



省创 SRTP 项目研究报告

基于嵌入式系统与深度学习的 3D 感知、抓取与装配机器人

秦雨扬 潘力豪 盛晨璐 刘山 *

摘要:第一代基础机械臂采用传统机器视觉方法进行俄罗斯方块的分割、识别与定位，使用 PD 算法进行方案规划，并使用大象机器人完成操作。视觉部分使用高斯模糊与 Canny 边缘提取进行膨胀，然后根据俄罗斯方块的面积一致性生成候选联通块。方案规划部分使用改进的算法对俄罗斯方块放置的位置进行规划，考虑了放置后产生的“空洞”数目等因素。机器人操作采用棋盘格进行相机标定，并支持多批放入方块。第二代智能机械臂在第一代基础上做出改进，采用深度学习算法处理视觉传感器的输入，改进了 YOLOv3 算法，使其能够进行俄罗斯方块的位姿估计；引入手眼标定和双阶段机械臂伺服控制方案，提高了抓取的准确性；引入 ROS 框架，建立了视觉、图像处理和控制节点，提高了系统的实时性。第三代解决方案针对前两代重难点展开研发。在俄罗斯方块的位姿估计与识别中采用了 MASK R-CNN 进行识别与分割，再结合最小旋转矩阵算法得到倾角。基于 Semgent-anything Model 开发快捷分割标注工具。在机器人控制方案上，采用正逆运动学解算，基于雅克比迭代求解逆运动学算法设计了逆运动学求解器，提高了速度与加速度的利用率，更为精准、鲁棒和高效。

浙江大学
控制科学与工程学院

2024 年 5 月 15 日



Contents

1. 背景介绍	3
2. 过程与内容	3
2.1 第一代基础机械臂	3
2.2 第二代智能机械臂	3
2.3 第三代精准、鲁棒、高效的解决方案	4
3. 原理与方法	4
3.1 俄罗斯方块识别与位姿检测	4
3.1.1 基于传统机器视觉的俄罗斯方块分割	5
3.1.2 基于最小旋转矩阵算法的俄罗斯方块位姿估计与类别分析	6
3.1.3 基于 YOLO 的俄罗斯方块类别、位置与姿态检测	6
3.1.4 基于 Segment Anything 的数据标注	7
3.1.5 基于 Mask R-CNN 的俄罗斯方块智能分割方法	8
3.1.6 基于 DeepSORT 的俄罗斯方块跟踪策略	9
3.2 坐标系转换与基于棋盘格的标定	9
3.2.1 基于 numpy 的实现	9
3.2.2 基于 POINT-POINT Iterative Closest Point 的实现	9
3.3 机械臂位姿解算与运动规划	10
3.3.1 基于雅克比迭代的 IK Solver	10
3.4 俄罗斯方块方案生成	11
3.4.1 已知所有块情况的生成方案	11
3.4.2 仅知道当前盘面情况下的在线生成方案	11
4. 结果与展示	12
4.1 俄罗斯方块识别与位置与姿态检测	12
4.1.1 基于传统机器视觉的俄罗斯方块分割、位姿估计与类别分析	12
4.1.2 基于 YOLO 的俄罗斯方块类别、位置与姿态检测	12
4.1.3 基于 Segment Anything 的数据标注	13
4.1.4 基于 Mask R-CNN 的俄罗斯方块智能分割方法	13
4.2 机械臂位姿解算与运动规划	13
4.2.1 基于雅克比迭代的 IK Solver	14
4.3 俄罗斯方块方案生成	16
4.3.1 已知所有块情况的生成方案	16
4.3.2 仅知当前盘面情况下的在线生成方案	16
5. 总结与展望	16



1. 背景介绍

机械臂在汽车、电子、食品、制药等工业生产中得到广泛应用，可以完成重复性高、危险性大、难度大的工作，提高生产效率和质量，降低人力成本和风险。目前应用较多的是自动化机械臂，它通常用于单一、重复性较高的工业制造任务，如装配、焊接、喷涂等，其控制方式一般为预先编写好的程序或者基于传感器反馈的简单反馈控制系统，且通常不具备感知和决策的能力，需要人工设定和监控其工作流程。物联网技术、人工智能的蓬勃发展，让我们有新的工具去探索研发更加智能机械臂。而中国制造 2025 提出把智能制造作为推进工业化与信息化融合的主攻方向，智能的机械臂是其中的重要一环。

