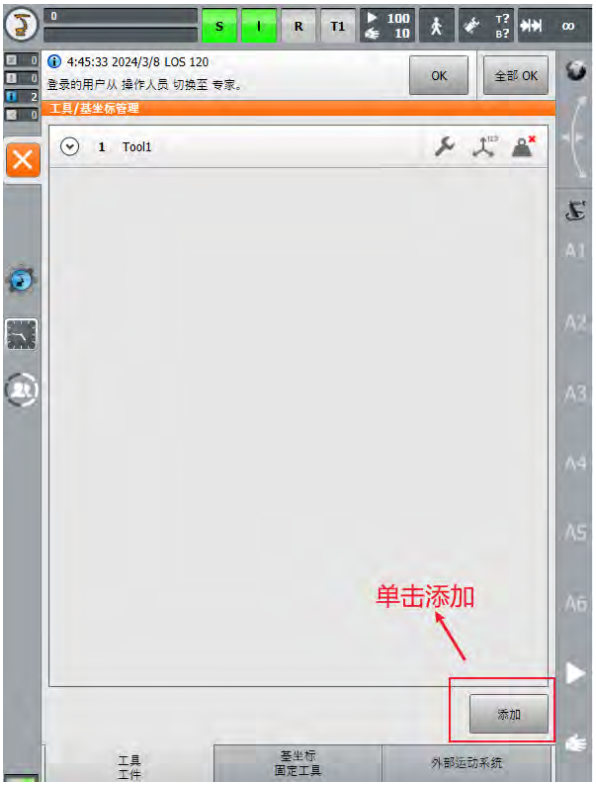
1. 首先将合金工具头装在机械臂末端法兰盘上，此时我们要获取合计头末端的坐标，就需要四点标定法来标定合金头工具末端坐标。

图 7.2: 机械臂安装合金工具头

依次点击主界面-> 投入运行-> 工具/基坐标管理



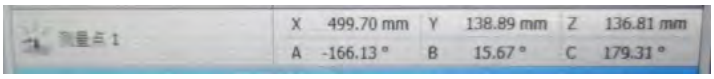
点击添加



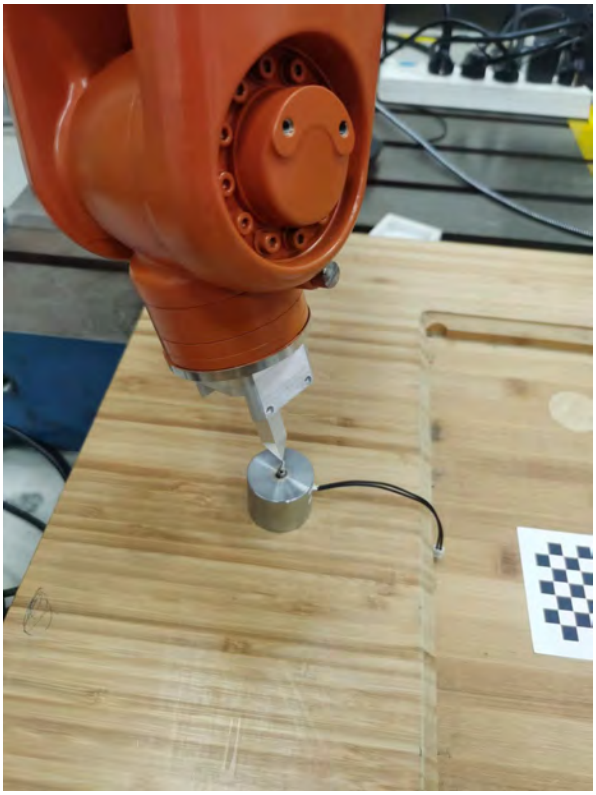
点击测量，选择 XYZ4 点测量法

示教器控制机械臂使合金头与目标物体触碰，每次触碰的点位要相同

单击 Touch-Up 记录坐标



变换机械臂的姿态去触碰目标物体



重复以上步骤获取到 4 个坐标，点击保存即可获取到工具坐标。

我们在此处获得的工具坐标是 [Tool1]

图 7.3: 获取到的工具坐标

2. 用刚刚获取到的工具坐标系来点击标定板的上的黑白格子角点，获取到标定板坐标系的 X，Y，原点在示教器上的坐标。

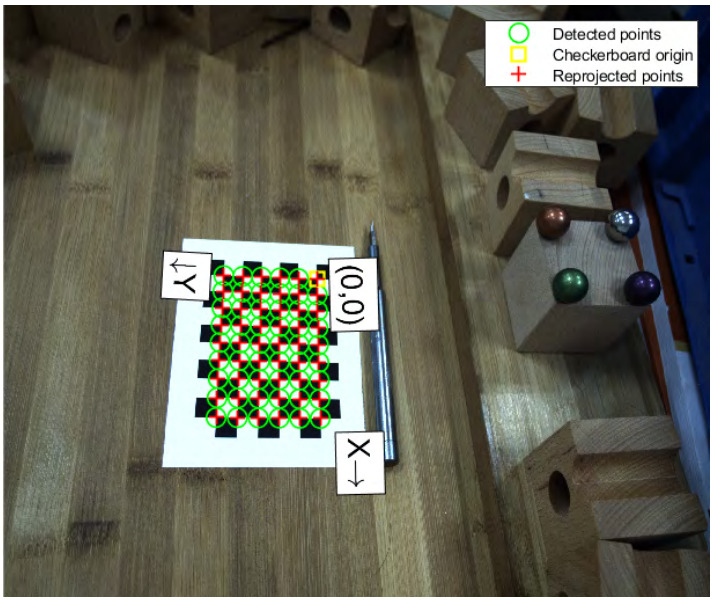


图 7.4: 标定板坐标系

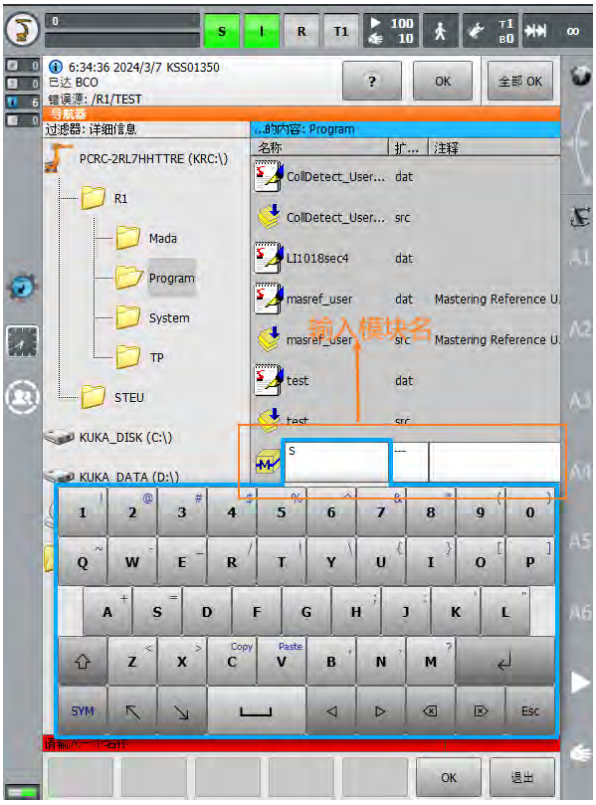
示教器坐标读取方式：

我们要正确读取到换上合金头末端的点坐标，首先要更改工具坐标为标定得到的 [TOOL1]。下面是操作步骤：

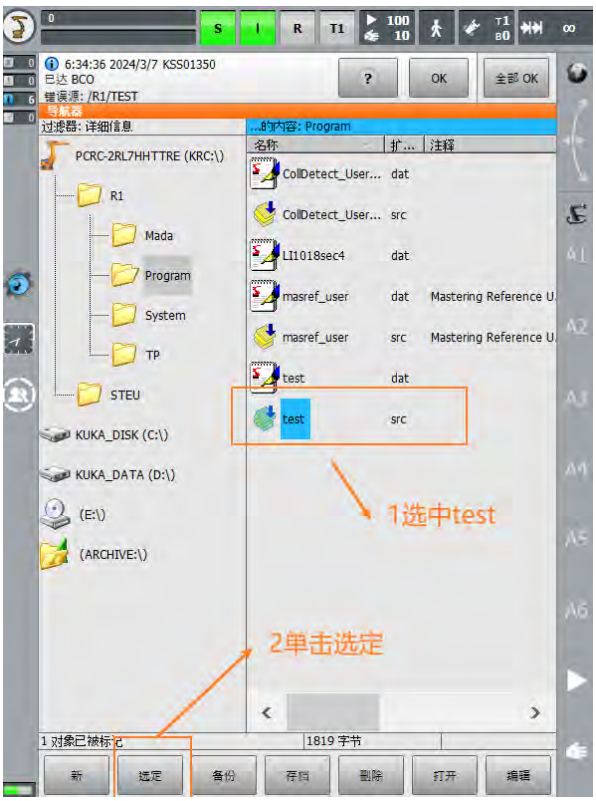
点击示教器左下方的 [新] 选项，

选择 Modul 模块，

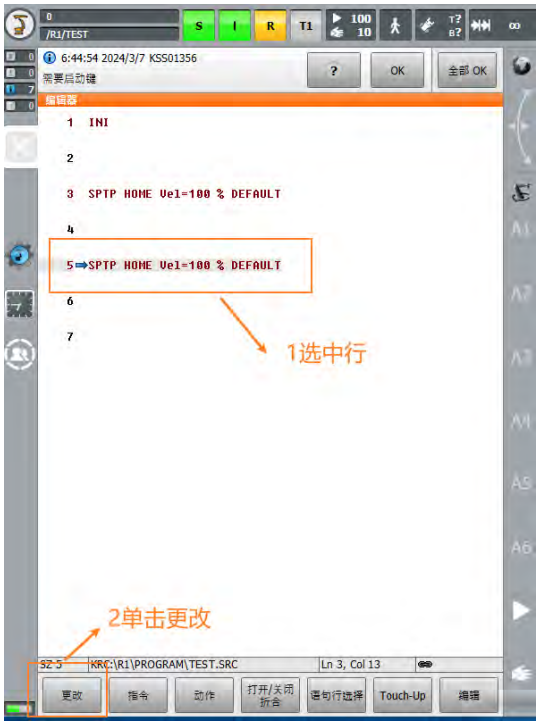
点击 OK，并输入模块名



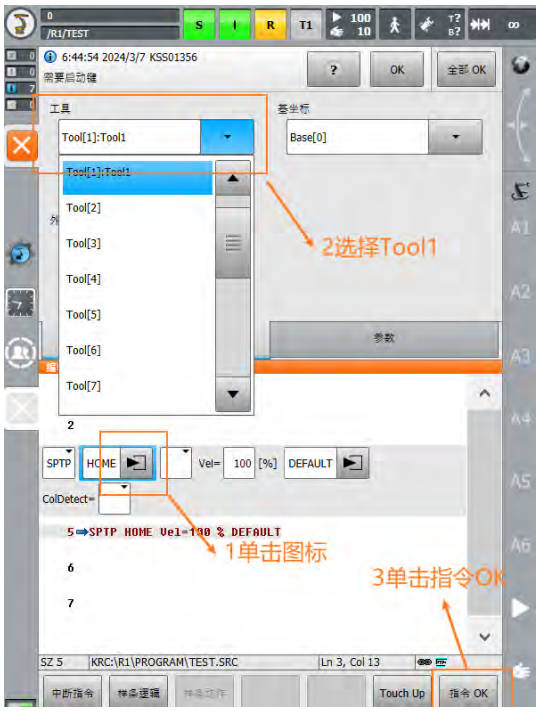
我们在这里设置为 test，选中 test，再点击示教器左下方的选定按钮，选中程序



选中默认的 HOME 行，点击示教器下方的更改按钮

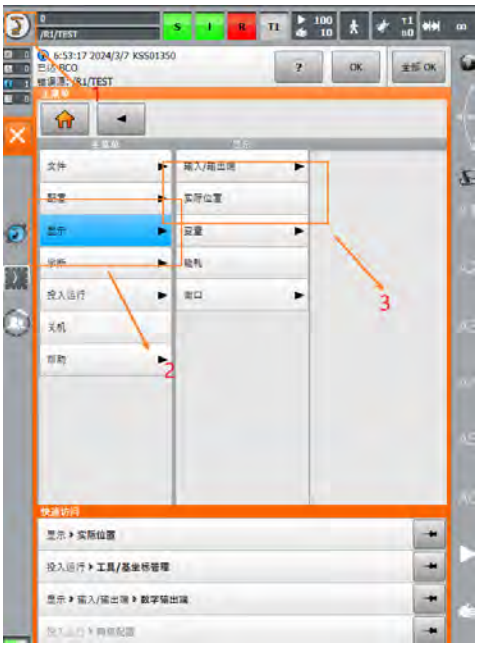


选中 HOME 选项，然后修改工具坐标为刚刚标定好的工具坐标 [Tool1]，基坐标保持不变



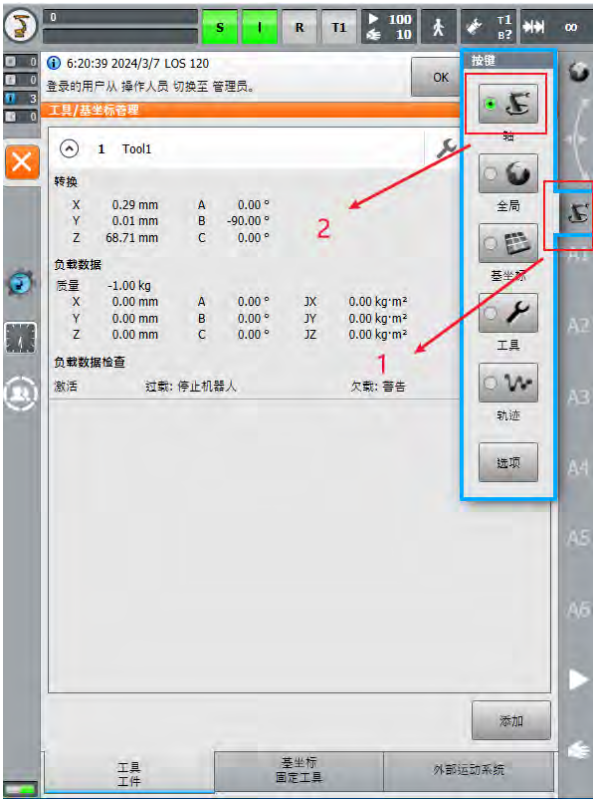
然后点击指令 OK，按住使能键和运行键，当代码运行到 HOME 位置时示教器会弹出“已达 BCO”的提示

此时打开示教器主界面，选择显示-> 实际位置



此时可以看到工具坐标和基坐标都已经初始化好了，

点击蓝色框处，选择运动方式为轴，然后按动 A1,A2,A3,A4,A5,A6 对应的示教器按键，机械臂会根据各个轴来进行旋转移动。并且示教器上的位置坐标读数也会随之改变。



控制机械臂将合金工具的尖头移动至标定板上的 X，Y，原点处，在示教器上读取到这三个点的读数（单位是 mm）

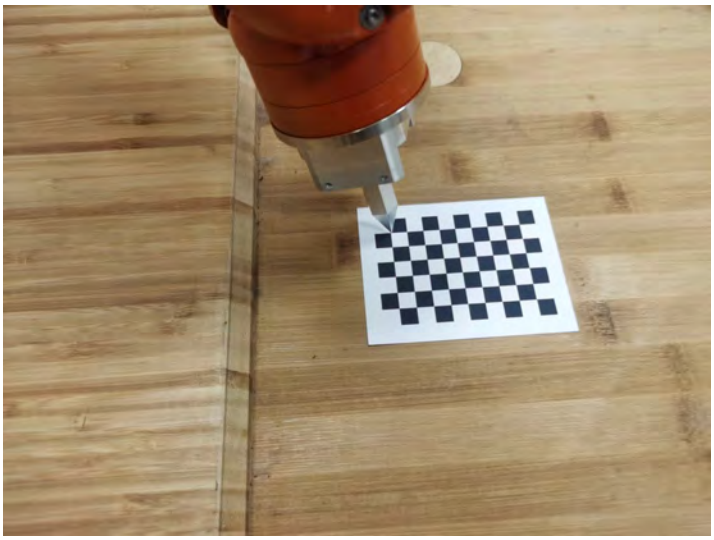


图 7.5: 读取标定板三点坐标

3. 将获取到的三个坐标输入到 Matlab 代码 calibrate.m 文件中的 base_p1,base_p2,base_p3 变量

```
base_p1 = [626.43, 34.49, 9.73];%原点
base_p2 = [576.01, 36.44, 9.92];%y
base_p3 = [623.67, -45.23, 9.67];%x
```

由此有了 3 个点，根据右手法则叉乘得到法向量，就能建立标定板坐标系。

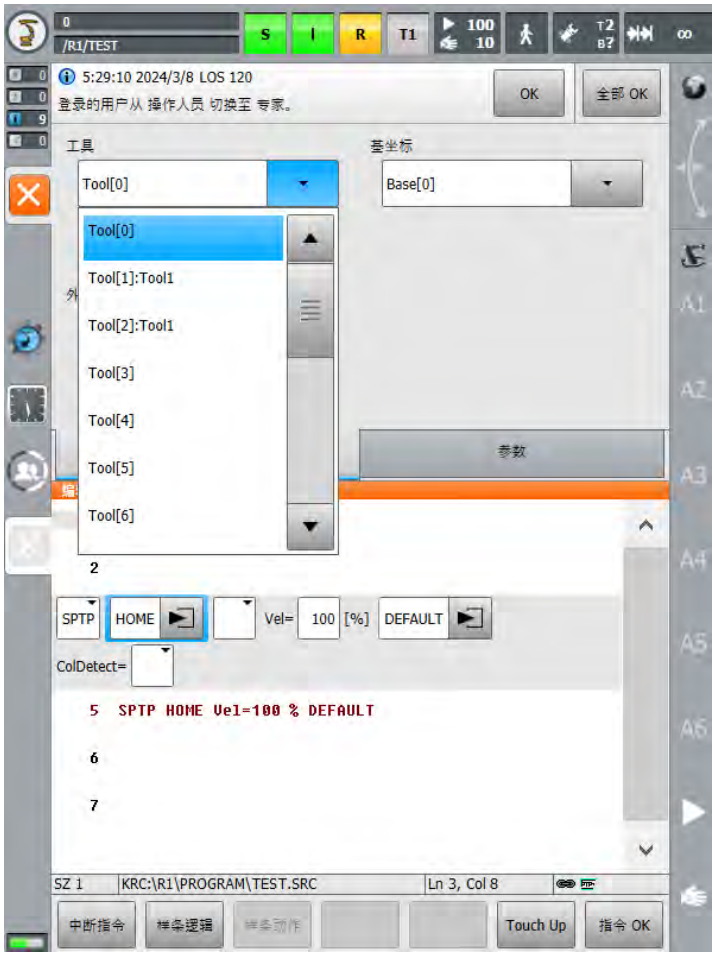
```
base_p1 = [626.43, 34.49, 9.73];%原点
base_p2 = [576.01, 36.44, 9.92];%y
base_p3 = [623.67, -45.23, 9.67];%x
board_axis_x=(base_p3-base_p1)/norm(base_p3-base_p1);
board_axis_y=(base_p2-base_p1)/norm(base_p2-base_p1);
board_axis_z=cross(board_axis_x,board_axis_y)/norm(cross(board_axis_x,board_axis_y));
board_axis_y=board_axis_y-board_axis_x*board_axis_y'*board_axis_x;
board_axis_z=board_axis_z-board_axis_x*board_axis_z'*board_axis_x-board_axis_y*board_axis_z'*board_axis_y;
```

7.1.3 获取照片

接下来就要获取标定板照片。

要将合金工具头重新取下来，在法兰上装上相机，

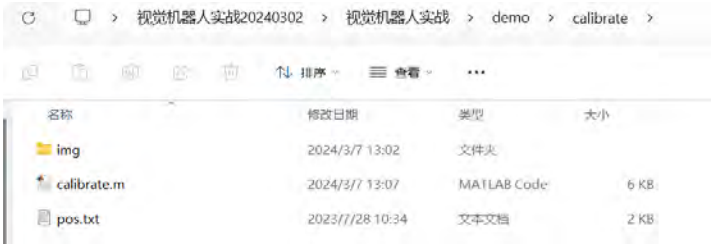
调整相机的光圈与焦距，控制机械臂在各个角度拍摄标定板照片（获取清晰的照片）。同时记录每张照片对应的法兰位置。(由于此时法兰上已经装上相机，我们不能继续使用标定好的 Tool1 工具坐标系，要将其更换为默认的法兰坐标系，只要参照上述修改坐标系的流程，将工具坐标改为没有进行任何测量的 Tool0 即可)



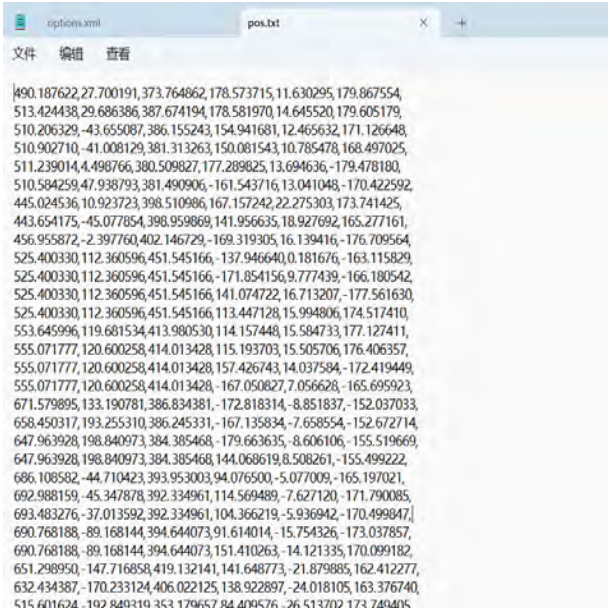
(拍照片的方式参照第 7 章 KUKA 机器人的视觉积木抓取实战中 VisionMaster 功

能模块中的输出图像)

示例中的标定板照片与照片法兰位置都已存放至 demo/cailbrate 文件夹下的 img 和 pos.txt 文件中.



标定板图片

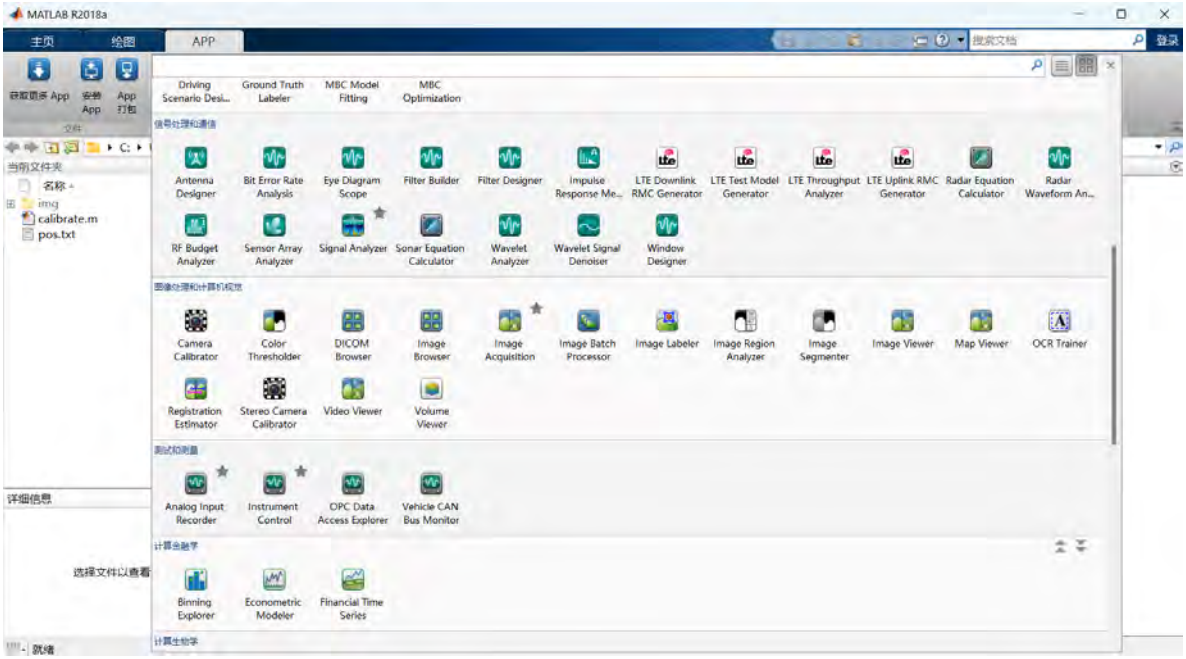


对应位置

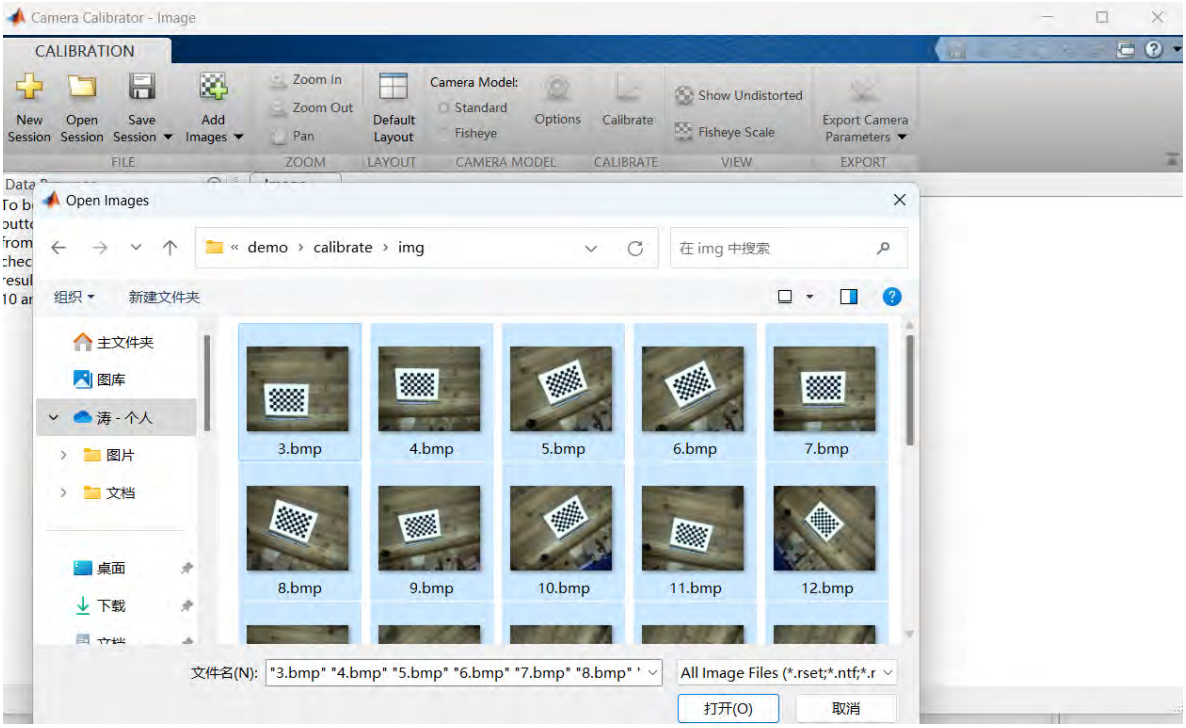
(每张照片一次对应 pos.txt 中的每行位置)

7.1.4 matlab 工具箱相机标定

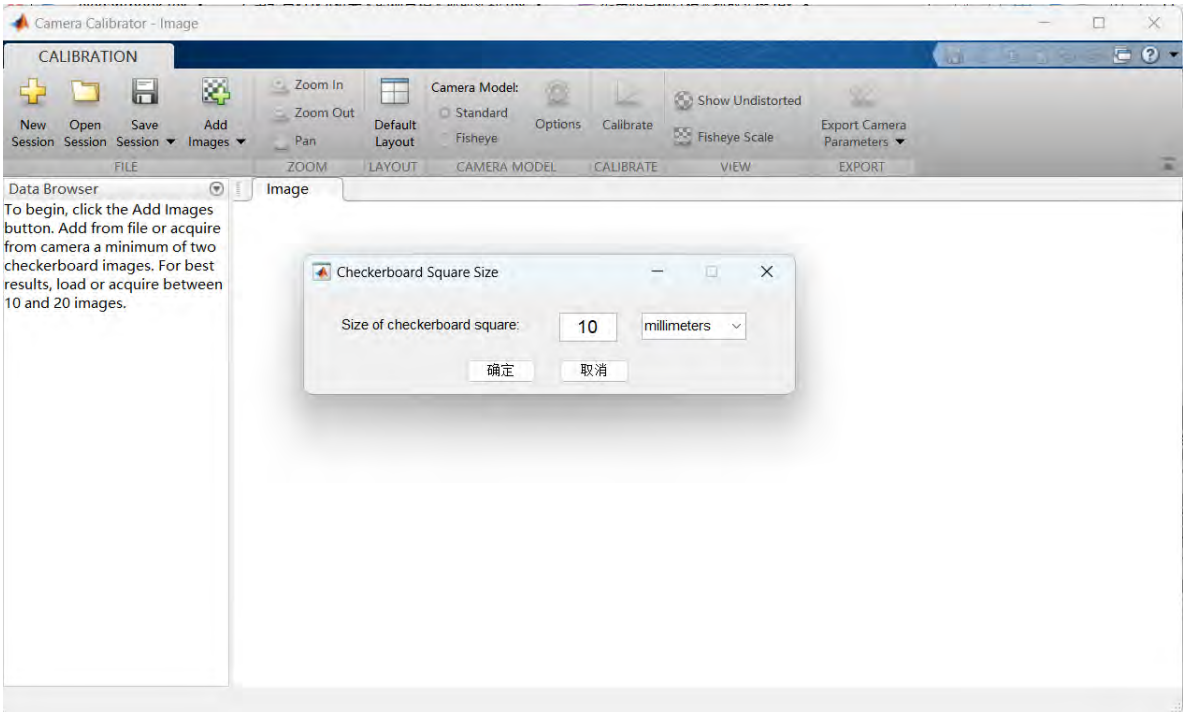
在 Matlab 中点击 APP 图标、选择图像处理和计算机视觉中的 Camera Calibrator 模块进行相机标定



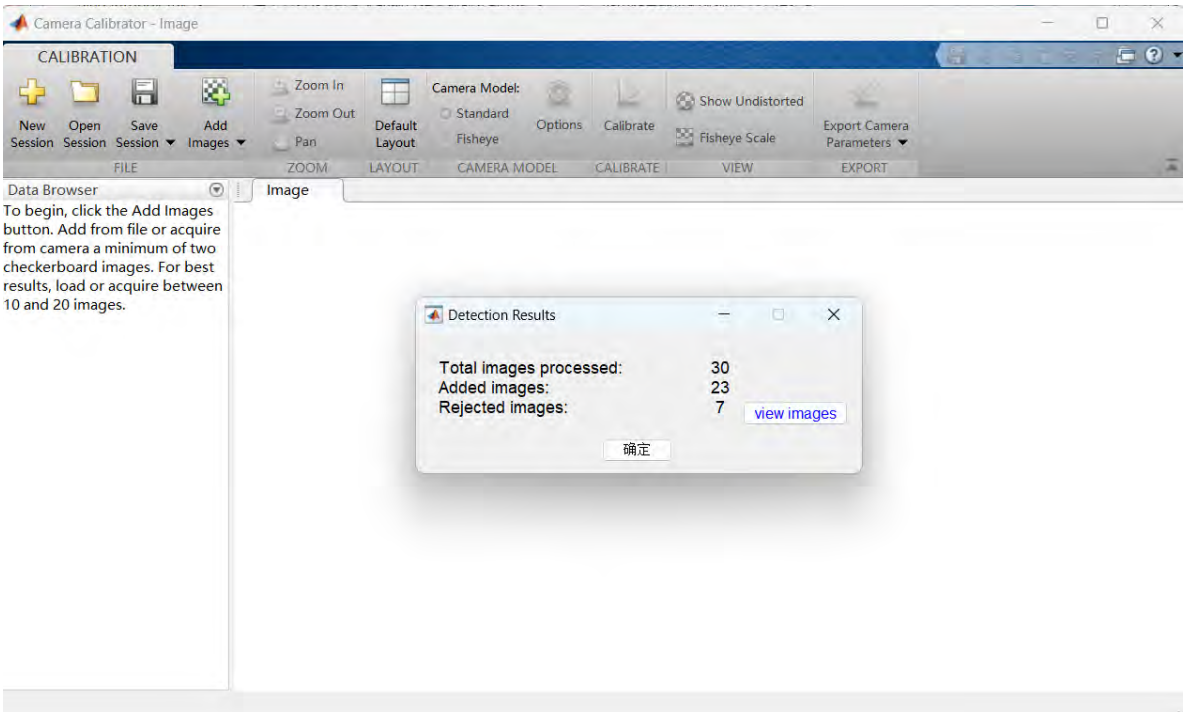
在弹出来的界面中选择 Add Images 添加拍摄好的相机图片，在 demo/calibrate/img 路径下全选照片，单击打开



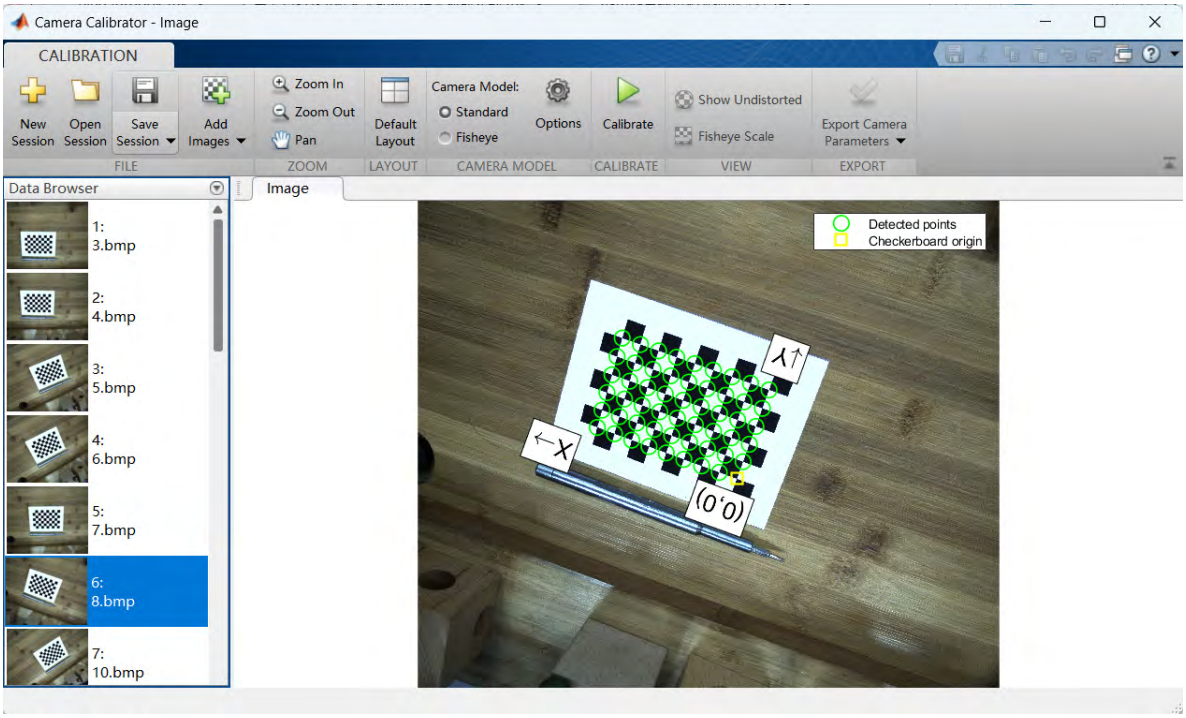
在弹出来的窗口中选择棋盘格的格子大小，我们在此选择 10 milimeters，单击确定



可以看到总共 30 张示例图片，成识别到 23 张图片，有 7 张未能正确计算



在识别成功的图片中可以查看到照片名，以及 Matlab 相机标定得到的标定板的角点，

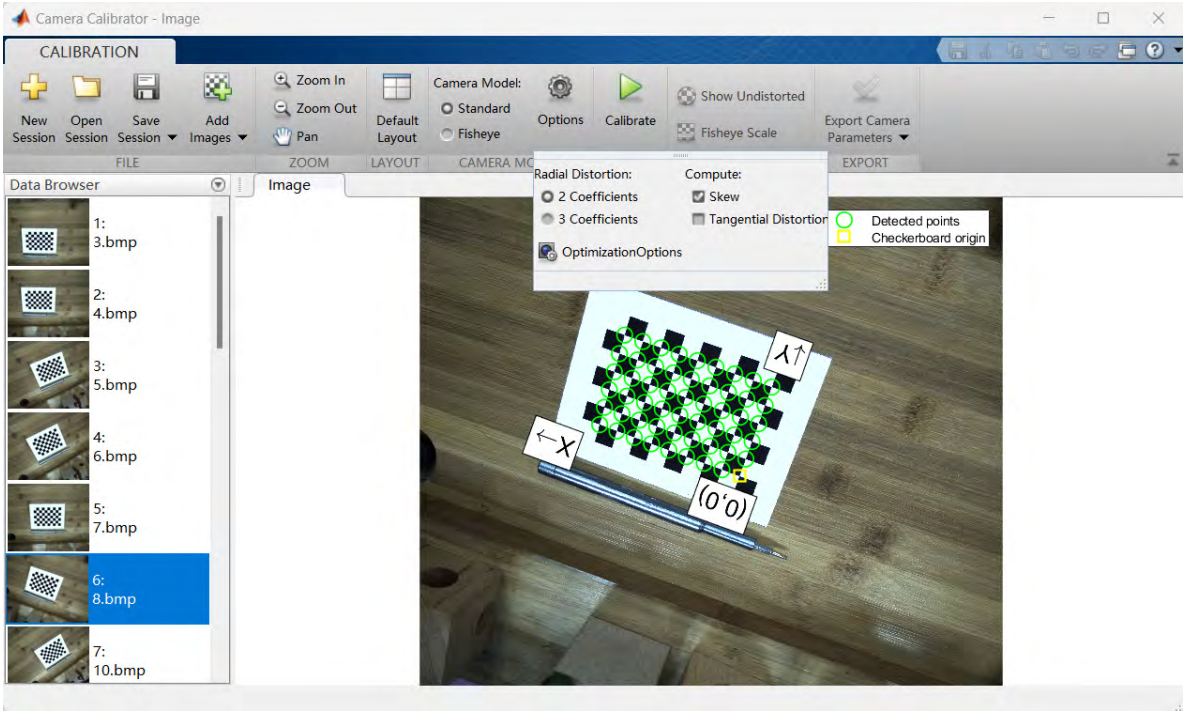


然后将左侧识别到的标定图片对应的相机位置输入到 Matlab 代码 calibrate.m 文件中的 flange_pos_list 变量之中。

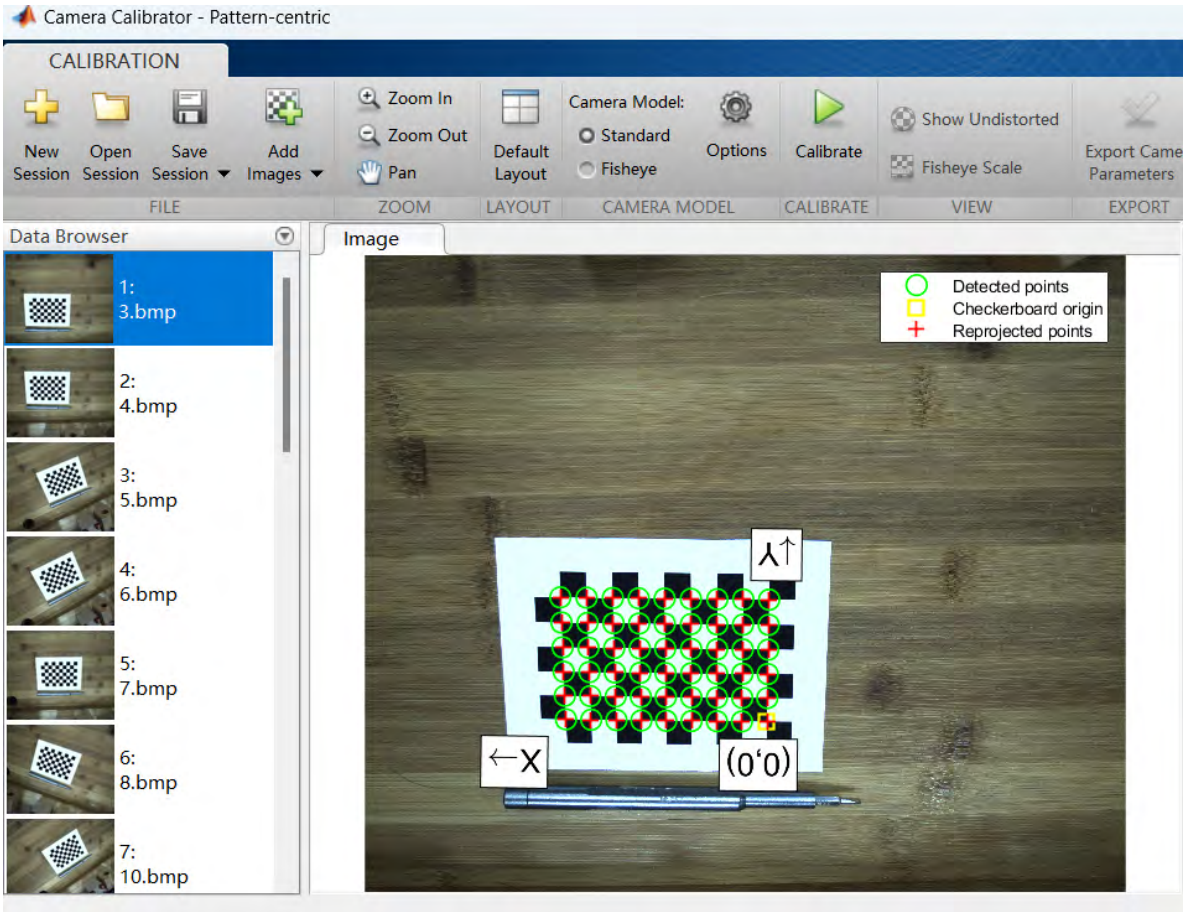
```
flange_pos_list=[
490.187622, 27.700191, 373.764862, 178.573715, 11.630295, 179.867554;
513.424438, 29.686386, 387.674194, 178.581970, 14.645520, 179.605179;
510.206329, -43.655087, 386.155243, 154.941681, 12.465632, 171.126648;
510.902710, -41.008129, 381.313263, 150.081543, 10.785478, 168.497025;
511.239014, 4.498766, 380.509827, 177.289825, 13.694636, -179.478180;
510.584259, 47.938793, 381.490906, -161.543716, 13.041048, -170.422592;
443.654175, -45.077854, 398.959869, 141.956635, 18.927692, 165.277161;
525.400330, 112.360596, 451.545166, -137.946640, 0.181676, -163.115829;
525.400330, 112.360596, 451.545166, 141.074722, 16.713207, -177.561630;
525.400330, 112.360596, 451.545166, 113.447128, 15.994806, 174.517410;
553.645996, 119.681534, 413.980530, 114.157448, 15.584733, 177.127411;
555.071777, 120.600258, 414.013428, 115.193703, 15.505706, 176.406357;
647.963928, 198.840973, 384.385468, -179.663635, -8.606106, -155.519669;
647.963928, 198.840973, 384.385468, 144.068619, 8.508261, -155.499222;
686.108582, -44.710423, 393.953003, 94.076500, -5.077009, -165.197021;
692.988159, -45.347878, 392.334961, 114.569489, -7.627120, -171.790085;
693.483276, -37.013592, 392.334961, 104.366219, -5.936942, -170.499847;
690.768188, -89.168144, 394.644073, 91.614014, -15.754326, -173.037857;
690.768188, -89.168144, 394.644073, 151.410263, -14.121335, 170.099182;
651.298950, -147.716858, 419.132141, 141.648773, -21.879885, 162.412277;
632.434387, -170.233124, 406.022125, 138.922897, -24.018105, 163.376740;
515.601624, -192.849319, 353.179657, 84.409576, -26.513702, 173.749405;
564.366516, -208.410614, 403.313629, 90.331535, -30.176939, 169.490906;

];
```

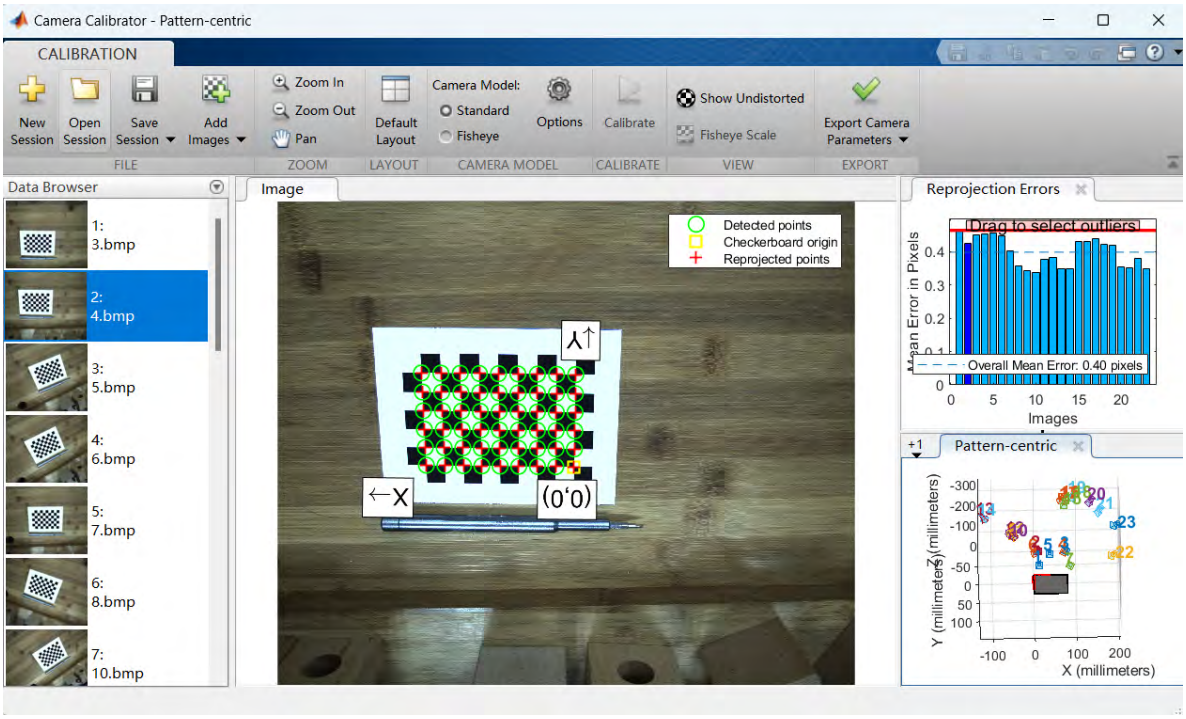
在 Options 中径向畸变系数 (Radial Distortion) 选为两阶 (2 Coefficients)，计算 (Compute) 选择 Skew



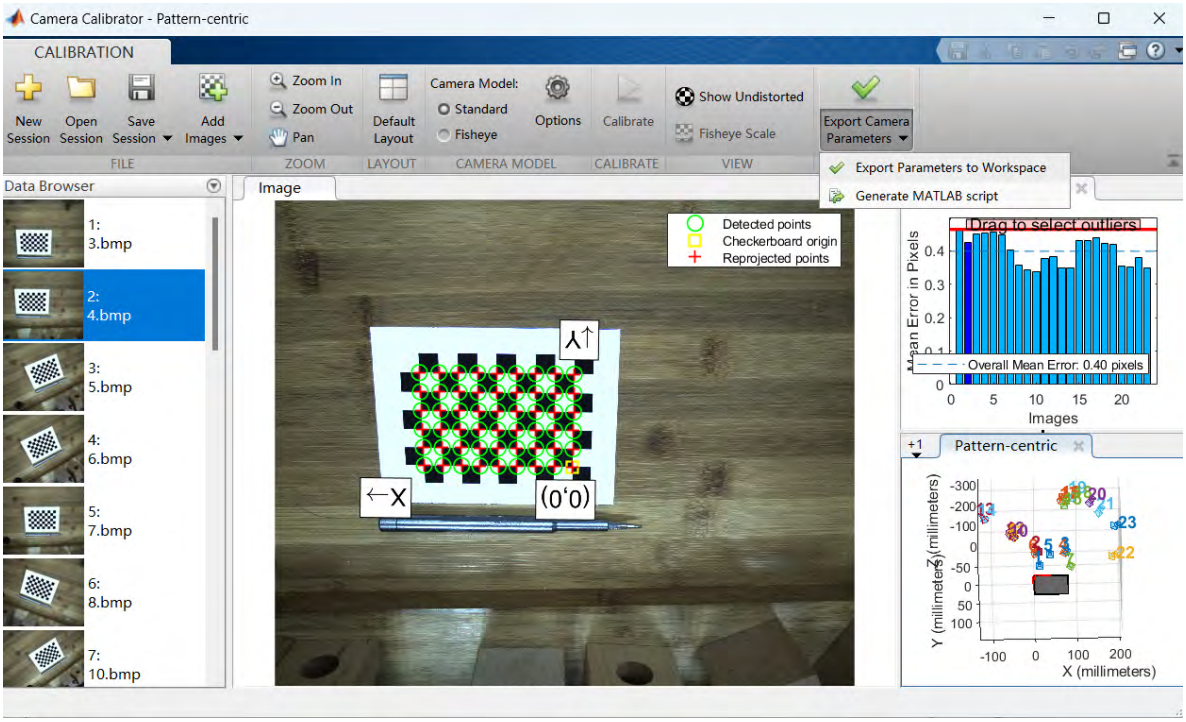
点击 Calibrate 图标进行相机标定



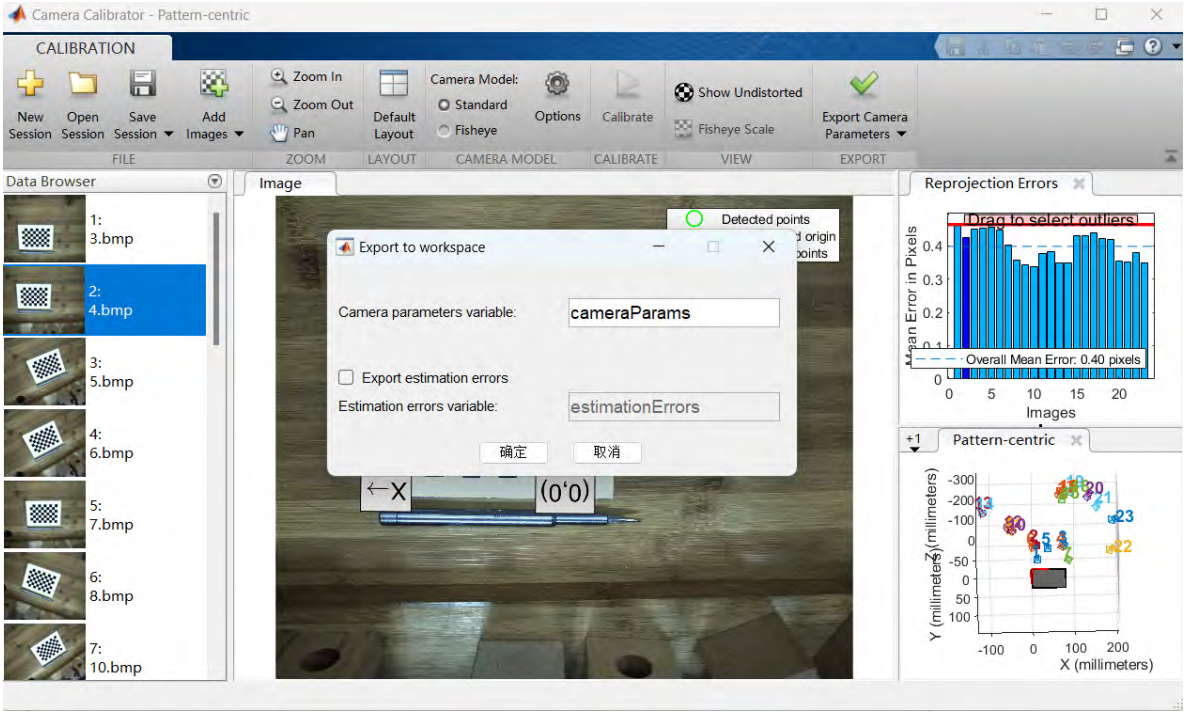
在相机右下角可以看到标定好的相机相对于标定板的位置



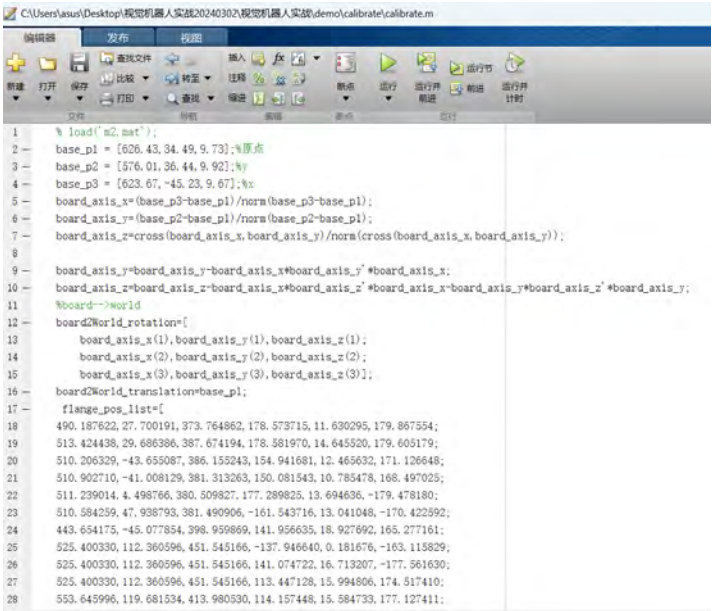
点击 Export Camera Parameter 将标定好的相机参数导出



变量默认为 cameraParams



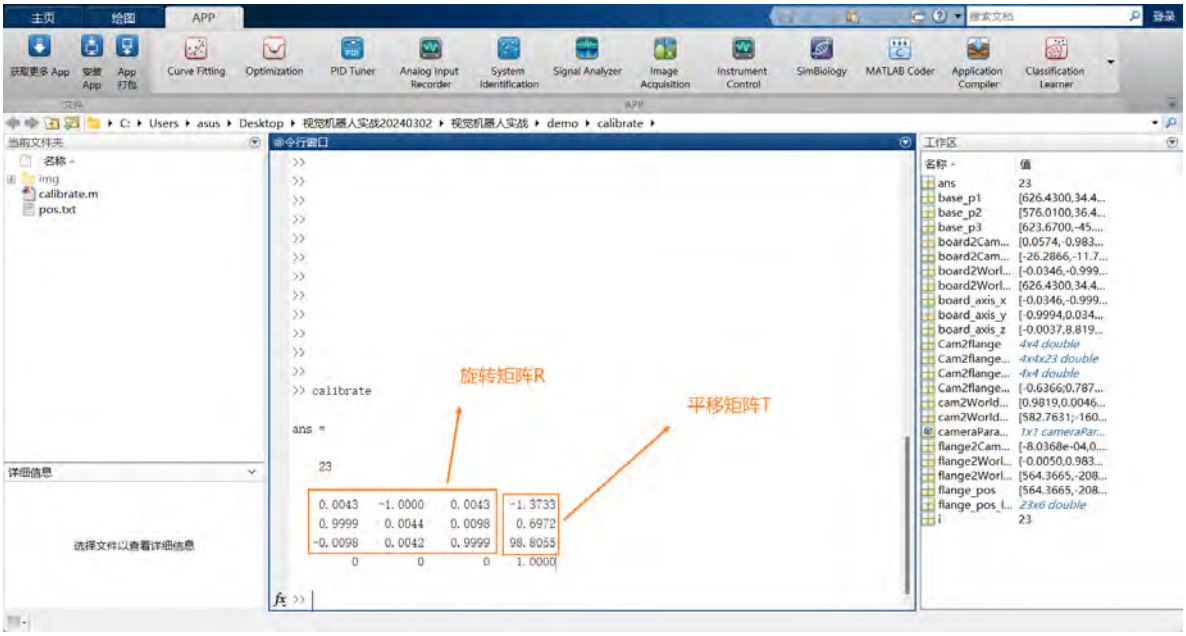
然后运行 calibrate.m 代码



```
% load('m2.mat');
base_p1 = [626.43, 34.49, 9.73];%原点
base_p2 = [576.01, 36.44, 9.92];%y
base_p3 = [623.67, -45.23, 9.67];%x
board_axis_x=(base_p3-base_p1)/norm(base_p3-base_p1);
board_axis_y=(base_p2-base_p1)/norm(base_p2-base_p1);
board_axis_z=cross(board_axis_x,board_axis_y)/norm(cross(board_axis_x,board_axis_y));

board_axis_y=board_axis_y-board_axis_x*board_axis_y'*board_axis_x;
board_axis_z=board_axis_z-board_axis_x*board_axis_z'*board_axis_x-board_axis_y*board_axis_z'*board_axis_y;
%board-->world
board2world_rotation=[
    board_axis_x(1),board_axis_y(1),board_axis_z(1);
    board_axis_x(2),board_axis_y(2),board_axis_z(2);
    board_axis_x(3),board_axis_y(3),board_axis_z(3)];
board2world_translation=base_p1;
 flange_pos_list=[
490.187622,27.700191,373.764862,178.573715,11.630295,179.867554;
513.424438,29.686386,387.674194,178.581970,14.645520,179.605179;
510.206329,-43.655087,386.155243,154.941681,12.465632,171.126648;
510.902710,-41.008129,381.313263,150.081543,10.785478,168.497025;
511.239014,4.498766,380.509827,177.289825,13.694636,-179.478180;
510.584259,47.938793,381.490906,-161.543716,13.041048,-170.422592;
443.654175,-45.077854,398.959869,141.956635,18.927692,165.277161;
525.400330,112.360596,451.545166,-137.946640,0.181676,-163.115829;
525.400330,112.360596,451.545166,141.074722,16.713207,-177.561630;
525.400330,112.360596,451.545166,113.447128,15.994806,174.517410;
553.645996,119.681534,413.980530,114.157448,15.584733,177.127411;
```

此时在 Matlab 命令行窗口就能看到相机相对于法兰的旋转平移矩阵 (R,T)



根据本章开头的描述，我们算得了相机坐标系相对于法兰坐标系的相对位置关系(Tips：相机和法兰之间的连接是固定的，所以相对位置关系在机械臂运动时保持不变)，法兰相对于机械臂 Base 坐标系的相对关系也是已知的，因此，相机坐标系相对于机械臂 Base 坐标系的相对关系就得到了。

至此手眼标定结束，我们已经可以计算出相机坐标系在机械臂 Base 坐标系下的位置了。

7.2 积木抓取模块硬件搭建

视觉模块包括：工业相机，相机镜头，可调光源，视觉控制器，VisionMaster 加密狗。

其中工业相机，相机镜头，可调光源，视觉控制器的具体选型已经在“第 7 章 KUKA 机器人的视觉积木抓取实战”的“视觉系统构成”章节描述的很完整了，我们主要看抓取模块。(VisionMaster 必须要在视觉控制器中插入加密狗才能正常运行。)

图 7.6: 加密狗工作状态红灯亮起

抓取模块包括：

1. 气泵

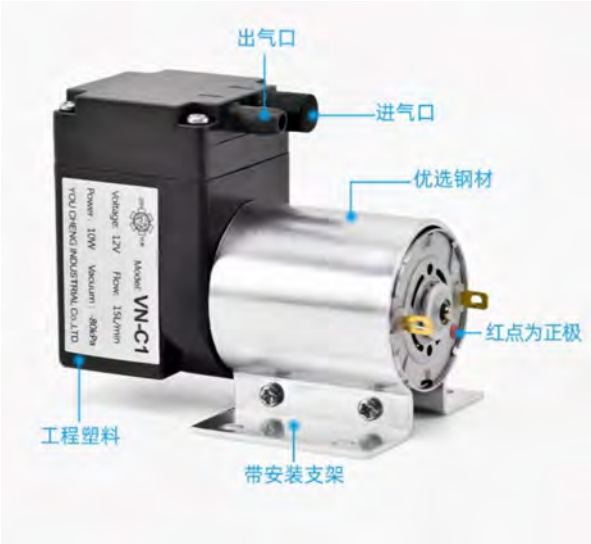


图 7.7: 微型直流气泵

2. 电磁阀

图 **7.8:** 常闭电磁阀

3. 真空吸盘

图 **7.9:** 硅胶吸盘

4. 继电器

图 **7.10:** 直流控制直流固态继电器

5. 气管

图 **7.11:** PU 气管

6. 气管快速接头

图 **7.12:** 各种型号的气管快速接头

7.24v 开关电源

图 **7.13:** 24V 直流开关电源

8. 气压传感器

图 7.14: SMC 数显压力表开关

9.3D 打印法兰-相机连接装置

图 7.15: 法兰-相机连接装置

所有的设计都是围绕抓取积木展开的，常见的抓取结构有夹爪抓取和吸盘抓取两种。

夹爪可以将每次抓取的积木放于工具头固定位置，但是结合考虑积木搭建场景的要求，夹在积木侧边的夹爪在旋转的时候会触碰到已经搭建好的积木部分，导致搭建失败，所以在本项目中选取吸盘结构。吸盘结构抓取部分整体都位于积木上方，所以在搭建同一层积木的时候，平移旋转的干涉会变得最小。

可以看到在 3D 打印法兰-相机连接装置的末端有四个气孔，可以连接吸盘。配合气泵的使用可以轻松抓起积木。

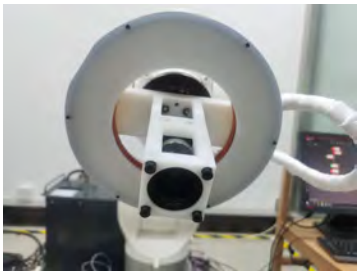


图 7.16: 连接吸盘的抓取末端

光源线和气管连接至如图 (a) 的机械臂第四轴，再通过如图 (b) 的机械臂底座连接到下图 (c) 的硬件控制模块。

(a) 机械臂第四轴 (b) 机械臂底座背面 (c) 硬件控制模块

气管与电磁阀和气泵相连，一个电磁阀配合一个气泵组成四组通路，因此抓取末端的 4 个吸盘都可以独立控制吸放。当气泵工作，气管内部抽成真空状态，当接触积木表面时气压就会将积木吸住，达到抓取的功能。当电磁阀工作就会将气管和大气导通，负压消失，积木由于重力作用达到放的效果。

图 7.17: 四组通路

现在吸放的功能已经实现了，要如何才能反馈积木是否吸住呢，此时就需要加入 SMC 数显压力表开关传感器来检测气管内部的气压，可以根据此气压值判断积木是否吸住，若积木完整吸住，气压小于预定值时 SMC 会发出电压信号，将此电信号传输给机械臂就能判断积木是否完全吸住，极大的增加了系统的稳定性。

气泵打开 SMC 读数　　吸住小球 SMC 读数

抓取功能已经实现，现在只要解决传感器、气泵、电磁阀、光源与机械臂之间的控制问题即可。

此时就需要配置机械臂控制输入输出的 X12 接口。

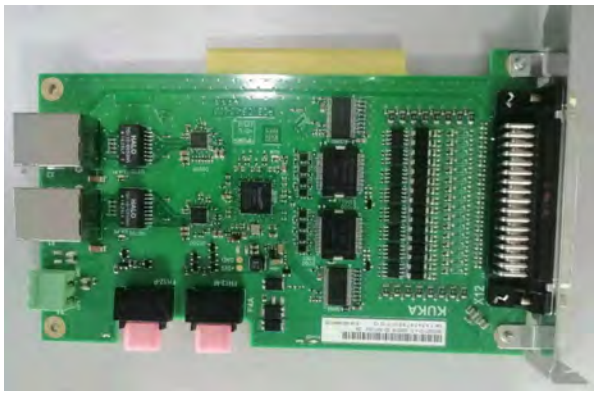
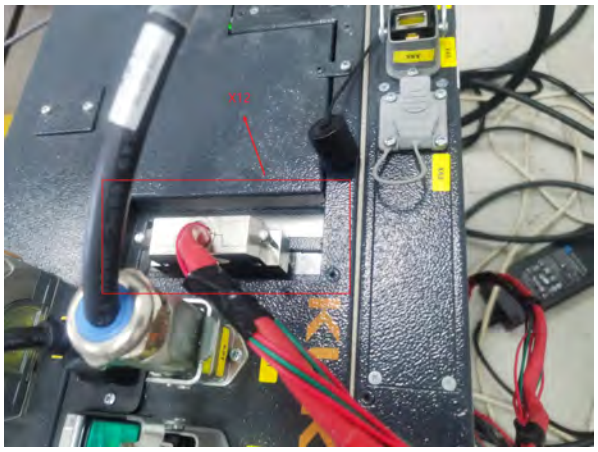


图 7.18: X12 在控制柜内部的 FPGA

X12 在控制柜的位置如下图所示



现在给出 X12 接口的数字序号图，如 7.19 所示

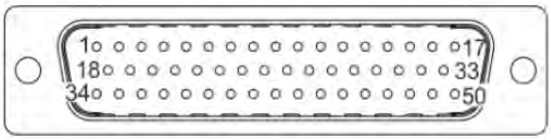


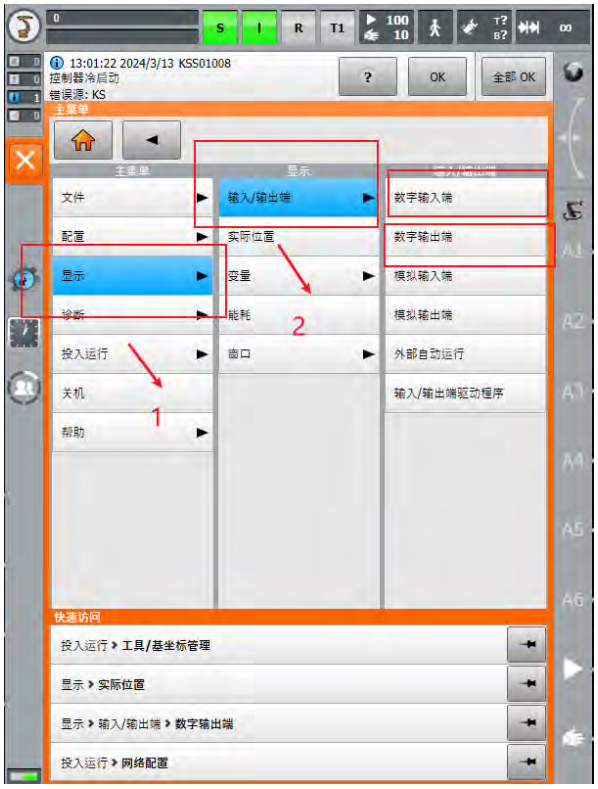
图 7.19: 接口编号

同时给出接口对应的输入输出

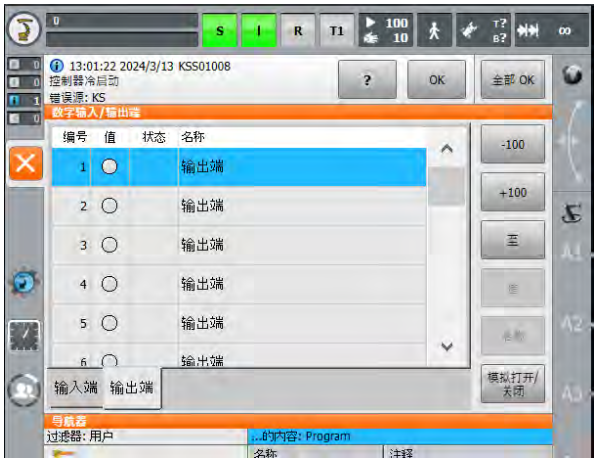
X12 IOB 插头配置

针脚	说明	功能
1	数字输入端1	0V参考点
2	数字输入端2	
3	数字输入端3	
4	数字输入端4	
5	数字输入端5	
6	数字输入端6	
7	数字输入端7	
8	数字输入端8	
9	数字输入端9	
10	数字输入端10	
11	数字输入端11	
12	数字输入端12	
13	数字输入端13	
14	数字输入端14	
15	数字输入端15	
16	数字输入端16	
17	数字输出端17高端	24v开关
18	数字输出端18高端	
19	数字输出端19高端	
20	数字输出端20高端	
21	数字输出端21高端	
22	数字输出端22高端	
23	数字输出端23高端	
24	数字输出端24高端	
25	数字输出端25高端	
26	数字输出端26高端	
27	数字输出端27高端	
28	数字输出端28高端	
29	数字输出端29高端	
30	数字输出端30高端	
31	数字输出端31高端	
32	数字输出端32高端	
33	GND	最大 4A 的保险装置 (FH12-N)
34	数字输出端34低端	0V开关
35	数字输出端35低端	
36	数字输出端36低端	
37	数字输出端37低端	
38	数字输出端38低端	
39	数字输出端39低端	
40	数字输出端40低端	
41	数字输出端41低端	
42	数字输出端42低端	
43	数字输出端43低端	
44	数字输出端44低端	
45	数字输出端45低端	
46	数字输出端46低端	
47	数字输出端47低端	
48	数字输出端48低端	
49	数字输出端49低端	
50	外部供电	24 V，4A 保险装置 (FH12-P)