装配机器人在投入实际应用过程后，其操作的精确度、装配过程的自主化、工作流程的安全性都是非常重要的环节，因此，对目标的识别、获取识别系统所提供的位置信息后将待装配的零件从当前位置移动到目标位置所使用的算法、以及机器人在装配过程中可能涉及到的决策都是较为重要的过程，同时，如何使机器人在有障碍的复杂工作环境中安全地完成装配任务也是值得关注。

刘山老师的课题“目标识别抓取与拼接机器人”结合智能机器人创意竞赛的项目，提出了俄罗斯方块的拼接任务。本项目以俄罗斯方块的拼接任务为中心，迭代式开发更加精准、鲁棒、高效的系统。

2. 过程与内容

2.1. 第一代基础机械臂

第一代俄罗斯方块机器人基于传统机器视觉方法进行俄罗斯方块分割、识别与定位，使用 PD 算法进行拼接方案规划，基于大象机器人平台完成操作。

在传统视觉中，本机器人使用高斯模糊与 canny 边缘提取，进行膨胀后得到众多联通块。本机器人基于俄罗斯方块的面积一致性，生成候选联通块实

现识别与分割。在分割的基础上，对每个方块，使用最小旋转矩阵算法得到倾角、中心点与长宽，基于此得到分类、抓取点与旋转角度。

在方案规划上，本机器人使用基于 Pierre Dellacherie 改进的算法 [1] 对俄罗斯方块放置的位置进行规划。该算法通过对放置后产生的“空洞”数目、能够消除的行数等因素计算每个可行位置的价值，并将当前的俄罗斯方块放置在价值最大的位置，以此达到实施规划的效果。规划部分从视觉部分获得当前俄罗斯方块的信息，生成目标位置和转角并反馈到控制模块，从而实现俄罗斯方块的拼装。

在机器人操作上，本机器人使用棋盘格进行相机标定，在放入方块后输入总数，自动识别所有放入方块后开始拼接，动态生成拼接方案，支持多批放入方块

2.2. 第二代智能机械臂

第二代俄罗斯方块机器人在第一版的基础上，使用深度学习算法对视觉传感器的输入进行处理，改进了目标检测领域性能良好的 YOLOv3[2] 算法，使网络在完成目标检测任务的同时，能够进行俄罗斯方块的位姿估计。Yolov3 算法是一种一阶段目标检测网络，由 Backbone, FPN, Yolo Head 三个部分组成，其中，Backbone 的部分由多个残差模块堆叠构成，残差模块之间间隔了一个 $\text{kernel_size}=3 \times 3$, $\text{stride}=2$ 卷积层，作用主要是对网络输入图像进行下采样。FPN 从 backbone 获取三个有效特征层后，进一步提取特征，进行特征融合。

我们加入了手眼标定，采用了基于全局相机和手眼相机的两阶段机械臂伺服控制方案。第一阶段使用深度网络处理全局相机所采集的图像，得到待抓取方块的抓取点像素坐标，使用相机内参、外参矩阵将像素坐标转换为机械臂坐标系下的二维坐标，使末端在水平面内移动到该抓取位置，进入第二阶段，使用视觉伺服方案控制末端精确对准抓取中心，从而完成方块抓取任务。

在方案规划上，我们继续采用第一代机器人的算法进行方案的规划。同时第二代机器人引入了先



进的 ROS[3] 框架，建立了视觉节点 (Vision Node)；图像处理节点 (ImageProcess Node) 以及控制节点 (Robot Node)，分别实现获取 realsense 相机的图像数据，对数据进行处理并计划抓取方案，以及对机器人控制柜下达具体抓取指令，实现了不同节点的协同工作，取得了比第一代机器人更好的实时性。

2.3. 第三代精准、鲁棒、高效的解决方案

在开发第三代俄罗斯方块机器人的过程中，实验室的大象机械臂被告知意外损坏，已返厂维修，直至本文完稿尚未寄回。故第三代解决方案未能在实物上进行系统联调，故此处称为解决方案。但此方案针对原有两个方案，精准解决了其中的痛点与难点，是我们后半年攻坚克难的集中体现。

在俄罗斯方块的位姿估计与识别中，第一代方案需要输入俄罗斯方块总数，且无法实时识别；第二代方案俄罗斯方块的旋转角度由 YOLO 输出，但精度较差。所以我们将两者相结合，通过 MASK R-CNN 进行俄罗斯方块的识别与分割，再运用方案一中使用的最小旋转矩阵算法得到倾角。这样子既克服了传统机器视觉方法中鲁棒性与实时性差的问题，又解决了方案二中旋转角度识别不准确问题。

而为了训练 MASK R-CNN 进行俄罗斯方块的识别与分割，我们采集了 100 张各个角度、高度拍摄的多样化的俄罗斯方块图像。为了更方便的标注数据，我们使用了 Meta AI 2023 年发布的 Segment-anything Model 作为辅助，基于 PyQt5 开发的标注界面，实现了人机交互式的高效数据标注。同时，对于 Mask R-CNN 的训练，我们尝试了不同的超参数，并尝试冻结 ResNet50 的 backbone，最终达到了 accuracy 和 Recall 均超过 98% 的效果。

此外，在视频的连续监测场景下，有时候会出现某一个俄罗斯方块没有检测出来的问题，影响跟踪精度。于是我们引入了 DeepSORT 跟踪器对俄罗斯方块进行跟踪，有效缓解了某帧漏检或误检带来的系统误判。

在机器人控制方案上，此前我们一直采用在欧式空间的直线规划，通过网络接口向大象机器人的控制传输目标点信息。为了避免奇异点的产生，我们规划了一个中间点。但这样一种控制方案没有充分利用每个关节的速度与加速度，不利于速度的提升。在学习“机器人建模与控制课程”后，我们为大象机器人列写 DH 表，进行正逆运动学解算，并在角度空间基于 Slerp 方法进行规划，取得了较好的效果。其中由于课程项目中的求解只适配其机器人参数，我们基于雅克比迭代求解逆运动学算法自行设计了逆运动学求解器，用于对大范围的移动过程采用关节角规划。由于机械臂的故障，我们通过 coppeliaSim 在仿真环境中对算法进行了验证，并在取得了不错的效果。

在方案规划上，我们改变了原有的固定方案，让机器人能够仅基于当前盘面情况以及目标块的类型，选择最优的放置位置。我们保留了 Pierre Del-lacherie 算法中对每个由放置位置和姿态组成的可行解计算价值函数的方法，并在此基础上修改了价值函数的参数以及权重分配。通过对目标位置的相邻块数量、放置后盘面内连通块的数量等因素进行考量，得到最优的放置位置和姿态，一方面能够保证对盘面的充分利用，另一方面，由于方案生成过程中尽量避免了目标装配位置两边都有限制的情况，极大程度上提高了装配的稳定性和安全性。

3. 原理与方法

3.1. 俄罗斯方块识别与位姿检测

俄罗斯方块的识别与位姿检测需要从复杂环境中识别俄罗斯方块，判断其类别，并得到其在 RGB-D 相机坐标系下的位置与旋转角度。



3.1.1. 基于传统机器视觉的俄罗斯方块分割

由于俄罗斯方块颜色单一，且放置在白色的桌子上（若不满足可以人为铺上桌布），非常适合使用边缘检测进行分割。

Canny 边缘检测器 [4] 是一种多阶段的边缘检测算法，其主要步骤包括高斯滤波、计算梯度、非极大值抑制和双阈值检测及边缘连接，其具体的算法流程如1所示。它能够有效地检测图像中的边缘，并抑制噪声和虚假边缘，从而获得精确的边缘信息。

其中第二步计算图像的梯度幅值和方向，可以采用 Roberts 交叉算子、Sobel 算子、Prewitt 算子等。本实验中采用了 Sobel 算子，它的 Sobel 算子的卷积核为：

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

但 Canny 边缘算子可能会存在断点，俄罗斯方块的边缘可能无法完全包围俄罗斯方块。俄罗斯方块内部与外部仍然联通。但基于有断点但是不多的条件，我们采用了膨胀算法，用 3×3 的核膨胀了三次，实现了对边缘断点的消除，俄罗斯方块内部与外部不再联通。