然后打开示教器显示-> 输入/输出端, 里有数字输出端和数字输入端两个选项

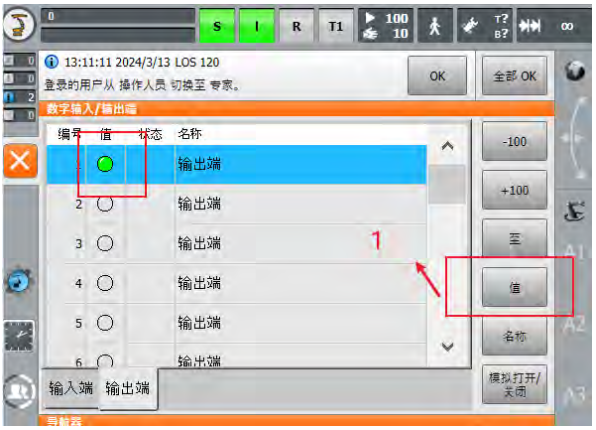


先看数字输出端，可以看到被编号 1-4096 的输出端



在图中的编号 1 输出端对应的是 X12 接口中的第 17 针脚，编号 8 输出端对应的是 X12 接口中的第 18 针脚依次对应下来，输出 IO 口直到 32 针脚数字输出 32 高端截止。

当我们选定编号 1 输出端，按下使能键与值键是，编号 1 输出端的值将从灰色改成绿色，此时 X12 口对应的第 17 针脚有高电压输出，17 针脚与 34-49 数字输出低端任意针脚连通即可生成 24v 的回路，将其接在固态继电器 A1，A2 控制端即可通过示教器上编号 1 输出端的值的状态去控制固态继电器 L1，L2 端的通断。因此只要将固态继电器 L1，L2 端一对一串联进气泵，电磁阀，光源的供电电路中即可达到机械臂控制外部元件的目的。



在本项目中总共用到了 4 个电磁阀，4 个气泵，1 个光源，4 个 SMC 气压传感器，对应的使用了 13 个固态继电器达到小电流控制大电流的目的。这 13 个设备都是并联在 24V 开关电源处，因此每个器件都能做到单独控制。

图 7.20: 继电器 A1,A2 控制端

在本项目中在 X12 针脚的输出端用到了 19,20,21,22,24,25,26,27,28 数字输出高端针脚和 34,35 数字输出低端针脚，数字输出高端针脚引出的导线分别连接至 9 个固态继电器的 A1 接口处，在 A2 接口处引出导线接至 34,35 数字输出低端针脚处，这样回路就行成了。(需要注意的是 34,35 数字输出低端对应的是 17,18 数字输出高端，在示教器上的输出端编号为 1,2, 当 1,2 输出端的值变为绿色时 34,35 数字输出低端和对应 17,18 数字输出高端才会被打开，此时其他针脚的数字输出高端若值为绿色，与 34,35 数字输出低端相连也能生成回路。)

接下来讲一下在本项目中数字输出高端针脚对应控制的设备

19,20,21,22 数字输出高端, 即示教器 3,4,5,6 输出端——控制 4 个电磁阀

25,26,27,28 数字输出高端, 即示教器 9,10,11,12 输出端——控制 4 个气泵

24 数字输出高端, 即示教器 8 输出端——控制 1 个光源

在示教器数字输出端界面将输出端 1,2,8 值打开

图 7.21: 输出端 1,2,8 值变绿

则光源打开：

图 7.22: 光源开启

气泵、电磁阀的开启方式同理，不再赘述。

输出端控制外部硬件设备的流程已经介绍完了，接下来介绍机械臂控制柜如何得到气压传感器的电信号。

在上文中我们知道当 SMC 气压传感器的气压低于设定的阈值时就会传出高电平型号，类似的将此信号放在固态继电器的 A1，A2 控制端口，L1，L2 口将电信号传入到 X12 数字输入端中，当控制柜收到电信号时输入端值会变绿。

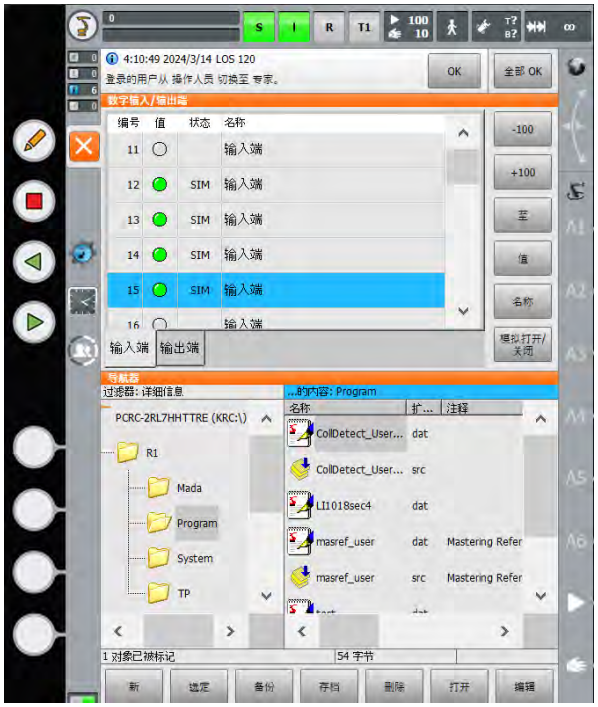


图 7.23: 机械臂控制柜收到电信号

在本项目中 4 个气压传感器的对应的 X12 数字输入端针脚为 12,13,14,15，同时示教器上对应的数字输入端为 12,13,14,15。当我们打开气泵，堵住机械臂末端的吸盘，对应的输入端值就会变绿。

在 KUKA 机械臂 KRC 程序中输入输出端的值状态即为变量 $IN[编号]=TRUE/FALSE, 或者 $OUT[编号]=TRUE/FALSE 的值。

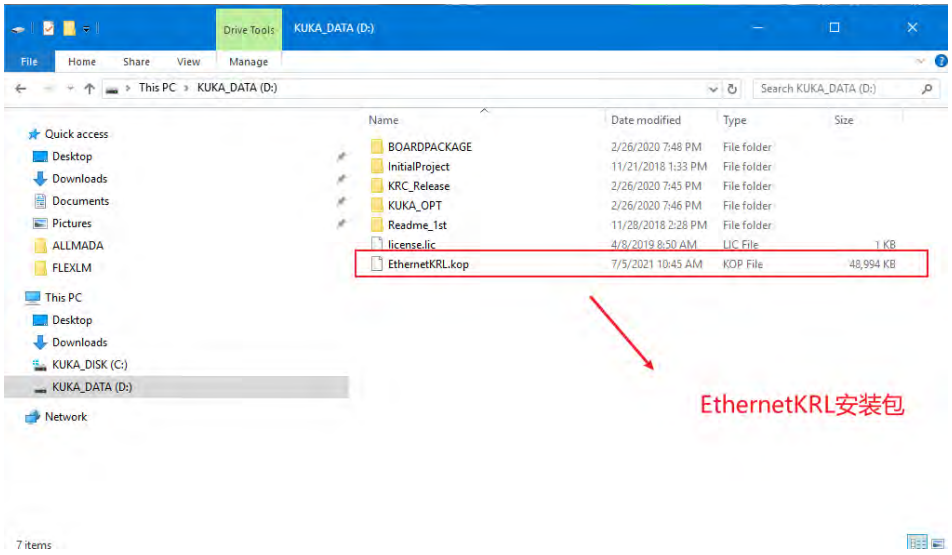
7.3 VisionMaster——机械臂与视觉控制器通信桥梁

我们在海康威视官网上找到并下载 VisionMaster 软件安装包，在视觉控制器上安装海康 VisionMaster 算法平台 4.2.0 版本。

7.4 视觉机械臂实战——通信

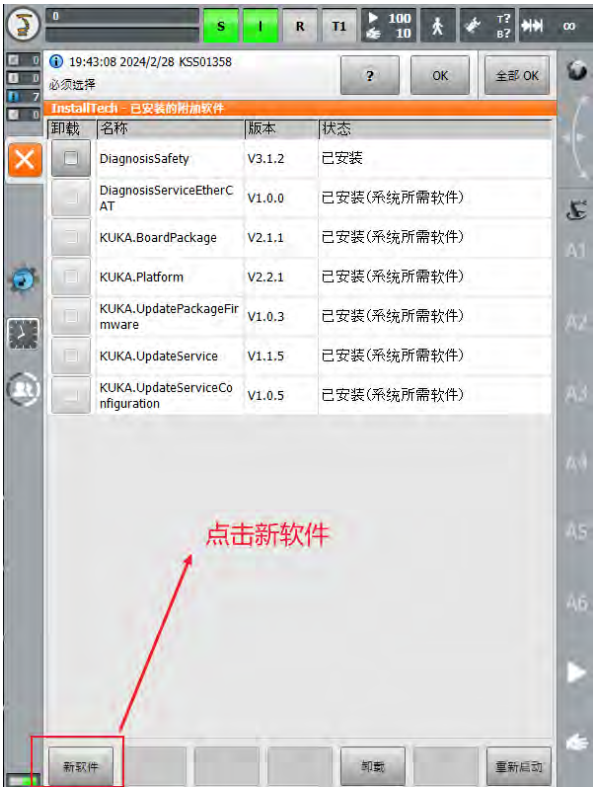
7.4.1 机械臂信息传输到视觉控制器

在本项目中通过 EthernetKRL 工具包实现 kuka 机械臂与外界设备的网络通信。下面进行 EthernetKRL 软件包的安装，将 EthernetKRL.KOP 安装包复制到 D 盘文件下，

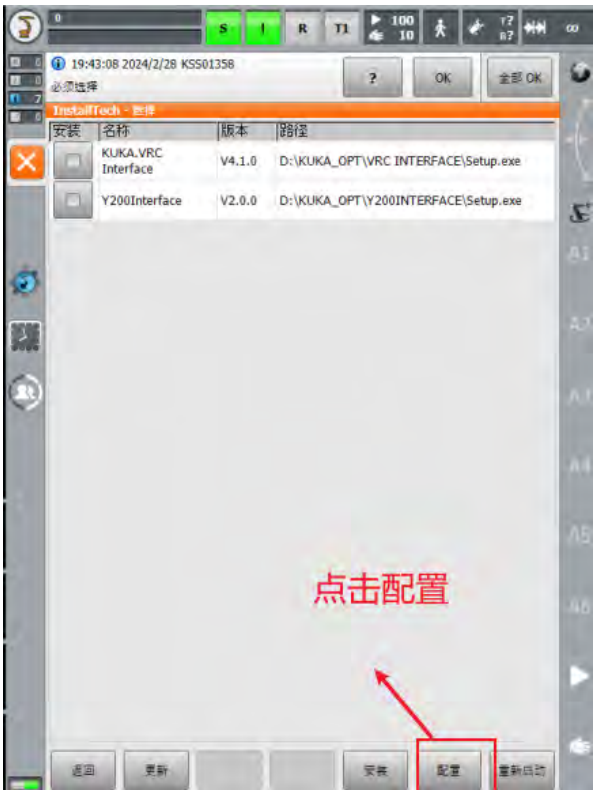


再次在示教器主界面上选择投入运行-> 辅助软件

然后点击新软件选项

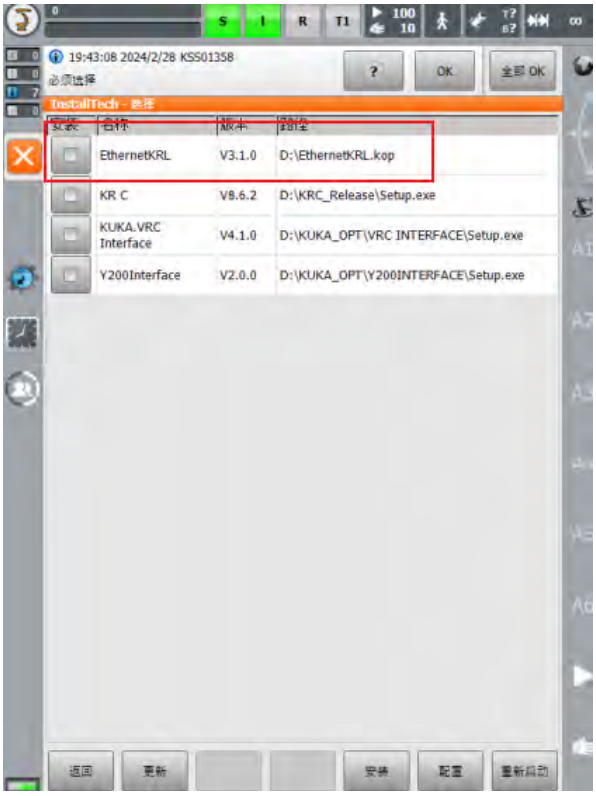


点击配置选项

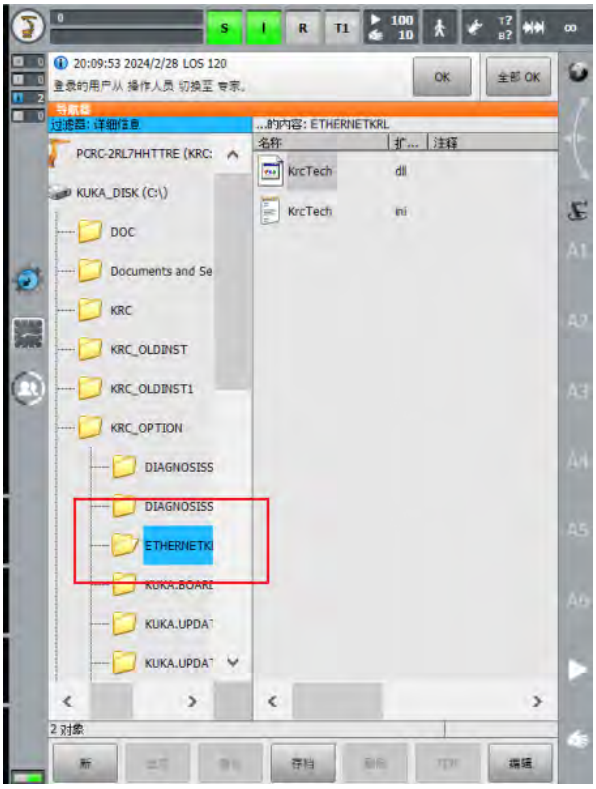


点击路径选择，选择我们放置 EthernetKRL.KOP 安装包路径，D:/之后点击保存

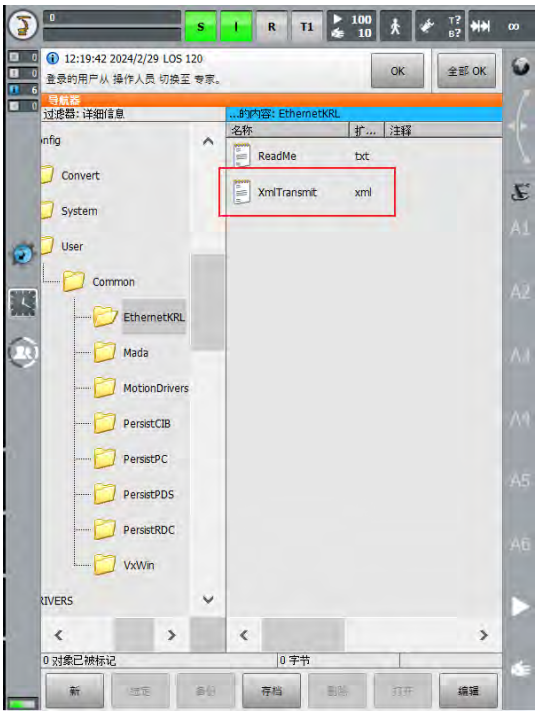
此时在安装界面可以选择安装 EthernetKRL



安装完成之后可以在 C:/KRC_OPTION/路径下找到 ETHERNETKRL 文件夹



机械臂以太网连接通过 xml 文件进行配置，针对每个连接，必须在机器人控制系统的目录 C:/KRC/ROBOTER/Config/User/Common/EthernetKRL 中定义一个配置文件，在 C:/KRC/ROBOTER/Config/User/Common/EthernetKRL 中创建 XmlTransmit.xml 文件。



本项目的 XmlTransmit.xml 文件内容在第七章给出。xml 文档中指定了外部通讯

设备的 IP 地址，为了确保通信正常，需要将机械臂的 IP 地址设为同一网段。

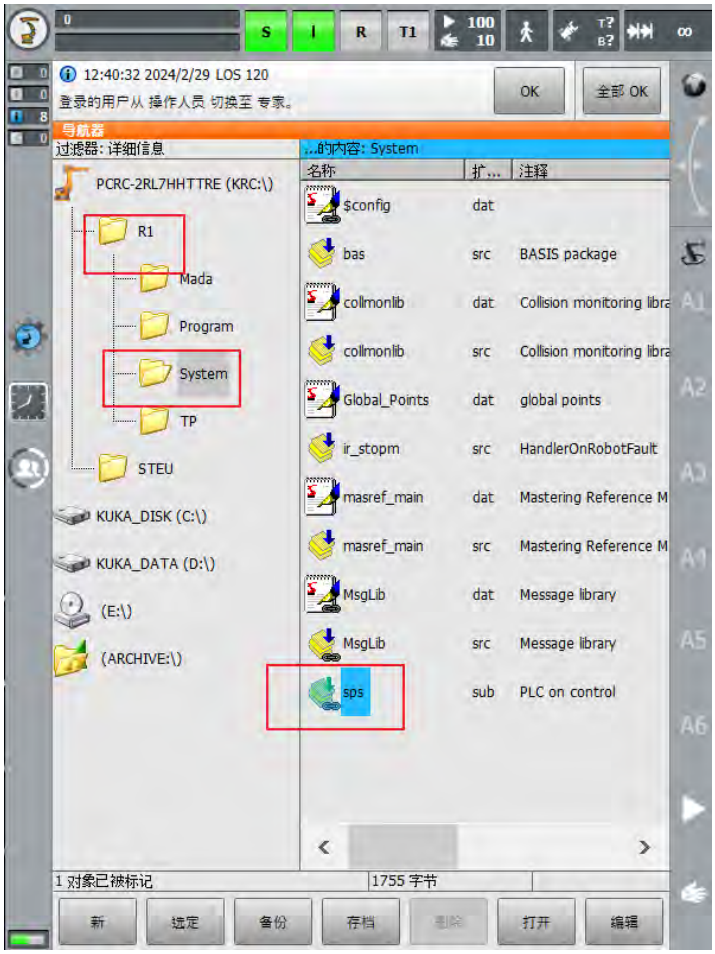
首先要在示教器中将用户组切换到专家模式。

然后一次点击主界面-> 投入运行-> 网络配置，将 IP 地址设置为 192.168.1.110

然后单击保存，重启设备更新 IP 地址。

为了实时获取机械臂的运动位姿，因此需要配置机器人提交解释器的 SUB 文件。

在 R1/System 文件下找到 sps.sub 文件：



找到注释行;FOLD USER DECL,在此处声明 EthernetKRL 变量 RET:DECL EKI_STATUS RET

```
1   &ACCESS RVO
2   &REL 37
3   &COMMENT PLC on control
4   DEF  SPS ( )
5     ;FOLD DECLARATIONS
6   BASISTECH DECL
11    ;FOLD USER DECL
12    ; Please insert user defined declarations
13
14     DECL EKI_STATUS RET
15
16    ;ENDFOLD (USER DECL)
17    ;ENDFOLD (DECLARATIONS)
```

找到代码中的 LOOP 与 ENDLOOP 代码行在

;FOLD USER PLC

;Make your modifications here

注释行下添加发送坐标的代码，LOOP 与 ENDLOOP 是循环执行的，因此坐标可以循环发送

```
43    LOOP
44      WAIT FOR NOT($POWER_FAIL)
45  BASISTECH PLC
48  BACKUPMANAGER PLC
53      ;FOLD USER PLC
54      ;Make your modifications here
55  number1=1
56  IF (number1==1) THEN
57   REPEAT
58   UNTIL number1<>1
59  ENDIF
60  LOOP
61  number=0
62  IF (number==1) THEN
63   REPEAT
64   UNTIL number<>1
65  ENDIF
66  ;FOLD Write data to connection
67    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@x",$POS_ACT.X)
68    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@y",$POS_ACT.Y)
69    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@z",$POS_ACT.Z)
70    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@a",$POS_ACT.A)
71    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@b",$POS_ACT.B)
72    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@c",$POS_ACT.C)
73  ;ENDFOLD (Write data to connection)
74  ;FOLD Send data to external program
75    RET = EKI_Send("XmlTransmit","Robot")
76    WAIT SEC 0.05
77
78  IF (number1==1) THEN
79   REPEAT
80   UNTIL number1<>1
81  ENDIF
82  ;ENDFOLD (Send data to external program)
83  ENDLOOP
84      ;ENDFOLD (USER PLC)
85    ENDLOOP
```

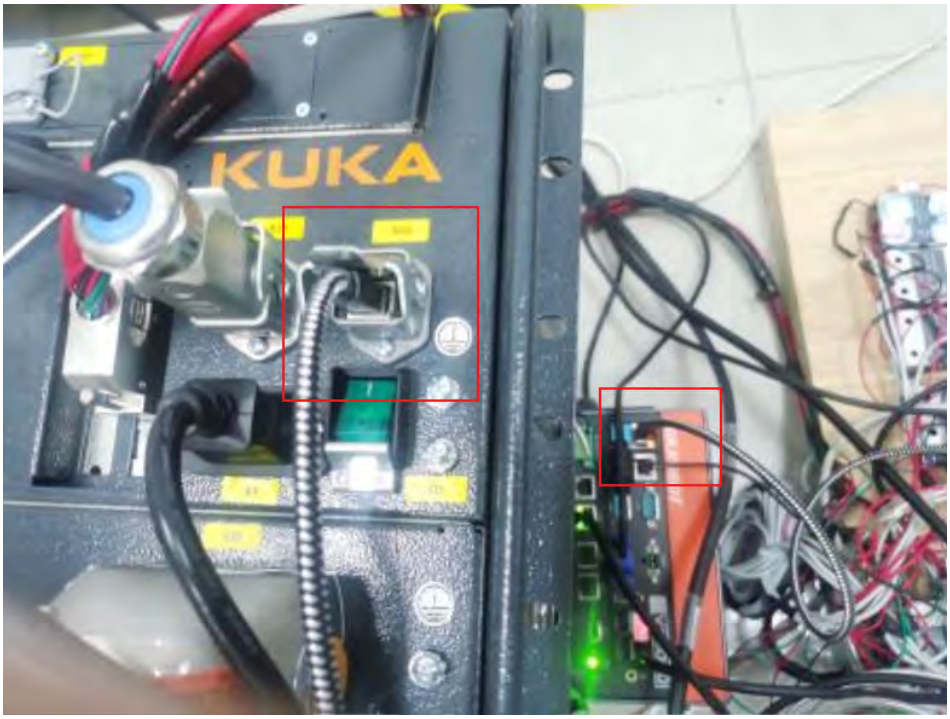
之后只要在机械臂主程序 LI_1018sec4.src 代码中加入以下代码：

```
DECL EKI_STATUS  RET
RET=EKI_Init(”XmlTransmit”)
RET=EKI_Open(”XmlTransmit”)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@f”,0)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@p”,0)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@e”,3500)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@q”,0)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@n”,0)
```

到这里机械臂端向外界设备发送消息的基础配置就已经完成了。

接下来我们讲解视觉控制器端的通讯配置。

将 KUKA 控制柜上的 X66 接口通过网线连接到 MV-VC3301P-128G60 视觉控制器上的网口，组成局域网。



之后将 MV-VC3301P-128G60 视觉控制器的 IP 地址修改为 XmlTransmit.xml 中配置的 IP 地址 192.168.1.115，子网掩码设置为 255.255.255.0。

```
以太网适配器 以太网_5:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :

以太网适配器 以太网_6:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::f99e:26fb:14a3:bebc%15
   IPv4 地址 . . . . . . . . . . . . : 192.168.1.115
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . :

以太网适配器 以太网_3:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::ddb1:8f56:e131:9190%6
   IPv4 地址 . . . . . . . . . . . . : 192.168.1.100
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.1.1

隧道适配器 isatap.{9BB80175-947C-4AEB-9B13-8590B46E882D}:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :

隧道适配器 Teredo Tunneling Pseudo-Interface:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :
```

接着打开 VisionMaster 软件，点击软件菜单栏上的通信管理

选择设备管理，单击设备列表的加号，选择类型 TCP 服务器，将本机 IP，端口与 XmlTransmit.xml 中的配置保持一致，然后打开数据上传与数据自动重连。

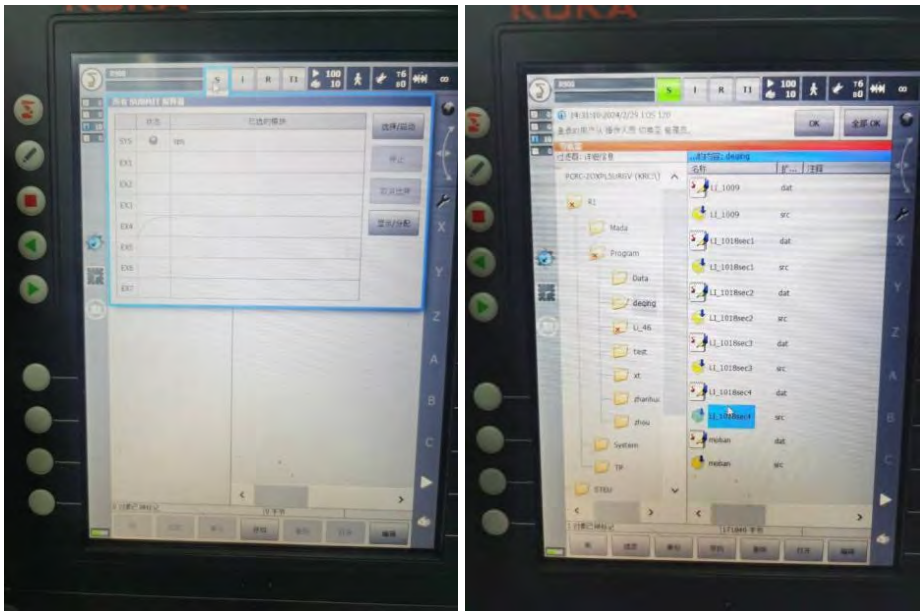
然后在接收事件创建事件管理，处理方式选择文本，事件类型选择协议解析

接着选择创建好的 [1 文本-协议解析]，在绑定设备处选择上文创建好的 1TCP 服务端，分隔符选择“;”，在输出列表中新加一行，名称命名为 out1，类型选择 string。

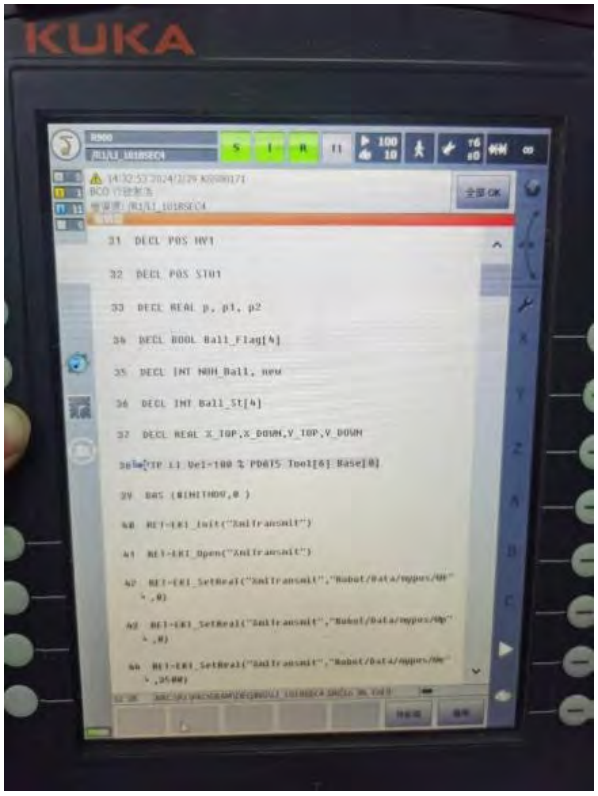
之后在左侧模块栏找到通信模块。

创建 [9 接收数据 1] 模块和 [10 协议解析 1] 模块，将两个模块拖入到 [流程 1] 中，点击接收数据，选择数据源为通信设备，在通信设备选择 [1TCP 服务端]，输入数据命名为 out0

之后在机械臂示教器上选定 S 程序，选定成功 S 图标会变绿



然后选定主程序 LI_1018sec4.src，按住使能键执行运行：



此时可以在 VisionMaster 中的通信管理的接收数据里看到机械臂发过来的位姿坐标信息。

点击 [10 协议解析 1] 配置参数，在方式选择栏内选择文本解析，解析内容选择模块 [9 接收数据 1] 接收数据所获得的字符串变量 out0。

在 [10 协议解析 1] 输出列表内以 string+float+string+……的形式提取所需的数据。

点击右边的链接来将值传给上文所设置的全局变量，到此机械臂的数据已经被完整的获取到了。

图 7.24: 可以将数据传给上文设置的全局变量

7.4.2 视觉控制器信息传输到机械臂

首先设置在机械臂编程中接收数据，在 LI_1018sec4.src 代码中找到第 466-490 行

```
466          RET=EKI_ClearBuffer("XmlTransmit","Sensor")
467          $FLAG[1] = FALSE
468          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
469          WHILE iu < 2
470
471          IF num >= 2 THEN
472             EXIT
473          ENDIF
474
475          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",1111)
476          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",98765111)
477          WAIT FOR  ( $FLAG[1] )
478          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
479          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",0)
480          number=1
481          RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
482          RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
483          RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
484          RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
485          RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
486          RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
487          RET=EKI_GetReal("XmlTransmit","Sensor/p",p)
488
489          WAIT SEC wait_sec
490          number=2
```

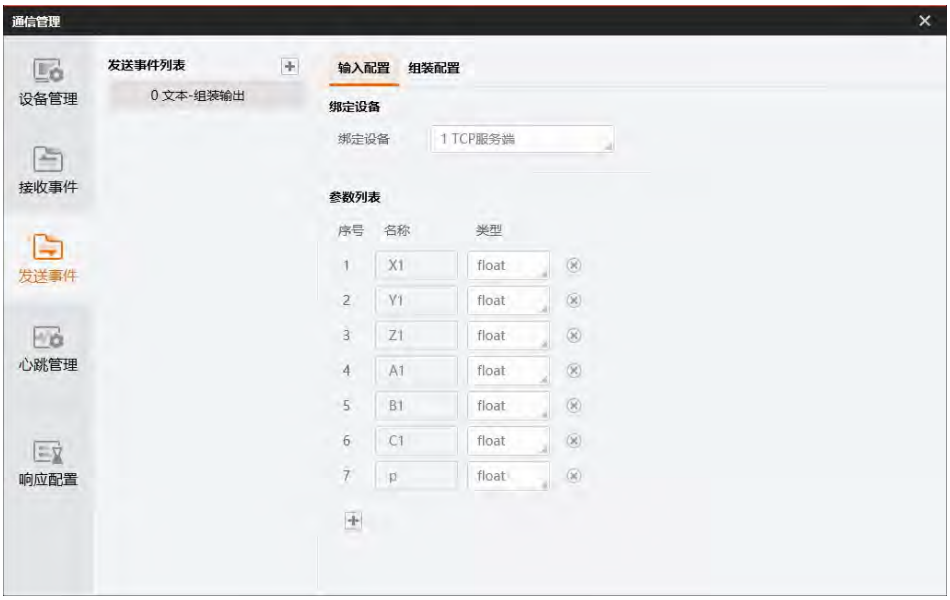
函数 EKI_GetReal 用来接收视觉控制器向机械臂发送数据的变量值，并将值赋给参数的第二个变量 DA1，DA2，DA3，DA4，DA5，DA6，p：

```
RET=EKI_GetReal(”XmlTransmit”,”Sensor/X1”,DA1)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/Y1”,DA2)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/Z1”,DA3)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/A1”,DA4)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/B1”,DA5)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/C1”,DA6)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/p”,p)
```

接着在通信模块中将 [2 发送数据模块 8] 拖入到流程 1 界面中

现在可以配置 VisionMaster 中的发送事件，点击发送事件列表添加事件管理，选择处理方式为文本，事件类型为组装输出。

创建完成之后需要在输入配置中设置绑定设备为上文创建的 [1TCP 服务端]，



参数列表设成 EKI_GetReal 中需要接收的 X1,Y1,Z1,A1,B1,C1,p.

开始组装配置，

之后点击 [2 发送数据 8] 修改基本参数，将输出至修改为发送事件，发送事件选择刚刚生成的 [0 文本-组装输出]，

点击 VisionMaster 菜单栏的单次执行，模块 [2 发送数据 8] 会向 TCP 客户端, 即机械臂发送一次消息。

7.4.3 其他软件与 VisionMaster 的交互

在项目中，信息的传输路径是外部 APP->VisionMaster->KUKA 机械臂。

(1) 创建第二个 TCP 服务端，地址依旧设为 192.168.1.115，端口设为 59151

(2) 按模块运行流程依次配置模块参数。

在 [[11 接收数据 2] 模块选择上文创建的通信设备 [2TCP 服务端]，用于接收其他软件发送的数据。

修改 [12 协议解析 2] 解析接收的数据格式，在本项目中外部 APP 发送的信息格式与上文的协议解析设置相同，都是 string+float+string+……的格式

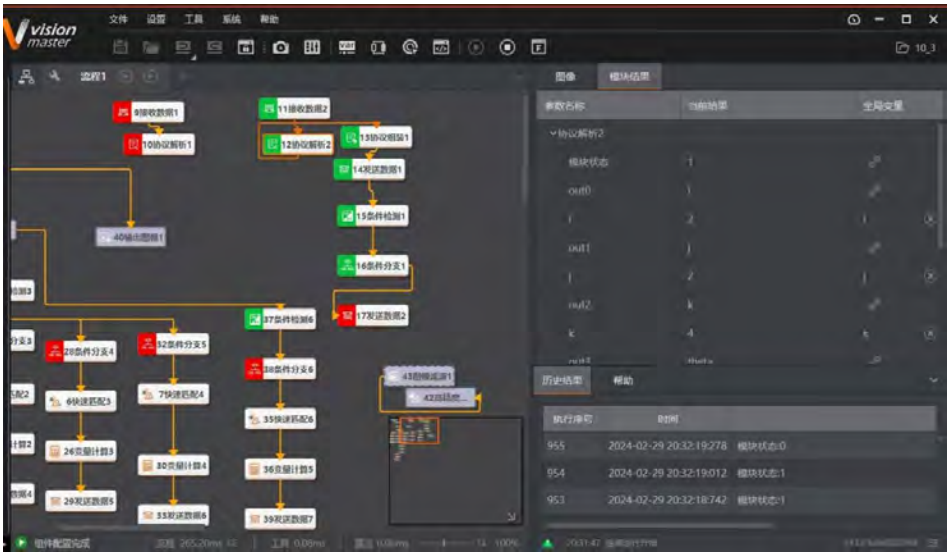


图 7.25: VisionMaster 中的模块结果

紧接着通过 [13 协议组装 1] 组装要发送给外部 APP 的应答消息。

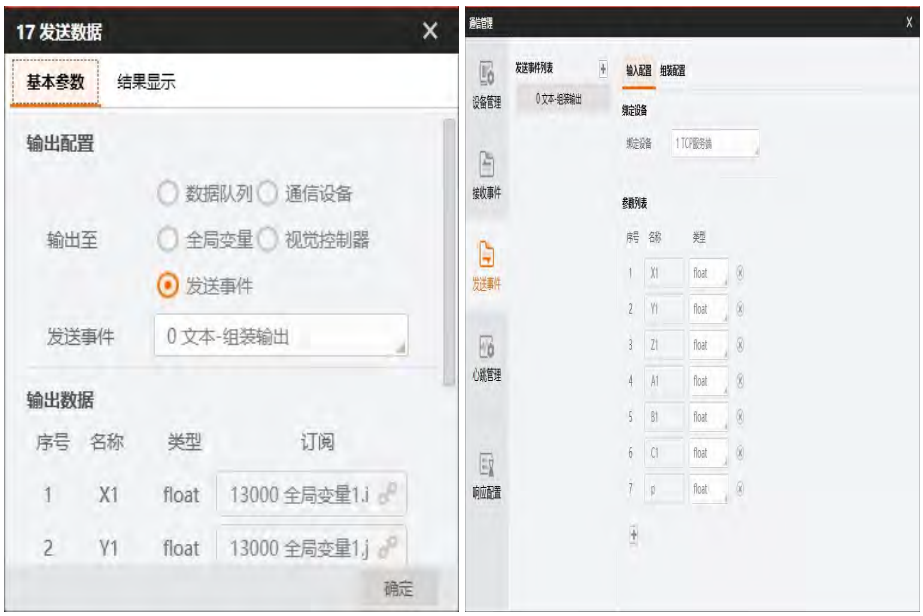
然后将组合好的数据通过 [14 发送数据 1] 传输回外部 APP。

配置模块 [15 条件检测 1]，判断方式选择全部即在下面创建的检测条件变量满足

有效值范围内则判断结果为 OK，在本项目中通过协议解析的变量 i 来控制模块是否继续往下运行。

接着执行模块 [16 条件分支 1]，当模块 [15 条件检测 1] 输出为 OK 时，执行模块 17

模块 17 的作用就是将协议解析得到的变量值传送给 KUKA 机械臂，发送事件选择上文所创建的 [0 文本-组装输出]，通过 [1TCP 服务端] 交互数据。



到此各设备之间的数据交互已经讲完了。

7.5 VisionMaster——图像定位识别

7.5.1 VisionMaster——相机配置

既然是视觉机械臂实战，在整个 VisionMaster 项目中最重要的模块就是图像采集，首先就需要配置 MV-CS-10UC 工业相机。将工业相机插入 USB3.0 接口之中，相机上侧蓝灯闪烁表示连接成功。

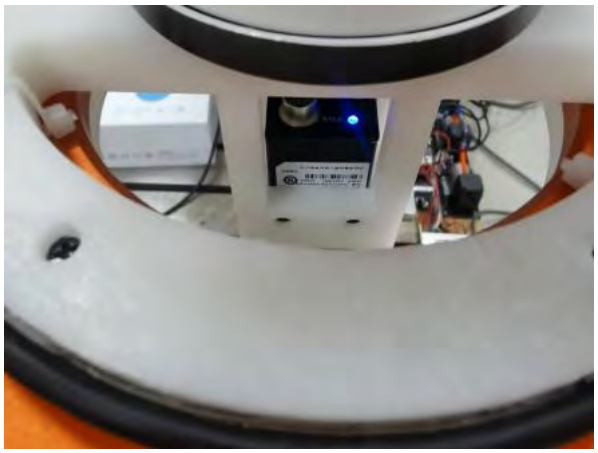
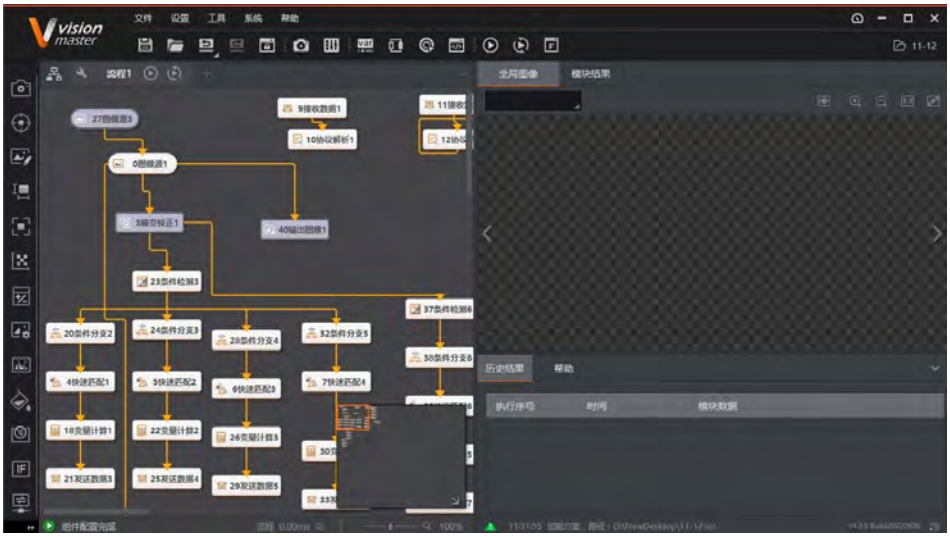


图 7.26: 相机 LED 闪烁

打开 VisionMaster 页面，



打开 VisionMaster 上排菜单栏中的相机管理:

点击相机管理界面的设备列表，选择全局相机并确定。

相机如果连接成功，则可以在选择相机一栏读到我们的相机型号，选择 MV-CS-10UC 工业相机。

之后，根据项目需求，调整相机曝光、Gamma 使能等参数，并对相机的光圈大小以及焦距进行了调节，确保拍摄到的图片在保持清晰度的同时满足项目要求。

图 7.27: 相机参数调整

接着修改相机管理-> 触发设置-> 触发源为 SOFTWARE。

现在我们可以选择采集模块中的图像源，将其拖入到流程界面，得到 [0 图像源 1]。

双击 [0 图像源 1] 修改:

图像源参数-> 相机

1. 关联相机-> 全局相机 1(上文创建)

2. 控制曝光-> 全局变量 e

3.SN 初始值->10000

4. 输出 Mono8-> 打开 (图像必须要输出为 Mono8 格式，否则后续的图像处理会失效)

然后点击 VisionMaster 上排菜单栏中的连续执行，

此时 [0 图像源 1] 变绿，双击 [0 图像源 1 图标]，在右侧显示界面成功显示相机视频流。

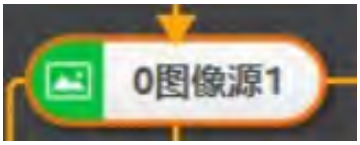


图 7.28: 获取到的相机视频流

7.5.2 VisionMaster——设置全局变量

全局变量，有编程基础的朋友们相信不会陌生，是一种可以被局部获取或设置的变量。在这里将本项目中所用到的全局变量全部设置好：

序号	名称	注释	类型	当前值	输入来源	目标输出	初始化	操作
1	X1		float	650.25	------			
2	Y1		float	306.14	------			
3	Z1		float	450	------			
4	A1		float	0	------			
5	B1		float	180	------			
6	C1		float	0	------			
7	X		float	580.13	流程1.协议解析1.X			
8	Y		float	-54.27	流程1.协议解析1.Y			
9	Z		float	0	------			
10	A		float	180	流程1.协议解析1.A			
11	B		float	0	------			
12	K		float	2100	流程1.协议解析1.K			

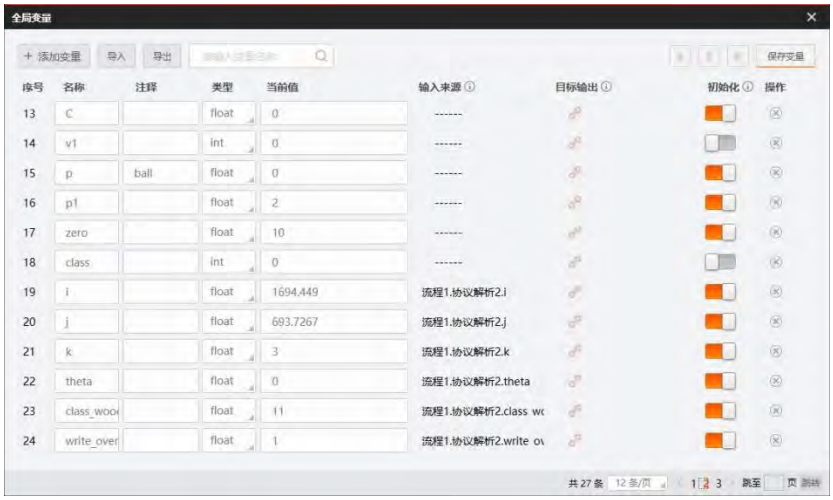


图 7.29: VisionMaster 项目所用到的所有全局变量

7.5.3 VisionMaster——定位

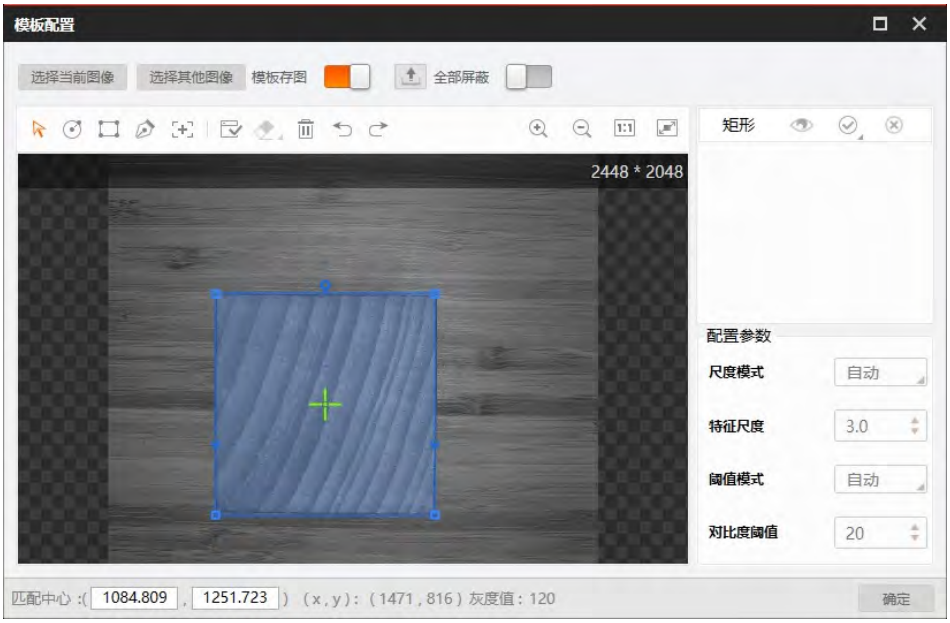
获取到图像之后, 就可以处理图像并进行定位了，在定位中选择快速匹配。

1. 模板训练

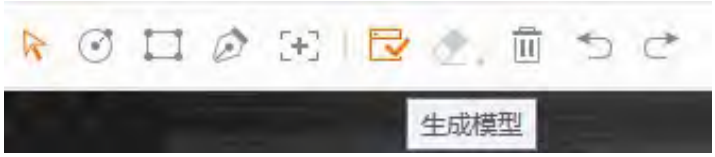
在相机获取到一张图片之后，双击快速匹配模块，打开参数配置页面，选择特征模板选项卡，

点击创建按钮，进入模板配置页面，创建矩形掩膜。

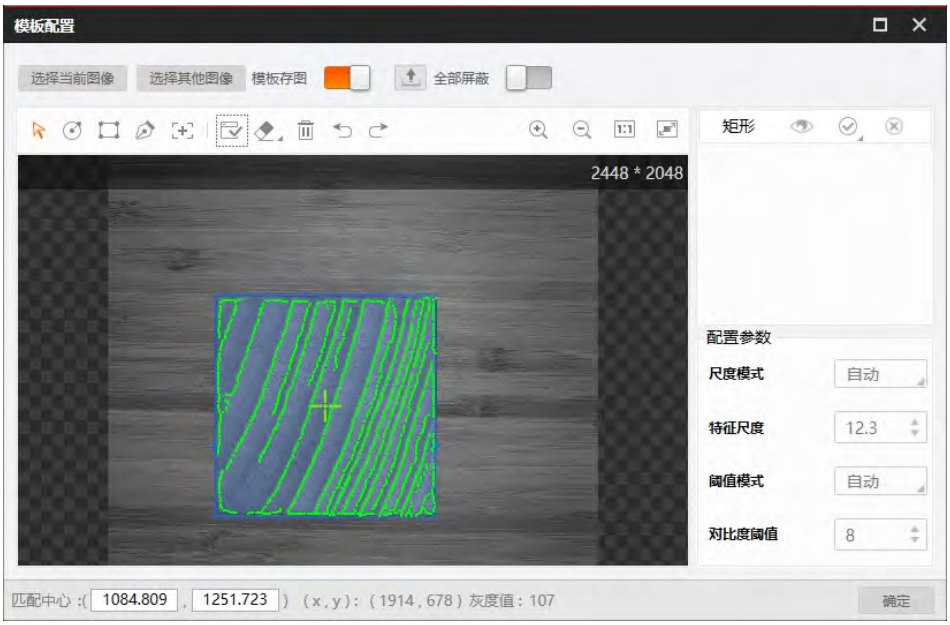
框出需要识别的积木表面



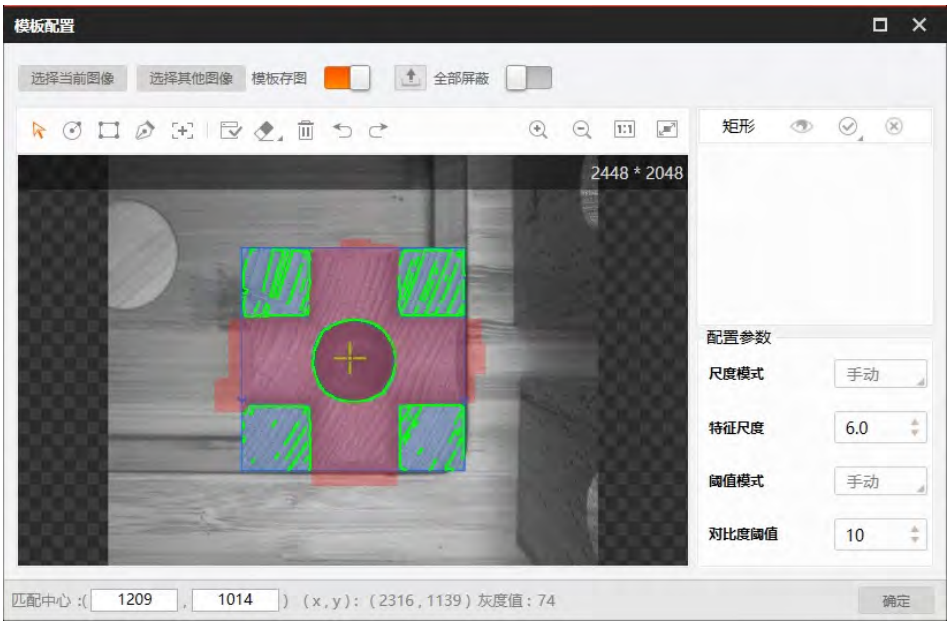
然后点击生成模型



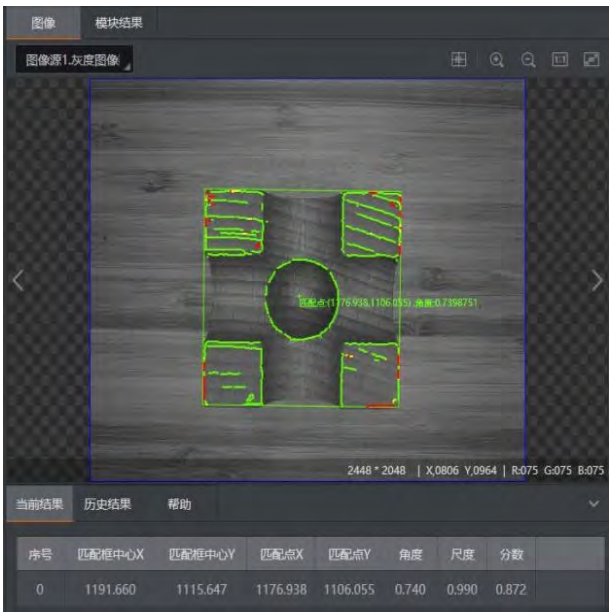
生成绿色模版匹配点，可以通过修改特征尺度和对比度阈值来改变匹配点的数量与结构。

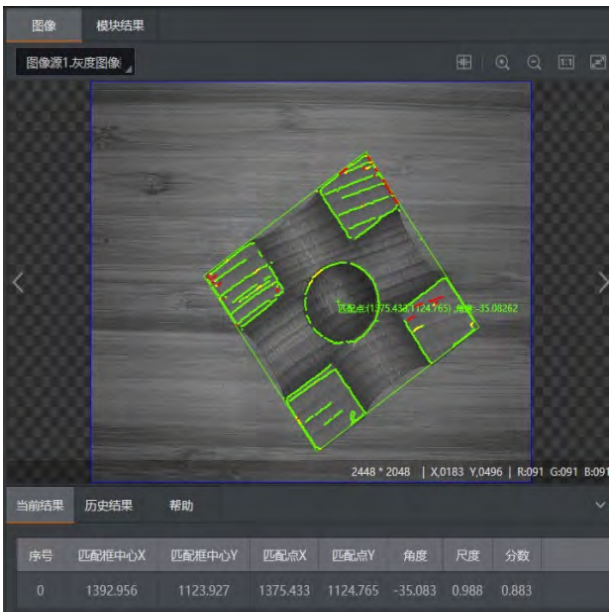


接下来，以拥有轨道面的积木为例，将生成模板轨道上的点全部抹去，如下所示：

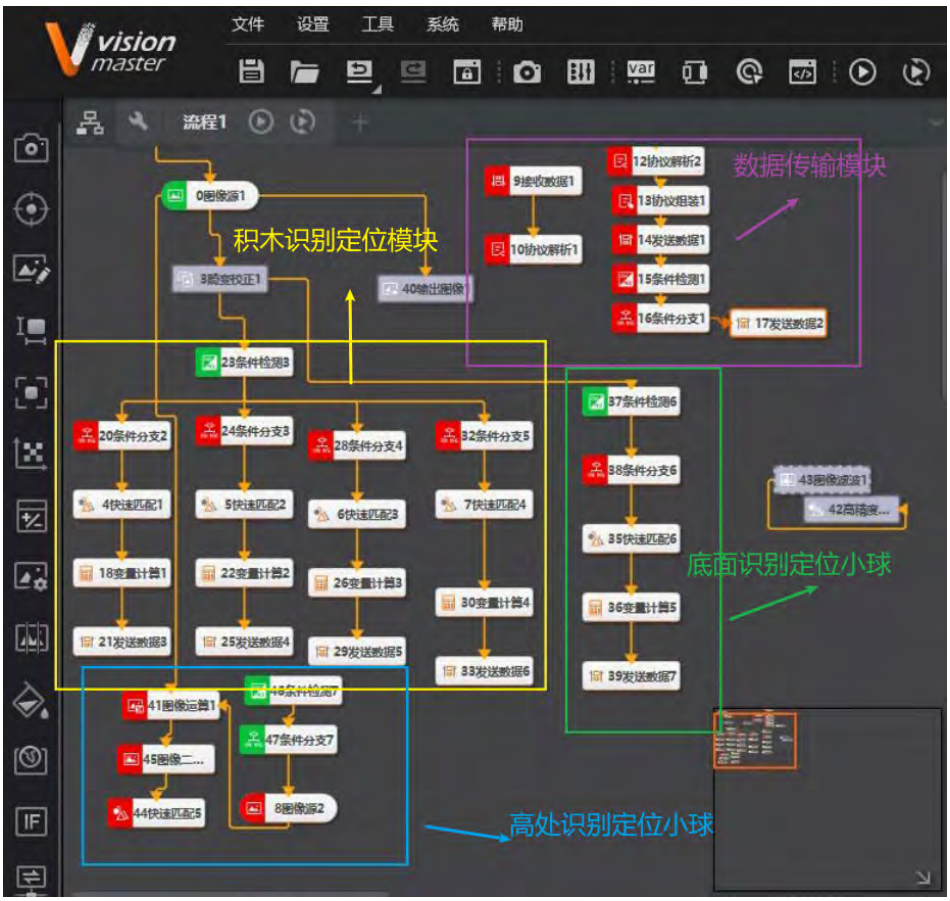


检测的匹配效果:



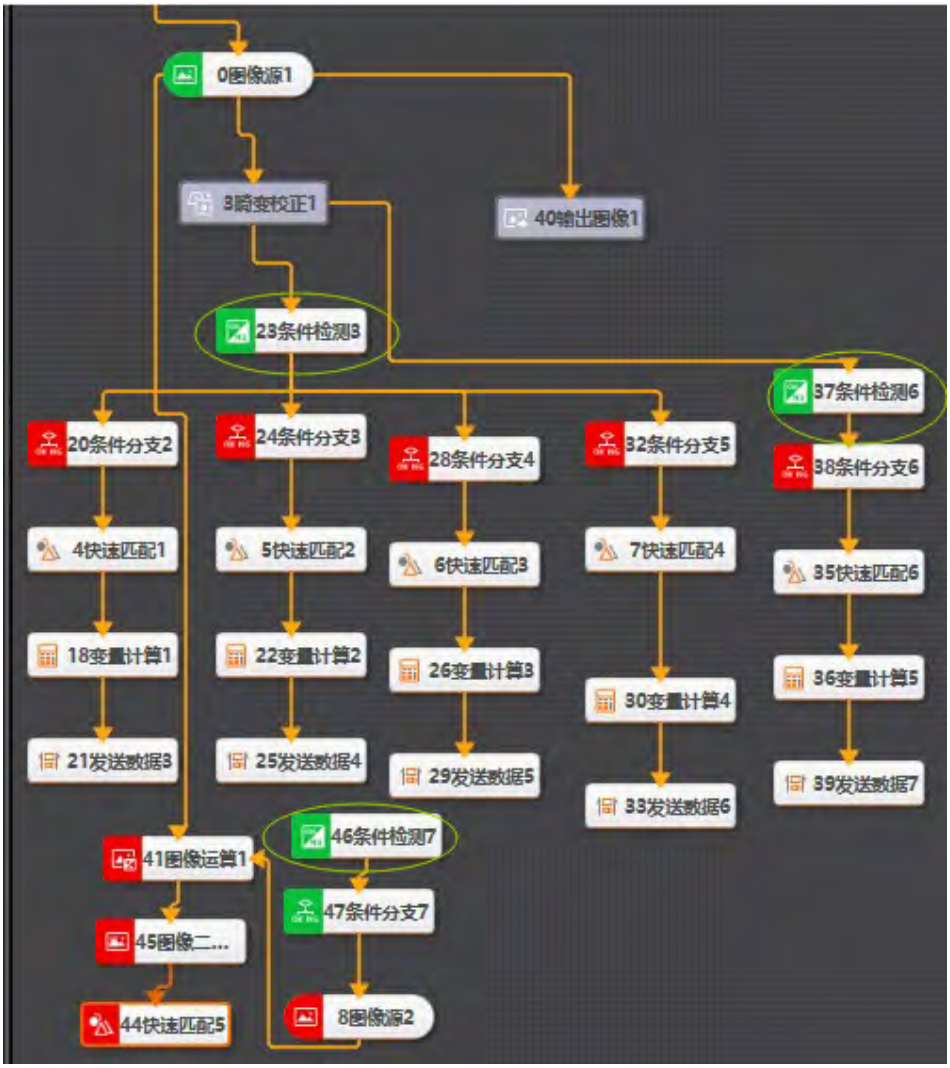


7.6 VisionMaster 总流程搭建

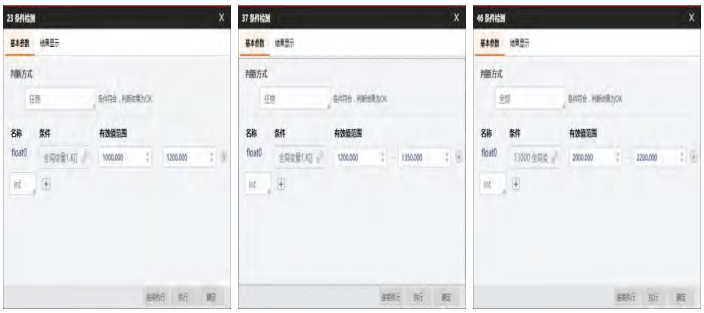


视觉机械臂积木抓取项目总流程包括 4 个模块

1. 数据传输模块
2. 积木识别定位模块
3. 高处识别定位小球模块
4. 低处识别定位小球模块



设置在 [23 条件检测 3],[37 条件检测 6],[46 条件检测 7] 中全局变量 K 的取值范围。

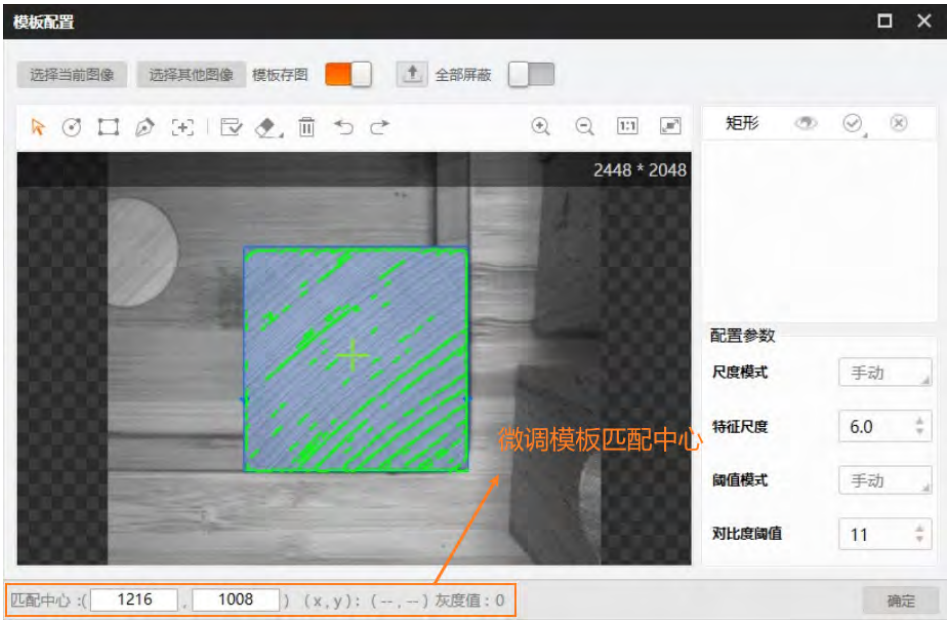


7.6.1 积木识别定位模块

创建 4 条执行路线，区别只有快速匹配的模板不同。

图 7.30: 4 类积木的 4 条执行路线

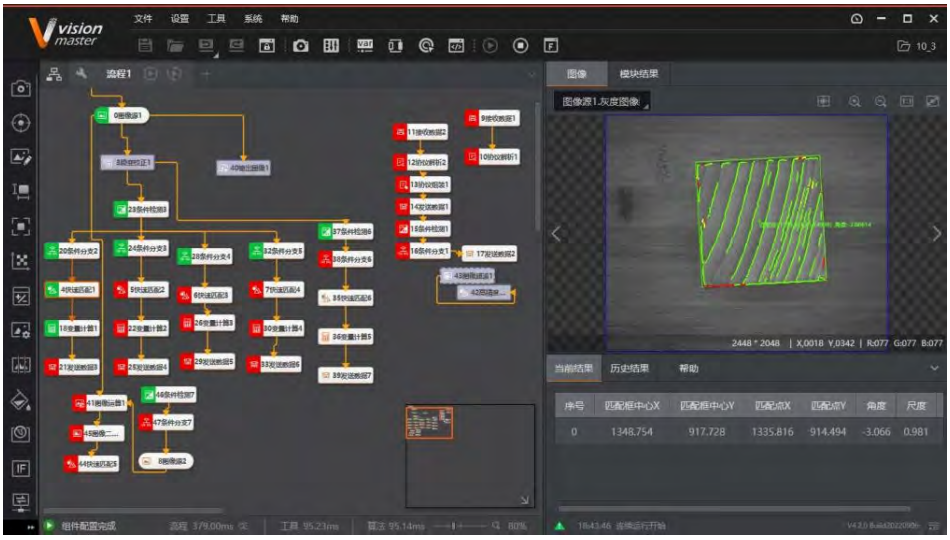
图 7.31: 4 个快速匹配分别存放了 4 种不同类型的积木轨道模板



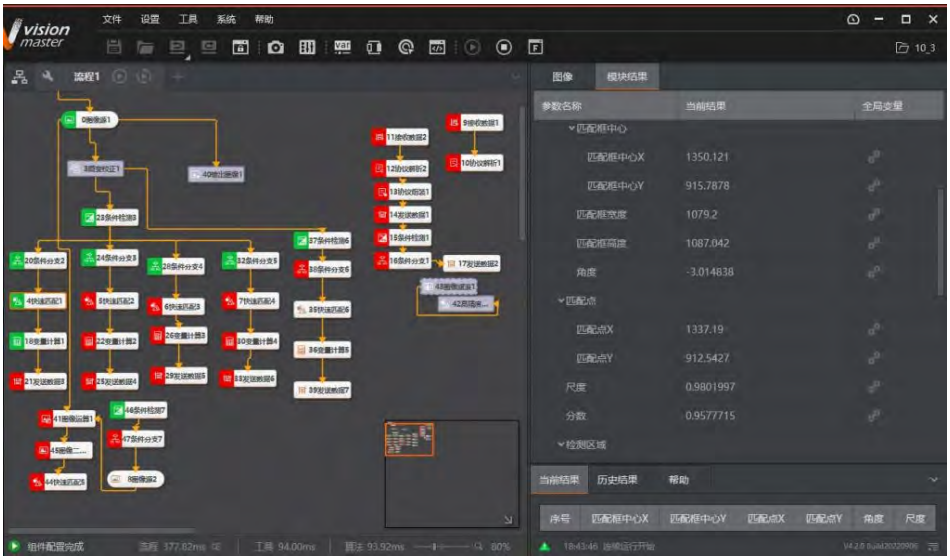
下面取最左边的执行路线为例，

此时 [23 条件检测 3] 木块已经满足条件，条件项状态的值为 1，在有效值范围之内，接着执行模块 [4 快速匹配 1]，

当 [4 快速匹配 1] 匹配到积木时



可以看到模块结果中得到了匹配框中心点的像素坐标，匹配框的角度，



然后将参数传到变量计算中，通过变量计算得出机械臂应该带抓取的积木位置。

输入变量 var0，var1 对应的 X，Y 坐标计算公式：

上式中全局变量 X，Y 是机械臂传输过来的实时坐标，0.04449687 是像素与实际距离之间的转换系数。

整个项目的机械臂动作流程为：识别抓取积木搭建—> 吸取小球放入积木轨道—> 将积木重新放于原位—> 识别定位抓取小球并放回至小球抓取位置—> 识别抓取积木搭建……