此时，我们进行联通块检测。由于俄罗斯方块的数量较多且不同类型的俄罗斯方块体积一致。我们假设有 N 个俄罗斯方块，则我们的目标就是寻找 N 个体积较大且体积近似的联通块。得到了这 N 个联通块则代表了 N 个俄罗斯方块的分割。

这一种方案虽然简单，但是在实践中证明十分有效。但它要求 RGB-D 相机的视野中不能出现与俄罗斯的方块大小相近的块，且每一次放入俄罗斯方块后需要给出放入俄罗斯方块的个数。

Algorithm 1: Canny 边缘检测器

Input: 输入灰度图像:(M, N)

Output: 边缘 Mask:(M, N)

1. 高斯滤波

首先，通过高斯滤波器来平滑图像，以减少噪声对边缘检测的影响。高斯滤波器的公式为：

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

其中， σ 是高斯核的标准差。

2. 计算梯度

梯度的幅值 $M(x, y)$ 和方向 $\theta(x, y)$ 计算如下：

$$M(x, y) = \sqrt{G_x^2 + G_y^2}$$

$$\theta(x, y) = \arctan\left(\frac{G_y}{G_x}\right)$$

3. 非极大值抑制

对梯度幅值进行非极大值抑制，细化边缘。

- 根据梯度方向 $\theta(x, y)$ ，比较当前像素与梯度方向上相邻的两个像素值。
- 如果当前像素的梯度幅值不是最大的，则将其置为 0。

4. 双阈值检测及边缘连接

通过双阈值方法检测和连接边缘。设高阈值为 T_H ，低阈值为 T_L ，其中 $T_H > T_L$ 。

- 对梯度幅值大于高阈值 T_H 的像素标记为强边缘。
- 对梯度幅值在低阈值 T_L 和高阈值 T_H 之间的像素标记为弱边缘。
- 通过滞后连接，将弱边缘连接到强边缘上。其具体算法为：1) 遍历图像中所有标记为弱边缘的像素。2) 如果该像素的邻域中存在强边缘，则将该像素也标记为强边缘，否则将其置为 0。



3.1.2. 基于最小旋转矩阵算法的俄罗斯方块位姿估计与类别分析

最小外接旋转矩形(Minimum Bounding Rectangle, MBR) [5] 是一种用于确定覆盖二维空间中一组点的最小矩形的算法，其算法流程如算法2所示。

对于每个识别出的俄罗斯方块分割，我们先采用 `cv2.findContours` 函数得到边缘点集，再采用最小外接旋转矩形得到矩阵。从而得到了旋转角。

对于第一代的方案，我们还需要根据旋转矩阵与 Mask 得到联通块的类比。

首先判断矩阵长宽是否接近，若接近则为正方形块 O

其他的矩阵均为 2×3 形状，我们将其分为 6 个正方形块，对每个块的中心在 Mask 上采样，得到 6 维的 0/1 数组，其中有四个位为 1。根据该数组既可以判断方块的类型与是否旋转 180 度。

3.1.3. 基于 YOLO 的俄罗斯方块类别、位置与姿态检测

对不同俄罗斯方块的类别与位姿检测是通过 Yolo 网络训练而成的，大致可分为物体边框检测，分类预测与抓取角度预测三方面。

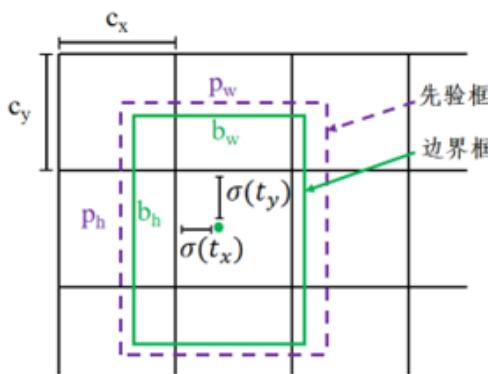


Figure 1. 边框检测

抓取网络模型如图使用多个先验框来预测物体的边界框，原理图如图1所示。

Algorithm 2: 最小外接矩形

Input: 点集 S

Output: 最小外接矩形

1. 计算凸包：

通过 Graham 扫描或 Andrew's 算法，计算给定点集的凸包。

假设凸包的顶点依次为

$(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ 。

2. 旋转卡壳法：

- 遍历每条凸包的边作为矩形的一个边，计算与此边平行和垂直的矩形。

- 对于每条边，计算其方向向量并找到矩形的宽和高。方向向量可以表示为：

$$\vec{d} = (x_{i+1} - x_i, y_{i+1} - y_i)$$

3. 计算与边平行和垂直的点投影来确定矩形的宽和高。

- 投影点到边的方向向量上的坐标：

$$\text{proj}_{\vec{d}}(x, y) = \frac{(x - x_i)(x_{i+1} - x_i) + (y - y_i)(y_{i+1} - y_i)}{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$$

- 计算宽和高：

$$\text{width} = \max(\text{proj}_{\vec{d}}(x_j, y_j)) - \min(\text{proj}_{\vec{d}}(x_j, y_j))$$

$$\text{height} = \max(\text{proj}_{\vec{d}^\perp}(x_j, y_j)) - \min(\text{proj}_{\vec{d}^\perp}(x_j, y_j))$$

其中， \vec{d}^\perp 是 \vec{d} 的垂直方向向量。

4. 计算面积并选择最小值：

对于每个边方向计算的矩形，其面积：

$$\text{area} = \text{width} \times \text{height}$$

选择面积最小的那个矩形作为最小旋转矩形。



训练时的具体方法见算法3

Algorithm 3: 边框检测训练

Input: 网络为每个边界框的 4 个预测值 t_x, t_y, t_w, t_h 。实际此栅格相对于图像左上角偏移量 (c_x, c_y) , 边界先验框的宽和高为 p_w, p_h

Output: 计算边界框:

$$x = t_x + c_x$$

$$y = t_y + c_y$$

$$w = p_w e^{t_w}$$

$$h = p_h e^{t_h}$$

式中 $(.)$ 表示 sigmoid 函数, 其输出范围为 $(0,1)$, 因此可以将网络对边界框中心点的预测输出值 x, y 限制在栅格之中

训练中使用均方误差计算预测边界框的损失。

而分类预测中, 对于每个边界框使用多标签分类来预测边界框可能包含的类别。由于抓取目标为俄罗斯方块, 共包含 7 种不同形状和颜色的目标, 训练中使用交叉熵损失进行类别预测, 计算公式如下:

$$\text{LOSS} = - \sum_{i=0}^{S^2} \sum_{j=0}^B \left(1_{ij}^{obj} \sum_{c \in \text{classes}} \hat{p}_i(c) \log(p_i(c)) \right)$$

其中 $p_i(c)$ 和 $\hat{p}_i(c)$ 分别表示网络对于第 i, j 处的边界框中物体是否为类别 c 的预测值和标签值。

而在抓取角度预测中, 对于 “L”、“J” 和 “T” 形状俄罗斯方块, 在平面内以形心作为原点的旋转角度范围为 $\pm 180^\circ$ 。我们将抓取角度 ϕ 编码为单位圆上的两个矢量 $\cos \phi$ 和 $\sin \phi$, 这样可以将抓取角度从 $[-180^\circ, +180^\circ]$ 压缩到 $[-1, +1]$ 之间, 以消除数据中角度出现的任何不连续性, 有利于网络的学习。同时又保证了两个分量在 $[-180^\circ, +180^\circ]$ 范围内是唯一的, 并在 $\pm 180^\circ$ 处对称。网络通过学习

输出 $\cos \phi$ 和 $\sin \phi$, 之后通过下式求得抓取角度 ϕ :

$$\phi = \arctan \left(\frac{\sin \phi}{\cos \phi} \right)$$

同理, “I”、“S” 和 “Z” 形状俄罗斯方块的抓取角度为 $[-90^\circ, +90^\circ]$, 将抓取角度 ϕ 编码为 $\cos 2\phi$ 和 $\sin 2\phi$, 正方形物体的抓取角度为 $[-45^\circ, +45^\circ]$, 将抓取角度 ϕ 编码为 $\cos 4\phi$ 和 $\sin 4\phi$, 同样可以将抓取角度 ϕ 的标签范围设置在 $[-1, +1]$ 之间, 消除了角度的不连续性, 便于网络对角度的学习。