由此实现了流程的闭环控制，可以一直循环往复的运行下去。

在识别定位抓取小球我们分为两个步骤进行：高处识别定位小球模块和低处识别定位小球模块。

7.6.2 高处识别定位小球模块

对于搭建完成后散落在木板内的小球，由于相机的拍摄视野是有限的，想要一次性识别到木板上的所有小球必须将相机抬高获取更大的视野。

图 7.32: 高处识别定位小球模块

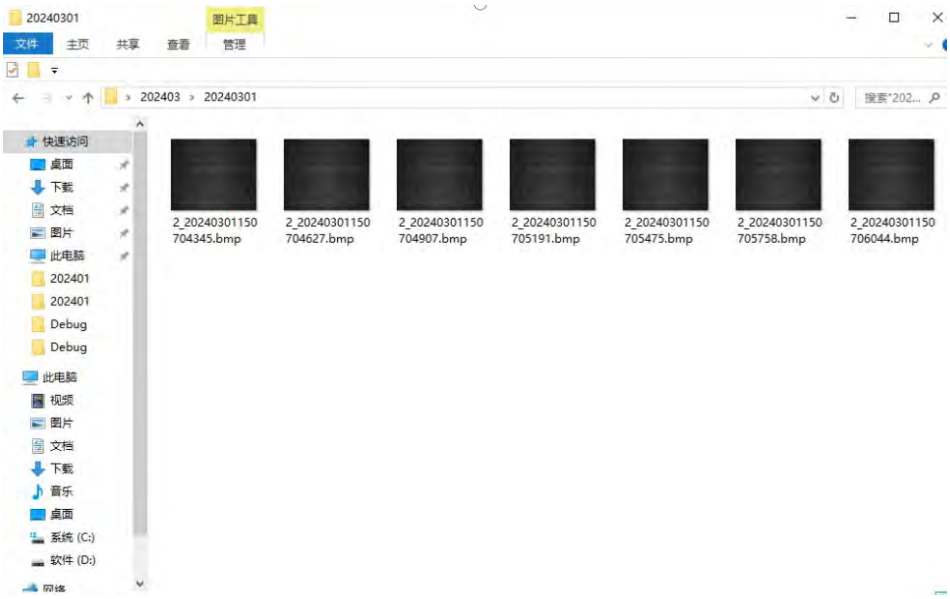
首先需要选定一个可以观察拍摄到所有小球的，固定高度位置，将机械臂移动到该位置，通过输出图像模块获取一张不含任何物体的背景图片。

在 [40 输出图像 1] 模块中，配置输入源为图像源 1 灰度图像，打开存图使能与生成目录，

之后打开保存原图，并设置图像的保存路径。

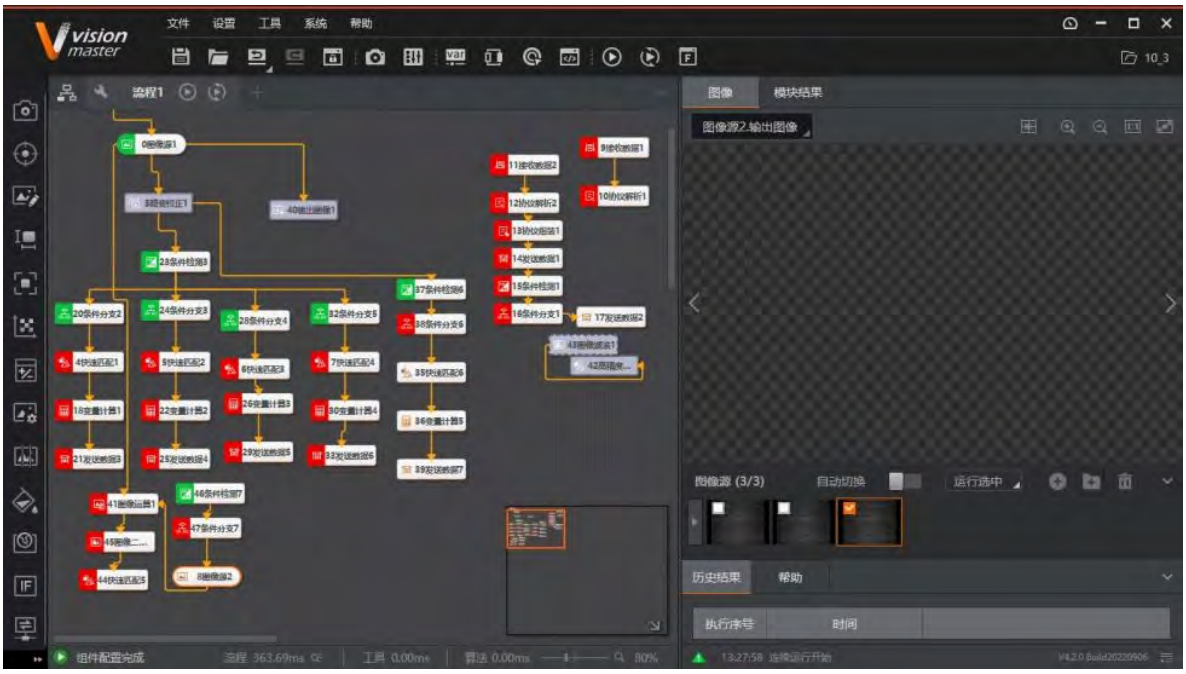
并选择输出使能，像素格式 RGB24

点击单次执行获取到背景图片



在 [8 图像源 2] 模块中图像源设置为本地图像，像素格式 MONO8，

在 VisionMaster 界面选择拍摄好的背景图，



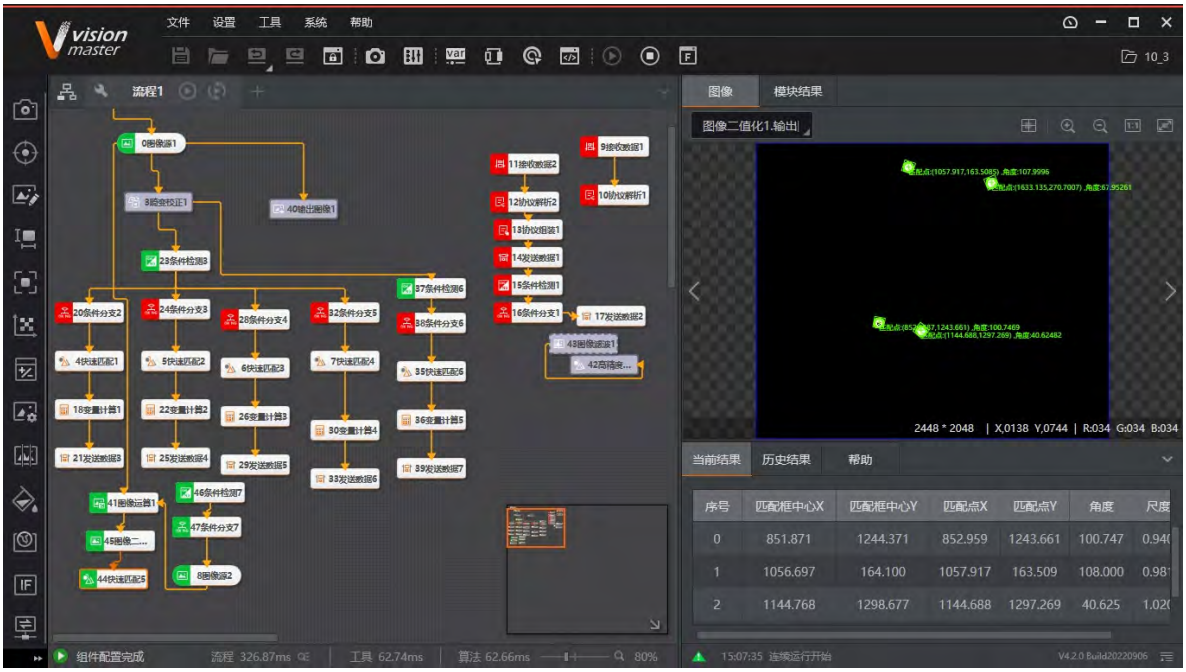
设置 [41 图像运算 1] 模块，将输入源 1 设置为图像源 1，将输入源 2 设置为图像源 2，

在图像运算参数设置为图像绝对差。

之后设置 [45 图像二值化] 的输入源为上文的图像运算的输出图像，

运行参数修改为硬阈值二值化，

此时放入小球，观察效果：



在此高处识别定位小球模块也讲解完了。

7.6.3 低处识别定位小球模块

本项目在高处粗略定位的基础上，结合低处进行识别，精准定位。

低处识别定位小球模块的识别方法与积木识别定位的方法几乎完全相同，在此仅对不同点进行配置。

在本项目中，只需要一个通用的模板即可精确识别定位到小球。

7.7 KUKA 运动控制搭建积木

有了以上准备工作就可以开始 KUKA 机械臂 KRL 程序的配置。

在本项目中 KRL 程序名为 LI_1018sec4.src，代码共计 5398 行。

代码的项目流程为：接收积木坐标-> 抓取搭建积木-> 抓取小球-> 将小球放入轨道-> 将积木放回原位-> 抓取散落的小球-> 放回小球-> 接受积木坐标……

形成闭环控制，下面以此流程逐个介绍：

1. 在代码的开头就是变量的初始化工作，在下图中列出项目中所有的变量。

```
7  ;FOLD Declaration
8  DECL  EKI_STATUS RET
9
10 ;ENDFOLD (Declaration)
11 DECL INT i, j, k, iu, num,ball_get_i, i_num,j_num,k_num,BallRest
12 DECL REAL Num1Start_Z, Num2Start_Z, Num3Start_Z, Num4Start_Z, Num5Start_Z, Num6Strat_Z, Num7Start_Z, Num8Start_Z, Num9St
13 DECL REAL Ball_StartX, Ball_StartY, Ball_StartZ, Threshold, Threshold_1
14 DECL REAL Ball_EndX, Ball_EndY, Ball_EndZ, scale_h,scale_low,z_high,z_low,Ball_HX,Ball_HY,Ball_HZ
15 DECL REAL Block_1x[24], Block_1y[24], Block_1z[24], ball_get_x[4],ball_get_y[4],Ball_locX[4],Ball_locY[4]
16 DECL REAL Block_2x[4], Block_2y[4], Block_2z[4]
17 DECL REAL Block_3x[4], Block_3y[4], Block_3z[4]
18 DECL REAL Block_4x[4], Block_4y[4], Block_4z[4]
19 DECL REAL Block_5x[4], Block_5y[4], Block_5z[4]
20 DECL REAL Block_6x[4], Block_6y[4], Block_6z[4]
21 DECL REAL Block_7x[4], Block_7y[4], Block_7z[4]
22 DECL REAL Block_8x[4], Block_8y[4], Block_8z[4]
23 DECL REAL Block_9x[4], Block_9y[4], Block_9z[4]
24 DECL REAL Block_10x[4], Block_10y[4], Block_10z[4]
25 DECL REAL Block_11x[6], Block_11y[6], Block_11z[6]
26 DECL REAL Block_12x[4], Block_12y[4], Block_12z[4]
27 DECL REAL Block_13x,Block_13y,Block_13z,down_hight,get_down_hight,limit_high
28 DECL REAL Left_Mountain_x_FL, Left_Mountain_y_FL, Left_Mountain_z_FL
29 DECL REAL DAA1,DAA2,DA41,ERROR_X,ERROR_Y
30 DECL REAL ACT_X,ACT_Y,q1,x_start,y_start_left,y_start_right,x_jianju,wait_sec
31 DECL POS STU
32 DECL INT Ball_Stu[4]
33 DECL REAL Ball_X[4],Ball_Y[4]
34 DECL INT num_1, num_2, num_3, num_4,num_5, num_6, num_7, num_8,num_9, num_10, num_11, num_12
35 DECL REAL TopX,TopY,DownX,DownY,BallDown
36 DECL INT A[5,6,6],A1[5,6,6]
37 DECL INT B[5,6,6]
38 DECL REAL Location_XStart, Location_YStart, Location_ZStart,class_wood,write_over,start_wood,theta,i1,j1,k1
39 DECL POS MY1
40 DECL POS STU1
41 DECL REAL p, p1, p2
42 DECL BOOL Ball_Flag[4]
43 DECL INT NUM_Ball, new
44 DECL INT Ball_St[4]
45 DECL REAL X_TOP,X_DOWN,Y_TOP,Y_DOWN
```

2. 接受积木坐标及旋转角度，i1，j1，k1 传入的是积木搭建要放置的位置编号，theta 对应的是角度信息，class_wood 为积木的种类编号，write_over 标志传输结束的截止信号。

```
346  $FLAG[1]=FALSE
347  RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
348   WHILE TRUE
349
350              WAIT FOR  ( $FLAG[1] )
351              number=1
352              RET=EKI_GetReal("XmlTransmit","Sensor/X1",i1)
353              RET=EKI_GetReal("XmlTransmit","Sensor/Y1",j1)
354              RET=EKI_GetReal("XmlTransmit","Sensor/Z1",k1)
355              RET=EKI_GetReal("XmlTransmit","Sensor/A1",theta)
356              RET=EKI_GetReal("XmlTransmit","Sensor/B1",class_wood)
357              RET=EKI_GetReal("XmlTransmit","Sensor/C1",write_over)
358
359              number=2
360              IF write_over>2 THEN
361                 RET=EKI_ClearBuffer("XmlTransmit","Sensor")
362                 $FLAG[1] = FALSE
363                 WAIT SEC 0.01
364              EXIT
365              ELSE
366                 i=i1
367                 j=j1
368                 k=k1
369                 A[i,j,k]=class_wood
370                 B[i,j,k]=theta
371              ENDIF
372              RET=EKI_ClearBuffer("XmlTransmit","Sensor")
373              $FLAG[1] = FALSE
374
375              WAIT SEC 0.01
376   ENDWHILE
377   WAIT SEC 1
```

```
i_1_j_1_k_1_theta_0_classwood_1_writeover_1_startwood_1_
i_1_j_1_k_2_theta_0_classwood_1_writeover_1_startwood_1_
i_1_j_1_k_3_theta_0_classwood_1_writeover_1_startwood_1_
i_1_j_2_k_3_theta_0_classwood_1_writeover_1_startwood_1_
i_1_j_3_k_3_theta_0_classwood_1_writeover_1_startwood_1_
i_1_j_4_k_3_theta_0_classwood_3_writeover_1_startwood_1_
i_1_j_5_k_3_theta_-90_classwood_11_writeover_1_startwood_1_
i_2_j_1_k_1_theta_0_classwood_1_writeover_1_startwood_1_
i_2_j_1_k_2_theta_0_classwood_1_writeover_1_startwood_1_
i_2_j_1_k_3_theta_0_classwood_1_writeover_1_startwood_1_
i_2_j_2_k_3_theta_0_classwood_1_writeover_1_startwood_1_
i_2_j_3_k_3_theta_0_classwood_12_writeover_1_startwood_1_
i_3_j_1_k_1_theta_0_classwood_1_writeover_1_startwood_1_
i_3_j_1_k_2_theta_90_classwood_3_writeover_1_startwood_1_
i_3_j_1_k_3_theta_0_classwood_6_writeover_1_startwood_1_
i_3_j_2_k_3_theta_0_classwood_12_writeover_1_startwood_1_
i_4_j_1_k_1_theta_-90_classwood_12_writeover_1_startwood_1_
i_4_j_1_k_1_theta_-90_classwood_12_writeover_3_startwood_1_
```

图 **7.33:** KUKA 机械臂控制柜所接收的信息格式

3. 抓取搭建积木总共分为 3 各部分，第一部分是运动到要抓取积木的位置，接收来自 VisionMaster 模板匹配的结果

```
448     ;insert your code here
449     FOR j = 1 TO 6 STEP 1
450        FOR k = 1 TO 6 STEP 1
451           IF A[i,j,k] == 0 THEN
452              CONTINUE
453           ENDIF
454           IF A[i,j,k] == 1 THEN
455              ;Get the Block Previewly.
456              A1[i,j,k] = 1
457              MY1.X = Block_1x[num_1]
458              MY1.Y = Block_1y[num_1]
459              MY1.Z = Block_1z[num_1] + 72.72
460              MY1.A = -180
461              MY1.B = 0
462              MY1.C = 180
463              PTP MY1
464              num = 0
465              WAIT SEC 0.02
466              RET=EKI_ClearBuffer("XmlTransmit","Sensor")
467              $FLAG[1] = FALSE
468              RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
469              WHILE iu < 2
470
471              IF num >= 2 THEN
472                 EXIT
473              ENDIF
474
475              RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",1111)
476              RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",98765111)
477              WAIT FOR  ( $FLAG[1] )
478              RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
479              RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",0)
480              number=1
481              RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
482              RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
483              RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
484              RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
485              RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
486              RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
487              RET=EKI_GetReal("XmlTransmit","Sensor/p",p)
488
```

将结果与实际位置进行比对，当误差小于阈值时进行积木抓取。

```
489     WAIT SEC wait_sec
490     number=2
491
492     IF ((ABS(DAA1 - DA1))<Threshold) AND (ABS(DAA2-DA2)<Threshold) AND (ABS(DA4)<0.04) THEN
493        iu = 3
494        ELSE
495     WAIT SEC 0.01
496     DAA1=DA1
497     DAA2=DA2
498     STU.X=DA1
499     STU.Y=DA2
500     STU.Z=Block_1z[num_1] + 72.72
501     STU.A=$POS_ACT.A-DA4
502     STU.B=0
503     STU.C=-180
504     DA41=DA4
505     PTP STU
506     WAIT SEC 0.01
507     ENDIF
508     RET=EKI_ClearBuffer("XmlTransmit","Sensor")
509     $FLAG[1] = FALSE
510     WAIT SEC 0.01
511     num = num + 1
512
513     ENDWHILE
```

第二部分，当机械臂运动到吸盘与积木表面接触，通过 $OUT[12] = TRUE,$OUT[10]

= TRUE,$OUT[9] = TRUE,$OUT[11] = TRUE 变量控制气泵打开，

判断 $IN[12]==TRUE,$IN[13]==TRUE,$IN[14]==TRUE,$IN[15]==TRUE, 即气压传感器的信号, 当四个吸盘都吸住积木时才抓起积木。

```
$OUT[12] = TRUE
$OUT[10] = TRUE
$OUT[9] = TRUE
$OUT[11] = TRUE
down_hight=0
get_down_hight=STU.Z
 WHILE TRUE
   IF ($IN[12]==TRUE) AND ($IN[13]==TRUE) AND ($IN[14]==TRUE) AND ($IN[15]==TRUE) THEN
      down_hight=0
      EXIT
   ELSE
      WAIT SEC 0.1
      down_hight=down_hight+0.05
      IF down_hight<limit_high THEN
      STU.Z=get_down_hight-down_hight
      PTP STU
      ENDIF
   ENDIF
ENDWHILE
```

第三部分根据程序预设的位置与角度搭建积木

```
RET=EKI_ClearBuffer("XmlTransmit","Sensor")
$FLAG[1] = FALSE
;Lay out Previewly
MY1.X = Location_XStart + (k - 1) * 48 + 15
MY1.Y = Location_YStart - (j - 1) * 48 - 15
MY1.Z = Location_ZStart + (i - 1) * 48 + 144
PTP MY1
;Lay out Previewly again
MY1.X = MY1.X - 2.5
MY1.Y = MY1.Y + 2.5
MY1.Z = MY1.Z - 90
PTP MY1
num_1 = num_1 + 1
;Lay out
MY1.X = Location_XStart + (k - 1) * 48
MY1.Y = Location_YStart - (j - 1) * 48
MY1.Z = Location_ZStart + (i - 1) * 48 + 0.5
PTP MY1
;Exhaust
$OUT[12] = FALSE
$OUT[10] = FALSE
$OUT[9] =  FALSE
$OUT[11] = FALSE
$OUT[3] = TRUE
$OUT[4] = TRUE
$OUT[5] = TRUE
$OUT[6] = TRUE
WAIT SEC 0.1
RET=EKI_ClearBuffer("XmlTransmit","Sensor")
$FLAG[1] = FALSE
```

4. 抓取小球

搭建完积木，就要放置小球在木质轨道上运动，首先就要在固定位置抓取小球，Ball_StartX,Ball_StartY,Ball_StartZ 就是小球所在位置，打开四个吸盘对应的气泵就能

吸取 4 个小球。

```
i1 = 5
j1 = 6
k1 = 6
Ball_EndX = MY1.X + 19
Ball_EndY = MY1.Y - 18
Ball_EndZ = MY1.Z - 160

MY1.Z = MY1.Z + 2 * 48
MY1.A = 0
MY1.B = 180
MY1.C = 0
PTP MY1
;Put the Ball drop out

MY1.X = Ball_StartX
MY1.Y = Ball_StartY
MY1.Z = Ball_StartZ + 48 * 4

PTP MY1
MY1.Z = Ball_StartZ
PTP MY1
$OUT[12] = TRUE
$OUT[10] = TRUE
$OUT[9] =  TRUE
$OUT[11] = TRUE
WAIT SEC 1
```

5. 将小球放入轨道

将小球移动至最上层积木处，设置 $OUT[9|10|11|12]=FALSE 关闭对应位置的气泵，并打开对应位置的电磁阀 $OUT[3|4|5|6]=TRUE, 将小球落下，旋转末端机械臂，重复此动作将所有小球送入积木轨道。

```
2622    MY1.Z = MY1.Z + 4 * 48
2623    PTP MY1
2624    MY1.X = Ball_EndX
2625    MY1.Y = Ball_EndY
2626    MY1.Z = Ball_EndZ + 48
2627    PTP MY1
2628    MY1.Z = MY1.Z - 20-6
2629    PTP MY1
2630    $OUT[11] = FALSE
2631    $OUT[5] = TRUE
2632    WAIT SEC 0.5
2633    MY1.X = MY1.X - 2.4
2634    MY1.Y = MY1.Y + 2.4
2635    MY1.A = -90
2636    MY1.B = 0
2637    MY1.C = -180
2638    PTP MY1
2639    $OUT[12] = FALSE
2640    $OUT[3] = TRUE
2641    WAIT SEC 0.5
2642    MY1.X = MY1.X - 1.3
2643    MY1.Y = MY1.Y - 2
2644    MY1.A = 0
2645    MY1.B = 0
2646    MY1.C = -180
2647    PTP MY1
2648    $OUT[10]= FALSE
2649    $OUT[4] = TRUE
2650    WAIT SEC 0.5
2651    MY1.X = MY1.X + 4
2652    MY1.Y = MY1.Y - 3.4
2653    MY1.A = 90
2654    MY1.B = 0
2655    MY1.C = 180
2656    PTP MY1
2657    $OUT[9] = FALSE
2658    $OUT[6] = TRUE
```

6. 将搭建好的积木放回原位

这个步骤与 3. 抓取搭建积木步骤相同，只是抓的位置与放的位置互换，不再赘述。

7. 抓取散落的小球

结合上文中的介绍，抓取散落的小球在 VisionMaster 中有两个模块：高处识别定

位小球模块与低处识别定位小球模块。相应的代码段也分为两个部分。

首先是高处识别定位小球模块，通过 Robot/Data/mypos/@e 将 e 的值 7000 发送给 VisionMaster 端，VisionMaster 通过全局变量，改变相机的曝光时间值。Robot/Data/mypos/@f 将 2100 的值发送给 VisionMaster 的全局变量 K，控制 VisionMaster 只执行高处识别定位小球模块。

```
4650        RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@e",7000)
4651        RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",2100)
4652
4653        WAIT SEC 0.3
4654           RET=EKI_ClearBuffer("XmlTransmit","Sensor")
4655           $FLAG[1] = FALSE
4656        RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",98765111)
4657        FOR i = 1 TO 4 STEP 1
4658           BallContinue:
4659           IF i > 4 THEN
4660             EXIT
4661            ENDIF
4662           IF Ball_Stu[i] == 1 THEN
4663              i = i + 1
4664              GOTO BallContinue
4665           ENDIF
4666
4667           WAIT FOR  ( $FLAG[1] )
4668           number=1
4669           RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
4670           RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
4671           RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
4672           RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
4673           RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
4674           RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
4675
4676           number=2
4677
4678           ball_get_x[i]=DA1
4679           ball_get_y[i]=DA2
4680
4681           RET=EKI_ClearBuffer("XmlTransmit","Sensor")
4682           $FLAG[1] = FALSE
4683
4684
4685        ENDFOR
```

此时上位机会将 [44 快速匹配 5] 匹配到的 4 个小球的匹配中心点位置先发送给 VisionMaster, 在通过 VisionMaster 的服务端发送到 KUKA 机械臂控制柜处。并通过 4669-4674 代码行的 RET=EKI_GetReal() 函数获取到 4 个小球的匹配中心点。机械臂移动到对应的低处识别位置，再进行低处识别定位小球模块的识别。

```
LI_1018sec4 ( )                                  RET(EKI_STATUS)
4686        RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
4687        RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@e",3500)
4688        ;j = BallRest + 1
4689
4690        STU1.A = 180
4691        STU1.B = 0
4692        STU1.C = -180
4693
4694        new = 1
4695        FOR i=1 TO 4 STEP 1
4696
4697           Ball2:
4698           IF i > 4 THEN
4699             EXIT
4700            ENDIF
4701           IF Ball_Stu[i] == 1 THEN
4702              i = i + 1
4703              GOTO Ball2
4704           ENDIF
4705
4706           WAIT SEC 0.01
4707           Ball_locX[i]=Ball_HX + scale_h*(-ball_get_y[i] + 1024)
4708           Ball_locY[i]=Ball_HY + scale_h*(-ball_get_x[i] + 1224)
4709           STU.X = Ball_locX[i]
4710           STU.Y = Ball_locY[i]
4711           STU.Z=z_low
4712           STU.A=180
4713           STU.B=0
4714           STU.C=-180
4715           ;PTP STU
4716           IF new == 1 THEN
4717              IF Ball_locY[i] >= 180 THEN
4718
4719                 STU1.X = Ball_locX[i]
4720                 STU1.Y = Ball_locY[i] - 10
4721                 STU1.Z = z_low + 20
4722                 CIRC STU1,STU
4723              ENDIF
4724              IF Ball_locY[i] <= -300 THEN
```

图 7.34: 低处识别定位小球代码

当机械臂运动到低处识别小球位置时，再对小球进行精确定位。识别步骤与识别积木相同，只有计算位置 Ball_X[i],Ball_Y[i] 上有所区别。

```
4789                 WAIT SEC 0.01
4790                 Ball_X[i]=$POS_ACT.X+scale_low*(-DA2 + 1024)
4791
4792                 Ball_Y[i]=$POS_ACT.Y+scale_low*(-DA1 + 1224)
```

之后就是应对各种突发情况的细节判断代码，例如小球没吸住，小球吸住之后在机械臂运动过程中意外掉落，等等……

LI_1018sec4 ( ) RET(EKI_STATUS)

```
4821
4822            RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",0)
4823        RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
4824        FOR i = 4 TO 1 STEP -1
4825           ;FOLD FENLEI
4826
4827           Ball3:
4828           IF i < 1 THEN
4829             EXIT
4830           ENDIF
4831           IF Ball_Stu[i] == 1 THEN
4832              i = i - 1
4833              GOTO Ball3
4834           ENDIF
4835                 IF (Ball_X[i] >= DownX) AND (Ball_Y[i]>= TopY) THEN
4836                       ;USE FOUR
4837                       IF i == 2 THEN
4838                       STU.X = Ball_X[i] - 18.5 - 3.4
4839                       STU.Y = Ball_Y[i] - 18.5 + 1.4
4840                       STU.Z = z_low
4841                       STU.A = -90
4842                       STU.B = 0
4843                       STU.C = -180
4844                       IF STU.Y > Y_TOP THEN
4845                          GOTO BallEnd
4846                       ENDIF
4847                       IF STU.X > X_TOP THEN
4848                          GOTO BallEnd
4849                       ENDIF
4850                       PTP STU
4851                       STU.Z = BallDown
4852                       PTP STU
4853                       $OUT[10] = TRUE
4854                       WAIT SEC 0.5
4855
4856                       GOTO BallEnd
4857                       ENDIF
4858                       IF i == 4 THEN
4859                          STU.X = Ball_X[i] - 18.5 - 4.3
4860                          STU.Y = Ball_Y[i] - 18.5 + 0.1
4861                          STU.Z = z_low
4862                          STU.A = 90
```

7.8 上位机搭建

7.9 展示成果

第 8 章 机器视觉简介

8.1 机器视觉——模拟人眼获取光学信号

1. 机器视觉发展史

机器视觉技术，就是指用计算机来模拟人的视觉功能，但这并不仅仅是人眼的简单延伸，更重要的是具有人脑的一部分功能——从客观事物的图像中提取信息，进行处理并加以理解，最终用于实际检测、测量和控制。

机器视觉技术是计算机学科的一个重要分支，自起步发展至今，其功能以及应用范围随着工业自动化的发展逐渐完善和推广。

20 世纪 50 年代开始研究二维图像的统计模式识别。

60 年代美国学者 Roberts 开始进行多面体积木世界视觉图像的研究。

70 年代中，MIT 人工智能实验室正式开设“机器视觉”课程。

80 年代开始，开始了全球性的研究热潮，机器视觉获得了蓬勃发展，新概念、新理论不断涌现。

现在，机器视觉仍然是一个非常活跃的研究领域，与之相关的学科涉及：图像处理、计算机图形学、模式识别、人工智能、人工神经元网络等。

机器视觉在中国的发展历史：

1990 年以前，仅仅在大学和研究所中有一些研究图像处理和模式识别的实验室。

20 世纪 90 年代初，一些来自这些研究机构的工程师成立了他们自己的视觉公司，开发了第一代图像处理产品，例如基于 ISA 总线的灰度级图像采集卡，和一些简单的图像处理软件库，他们的产品在大学的实验室和一些工业场合得到了应用，人们能够做一些基本的图像处理和分析工作。

1990-1998 年为初级阶段。期间真正的机器视觉系统市场销售额微乎其微。主要的国际机器视觉厂商还没有进入中国市场。自从 1998 年，越来越多的电子和半导体工厂，包括香港和台湾投资的工厂，落户广东和上海。带有机器视觉的整套的生产线和高级设备被引入中国。

1998-2002 年定义为机器视觉概念引入期。在此阶段，许多著名视觉设备供应商，诸如：Cognex, DVT, CCS, Data Translation, Matrix, Coreco, 开始接触中国市场寻求本地合作伙伴，但符合要求的本地合作伙伴寥若晨星。例如，北京和利时电机技术有限公司曾经被五家外国公司选做主要代理商或解决方案提供商。

从 2002 年至今，我们称之为机器视觉发展期，中国机器视觉呈快速增长趋势。

机器视觉发展现状： 机器视觉形成几个重要研究分支：一、目标制导的图像处理；二、图像处理和分析的并行算法；三、从二维图像提取三维信息；四、序列图像分析和运动参量求值；五、视觉知识的表示；六、视觉系统的知识库等。

当前，机器视觉的应用已经超越了其传统的检验领域，向着更深层、更为多样化的领域扩展。与此同时，机器视觉产品的采购量也在节节攀升。

国内国外客观比较上看我国机器视觉行业的起步比较晚，集中度也不是很高，最开始主要是代理国外品牌。近几年，很多的经销商开始自主开发产品，但在行业分布、渠道分销以及成熟的自动化产品等方面还是和国外有一定差距，市场也远远没有饱和。机器视觉企业大体可以分为层开发厂商、二次开发厂商和产品代理商。国内机器视觉企业主要为机器视觉产品代理商和系统二次开发厂商。我国正处于由劳动密集型向技术密集型转型的时期，对提高生成效率、降低人工成本的机器视觉方案有着旺盛的需求，因此正成为全球机器视觉技术发展最为活跃的地区之一。长三角和珠三角作为国际电子和半导体技术的转移地，同时成为机器视觉技术的聚集地。许多具有国际先进水平的机器视觉系统进入中国，国内的机器视觉企业在与国际机器视觉企业的良性竞争中不断茁壮成长，许多大学和研究所也都在致力于机器视觉技术的研究。近些年国内机器视觉相关行业的企业海康，大华，奥比中光，商汤，矩子等不断成长壮大。

2. **机器视觉——模拟人眼获取光学信号**

人眼是复杂的视觉器官，能够感知并解释外界的光学信号，使我们能够看到这个世界。以下是人眼如何看到世界的基本过程：

(1) 光的折射和聚焦：当光线通过空气、水或其他透明介质时，会发生折射，即光线的方向改变。光线通过角膜、水晶体和玻璃体等眼睛的透明组织时，也会发生折射。这个过程使得光线能够聚焦在视网膜上。

(2) 视网膜的感光细胞：视网膜是眼睛的内层，包含两类主要的感光细胞：锥状细胞和杆状细胞。锥状细胞对颜色敏感，而杆状细胞对低光条件更为敏感。这些细胞将光信号转化为神经信号，并传递到视神经。

(3) 视神经传递：视神经是由视网膜到大脑的神经通道，将光信号传递到大脑的视觉中枢。在视觉中枢，光信号经过复杂的处理和解释，形成我们感知为图像的信息。

(4) 大脑的视觉皮层：视觉信息最终到达大脑的视觉皮层，该区域对图像进行高级处理，包括边缘检测、形状识别、颜色分辨等。这些处理过程使我们能够看到和理解复杂的视觉场景。

(5) 视觉感知：最终，大脑将处理过的信息解释为我们所看到的世界。我们的感知过程涉及对形状、颜色、运动和深度等多个方面的集成。

(6) 视觉反馈：视觉系统还包括与其他感官系统的交互，以及大脑对感知信息的认知和情感反馈。这使我们能够更全面地理解和体验所看到的世界。

整个过程是高度复杂而协调的，涉及多个层次的信息处理和感知。人眼通过这

一生物学过程使我们能够感知并理解世界，为我们的日常生活提供了丰富的视觉体验。

我们希望建立一个综合系统，模拟人眼和大脑的信号处理过程。这一系统将整合视觉传感器、机械臂、机械臂末端执行元件以及控制系统，以实现更全面的感知和执行功能。通过模仿人眼的视觉捕捉、感知和大脑的信号处理方式，构建一个具备高级认知和决策能力的系统，能够更自主地理解并与环境互动。这种集成系统有望应用于自动化、智能机器人、工业生产等领域，为现代科技带来更广泛的应用和发展。

8.2 视觉机械臂的应用场景

视觉机械臂在各行各业都有广泛的应用，以下是一些常见的应用场景例子：

工业自动化：视觉机械臂被广泛用于制造业，用于自动化生产线上的零部件组装、定位和质量检测。视觉系统能够提供高精度的定位和识别功能，使机械臂能够有效地执行各种任务。

仓储和物流：在仓储和物流领域，视觉机械臂可用于自动化货物的装卸、分拣和包装。视觉系统可以识别不同形状和大小的物体，使机械臂能够灵活地处理不同的物流任务。

医疗手术：在医疗领域，视觉机械臂被应用于手术机器人系统。这些系统可以通过视觉引导实现高精度的手术操作，提高手术的精准性和安全性。

农业：在农业自动化中，视觉机械臂可用于种植、收割和检测农作物。通过视觉系统，机械臂能够识别植物的生长状态、成熟度和病害，从而进行智能化的农业操作。

食品加工：在食品生产中，视觉机械臂可用于食品的分拣、包装和检测。视觉系统能够检测食品的外观、大小和质量，确保产品符合标准。

电子组装：在电子制造领域，视觉机械臂可用于电子元件的组装和焊接。通过视觉引导，机械臂可以准确地识别和定位微小的电子部件。

研究和实验室：在科研领域，视觉机械臂被广泛用于实验室操作，如样品处理、实验设备操作等。视觉系统可以提供对实验环境的实时感知和控制。

这些应用场景突显了视觉机械臂在提高自动化水平、提升生产效率和解决复杂任务中的关键作用。

在本书中，我们以趣味性和实用性为出发点，选择了一个引人入胜的课题研究，即控制机械臂进行积木抓取和搭建的应用场景。通过这个研究，读者将有机会深入了解机械臂控制、机器视觉以及实际应用的交叉领域，既能够享受学习的乐趣，又能够获得解决实际问题的能力。

图 **8.1:** ElonMusk 的 Optimus 机器人搭建积木

8.3 摄像机几何

首先，让我们详细介绍一下传感器元件——相机。作为外部信息获取的核心组成部分，相机在系统中扮演着类似于人眼的角色，负责捕捉并记录周围环境的图像信息。

常见的相机可以根据其配置和工作原理进行分类，其中包括但不限于：

单目相机（单摄像头）：单目相机只包含一个摄像头，通过单一透镜捕捉场景。它广泛应用于一般的摄影、视频录制、监控系统等领域。

图 **8.2:** 海康威视 500 万像素单目相机

双目相机（双摄像头）：双目相机包含两个摄像头，通常排列在固定的距离上，以模拟人类双眼视觉。通过测量双目之间的视差，双目相机可以实现深度感知，用于三维重建和距离测量。

立体相机：类似于双目相机，立体相机也是用于产生深度信息的一种摄像系统。它通过两个或多个摄像头捕捉同一场景，利用视差和三角法则计算物体的深度。

红外相机：红外相机可以感测红外辐射，通常应用于夜视、热成像和安防监控等领域。红外相机可以捕捉在可见光范围之外的信息。

全景相机：全景相机能够捕捉全景图像，提供更广阔的视野。这种相机通常用于虚拟现实、全景照片和实时监控等领域。

运动相机：运动相机设计用于捕捉高速运动场景，例如运动员比赛、高速拍摄和运动分析。

深度相机：深度相机能够感知场景中物体的距离，通常通过结合红外或激光技术，例如 Time-of-Flight（TOF）或结构光。

图 8.3: 深度相机

无人机相机：专门设计用于安装在无人机上的相机，以进行航拍、地理测绘、农业监测等应用。

这些相机类型在不同的应用领域发挥着重要的作用，满足了各种不同的拍摄和感测需求。

在本项目中，考虑到成本和科研的因素，我们选择采用单目相机。相机镜头通常可分为两种主要模型，即针孔模型和鱼眼模型。我们采用较为经典的针孔镜头模型，下面介绍针孔模型的成像原理。

（针孔模型：针孔模型是一种简单而经典的相机模型，基于针孔成像原理。它假设在光学系统中，光线通过一个非常小的孔洞时，会形成图像。这种模型通常用于一般的摄影和传统相机系统。鱼眼模型：鱼眼模型是一种设计用于提供极大视场角的相机镜头，相对于针孔模型而言，具有更大的视场角，但也引入了一些畸变，特别是图像边缘的畸变较为显著。）

8.4 小孔成像模型

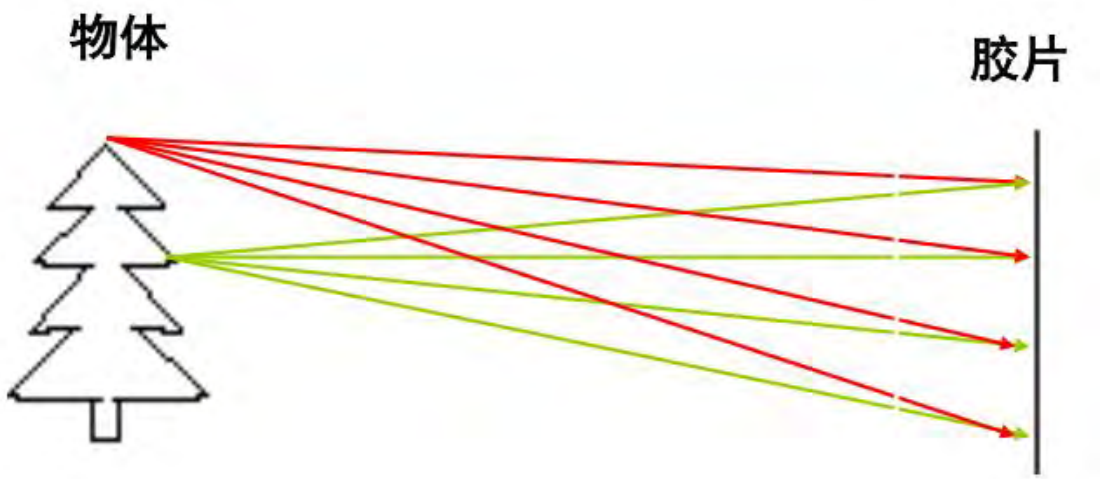


图 8.4: 直接将胶片置于物体前

我们了解到光是沿着直线传播的。在图9.4中，物体上的一点可以向各个方向反射或折射光线，而这些光线最终会在胶片上的每一个位置交汇。例如，树的顶部和中部的某一点反射的光线都可能到达胶片上的相同位置，导致光线叠加并形成混合的成像。这种现象使得成像失真，因为胶片上的每一点都受到来自不同方向的光线的影响。

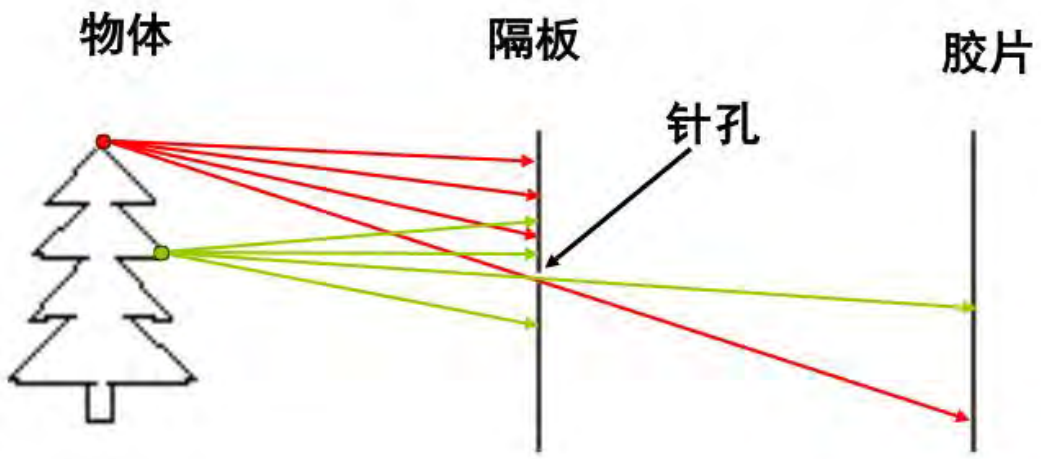


图 8.5: 在物体和胶片之间添加带有针孔的隔板

如上图9.5所示在物体和胶片之间加入带有针孔的隔板后，我们对光的传播路径进行分析。

观察到树顶上的光点只有通过隔板上的针孔才能射到胶片上。在极端情况下，如

果每个方向上的针孔只允许通过一条光线，那么物体上的每一点和胶片上的对应点将形成一一映射。

通过简单的分析，我们可以得知，在这种条件下形成的图像将是倒立的。

这种装置模拟了针孔相机，通过限制光线的传播路径，使得物体的光影在胶片上呈现出倒立的投影。

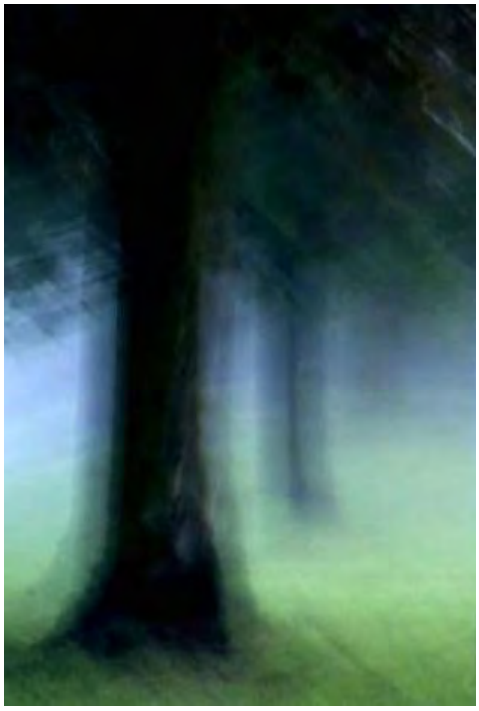


图 8.6: 光圈增大成像会变模糊

（设想一下，如果隔板上的针孔稍微扩大，物体上的一点通过针孔会到达胶片上相邻的一片区域。在这种情况下，图像变得模糊了，因为光线的传播路径更为广泛。然而，由于孔的扩大，光线经过的区域增多，胶片上呈现的像会更为明亮。这种模糊效应和亮度增加是光通过扩大孔径时常见的现象，但同时也导致了图像的失真和模糊。）

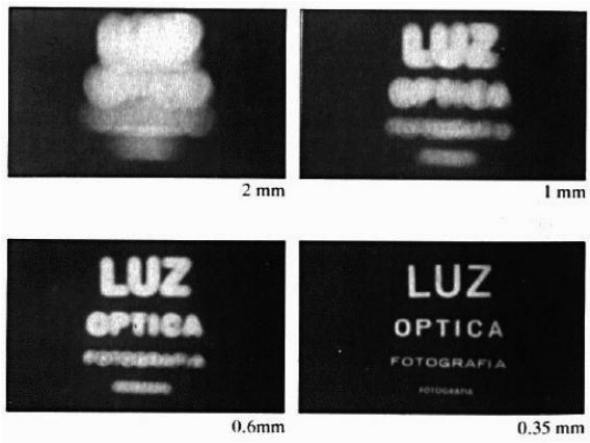


图 8.7: 光圈直径逐渐缩小，成像更加清晰，亮度变暗

为了解决光圈直径逐渐缩小导致成像清晰度降低和亮度减弱的问题，我们可以考虑增加透镜，使得同一个点的光线在成像平面上汇聚在一起，从而提高照片的亮度。

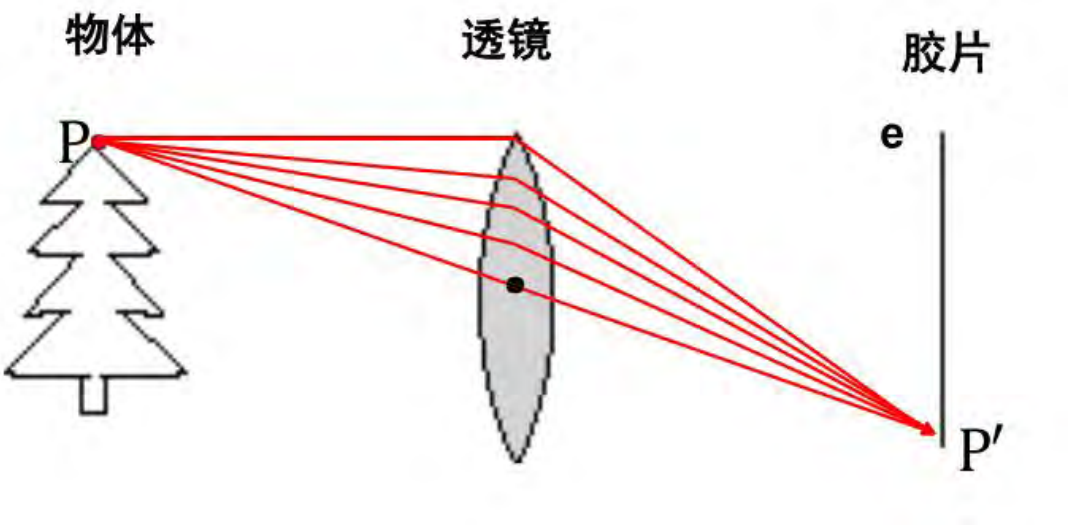


图 **8.8:** 增加透镜

透镜的引入会产生径向畸变，图像像素点以畸变中心为中心点，沿着径向产生的位置偏差，常见的有枕形畸变和桶形畸变。其产生原因是光线在远离光线中心的地方比靠近中心更加弯曲。

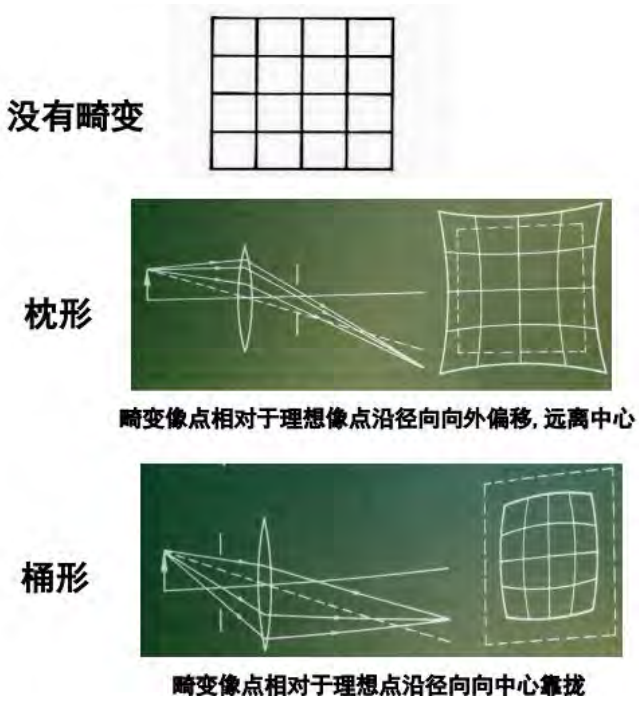


图 **8.9:** 图像的畸变类型

图 **8.10:** 桶形畸变

8.5 相机参数的定义

通过小孔成像模型，我们可以对相机的各个参数进行定义，如图9.10所示：

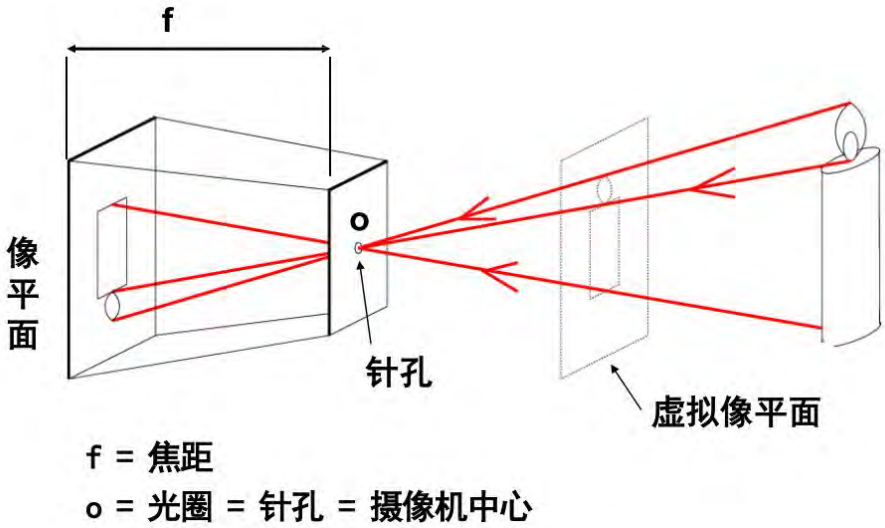


图 **8.11:** 相机参数定义

我们将像平面与针孔之间的距离定义为焦距 f，针孔 o 的大小即为光圈的大小，其中心为相机坐标系的中心。由此建立坐标系如下图9.11所示：

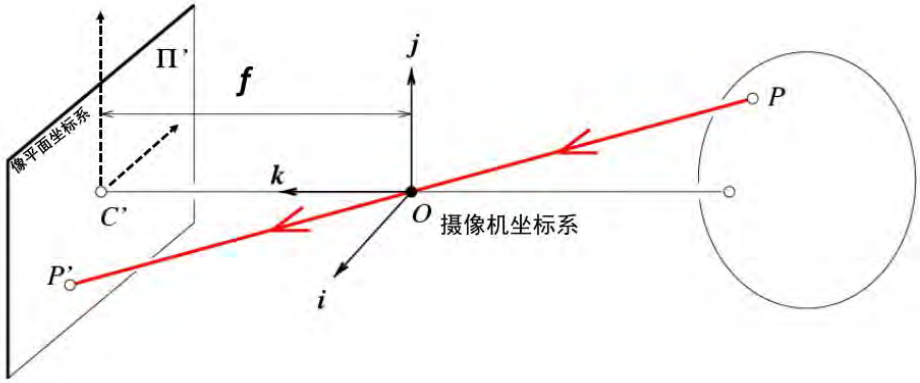
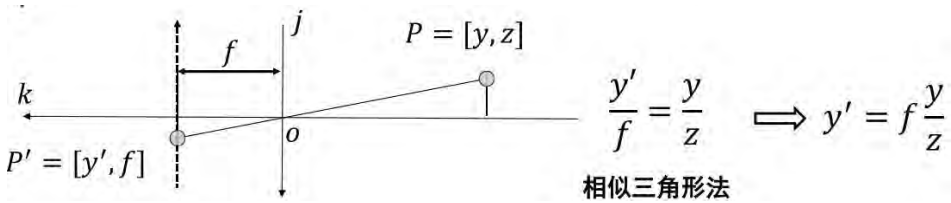


图 8.12: 摄像机模型

由此得到二维的像平面坐标系和三维的摄像机坐标系，于是根据相似三角形法则可以得到到点的映射关系 (i,j,k 依次对应 x,y,z):



对 x 和 x' 的推到流程也是一样，于是我们得到:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \qquad \begin{cases} x' = f\dfrac{x}{z} \\ y' = f\dfrac{y}{z} \end{cases}$$

由于实际安装环境的影响，像平面坐标系的中心会和理论上存在一个偏置量，如下图8.13所示:

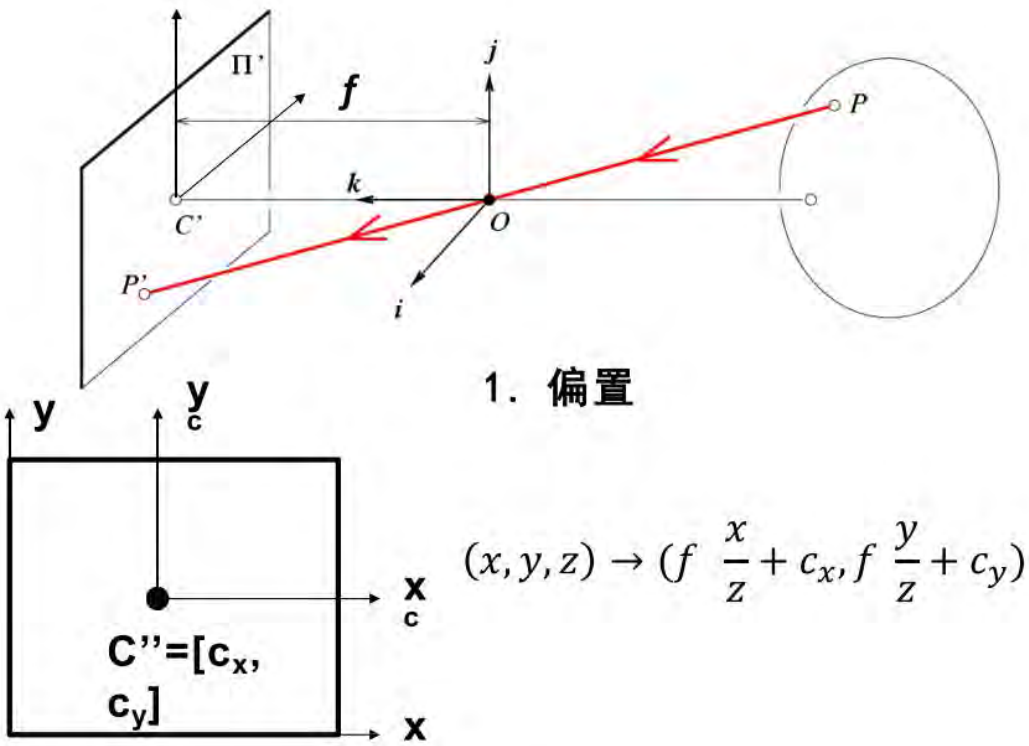


图 8.13: 偏置

像平面坐标系的单位是像素，物体在摄像机坐标系上的单位是米，两者之间存在一个单位转换的系数:

$$(x,y,z) \rightarrow (f\ k\frac{x}{z} + c_x, f\,l\frac{y}{z} + c_y)$$

由于 fk，fl 都是常数，不会改变，我们可以将其合并为 α,β:

$$P = (x,y,z) \rightarrow P' = \left(\alpha\frac{x}{z} + c_x, \beta\frac{y}{z} + c_y\right)$$

为了运算方便我们引入齐次坐标：

E → H

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

图像点的齐次坐标 **空间点的齐次坐标**

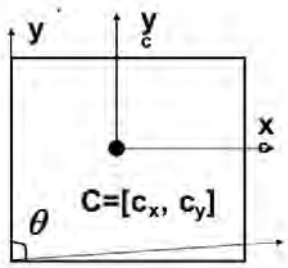
H → E

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w) \qquad \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

那么 P 和 P' 点的齐次坐标点 P_h 和 P'_h 可以表示为:

$$P_h' = \begin{bmatrix} \alpha x + c_x z \\ \beta y + c_y z \\ z \end{bmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{P_h}$$

再将像平面偏斜的影响考虑进去可以得到方程:



$$P' = \begin{bmatrix} \alpha & -\alpha cot\theta & c_x & 0 \\ 0 & \beta/sin\theta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$= K[I \quad 0]P$$

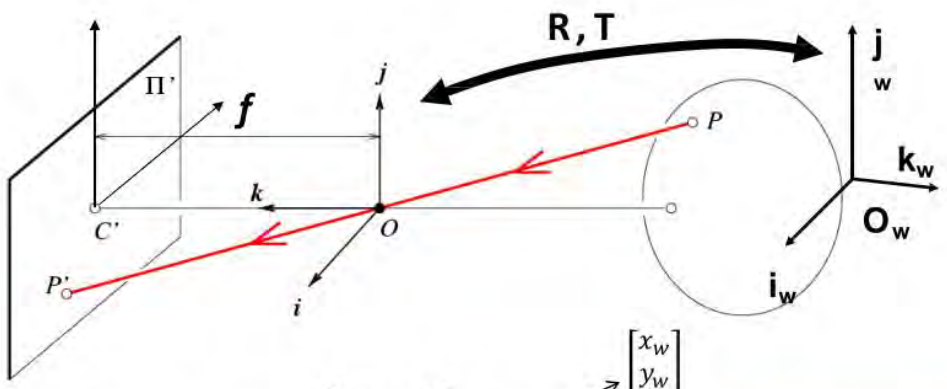
其中 K 为摄像机内参矩阵, 如下所示:

$$K=\begin{bmatrix}\alpha & -\alpha cot\theta & c_x\\ 0 & \dfrac{\beta}{sin\theta} & c_y\\ 0 & 0 & 1\end{bmatrix}$$

由矩阵的未知数数量可知，K 具有 5 个自由度。

8.6 摄像机外参

在实际应用中，我们往往难以直接获取物体上某一点在摄像机坐标系中的坐标（因为摄像机坐标系的原点位于相机内部）。因此，为了描述空间物体的三维信息，我们引入世界坐标系。



齐次坐标系: $P=\begin{bmatrix}R & T\\ 0 & 1\end{bmatrix}\boldsymbol{P}_w$ $\begin{bmatrix}x_w\\ y_w\\ z_w\\ 1\end{bmatrix}$ 世界坐标系到摄像机坐标系转换!

根据线性代数的知识，世界坐标系与摄像机坐标系之间相差了一个旋转平移矩阵 $[R,T]$，我们将其称为外参矩阵，如下所示:

外参数矩阵

$$P'=K[I\quad 0]P=K[I\quad 0]\begin{bmatrix}R & T\\ 0 & 1\end{bmatrix}P_w=K\boxed{[R\quad T]}P_w=MP_w$$

结合上述的推导内外参矩阵，我们得到了完整的摄像机模型:

易知旋转矩阵具有 3 个自由度，平移矩阵也具有 3 个自由度，结合内参矩阵的 5 个自由度，整个摄像机模型共 11 个自由度。我们所知的相机标定的过程就是求解这 11 个自由度的过程。

内参矩阵　　外参矩阵

$$P' = K[I \quad 0]P = K[I \quad 0]\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} P_w = K\boxed{[R \quad T]}P_w = MP_w$$

完整的摄像机模型！　　投影矩阵

8.7 摄像机标定

在前文的基础上，我们了解到摄像机标定的目标是求解摄像机的内参矩阵和外参矩阵。这些矩阵在计算机视觉中扮演着关键的角色，因为它们描述了三维世界与二维像素之间的映射关系。

相机成像系统中，共包含四个坐标系：世界坐标系、相机坐标系、图像坐标系、像素坐标系。上文介绍了世界坐标系、相机坐标系、像素坐标系之间的转换，将图像坐标系与像素坐标系之间的比例关系用 α,β 来表示，现在我们将其写出来：

$$Z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \underbrace{\underbrace{\begin{pmatrix} \frac{1}{dX} & -\frac{cot\theta}{dX} & u_0 \\ 0 & \frac{1}{dYsin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix}}_{\text{仿射变换}} \underbrace{\begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{\text{透视投影}}}_{\text{内参矩阵}} \underbrace{\begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}}_{\text{刚体变换}} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

外参矩阵

其中，(U,V,W) 为在世界坐标系下一点的物理坐标，(u,v) 为该点对应的在像素坐标系下的像素坐标，Z 为尺度因子。我们将内参矩阵合并:

$$\begin{pmatrix} \frac{1}{dX} & -\frac{\cot\theta}{dX} & u_0 \\ 0 & \frac{1}{dY\sin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} \frac{f}{dX} & -\frac{f\cot\theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY\sin\theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

其中，f 为像距，dX,dY 分别表示 X,Y 方向上的一个像素在相机感光板上的物理长度（即一个像素在感光板上是多少毫米），u_0,v_0 分别表示相机感光板中心在像素坐标系下的坐标，θ 表示感光板的横边和纵边之间的角度（90^0 表示无误差）。

我们将矩阵：$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$ 称为相机的外参矩阵，外参矩阵取决于相机坐标系和世界坐标系的相对位置，R 表示旋转矩阵，T 表示平移矢量。

即单点无畸变的相机成像模型如下：

$$Z\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{f}{dX} & -\frac{f\cot\theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY\sin\theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix}$$

我们再简单介绍一下畸变与畸变矫正, 畸变模型包括径向畸变和切向畸变。

径向畸变公式（3 阶）如下：

$$\hat{x} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$\hat{y} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

切向畸变公式如下：

$$\hat{x} = x + (2p_1 y + p_2(r^2 + 2x^2))$$
$$\hat{y} = y + (p_1(r^2 + 2y^2) + 2p_2 x)$$

其中，(x, y)，$(\hat{x}, \hat{y})$ 分别为理想的无畸变的归一化的图像坐标、畸变后的归一化图像坐标, r 为图像像素点到图像中心点的距离，即 $r^2 = x^2 + y^2$。

相机标定的第二个目的就是获得相机的畸变参数，如上式中的 k_1,k_2,k_3,p_1,p_2 等，进而对拍摄的图片进行去畸变处理。

8.8 常见标定法简介

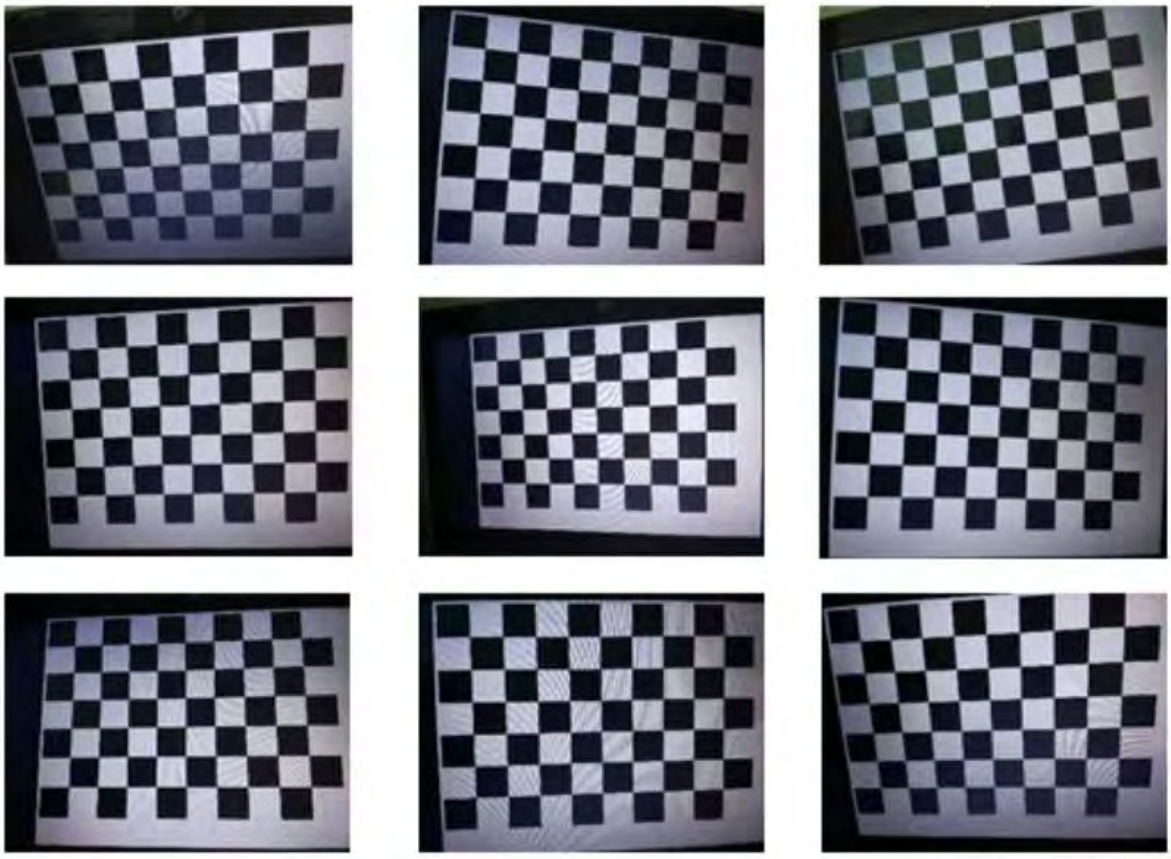


图 8.14: 棋盘格

常见标定法利用如上图所示的棋盘格标定板，在得到一张标定板的图像之后，可以利用相应的图像检测算法得到每一个角点的像素坐标 (u, v)。

常见标定法将世界坐标系固定于棋盘格上，则棋盘格上任一点的物理坐标 $W = 0$，由于标定板的世界坐标系是人为事先定义好的，标定板上每一个格子的大小是已知的，我们可以计算得到每一个角点在世界坐标系下的物理坐标 $(U, V, W = 0)$。

我们将利用这些信息：每一个角点的像素坐标 (u, v) 、每一个角点在世界坐标系下的物理坐标 $(U, V, W = 0)$，来进行相机的标定，获得相机的内外参矩阵、畸变参数。

常见标定法标定相机的内外参数的思路如下：

(1) 求解内参矩阵与外参矩阵的积；

(2) 求解内参矩阵；

(3) 求解外参矩阵。

8.8.1 求解内参矩阵与外参矩阵的积

将世界坐标系固定于棋盘格上，则棋盘格上任一点的物理坐标 $W=0$，因此，原单点无畸变的成像模型可以化为下式。其中，R_1，R_2 为旋转矩阵 R 的前两列。为了简便，将内参矩阵记为 A。

$$Z\begin{pmatrix}u\\v\\1\end{pmatrix}=\begin{pmatrix}\frac{f}{dX} & -\frac{f\cot\theta}{dX} & u_0\\0 & \frac{f}{dY\sin\theta} & v_0\\0 & 0 & 1\end{pmatrix}\begin{pmatrix}R1 & R2 & T\end{pmatrix}\begin{pmatrix}U\\V\\1\end{pmatrix}=A\begin{pmatrix}R1 & R2 & T\end{pmatrix}\begin{pmatrix}U\\V\\1\end{pmatrix}$$

我们对于上式做一定的说明。对于不同的图片，内参矩阵 A 为定值；对于同一张图片，内参矩阵 A，外参矩阵 (R_1,R_2,T) 为定值；对于同一张图片上的单点，内参矩阵 A，外参矩阵 (R_1,R_2,T)，尺度因子 Z 为定值。

我们将 $A(R_1,R_2,T)$ 记为矩阵 H，H 即为内参矩阵和外参矩阵的积，记矩阵 H 的三列为 (H_1,H_2,H_3)，则有：

$$\begin{pmatrix}u\\v\\1\end{pmatrix}=\frac{1}{Z}H\begin{pmatrix}U\\V\\1\end{pmatrix}=\frac{1}{Z}\begin{bmatrix}H_{11} & H_{12} & H_{13}\\H_{12} & H_{22} & H_{32}\\H_{31} & H_{32} & H_{33}\end{bmatrix}\begin{pmatrix}U\\V\\1\end{pmatrix}$$

利用上式，消去尺度因子 Z，可得：

$$u=\frac{H_{11}U+H_{12}V+H_{13}}{H_{31}U+H_{32}V+H_{33}}$$
$$v=\frac{H_{21}U+H_{22}V+H_{23}}{H_{31}U+H_{32}V+H_{33}}$$

此时，尺度因子 Z 已经被消去，因此上式对于同一张图片上所有的角点均成立。(u,v) 是像素坐标系下的标定板角点的坐标，(U,V) 是世界坐标系下的标定板角点的坐标。通过图像识别算法，我们可以得到标定板角点的像素坐标 (u,v)，又由于标定板的世界坐标系是人为定义好的，标定板上每一个格子的大小是已知的，我们可以计算得到世界坐标系下的 (U,V)。