训练中使用均方误差计算预测抓取角度的损失:

$$\text{Loss_ang} = \sum_{i=0}^{S^2} \sum_{j=0}^B 1_{ij}^{obj} [(\cos \phi_i - \cos \hat{\phi}_i)^2 + (\sin \phi_i - \sin \hat{\phi}_i)^2]$$

式中, $\cos \phi_i, \sin \phi_i$ 和 $\cos \hat{\phi}_i, \sin \hat{\phi}_i$ 分别表示网络对于物体抓取角度的预测值和标签值。

3.1.4. 基于 Segment Anything 的数据标注

Segment Anything Model (SAM) [6] 是 Meta AI 的一种新的人工智能模型, 只需单击即可利用它“剪切”任何图像中的任何对象。它已经学会了关于物体的一般概念, 可以为任何图像或视频中的任何物体生成 mask, 甚至包括在训练过程中没有遇到过的物体和图像类型。SAM 足够通用, 可以涵盖广泛的用例, 并且可以在新的图像『领域』即开即用, 无需额外的训练。

SAM 的出现大大提高了数据标注的速度与精度(类似于 Photoshop 中的魔棒), 标注分割数据不再需要用非常多的点去抠边框(类似于 Photoshop 中的钢笔)。

SAM 辅助下的数据标注如图3所示, 通过正例点选中了青蛙和蜗牛, 再通过反例点将蜗牛去除。

SAM 的流程图如图2所示, 我们将所有需要标注的图像分别输入模型后得到 *imageembedding*, 解决了模型大部分的计算负担。然后我们将轻量化的 *maskencoder* 和 *promptencoder* 从 Pytorch 导

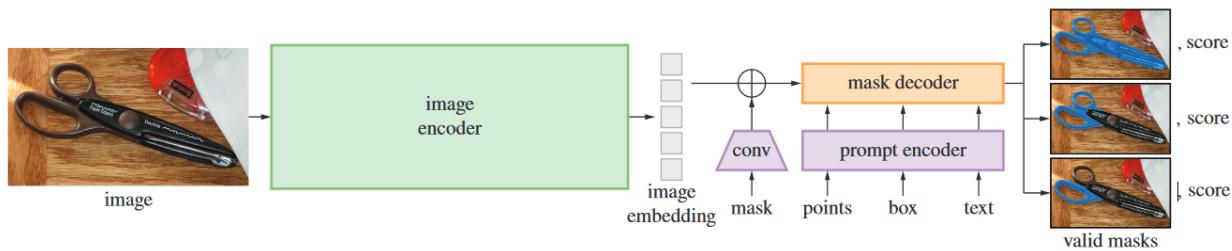


Figure 2. SAM 网络结构



Figure 3. SAM 示意

出为 ONNX 格式，从而可以被软件调用并快速运行。

SAM 开源了一款标注软件，但它只支持单张图片标注，切换图片需要重启软件，非常不方便。于是我们参考了一个开源的代码，基于 PyQt5[7] 搭建的数据标注平台，很好的实践了 segmentation 数据的标注。本平台标注的数据采用 MS COCO[8] 格式。

3.1.5. 基于 Mask R-CNN 的俄罗斯方块智能分割方法

Mask R-CNN 是一个实例分割 (Instance segmentation) 算法，它在 17 年由 Facebook AI Research 提出。它沿用了 Faster RCNN 的思想，特征提取采用 ResNet-FPN 的架构，另外多加了一个 Mask 预测分支。它的网络结构如图4 所示，主要分为两个部分，一部分是引出 class box 的分支，它实际上是 Faster RCNN 结构；另一部分是引出 Mask 的分支，它是一个 FCN 结构。也就是说，Mask RCNN 是在 Faster RCNN 的基础上添加了一个 FCN 结构。

而在本项目中，采用3.1.4中标注的数据集，尝试了多种训练超参数，并对尝试冻结 ResNet-FPN 的 backbone，最终得到 Accuracy 和 Recall 均高于 98% 的模型权重。