由这里的 H 是齐次矩阵，有 8 个独立未知元素。每一个标定板角点可以提供两个约束方程 (u,U,V) 的对应关系、(v,U,V) 的对应关系），因此，当一张图片上的标定板角点数量等于 4 时，即可求得该图片对应的矩阵 H。当一张图片上的标定板角点数量大于 4 时，利用最小二乘法回归最佳的矩阵 H。

8.8.2 求解内参矩阵

我们已知了矩阵 $H = A(R_1, R_2, T)$ ，接下来需要求解相机的内参矩阵 A。

我们利用 R_1, R_2 作为旋转矩阵 R 的两列，存在单位正交的关系，即：

$$
\begin{aligned}
&R1^T R2 = 0 \\
&R1^T R1 = R2^T R2 = 1
\end{aligned}
$$

则由 H 和 R_1, R_2 的关系，可知：

$$
\begin{aligned}
R1 &= A^{-1} H1 \\
R2 &= A^{-1} H2
\end{aligned}
$$

代入可得：

$$
\begin{aligned}
&H1^T A^{-T} A^{-1} H2 = 0 \\
&H1^T A^{-T} A^{-1} H1 = H2^T A^{-T} A^{-1} H2 = 1
\end{aligned}
$$

另外，我们发现，上述两个约束方程中均存在矩阵 $A^{-T}A^{-1}$。因此，我们记 $A^{-T}A^{-1} = B$ ，则 B 为对称阵。我们试图先求解出矩阵 B，通过矩阵 B 再求解相机的内参矩阵 A。

同时，为了简便，我们记相机内参矩阵 A 为：

$$
A = \begin{pmatrix} \frac{f}{dX} & -\frac{f\cot\theta}{dX} & u_0 & 0 \\ 0 & \frac{f}{dY\sin\theta} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}
$$

则:

$$
A^{-1} = \begin{bmatrix} \frac{1}{\alpha} & -\frac{\gamma}{\alpha\beta} & \frac{\gamma v_0 - \beta u_0}{\alpha\beta} \\ 0 & \frac{1}{\beta} & -\frac{v_0}{\beta} \\ 0 & 0 & 1 \end{bmatrix}
$$

则用矩阵 A 表示矩阵 B 得：

$$B = A^{-T}A^{-1} = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma v_0 - \beta u_0}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{1}{\beta^2} + \frac{\gamma^2}{\alpha^2\beta^2} & \frac{\gamma(\beta u_0 - \gamma v_0)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{\gamma v_0 - \beta u_0}{\alpha^2\beta} & \frac{\gamma(\beta u_0 - \gamma v_0)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(\beta u_0 - \gamma v_0)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix}$$
$$= \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix}$$

注意：由于为 B 对称阵，上式出现了两次 B_{11}, B_{12}, B_{23}。

这里，我们可以使用 $B = A^{-T}A^{-1}$ 将前面通过 R_1, R_2 单位正交得到的约束方程化为：

$$H1^T BH2 = 0$$
$$H1^T BH1 = H2^T BH2 = 1$$

因此，为了求解矩阵 B，我们必须计算 $H_i^T BH_j$。则：

$$H_i^T BH_j = [H_{1i} \quad H_{2i} \quad H_{3i}] \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{32} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \begin{bmatrix} H_{1j} \\ H_{2j} \\ H_{3j} \end{bmatrix}$$
$$= [H_{1i}H_{1j} \quad H_{1i}H_{2j} + H_{2i}H_{1j} \quad H_{2i}H_{2j} \quad H_{1i}H_{3j} + H_{3i}H_{1j} \quad H_{2i}H_{3j} + H_{3i}H_{2j} \quad H_{3i}H_{3j}] \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{bmatrix}$$

上述方程看起来有点复杂，但是其实不然，我们可以记：

$$v_{ij}$$
$$= [H_{1i}H_{1j} \quad H_{1i}H_{2j} + H_{2i}H_{1j} \quad H_{2i}H_{2j} \quad H_{1i}H_{3j} + H_{3i}H_{1j} \quad H_{2i}H_{3j} + H_{3i}H_{2j} \quad H_{3i}H_{3j}]^T$$
$$b = [B_{11} \quad B_{12} \quad B_{22} \quad B_{13} \quad B_{23} \quad B_{33}]^T$$

则上述方程化为：$H_i^T BH_j = v_{ij}b$

此时，通过 R_1, R_2 单位正交得到的约束方程可化为：

$$v_{12}^T b = 0$$
$$v_{11}^T b = v_{22}^T b = 1$$

即：

$$\begin{bmatrix} v_{12}^T \\ v_{11}^T - v_{22}^T \end{bmatrix} b = vb = 0$$

其中，矩阵 $v = \begin{bmatrix} v_{12}^T \\ v_{11}^T - v_{22}^T \end{bmatrix}$。

由于矩阵 H 已知，矩阵 v 又全部由矩阵 H 的元素构成，因此矩阵 v 已知。

此时，我们只要求解出向量 b，即可得到矩阵 B。每张标定板图片可以提供一个 $vb = 0$ 的约束关系，该约束关系含有两个约束方程。但是，向量 b 有 6 个未知元素。因此，单张图片提供的两个约束方程是不足以解出来向量 b。因此，我们只要取 3 张标定板照片，得到 3 个 $vb = 0$ 的约束关系，即 6 个方程，即可求解向量 b。当标定板图片的个数大于 3 时（事实上一般需要 15 到 20 张标定板图片），可采用最小二乘拟合最佳的向量 b，并得到矩阵 B。（B 的各元素值已解得。）

$$B = \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma v_0 - \beta u_0}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{1}{\beta^2} + \frac{\gamma^2}{\alpha^2\beta^2} & \frac{\gamma(\beta u_0 - \gamma v_0)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{\gamma v_0 - \beta u_0}{\alpha^2\beta} & \frac{\gamma(\beta u_0 - \gamma v_0)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(\beta u_0 - \gamma v_0)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix}$$

根据矩阵 B 的元素和相机内参 $\alpha, \beta, \gamma, u_0, v_0$ 的对应关系（如上式），可得到：

$$v_0 = \frac{B_{12}B_{13} - B_{11}B_{23}}{B_{11}B_{22} - B_{12}^2}$$
$$\alpha = \sqrt{\frac{1}{B_{11}}}$$
$$\beta = \sqrt{\frac{B_{11}}{B_{11}B_{22} - B_{12}^2}}$$
$$\gamma = -B_{12}\alpha^2\beta$$
$$u_0 = \frac{\gamma v_0}{\beta} - B_{13}\alpha^2$$

即可求得相机的内参矩阵:

$$A=\begin{pmatrix}\frac{f}{dX} & -\frac{f\cot\theta}{dX} & u_0 & 0\\ 0 & \frac{f}{dY\sin\theta} & v_0 & 0\\ 0 & 0 & 1 & 0\end{pmatrix}=\begin{bmatrix}\alpha & \gamma & u_0\\ 0 & \beta & v_0\\ 0 & 0 & 1\end{bmatrix}$$

8.8.3 求解外参矩阵

这里再次强调一下，对于同一个相机，相机的内参矩阵取决于相机的内部参数，无论标定板和相机的位置关系是怎么样的，相机的内参矩阵不变。

这也正是在第 2 部分“求解内参矩阵”中，我们可以利用不同的图片（标定板和相机位置关系不同）获取的矩阵 H，共同求解相机内参矩阵 A 的原因。

但是，外参矩阵反映的是标定板和相机的位置关系。对于不同的图片，标定板和相机的位置关系已经改变，此时每一张图片对应的外参矩阵都是不同的。

在关系：$H=A(R_1R_2T)$ 中，我们已经求解得到了矩阵 H（对于同一张图片相同，对于不同的图片不同）、矩阵 A（对于不同的图片都相同）。

通过公式：$A^{-1}H=(R_1R_2T)$ ，即可求得每一张图片对应的外参矩阵 (R_1R_2T)。

注意，这里值得指出，完整的外参矩阵为 $v=\begin{bmatrix}R & T\\ 0 & 1\end{bmatrix}$。

但是，由于此标定法将世界坐标系的原点选取在棋盘格上，则棋盘格上任一点的物理坐标 $W=0$ ，将旋转矩阵 R 的第三列 R_3 消掉，因此，R_3 在坐标转化中并没有作用。又 R_3 要使得 R 满足旋转矩阵的性质，即列与列之间单位正交，因此可以通过向量 R_1,R_2 的叉乘，即 $R_3=R_1\times R_2$，计算得到。

此时，相机的内参矩阵和外参矩阵均已得到。

8.8.4 相机标定的畸变参数

常见标定法仅仅考虑了畸变模型中影响较大的径向畸变。径向畸变公式（2 阶）如下：

$$\hat{x}=x\left(1+k_1r^2+k_2r^4\right)$$
$$\hat{y}=y\left(1+k_1r^2+k_2r^4\right)$$

其中，(x,y)，$(\hat{x},\hat{y})$ 分别为理想的无畸变的归一化的图像坐标、畸变后的归一化图像坐标, r 为图像像素点到图像中心点的距离，即 $r^2=x^2+y^2$。

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{dX} & -\frac{\cot\theta}{dX} & u_0 \\ 0 & \frac{1}{dY\sin\theta} & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

其中，(u, v) 为理想的无畸变的像素坐标。由于 θ 接近于 90^0，则上式近似为：

$$u = \frac{x}{dX} + u_0$$
$$v = \frac{y}{dY} + v_0$$

同理可得畸变后的像素坐标 $(\hat{u}, \hat{v})$ 的表达式为：

$$\hat{u} = \frac{\hat{x}}{dX} + u_0$$
$$\hat{v} = \frac{\hat{y}}{dY} + v_0$$

代入径向畸变公式（2 阶）则有：

$$\hat{u} - u_0 = (u - u_0)\left(1 + k_1 r^2 + k_2 r^4\right)$$
$$\hat{v} - v_0 = (v - v_0)\left(1 + k_1 r^2 + k_2 r^4\right)$$

可化简得：

$$\hat{u} = u + (u - u_0)\left(k_1 r^2 + k_2 r^4\right)$$
$$\hat{v} = v + (v - v_0)\left(k_1 r^2 + k_2 r^4\right)$$

即为:

$$\begin{bmatrix} (u - u_0) r^2 & (u - u_0) r^4 \\ (v - v_0) r^2 & (v - v_0) r^4 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \hat{u} - u \\ \hat{v} - v \end{bmatrix}$$

上式中的 $\hat{u}, u, \hat{v}, v$ 可以通过识别标定板的角点获得，每一个角点可以构造两个上述等式。有 m 幅图像，每幅图像上有 n 个标定板角点，则将得到的所有等式组合起来，

可以得到 mn 个未知数为的 $k = [k_1, k_2]^T$ 约束方程，将约束方程系数矩阵记为 D，等式右端非齐次项记为 d，可将其化为矩阵形式：

$$Dk = d$$

则使用最小二乘法可求得：

$$k = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \left(D^T D\right)^{-1} D^T d$$

此时，相机的畸变矫正参数已经标定好。

需要指出，上述公式推导的时候以 2 阶径向畸变为例，但实际上更高阶的径向畸变同理，只是需要的约束方程个数更多而已。

到此，相机的标定过程都已经叙述完了。接下来回到我们的机械臂积木抓取搭建中。

第 9 章 机器视觉抓取实战

在我们进入本章之前，让我们明确一下该项目需要达成的目标。

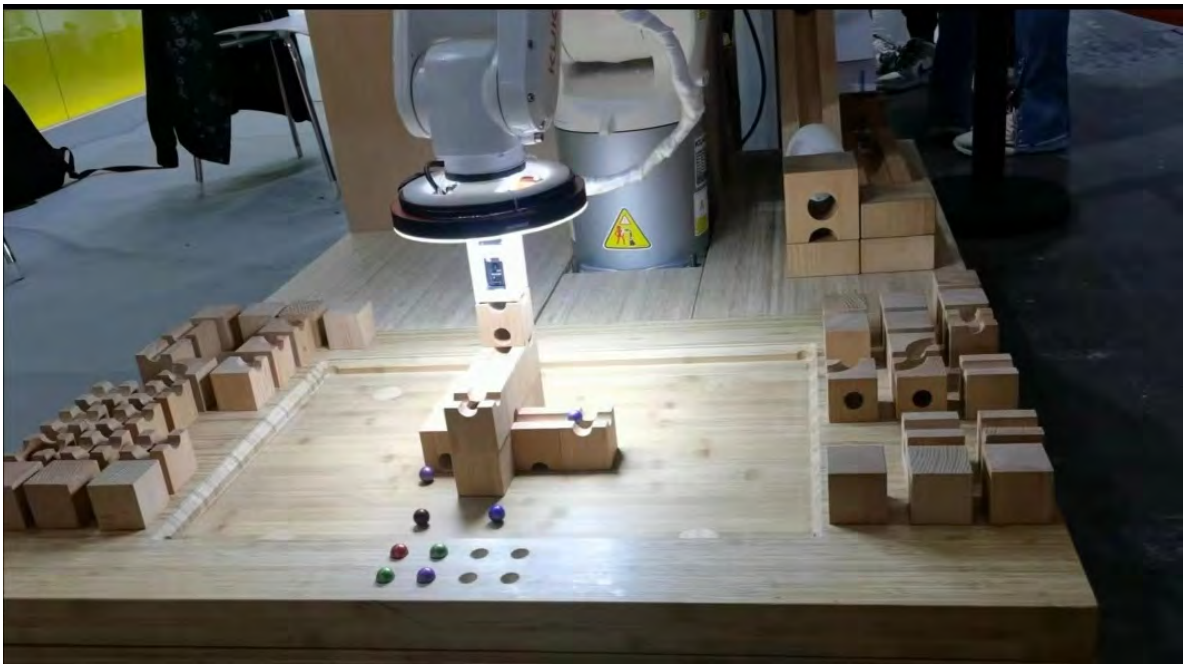


图 9.1: 机械臂搭建积木

我们的期望是通过视觉控制机械臂来巧妙地抓取积木并进行搭建，最终让小球沿着轨道顺利落下。在本项目中我们选用的积木是惟木匠心高精度木质轨道积木，因其持续精度高和良好表面质量，最终选取了该品牌的轨道积木作为研究对象。

图 9.2: 惟木匠心高精度木质轨道积木

惟木匠心品牌简介

惟木匠心 (wayMove) 是一家致力于研发生产高精度木质轨道积木的科技型企业，研发工作是由位于美国印第安纳波利斯总部的 WOODENCUBE LLC 公司完成，占地 2 英亩。国内的生产基地位于杭州市西湖区，占地 5 亩，拥有标准的厂房 2000 平方，是国家科技创新基金项目承担单位。为确保成品积木的持续精度和良好表面质量，只使用经 PEFC 认证的德国 AA 级榉木和质地非常坚硬的非洲樱桃木作为主要积木原料，通过 KUKA 六轴铣削机器人一次成型加工，采用美国原装进口的 Tried&True 牌亚麻籽油等对积木表面进行持久保护。产品远销欧美日韩港澳台等国家和地区，深受玩家的喜爱。

为了成功完成整个项目，我们必须对积木搭建的流程有深入的了解。

9.1 积木搭建

9.1.1 术语说明

白方块: 无轨道的方块，用来搭建基座。

图 9.3: 白方块

轨道: 在方块表面的轨道，也称面轨道。包括以下五种:

(1) 双向直轨道

(2) 十字交叉轨道

(3) 双向弯轨道

(4) 四向弯轨道

(5) 三向直轨弯轨组合

图 9.4: 五种面轨道

隧道: 在方块内部的轨道，也称内轨道。既有水平轨道，又有倾斜轨道，出入口所在的面和位置不同。下边是常见的 7 种基本隧道:

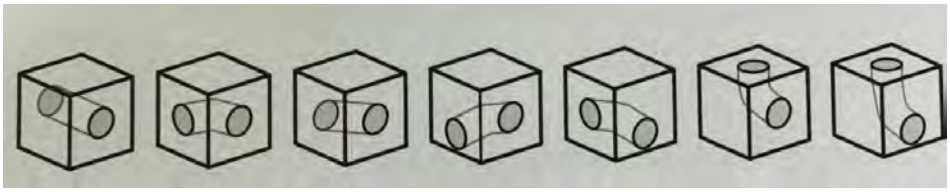


图 9.5: 积木内部隧道

底轨道: 与下层面轨道相合，又称避空轨道。

使弹珠能顺利通过。

图 9.6: 上下底轨道相互配合

轨道的阶: 整块积木轨道出入口根据圆心位置的高度不同共分五阶: 上、中上、中、中下、下分别用编号 1、2、3、4、5 对应。如右图所示，出入口皆为 1 阶的轨道即面轨道，出入口皆为 5 阶的轨道即底轨道。

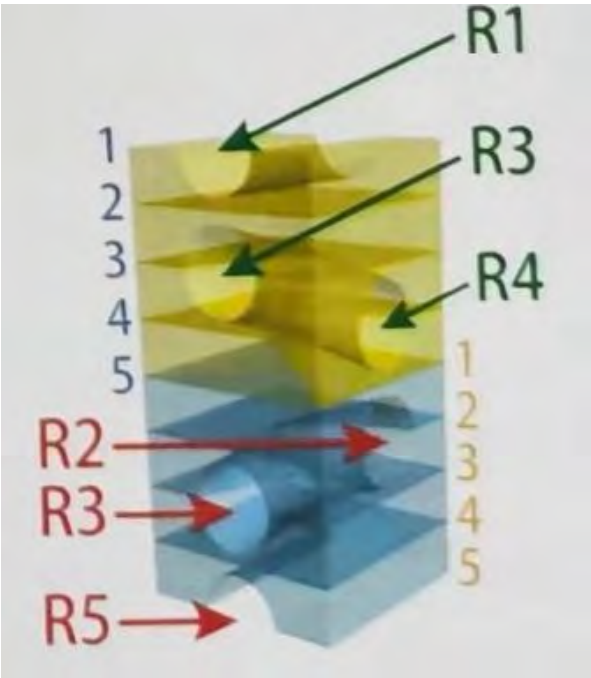


图 9.7: 轨道的阶

(分五阶是惟木匠心轨道积木的创新之一，配合其先进的加工工艺，生产出有更多搭建可能性的轨道积木。在五阶体系下，所有基本轨道共有 24 种，由此组合而成的积木共有 10 万多种。)

9.1.2 搭建入门

轨道积木本身没有什么规则，我们也不愿给玩家太多限制，很多时候玩家的创意超出我们想象。但是由于几何结构和物理力学的限制，我们有如下两个原则：

连通性原则: 轨道必须是连通的 (这是空间几何结构的限制)

动能势能原则: 由降阶或降层的势能转化为弹珠前进的动能 (这是物理力学的限制)

即首先要保证轨道的通畅; 其次，一个降阶轨道之后不能放置过多的水平轨道，否则摩擦力将减慢弹珠的速度直到停止。玩家可在游戏中不断实践这两个原则。此积木的优点: 五阶设计有更多的连通可能性，轨道精密衔接减少阻力，优化降阶轨道降低能量损耗。

本项目采用的是 Standard 标准款套系，共计 50 块，构成如下:

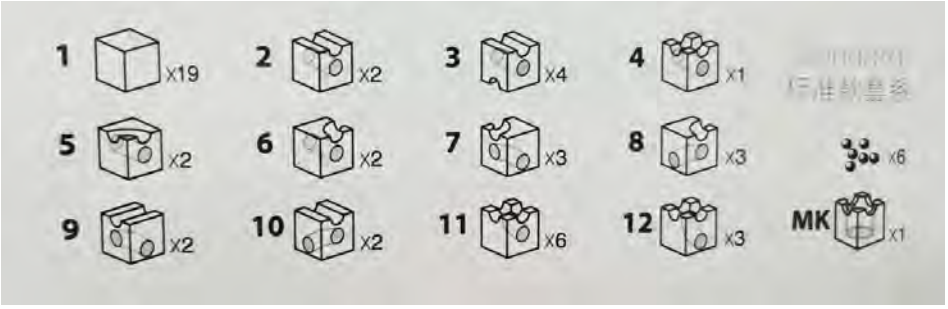


图 9.8: 惟木匠心 Standard 标准款套系

9.1.3 积木分类

轨道积木是一个六面体。在视觉系统识别一块积木类型时，首先会识别到积木的一个面，然后通过分析积木相邻面的关系，我们就能够确定该积木的类型编号，以及

放置位置所需的旋转角度，在积木的搭建过程中还需考虑到积木内部以及表面轨道的连通性，积木的搭建必须非常准确，才能使小球顺着轨道滑落地面。关于积木编码的详细过程，可以参考知网论文《具有旋转不变性的立体轨道积木编码系统》。

因此，要准确识别积木的种类并非易事。为了解决这一问题，我们进行了多方面的尝试。接下来，我们将从模板匹配和三维重建两个角度进行介绍。在这之前我们先引入机械臂与视觉系统之间的手眼标定过程。

9.2 机械臂手眼标定

通常情况下，对于独立的视觉系统（不包括激光雷达），要确定一系列拍摄的照片中每张照片相机坐标系的相对位置并非易事。然而，一旦将视觉系统绑定到机械臂上，情况就发生了变化。谈及世界工业机器人，就绕不开以发那科、库卡、ABB、安川电机为代表的四大家族，本项目中使用的机械臂就是库卡 KR6 R900-2。

图 9.9: 机械臂四大家族

工业机械臂经常被设计成坚固而稳定的结构，以适应在高强度的工业生产环境中的工作要求。它们具备承受各种负载和惯性力的能力，确保在操作过程中保持出色的稳定性。这使得工业机械臂在执行高强度、长时间的生产任务时能够维持高效而稳定的运行状态。机械臂的关节和执行器通常采用精密设计和制造，以确保在运动中能够达到卓越的重复定位精度。

以库卡机械臂为例，其重复定位精度可达到 0.02mm 水平，这意味着在执行相同任务的多次运动中，机械臂能够以高度准确的方式完成各个动作。当视觉系统与机械臂结合时，这种高精度的机械性能为相机在运动中实时测量位置信息提供了可靠的基础。这一层先验知识的加入使得视觉处理任务变得更为简化。

下面介绍机械臂的手眼标定。

所谓手眼系统，就是人眼睛看到一个东西的时候要让手去抓取，就需要大脑知道

眼睛和手的坐标关系。而相机知道的是像素坐标，机械手是空间坐标系，所以手眼标定就是得到像素坐标系和空间机械手坐标系的坐标转化关系。

目前工业上通常使用两种方法进行机械臂的手眼协作。

第一种是：眼在手外 (eye-to-hand)，即摄像机与机械臂分离，应用场景为：机械臂的工作位置固定，工作平面固定，摄像机置于合理的位置以识别机械臂要工作的平面，例如：货物码垛、货物搬运等。

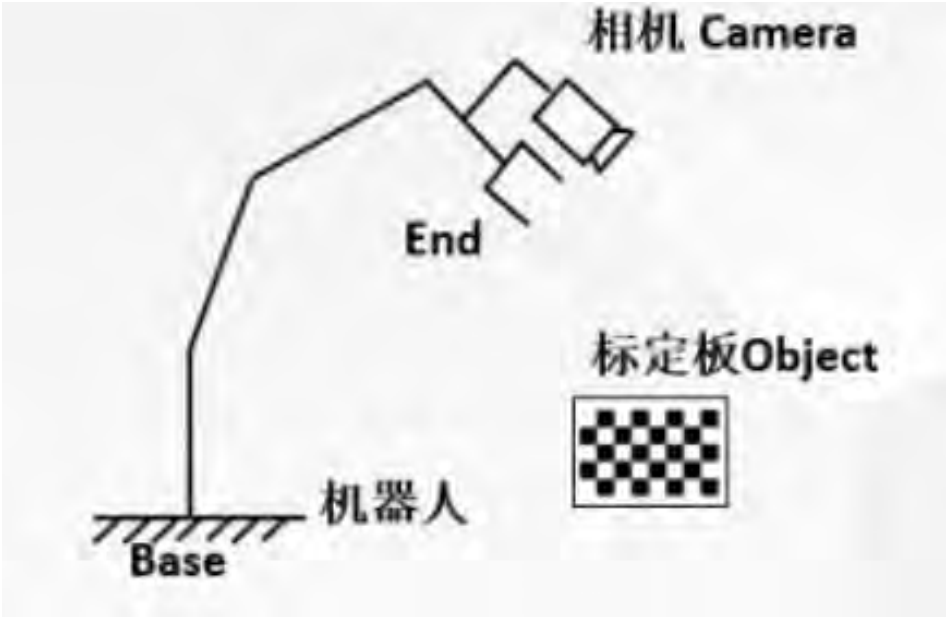


图 9.10: 眼在手上 (eye-in-hand)

第二种是：眼在手上 (eye-in-hand)，即摄像机布置在机械臂末端，应用场景为：机械臂移动式的工作，例如：果园采摘、货物运转等。

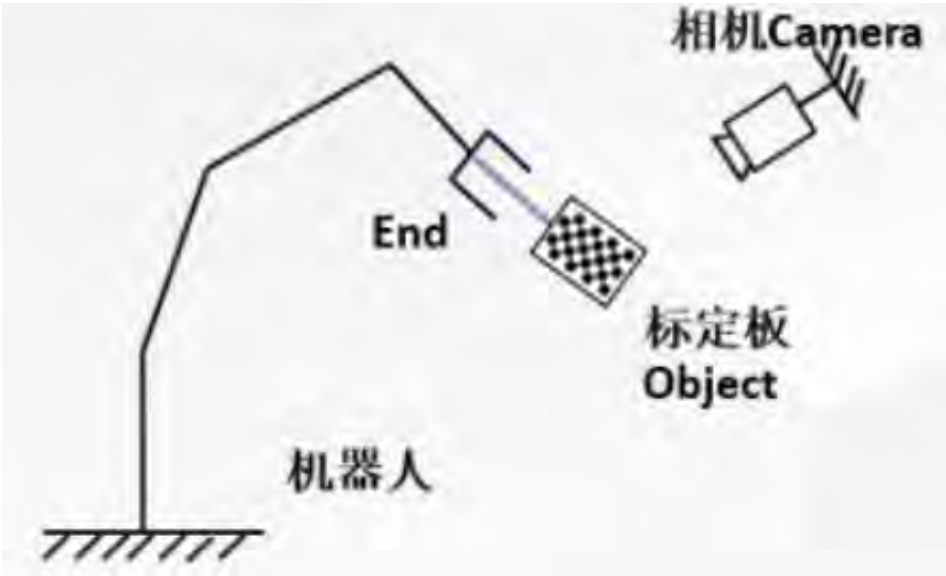
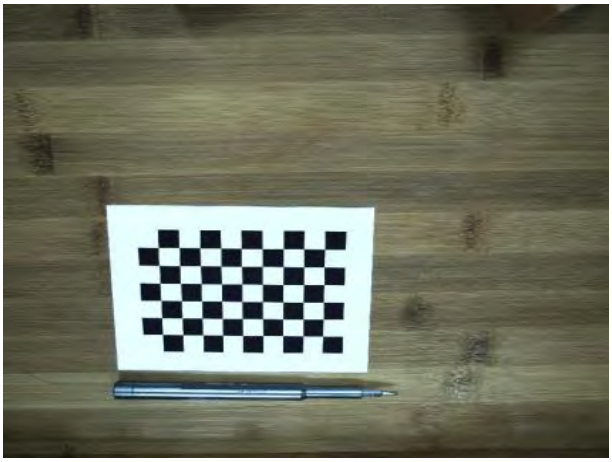


图 9.11: 眼在手外 (eye-to-hand)

总的来说，采用手在眼上可以适应更多的应用场景，所以本项目的摄像机搭载方案为眼在手上。

9.2.1 机械臂手眼标定步骤

1．我们的目标是要获取棋盘格上的格点在机械臂 base 坐标系下的坐标。为了确定空间中一个点的坐标，需要首先进行机械臂 tool 坐标的标定，此举是为了求得铝合金尖头末端相对于机械臂法兰坐标系原点的位置，也就获得了铝合金尖头末端在机械臂 base 坐标系下的坐标。需要换上铝合金头工具后使用 4 点法进行标定。(注: 铝合金头要保持尖锐，测量才能达到更高精度)



棋盘格

用于坐标定位的铝合金头

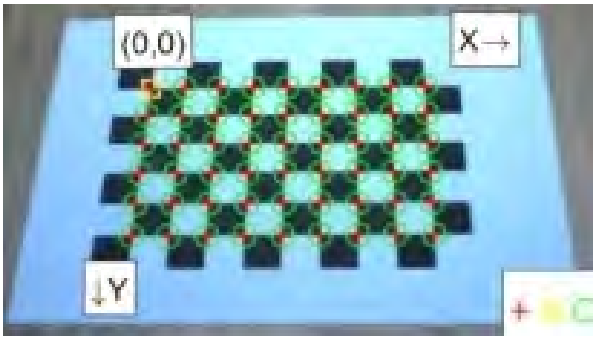
图 9.12: 标定板和铝合金头

安装铝合金头

四点法标定的结果

图 9.13: 进行四点标定

2．标定结束后，操纵机器臂将工具的点手动示教依次触碰标定板的三个点，确定标定板的原点，X 轴方向、Y 轴方向。有了这些点的数据就可以求出标定板坐标系与机械臂 base 坐标系之间的相对关系，即旋转平移矩阵。



3．将针尖工具换上相机，并对着标定板进行拍摄，记录每次拍摄的法兰坐标 (X, Y, Z, A, B, C)，拍摄 20 张左右。

图 9.14: 末端安装相机的 KUKA 机械臂

可以通过 KUKA 示教器的显示——> 实际位置界面来读取机械臂末端的位置。

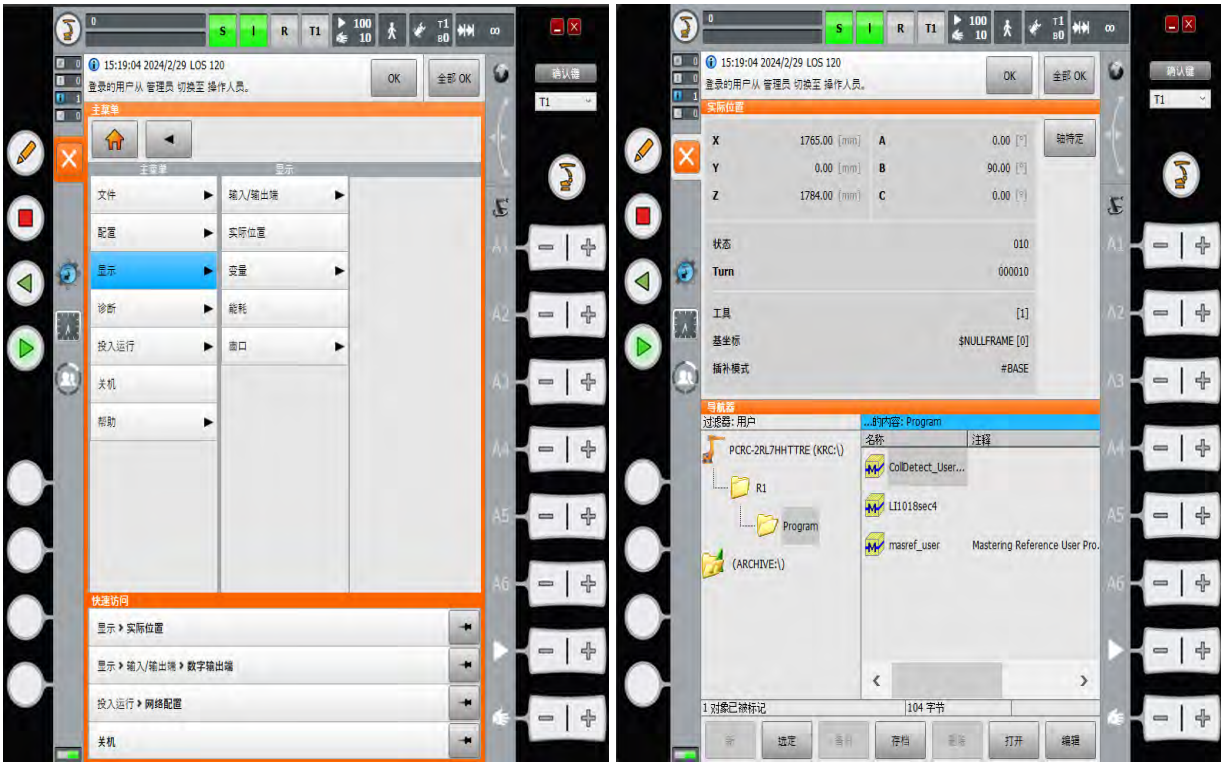


图 9.15: 示教器上显示的坐标

4．使用 matlab 进行相机标定，计算出相机的内参以及每张图片的外参。

5．计算：首先我这里将所有坐标系都列出来

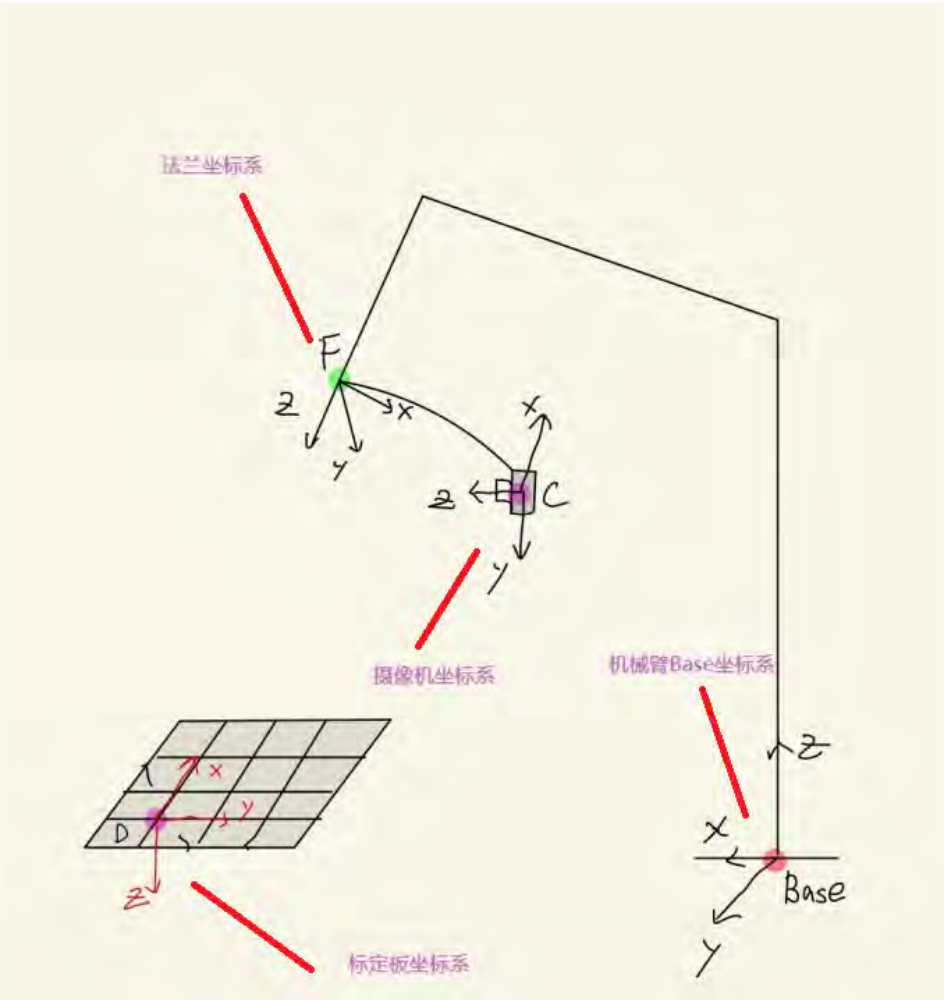


图 9.16: 所有坐标系

6．通过第 1 步中的三点坐标，我们可以很容易地建立标定板坐标系，这里我们建立的标定板坐标系原点为 D, x, y 轴如图，根据向量叉乘和右手定则得到 z 轴，这里刻意将 z 轴指向地面，是为了和 MATLAB 中的相机标定过程中的世界坐标系相一致。

7．首先明确我们的目的是为了求法兰到相机的旋转平移矩阵，这里我们将旋转矩阵记为 R，平移矩阵记为 T，为了方便我们求解，我们分开求，先求平移向量 T，则由平移向量的性质，我们可以知道：

$$T_F^C = T_F^B + T_B^D + T_D^C \tag{1.1}$$

这本质上与向量的加减相同，其中 T_F^C 为法兰坐标点 F 到相机坐标点 C 的在 Base 坐标系下平移向量，它就可以分解为 T_F^B，即法兰到 Base 坐标点的平移向量，加上 T_B^D，即 Base 坐标点到标定板坐标点的平移向量，再加上 T_D^C，即标定板坐标系到相机坐标系的平移向量，T_F^B 我们可以通过示教器就可以很轻松获得，T_B^D 我们可以通过我们在第 2 步中得到的原点 D 的坐标得到，从而问题就在于得到 T_D^C。

我们在第 3 步中进行了张正友相机标定，从标定结果中我们可以知道标定板坐标

系原点 D 在相机坐标系中的坐标，即向量 $\vec{CD}$，这个向量是在相机坐标系中的，而上面的 T_F^C，T_B^D 和 T_D^C 都是在 Base 坐标系下，从而我们需要转换，即把这个向量 $\vec{CD}$ 左乘一个摄像机坐标系到 Base 坐标系的旋转矩阵 R_C^B，而这个旋转矩阵，我们可以根据 $R_D^B * R_c^D$ 得到，从而我们就得到了在 Base 坐标系下的 D 点到 C 点的平移向量，我们这里记作 $T_D^{C'}$，从而我们就得到了 (1.1) 右式中所有的量，进而我们就求得了法兰坐标点 F 到相机坐标点 C 的平移向量 T_F^C。千万不能认为现在就已经计算完成了，这个平移向量是在 Base 坐标系中，而这个向量我们每换一个拍照姿势，就会变化，并不是不变量，我们需要将这个向量转换到法兰坐标系中，那么这个向量就是以法兰原点为起点，指向照相机，那么这个向量就不会因为机械臂的运动而发生变化，我们就需要在这个向量左乘一个 Base 到法兰的旋转矩阵 R_B^F，那么我们就得到了这个不变量法兰到相机平移向量。

下面求一下法兰到相机的旋转矩阵，即 R_F^C，而 $R_F^C = R_D^C * R_B^D * R_F^B$, 其中 R_D^C 为标定板到相机的旋转矩阵，R_B^D 为 Base 到标定板坐标系的旋转矩阵，R_F^B 为法兰到 Base 的旋转矩阵。

R_F^B 即法兰到 Base 坐标系的旋转矩阵可以直接通过坐标的 A, B, C 欧拉角直接转换出来，R_B^D 可以通过我们在第 2 步中得到，利用确定的标定板坐标系原点的 x, y, z 坐标在 Base 的 x, y, z 坐标进行向量内积得到，而 R_D^C 我们就可以通过标定得到的那张图片的外参矩阵求逆得到，至此，我们所有 R_F^C 就可以直接求得。

通过上面的求解，我们就得到了旋转平移矩阵 $\begin{bmatrix} R_F^C & T_F^C \end{bmatrix}$。

9.2.2 旋转平移矩阵的检验

这里主要用到的知识是空间中直线和平面的交点。

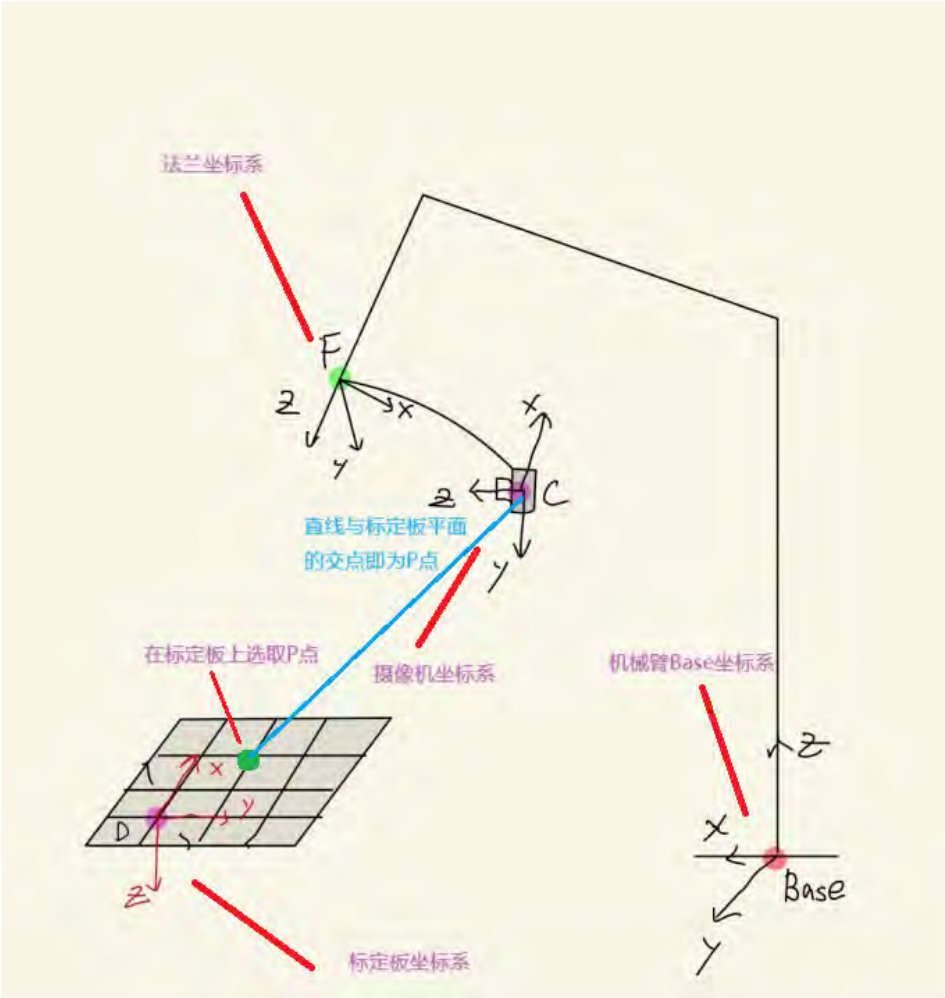
我们的目的在于通过相机光心以及棋盘格上的一个点组成的一条射线与棋盘格的平面相交于一点，将该点坐标算出，并且与现实中的点进行对比从而获取我们的旋转平移矩阵的精度。

对着棋盘格任意拍摄一张照片并且在其上找到一角点 P，记 P 在机械臂 Base 坐标系中的坐标为 (X, Y, Z)，并且在照片上像素坐标系中找到这一点，记录像素坐标为 (u, v)，为了计算写为齐次形式 $(u, v, 1)$。

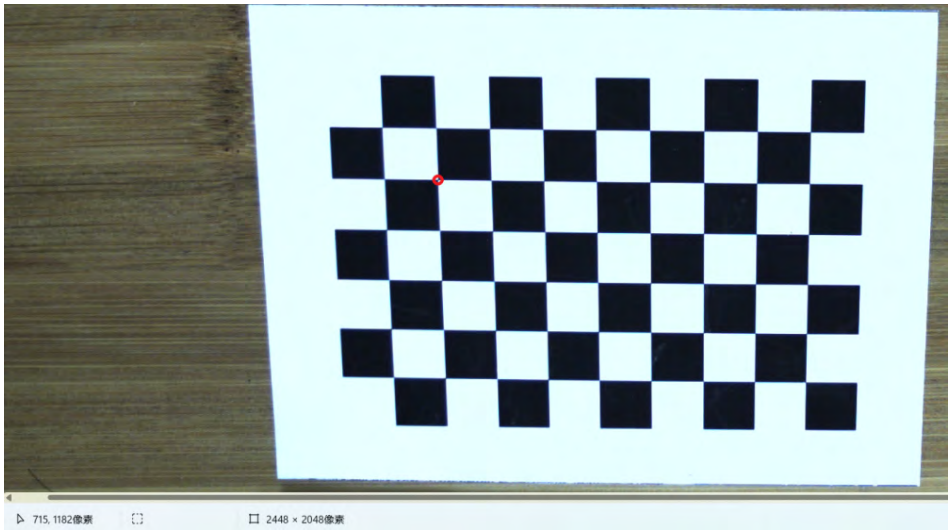
并且张正友标定法中我们知道像素坐标系和相机坐标系之间的关系，从而我们可以根据等式 $\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix}$，从而知道在相机坐标系下这个射线的方向，再根据计算出的相机相对于法兰的旋转平移矩阵以及法兰坐标系相对于机械臂 Base 坐标系的旋转平移矩阵，我们就可以求出这个射线在机械臂 Base 坐标系下的方向，以及光心（即摄像机坐标系原点）在机械臂 Base 坐标系下的点，从而我们可以轻松写出这个直线方程（此直线是在机械臂 Base 坐标系中表示）。

三维空间中的直线方程可以又两个三元一次方程写出。再加上标定板的平面方程

（根据之前合金头工具所获取到的在机械臂 Base 坐标系下的标定板上角点坐标可以联立方程，3 点可以确定一个平面），从而 3 个方程、3 个未知量，我们直接求这个方程的解，就是 P 点在机械臂 Base 坐标系下的坐标，将这个坐标与我们一开始就计算的坐标相对比，从而获得我们旋转平移矩阵的精度。



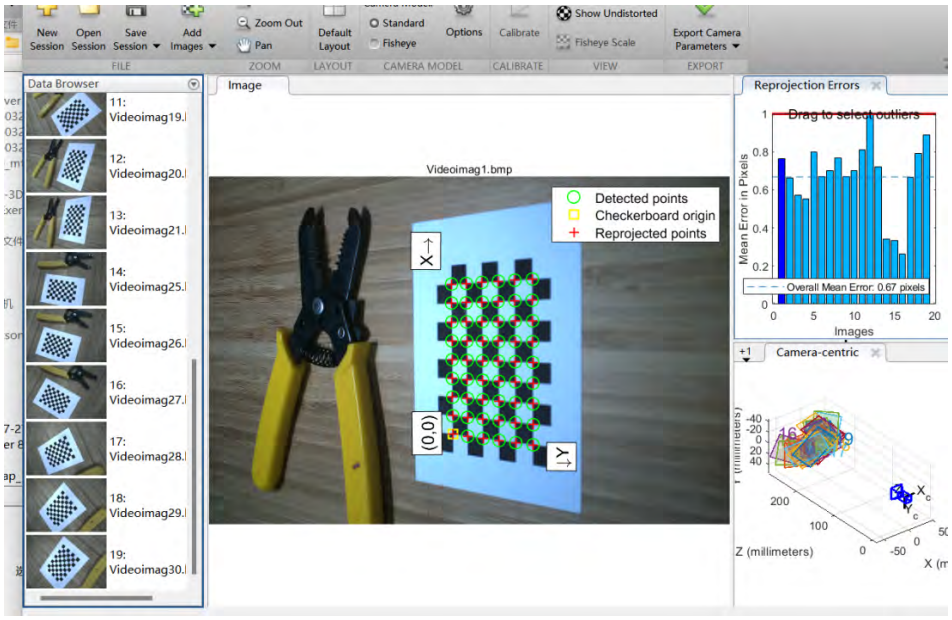
注: 在标定板上选取的 P 点是标定板黑白格上的交点，并且由于我们之前已经装上合金头工具测到了标定板上各格点的机械臂 Base 坐标系下的坐标，因此这个坐标我们是提前已知的。



而标定板上 P 点 (红色圆圈圈出) 的像素 (u, v) 可以用画图工具粗略表示，也可以用 matlab 或者 OpenCV 等其他标定板角点识别软件精确识别出角点的亚像素位置。(棋盘格识别角点像素比较容易，此处不再赘述)。

三、本次实验结果

我们使用的是 matlab 的标定工具箱，总共标定的图片 30 张，检测正确 19 张。



得到的旋转和平移矩阵如下图所示:

```
命令行窗口
flange2Cam_translation =

    0.1305
    0.0029
   -0.0364

    0.8642   -0.0040    0.5032    0.1282
    0.0027    1.0000    0.0033    0.0019
   -0.5032   -0.0015    0.8642   -0.0344
         0         0         0    1.0000
```

可以看到旋转阵 $R = \begin{bmatrix} 0.8642 & -0.0040 & 0.5032 \\ 0.0027 & 1.0000 & 0.0033 \\ -0.5032 & -0.0015 & 0.8642 \end{bmatrix}$

平移矩阵 $T = \begin{bmatrix} 0.1282 \\ 0.0019 \\ -0.0344 \end{bmatrix}$

取最后一张标定成功的照片的所有棋盘格角点像素值

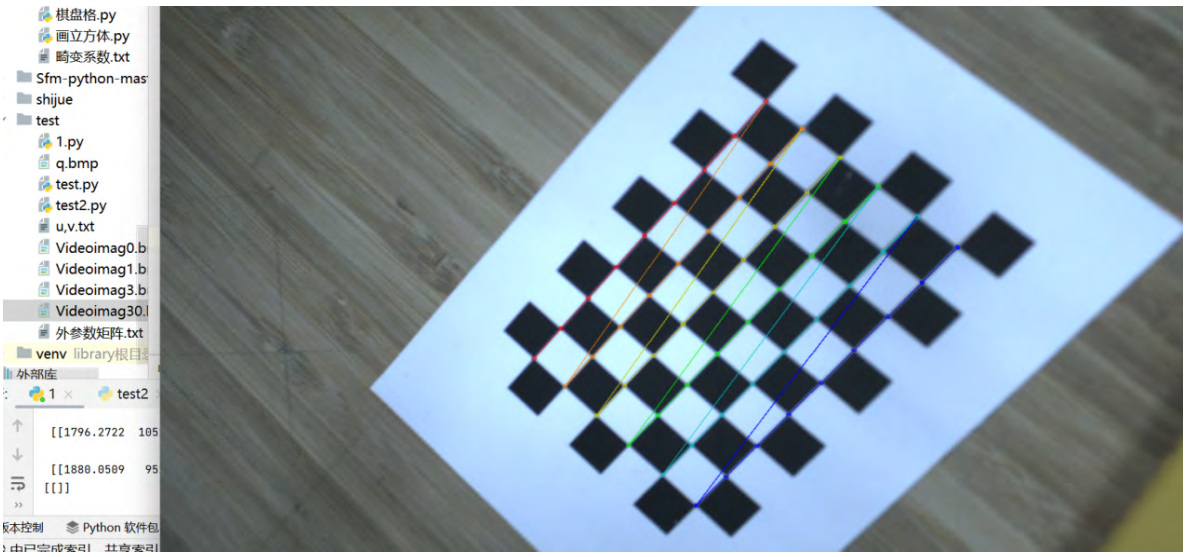


图 **9.17:** matlab 工具箱得到的标定板角点像素

计算得到棋盘格角点在机械臂 Base 坐标系上的坐标:

编辑器 - zuizhongweizi.m

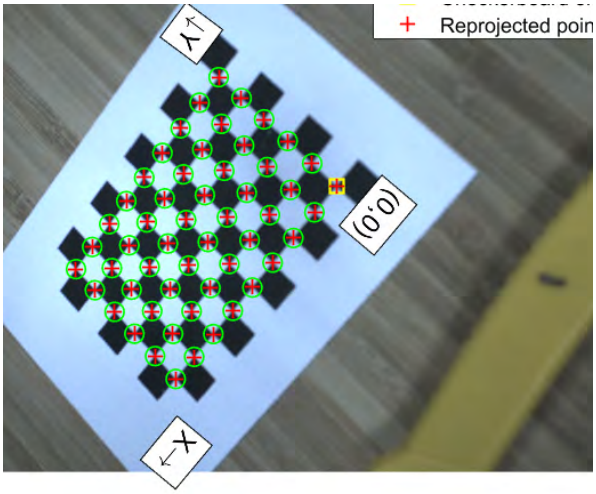
PP_LIST　offset

54x3 double

	1	2	3	4
1	0.7621	0.0241	0.0283	
2	0.7622	0.0320	0.0283	
3	0.7622	0.0401	0.0282	
4	0.7623	0.0481	0.0282	
5	0.7623	0.0561	0.0282	
6	0.7624	0.0642	0.0281	
7	0.7624	0.0723	0.0281	
8	0.7625	0.0803	0.0281	
9	0.7625	0.0884	0.0280	
10	0.7702	0.0240	0.0282	
11	0.7702	0.0320	0.0282	
12	0.7703	0.0400	0.0282	
13	0.7703	0.0480	0.0281	
14	0.7703	0.0561	0.0281	
15	0.7704	0.0641	0.0281	
16	0.7704	0.0722	0.0280	
17	0.7705	0.0803	0.0280	
18	0.7705	0.0883	0.0280	
19	0.7782	0.0239	0.0282	
20	0.7782	0.0319	0.0281	
21	0.7783	0.0399	0.0281	
22	0.7784	0.0480	0.0281	

用工具针尖获取到的机械臂 Base 坐标系坐标系下的坐标：

```
base_p1 = [0.80591, 0.08940, 0.02771];%o
base_p2 = [0.76576, 0.08950, 0.028];%y
base_p3 = [0.80547, 0.02511, 0.02797];%x
```



新拍摄一张照片选取标定板上的一个角点，计算误差:

```
cam2World_rotation =

   -0.7047    0.1696    0.6889
    0.1486    0.9847   -0.0905
   -0.6938    0.0386   -0.7192

    0.8191
    0.0902
    0.0276
```

用铝合金工具头得到机械臂 Base 坐标系中的坐标（单位: 米）:

[0.81891,0.08940,0.02771]

其中 X,Y 和 Z 方向的误差为 $1mm$。

为了更精确计算误差，我们再次拍摄，计算坐标：

```
cam2World_rotation =

   -0.7002    0.2037    0.6843
    0.1719    0.9783   -0.1152
   -0.6929    0.0370   -0.7201

    0.8161
    0.0921
    0.0276
```

可以看出，两次计算的结果误差类型基本相同，可以通过后续 OpenCV 代码进行补偿。

误差分析：

1. 标定过程中图片不够清晰，可以改进。
2. 像素坐标取的不够准确。
3. 机械臂下面的标定板可能被移动过，对我们误差计算有影响。
4. 拍照距离待测点的远近也会对误差进行影响。
5. 相机镜头照片的畸变也会对计算造成影响。
6. 张正友标定法算出来的参数只在图像局部区域内精确。

因此拍摄出高质量的照片是很有必要的。

9.3 本次实验代码

标定过程使用 matlab 工具箱进行标定。

Matlab 代码

```
base_p1 = [0.80591,0.08940,0.02771];
base_p2 = [0.76576,0.08950,0.028];
base_p3 = [0.80547,0.02511,0.02797];
%标定板上点在机械臂Base坐标系上的点（用合金工具头测得）

board_axis_x=(base_p3-base_p1)/norm(base_p3-base_p1);
board_axis_y=(base_p2-base_p1)/norm(base_p2-base_p1);
board_axis_z=cross(board_axis_x,board_axis_y)/ ...
norm(cross(board_axis_x,board_axis_y));
%计算测得的标定板坐标系向量，并进行单位化

board_axis_y=board_axis_y-board_axis_x*board_axis_y'*board_axis_x;
board_axis_z=board_axis_z-board_axis_x*board_axis_z'*board_axis_x- ...
board_axis_y*board_axis_z'*board_axis_y;
%使用施密特正交化将测得的坐标系向量互相垂直
%board-->world　这里的World实际表示机械臂Base坐标系，
%即标定板相对机械臂Base坐标系的旋转平移矩阵
board2World_rotation=[
board_axis_x(1),board_axis_y(1),board_axis_z(1);
board_axis_x(2),board_axis_y(2),board_axis_z(2);
board_axis_x(3),board_axis_y(3),board_axis_z(3)];
board2World_translation=base_p1;

%此处flange_pos_list是
%拍摄照片时在机械臂示教器上读取到的机械臂Base坐标系中的法兰坐标系原点位置
flange_pos_list=[0.76937,0.04262,0.27908,177.90,7.64,-179.74;
0.76937,0.04262,0.27908,-178.41,7.59,-152.10;
0.76937,0.04262,0.27908,-176.51,6.92,-137.15;
0.76937,0.04262,0.27908,-173.86,4.74,-111.55;
0.76937,0.04262,0.27908,176.53,6.93,169.31;
0.76937,0.04262,0.27908,173.66,4.47,141.00;
0.77081,-0.00515,0.27085,165.98,-2.17,135.42;
0.77081,-0.00515,0.27085,165.84,0.94,147.77;
```

```
0.76487,0.08984,0.27012,-167.91,1.34,-140.03;
0.76245,0.12367,0.27247,-163.69,-7.56,-114.87;
0.76245,0.12367,0.27247,-162.14,-1.67,-133.89;
0.76245,0.12367,0.27247,-164.22,8.63,-167.02;
0.76245,0.12367,0.27247,-165.70,10.94,-175.73;
0.757,0.13375,0.26885,-166.65,-17.94,-87.49;
0.75852,0.12484,0.26885,-170.15,-21.82,-78.37;
0.75852,0.12484,0.26885,-176.36,-23.58,-62.35;
0.75586,0.07706,0.25178,174.27,0.15,65.32;
0.75568,0.07706,0.25178,174.45,-1.42,49.51;
0.75787,0.05014,0.25171,178.76,-5.10,41.29];
length(flange_pos_list)
%加载matlab标定工具箱标定得到的相机参数
load("2.mat");
%initial res
%法兰相对于相机的旋转平移矩阵，因为图片有19张，我们可以算出19组
flange2Cam_list=zeros(4,4,length(flange_pos_list));

for i=1:length(flange_pos_list)
flange_pos=flange_pos_list(i,:);
flange2Cam_list(4,4,i)=1;
%board-->cam 标定板相对于相机的旋转平移矩阵由标定给出，
%load("2.mat");这一命令已经导入参数，此时我们可以调用

board2Cam_rotation=cameraParams.RotationMatrices(:,:,i)';
board2Cam_translation=0.001*cameraParams.TranslationVectors(i,:);
%translation(3,1):cam-->board-->world
%计算相机相对于机械臂Base坐标系的旋转平移矩阵，
%这里需要运用线性代数的知识，最好画出草图手动推导一下
cam2World_translation=board2World_rotation*board2Cam_rotation'* ...
(-board2Cam_translation')+board2World_translation';
cam2World_rotation=board2World_rotation*board2Cam_rotation';
%translation(1,3):flange pos in world
%法兰坐标系原点在机械臂Base坐标系上的平移量
flange2World_translation=flange_pos(1:3);
%rotation:flange-->world法兰坐标系原点在机械臂Base坐标系上的旋转量
```

```
flange2World_rotation= ...
eul2rotm(deg2rad([flange_pos(6),flange_pos(5),flange_pos(4)]));
%flange2Cam_translation=flange2World_rotation'*
%(cam2World_translation'-flange2World_translation)';
%rotation:flange-->world-->board-->cam
%计算法兰坐标系相对于相机坐标系的旋转平移量，
%通过此量可以求出我们要求的摄像机坐标系在机械臂Base坐标系的位置
flange2Cam_rotation=board2Cam_rotation*board2World_rotation' ...
*flange2World_rotation;
flange2Cam_translation=flange2Cam_rotation*flange2World_rotation' ...
*(cam2World_translation'-flange2World_translation)';
%save it 将数据保存
flange2Cam_list(1:3,4,i)=flange2Cam_translation;
flange2Cam_list(1:3,1:3,i)=flange2Cam_rotation;
end
%average 因为每张图片我们都可以求得一个值，我们可以将所有结果平均化
flange2Cam_sum=zeros(4,4);
for i=1:length(flange_pos_list)
flange2Cam_sum=flange2Cam_sum+flange2Cam_list(:,:,i);
end
flange2Cam=flange2Cam_sum/length(flange_pos_list);
%schrmit 对旋转矩阵正交化
flange2Cam(1:3,1)=flange2Cam(1:3,1)/norm(flange2Cam(1:3,1));
flange2Cam(1:3,2)=flange2Cam(1:3,2)-flange2Cam(1:3,1)'* ...
flange2Cam(1:3,2)*flange2Cam(1:3,1);
flange2Cam(1:3,2)=flange2Cam(1:3,2)/norm(flange2Cam(1:3,2));
flange2Cam(1:3,3)=cross(flange2Cam(1:3,1),flange2Cam(1:3,2));
disp(flange2Cam)
%此处就是我们手眼标定所求得的结果


%%%
```

```
%%%
```

```
board_axis_z(1),board_axis_z(2),board_axis_z(3);
1/direction_in_world(1),-1/direction_in_world(2),0;
1/direction_in_world(1),0,-1/direction_in_world(3)
];
b=[
base_p1*board_axis_z';
(start_point_pos(1)/direction_in_world(1))-  ...
(start_point_pos(2)/direction_in_world(2));
(start_point_pos(1)/direction_in_world(1))-  ...
(start_point_pos(3)/direction_in_world(3));
];
point_in_world=A\b;
%point_in_world=start_point_pos'+
%((board_axis_z*(base_p1-start_point_pos)')/
%(board_axis_z*direction_in_world))*direction_in_world;
PP_LIST(i,:)=point_in_world';
end



%%%
```

```
%       disp(start_point_pos)
%camera-->flange-->world
%direction_in_world=flange2World_rotation*flange2Cam(1:3,1:3)'*
%point_in_cam01'-flange2World_rotation*
%flange2Cam(1:3,4)+flange2World_translation';
cam2World_rotation=flange2World_rotation*flange2Cam_rotation';
direction_in_world=cam2World_rotation*point_in_cam01';
%calculate eqs
A=[
board_axis_z(1),board_axis_z(2),board_axis_z(3);
1/direction_in_world(1),-1/direction_in_world(2),0;
1/direction_in_world(1),0,-1/direction_in_world(3)
];
b=[
base_p1*board_axis_z';
(start_point_pos(1)/direction_in_world(1))- ...
(start_point_pos(2)/direction_in_world(2));
(start_point_pos(1)/direction_in_world(1))- ...
(start_point_pos(3)/direction_in_world(3));
];
point_in_world=A\b;
disp(point_in_world)
%%%
```

```
offset=flange2Cam(1:3,1:3)'*(flange2Cam(1:3,4));
offset_in_world=flange2World_rotation*offset;
start_point_pos=flange_pos(1:3)+offset_in_world';
%       disp(start_point_pos)
%camera-->flange-->world
%direction_in_world=flange2World_rotation*flange2Cam(1:3,1:3)'*
%point_in_cam01'-flange2World_rotation*flange2Cam(1:3,4)+
%flange2World_translation';
cam2World_rotation=flange2World_rotation*flange2Cam_rotation';
direction_in_world=cam2World_rotation*point_in_cam01';
%calculate eqs
A=[
board_axis_z(1),board_axis_z(2),board_axis_z(3);
1/direction_in_world(1),-1/direction_in_world(2),0;
1/direction_in_world(1),0,-1/direction_in_world(3)
];
b=[
base_p1*board_axis_z';
(start_point_pos(1)/direction_in_world(1))- ...
(start_point_pos(2)/direction_in_world(2));
(start_point_pos(1)/direction_in_world(1))- ...
(start_point_pos(3)/direction_in_world(3));
];
point_in_world=A\b;
disp(point_in_world)
```

至此手眼标定的内容告一段落，接下来我们从三维重建开始讲起。

9.4 机械臂-视觉系统实战之三维重建

下面简单介绍三维重建方法：目前，三维重建技术已经在游戏、电影、测绘、定位、导航、自动驾驶、虚拟现实/增强现实、工业制造以及消费品领域等方面取得了广泛应用。这些方法的种类繁多，按照原理可分为两大类：

- 基于传统多视图几何的三维重建算法
- 基于深度学习的三维重建算法

总体而言，尽管目前传统的三维重建算法仍然占据研究的主导地位，然而越来越多的研究者开始专注于利用卷积神经网络（CNN）来探索三维重建，或者说，两者之间的交叉与融合也成为研究的焦点。

动物跟人类直接基于大脑而非严格的几何计算来进行物体的三维重建，那么直接基于深度学习的方法在原理上也是可行的。特别需要注意的是，在一些研究中，有些方法直接基于单张图像（非单目，单目指利用单个摄像头）进行三维重建。

理论上讲，单张图像已经丢失了物体的三维信息，因此在原理上即不能恢复深度信息，但是人类又能凭借经验大致估计物体的距离，因而也具有一定的“合理性”。

两者形成了各自的理论和体系，但未来三维重建领域研究一定是传统优化方法与深度学习的结合。目前，这方面研究仍处于起步阶段，还有许多问题亟待解决。

首先我们来简单介绍一下基于传统多视图几何的三维重建算法。传统的三维重建算法按传感器是否主动向物体照射光源可以分为**主动式**和**被动式**两种方法。

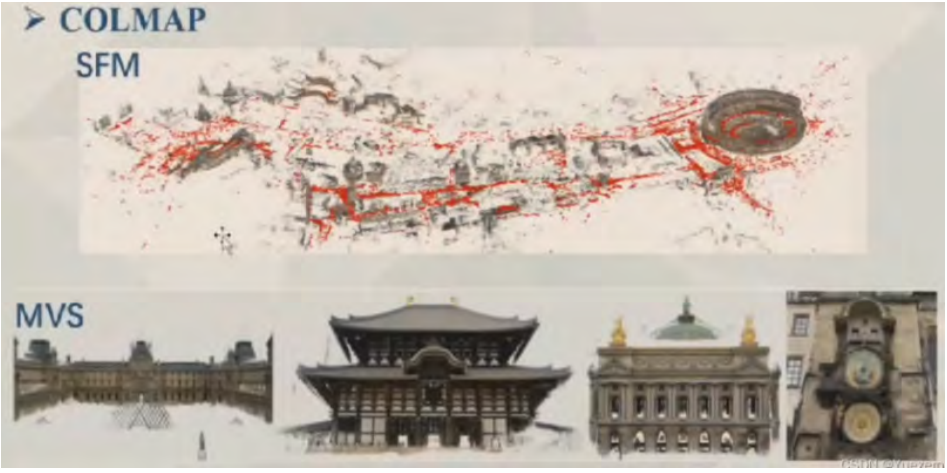
主动式，指通过传感器主动地向物体照射信号，然后依靠解析返回的信号来获得物体的三维信息，常见的有：

- 结构光

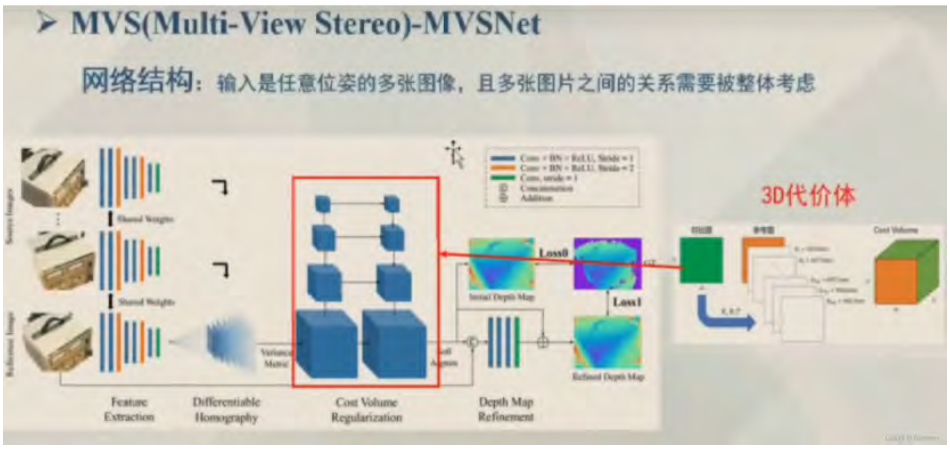
- TOF 激光飞行时间法

被动式，直接依靠周围环境光源来获取 RGB 图像，通过依据多视图几何原理对图像进行解析，从而获取物体的三维信息。常见的依据原理可以分为：

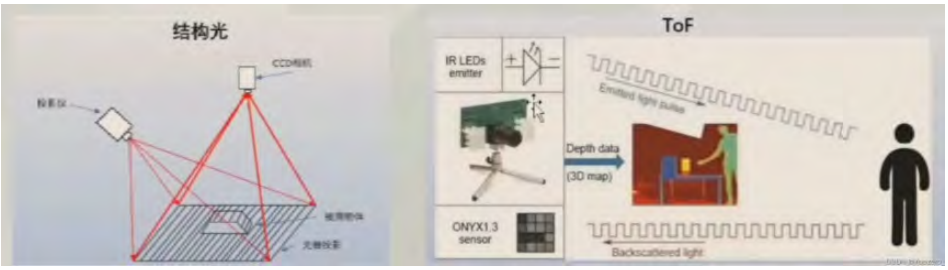
- 单目视觉



- 双目/多目视觉



-基于消费级 RGB-D 相机



深度相机 (RGB-D Camera): 可以直接获取被测对象的三维信息 (深度图) 的相机。

深度相机具有一个 RGB 相机和一个深度传感器，相比于单独使用深度传感器有如下优势：首先可以显示场景的 RGB 图像让人们有更直观的认知；其次可以将点云配准和二维图像特征点匹配相结合提高配准精度；最后可以将彩色值用于纹理贴图使重建模型具有颜色属性，增加重建的真实度和还原度。

可以看见三维重建的方法繁多，在本项目中我们采用的是单目相机。下面介绍一种利用单目相机进行三维重建的方法。

9.5 基于 Context capture 三维重建

Context capture , 是一款具有突破性的摄影测量产品，能够将来自于从智能手机到激光雷达的任何源头的影像数据，处理成与设计建模和施工建模环境中的背景相匹配的表示形式。

Context capture 建模优势：

(1) 快速，简单，全自动

Context capture 能无需人工干预地从简单连续影像中生成最逼真的实景真三维场景模型。无需依赖昂贵且低效率的激光点云扫描系统或 POS 定位系统，仅仅依靠简单连续的二维影像就能还原出最真实的实景真三维模型。

(2) 身临其境的实景真三维模型

Context capture 不同于传统技术仅仅依靠高程生成的缺少侧面等结构的 2.5 维模型，Context capture 可运算生成基于真实影像的超高密度点云，并以此生成基于真实影像位纹理的高分辨率实景真三维模型，对真实场景在原始影像分辨率下的全要素级别的还原达到了无限接近真实的极致。

(3) 广泛的数据源兼容性

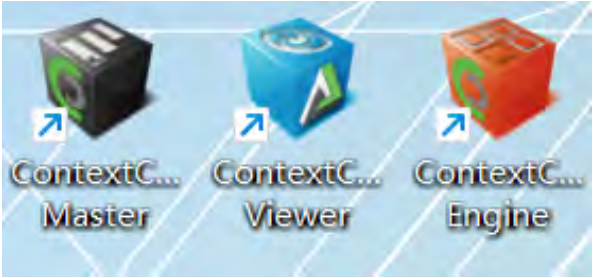
Context capture 能接受各种硬件采集的各种原始数据，包括大型固定翼飞机，载人直升机，大中小型无人机，街景车，手持式数码相机甚至手机，并直接把这些数据还原成连续真实的三维模型，无论大型海量城市级数据，还是考古级精细到毫米的模型，都能轻松还原出最接近真实的模型。

(4) 优化的数据格式输出

Context capture 能够输出包括 obj,osg(osgb),dae 等通用兼容格式能够方便地导入各种主流 GIS 应用平台，而且它能生成超过 20 级金字塔级别的模型精度等级，能够流畅应对本地访问或是基于互联网的远程访问浏览。

图 9.18: Context capture 建模效果图

Context capture 软件由三部分组成包括 Master(主控台)、Engine(引擎) 、Viewer(浏览)。



1、Master(主控台) 是一个非常友好的人机交互界面，相当于一个管理者，它创建任务，管理任务，监视任务的进度等。

2、Engine (引擎端)，只负责对所指向的 Job Queue 中的任务进行处理，可以独立于 Master 打开或者关闭。

3、Viewer(浏览) 则可预览生成的三维场景和模型。

Context capture 软件的使用大致有下面几个步骤:

1. 获取图像 (对该物体不同角度的照片图像)

2. 检查图像 (软件会检查图像格式内容是否满足计算条件)

3. 提交照片进行空三运算。(即空中三角测量计算，这一步类似于相机的自标定，软件会通过算法自动计算出提交图片的内、外参参数，以及图片的畸变系数。空三运算完以后会提取到每张图片中的特征点以及特征点的匹配关系，从而算得稀疏点云)

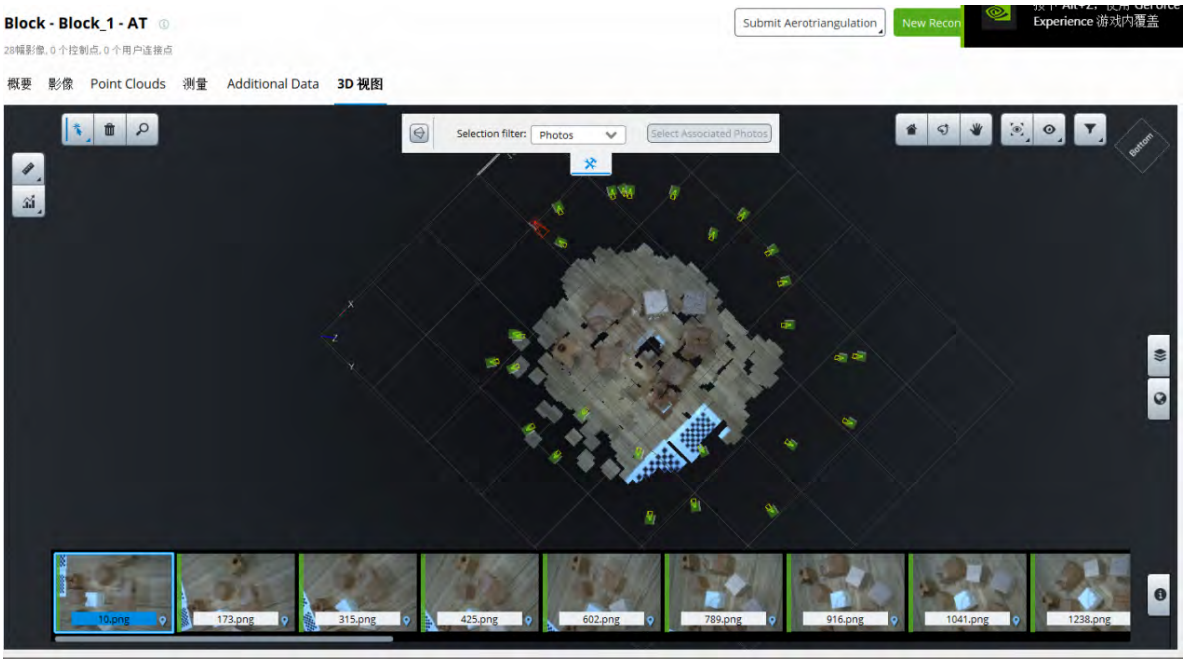


图 9.19: 图中绿色就代表空三计算得到的每张图片对应的相机位置 (图片数量 28 张)

4. 稠密重建

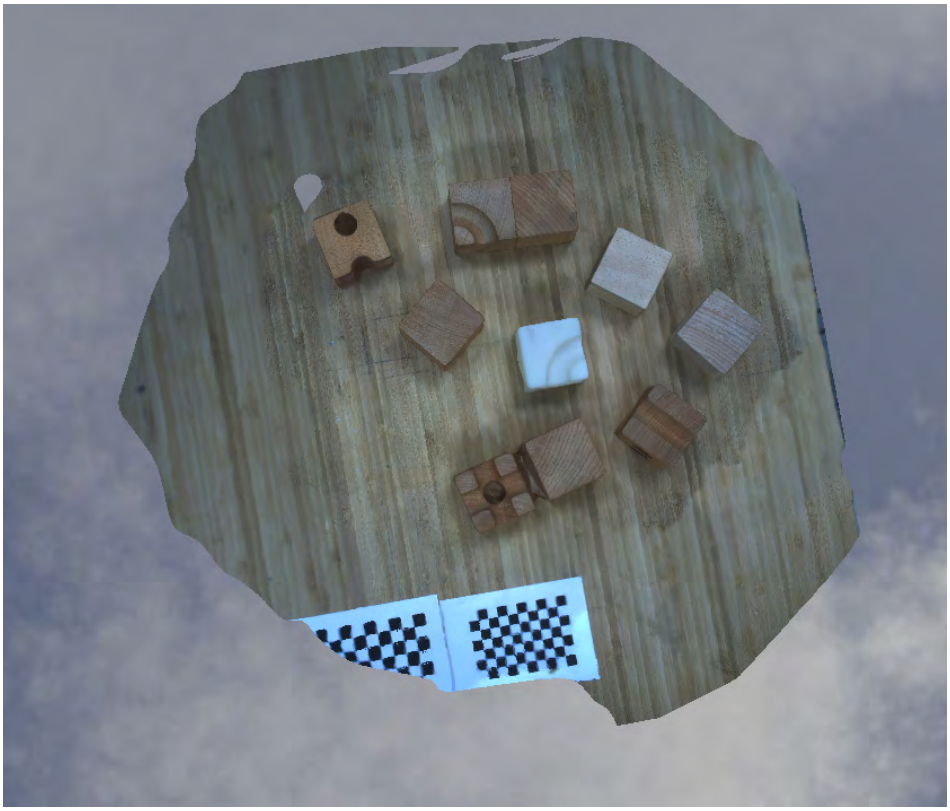


图 **9.20:** Context Capture 三维重建效果

图 **9.21:** 重建时长 10.5min

Context Capture 的重建效果十分出色，尤其是在积木表面，纹理清晰可见。即使是纯色白积木表面这类难以重建的场景，其处理效果同样令人印象深刻，真实感非常

强。

图 **9.22:** Context Capture 木质纹理（重建时长 10.5min）

而对比同类型的建模软件 colmap，同样的 28 张重建时间需要 20 分钟，时长大致是 Context Capture 的两倍，并且对纹理贴图的细节处理方面效果也弱于 Context Capture，积木侧边出现空洞，对于白积木的重建也不尽如人意。

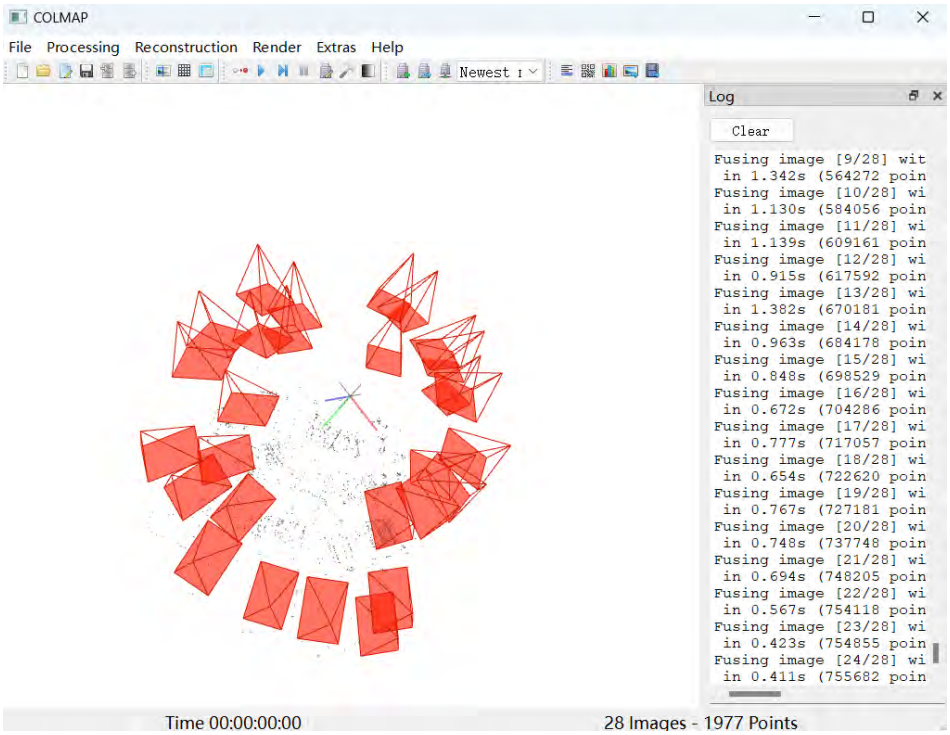


图 **9.23:** Colmap 相机标定

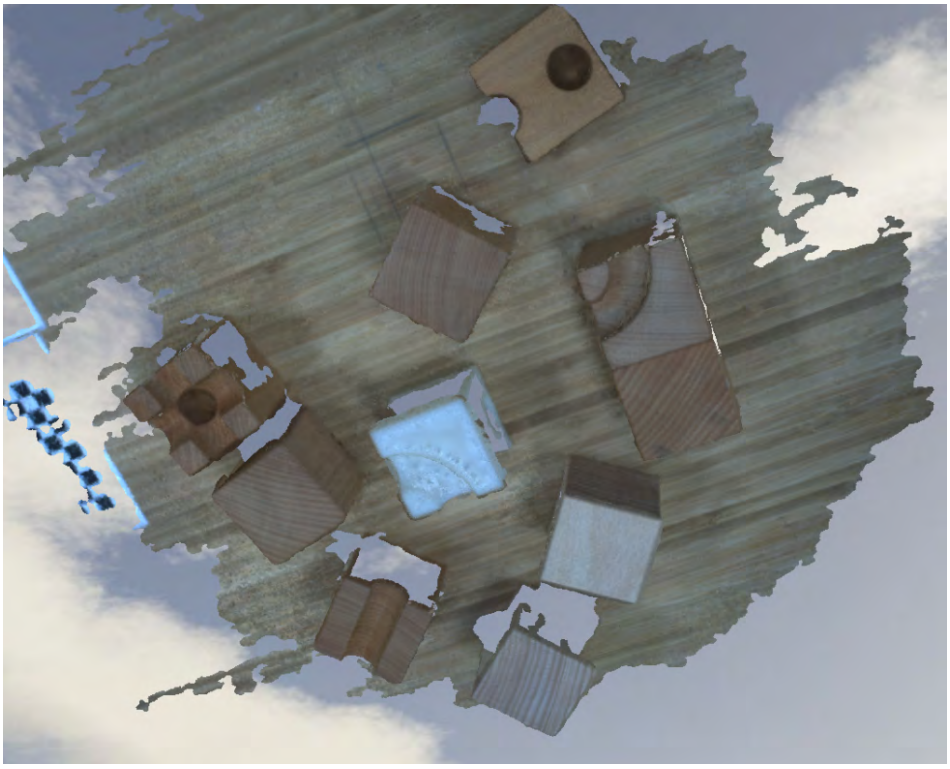


图 9.24: Colmap 木质纹理（重建时长 20min）

软件上的操作十分简单，有兴趣的读者可以查阅相关资料自行摸索，我们直接介绍 Context capture 软件 SDK 的应用。

9.6 Context capture 软件 SDK 应用

我们建议在虚拟环境中配置运行 SDK。

1. 首先需要安装 Anaconda(Anaconda 的安装教程请自行查找。)

2. 在 CMD 命令行通过 conda create -n Smart3D python=3.6 命令来创建虚拟环境。

(注：Context capture SDK 的 python 环境必须是 3.6，如果在其他版本的 python 下会导致安装失败。)

```
C:\Users\asus>conda create -n Smart3D1 python=3.6
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 22.9.0
  latest version: 24.1.2

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: E:\anaconda\envs\Smart3D1

  added / updated specs:
    - python=3.6
```

图 9.25: 创建 Smart3D 虚拟环境

```
The following NEW packages will be INSTALLED:

  certifi            pkgs/main/win-64::certifi-2021.5.30-py36haa95532_0 None
  pip                pkgs/main/win-64::pip-21.2.2-py36haa95532_0 None
  python             pkgs/main/win-64::python-3.6.13-h3758d61_0 None
  setuptools         pkgs/main/win-64::setuptools-58.0.4-py36haa95532_0 None
  sqlite             pkgs/main/win-64::sqlite-3.41.2-h2bbff1b_0 None
  vc                 pkgs/main/win-64::vc-14.2-h21ff451_1 None
  vs2015_runtime     pkgs/main/win-64::vs2015_runtime-14.27.29016-h5e58377_2 None
  wheel              pkgs/main/noarch::wheel-0.37.1-pyhd3eb1b0_0 None
  wincertstore       pkgs/main/win-64::wincertstore-0.2-py36h7fe50ca_0 None


Proceed ([y]/n)?
```

Proceed 输入 y 即可。

```
C:\Users\asus>conda info --envs
# conda environments:
#
base                  E:\anaconda
Smart3D               E:\anaconda\envs\Smart3D
Smart3D1              E:\anaconda\envs\Smart3D1
labelme               E:\anaconda\envs\labelme
mmrotate              E:\anaconda\envs\mmrotate
```

图 9.26: 显示已创建的虚拟环境

当命令行执行结束的时候输入命令 conda info –envs 若跳出我们所安装的虚拟环境 Smart3D 则安装成功。

3. 我们找到 Context Capture 的安装目录，在其 sdk-dist 文件目录下有 sdk 库的安装包 ccmasterkernel-10.20.0.4117-cp36-cp36m-win_amd64.whl。

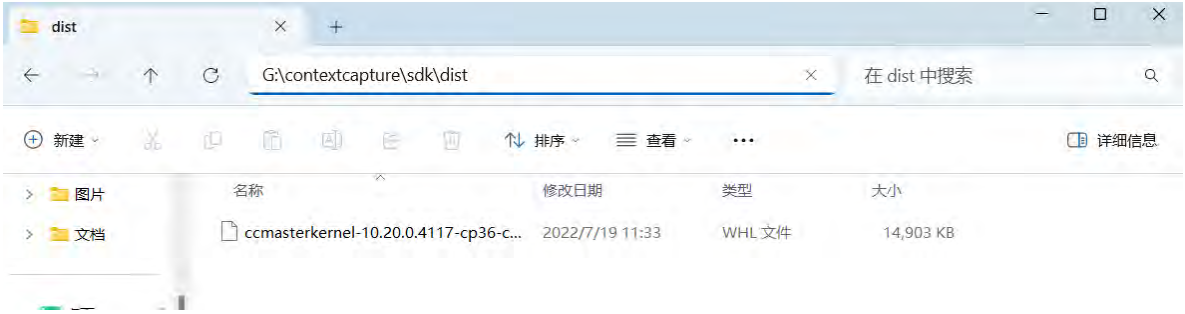


图 9.27: Context capture 安装包

在当前路径中输入 cmd 然后回车在终端中打开。

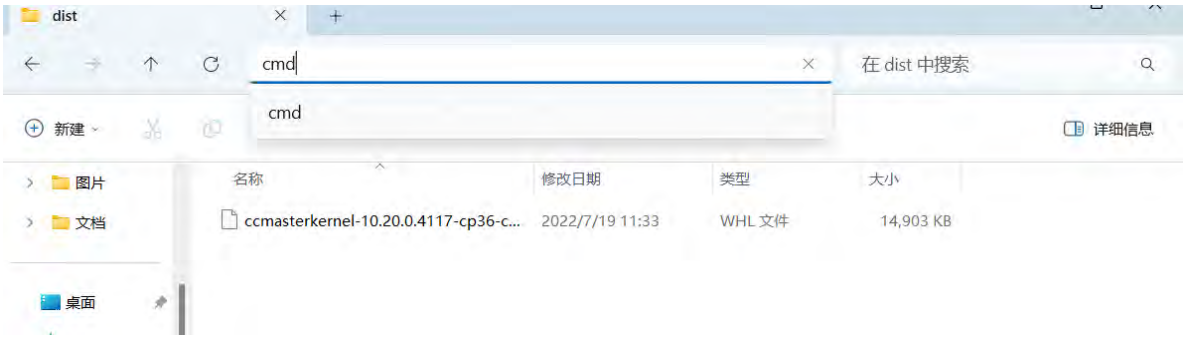


图 9.28: 在当前路径输入 cmd

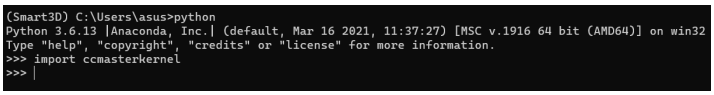
我们需要通过 conda activate Smart3D 命令切换到刚刚创建好的虚拟环境 Smart3D 此时只需要输入命令:

pip install ccmasterkernel-10.20.0.4117-cp36-cp36m-win_amd64.whl

即可顺利配置环境。

图 9.29: 激活虚拟环境

在虚拟环境中输入 python，如果 import ccmasterkernel 没有报错，则环境配置成功。



配置好环境之后可以在 context capture/sdk/samples 路径下找到 sdk 调用的示例代码，我们这里主要修改 automaster.py。

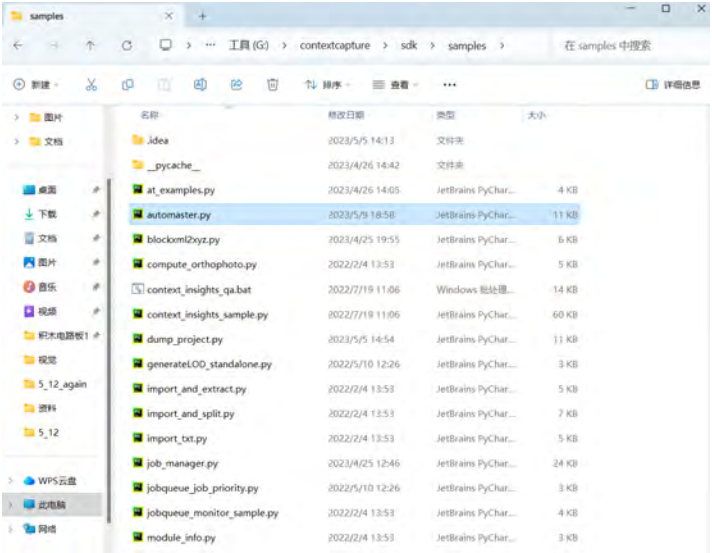
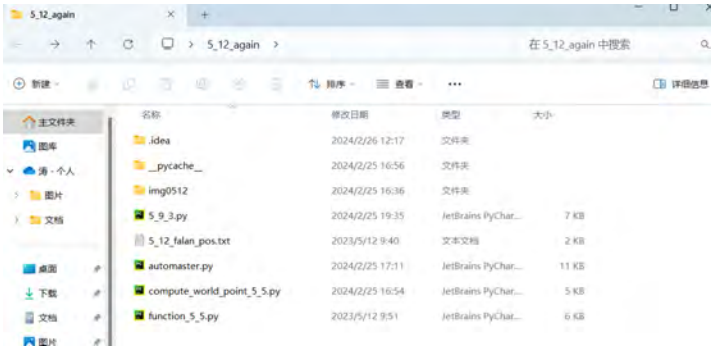


图 9.30: Context Capture demo

我们创建自己的项目文件夹，将 automaster.py 复制到文件夹下，img0512 文件夹中存放的是上文中重建的 28 张积木图片。



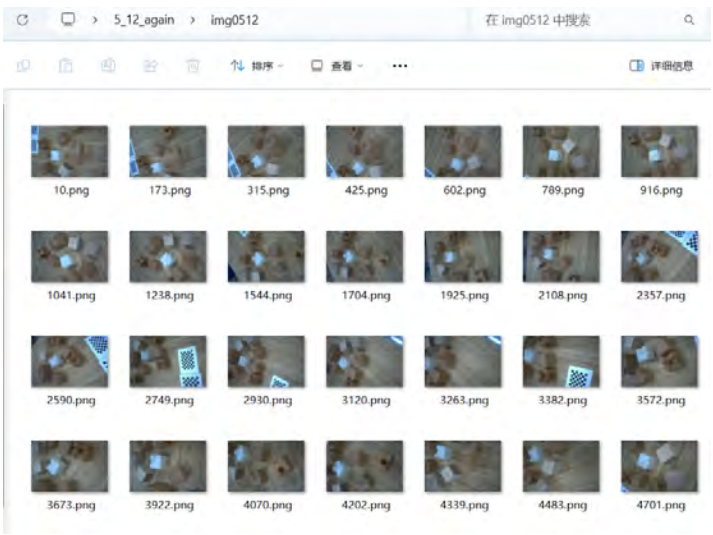


图 9.31: 用于重建的图片

9.6.1 代码解析

我们找到 automaster.py 代码中的第 24,25 行

修改照片文件夹路径 photosDirPath，项目文件夹路径 projectDirPath

automaster.py 5_9_3.py compute_world_point_5_5.py function_5_5.py

```
10  #
11  # Script description:
12  #    This script browses an input photo directory and manages a complete reconstruction project with
13  #    from these photos including:
14  #    - AT,
15  #    - Reconstruction,
16  #    - Production in OBJ format.
17  # ******************************************************************************
18
19  import sys
20  import os
21  import time
22  import ccmasterkernel
23  # Note: replace paths with your references, or install SDK sample data in C:/CC_SDK_DATA
24  photosDirPath = 'C:/Users/asus/Desktop/5_12_again/img0512'
25  projectDirPath = 'C:/Users/asus/Desktop/5_12_again'
```

代码总共分为 6 个部分

1.create project，

```
36  # ------------------------------------------------------------------------
37  # create project
38  # ------------------------------------------------------------------------
39  projectName = os.path.basename(projectDirPath)
40
41  project = ccmasterkernel.Project()
42  project.setName(projectName)
43  project.setDescription('Automatically generated from python script')
44  project.setProjectFilePath(os.path.join(projectDirPath, projectName))
45  err = project.writeToFile()
46  if not err.isNone():
47      print(err.message)
48      sys.exit(0)
```

2.create block，

```
53  # ------------------------------------------------------------------------
54  # create block
55  # ------------------------------------------------------------------------
56  block=ccmasterkernel.Block(project)
57  project.addBlock(block)
58
59  block.setName('block #1')
60  block.setDescription('input block')
61  photogroups = block.getPhotogroups()
62  files = os.listdir(photosDirPath)
```

3.AT 计算，

```
# ------------------------------------------------------------------
# AT
# ------------------------------------------------------------------
blockAT=ccmasterkernel.Block(project)
project.addBlock(blockAT)
blockAT.setBlockTemplate(ccmasterkernel.BlockTemplate.Template_adjusted, block)
```

4.Reconstruction 重建，

```
# ------------------------------------------------------------------
# Reconstruction
# ------------------------------------------------------------------
reconstruction = ccmasterkernel.Reconstruction(blockAT)
blockAT.addReconstruction(reconstruction)

if reconstruction.getNumInternalTiles() == 0:
    print('Error: Failed to create reconstruction layout.')
    sys.exit(0)

print('Reconstruction item created.')
```

5.Production 生成 PLY 格式，

```
# ------------------------------------------------------------------
# Production
# ------------------------------------------------------------------
production = ccmasterkernel.Production(reconstruction)
reconstruction.addProduction(production)
#我们设置生成的重建格式是ply
production.setDriverName('PLY')
production.setDestination( os.path.join(project.getProductionsDirPath(), production.getName()) )
driverOptions = production.getDriverOptions()
driverOptions.put_bool('TextureEnabled', True)
driverOptions.put_int('TextureCompressionQuality', 30)
driverOptions.writeXML( os.path.join(project.getProductionsDirPath(), "options.xml") )
```

6.Report 项目报告

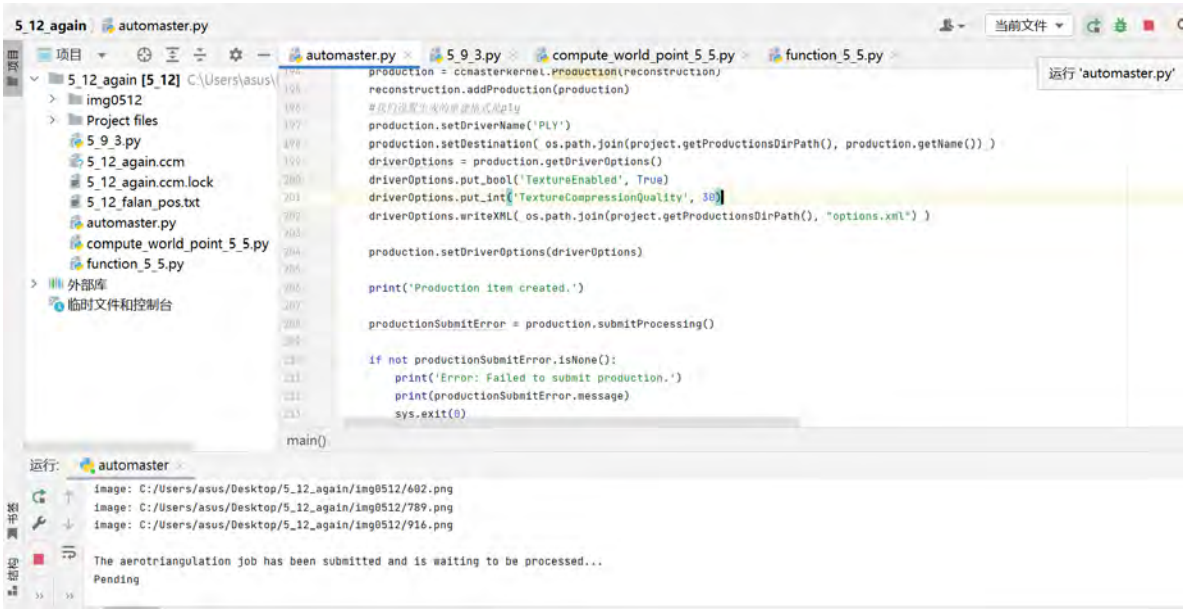
```
# ------------------------------------------------------------------
# Report
# ------------------------------------------------------------------
print('Production completed.')
print('Output directory: %s' % production.getDestination())
PLY = ccmasterkernel.bindings.ExportPointCloud.PLY
print(type(PLY))
q1 = blockAT.getPointClouds()
print(q1.getNumPoints)
```

我们这里需要修改 Production

```
# ------------------------------------------------------------------
# Production
# ------------------------------------------------------------------
production = ccmasterkernel.Production(reconstruction)
reconstruction.addProduction(production)
#我们设置生成的重建格式是ply，
```

```
#将production.setDriverName中的参数改为PLY即可
production.setDriverName('PLY')
production.setDestination(os.path.join(\
project.getProductionsDirPath(), production.getName()))
driverOptions = production.getDriverOptions()
#这里需要将第二个参数改成True，才能输出oly文件
driverOptions.put_bool('TextureEnabled', True)
#此函数可以调节输出点云的质量
driverOptions.put_int('TextureCompressionQuality', 30)
```

之后在 pycharm 中点击运行即可。



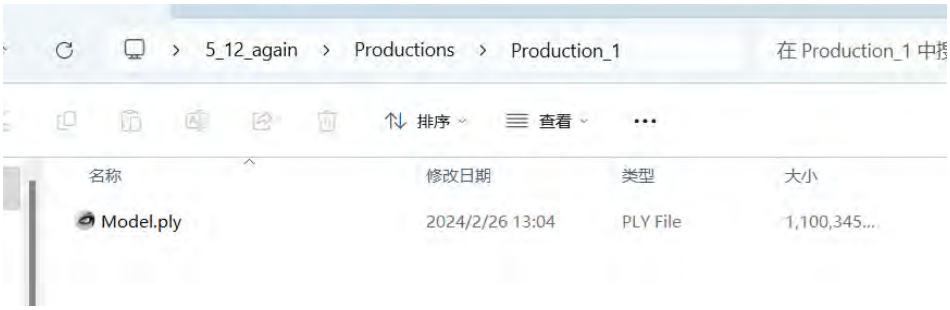
在运行过程中我们需要打开 Context Capture Engine

```
Aerotriangulation completed.
AT report available at C:/Users/asus/Desktop/5_12_again\Project files\Block_2\QualityReport\QualityReport.html
Ready for reconstruction.
Reconstruction item created.
Production item created.
The production job has been submitted and is waiting to be processed...
Pending
Running
2% - Initializing...
6% - Initializing...
7% - Initializing...
12% - Initializing...
27% - Initializing...
34% - Initializing...
95% - Initializing...
Completed

Production completed.
Output directory: C:/Users/asus/Desktop/5_12_again\Productions\Production_1
```

在 pycharm 终端命令行显示的路径下我们可以找到重建好的 ply 文件，并且项目文件夹下会多出 Productions 文件夹和 Project files 文件夹，还有 5_12_again.ccm 工程文件。

Output directory: C:/Users/asus/Desktop/5_12_again/Productions/Production_1



此时算出的 ply 点云坐标是 Context Capture 算得的坐标，我们需要将其转换到机械臂 Base 坐标下才能实现机械臂抓取功能。

在项目文件夹中 5_9_3.py 是我们实现坐标转换功能以及点云处理的主函数，py 用来坐标计算的调用函数，function_5_5.py 用来提取 Context Capture 生成的 5_12_again.ccm 工程文件中的相机内外参。

```
    K = np.array([[fx, 0, principal[0]],
                  [0, fx, principal[1]],
                  [0, 0, 1]])
    M=np.zeros([int(len(posecenter)/3),3,4])
    poserotation=np.array(poserotation)
    posecenter=np.array(posecenter)
    for i in range(int(len(posecenter)/3)):
        #相机相机/cc
        M[i,0:3,0:3]=poserotation[0+i*9:9+i*9].reshape(3,3).T
        M[i,0:3,3]=posecenter[0+i*3:3+i*3]
    return posecenter, poserotation, K, k1, k2, k3, p1, p2,M


posecenter, poserotation, K, k1, k2, k3, p1, p2,M=function(r'C:\Users\asus\Desktop\5_12_again\5_12_again.ccm')
```

图 9.32: function_5_5.py 提取到每张图片的摄像机内外参

我们先来介绍 function_5_5.py 的作用，function_5_5.py 可以提取到每张图片的摄像机坐标系中心点，相对于世界坐标系的旋转矩阵，以及摄像机内参矩阵，和畸变系数，这些参数都是在 AT 计算过程生成的。

然后我们着重讲一下 compute_world_point_5_5.py 计算坐标的原理。我们先分析一下已知条件，我们通过之前的手眼标定算出了相机相对于法兰的旋转平移矩阵:

```
Cam2flange_transport=np.array([127.165898017929,0.402238432010192,42.2654327570533])
Cam2flange_rotation=np.array([[0.852167525920280,   0.00548851257250702,    -0.523240273676109],
                              [-0.00616715971152809, 0.999980883779632,  0.000445215002137392],
                                [0.523232714867869  ,0.00284750856839022,   0.852185084231678]])
```

图 9.33: 摄像机相对法兰的旋转平移

以及相机的内参矩阵，由于我们使用的单目相机光圈焦距都是固定的，所以内参矩阵不会改变:

```
K=np.array([[3.485056092685247e+03,7.915947609449449 ,  1.524653816279419e+03],
         [0,3.481056219837574e+03, 1.046618681876424e+03],
         [0 , 0 ,1]])
```

图 9.34: 摄像机内参矩阵

并且我们知道拍摄 28 张照片时每一张照片的机械臂法兰坐标系原点在机械臂 Base 坐标系下的坐标:

options.xml　5_12_falan_pos.txt

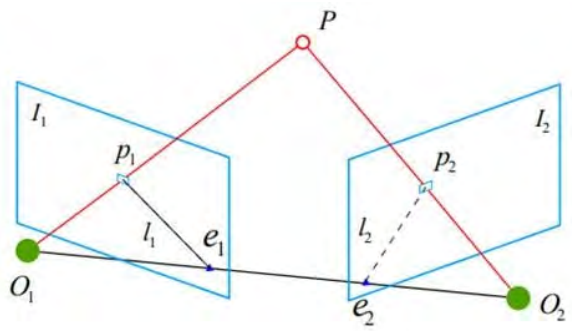
文件　编辑　查看

```
411.759857, 100.616745, 450.761993, -179.931366, -1.347248, 179.969620
411.759857, 100.616745, 450.761993, 158.134918, -1.261110, -179.524979
443.333160, 158.928833, 450.048706, 144.554367, -1.114339, -179.242157
443.333160, 158.928833, 450.048706, 130.490753, -0.896802, -178.994110
473.021606, 194.900055, 448.558136, 116.489990, -0.626793, -178.807022
544.108459, 238.989792, 446.908966, 97.464966, -0.203613, -178.667877
581.211609, 231.312149, 446.032471, 78.314636, 0.244735, -178.674820
603.242432, 220.078949, 445.508698, 62.995964, 0.586184, -178.786560
628.329102, 194.955643, 444.906067, 44.974083, 0.932853, -179.027435
492.817902, 27.268257, 448.013031, -137.435059, -0.973002, 179.067307
475.044891, -20.051136, 448.407227, -126.573730, -0.779824, 178.900681
550.961243, -35.574772, 446.613678, -101.551132, -0.241585, 178.674026
581.272522, -34.283195, 445.901337, -85.038994, 0.145314, 178.660049
705.321594, -20.738447, 442.990204, -52.266453, 0.847576, 178.952026
719.048340, 21.984840, 442.688843, -35.065418, 1.119566, 179.249527
797.644714, 55.392136, 440.857513, -8.342658, 1.337424, 179.833069
792.254272, 197.162292, 441.056488, 19.630049, 1.259508, -179.520157
742.140503, 256.729431, 442.266083, 36.940472, 1.059720, -179.167145
739.959412, 256.776154, 349.542786, 36.940472, 1.059720, -179.167145
763.178345, 130.849777, 348.933136, 10.453418, 1.319898, -179.727142
659.421448, 66.890236, 351.342133, -17.394775, 1.294520, 179.624756
628.433228, 46.055759, 352.060547, -56.100586, 0.775615, 178.897705
565.594055, 49.427261, 353.543915, -86.261871, 0.116832, 178.658310
472.363434, 7.037272, 355.713715, -113.076508, -0.501074, 178.749893
475.363953, 20.789789, 355.650177, -138.879898, -0.995282, 179.092651
445.108490, 147.076096, 356.425629, -166.515793, -1.302550, 179.657776
452.937256, 221.468704, 356.279388, 158.237747, -1.261289, -179.527863
559.953125, 211.463608, 353.759583, 112.974014, -0.549356, -178.770447
```

行 28，列 71　1,948 个字符

图 9.35: 法兰原点坐标

此时我们取出两张照片，在两张照片上找到积木上的同一点的像素坐标，我们就可以求得积木上这点在机械臂 Base 坐标系下的坐标。如下图所示:

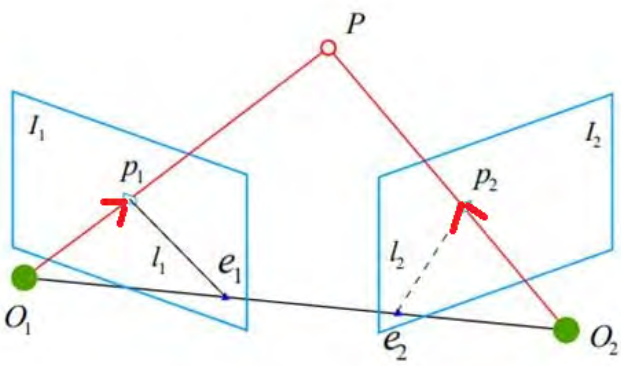


我们假设 P 点就是生成的 PLY 文件中积木上的一个点 (我们还不能得出其在机械臂 Base 坐标系下的位置信息)，O_1，O_2 分别表示在不同位置拍摄物体的相机坐标系的原点（即相机的位置，由于我们有机械臂作为我们的度量工具，因此相机的位置我们能够直接求出），I_1，I_2 表示拍摄的两张图片的像素坐标系，p_1，p_2 是 P 点在像素坐标系上的两个像素点。

我们设 $p_1 = \begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix}$，$p_2 = \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix}$，我们知道相机坐标系下的坐标可以通过相机内参矩阵 K 转换到像素坐标系下:$p = K * \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix}$

因此我们可以求得 p_1，p_2 像素点在相机坐标系下的坐标 (注: 内参矩阵 K 一定可逆):

$$\begin{pmatrix} X_{C1} \\ Y_{C1} \\ Z_{C1} \end{pmatrix} = K^{-1} \bullet p_1, \begin{pmatrix} X_{C2} \\ Y_{C2} \\ Z_{C2} \end{pmatrix} = K^{-1} \bullet p_2$$



此时我们得到在两个位置相机坐标系下的射线方向 O_1P_1，O_2P_2，可以通过左乘对应的旋转矩阵 $R_C^{Base} = R_C^{F_i} * R_{F_i}^{Base}, (i = 1, 2, 3,)$，（注: 因为法兰位置是会改变的，旋转矩阵一定要和图片对应起来），并且已知相机相对法兰的平移，法兰相对机械臂 Base 坐标系原点的平移，通过向量的加减法运算，易知 O_1，O_2 在机械臂 Base 坐标系上的坐标，有了原点坐标和射线方向，并且将其统一到了机械臂 Base 坐标系下，我们就可以联立方程，求解出 P 点在机械臂 Base 坐标系下的坐标。

下面是具体执行过程的代码：

```
# base_p1 = [0.80591,0.08940,0.02771];
# base_p2 = [0.76576,0.08950,0.028];
# base_p3 = [0.80547,0.02511,0.02797];
import numpy as np
import math
from numpy.linalg import lstsq # 解超定方程
def to_point(u1,v1,u2,v2,A_flange1,A_flange2):
Cam2flange_transport=np.array(\
[127.165898017929,0.402238432010192,42.2654327570533])
Cam2flange_rotation=np.array([[0.852167525920280,\
    0.00548851257250702,    -0.523240273676109],
[-0.00616715971152809,  0.999980883779632, \
0.000445215002137392],
[0.523232714867869   ,0.00284750856839022, \
 0.852185084231678]])
def eulerAngles2rotationMat(theta, format='degree'):
"""
Calculates Rotation Matrix given euler angles.
:param theta: 1-by-3 list [rx, ry, rz] angle in degree
:return:
RPY角，是ZYX欧拉角，依次 绕定轴XYZ转动[rx, ry, rz]
"""
if format == 'degree':
theta = [i * math.pi / 180.0 for i in theta]

R_x = np.array([[1, 0, 0],
[0, math.cos(theta[2]), -math.sin(theta[2])],
[0, math.sin(theta[2]), math.cos(theta[2])]
])
```

```
R_y = np.array([[math.cos(theta[1]), 0, math.sin(theta[1])],
[0, 1, 0],
[-math.sin(theta[1]), 0, math.cos(theta[1])]
])

R_z = np.array([[math.cos(theta[0]), \
-math.sin(theta[0]), 0],
[math.sin(theta[0]), math.cos(theta[0]), 0],
[0, 0, 1]
])
R = np.dot(R_z, np.dot(R_y, R_x))
return R
pixe1 = np.array([u1, v1, 1])
pixe2 = np.array([u2, v2, 1])
K=np.array([[3.485056092685247e+03,7.915947609449449\
 , 1.524653816279419e+03],
[0,3.481056219837574e+03, 1.046618681876424e+03],
[0 , 0 ,1]])
# 第一张图片像素点在camera系下坐标
point_in_cam01 = np.dot(np.linalg.inv(K),pixe1)
# 第二张图片像素点在camera系下坐标
point_in_cam02=np.dot(np.linalg.inv(K),pixe2)
####################flange1位姿（世界）
flange2World_translation_1 = A_flange1[0:3]
#x,y,z
theta1=A_flange1[3:6]
flange2World_rotation_1 = eulerAngles2rotationMat(theta1)
################flange2位姿（世界）
flange2World_translation_2 = A_flange2[0:3]
theta2=A_flange2[3:6]
flange2World_rotation_2 = eulerAngles2rotationMat(theta2)
#相机相对法兰平移
offset = Cam2flange_transport
# 在世界(第一张图)
offset_in_world_1 = np.dot(flange2World_rotation_1 ,offset)
# 在世界(第二张图)
offset_in_world_2 = np.dot(flange2World_rotation_2 ,offset)
```

```
# 第一张图光心位置
start_point_pos_1 = A_flange1[0:3]+offset_in_world_1
# 第二张图光心位置
start_point_pos_2 = A_flange2[0:3]+offset_in_world_2
# 第一张图相机相对世界旋转
cam2World_rotation_1 = np.dot(flange2World_rotation_1 ,\
 Cam2flange_rotation)
# 第二张图相机相对世界旋转
cam2World_rotation_2 = np.dot(flange2World_rotation_2 ,\
Cam2flange_rotation)
#第一张图光线方向
direction_in_world_1 = np.dot(cam2World_rotation_1, \
point_in_cam01)
#第二张图光线方向
direction_in_world_2 = np.dot(cam2World_rotation_2 , \
point_in_cam02)
#解方程
A = np.array([
[1 / direction_in_world_1[0], -1 / direction_in_world_1[1], 0],
[1 / direction_in_world_1[0], 0, -1 / direction_in_world_1[2]],
[1 / direction_in_world_2[0], -1 / direction_in_world_2[1], 0],
[1 / direction_in_world_2[0], 0, -1 / direction_in_world_2[2]]
])
b = np.array([
(start_point_pos_1[0] / direction_in_world_1[0]) - \
(start_point_pos_1[1] / direction_in_world_1[1]),
(start_point_pos_1[0] / direction_in_world_1[0]) - \
(start_point_pos_1[2] / direction_in_world_1[2]),
(start_point_pos_2[0] / direction_in_world_2[0]) - \
(start_point_pos_2[1] / direction_in_world_2[1]),
(start_point_pos_2[0] / direction_in_world_2[0]) - \
(start_point_pos_2[2] / direction_in_world_2[2])])
#解超定方程
x = lstsq(A, b, rcond=None)
# print(x[0])
return x[0]
A_flange1 = np.array([607.635620, 31.831078, 584.701355,\
```

```
 -179.101730, -10.134048, -179.931778])
A_flange2 = np.array([623.212646, 81.775261, 485.962860, \
162.917816, -9.627864, -176.821472])
```

有了上面两个代码的铺垫，我们可以得到最终的结果了。在上文中有一个非常重要的条件我们并没有解释，那就是 P 点对应的像素点 p_1，p_2 是如何产生的。

注意我们已经拥有的条件，在上文中我们通过 automaster.py 代码得到了 28 张图片的 Model.ply 文件，并且在 function_5_5.py 提取到每张图片的摄像机坐标系中心点，相对于世界坐标系的旋转矩阵，以及摄像机内参矩阵，和畸变系数，这些条件已经足够将三维重建点云坐标转换到像素坐标上了。下面一起看 5_9_3.py 代码解析:

```
from function_5_5 import function
import numpy as np
from compute_world_point_5_5 import to_point
import open3d as o3d
import math
```

从 function_5_5.py 文件引入 function 函数用于导入参数，导入 numpy 库用于矩阵计算，从 compute_world_point_5_5.py 文件引入 to_point 函数用于计算机械臂 Base 坐标系下的坐标，导入 open3d 用于点云处理。

下面介绍计算像素点函数，其他过程不再赘述。

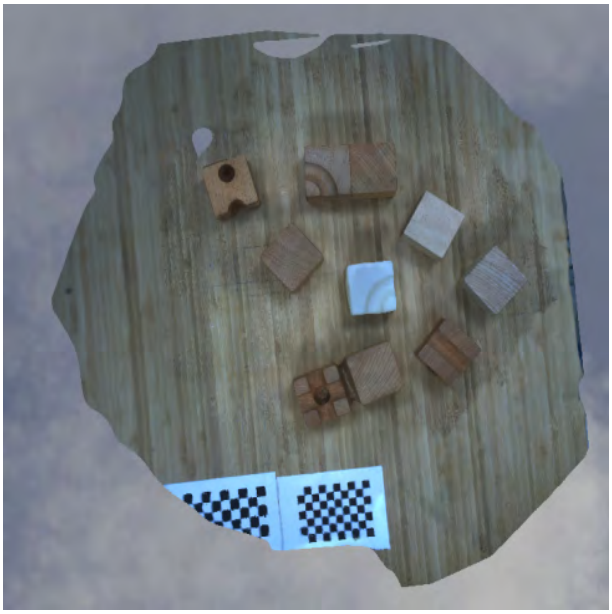
def compute_uv(M,point_in_cc): 得到的三维坐标是在 cc 世界坐标系上的点，只需乘上 $R_{cc_world}^{camera}$ 这一旋转矩阵就能转换到 camera 坐标系上，再乘上内参矩阵 K 就能够得到像素点。

```
def compute_uv(M,point_in_cc):
M_0=np.zeros([4,4])
M_0[3][3]=1
M_0[0:3,0:4]=M[0]
M_cc2cam=np.linalg.inv(M_0)[0:3,0:4]
u1,v1,_1=np.dot(np.dot(K,M_cc2cam),point_in_cc)
M_0=np.zeros([4,4])
M_0[3][3]=1
M_0[0:3,0:4]=M[1]
M_cc2cam=np.linalg.inv(M_0)[0:3,0:4]
u2,v2,_2=np.dot(np.dot(K,M_cc2cam),point_in_cc)
return u1/_1,v1/_1,u2/_2,v2/_2
point_list=[]
# 遍历所有点
i=0
```

```
point_in_world=[]
A_flange1 = np.array([411.759857, 100.616745, 450.761993, -179.931366, -
A_flange2 = np.array([603.242432, 220.078949, 445.508698, 62.995964, 0.58
for vertex in plydata.points:
i=i+1
if i%1000==0:
x = vertex[0]  # 获取X坐标
y = vertex[1]  # 获取Y坐标
z = vertex[2]  # 获取Z坐标
point_list+=[x,y,z]
point_in_cc = np.array([x, y, z, 1])
u1,v1,u2,v2=compute_uv(M,point_in_cc)
if to_point(u1 ,v1,u2 ,v2,A_flange1,A_flange2)[2]>60 and to_point(u1 ,v1
point_in_world.append(list(to_point(u1 ,v1,u2 ,v2,A_flange1,A_flange2)))
# point_in_world.append(list(to_point(u1, v1, u2, v2, A_flange1, A_flange
# 处理点数据...
max_z = max(point[2] for point in point_in_world)
min_z = min(point[2] for point in point_in_world)
print(max_z,min_z)
```

结果展示 (关于点云处理的聚类，凸包方法不再一一介绍, 点云数据截取了木板表面往上的一部分):

计算到的点云



重建的 ply 文件

上图是坐标转换之后的点云图，右边是 Context Capture 稠密重建之后的效果。下

图是点云聚类，根据颜色来划分种类个数。

图 9.36: 点云聚类

接下来根据点云的坐标来进行凸包计算:

图 9.37: 凸包计算

这里还有一个先验条件，我们已知积木尺寸大小为 48mm*48mm*48mm，所以我们可以根据先验条件将上图中的青绿色积木分离开来。

图 9.38: 并在一起的青绿色积木

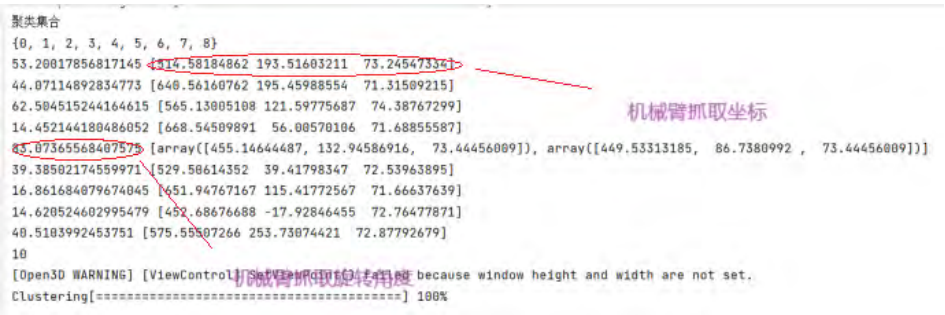


图 9.39: 计算机械臂抓取位置与角度

接下来让我们分析结果，聚类集合得到了 9 类的木块，我们根据先验条件得到了 10 块积木，并且得到了机械臂 Base 坐标系上的抓取坐标。至此我们可以通过单目相机识别并抓取积木了! 但显然此方法并没有满足我们的项目要求，暴露出了诸多问题:

1.Context Capture 三维重建花费时间较多，对于 28 张照片的重建时长就达到了 20 分钟以上。

2. 用 Context Capture 重建需要的硬件设备要求非常高，必须要具备显卡的设备才能快速运行。

3. 如果图片较少的情况下三维重建可能失败。

4. 仅用单目相机进行三维重建对相机的要求非常高，必须要得到清晰的像，并且镜头畸变要尽可能的小，否则可能损失精度。

5. 用此方法展示机械臂在拍好照片之后就会停止不动，显然不够优雅，显得十分笨拙。查阅相关资料，并且付诸实践之后我们找到了一种切实可行的方案——模版匹配。我们的项目也在各大展会上的展示中得到了检验。

图 **9.40:** 德清科技节

图 9.41: 第二十二届中国上海国际玩具展览会&潮流玩具展

第 10 章 KUKA 机器人的视觉积木抓取实战

10.1 视觉系统构成

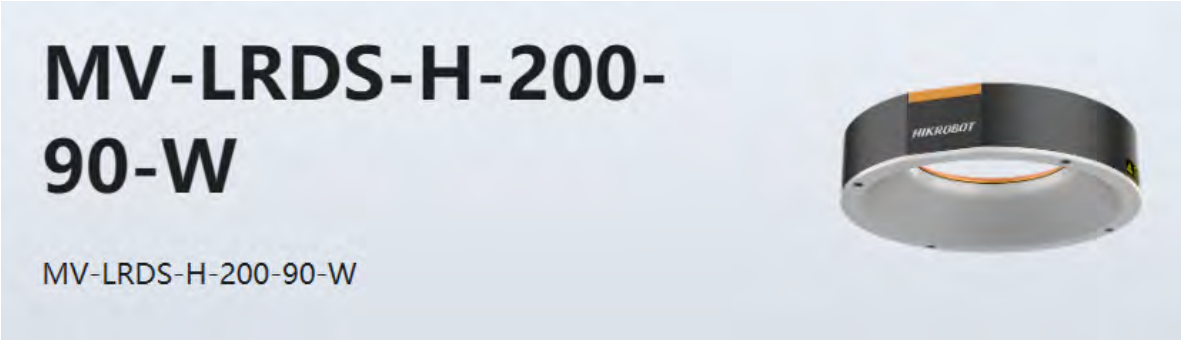
1. 工业相机方面我们选择了海康威视的 MV-CS-10UC 彩色面阵相机，具备出色的性能特征，包括高达 2448 × 2048 的分辨率、500 万像素以及最大帧率达 60.0fps。

MV-CS050-10UC	IMX264	2448 × 2048	60.0 fps	USB3.0		已发布

图 10.1: 工业相机参数

2. 在镜头选择方面，我们搭配了 MVL-MF0824M-5MPE 工业相机镜头，这款镜头以其卓越性能而脱颖而出。其高分辨率、画面清晰度一致、低畸变以及出色的相对照度使其在各种场景中表现出色。采用消色差设计，与彩色相机搭配使用效果更佳。此外，其在高低温环境下的稳定性出众，适用于严苛的工业环境。在微距拍摄方面，表现卓越，光学性能在近距离下同样令人满意。

3. 为了保持视觉检测环境的相对稳定状态，相机光源的选择同样至关重要。我们选择了 MV-LRDS-H-200-90-W，它具备紧凑的结构设计和优良的散热设计。此外，配备漫射板的设计有助于提升光照的均匀性，确保在各个检测点获得一致的照明效果。



4. 拥有工业相机、工业镜头和相机光源后，视觉控制器成为不可或缺的组成部分。MV-VC3301P-128G60 作为这个系统的“大脑”，搭载了桌面级 Intel 第八代 CPU，提供卓越的运算性能。其强大的扩展槽配置满足机器视觉常用图像采集接口的需求，保障性能的稳定性。此外，支持 GPIO 功能，输出支持 NPN/PNP 切换，采用 Intel 千兆网口，实现图像高速稳定传输。该控制器还具备可加装光源和 IO 扩展模块的能力，最多支持 2 块，进一步增强了系统的灵活性和功能性。

我们通过 3D 建模设计出了与机械臂法兰连接的工具，整合了工业相机、工业镜头、相机光源以及视觉控制器，从而成功打造了我们的“eye-in-hand”（眼在手上）模型。

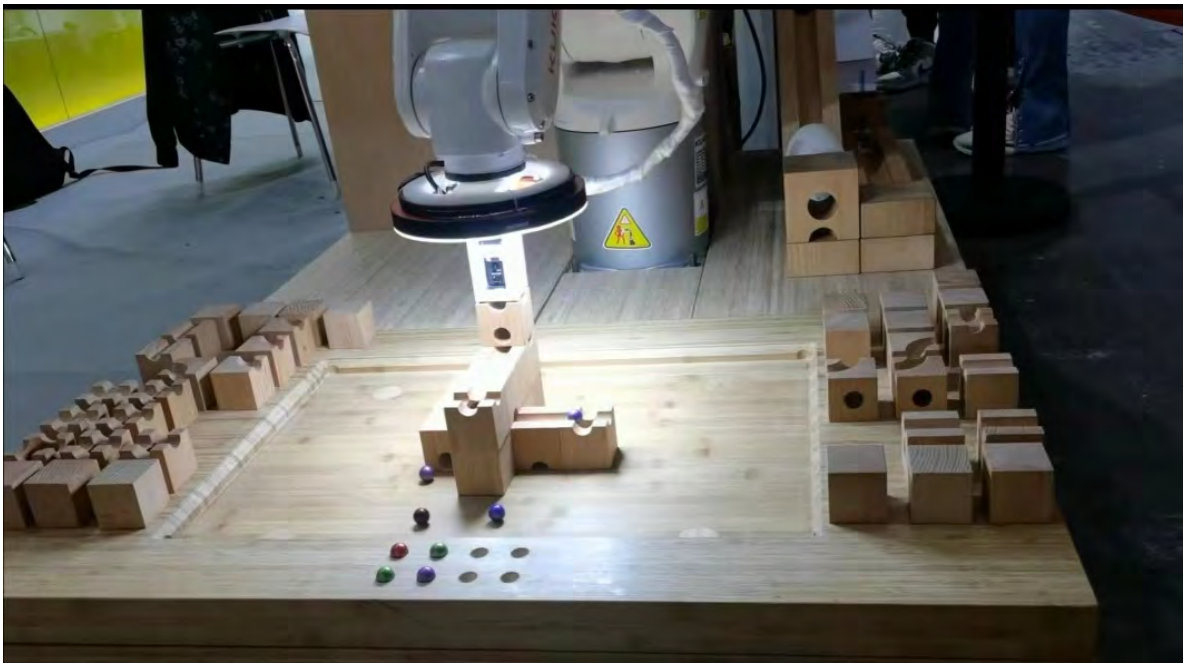
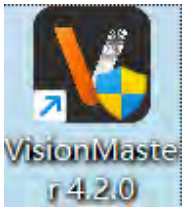


图 10.2: 视觉系统（工业相机、工业镜头、相机光源安装在机械臂末端，视觉控制器隐藏在木板下）

10.2 视觉系统软件部分——VisionMaster

我们在海康威视官网上找到并下载 VisionMaster 软件安装包，在视觉控制器上安装海康 VisionMaster 算法平台 4.2.0 版本。



我们对 VM 算法平台做一个简介:VM 算法开发平台是由海康机器人自主研发的机器视觉软件，专注于快速搭建视觉应用和解决视觉检测难题的算法工具。该平台支持多种机器视觉应用，包括视觉定位、尺寸测量、缺陷检测以及信息识别，搭建了相机图像处理和机械臂信息交互的桥梁。

当前市场上存在众多机器视觉算法平台,包括学习门槛较高的开源算法库 OpenCV、学习门槛次之的德国公司 MVTec 的 Halcon，以及诸如美国公司康耐视的 VisionPro、国内的海康、大华、维视、奥普特、汇川等公司开发的相关视觉算法平台。我们特别推荐海康的算法平台，原因如下：

1. 价格，国内企业大部分为价格敏感型企业，同样的产品，成本越低越好。在一般应用的场景，海康的算法平台已经足够完成项目需求，在目前海康的算法平台价格在上述提到的几家公司中是具备成本优势的。

2. 学习成本，VM 兼顾基础版和进阶版选手，基础版可实现零代码开发，对新手友好，学习成本低，快速开发部署；进阶版具备足够的开放性，可结合高级语言进行二次开发，实现定制化开发，学习成本相对较高。

10.3 VisionMaster 功能模块

VisionMaster 包含采集、定位、图像生成、测量、识别、标定运算、图像处理、拆分组合、颜色处理、缺陷检测、逻辑工具，通信 12 个功能模块。

在我们的项目中，主要运用了采集、定位、运算、图像处理、逻辑工具以及通信模块等关键模块。接下来，我们将重点围绕这些模块展开，以实现对小球的精准抓取。

10.3.1 VisionMaster——图像采集

在介绍采集功能之前我们需要配置 MV-CS-10UC 工业相机，首先将工业相机插入 USB3.0 接口之中，相机上侧蓝灯闪烁表示连接成功。

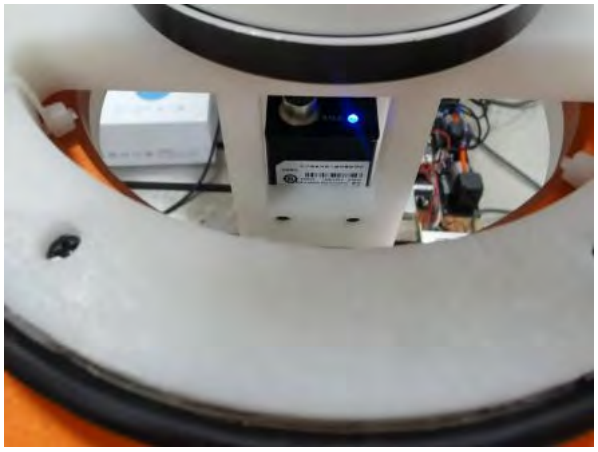


图 10.3: 相机 LED 闪烁

打开 VisionMaster 上排菜单栏中的相机管理:

点击相机管理界面的设备列表，选择全局相机并确定。

相机如果连接成功，则可以在选择相机一栏读到我们的相机型号，我们选择 MV-CS-10UC 工业相机。

随后，根据项目需求，我们进行了相机曝光参数、Gamma 使能等参数的调整，并对相机的光圈大小以及焦距进行了调节，确保拍摄到的图片在保持清晰度的同时满足项目要求。

图 10.4: 相机参数调整

接着修改相机管理-> 触发设置-> 触发源为 SOFTWARE。

现在我们可以选择采集模块中的图像源，将其拖入到流程界面，得到 [0 图像源 1]。

双击 [0 图像源 1] 修改:

图像源参数-> 相机

关联相机-> 全局相机 1(上文创建)

控制曝光-> 全局变量 e

SN 初始值->10000

输出 Mono8-> 打开 (图像必须要输出为 Mono8 格式，否则后续的图像处理会失效)

然后点击 VisionMaster 上排菜单栏中的连续执行，

此时 [0 图像源 1] 变绿，双击 [0 图像源 1 图标]，在右侧显示界面成功显示相机视频流。

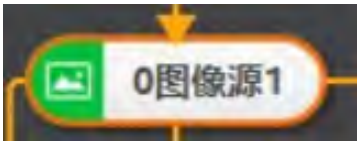


图 10.5: 获取到的相机视频流

10.3.2 VisionMaster——全局变量

在介绍下个功能模块之前我们先简单了解一下 VisionMaster 中的全局变量功能。全局是指一切从系统整体及全过程出发的思想。这种全局概念同样应用在 VisionMaster 中，在 VisionMaster 的快捷工具条中有一组全局工具，例如相机管理、控制器管理、全局变量、通信管理、全局触发、全局脚本等，我们主要介绍全局变量。

全局变量，有编程基础的朋友们相信不会陌生，是一种可以被局部获取或设置的变量。在 VisionMaster 中，全局变量可以被模块的运行参数订阅（获取全局变量的值），也可以绑定模块的结果（设置全局变量的值），在 VisionMaster 中起到多流程间数据传递的桥梁作用。另外，全局变量常用场景有：接收通讯数据，流程之间数据交换，计数等等。

我们先不做介绍的将本项目中所用到的全局变量全部列出：

全局变量

+ 添加变量　导入　导出　保存变量

序号	名称	注释	类型	当前值	输入来源	目标输出	初始化	操作
1	X1		float	650.25	------			
2	Y1		float	306.14	------			
3	Z1		float	450	------			
4	A1		float	0	------			
5	B1		float	180	------			
6	C1		float	0	------			
7	X		float	580.13	流程1.协议解析1.X			
8	Y		float	-54.27	流程1.协议解析1.Y			
9	Z		float	0	------			
10	A		float	180	流程1.协议解析1.A			
11	B		float	0	------			
12	K		float	2100	流程1.协议解析1.K			

共 27 条　12 条/页　1 2 3　跳至　页　跳转

全局变量

+ 添加变量　导入　导出　保存变量

序号	名称	注释	类型	当前值	输入来源	目标输出	初始化	操作
13	C		float	0	------			
14	v1		int	0	------			
15	p	ball	float	0	------			
16	p1		float	2	------			
17	zero		float	10	------			
18	class		int	0	------			
19	i		float	1694.449	流程1.协议解析2.i			
20	j		float	693.7267	流程1.协议解析2.j			
21	k		float	3	流程1.协议解析2.k			
22	theta		float	0	流程1.协议解析2.theta			
23	class_wo		float	11	流程1.协议解析2.class_wo			
24	write_over		float	1	流程1.协议解析2.write_ov			

共 27 条　12 条/页　1 2 3　跳至　页　跳转

全局变量

+ 添加变量　导入　导出　保存变量

序号	名称	注释	类型	当前值	输入来源	目标输出	初始化	操作
25	wood_star	开始搭积木	float	1	流程1.协议解析2.start			
26	e		float	7000	流程1.协议解析1.e			
27	q		float	0	流程1.协议解析1.q			

共 27 条　12 条/页　1 2 3　跳至　页　跳转

图 10.6: VisionMaster 项目所用到的所有全局变量

10.3.3 VisionMaster——定位

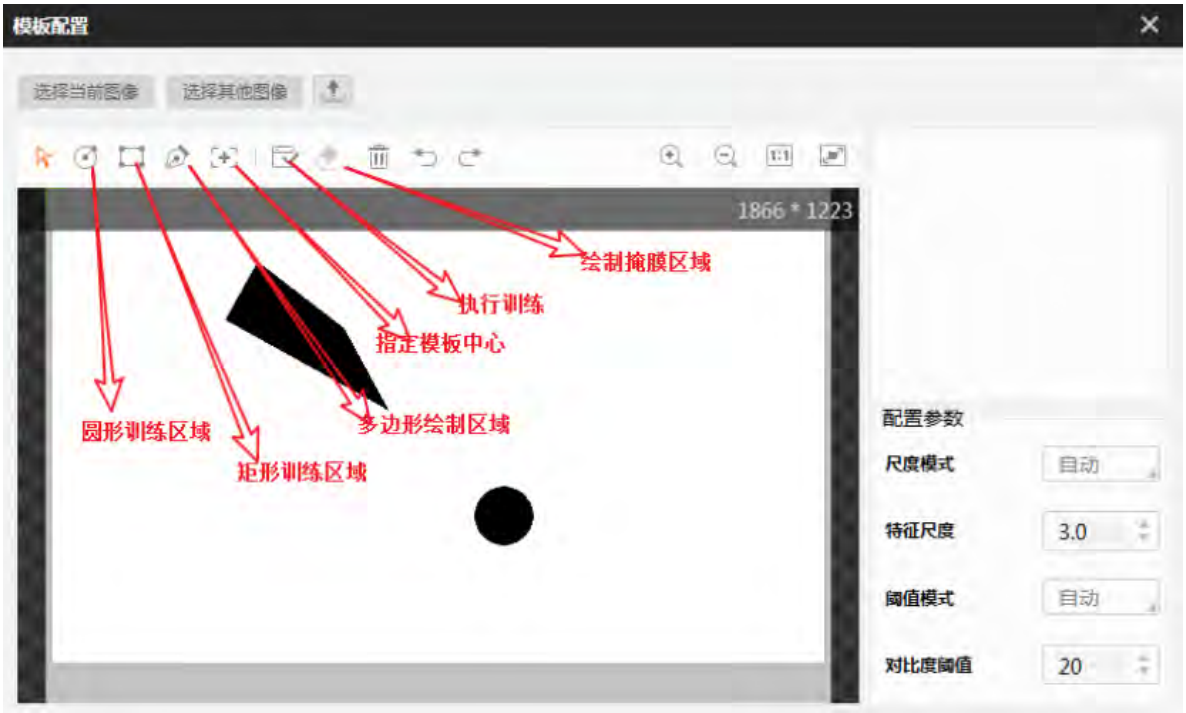
获取到图像之后, 就可以处理图像并进行定位了。

VisionMaster 中提供了基于边缘的模板匹配与基于灰度的模板匹配两种方式，其中基于边缘的模板匹配又分为高精度匹配与快速匹配。无论那种方式，都分为模板训练与模板查找两个步骤，且模板训练的过程大体相同。我们主要应用 VisionMaster 定位模块中的高精度匹配、快速匹配，灰度匹配。

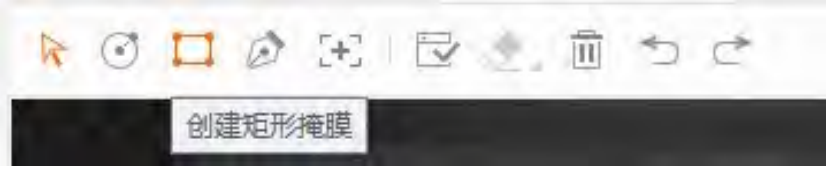
1. 模板训练

这里以快速匹配为样例进行说明。双击快速匹配模块，打开参数配置页面，选择特征模板选项卡，

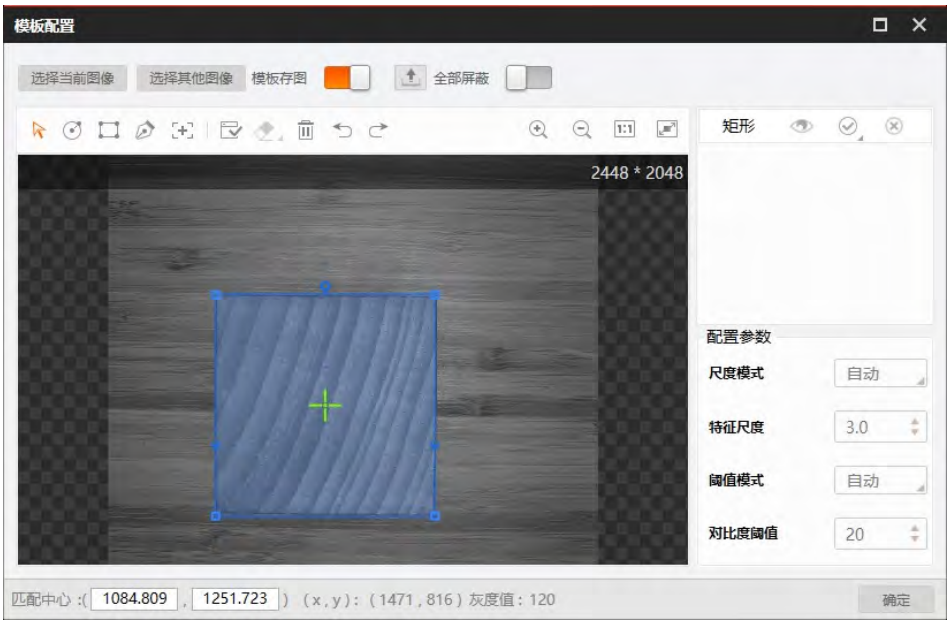
点击创建按钮，进入模板配置页面



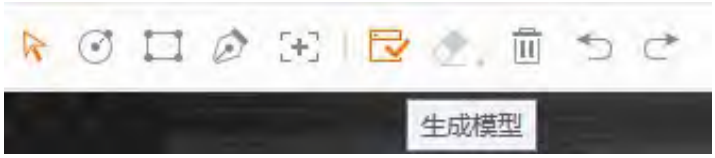
根据待训练模板的形状，选择合适训练区域类型，我们要识别的目标是积木，所以根据积木形状选择矩形区域



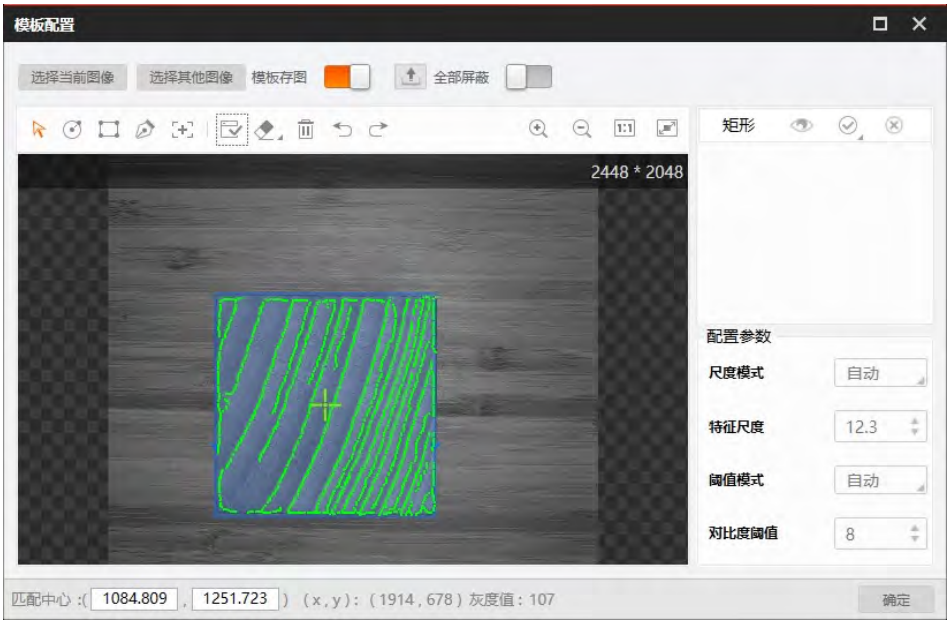
我们框出需要识别的积木表面



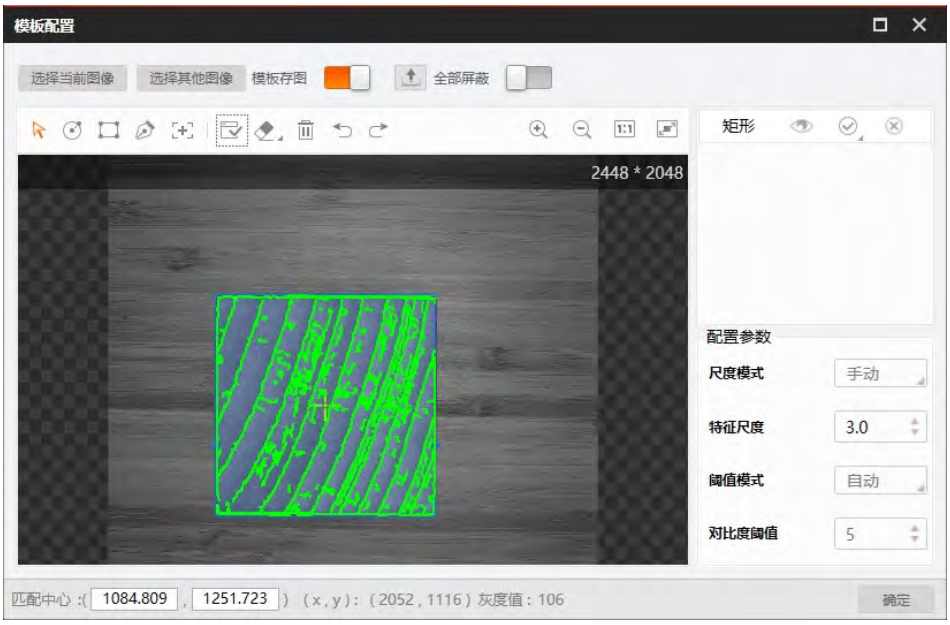
然后点击生成模型



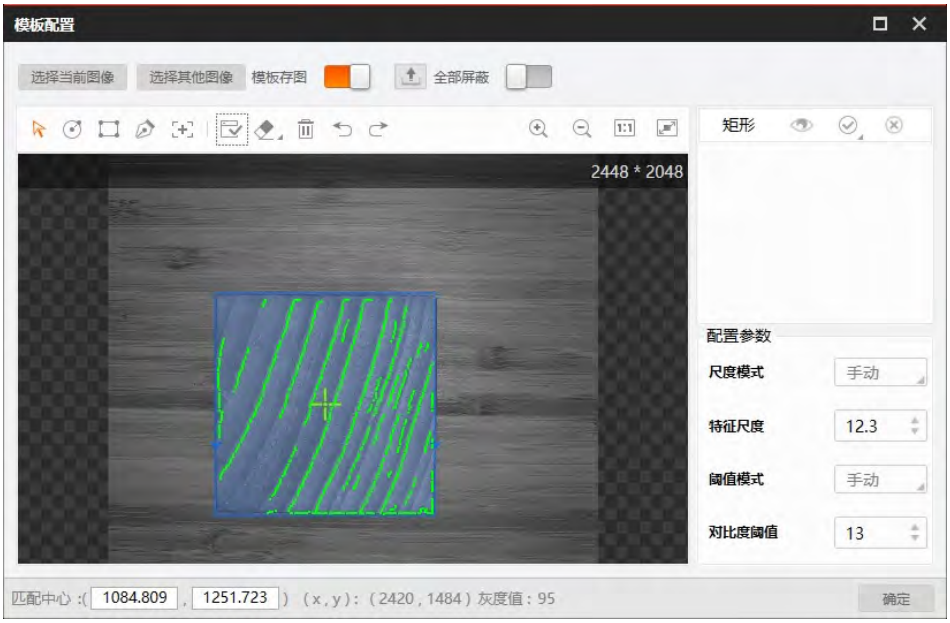
可以看到生成的绿色模版匹配点，可以通过修改特征尺度和对比度阈值来改变匹配点的数量与结构。



首先将尺度模式设置为手动，保持对比度阈值不变将特征尺度缩小为 3.0，可以观察到模版变得更加密集，匹配的点相应会增多。



同理，将阈值模式设置为手动，保持特征尺度不变将对比度阈值增大到 13.0，可以观察到模版匹配的点明显变少。



值得注意的是，对于木质轨道积木而言，由于树木纹理的差异，即使是相同形状的两个轨道面，模板匹配的结果可能存在较大的差距。在这种情况下，我们需要思考是否真的需要使用如此多的匹配点纹理来实现积木表面的识别。

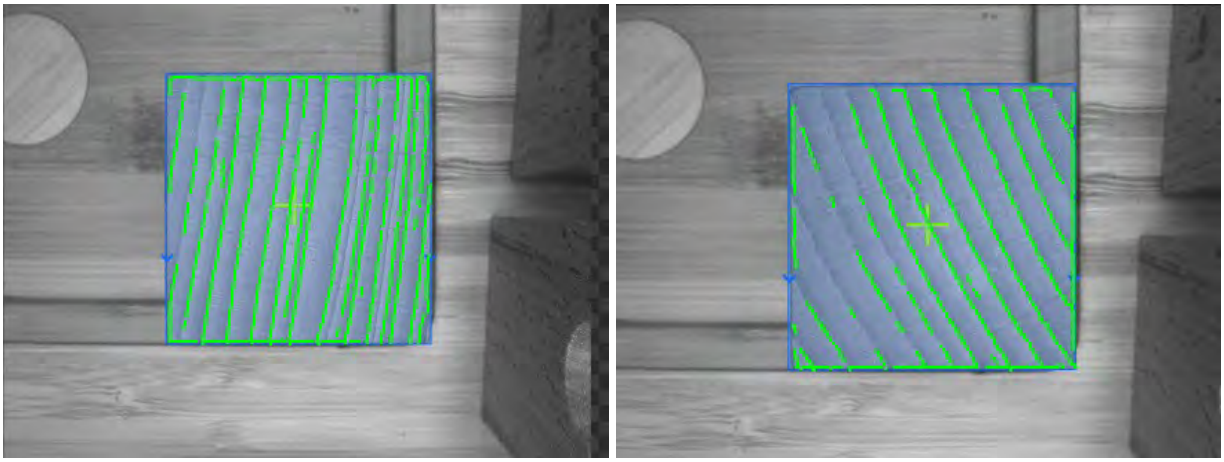
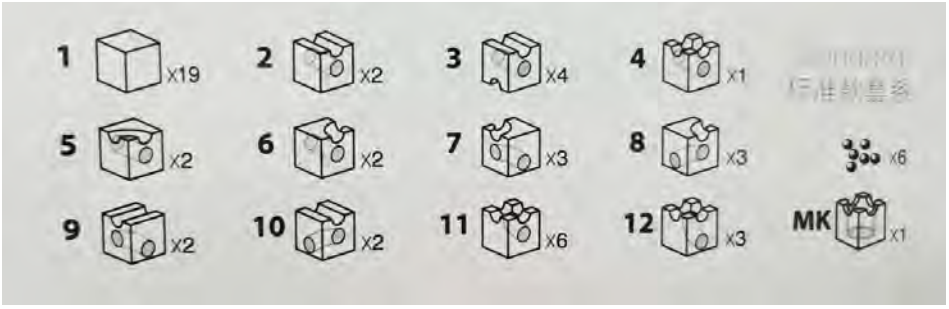
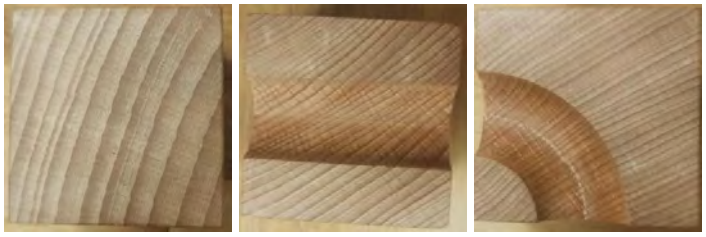


图 10.7: 两块积木拥有同样的轨道面生成的模版却相距甚远

让我们一起观察一下 Standard 标准款套系的积木的轨道表面



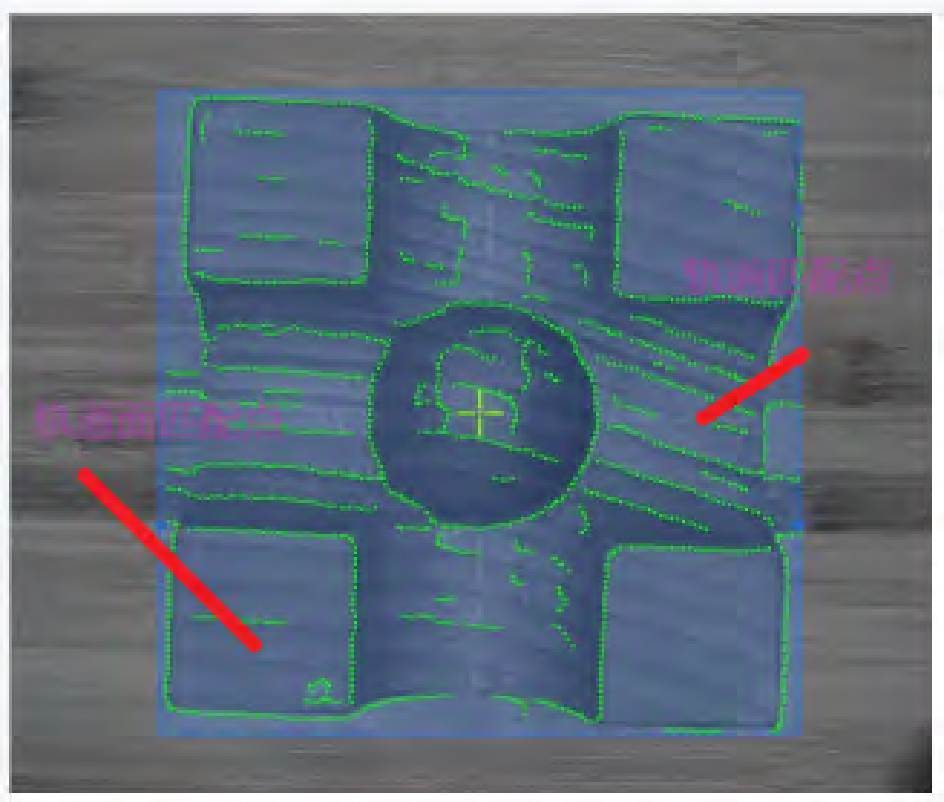
在不考虑翻转积木的情况下我们可以将轨道面分为 4 种类型:



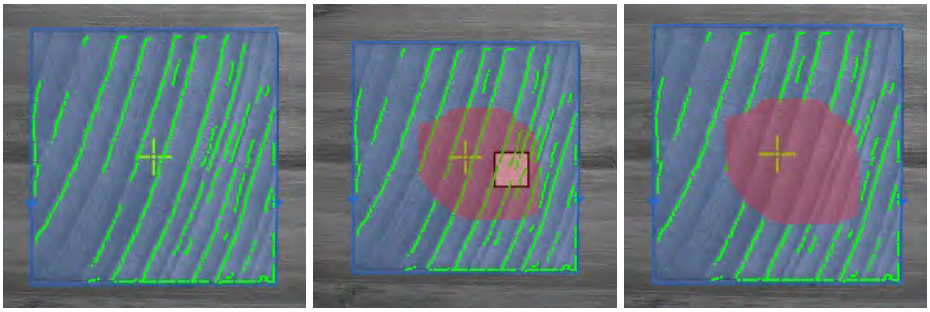
普通无轨道面　　直线轨道面　　圆弧轨道面

图 10.8: 十字轨道面

根据刚刚的设想，我们要识别的是积木的轨道面，并不需要识别积木表面的纹路，而是要将思路略过纹理放在识别积木轨道面边界形状上面。仔细观察上述 6 张图片我可以得到一个非常有用的信息，那就是轨道面上的轨道都是呈下凹形式，轨道上特征点都是要低于轨道面的。而模板上的特征点之间都存在对应的尺度关系，随着机械臂带动相机镜头的旋转移动，轨道上特征点和轨道面上的特征点之间的尺度关系会发生变化，从而造成识别失真。



下面分享一下处理方法，我们注意到模板配置页面上有绘制掩膜工具，我们先看看效果:

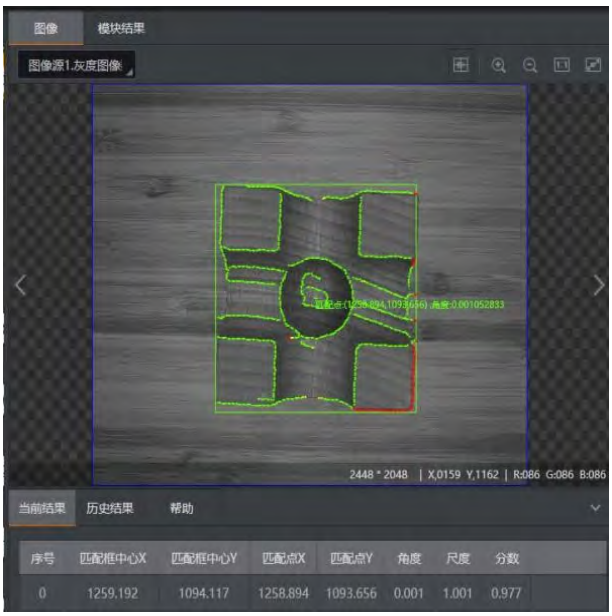


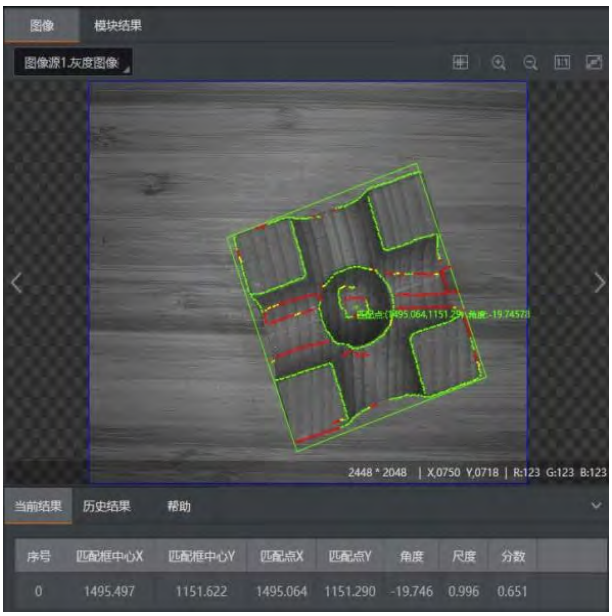
1　　2　　3

1. 矩形工具绘制生成模板范围 -> 2. 绘制掩膜工具绘制区域 ->3. 再次生成模型

我们可以看到在掩膜绘制的区域并没有生成模板。现在问题就迎刃而解了，下面我们以十字轨道面为例进行讲解。

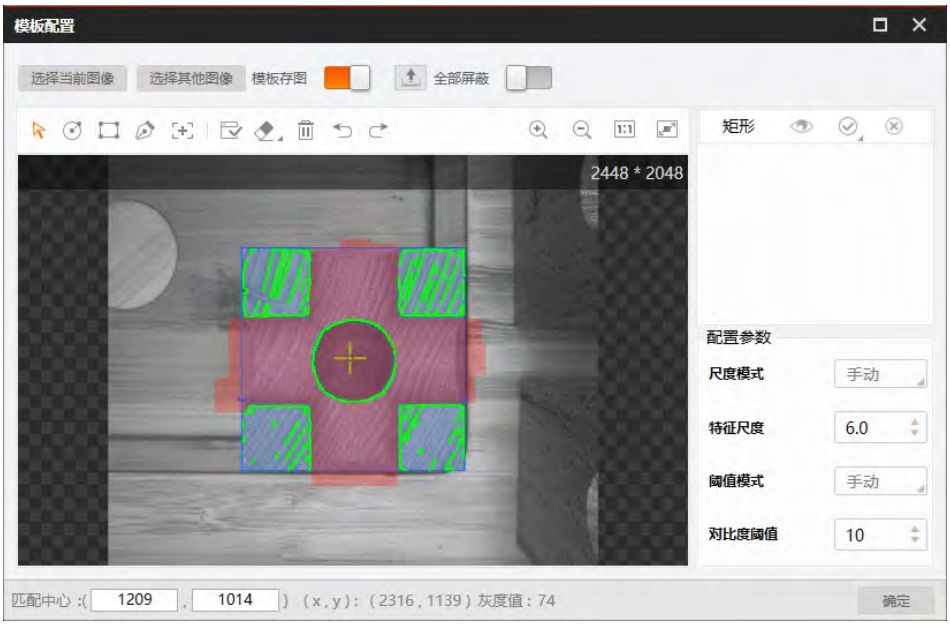
我们先来观察完整模板的匹配效果：



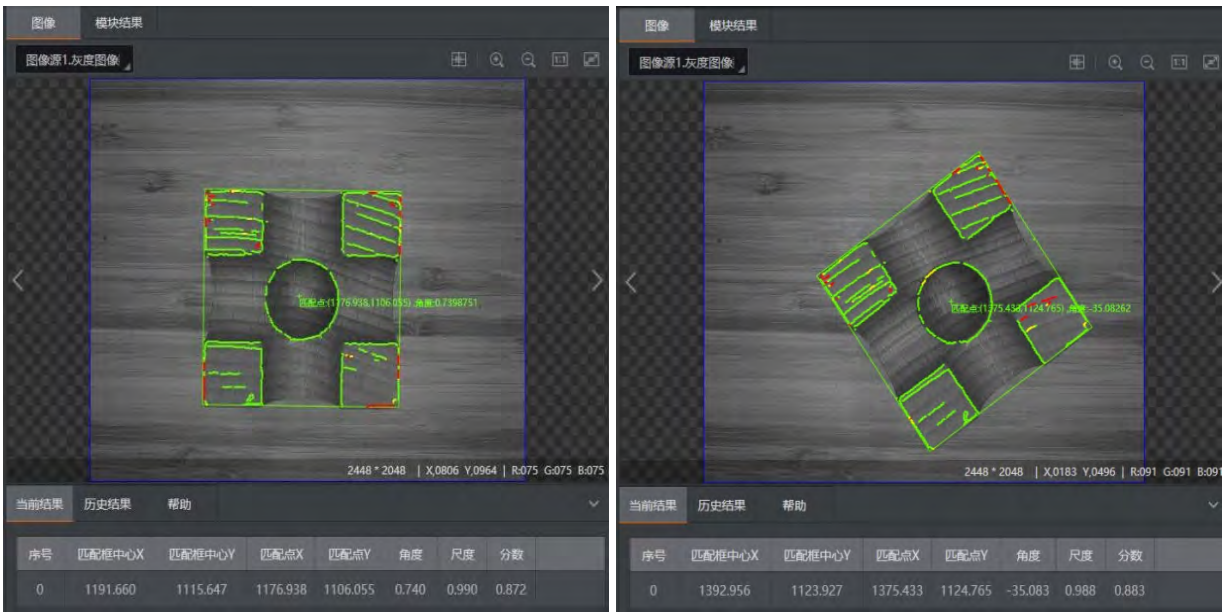


可以看到只要镜头稍微旋转一下匹配分数就从 0.977 下降到 0.651，当继续旋转分数小于阈值时模板匹配就失效了。

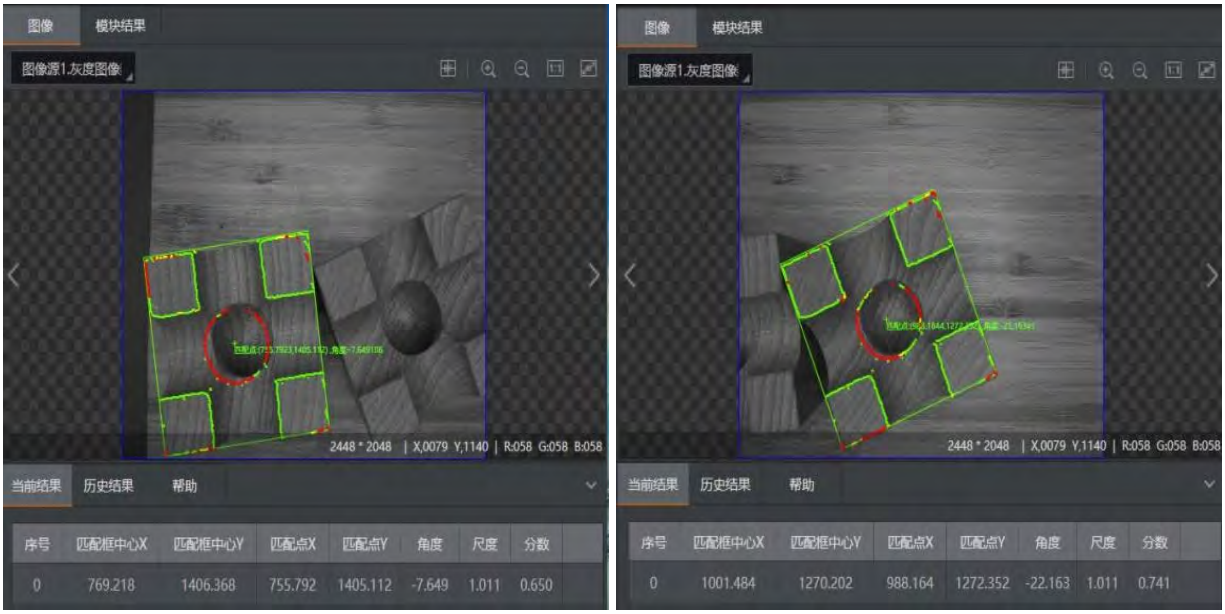
接下来，我们将生成模板轨道上的点全部抹去，如下所示：



我们来看看检测的匹配效果:



我们稍微施加一点旋转匹配分数从 0.872 变到了 0.883，匹配分数维持稳定说明该方法能很好地匹配到旋转的轨道积木表面。但此方法的通用性依旧存在问题，由于模板上的纹理依旧存在，对同一型号的另一块积木，模板匹配就难以进行了。为了解决模板的通用性问题，我们可以在此基础上继续做减法。将轨道积木表面的模板纹理也去除，观察效果：



可以看到，在抓住重点特征之后生成的这个模板，即使是不同的木块，只要轨道面的类型相同，也能够匹配到。在此我们找到了一种普适的方法制作模板来识别一个

类型的轨道面，在此基础上抓起积木已经不是难事了。但是离搭建具有连通性的轨道积木仍有一定差距。

10.4 轨道积木搭建难点

接下来分析一下轨道积木搭建的难点：

1. 惟木匠心的轨道积木精度很高，但是在使用过程中由于热胀冷缩、磨损、受潮、应力释放，不同年份批次制作的木头密度不同等等原因导致积木体积大小存在之间的误差。这些差异通过肉眼很难观察到。

图 10.9: 积木之间存在误差

2. 由于积木大小存在误差，机械臂在抓取旋转的过程中会导致积木与积木之间存在缝隙，这些误差累积在一起轨道的连通性就无法保证了，肉眼观察会觉得很不舒服。这说明积木搭建需要较高的识别精度才能实现。

图 10.10: 积木搭建过程中存在缝隙

3.KUKA 机械臂的重复定位精度很高，但是绝对位置存在一定误差，所以会出现第一次抓取积木搭建存在较大缝隙，但是机械臂将积木放回去再次重复搭建的积木位置会变得更加准确。

4. 放置积木的木板可能倾斜，导致识别出现误差。

5. 相机镜头原因产生的图片畸变，导致匹配中心点出现误差。

6. 为了观看体验更加友好，我们需要尽可能剪短识别时间，加速机械臂运动同时还要保持较高的精度，这也对算法提出很高的要求。设想一下如果每个木块从识别、抓取到放到目标搭建位置如果需要耗时 30s 的话，对于一个需要 20 块积木搭建的模型则需要 10 分钟的时间，这显然达不到能够展示的要求，也不够优雅。

7. 机械臂在快速运行的过程中由于惯性，平台会产生抖动，对于识别定位都存在一定的误差影响。

8. 弹珠在下落过程中由于惯性也会冲散积木，而轨道过长会导致弹珠卡在轨道中间，因此设计合理的弹珠运行轨道也是必要的。

9. 在机械臂抓取积木并摆放的过程中如何规划运行路径，躲避已经搭建好的积木也是要在考虑之中的。

10. 如何设计有趣且好玩积木搭建方案也是重中之重。

11. 机械臂在运行过程中的安全控制应该至于首位。

12. 外界环境的影响例如光照的变化，在阳光直射下，明暗交界处会对图像的识别造成影响；在温度较低的室外，机械臂关节处的油会更加粘稠，导致机械臂运行出现问题。

13. 我们希望机械臂展示的速度尽可能快，因此在机械臂与视觉控制器之间的信息交互要保持高速、流畅。

14. 机械臂要配合各种传感器才能完成整个环节的演示，各种外部电路的累积致使线路变得非常复杂，所以如何隐藏这些线路也是一个难题。

为了解决上述难题，我们首先从整套系统的通信开始讲起。

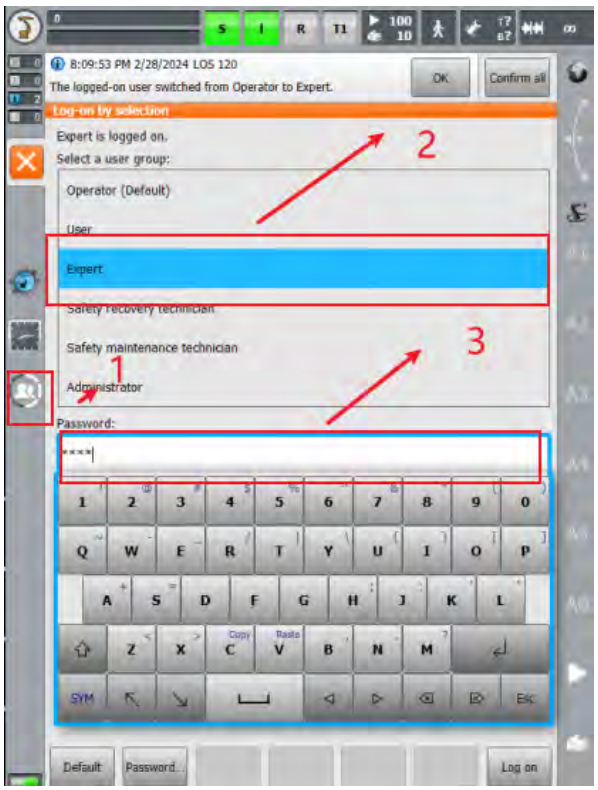
10.5 视觉机械臂实战——通信

10.5.1 机械臂信息传输到视觉控制器

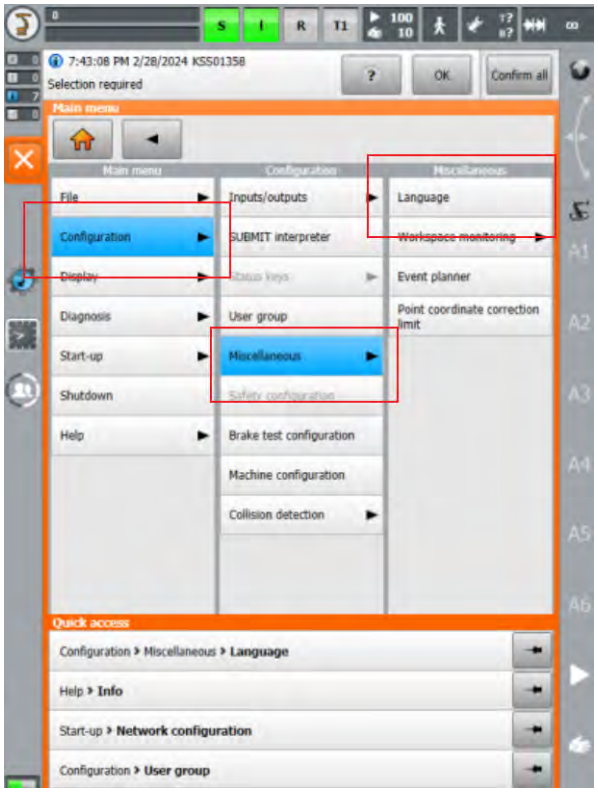
初始的示教器页面是英文显示的，为了叙述方便现将其转为中文。

首先要将用户组切换为专家及以上的等级。

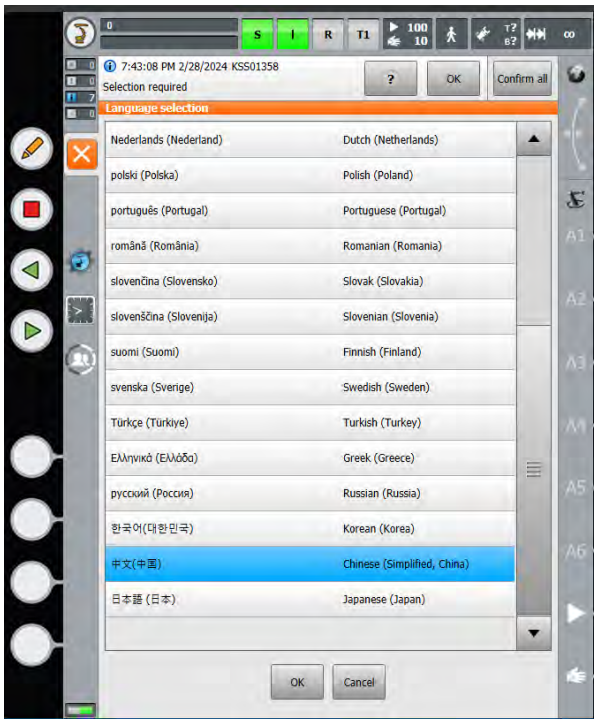
权限密码为 kuka



点击示教器主界面图标，选择 Configuration->Miscellaneous->Language

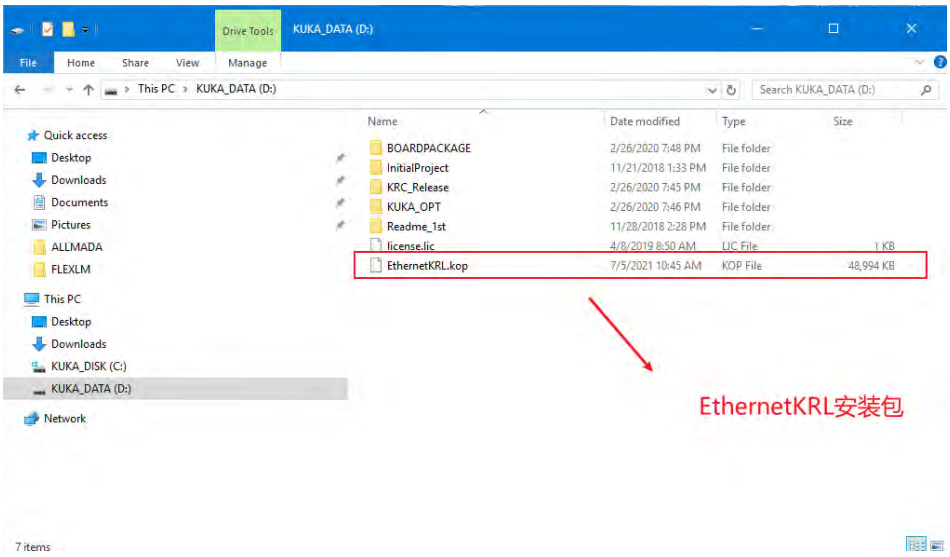


下拉找到中文，然后点击 OK，语言界面就切换完成了。



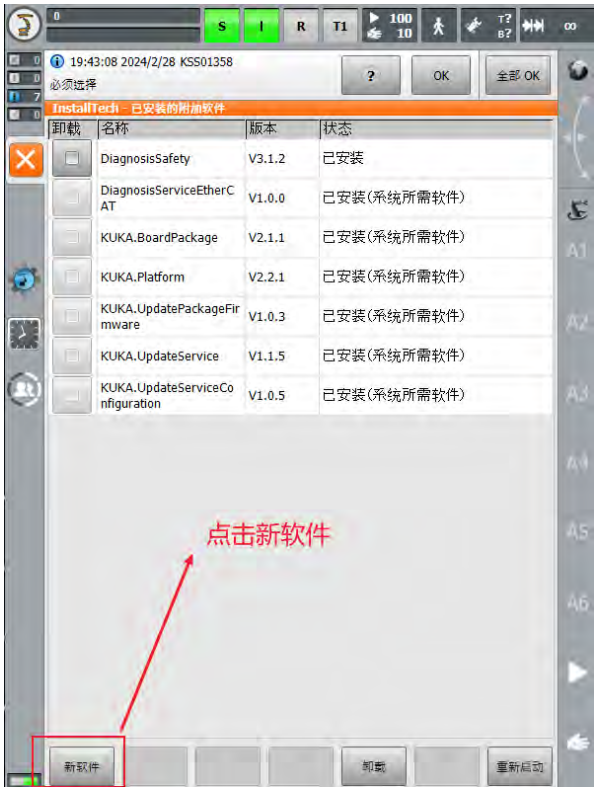
我们通过 EthernetKRL 工具包实现 kuka 机械臂与外界设备的网络通信，备选软件包 KUKA.Ethernet KRL 是为在提交解释器或机器人解释器程序运行时通过以太网在机器人控制系统和外部系统之间进行数据交换而设计的。

下面进行 EthernetKRL 软件包的安装，将 EthernetKRL.KOP 安装包复制到 D 盘文件下，

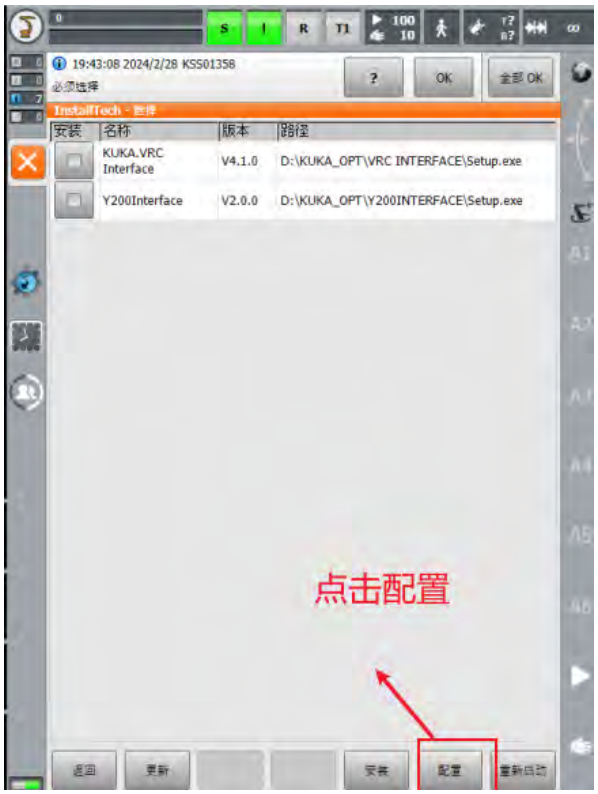


再次在示教器主界面上选择投入运行-> 辅助软件

然后点击新软件选项

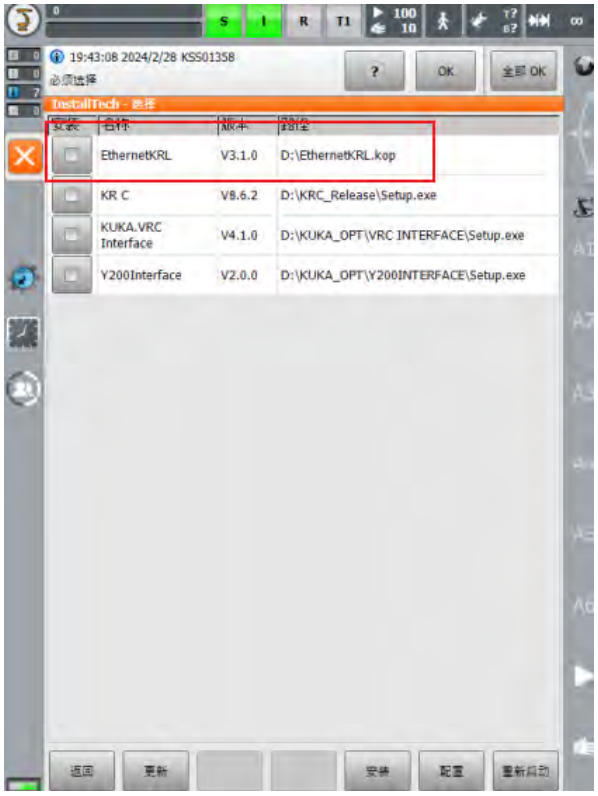


点击配置选项

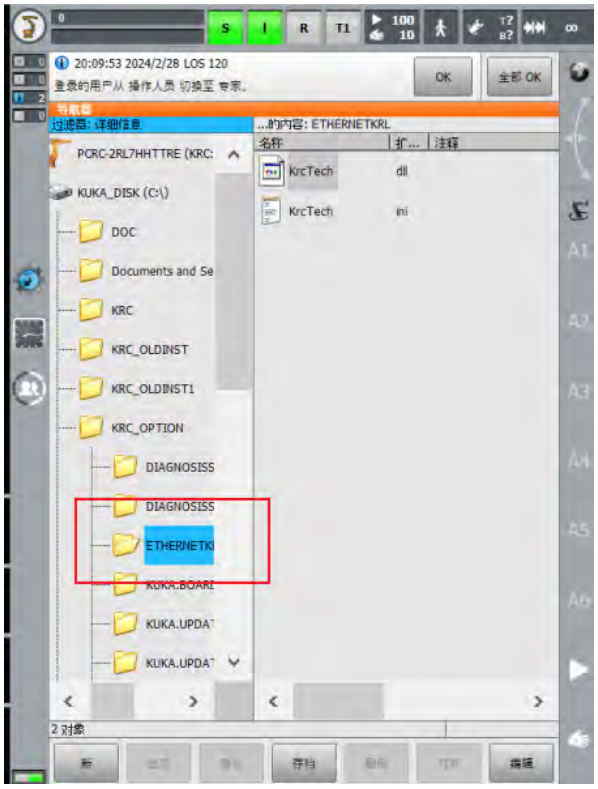


点击路径选择，选择我们放置 EthernetKRL.KOP 安装包路径，D:/之后点击保存

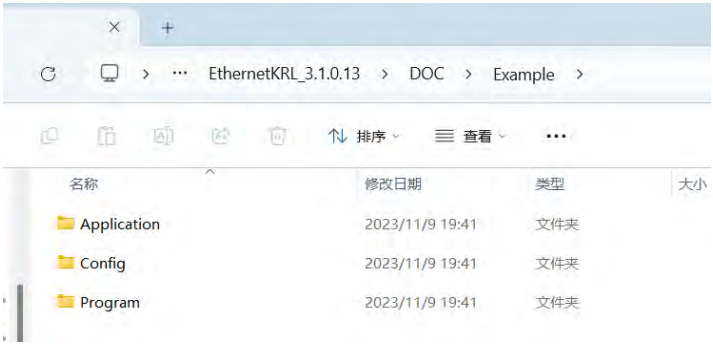
此时在安装界面可以选择安装 EthernetKRL



安装完成之后可以在 C:/KRC_OPTION/路径下找到 ETHERNETKRL 文件夹



可以在 EthernetKRL 软件包的 DOC->Example 路径下找到找到示例程序



Application 中保存的是 EthernetKRL 的应用程序。

Config 文件中保存的是 xml 文件，机械臂以太网连接通过 xml 文件进行配置。针对每个连接，必须在机器人控制系统的目录 C:/KRC/ROBOTER/Config/User/Common/EthernetKRL 中定义一个配置文件。

Program 文件中保存的是机械臂编程信息交互的示例代码。

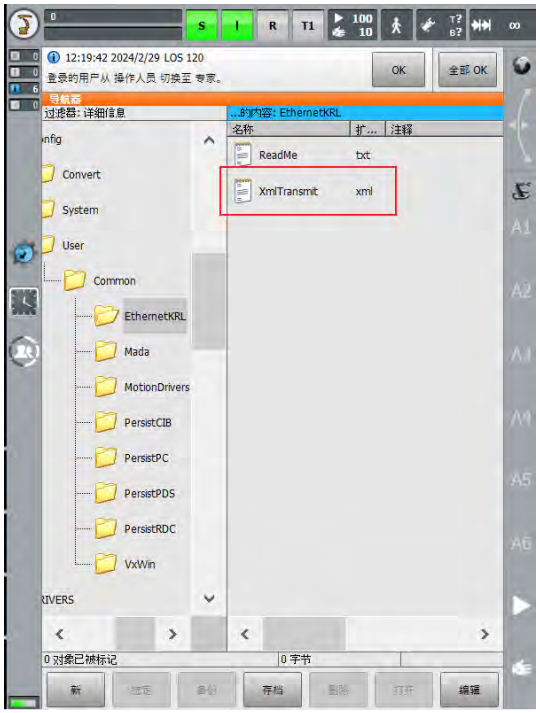
下面介绍一下 xml 文件的代码含义：

```
<ETHERNETKRL>
  <CONFIGURATION>
    <EXTERNAL></EXTERNAL>
    <INTERNAL></INTERNAL>
  </CONFIGURATION>
  <RECEIVE>
    <ELEMENTS></ELEMENTS>
  </RECEIVE>
  <SEND>
    <ELEMENTS></ELEMENTS>
  </SEND>
</ETHERNETKRL>
```

EthernetKRL 的 xml 配置文件都是以 <ETHERNETKRL> 为开头，</ETHERNETKRL> 结尾

段	说明
<CONFIGURATION> … </CONFIGURATION>	配置外部系统和 EKI 之间的连接参数
<RECEIVE> … </RECEIVE>	配置机器人控制系统接收的接收结构
<SEND> … </SEND>	配置机器人控制系统发送的发送结构

我们在 C:/KRC/ROBOTER/Config/User/Common/EthernetKRL 中找到 XmlTransmit.xml 文件



接下来结合本项目的 XmlTransmit.xml 文件进行讲解：

```
<ETHERNETKRL>
<CONFIGURATION>
<EXTERNAL>
<IP>192.168.1.115</IP>
;与机械臂通信设备的IP地址
<PORT>59152</PORT>
;IP地址端口号
</EXTERNAL>
</CONFIGURATION>
<RECEIVE>
;设置接收消息格式
<XML>
<ELEMENT Tag="Sensor/X1" Type="REAL" Set_Flag="1"/>
; 设置接收消息的元素名称，类型（REAL，INT，BOOL，STRING等）
<ELEMENT Tag="Sensor/Y1" Type="REAL" />
<ELEMENT Tag="Sensor/Z1" Type="REAL" />
<ELEMENT Tag="Sensor/A1" Type="REAL" />
<ELEMENT Tag="Sensor/B1" Type="REAL" />
<ELEMENT Tag="Sensor/C1" Type="REAL" />
<ELEMENT Tag="Sensor/p" Type="REAL" />
</XML>
</RECEIVE>
<SEND>
;设置发送消息格式
<XML>
<ELEMENT Tag="Robot/Data/mypos/@x"/>
;发送消息不用指定变量类型
<ELEMENT Tag="Robot/Data/mypos/@y"/>
<ELEMENT Tag="Robot/Data/mypos/@z"/>
<ELEMENT Tag="Robot/Data/mypos/@a"/>
<ELEMENT Tag="Robot/Data/mypos/@b"/>
<ELEMENT Tag="Robot/Data/mypos/@c"/>
<ELEMENT Tag="Robot/Data/mypos/@f"/>
<ELEMENT Tag="Robot/Data/mypos/@p"/>
<ELEMENT Tag="Robot/Data/mypos/@e"/>
<ELEMENT Tag="Robot/Data/mypos/@q"/>
<ELEMENT Tag="Robot/Data/mypos/@n"/>
```

```
;<ELEMENT Tag="Robot/Data/mypos/@class"/>
<ELEMENT Tag="Robot/Status"/>
<ELEMENT Tag="Robot/Mode"/>
<ELEMENT Tag="Robot/RobotLamp/GrenLamp/LightOn"/>
</XML>
</SEND>
</ETHERNETKRL>
```

在上面的语句 <ELEMENT Tag="Sensor/X1" Type="REAL" Set_Flag="1"/> 中有 Set_Flag="1" 这一参数，我们先将图表展示在下，对意义先不做过多讲解。

ELEMENT	Set_Out	接收到元素时给输出端或旗标赋值（可选）
	Set_Flag	输出端编号： • 1 ... 4096 旗标编号： • 1 ... 1024 提示：如果通过 Set_Out/Set_Flag 设置一个输出端或插上一个旗标，那么程序代码中的相应系统变量 $OUT[]/$FLAG[] 必须被重新复位。

在上面的 xml 文档中我们指定了外部通讯设备的 IP 地址，为了确保通信正常，我们需要将机械臂的 IP 地址设为同一网段。

首先要在示教器中将用户组切换到专家模式。

然后一次点击主界面-> 投入运行-> 网络配置，将 IP 地址设置为 192.168.1.110

然后单击保存，重启设备更新 IP 地址。

我们想要实时获取机械臂的运动位姿，因此需要修改机器人提交解释器的 SUB 文件。

机器人控制系统上平行执行着 2 个任务：

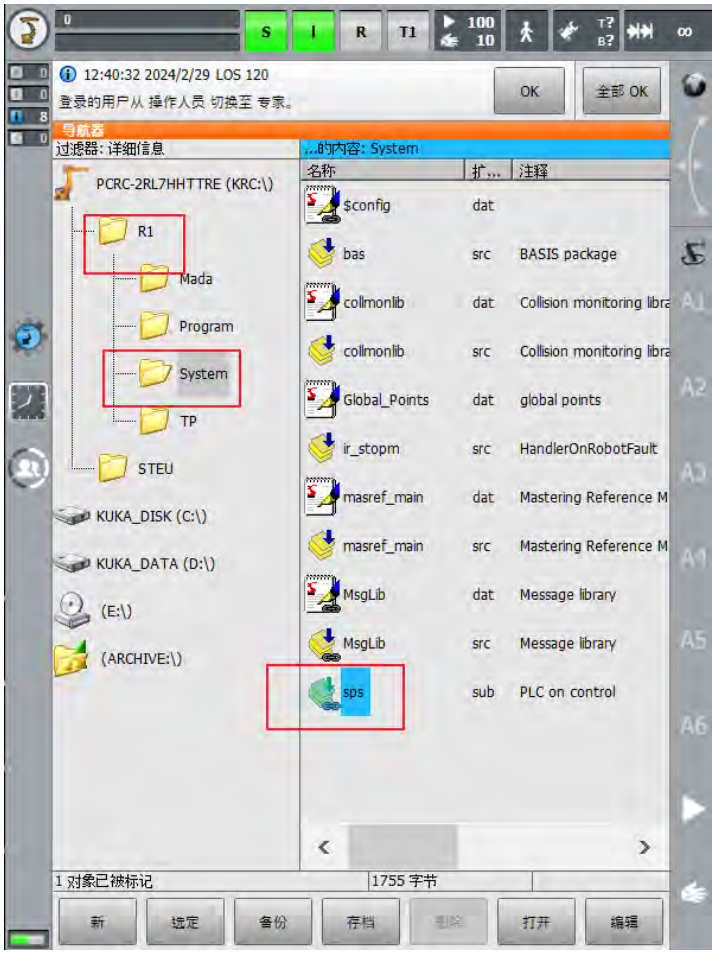
- 机器人解释器
  运动程序通过机器人解释器执行。
- 提交解释器
  提交程序（SUB-Programm）通过提交解释器执行。
  SUB 程序可以执行操作或者监控任务。示例：监控防护装置；监控冷却循环回路。
  如果是简单应用则无需使用 PLC，机器人控制系统即可执行此类工作。

提交解释器在机器人控制系统接通时自动启动。此时启动程序 SPS.SUB。

提交解释器可以手动停止或取消选择，也可以重新启动。

提交解释器在机器人系统接通之后就会自动启动，因此机械臂坐标的位置可以实时发送出去。

在 R1/System 文件下找到 sps.sub 文件：



找到注释行;FOLD USER DECL,在此处声明 EthernetKRL 变量 RET:DECL EKI_STATUS RET

```
1   &ACCESS RVO
2   &REL 37
3   &COMMENT PLC on control
4   DEF  SPS ( )
5     ;FOLD DECLARATIONS
6   BASISTECH DECL
11    ;FOLD USER DECL
12    ; Please insert user defined declarations
13
14     DECL EKI_STATUS RET
15
16    ;ENDFOLD (USER DECL)
17    ;ENDFOLD (DECLARATIONS)
```

找到代码中的 LOOP 与 ENDLOOP 代码行，可以在

;FOLD USER PLC

;Make your modifications here

注释行下添加发送坐标的代码，LOOP 与 ENDLOOP 是循环执行的，因此坐标可以循环发送

```
43    LOOP
44      WAIT FOR NOT($POWER_FAIL)
45  BASISTECH PLC
48  BACKUPMANAGER PLC
53      ;FOLD USER PLC
54      ;Make your modifications here
55  number1=1
56  IF (number1==1) THEN
57   REPEAT
58   UNTIL number1<>1
59  ENDIF
60  LOOP
61  number=0
62  IF (number==1) THEN
63   REPEAT
64   UNTIL number<>1
65  ENDIF
66  ;FOLD Write data to connection
67    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@x",$POS_ACT.X)
68    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@y",$POS_ACT.Y)
69    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@z",$POS_ACT.Z)
70    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@a",$POS_ACT.A)
71    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@b",$POS_ACT.B)
72    RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@c",$POS_ACT.C)
73  ;ENDFOLD (Write data to connection)
74  ;FOLD Send data to external program
75    RET = EKI_Send("XmlTransmit","Robot")
76    WAIT SEC 0.05
77
78  IF (number1==1) THEN
79   REPEAT
80   UNTIL number1<>1
81  ENDIF
82  ;ENDFOLD (Send data to external program)
83  ENDLOOP
84      ;ENDFOLD (USER PLC)
85    ENDLOOP
```

编写 55-65,78-81 语句行 IF 逻辑判断的目的是想在主程序执行后可以通过变量 number，number1 来控制坐标系的发送。

```
55  number1=1
56  IF (number1==1) THEN
57   REPEAT
58   UNTIL number1<>1
59  ENDIF
60  LOOP
61  number=0
62  IF (number==1) THEN
63   REPEAT
64   UNTIL number<>1
65  ENDIF
```

```
78  IF (number1==1) THEN
79   REPEAT
80   UNTIL number1<>1
81  ENDIF
```

代码的第 66-76 行是实时发送的机械臂坐标

```
66  ;FOLD Write data to connection
67     RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@x",$POS_ACT.X)
68     RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@y",$POS_ACT.Y)
69     RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@z",$POS_ACT.Z)
70     RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@a",$POS_ACT.A)
71     RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@b",$POS_ACT.B)
72     RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@c",$POS_ACT.C)
73  ;ENDFOLD (Write data to connection)
74  ;FOLD Send data to external program
75     RET = EKI_Send("XmlTransmit","Robot")
76     WAIT SEC 0.05
```

$POS_ACT.X，$POS_ACT.Y，$POS_ACT.Z，$POS_ACT.A，$POS_ACT.B，$POS_

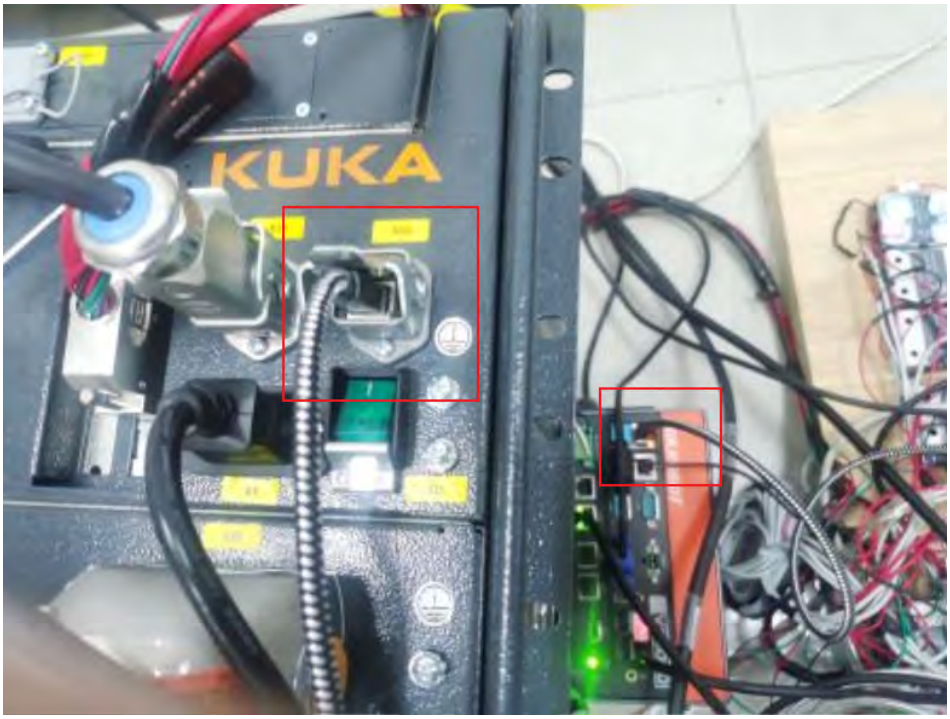
是机械臂位姿的系统变量，可以全局调用显示。这里每隔 0.05s 发送一次消息。之后只要在机械臂主程序 LI_1018sec4.src 代码中加入以下代码：

```
DECL EKI_STATUS  RET
;定义EthernetKRL状态变量
RET=EKI_Init("XmlTransmit")
RET=EKI_Open("XmlTransmit")
;初始化并打开信号传输的XmlTransmit.xml配置文件
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@p",0)
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@e",3500)
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@q",0)
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",0)
```

到这里机械臂端向外界设备发送消息的基础配置就已经完成了。

接下来我们讲解视觉控制器端的通讯配置。

将 kuka 控制柜上的 X66 接口通过网线连接到 MV-VC3301P-128G60 视觉控制器上的网口，组成局域网。



之后将 MV-VC3301P-128G60 视觉控制器的 IP 地址修改为 XmlTransmit.xml 中配置的 IP 地址 192.168.1.115，子网掩码设置为 255.255.255.0。

```
以太网适配器 以太网_5:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :

以太网适配器 以太网_6:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::f99e:26fb:14a3:bebc%15
   IPv4 地址 . . . . . . . . . . . . : 192.168.1.115
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . :

以太网适配器 以太网_3:

   连接特定的 DNS 后缀 . . . . . . . :
   本地链接 IPv6 地址. . . . . . . . : fe80::ddb1:8f56:e131:9190%6
   IPv4 地址 . . . . . . . . . . . . : 192.168.1.100
   子网掩码  . . . . . . . . . . . . : 255.255.255.0
   默认网关. . . . . . . . . . . . . : 192.168.1.1

隧道适配器 isatap.{9BB80175-947C-4AEB-9B13-8590B46E882D}:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :

隧道适配器 Teredo Tunneling Pseudo-Interface:

   媒体状态  . . . . . . . . . . . . : 媒体已断开连接
   连接特定的 DNS 后缀 . . . . . . . :
```

接着打开 MV-VC3301P-128G60 视觉控制器中下载的 Visionmaster 软件，点击软件菜单栏上的通信管理

选择设备管理，单击设备列表的加号，选择类型 TCP 服务器，将本机 IP，端口与 XmlTransmit.xml 中的配置保持一致，然后打开数据上传与数据自动重连。

创建完成之后 1TCP 服务端自动重连显示打开状态，则机械臂与视觉控制器的连接是正常的。

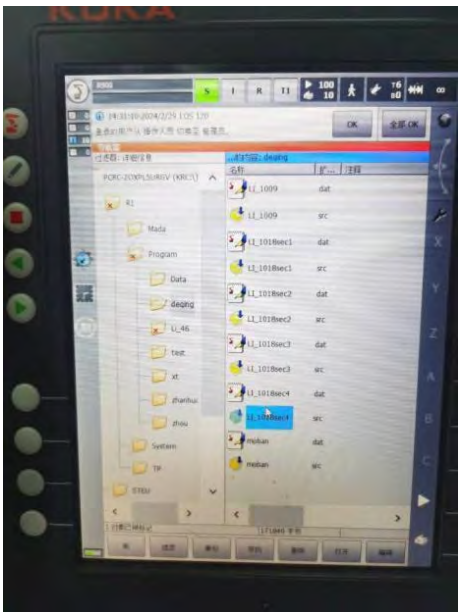
然后在接收事件创建事件管理，处理方式选择文本，事件类型选择协议解析

接着选择创建好的 1 文本-协议解析，在绑定设备处选择上文创建好的 1TCP 服务端，分隔符选择“;”，在输出列表中新加一行，名称命名为 out1，类型选择 string

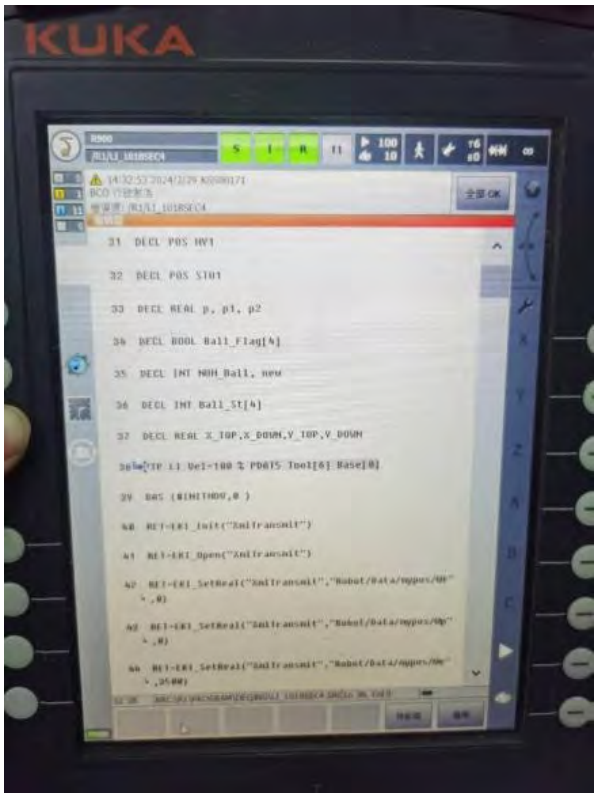
之后在左侧模块栏找到通信模块

我们用到的是接收数据模块和发送数据模块，将两个模块拖入到流程 1 中，点击接收数据，选择数据源为通信设备，在通信设备选择 1TCP 服务端，输入数据命名为 out0

之后在机械臂示教器上选定 S 程序，选定成功 S 图标会变绿

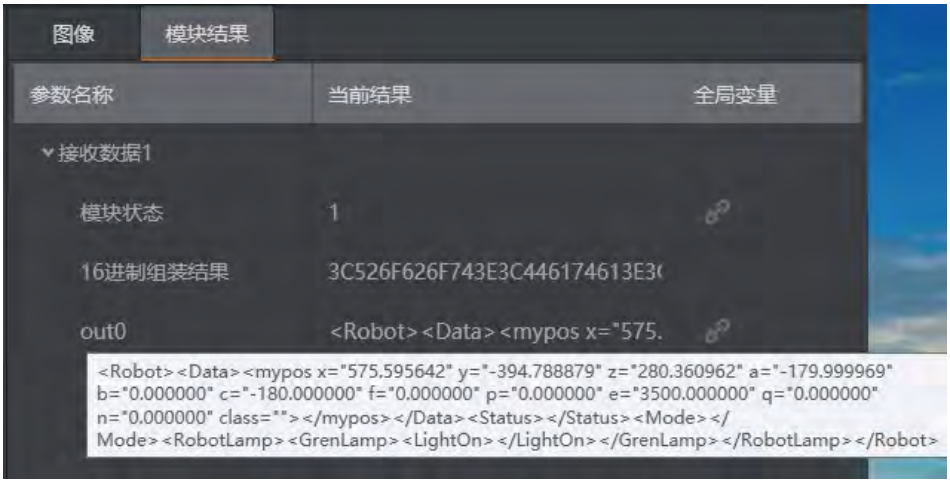


然后选定主程序 LI_1018sec4.src，按住使能键执行运行：



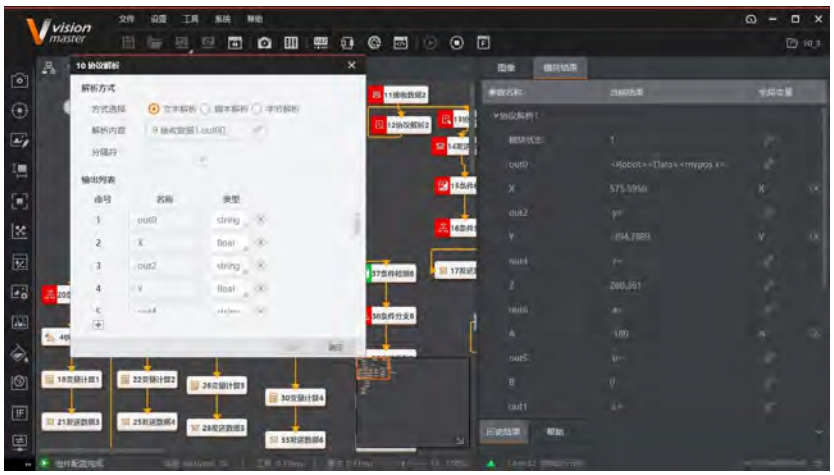
此时可以在 VisionMaster 中的通信管理的接收数据里看到机械臂发过来的位姿坐标信息。(观察时间信息戳，两个相邻的时间戳 13:42:04:553,13:42:04:637，其时间间隔计算可得 0.084s，相比与上文中程序设置的 0.05s 时间有所增加，因此在接收消息与发送消息之间存在多余的延迟时间)

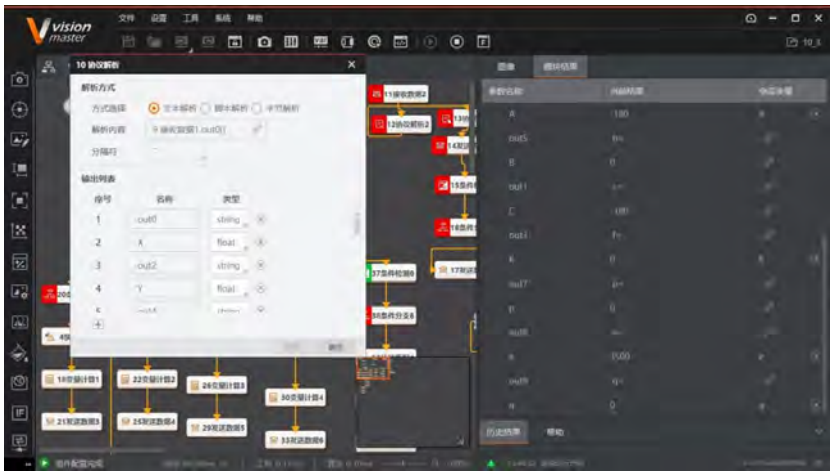
接着点击 9 接收数据，观察其模块结果，我们可以看到 out0 变量接收到的信息字符串，可以根据其结构在协议解析里解析坐标变量。



点击 10 协议解析配置参数，在方式选择栏内选择文本解析，解析内容选择模块 9 接收数据所获得的字符串变量 out0。（字符串组成结构为字符 + 数字 + 字符……这种循环的组合方式）

所以在输出列表内以 string+float+string+……的形式提取所需的数据。之后仔细观察 10 协议解析所得到的模块结果：





可以看到我们定义的 float 变量都成功提取到了机械臂发来的数据，可以点击右边

的链接来将值传给上文所设置的全局变量，到此机械臂的数据已经被我们完整的获取到了。

图 10.11: 可以将数据传给上文设置的全局变量

10.5.2 视觉控制器信息传输到机械臂

视觉控制器是控制相机的系统，需要将相机获取到的积木信息处理之后发送给机械臂实现闭环的信号反馈。

首先介绍机械臂编程中如何接收数据，在 LI_1018sec4.src 代码中找到第 466-490 行

```
466          RET=EKI_ClearBuffer("XmlTransmit","Sensor")
467          $FLAG[1] = FALSE
468          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
469          WHILE iu < 2
470
471          IF num >= 2 THEN
472             EXIT
473          ENDIF
474
475          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",1111)
476          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",98765111)
477          WAIT FOR  ( $FLAG[1] )
478          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
479          RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",0)
480          number=1
481          RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
482          RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
483          RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
484          RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
485          RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
486          RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
487          RET=EKI_GetReal("XmlTransmit","Sensor/p",p)
488
489          WAIT SEC wait_sec
490          number=2
```

在接收消息前需要清除寄存器上的缓存队列 RET=EKI_ClearBuffer(”XmlTransmit”,”Sensor”)

在上文中我们并没有解释下图中 Set_Flag 的意义

```
<ELEMENT  Tag="Sensor/X1"  Type="REAL"  Set_Flag="1"/>
; 设置接收消息的元素名称，类型（REAL，INT，BOOL，STRING 等）
```

Set_Flag=”1”表示的是系统变量 $FLAG[1] 的值，该值默认为 FALSE，在 WAIT FOR（$FLAG[1]）语句中会等待信号变成挂起状态，直到机械臂接收到 Sensor/X1 信号，$FLAG[1] 的值会变成 TRUE，程序才会继续运行。

函数 EKI_SetReal 用来设置机械臂向视觉控制器发送数据的变量值：

```
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@f”,0)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@f”,1111)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@n”,98765111)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@f”,0)
RET=EKI_SetReal(”XmlTransmit”,”Robot/Data/mypos/@n”,0)
```

函数 EKI_GetReal 用来接收视觉控制器向机械臂发送数据的变量值，并将值赋给参数的第二个变量 DA1，DA2，DA3，DA4，DA5，DA6，p（在变量初始化的时候类型要保持一致为 REAL。注意：EKI_GetReal 不能放置于 WAIT FOR($FLAG[1]) 语句之前，视觉控制器未发送信息就执行接收命令程序会报错，并且 $FLAG[1] 值变为 true 后要手动设为 false 才能确保程序的循环运行）：

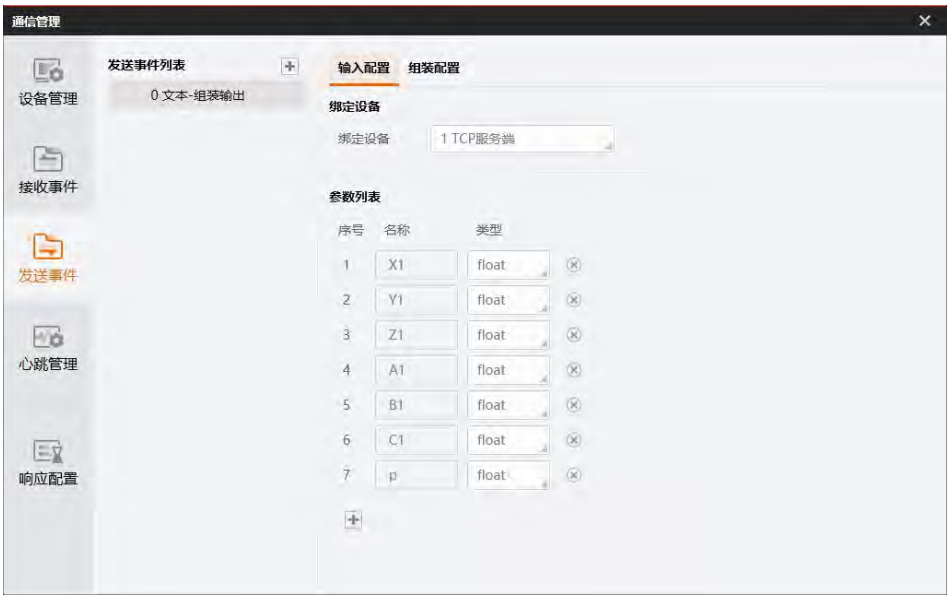
```
RET=EKI_GetReal(”XmlTransmit”,”Sensor/X1”,DA1)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/Y1”,DA2)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/Z1”,DA3)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/A1”,DA4)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/B1”,DA5)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/C1”,DA6)
RET=EKI_GetReal(”XmlTransmit”,”Sensor/p”,p)
```

number 变量用来控制信息的发送。（number 为全局变量）number=2

接着在通信模块中将发送数据模块拖入到流程 1 界面中

现在可以配置 VisionMaster 中的发送事件，点击发送事件列表添加事件管理，选择处理方式为文本，事件类型为组装输出。

创建完成之后需要在输入配置中设置绑定设备为上文创建的 1TCP 服务端，



参数列表设成 EKI_GetReal 中需要接收的 X1,Y1,Z1,A1,B1,C1,p

```
RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
RET=EKI_GetReal("XmlTransmit","Sensor/p",p)
```

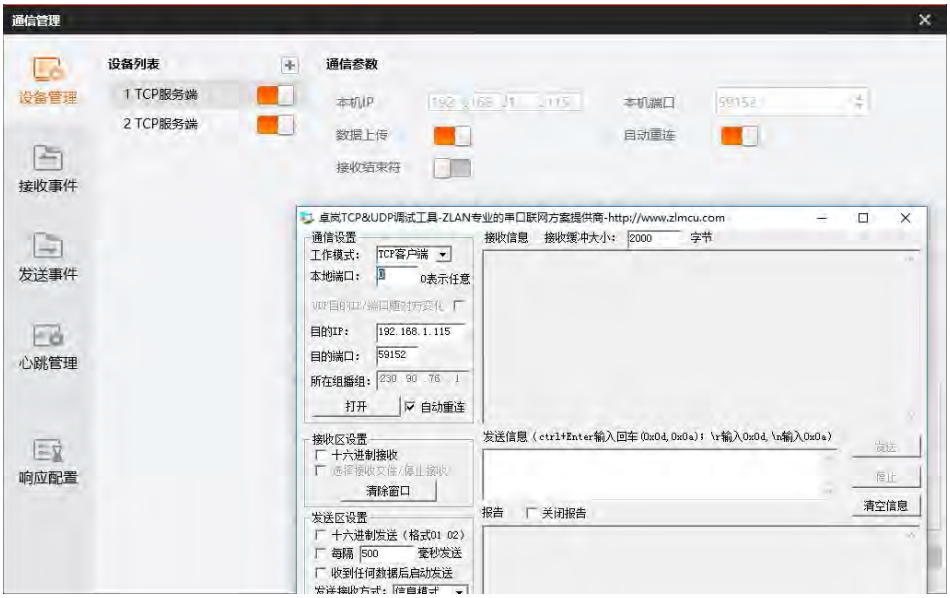
根据 xml 中设置的接收数据格式，

```
<RECEIVE>
;设置接收消息格式
    <XML>
        <ELEMENT  Tag="Sensor/X1"  Type="REAL"  Set_Flag="1"/>
        ；设置接收消息的元素名称，类型（REAL，INT，BOOL，STRING 等）
         <ELEMENT  Tag="Sensor/Y1"  Type="REAL"  />
         <ELEMENT  Tag="Sensor/Z1"  Type="REAL"  />
         <ELEMENT  Tag="Sensor/A1"  Type="REAL"  />
         <ELEMENT  Tag="Sensor/B1"  Type="REAL"  />
         <ELEMENT  Tag="Sensor/C1"  Type="REAL"  />
         <ELEMENT  Tag="Sensor/p"  Type="REAL"  />
    </XML>
</RECEIVE>
```

开始组装配置，

之后点击 2 发送数据 8 修改基本参数，将输出至修改为发送事件，发送事件选择刚刚生成的 0 文本-组装输出，

为了使接收的数据显示得更加直观，我们用 TCP&UDP 调试工具来显示发送的数据，我们需要创建一个 TCP 客户端来接收 VisionMaster 发送过来的数据，因此需要将目的 IP 改为 VisionMaster 1TCP 服务端的 IP192.168.1.115，端口号改为 59152，然后点击打开，



现在找到 VisionMaster 菜单栏的单次执行，

模块 2 发送数据 8 会向 TCP 客户端发送一次消息，

可以看到以 <Sensor> 开头，</Sensor> 结尾的数据，此数据的格式满足机械臂 xml 读取的要求。现在我们已经将视觉控制器数据传输到机械臂的方法也介绍完了。

现在又多出了一个问题，当我们想要为项目增加的更多功能时，在电脑上写了一个专门的软件，如何实现该软件与 VisionMaster 以及机械臂的通信呢？

10.5.3 其他软件与 VisionMaster 的交互

我们当然可以通过 VisionMaster 的 SDK 调用全局变量来交互数据，在这里我们使用另一种更加优雅的方法。

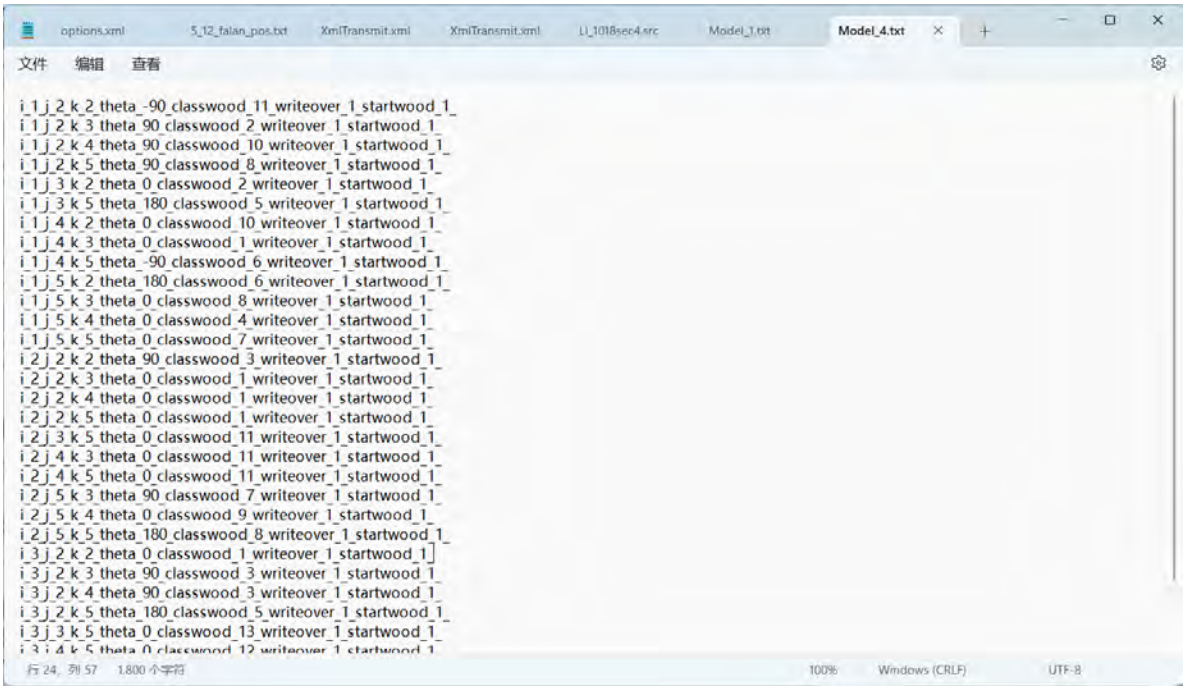
在项目中，我们运用到了接收数据，协议解析，协议组装，发送数据，条件检测，条件分支这些模块来实现我们的功能。信息的传输路径是外部 APP->VisionMaster->KUKA 机械臂。

(1) 创建第二个 TCP 服务端，地址依旧设为 192.168.1.115，端口设为 59151

(2) 按模块运行流程依次配置模块参数。

在 11 接收数据 2 模块选择上文创建的通信设备 2TCP 服务端，用于接收其他软件发送的数据。

修改 12 协议解析 2 解析接收的数据格式，在本项目中外部 APP 发送的信息格式如下所示。



数据格式与上文的协议解析设置相同，都是 string+float+string+……的格式

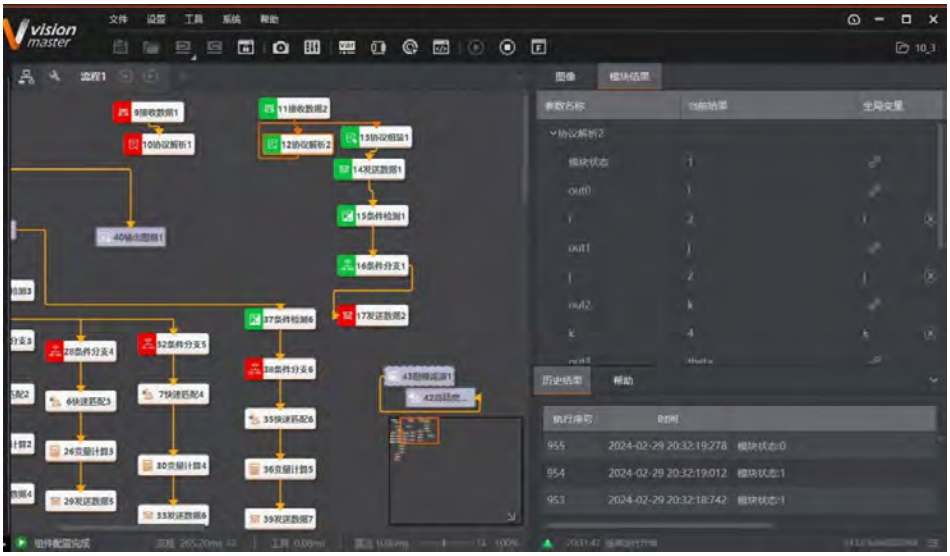


图 **10.12:** VisionMaster 中的模块结果

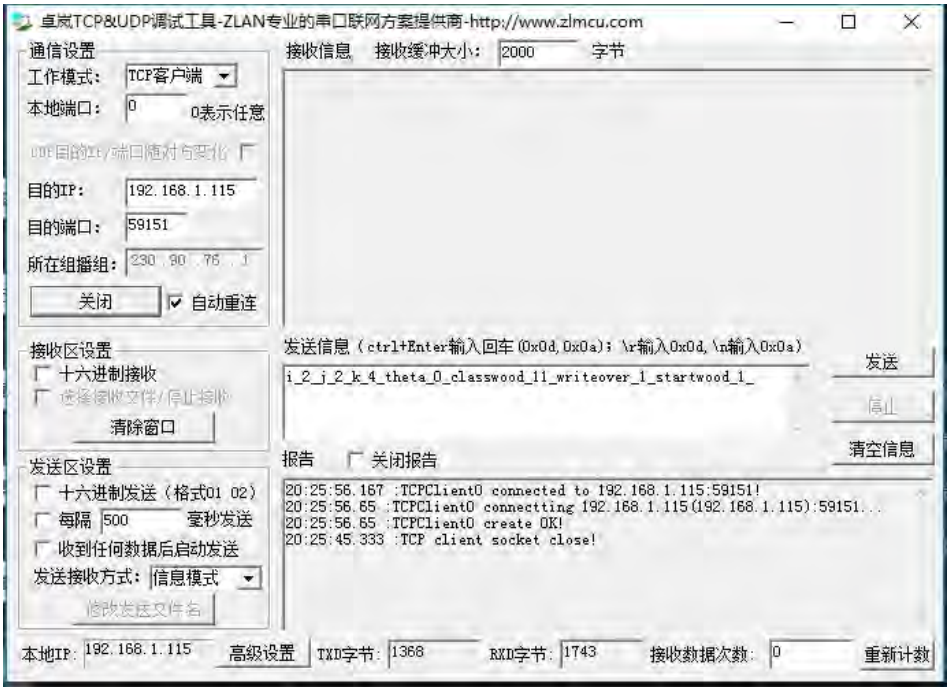
紧接着通过 13 协议组装 1 组装要发送给外部 APP 的应答消息。

然后将组合好的数据通过 14 发送数据传输回外部 APP。

到此处我们用 TCP&UDP 调试器进行测试。

首先打开 VisionMaster 的连续执行。

创建 TCP 客户端，将目的 IP 设为 192.168.1.115，目的端口 59151，并将客户端打开，



发送一行数据

i_2_j_2_k_4_theta_0_classwood_11_writeover_1_startwood_1_

可以看到在接受信息区域收到了 VisionMaster 回传过来的应答信息，本应收到的消息为：

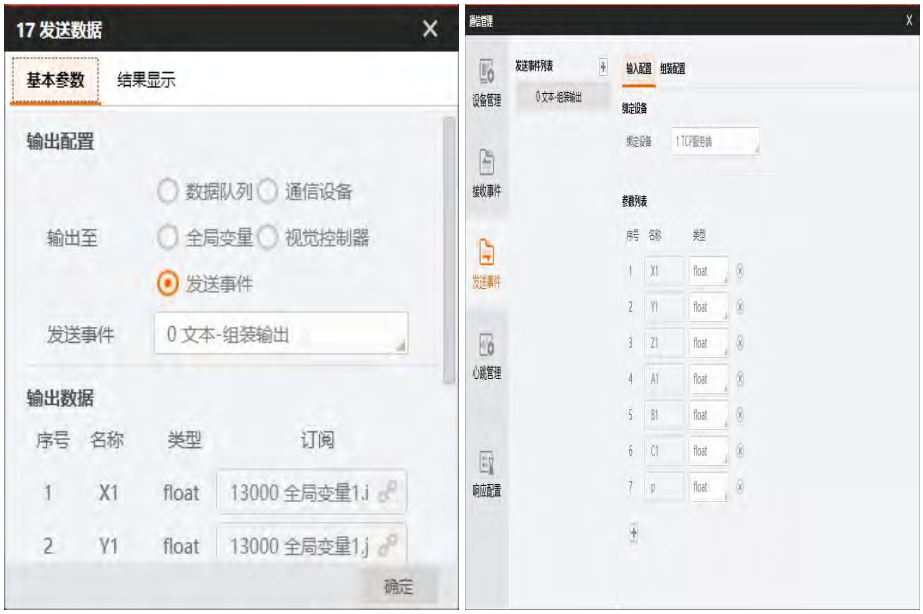
“收到消息，i_2_j_2_k_4_theta_0_classwood_11_writeover_1_startwood_1_，收完消息”

TCP&UDP 调试器上显示乱码是因为缺少了字体库，这并不影响数据的交互。

我们顺带讲一下模块 15 条件检测，判断方式选择全部即在下面创建的检测条件变量满足有效值范围内则判断结果为 OK，在本项目中通过协议解析的变量 i 来控制模块是否继续往下运行。

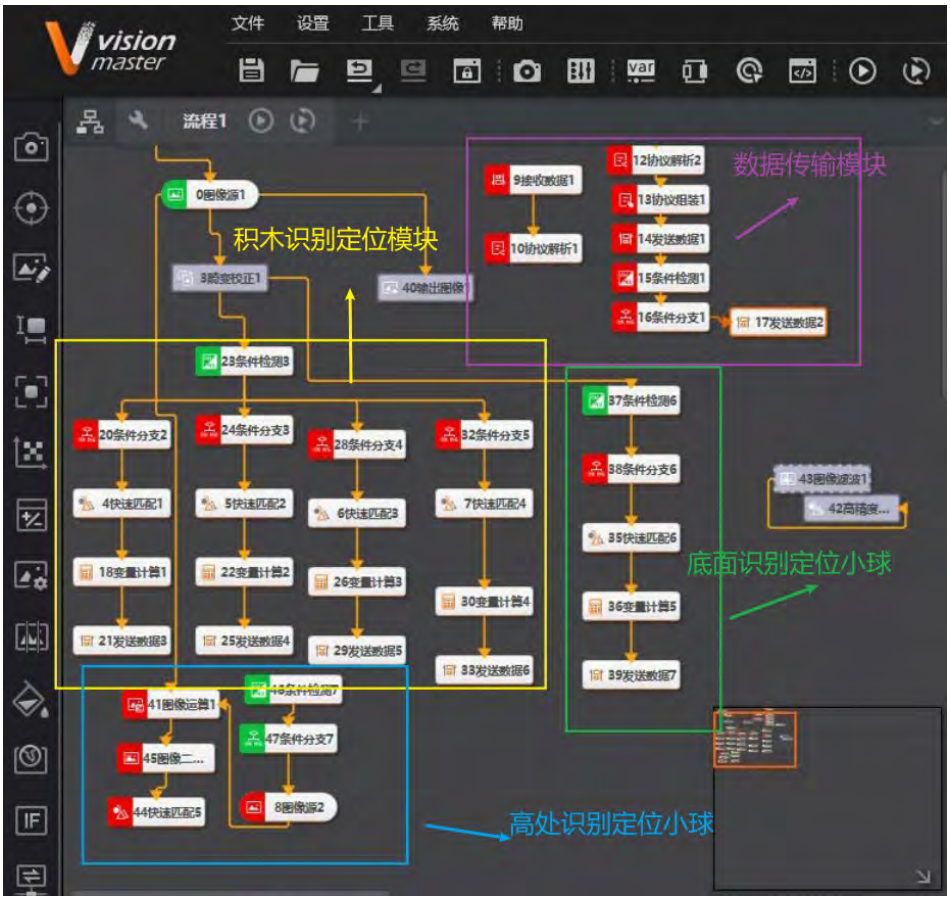
接着执行模块 16 条件分支，当模块 15 条件检测输出为 OK 时，即 15 条件检测结果状态的变量值为 1，满足条件之后执行模块 17

模块 17 的作用就是将协议解析得到的变量值传送给 KUKA 机械臂，发送事件选择上文所创建的 0 文本-组装输出，通过 1TCP 服务端交互数据。



到此各设备之间的数据交互已经讲完了。

10.6 视觉机械臂积木抓取项目总流程



视觉机械臂积木抓取项目总流程包括 4 个模块

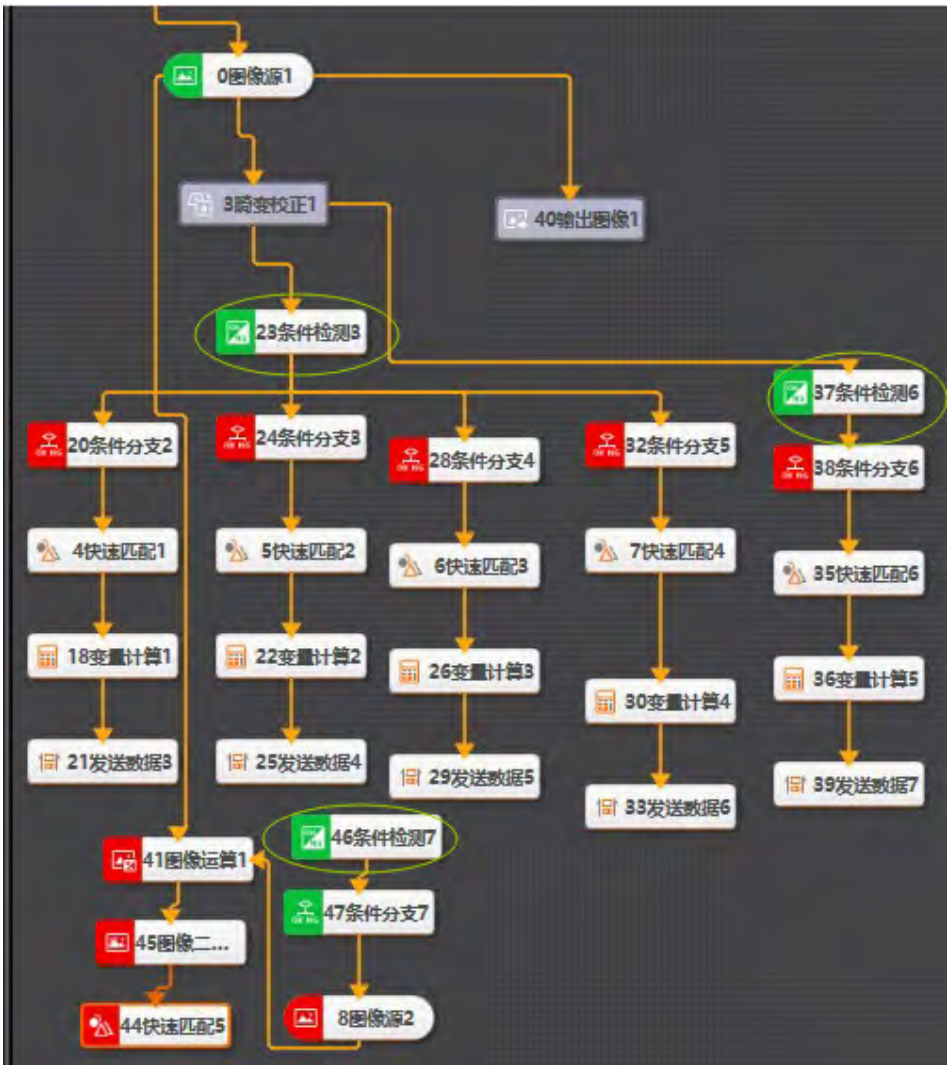
1. 数据传输模块

2. 积木识别定位模块

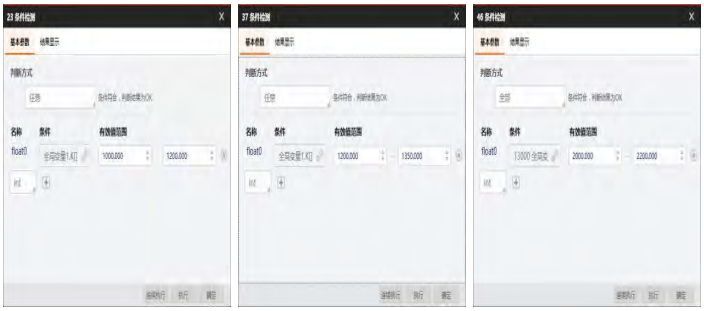
3. 高处识别定位小球模块

4. 低处识别定位小球模块

在本项目中通过 23 条件检测 3，46 条件检测 7，37 条件检测三个模块 6 分别控制 2 积木识别定位模块、3 高处识别定位小球模块，4 低处识别定位小球模块是否执行



这三个条件检测的模块都是通过同一个全局变量 K 的值来判断是否执行。



可以通过有效值范围的简单设置实现，在上图中

当我们将全局变量 K 值设置在 1000-1200 范围内时开始执行 2 积木识别定位模块，

全局变量 K 值设置在 1200-1350 范围内时开始执行 4 底面识别定位小球模块，

全局变量 K 值设置在 2000-2200 范围内时开始执行 3 高处识别定位小球模块。

这样我们就可以控制在识别积木时，其他两个模块能够处于休眠状态，很大程度上减少了视觉控制器的计算资源，提高了识别的速度。

其中的数据传输模块已经在上文中详细介绍了，接下来介绍 2 积木识别定位模块。关于积木模板识别的方法已经在上文章节“VisionMaster 功能模块”详细介绍过了，在下文中不再过多赘述。

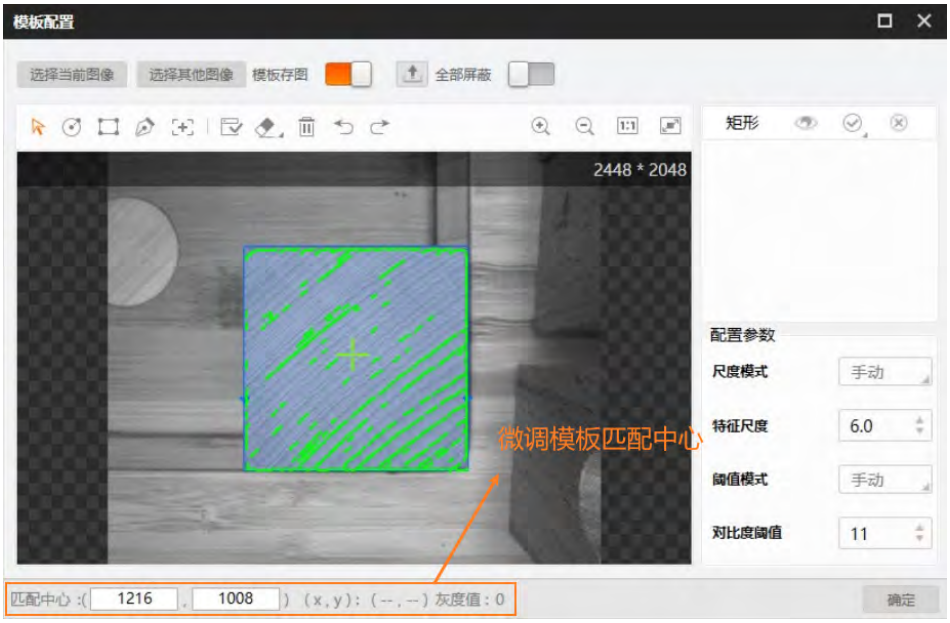
10.6.1 积木识别定位模块

当我们将全局变量 K 值设置在 1000-1200 范围内时开始执行积木识别定位模块。在这一模块中我们根据模板的分类 (1) 普通无轨道面 (2) 直线轨道面 (3) 圆弧轨道面 (4) 十字轨道面，将 Standard 标准款的 50 块积木的模板分为 4 类。

图 **10.13:** 4 类积木的 4 条执行路线

图 10.14: 4 个快速匹配分别存放了 4 种不同类型的积木轨道模板

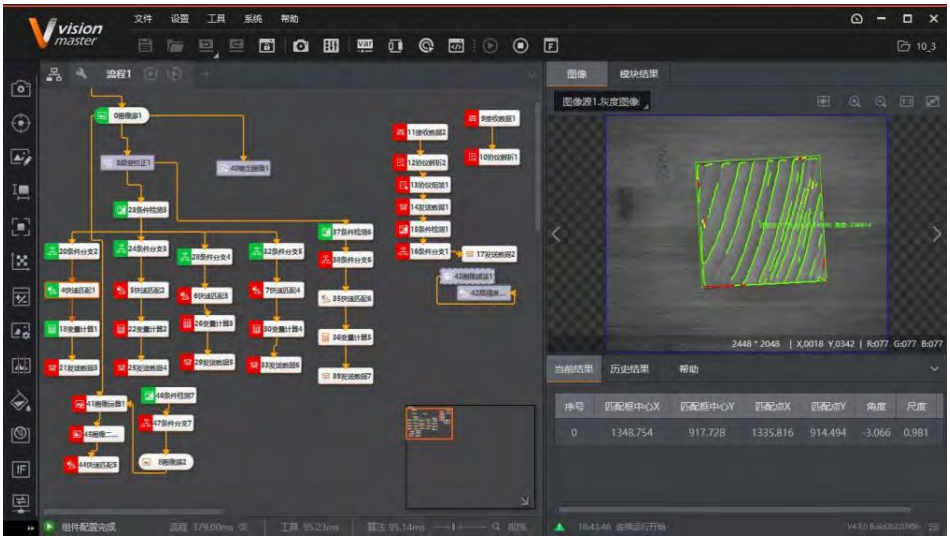
在本项目中我们对 50 块积木的轨道面都生成了木板，我们虽然可以用 4 个通用的木板匹配到轨道面，但是由于每个积木的大小不一致，匹配误差稍微大一些，就可能导致积木搭建失败，因此，考虑到精度，我们对每个积木模板的匹配中心位置进行了微调，从而达到了满意的效果。



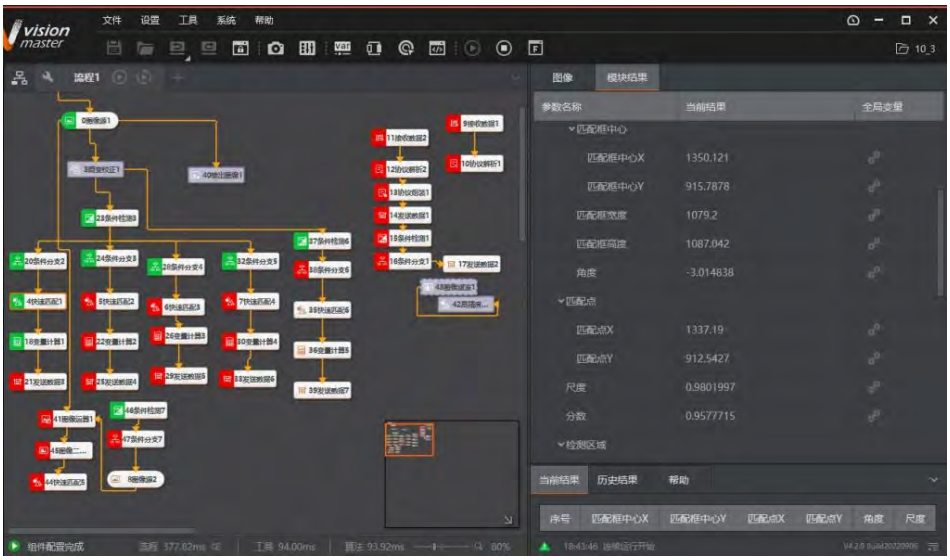
下面我们取最左边的执行路线为例，

此时 23 条件检测 3 木块已经满足条件，条件项状态的值为 1，在有效值范围之内，接着执行模块 ID4（4 快速匹配 1），

当 4 快速匹配 1 匹配到积木时



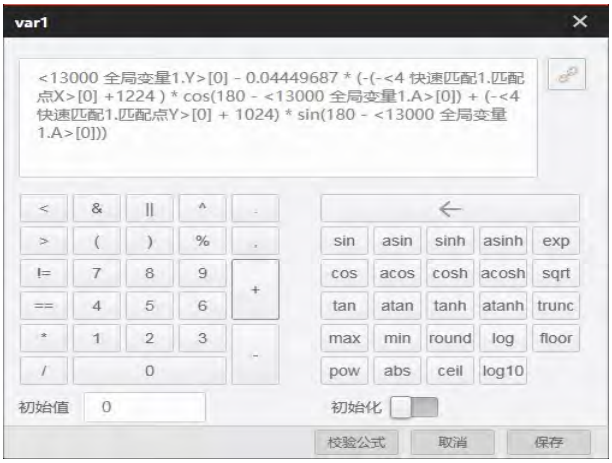
可以看到模块结果中我们得到了匹配框中心点的像素坐标，匹配框的角度，



然后将参数传到变量计算中，通过变量计算得出机械臂应该带抓取的积木位置。

变量 var0，var1 为传给机械臂抓取积木的 X,Y 坐标，var2 为传给机械臂的末端工具旋转角度 A。

接下来看看变量 var0，var1 对应的 X，Y 坐标是怎么计算的：



上式中全局变量 X，Y 是机械臂传输过来的实时坐标，0.04449687 是像素与实际距离之间的转换系数，后面括号中的是匹配点与像素中心点之间的距离，推导过程较为简单，不再赘述。下面介绍识别定位小球模块。

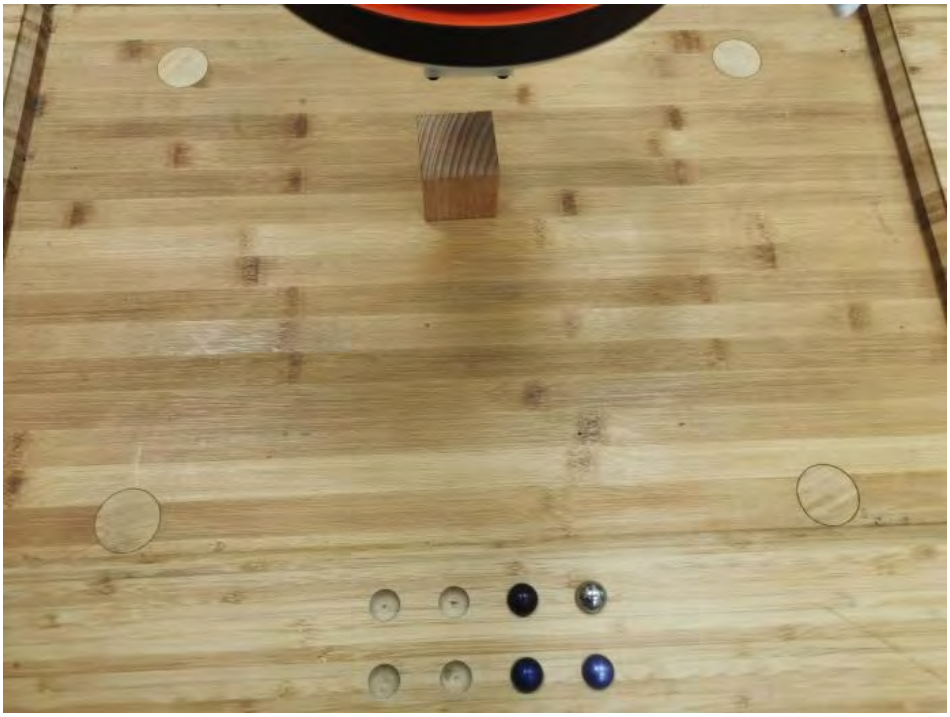


图 10.15: 小球抓取位置

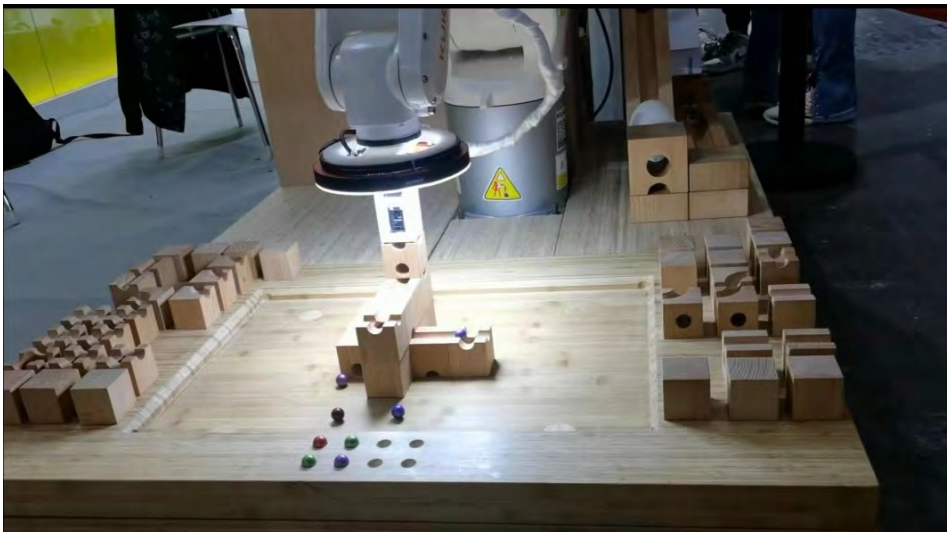


图 10.16: 木块搭建完成后放置小球

整个项目的机械臂动作流程为：识别抓取积木搭建—> 吸取小球放入积木轨道—> 将积木重新放于原位—> 识别定位抓取小球并放回至小球抓取位置—> 识别抓取积木搭建……

由此实现了流程的闭环控制，可以一直循环往复的运行下去。

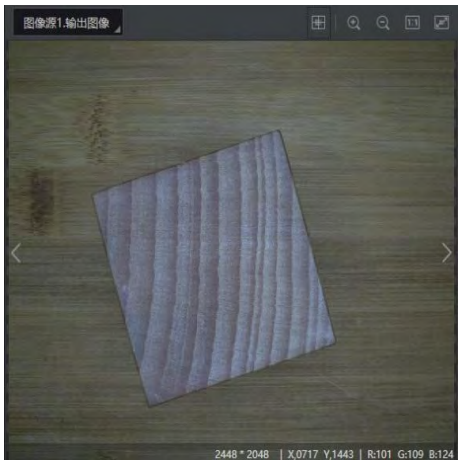
在识别定位抓取小球我们分为两个步骤进行：高处识别定位小球模块和低处识别定位小球模块。

10.6.2 高处识别定位小球模块

对于搭建完成后散落在木板内的小球，由于相机的拍摄视野是有限的，我们想要一次性识别到木板上的所有小球必须将相机抬高获取更大的视野。在这个过程中也相应的出现了两个两个问题。

问题 1：我们选用的 MV-CS-10UC 彩色面阵相机的相机焦距，光圈在机械臂运动过程中都是固定死的，相机并不能自动调焦，这就意味着在高处识别小球时画面会变得模糊，再加上在高处时，单位像素代表的实际距离增大，定位的精度会大幅下降。

问题 2：在底面设置曝光值 3500，获取到的明亮的图片，在高处就会变得非常暗，原因是我们采用的光源设备 MV-LRDS-H-200-90-W 的有效工作范围为 18 20cm，在高处光源的亮度就不够了，因此我们需要在相机中调高曝光值。（此时外界光对相机系统的影响会非常大）



低处图片明亮

高处亮度变暗

我们来看看解决方法。对于高处的照片较为模糊的问题我们通过图像运算和图像二值化得到了完美解决。

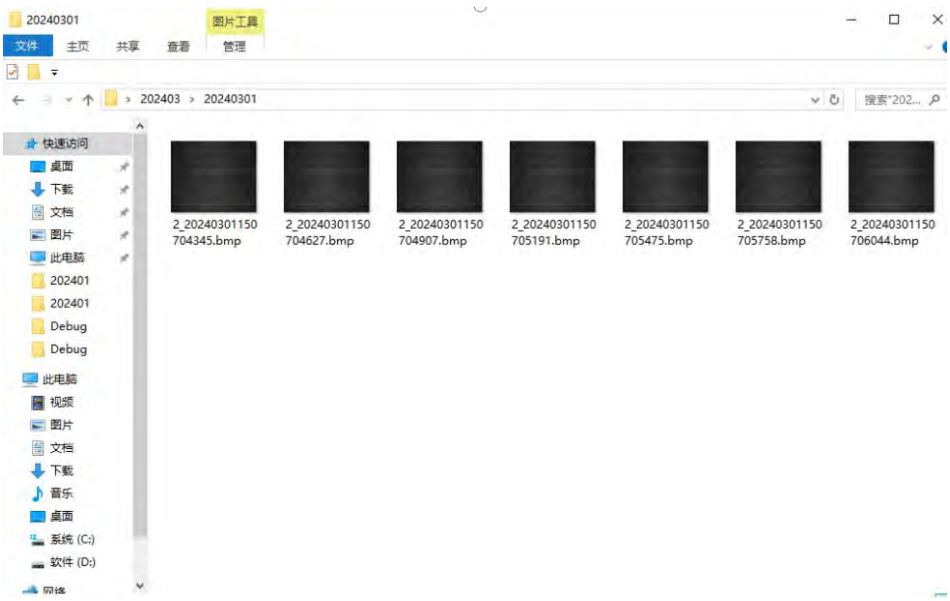
首先需要选定一个可以观察拍摄到所有小球的，固定高度位置，将机械臂移动到该位置，通过输出图像模块获取一张不含任何物体的背景图片。

输入源选择图像源 1 灰度图像，打开存图使能与生成目录，

之后打开保存原图，并设置图像的保存路径。

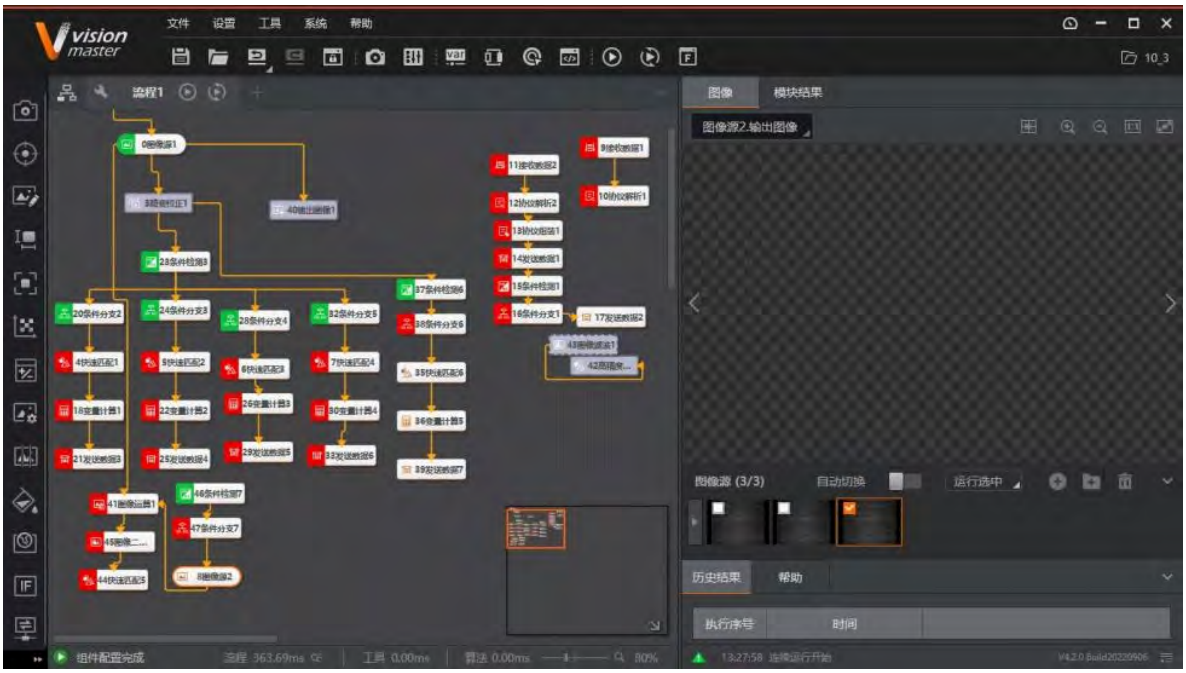
并选择输出使能，像素格式 RGB24

点击单次执行获取到背景图片



在 8 图像源 2 模块中图像源设置为本地图像，像素格式 MONO8，

在 VisionMaster 界面选择拍摄好的背景图，



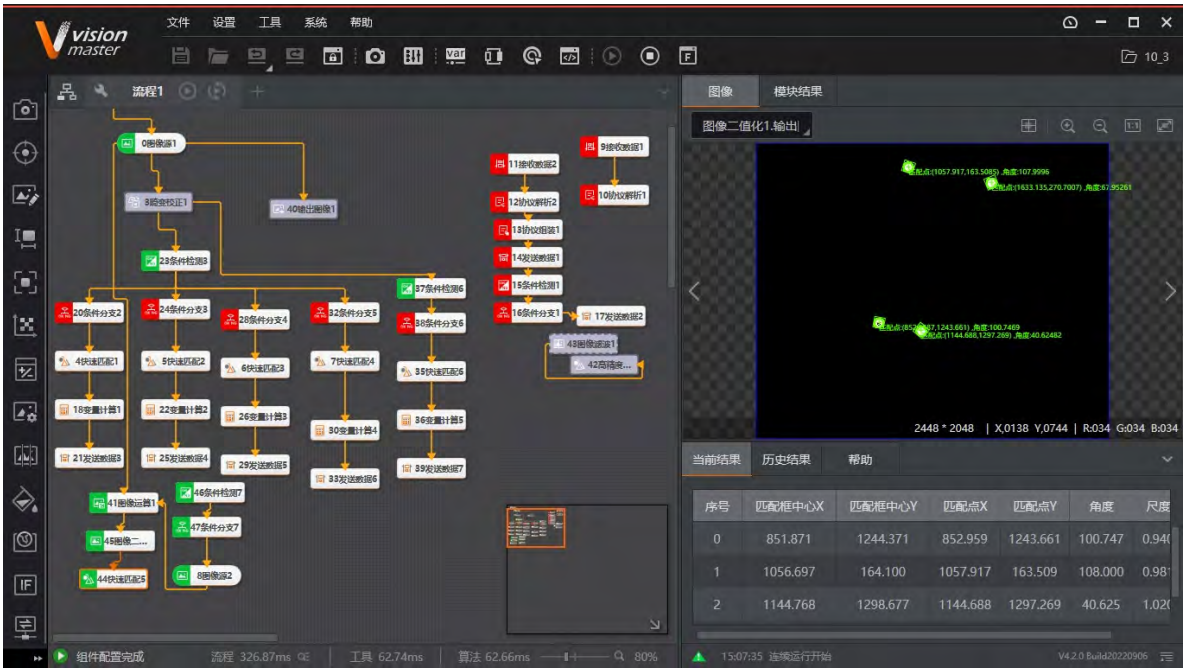
设置 41 图像运算 1 模块，将输入源 1 设置为图像源 1，将输入源 2 设置为图像源 2，

在图像运算参数设置为图像绝对差，由此可知，如果木板表面没有物体时，图像运算将输出纯黑色像素图像。

之后设置 45 图像二值化的. 输入源为上文的图像运算的输出图像，

运行参数修改为硬阈值二值化，

此时放入小球，让我们观察效果：



可以看到图像中的小球在此操作下会变得清晰明显，我们很容易就能识别到小球并进行定位。在此高处识别定位小球模块也讲解完了。

演示视频:

10.6.3 低处识别定位小球模块

在上文中我们分析了高处定位小球模块的缺点有精度较差，因此仅凭这一个步骤想要抓取小球是不切实际的。我们需要在高处粗略定位的基础上，在低处进行识别，精准定位。

低处识别定位小球模块的识别方法与积木识别定位的方法几乎完全相同，在此我们仅对不同点进行讲解。

与积木识别不同的是，识别小球只需要一个模板就可以了。因为小球表面并没有纹理，并且与木质表面的显示差异非常大，再由于小球的金属特性其大小误差要小于积木，所以只需要一个通用的模板即可精确识别定位到小球。

演示视频

到此整个项目的全部 VisionMaster 流程均已讲解完毕，视觉识别已经成功解决了。剩下的就是机械臂的运动控制代码，积木搭建的上位机模拟搭建，以及用于积木抓取的工具头硬件传感器模块的搭建。

```
&ACCESS RVP
&REL 12
&PARAM EDITMASK = *
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM DISKPATH = KRC:\R1\Program\Li_46
DEF LI_1018sec4( )
;FOLD Declaration
DECL  EKI_STATUS RET

;ENDFOLD (Declaration)
DECL INT i, j, k, iu, num,ball_get_i, i_num,j_num,k_num,BallRest
DECL REAL Num1Start_Z, Num2Start_Z, Num3Start_Z, Num4Start_Z, Num5Start_Z, Num6Strat_Z, Num7Start_Z, Num8Sta
DECL REAL Ball_StartX, Ball_StartY, Ball_StartZ, Threshold, Threshold_1
DECL REAL Ball_EndX, Ball_EndY, Ball_EndZ, scale_h,scale_low,z_high,z_low,Ball_HX,Ball_HY,Ball_HZ
DECL REAL Block_1x[24], Block_1y[24], Block_1z[24], ball_get_x[4],ball_get_y[4],Ball_locX[4],Ball_locY[4]
DECL REAL Block_2x[4], Block_2y[4], Block_2z[4]
DECL REAL Block_3x[4], Block_3y[4], Block_3z[4]
DECL REAL Block_4x[4], Block_4y[4], Block_4z[4]
DECL REAL Block_5x[4], Block_5y[4], Block_5z[4]
DECL REAL Block_6x[4], Block_6y[4], Block_6z[4]
DECL REAL Block_7x[4], Block_7y[4], Block_7z[4]
DECL REAL Block_8x[4], Block_8y[4], Block_8z[4]
DECL REAL Block_9x[4], Block_9y[4], Block_9z[4]
DECL REAL Block_10x[4], Block_10y[4], Block_10z[4]
DECL REAL Block_11x[6], Block_11y[6], Block_11z[6]
DECL REAL Block_12x[4], Block_12y[4], Block_12z[4]
DECL REAL Block_13x,Block_13y,Block_13z,down_hight,get_down_hight,limit_high
DECL REAL Left_Mountain_x_FL, Left_Mountain_y_FL, Left_Mountain_z_FL
DECL REAL DAA1,DAA2,DA41,ERROR_X,ERROR_Y
DECL REAL ACT_X,ACT_Y,q1,x_start,y_start_left,y_start_right,x_jianju,wait_sec
DECL POS STU
DECL INT Ball_Stu[4]
DECL REAL Ball_X[4],Ball_Y[4]
DECL INT num_1, num_2, num_3, num_4,num_5, num_6, num_7, num_8,num_9, num_10, num_11, num_12
DECL REAL TopX,TopY,DownX,DownY,BallDown
DECL INT A[5,6,6],A1[5,6,6]
DECL INT B[5,6,6]
DECL REAL Location_XStart, Location_YStart, Location_ZStart,class_wood,write_over,start_wood,theta,i1,j1,k1
DECL POS MY1
```

图 10.17: 机械臂运动控制代码

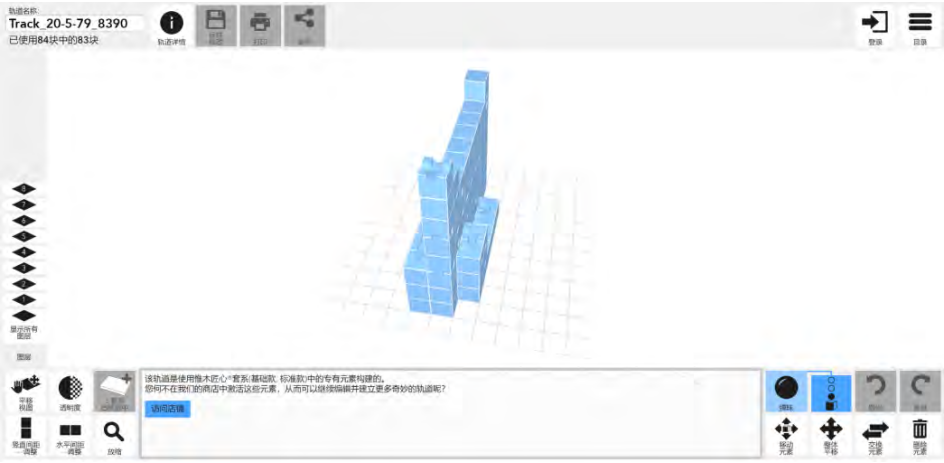


图 10.18: 网页端积木搭建

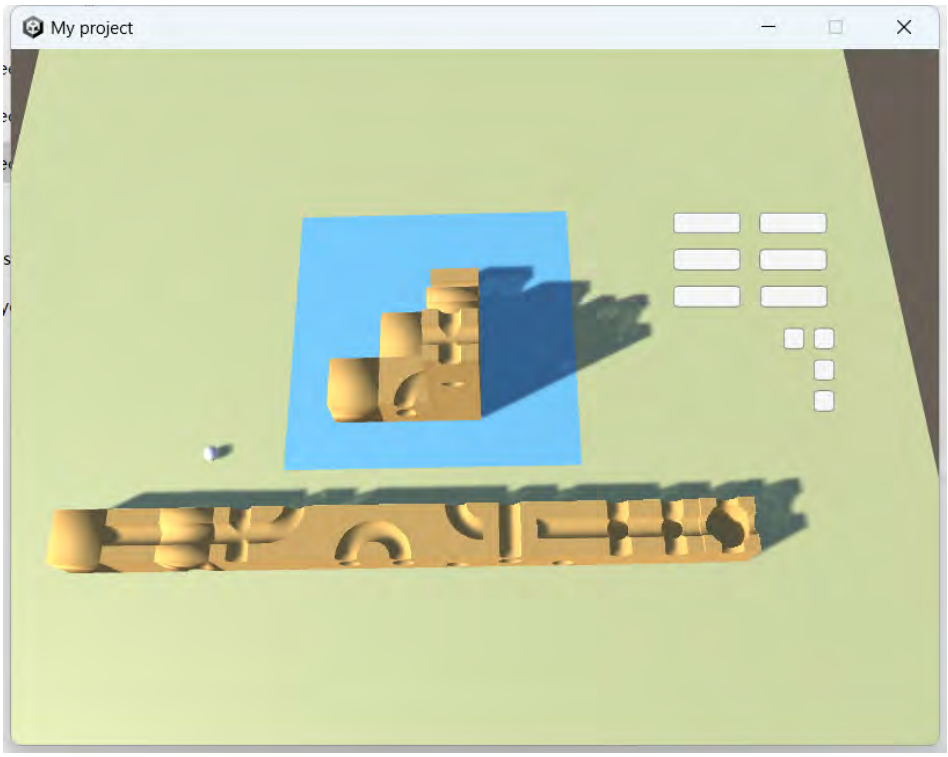


图 10.19: 软件端积木搭建

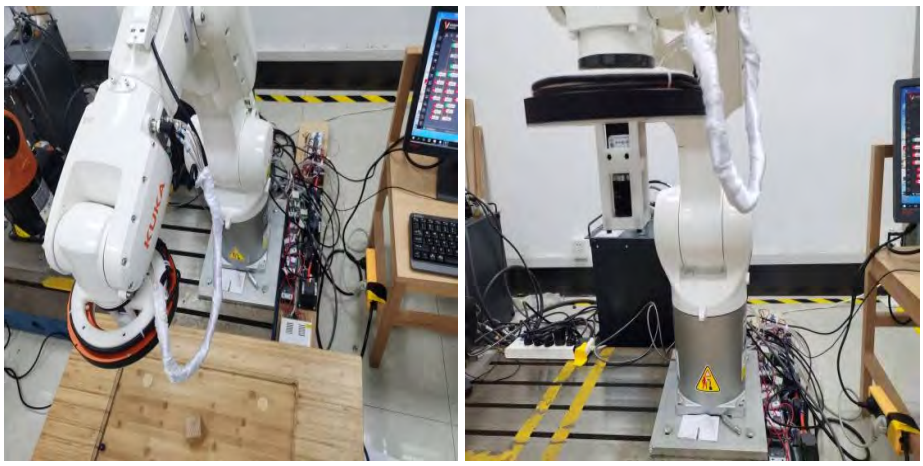


图 10.20: 机械臂末端积木抓取执行模块

10.7 KUKA 机器人代码详解

下面我们主要对 KUKA 示教器上的程序代码作详细解释，力求读者对于每一模块代码深入了解，理解其中设计原理及设计思路，阅读并理解其中代码设计，无论是对于读者的代码水平还是创作能力，都有很大的提升。

10.7.1 坐标定义

抓取积木，首先要做的是知道积木的位置，对于一块积木板上的积木，通过切换工具坐标系来示教的方式抓取，由于本次抓取使用的方法为二维平面识别方法，而本次相机为了积木抓取精度，需要事先了解积木的一个大概位置，从而移动于此区域，进行精确的积木识别并抓取。如图10.21所示，由于 KUKA 机器人的变量声明和定义放在两个不同的区域，故我们首先将所使用的全部变量进行声明。

```
7   ;FOLD Declaration
8   DECL  EKI_STATUS RET
9
10  ;ENDFOLD (Declaration)
11  DECL INT i, j, k, iu, num,ball_get_i, i_num,j_num,k_num,BallRest
12  DECL REAL Num1Start_Z, Num2Start_Z, Num3Start_Z, Num4Start_Z, Num5Start_Z, Num6Strat_Z, Num7Start_Z, Num8Start_Z, Num9Start_Z, Num10Start_
13  DECL REAL Ball_StartX, Ball_StartY, Ball_StartZ, Threshold, Threshold_1
14  DECL REAL Ball_EndX, Ball_EndY, Ball_EndZ, scale_h,scale_low,z_high,z_low,Ball_HX,Ball_HY,Ball_HZ
15  DECL REAL Block_1x[24], Block_1y[24], Block_1z[24], ball_get_x[4],ball_get_y[4],Ball_locX[4],Ball_locY[4]
16  DECL REAL Block_2x[4], Block_2y[4], Block_2z[4]
17  DECL REAL Block_3x[4], Block_3y[4], Block_3z[4]
18  DECL REAL Block_4x[4], Block_4y[4], Block_4z[4]
19  DECL REAL Block_5x[4], Block_5y[4], Block_5z[4]
20  DECL REAL Block_6x[4], Block_6y[4], Block_6z[4]
21  DECL REAL Block_7x[4], Block_7y[4], Block_7z[4]
22  DECL REAL Block_8x[4], Block_8y[4], Block_8z[4]
23  DECL REAL Block_9x[4], Block_9y[4], Block_9z[4]
24  DECL REAL Block_10x[4], Block_10y[4], Block_10z[4]
25  DECL REAL Block_11x[6], Block_11y[6], Block_11z[6]
26  DECL REAL Block_12x[4], Block_12y[4], Block_12z[4]
27  DECL REAL Block_13x,Block_13y,Block_13z,down_hight,get_down_hight,limit_high
28  DECL REAL Left_Mountain_x_FL, Left_Mountain_y_FL, Left_Mountain_z_FL
29  DECL REAL DAA1,DAA2,DA41,ERROR_X,ERROR_Y
30  DECL REAL ACT_X,ACT_Y,q1,x_start,y_start_left,y_start_right,x_jianju,wait_sec
31  DECL POS STU
32  DECL INT Ball_Stu[4]
33  DECL REAL Ball_X[4],Ball_Y[4]
34  DECL INT num_1, num_2, num_3, num_4,num_5, num_6, num_7, num_8,num_9, num_10, num_11, num_12
35  DECL REAL TopX,TopY,DownX,DownY,BallDown
36  DECL INT A[5,6,6],A1[5,6,6]
37  DECL INT B[5,6,6]
38  DECL REAL Location_XStart, Location_YStart, Location_ZStart,class_wood,write_over,start_wood,theta,i1,j1,k1
39  DECL POS MY1
40  DECL POS STU1
41  DECL REAL p, p1, p2
42  DECL BOOL Ball_Flag[4]
43  DECL INT NUM_Ball, new
44  DECL INT Ball_St[4]
```

图 10.21: 定义的声明

在图中的声明中，主要分为四种类型，我们分别讲解 DECL INT 用来定义一个整型变量或者整型数组，DECL REAL 用来定义一个浮点型变量或浮点型数组，浮点型即数字后可以带小数点，一般机器人坐标大部分都是要用浮点数来说明，它们的值均精确到小数点后几位，还有一些布尔类型的数据，布尔类型的变量仅仅可以存储 0 或 1。值得强调的是 DECL POS，类似于一个结构体类型，通过索引 X,Y,Z,A,B,C，直接将一点的坐标赋值在这个变量中去，通过 PTP,LIN 运动语句直接运动于该坐标点。讲完变量的定义，接下来肯定是要将变量进行赋值咯！不是这样的，对于 KUKA 机器人编程而言，我们既然是要将测定的坐标赋值到变量上去，那么就要知道坐标是在哪个坐标系下的，因此，首先要进行 PTP 到我们的 HOME 点，代码如图所示，将坐标系调整为测量坐标时所用的坐标系，这一点相信大家都在之前章节已经掌握，如果读者遇到问题，可以翻到之前的章节进行重新学习。

```
;FOLD PTP LI Vel=100 % PDAT5 Tool[6] Base[0] ;%{PE}
;FOLD Parameters ;%{h}
;Params IlfProvider=kukaroboter.basistech.inlineforms.movement.old; Kuka.IsGlobalPoint=False; Kuka.PointName=LI;
;ENDFOLD
$BWDSTART = FALSE
PDAT_ACT = PPDAT5
FDAT_ACT = FLI
BAS(#PTP_PARAMS, 100.0)
SET_CD_PARAMS (0)
PTP XLI
;ENDFOLD
```

图 10.22: PTP 到 HOME 点，设置坐标系

接下来，我们首先将需要设置传递给上位机的信号，如图10.23所示，为我们和上位机通信时使用的各个参数，我们首先将它们的初值设置一下。

```
57  BAS (#INITMOV,0 )
58  RET=EKI_Init("XmlTransmit")
59  RET=EKI_Open("XmlTransmit")
60  RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
61  RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@e",3500)
62  RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@q",0)
63  RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",0)
```

图 10.23: 设置传给上位机的初值

其中，第 58 行和 59 行必须有，两行代码将上位机通信打开，开始通信。60-64 行分别为初始化设置传递给上位机的值，每增加一个与上位机通信的值，都必须在这里初始化一次，否则下方会造成异常，在这里，参数 f 值的多少会决定相机选择哪个模块进行模板匹配，一般来说，不同积木会有不同的模板，而每次仅仅运行一个模板的识别会大大减少算法时间，故该参数是必须的。参数 e 帮助我们设置相机的曝光，由于不同的模块的位置，需要将相机的曝光值设置为不同的值，方便相机清晰地识别并完成所需的任务。参数 q 在我们每次进行小球抓取中，会清楚地记录还剩余几颗小球为抓取，从而传递给我们上位机进行程序更改。参数 n 将设置我们何时使用相机，由于相机总是给我们传递视频流，而视频流是每次抽取其中的一帧进行识别并返回结果，而通过该参数进行触发设置，每次相机使用时拍摄一张清晰图片，即可进行我们的任务，可以提高我们算法的效率和稳定性。接下来是一行 WHILE TRUE，这行代码就是循环

代码，WHILE 后跟循环停止条件，由于此时是 TRUE，故改行下面直到 ENDWHILE，将无限进行循环，这也是我们所设置的，命令其进行重复地轨道积木的搭建。下面我们要将机械臂对外部设备的输出端进行初始化。

```
78  $OUT[1] = TRUE
79  $OUT[2] = TRUE
80  $OUT[12] = FALSE
81  $OUT[3] = FALSE
82  $OUT[4] = FALSE
83  $OUT[5] = FALSE
84  $OUT[6] = FALSE
85  $OUT[8] = FALSE
86  $OUT[9] = FALSE
87  $OUT[10] = FALSE
88  $OUT[11] = FALSE
89  $OUT[12] = FALSE
```

图 10.24: 设置输出端初值

下面我们设置从下往上数第一排、从右往左数第一列的积木位置初值，该点通过示教获得，我们可以将机器人法兰垂直朝下，通过切换法兰坐标系进行示教，查看当前坐标来设置。紧接着下面四行代码设置了积木搭建整个平台的四个边界，防止积木出现越界行为。

```
90  Location_XStart = 493 + 48
91  Location_YStart = 77 - 48
92  Location_ZStart = 181.75 - 0.4
93  ;Set Table Edge Value
94  X_TOP = 783
95  X_DOWN = 426.5
96  Y_TOP = 230
97  Y_DOWN = -326
```

图 10.25: 初始化积木位置和边界

设置各项数值为初始值，大部分均为 0，各个变量的作用会在后续具体模块中进行说明，请读着耐心往下阅读。

```
;Done
p = 0
q1 = 0
i1=0
j1=0
k1=0
theta=0
class_wood=0
write_over=0
start_wood=0
i = 0
j = 0
k = 0
i_num = 0
j_num = 0
k_num = 0
num = 0

Threshold = 0.1
Threshold_1 = 0.6
DA1=0.0
DA2=0.0
DA3=0.0
DA4=0.0
DA5=0.0
DA6=0.0
DAA1=0
DAA2=0
iu=1
number1=0
ERROR_X = 0
ERROR_Y = 0
```

图 10.26: 初始化各项变量

由于后续我们要进行小球的抓取，故我们提前将小球的位置通过示教得出，将其存储在变量中，同时将小球最后要落下的位置提前进行计算得出。

```
;We set the Ball Start

Ball_StartX = 854.73
Ball_StartY = -14.85
Ball_StartZ = 161
;We set the Ball End
Ball_EndX = Location_XStart + 17
Ball_EndY = Location_YStart - 16
Ball_EndZ = Location_ZStart + 4 * 48
;We set all type block's START
```

图 10.27: 初始化小球的位置

接下来，我们将定义每块积木坐标的变量，由于积木仅放置一层，故得到每块积木平面的 x 和 y 坐标是我们要做的事，为了方便，我们将每块积木按照固定的间距进行摆放，如图10.28所示。

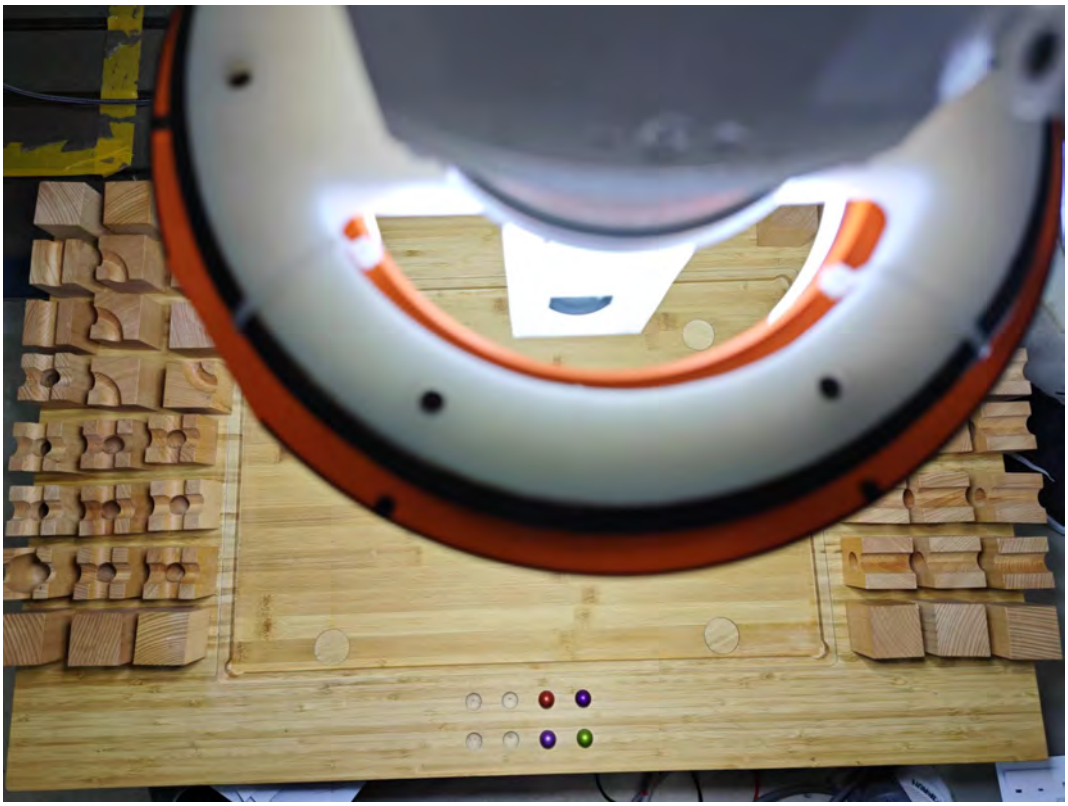


图 10.28: 初始化小球的位置

从而我们将最靠近机器人的一排积木 X 坐标通过示教得出，其余积木的 X 坐标均可以通过固定间距计算得到，再分别将两侧靠近内侧积木的 Y 坐标示教得到，再通过固定间距得到任意一块积木 Y 坐标。通过上述说明，我们这里仅通过设置 4 个变量即可，如图所示。

```
155
156  ;x_start开始x
157  x_start=335
158  y_start_left=292
159  y_start_right=-400
160  x_jianju=62
161  ;-----------------------------
162
```

图 10.29: 设置初始坐标值与积木间距

那么接下来，我们将各个种类积木的坐标按照所定的间距进行计算得到，如图10.30所示，为一号积木计算所得 X，Y 坐标。

```
163  Block_1x[2] = x_start
164  Block_1x[3] = x_start
165  Block_1x[4] = x_start
166  Block_1x[5] = x_start + x_jianju
167  Block_1x[6] = x_start + x_jianju
168  Block_1x[7] = x_start + x_jianju
169  Block_1x[8] = x_start + x_jianju * 2
170  Block_1x[9] = x_start + x_jianju * 2
171  Block_1x[1] = x_start + 12
172  Block_1x[10] = x_start + 12
173  Block_1x[11] = x_start
174  Block_1x[12] = x_start
175  Block_1x[13] = x_start
176  Block_1x[14] = x_start + x_jianju * 7
177  Block_1x[15] = x_start + x_jianju * 7
178  Block_1x[16] = x_start + x_jianju * 7
179  Block_1x[17] = x_start + x_jianju * 6
180  Block_1x[18] = x_start + x_jianju * 7
181  Block_1x[19] = x_start + x_jianju * 7
```

```
183  Block_1y[2] = y_start_left
184  Block_1y[3] = y_start_left + 65
185  Block_1y[4] = y_start_left + 65 * 2
186  Block_1y[5] = y_start_left
187  Block_1y[6] = y_start_left + 65
188  Block_1y[7] = y_start_left + 65 * 2
189  Block_1y[8] = y_start_left
190  Block_1y[9] = y_start_left + 65
191  Block_1y[10] = y_start_right + 65
192  Block_1y[11] = y_start_right
193  Block_1y[12] = y_start_right - 65
194  Block_1y[1] = y_start_left- 65
195  Block_1y[13] = y_start_right - 65 * 2
196  Block_1y[14] = y_start_left
197  Block_1y[15] = y_start_left + 65
198  Block_1y[16] = y_start_left + 65 * 2
199  Block_1y[17] = y_start_right - 65 * 2
200  Block_1y[18] = y_start_right - 65
201  Block_1y[19] = y_start_right - 65 * 2
```

图 10.30: 计算所得一号积木 X,Y 坐标

由于每块积木所放置的高度相同，从而将每块积木的 Z 轴坐标均设为相同的，如图10.31所示。

```
;Then Setting Z
FOR i = 1 TO 20  STEP 1
      Block_1z[i] = Num1Start_Z - 48
ENDFOR
;Let us set the 2th-12th(except 11th) Block Location
FOR i = 1 TO 4 STEP 1
   Block_2z[i] = Num2Start_Z - 48
   Block_3z[i] = Num3Start_Z - 48
   Block_4z[i] = Num4Start_Z - 48
   Block_5z[i] = Num5Start_Z - 48
   Block_6z[i] = Num6Strat_Z - 48
   Block_7z[i] = Num7Start_Z - 48
   Block_8z[i] = Num8Start_Z - 48
   Block_9z[i] = Num9Start_Z - 48
   Block_10z[i] = Num10Start_Z - 48
   Block_12z[i] = Num12Start_Z - 48
ENDFOR
;Let us set the 2th-12th(except 10th) Block Location
For i = 1 TO 6 STEP 1
      Block_11z[i] = Num11Start_Z - 48
ENDFOR
```

图 10.31: 设置初始坐标值与积木间距

下面对每个位置上的积木进行编码，我们要搭建一个属于我们自己的轨道积木，那么必须实现直到每个位置是否应该放置积木以及应该放置哪块积木，我们通过建立一个三维的数组，数组中的值分别为 0-12，对应没有积木以及 1 到 12 号积木种类。除此之外，积木还存在旋转的情况，我们另作一个三维数组变量，存储积木的旋转信息，分别为-90,90,180 三种，我们先对其初始化为 0，后分别设置各个位置的积木种类和旋转。

```
FOR i = 1 TO 5 STEP 1
   FOR j = 1 TO 6 STEP 1
      FOR k = 1 TO 6 STEP 1
         A[i,j,k] = 0
         A1[i,j,k] = 0
      ENDFOR
   ENDFOR
ENDFOR
;We still need to make the degree of every block
FOR i = 1 TO 5 STEP 1
   FOR j = 1 TO 6 STEP 1
      FOR k = 1 TO 6 STEP 1
         B[i,j,k] = 0
      ENDFOR
    ENDFOR
      ENDFOR
```

```
A[1,1,3] = 11
A[1,2,3] = 3
A[1,3,1] = 1
A[1,3,2] = 1
A[1,3,3] = 2
A[1,3,4] = 1
A[1,4,1] = 5
A[1,4,2] = 10
A[1,4,3] = 8
A[2,3,1] = 8
A[2,3,2] = 2
A[2,3,3] = 3
A[2,3,4] = 11
A[3,3,1] = 12
B[1,1,3] = 0
B[1,4,3] = 180
B[1,4,2] = 90
B[2,3,1] = -90
B[2,3,2] = 90
B[2,3,3] = 90
B[2,3,4] = 90
B[3,3,1] = -90
```

图 10.32: 计算所得一号积木 X,Y 坐标

10.7.2 抓取积木

下面我们进入抓取积木板块，我们的主要思想是，在一个 5*5*5 的立体空间中从底层向上依次放置积木。我们对我们在上面创建的一个三维矩阵进行遍历，当该位置为 0 时跳过，当该位置为 1-12 时，我们去抓取 1 号到 12 号积木中的一个。我们以 11 号积木为例来进行讲解，其余种类积木仅仅改变编号，写法几乎相同。首先我们需要记录当前该抓取第几个 11 号积木，从而我们需要提前定义 12 个变量来记录每块积木（如图10.33所示），，每当我们执行一次抓取代码，就需要将该变量进行更新。

```
num_1 = 1
num_2 = 1
num_3 = 1
num_4 = 1
num_5 = 1
num_6 = 1
num_7 = 1
num_8 = 1
num_9 = 1
num_10 = 1
num_11 = 1
num_12 = 1
```

图 10.33: 初始化每种积木的位置

接下来就直接抓取积木了，首先我们要运动到积木上方，如图10.34所示代码，我们将 11 号积木的第一块 X,Y 坐标赋值给当前位置的变量，高度为当前高度向上 72.72mm，并且向该点移动，如图右侧为已经到达改点。

```
IF A[i,j,k] == 12 THEN
   ;Get the Block Previewly
   MY1.X = Block_12x[num_12]
   MY1.Y = Block_12y[num_12]
   MY1.Z = Block_12z[num_12] + 72.72
   MY1.A = -180
   MY1.B = 0
   MY1.C = 180
   PTP MY1
```

图 10.34: 代码以及当前移动位置

接下来进入关键部分，通过模板匹配进行目标追踪，首先如图**??**所示，num 变量为了控制识别次数，在这里我们设置最多识别两次即进行积木的抓取，下面设置一个 WAIT 语句，从而当代码运动到那一点再开始识别，紧接着两行，用来消除当前传给机器人的变量缓存，防止接受旧的数据，并且将 FLAG 设置为 FALSE，从而可以进入下面 WHILE 识别语句中，下面的判断当前 num 的值，即当 num 大于 2 时，就不再进行识别，立刻下去抓取积木。再往下，设置 f 的值为 1111 为了触发识别木块代码，下面设置 n 为一个特殊数字，为了触发上位机相机的拍照模块，从而进行识别。再往下的就通过该 WAIT FOR 语句，通过 EKI_GetReal 获取上位机计算后应该调整的坐标，将该坐标分别赋值到 DA1 到 DA6，在此处还需再进行一些等待，否则可能会造成数据没有接受成功，同时这几行代码被 number==1 和 number=2 包含，这两行代码是数据接受的起始和终止。

```
num = 0
WAIT SEC 0.02

;Get the Block
RET=EKI_ClearBuffer("XmlTransmit","Sensor")
$FLAG[1] = FALSE
;FOLD Write data to connection

RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
WHILE TRUE

IF num >= 2 THEN
   EXIT
ENDIF

RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",1111)
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",98765111)
WAIT FOR  ( $FLAG[1] )
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",0)
number=1
RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
WAIT SEC wait_sec
number=2
```

图 10.35: 识别代码 1

接下来，我们判断要调整的位置和当前位置的差值，如若两者差值小于一定的区间，我们认为此时就是中心，此时退出循环，可以下去抓取积木，如若不然，需要将调整位置赋值为要运动的位置，并且进行调整，继续我们的循环。

```
IF ((ABS(DAA1 - DA1))<Threshold) AND (ABS(DAA2-DA2)<Threshold) AND (ABS(DA4)<0.04) THEN
   EXIT
ELSE
DAA1=DA1
DAA2=DA2
STU.X=DA1
STU.Y=DA2
STU.Z=Block_13z + 72.72
STU.A=$POS_ACT.A-DA4
STU.B=0
STU.C=-180
DA41=DA4

PTP STU

WAIT SEC 0.01
ENDIF
RET=EKI_ClearBuffer("XmlTransmit","Sensor")
$FLAG[1] = FALSE

WAIT SEC 0.01
num = num + 1

ENDWHILE
```

图 10.36: 识别代码 2

当我们从循环中退出就该下去抓取积木了，我们此时仅需将 Z 坐标向下减少到积木的高度即可，此时我们通过设置气泵开启，从而吸着积木将它抓起。然而，每块积木高度不等或地面不平，我们需要检验是否每个气泵都吸到了积木，下面的一个 WHILE TRUE 语句即在判断是否四个气管均吸到了积木。之前我们介绍到，气泵处连接着四个气体压力传感器，通过判断传感器的输入来判断积木是否吸取到，如若存在没有吸到的气管，我们就将高度进行下降，当然下降也要有个阈值，不能一直下降。

```
STU.Z = Block_13z
PTP STU
get_down_hight=STU.Z
STU.Z = Block_13z + 144
;Suck
$OUT[12] = TRUE
$OUT[10] = TRUE
$OUT[9] =  TRUE
$OUT[11] = TRUE
 down_hight=0
  WHILE TRUE
   IF ($IN[12]==TRUE) AND ($IN[13]==TRUE) AND ($IN[14]==TRUE) AND ($IN[15]==TRUE) THEN
      down_hight=0
      EXIT
   ELSE
      WAIT SEC 0.1
      down_hight=down_hight+0.05
      IF down_hight<limit_high THEN
      STU.Z=get_down_hight-down_hight
      PTP STU
      ENDIF
   ENDIF
ENDWHILE
```

图 10.37: 下去抓积木代码

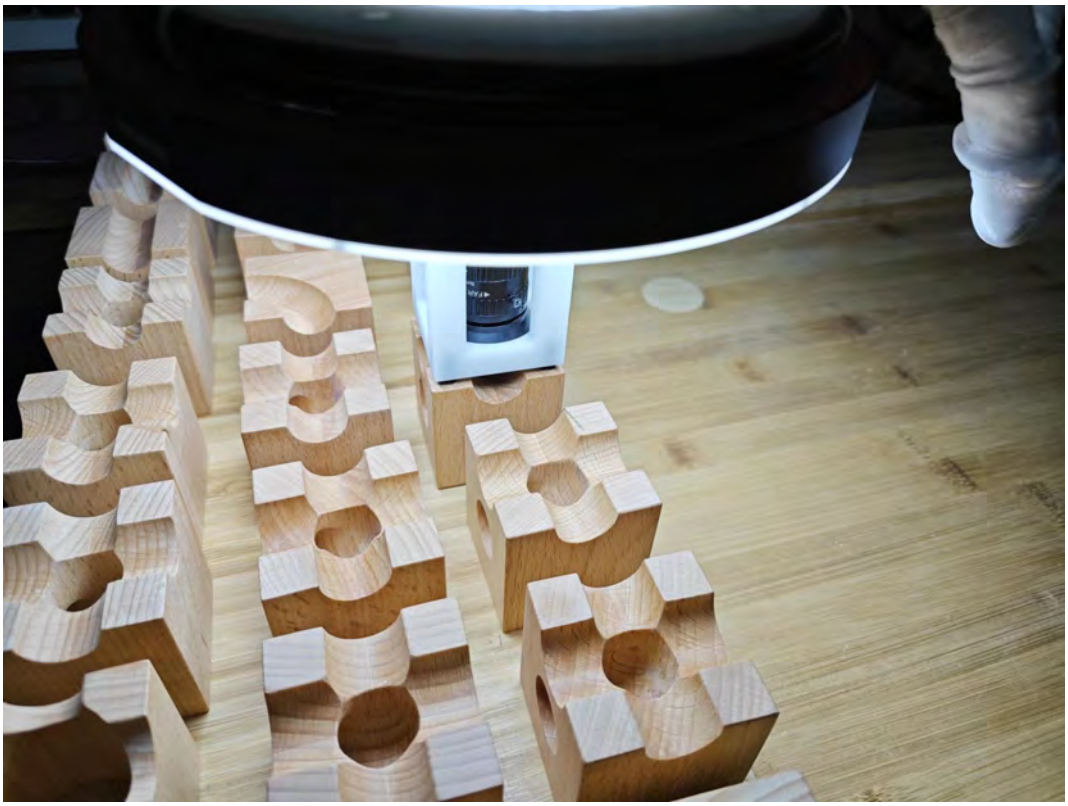


图 10.38: 正在抓积木

图 **10.39:** 抬起积木

待我们抓到积木后，在下面的代码中（如图10.40），我们将积木放置到距离待放点高约 50mm 的位置。我们先讲积木上升一定高度（如图所示），这个高度一定要足够，确保中途不会碰到其他积木，接下来我们将积木放到预放下的位置，我们知道，积木放下的精确位置即为起始点加减 48 的倍数，然而，如若垂直放下，难免可能会碰到之前已经放下的积木，我们这里选择斜着贴过去，先将 X,Y 同时改变一些，提高一个待放位置较高的位置，之后进行两次降低，将积木稳稳贴在已放置积木旁。

```
;Promote
STU.X = STU.X
STU.Y = STU.Y
STU.Z = Block_13z + 144
PTP STU
;Lay out Previewly
MY1.X = Location_XStart + (k - 1) * 48 + 15
MY1.Y = Location_YStart - (j - 1) * 48 - 15
MY1.Z = Location_ZStart + (i - 1) * 48 + 144
MY1.A = B[i,j,k] * (-1)
MY1.B = 180
MY1.C = 0
PTP MY1
;Lay out Previewly again
MY1.X = MY1.X - 2.5
MY1.Y = MY1.Y + 2.5
MY1.Z = MY1.Z - 90
PTP MY1
```

图 **10.40:** 放置积木

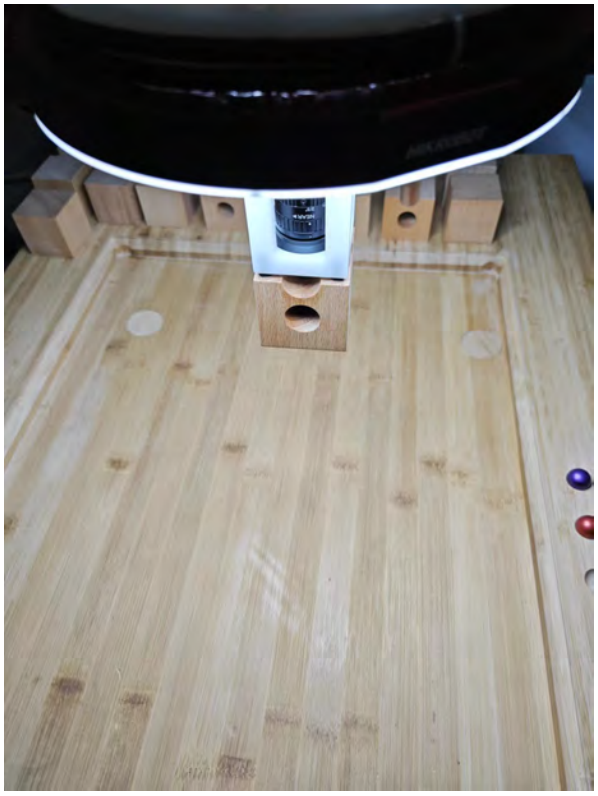


图 10.41: 移动到预放置点

下面我们将积木进行精确放置，首先，可以看到 4 个 IF 判断语句，如果当前的积

木是经过旋转后再放置的，由于法兰中心和工具中心并不在一个点，我们通过反复示教法兰中心与工具中心，从而得到在工具中心旋转时，坐标需要如何调整，此时减少误差，将积木搭建更加精准，之后再通过设置输出端将气泵关闭，电磁阀打开，即可放下积木，并抬高机械臂离开。

```
;Lay out
;REMOVE ERROR
IF B[i,j,k] == 180 THEN
   ERROR_X = 4.3
   ERROR_Y = 0
ELSE
   ERROR_X = 0
   ERROR_Y = 0
ENDIF
IF B[i,j,k] == 90 THEN
   ERROR_X = 2.4
   ERROR_Y = 2.4
ENDIF
IF B[i,j,k] == -90 THEN
   ERROR_X = 2.4
   ERROR_Y = -2.3
ENDIF
IF B[i,j,k] == 0 THEN
   ERROR_X = 0
   ERROR_Y = 0
ENDIF

MY1.X = Location_XStart + (k - 1) * 48 - ERROR_X
MY1.Y = Location_YStart - (j - 1) * 48 - ERROR_Y
MY1.Z = Location_ZStart + (i - 1) * 48 + 0.5
PTP MY1
;Exhaust
$OUT[12] = FALSE
$OUT[10] = FALSE
$OUT[9] =  FALSE
$OUT[11] = FALSE
$OUT[3] = TRUE
$OUT[4] = TRUE
$OUT[5] = TRUE
$OUT[6] = TRUE
WAIT SEC 0.1
```

图 10.42: 放置积木代码

图 10.43: 放置积木并离开

我们抓取积木完毕，准备离开，直接将机械臂向上升即可，此时如若我们是旋转了 180 度放置的积木，需要先讲积木往回旋转 90 度，再在起始抓积木位置再往回旋转 90 度从而将旋转调为零点，具体原因涉及机器人欧拉角的判断，本书不展开讨论。由于电磁阀不能长时间处于开启状态，我们移至上升点后，将四个电磁阀关闭。

```
;GO OUT
MY1.Z = MY1.Z + 160
IF B[i,j,k] == 180 THEN
   MY1.A = -90
   MY1.B = 0
   MY1.C = -180
   PTP MY1
ENDIF
PTP MY1
$OUT[3] = FALSE
$OUT[4] = FALSE
$OUT[5] = FALSE
$OUT[6] = FALSE
CONTINUE
ENDIF
```

图 10.44: 将机械手抬起高处完成一次抓取代码

在抓取完毕该积木后，如果该矩阵还存在非 0 值，即还有积木未搭建，我们将进行下一块积木的搭建，具体流程相同，仅仅更换了种类。

图 10.45: 继续搭建积木

10.7.3 将小球放置顶部积木中心滚下

在我们搭建完成后，积木移动到最后一块积木上方，如图10.46所示，此时我们要开始抓取小球并按顺序放入轨道中。

图 10.46: 完成最后一块积木搭建

如图10.47左图所示，由于弹珠基本是放入最后一块积木的中央，从而在此时记录最后一块积木中央的坐标，首先先将机械手往上抬起，防止碰撞，再接着我们运动到四颗弹珠上方，再垂直向下运动，将四个气孔贴到小球后停止，此时打开气泵，等待一会，再将机械手抬高，移动至最后一块积木上方。

```
Ball_EndX = MY1.X + 19
Ball_EndY = MY1.Y - 18
Ball_EndZ = MY1.Z - 160
MY1.Z = MY1.Z + 2 * 48
MY1.A = 0
MY1.B = 180
MY1.C = 0
PTP MY1
;Put the Ball drop out
MY1.X = Ball_StartX
MY1.Y = Ball_StartY
MY1.Z = Ball_StartZ + 48 * 4
PTP MY1
MY1.Z = Ball_StartZ
PTP MY1
$OUT[12] = TRUE
$OUT[10] = TRUE
$OUT[9] =  TRUE
$OUT[11] = TRUE
WAIT SEC 1
MY1.Z = MY1.Z + 4 * 48
PTP MY1
MY1.X = Ball_EndX
MY1.Y = Ball_EndY
MY1.Z = Ball_EndZ + 48
PTP MY1
MY1.Z = MY1.Z - 20-6
PTP MY1
$OUT[11] = FALSE
$OUT[5] = TRUE
WAIT SEC 0.5
MY1.X = MY1.X - 2.4
MY1.Y = MY1.Y + 2.4
MY1.A = -90
MY1.B = 0
MY1.C = -180
PTP MY1
$OUT[12] = FALSE
$OUT[3] = TRUE
WAIT SEC 0.5
MY1.X = MY1.X - 1.3
MY1.Y = MY1.Y - 2
MY1.A = 0
MY1.B = 0
MY1.C = -180
PTP MY1
$OUT[10]= FALSE
$OUT[4] = TRUE
WAIT SEC 0.5
MY1.X = MY1.X + 4
MY1.Y = MY1.Y - 3.4
MY1.A = 90
MY1.B = 0
MY1.C = 180
PTP MY1
$OUT[9] = FALSE
$OUT[6] = TRUE
WAIT SEC 0.5
```

图 10.47: 吸取与放下弹珠代码

图 **10.48:** 吸取小球 1

图 **10.49:** 吸取小球 2

如图10.47右图所示，下面我们要开始将弹珠放入积木中，我们采取的策略是将工具进行旋转，从而气泵的小球可以一颗一颗掉下，由于工具与法兰中心在一条线上，我们需要作偏差调整，每次小球放下将该气管气泵关闭电磁阀打开，之后记得及时将它们关闭。

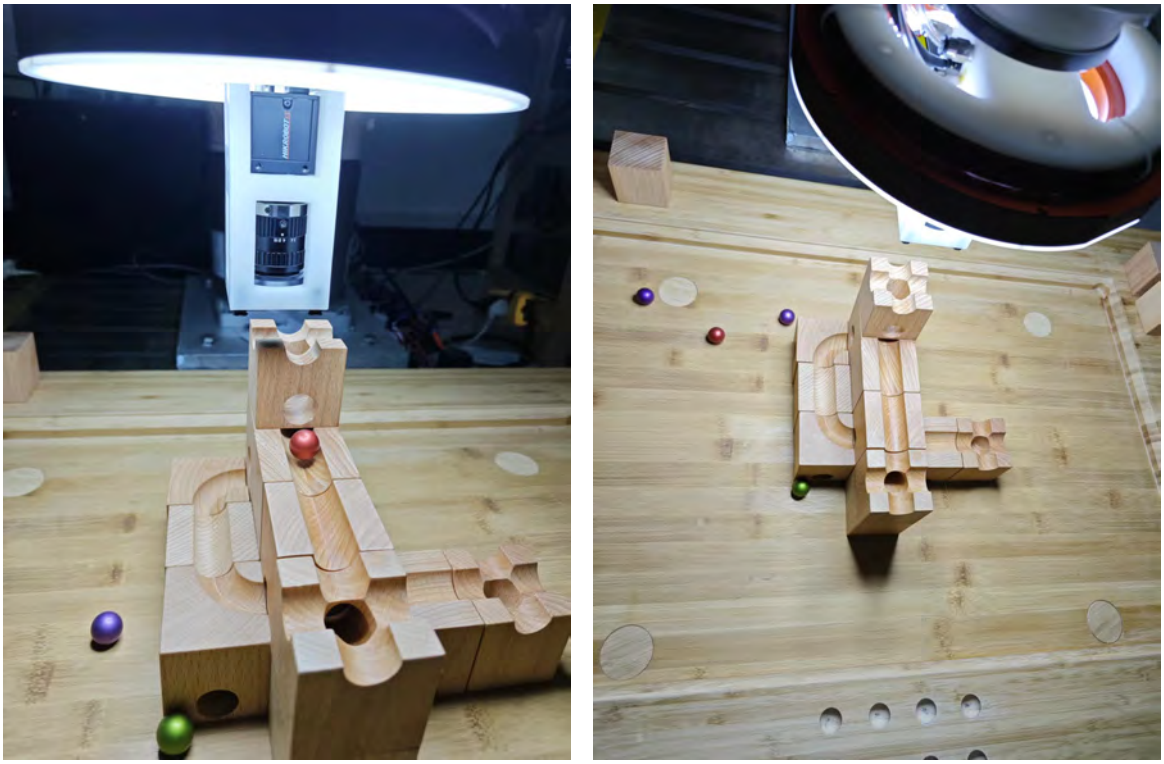


图 10.50: 弹珠正在运动以及弹珠全部掉下

10.7.4 将积木放回原位

将积木放回代码与抓取代码几乎相同，首先运行到积木上方，然后识别，识别到直接下去抓取，之后将其放入指定位置即可。其中要注意一点为在抬起积木，需要从一个角度抬起，因为你必须确保你的积木抬起不能影响到其余积木。

图 10.51: 吸取积木

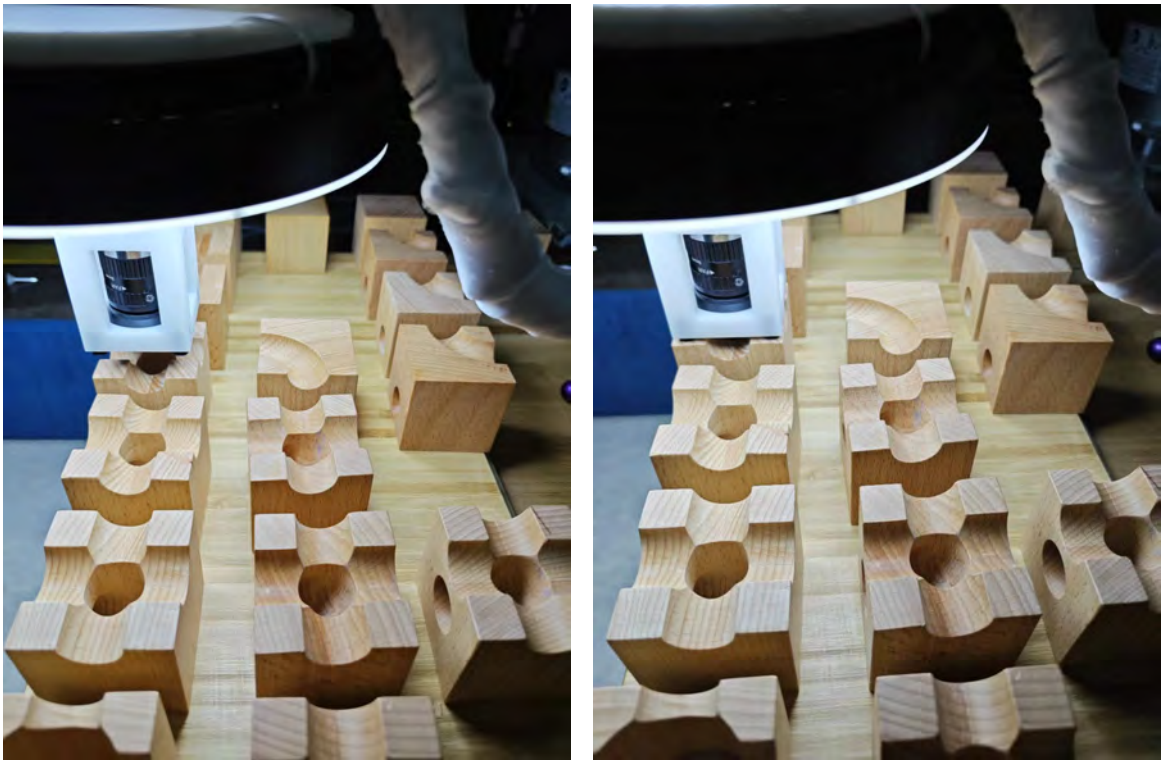


图 10.52: 将积木放到指定位置

全部放回后，如图所示

图 10.53: 将积木全部放回原位

10.7.5 将弹珠全部捡起

将弹珠捡起，具体的流程是，我们要先将整个机械臂进行抬高，使得相机视野足够看到全部的 4 颗弹珠，如图所示，左侧为机械手位置，右侧上位机显示为当前相机所看视野。

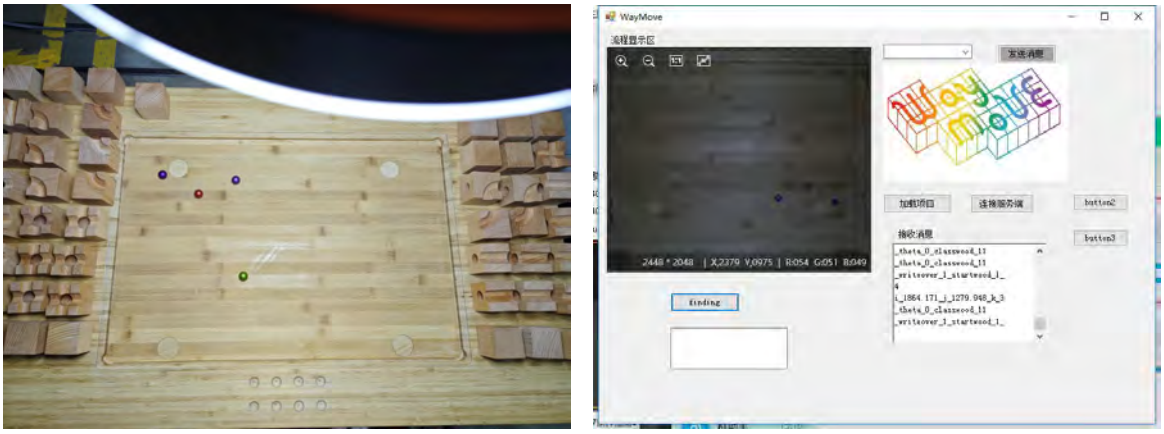


图 10.54: 将机械臂移动到最上方

如下图所示，左侧是将机械手移至上方代码，我们可以观察到其中存在一行设置曝光为 7000，这是因为我们之前的曝光值比较低，它在下面识别积木可以，而到了上方识别小球就变得困难，因此提高曝光是最好的办法。右侧我们通过与上位机建立通信，上位机将获得到的四颗弹珠的像素点坐标传给我们，我们再通过计算可以得到弹珠的位置，从而实现积木抓取。

```
RET=EKI_ClearBuffer("XmlTransmit","Sensor")
$FLAG[1] = FALSE
Ball_HX = 580.13
Ball_HY = -54.27
Ball_HZ = 671.75
STU.X = Ball_HX
STU.Y = Ball_HY
STU.Z = Ball_HZ
STU.A = 0
STU.B = 180
STU.C = 0
PTP STU
number1=0
NUM_Ball = 0
ball_get_i=BallRest + 1
   RET=EKI_ClearBuffer("XmlTransmit","Sensor")
   $FLAG[1] = FALSE
;Scratch Ball
i=1
$OUT[8] = TRUE
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@e",7000)
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",2100)
WAIT SEC 0.3
```

```
RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",98765111)
FOR i = 1 TO 4 STEP 1
   BallContinue:
   IF i > 4 THEN
     EXIT
    ENDIF
   IF Ball_Stu[i] == 1 THEN
      i = i + 1
      GOTO BallContinue
   ENDIF
   WAIT FOR  ( $FLAG[1] )
   number=1
   RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
   RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
   RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
   RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
   RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
   RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
   number=2
   ball_get_x[i]=DA1
   ball_get_y[i]=DA2
   RET=EKI_ClearBuffer("XmlTransmit","Sensor")
   $FLAG[1] = FALSE
ENDFOR
```

图 10.55: 左侧为将机械手移至上方代码，右侧为与上位机通信获取小球位置

得到坐标后，为了确保万无一失，我们先后 4 次运动到每个弹珠的位置，再次进行检测，我们在从上往下第一次运动时，为了放置触碰到两侧积木，我们使用 CIRC 语句将机械手运动到下面，后面 3 次运动小球位置即可使用 PTP 语句，从而使用一个 new 变量来初始化，如下图所示。

```
new = 1
FOR i=1 TO 4 STEP 1
   Ball2:
   IF i > 4 THEN
     EXIT
    ENDIF
   IF Ball_Stu[i] == 1 THEN
      i = i + 1
      GOTO Ball2
   ENDIF
   WAIT SEC 0.01
   Ball_locX[i]=Ball_HX + scale_h*(-ball_get_y[i] + 1024)
   Ball_locY[i]=Ball_HY + scale_h*(-ball_get_x[i] + 1224)
   STU.X = Ball_locX[i]
   STU.Y = Ball_locY[i]
   STU.Z=z_low
   STU.A=180
   STU.B=0
   STU.C=-180
```

```
IF new == 1 THEN
   IF Ball_locY[i] >= 180 THEN

      STU1.X = Ball_locX[i]
      STU1.Y = Ball_locY[i] - 10
      STU1.Z = z_low + 20
      CIRC STU1,STU
   ENDIF
   IF Ball_locY[i] <= -300 THEN
      STU1.X = Ball_locX[i]
      STU1.Y = Ball_locY[i] + 10
      STU1.Z = z_low + 20
      CIRC STU1,STU
   ENDIF
   ELSE
   IF STU.Y >=230 THEN
      STU.Y = Ball_locY[i] - 25
      PTP STU
   ENDIF
   IF STU.Y <= -315 THEN
      STU.Y = Ball_locY[i] + 25
      PTP STU
   ENDIF
ENDIF
STU.Y = Ball_locY[i]
PTP STU
```

图 10.56: 运动到各个小球代码

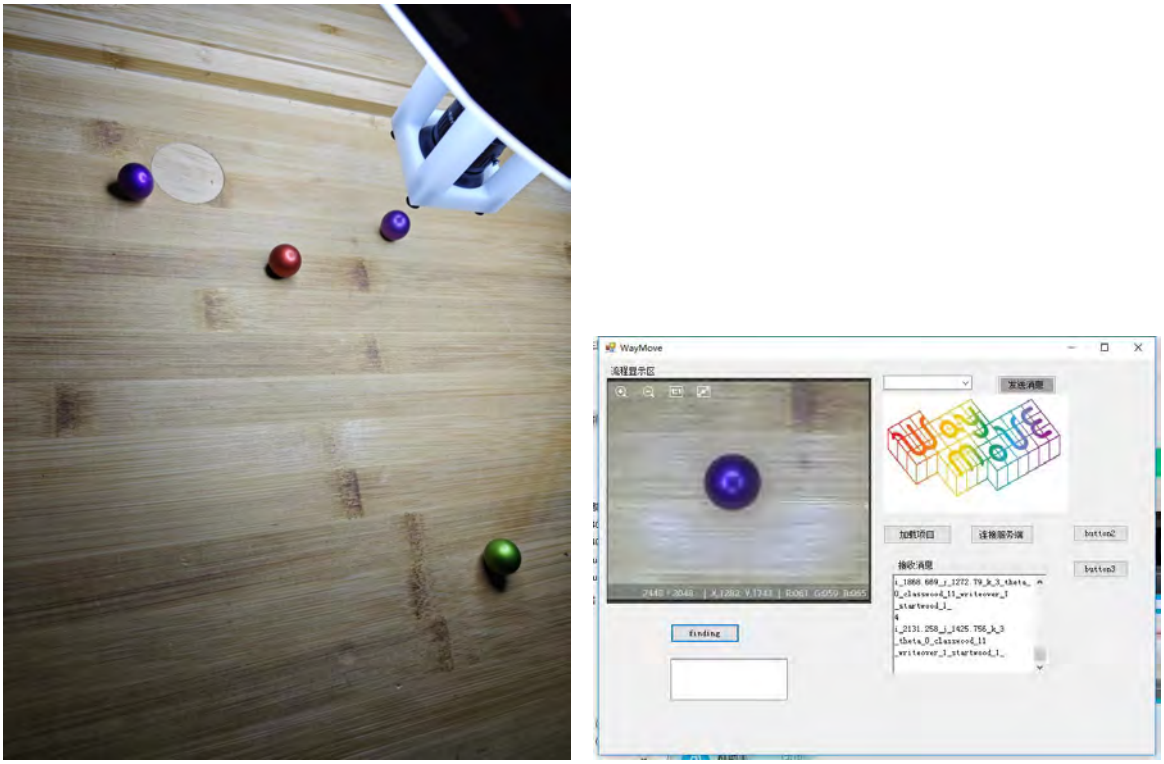


图 10.57: 运动到紫色小球上方

接下来我们还是与之前抓积木一样，识别小球位置，但是此次仅保留小球坐标在变量中，我们不作抓取，等到 4 颗小球坐标均保存下来，我们再进行每颗小球的抓取。

```
WHILE TRUE
   RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",1300)
   RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",98765111)
   WAIT FOR  ( $FLAG[1] )
   RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
   number=1
   RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
   RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
   RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
   RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
   RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
   RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
   WAIT SEC wait_sec
   number=2
   $FLAG[1] = FALSE
   RET=EKI_ClearBuffer("XmlTransmit","Sensor")
      WAIT SEC 0.01
      Ball_X[i]=$POS_ACT.X+scale_low*(-DA2 + 1024)
      Ball_Y[i]=$POS_ACT.Y+scale_low*(-DA1 + 1224)
   RET=EKI_ClearBuffer("XmlTransmit","Sensor")
   $FLAG[1] = FALSE
   WAIT SEC 0.01
   EXIT
ENDWHILE
```

图 10.58: 识别每颗弹珠位置并计算

那么接下来就是使用之前记录的弹珠坐标进行抓取，抓取过程保证每颗气孔分别吸取一颗弹珠，因此我们对每个气孔进行编号。由于我们在抓取模板四周存在边缘高处的部分，我们必须保证我们的机械手在抓取最下方积木时，使用最下方的气孔抓取，

否则将会造成后方工具的碰撞。因此我们需要计算，当小球处于木板边缘时，需要使用特定的气孔去抓取。如图所示，举例当抓取位于 Y 轴最高侧的第 2 颗弹珠使用 4 号气孔孔，此时需要将工具通过旋转 90 度来获得当前气孔，再进行抓取，而如果计算所得此时下落位置会触碰边缘木板，我们则不作抓取动作。

```
4563        WHILE TRUE
4564           RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",1300)
4565           RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@n",98765111)
4566           WAIT FOR  ( $FLAG[1] )
4567           RET=EKI_SetReal("XmlTransmit","Robot/Data/mypos/@f",0)
4568           number=1
4569           RET=EKI_GetReal("XmlTransmit","Sensor/X1",DA1)
4570           RET=EKI_GetReal("XmlTransmit","Sensor/Y1",DA2)
4571           RET=EKI_GetReal("XmlTransmit","Sensor/Z1",DA3)
4572           RET=EKI_GetReal("XmlTransmit","Sensor/A1",DA4)
4573           RET=EKI_GetReal("XmlTransmit","Sensor/B1",DA5)
4574           RET=EKI_GetReal("XmlTransmit","Sensor/C1",DA6)
4575           WAIT SEC wait_sec
4576           number=2
4577           $FLAG[1] = FALSE
4578           RET=EKI_ClearBuffer("XmlTransmit","Sensor")
4579              WAIT SEC 0.01
4580              Ball_X[i]=$POS_ACT.X+scale_low*(-DA2 + 1024)
4581              Ball_Y[i]=$POS_ACT.Y+scale_low*(-DA1 + 1224)
4582           RET=EKI_ClearBuffer("XmlTransmit","Sensor")
4583           $FLAG[1] = FALSE
4584           WAIT SEC 0.01
4585           EXIT
4586        ENDWHILE
```

图 10.59: 举例当抓取位于 Y 轴最高侧的第 2 颗弹珠使用 4 号气孔

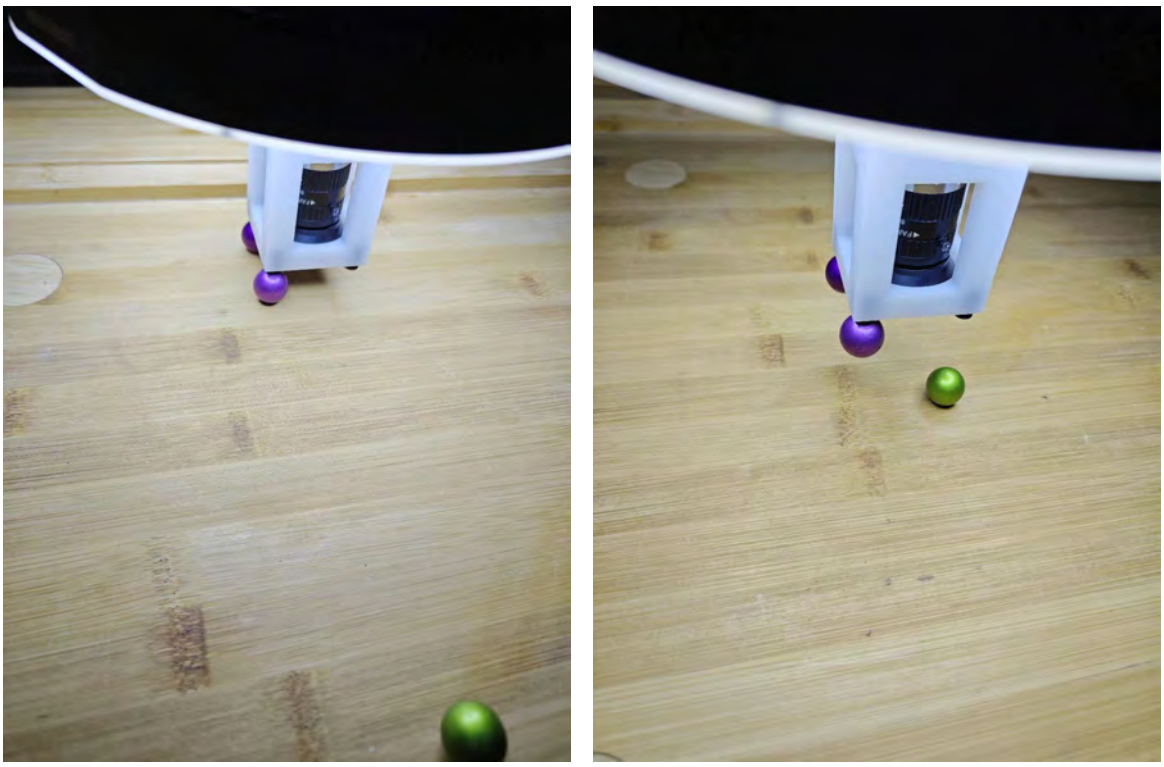


图 10.60: 挨个抓取小球

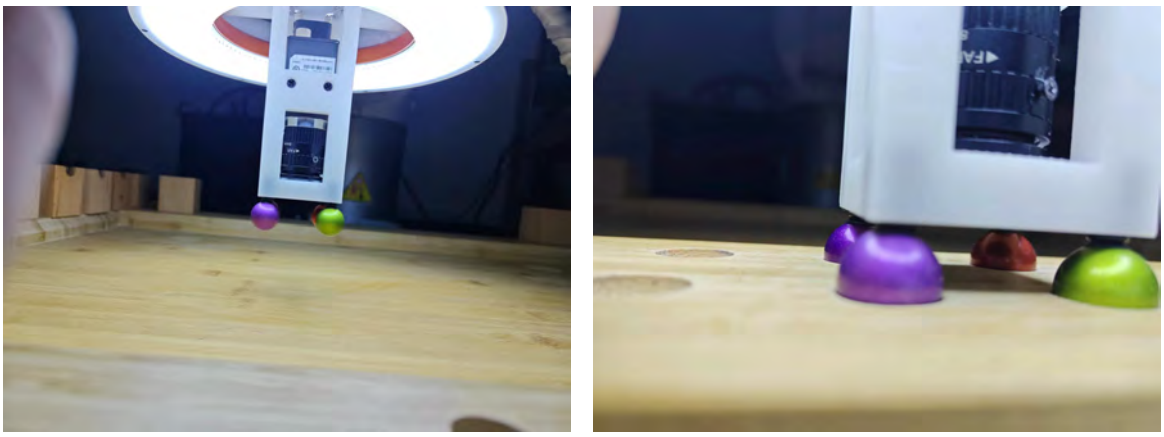


图 10.61: 成功抓取 4 颗小球并放回

10.8 上位机界面的制作

我们可以看到在上述 KUKA 代码中，存在一些我们所制作的上位机界面图，该上位机界面由 C# 进行编程，在本节主要向大家介绍这款上位机软件如何制作。

10.8.1 VisionMaster SDK 介绍

1.VM 算法平台和 VM SDK 的区别 VM 算法平台是海康机器人自主开发的机器视觉软件，致力于为客户提供快速搭建视觉应用、解决视觉检测难题的算法工具，能满足视觉定位、尺寸测量、缺陷检测以及信息识别等机器视觉应用。

VM SDK 是基于 VM 算法平台中的方案、流程和模块工具等进行对象级封装的二次开发工具包，用户可通过 VM SDK 提供的方案、流程和模块工具中各个对象的功能接口进行相应的数据交互与运行控制。同时提供相应的流程配置控件、参数配置控件、渲染控件、全局工具控件以及前端运行界面控件等，方便进行方案编辑、参数配置以及渲染展示，便于灵活进行开发，并扩展机器视觉应用。VM SDK 在安装 VM 软件后会同步安装在安装路径下的 Development 文件夹下，并提供了开发文档及丰富的二次开发示例。

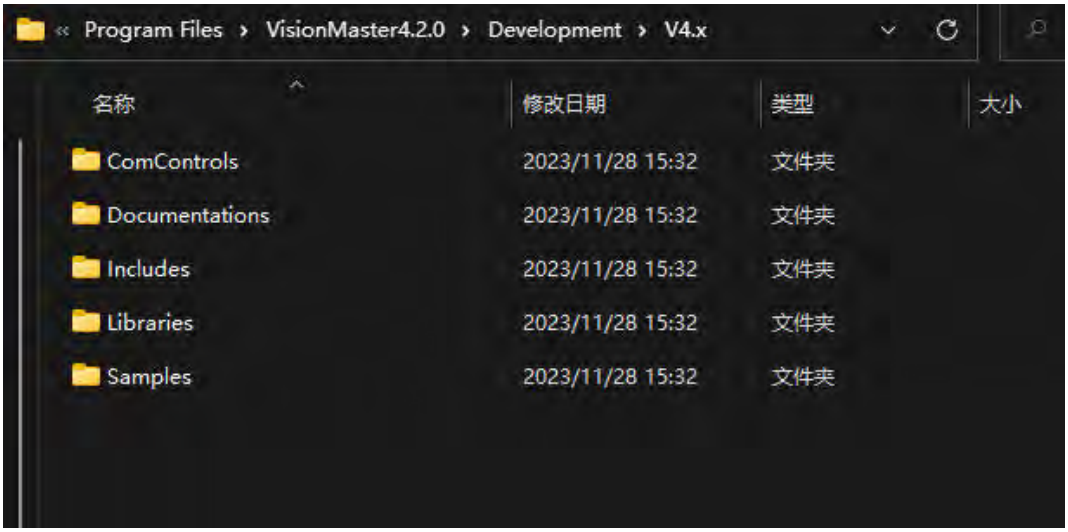


图 **10.62:** 开发组件

2. 使用 SDK 所需配置下面是我们开发所使用的配置要求，我们使用的是海康的专用视觉控制器进行开发，同时还需要拥有加密狗才可以使用。

运行环境（推荐配置）

操作系统:	Windows 7/10（64位中文操作系统）
CPU:	Intel Core i7-6700 3.4GHz 或以上（如需使用CPU相关深度学习功能，建议配置为i9-10900K及以上）
内存:	8 GB及以上
显卡:	显存1G以上显卡，GPU相关深度学习功能需要显存6GB及以上
.NET运行环境:	.NET4.6.1及以上
网卡:	Intel I210系列以上性能网卡
USB接口:	USB3.0的接口

运行环境（最低配置）

操作系统:	Windows 7/10（64位中英文操作系统）
CPU:	Intel Pentinum IV 2.0Ghz及以上
内存:	4 GB
显卡:	显存1G以上显卡，GPU相关深度学习功能需要显存6GB及以上
.NET运行环境:	.NET4.6.1及以上
网卡:	千兆网卡
USB接口:	USB3.0的接口

图 **10.63:** 运行环境 (推荐配置)

3. 新建工程此外，我们 C# 开发平台选择 VisualStudio2022，推荐 VS2017 新建工程 (要求 VS2013 及以上)，读者可在网上查找安装教程。同时，还需要安装 C# 开发框架（如图10.64所示），这里我们将生成框架选择.NET Framework 4.6.1 及以上。

图 **10.64:** 安装 C# 开发环境

此时我们要打开项目属性，在生成页面里将“首选 32 位”去掉勾选。一般情况下新建工程默认是勾选的，需要把勾选取消掉，因为 VM 里部分算子模块只支持 64 位的。

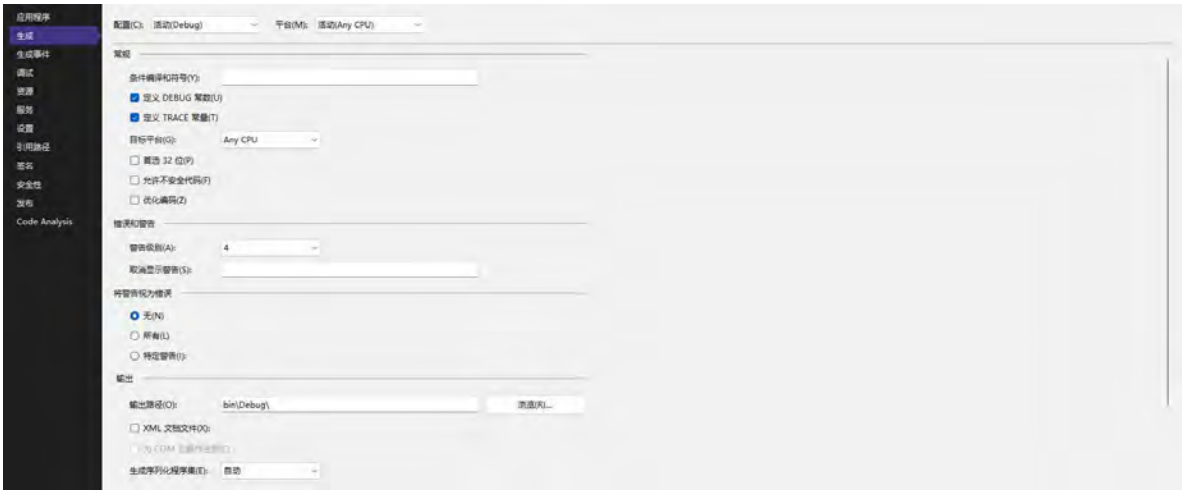


图 **10.65:** 对项目进行配置

VM SDK 的算子工具没有封装成一个整体的库，而是单独分开封装的，这样虽然引用较为繁琐，但是能更加灵活地控制工具的引用，不失为一种优雅的解决方案。同时 VM SDK4.2 引入了全局程序集缓存 (global assembly cache, GAC 机制)，作用是存放一些有很多程序要用到的公共程序集，如 System.Data、System.Windows.Forms 等，这样，很多程序就可以从 GAC 里获得程序集，而不需要再把所有要用到的程序集都拷贝到应用程序的执行目录下，减少了冗余，方便了开发使用。下面介绍一下如何添加算子工具的引用。使用 VM SDK 提供的引用批导入工具，工具路径为

VisionMaster4.2.0/Development/V4.x/ComControls/Tool/ImportRef.exe。首先需要选择新建工程的二级目录（工程配置文件的路径），然后选择需要导入的工具进行批量导入，具体操作步骤参考下图。

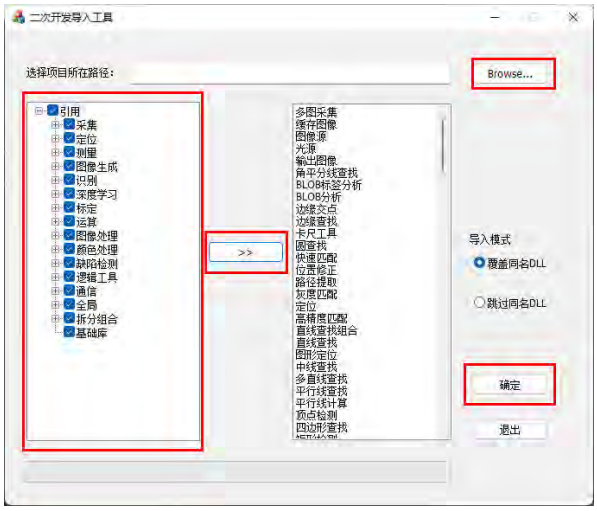


图 10.66: 将 SDK 提供工具导入项目

添加控件:

VM SDK 提供了丰富的控件供二次开发者使用，开发者可以使用这些控件实现更加灵活的界面布局设计，控件添加方法如下：

打开 VS 工具箱，[所有 Windows 窗体] 鼠标右击，再点击 [选择项]，此时弹出页面中选择“浏览”，浏览 VM4.2 的安装路径文件夹：VisionMaster4.2.0/Development/V4.x/ComControls/ssembly 选择 VMControls.Winform.Release.dll，添加后工具箱出现 VM 相关控件。

图 **10.67:** 在工具箱右键“选择项”

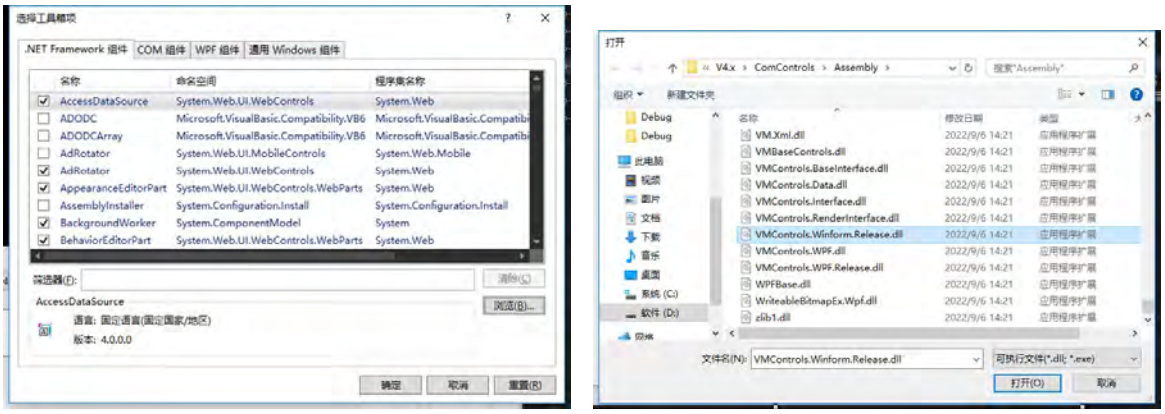


图 **10.68:** 在选择工具箱里，点击浏览，找到我们的文件

此时，我们搜索 VM，将出现二次开发所需的控件。

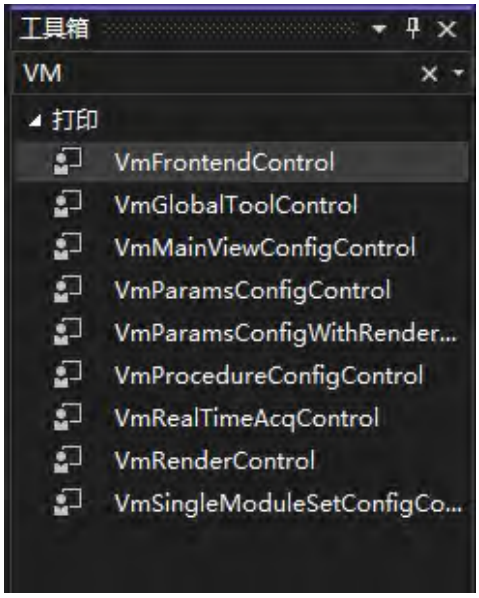


图 10.69: 搜索 VM，出现 VM 控件

工具箱中的控件如下所示:

VmFrontendControl 前端运行界面控件

VmGlobalToolControl 全局模块控件

VmMainViewConfigControl 主界面控件

VmParams 参数配置控件

VmParamsConfigWithRender 参数配置带渲染控件

VmProcedureConfigControl 流程配置控件

VmRealTimeAcqControl 实时取流控件 (V4.2 新增)

VmRenderControl 渲染控件

VmSingleModuleSet ConfigControl 独立 Group 控件 (V4.2 新增)

10.8.2 开发过程

方案操作过程

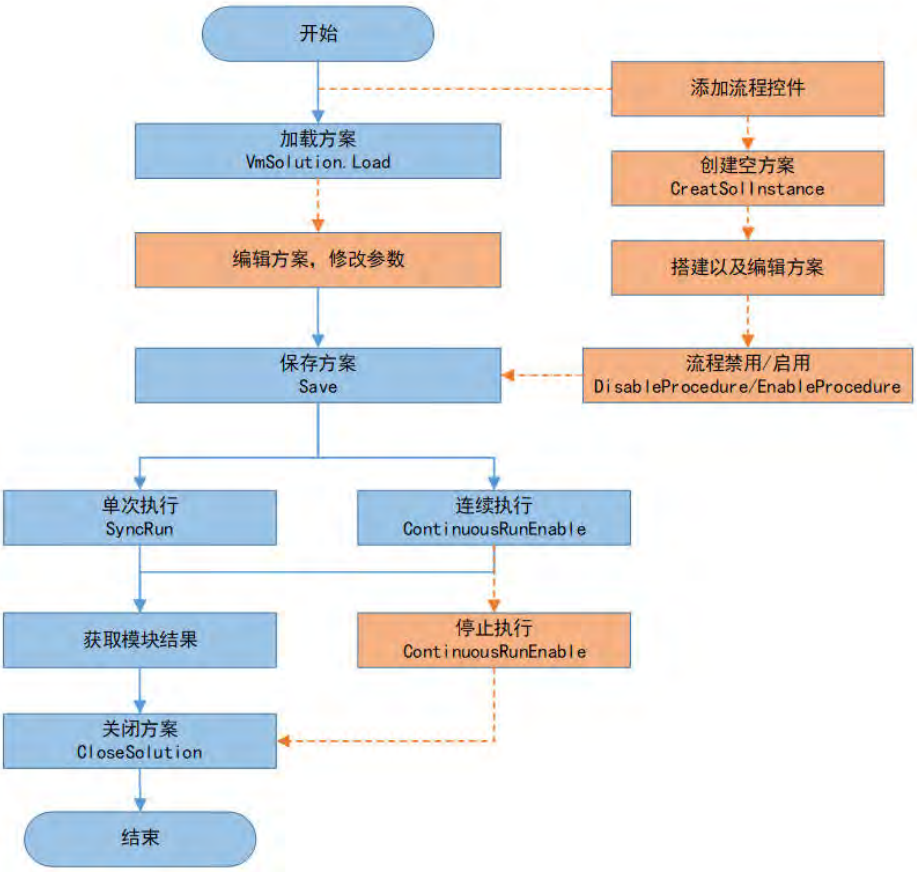


图 10.70: 方案操作流程

首先我们要将我们所做好的 VM 项目放到 C# 目录下，如图10.71所示。

图 10.71: 将 sol 文件放入 C# 目录下

下面，我们在界面上放置一个按钮，只要点击此按钮即可加载当前目录中的项目，为了使得我们加载项目的同时可以同时处理其余事项，我们在这里新建了一个启动项目的线程，在点击按钮后，我们在新的线程中加载项目，同时使用加载项目中的 syncrun() 异步方法，使得程序执行效率更高。

```
1 个引用
private void button1_Click(object sender, EventArgs e)
{
    // 如果找到.sol文件，则显示它们的路径
    Thread thread1 = new Thread(Vmload);

    if (solFiles.Length > 0)
    {
        thread1.Start();

    }
    else
    {
        ShowMsg("Can't find any File!");
    }

}
1 个引用
private void Vmload()
{
    VmSolution.Load(string.Join("/", solFiles));
    VmSolution.Instance.SyncRun();
    VmSolution.Instance.ContinuousRunEnable = false;
    ShowMsg(string.Join("",solFiles));
}
```

图 10.72: 加载项目代码

在加载项目后，我们需要进行上位机与我们 Vision Master 的连接，我们这里选择使用 TCP 进行通信，这样可以保证通信的可靠性，其中要注意要将我们的 IP 地址和端口号设置为与 Vision Master 相同。

```
//设为全局变量用于通讯连接和接收数据
Socket socketsend;
1 个引用
private async void button5_Click(object sender, EventArgs e)
{
    try
    {
        //创建用于通讯的socket
        socketsend = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
        IPEndPoint IPEPoint = new IPEndPoint(IPAddress.Parse("192.168.1.115"), Convert.ToInt32(59151));
        ShowMsg(Convert.ToString(IPEPoint));
        //连接对应的端口
        await socketsend.ConnectAsync(IPEPoint);
        ShowMsg(socketsend.RemoteEndPoint + ": 连接成功!");
    }
    catch { }
}
```

图 10.73: TCP 通信代码

项目和通信连接成功以后，我们此时就可以将我们事先做好的积木摆放顺序放上去了，需要提前设置哪一行哪一列摆放如何哪种类型的积木，从而将这些信息逐行发送给 Vision Master，Vision Master 再将这些信息传给机械臂即可。在下面的代码中，我们设置 3 个可以切换的积木图片以及积木编码，积木编码通过 txt 文本格式将其存储到与 Vision Master 同级的目录下，通过选择不同的选项，切换我们想搭的积木。

```
1 个引用
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    //string currentFilePath = Directory.GetCurrentDirectory();
    Bitmap Pic_Model_3 = new Bitmap(currentFilePath + "\\model_Pic\\Model_3.png");
    Bitmap Pic_Model_5 = new Bitmap(currentFilePath + "\\model_Pic\\Model_5.png");
    Bitmap Pic_Model_11 = new Bitmap(currentFilePath + "\\model_Pic\\Model_11.png");
    switch (comboBox1.SelectedItem.ToString())
    {
        case "Model3":
            pictureBox1.Image = Pic_Model_3;
            txt_Path = "Model3.txt";
            break;
        case "Model5":
            pictureBox1.Image = Pic_Model_5;
            txt_Path = "Model5.txt";
            break;
        case "Model11":
            pictureBox1.Image = Pic_Model_11;
            txt_Path = "Model11.txt";
            break;
```

图 10.74: 设置积木摆放次序代码

下面就是我们如何将我们设置好的积木次序，通过 TCP 通信将其发送到 Vision Master，首先将我们上面已经有的文本路径放进去，之后逐行将我们的消息发送出去即可，在这里要注意我们发送消息要有一些间隔时间，保证每条消息都可以完整地被对方接受到。

```
1 个引用
private async void button6_Click(object sender, EventArgs e)
{
    string filePath = Directory.GetCurrentDirectory() + '\\' + txt_Path;
    List<string> lines = new List<string>();
    using (StreamReader reader = new StreamReader(filePath))
    {
        string line;
        while ((line = await reader.ReadLineAsync()) != null)
        {
            lines.Add(line);
        }
    }
    foreach (string line in lines)
    {
        Byte[] buffer = System.Text.Encoding.UTF8.GetBytes(line);
        await socketsend.SendAsync(new ArraySegment<byte>(buffer), SocketFlags.None);
        ShowMsg(line);
        await Task.Delay(130);
    }
}
```

图 10.75: 发送积木编码与 VisionMaster 代码

接下来就是我们的抓小球代码了，其中仅在高空首次观察四个小球位置需要使用上位机作辅助作用，其余并不需要，然而代码却没有像的那样简单，首先我们要接受曝光值 e，以及全局向量 q 的值，曝光值即为相机当前的曝光数值，全局向量 q 记录了此时已经抓去了几颗小球。通过 Vision Master 的拓展方法我们均可以很容易访问得到。同时，为了放置识别数量错误，我们需要将此时 Vision Master 所识别的小球数量也要拿到，在下面的判断条件中，只有曝光值大于 6000 以及剩余小球数量等于识别小球数量时，才可以进入下方小球抓取程序中。在这里我们将识别到的信息拿到，从而将这些信息逐次发送给上位机去，可以发现其中存在一些等待的语句，这些用来防止消息通信太快，而对方没有接收到。

```
WindowsFormsApp1    WindowsFormsApp1.WayMove    scratchBall()
        1 个引用
        private async void scratchBall()
        {
            while (true)
            {
                var findBall = (IMVSFastFeatureMatchModuTool)VmSolution.Instance["流程1.快速匹配5"];
                GlobalVariableModuleTool globalVar = (GlobalVariableModuleTool)VmSolution.Instance["全局变量1"];
                string exposure = globalVar.GetGlobalVar("e");
                string q = globalVar.GetGlobalVar("q");
                int qInt = (int)float.Parse(q);
                float exposureFloat = float.Parse(exposure);
                int number = findBall.ModuResult.MatchNum;
                if (exposureFloat >= 6000 && number == 4 - qInt)
                {
                    await Task.Delay(300);
                    var findBall1 = (IMVSFastFeatureMatchModuTool)VmSolution.Instance["流程1.快速匹配5"];
                    ShowMsg($"检测数量{number}");
                    var point = findBall1.ModuResult.MatchPoint;
                    await Task.Delay(100);
                    for (int i = 0; i < point.Count; i++)
                    {
                    ShowMsg(point.Count.ToString());
                    string str_2 = $"i_{point[i].X.ToString()}_j_{point[i].Y.ToString()}_k_3_theta_0_classwood_1
                    Byte[] buffer_send = System.Text.Encoding.UTF8.GetBytes(str_2);
                    await socketsend.SendAsync(new ArraySegment<byte>(buffer_send), SocketFlags.None);
                    ShowMsg(str_2);
                    await Task.Delay(150);
                }
                    await Task.Delay(200);
                }
            }
        }
```

图 10.76: 抓小球代码

到此为止，我们的 C# 开发上位机的代码已经讲解完毕了，具体要遇到哪些困难，还是因人而异，但是我想告诉各位的是，办法总比困难多，你要相信，你在这个阶段所遇到的任何问题，都是已经被前人遇到过很多次的问题，都是可以解决的。