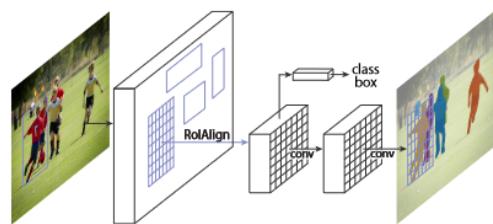


Figure 4. Mask R-CNN 网络结构



3.1.6. 基于 DeepSORT 的俄罗斯方块跟踪策略

SORT (Simple Online and Realtime Tracking) [9] 是一种综合型的目标跟踪框架，简单来说，它结合了 Faster R-CNN 框架、卡尔曼滤波和匈牙利算法，通过概率预测来辅助前后关联匹配。因此 SORT 能够做到快速准确实时的目标跟踪效果。在识别阶段，SORT 算法根据 Faster R-CNN 给出的对于单帧画面的预测，判断需要跟踪的目标在画面中的具体位置。而在预测阶段，SORT 建立了一个状态的传递模型，其中每个状态用 $x = [u, v, s, r, u', v', s']'$ 表示， u, v 代表物体的坐标， s, r 代表边界框的尺寸参数， u', v', s' 则表示预测的下一帧的相关参数，通过卡尔曼滤波来进行当帧的检测框情况与实际目标之间的关联。如果检测到的当帧位置与依照模型该目标可能运动到的位置出现较大偏差（比如说快速移动中的物体突然向反方向运动），则会降低这个关联的可能性。对这个指标进行量化，以此为准则计算目标与检测框的 IoU，再通过匈牙利算法进行匹配。如果指标经过匹配后大于一定阈值，则考虑为这个目标新建 ID。

为了更好地利用视频画面所带来的信息，在此基础之上的 DeepSORT[10] 中使用了深度学习特征提取网络，来对画面中目标的外观特征进行提取分析。DeepSORT 具体使用的是一个 2 个卷积层 + 6 个残差结构的网络，通过对大量行人特征进行提取训练，使得该结构可以有效提取出行人之间的不同特征。最后通过余弦距离，将不同检测框的特征之间的距离进行量化，也加入到匈牙利算法的匹配当中，这样就能够综合图像画面信息到轨迹串联环节里。

而在本实验中，我们将 DeepSORT 中的行人特征换为俄罗斯方块的特征，初步尝试了 DeepSORT 在俄罗斯方块跟踪上的应用，并取得了一定的效果。

3.2. 坐标系转换与基于棋盘格的标定

本项目中有三个坐标系，一个是 RGB-D 相机的坐标系 A ，一个是机器人的坐标系 U ，还有一个是放置盘的坐标系 B 。我们这里需要计算 ${}^B_U T$ 和 ${}^A_U T$ 。

对于 ${}^A_U T$ ，我们在相机前放置水平棋盘格，通过 OpenCV 的库检测该棋盘格的所有角点 P_A ，并调控机械臂末端，记下其在每个点时末端在机器人坐标系下的位置 P_U 。

则求解 ${}^B_U T$ 即对于 k 组 P_A, P_U ，拟合下列方程

$${}^A_U T \times P_U = P_A$$

使其均方误差最小。具体有如下两种实现方式，而 ${}^B_U T$ 同理。

不过在第一代的方案中，我们假设 RGB-D 相机水平放置，放置俄罗斯方块的桌面水平，从而忽略了 z 轴，以减少标定量。

3.2.1. 基于 numpy 的实现

numpy 中的 lstsq 库的输入输出如算法 4 所述。

Algorithm 4: numpy.linalg.lstsq

Input: $a: (M, N), b: \{(M,), (M, K)\}$

Output: $x: \{(N,), (N, K)\}$, which is the least-squares solution to the linear matrix equation

$$a \times x = b$$

3.2.2. 基于 POINT-POINT Iterative Closest Point 的实现

POINT-POINT Iterative Closest Point[11] 的算法流程如算法 5 所示。

**Algorithm 5:** POINT-POINT

Iterative Closest Point

Input: 点集合 $A = \{a_1, \dots, a_n\}$, $B = \{b_1, \dots, b_n\}$

Output: 旋转平移量 R, t , 使得两组数据形成最佳匹配, 即两组数据的距离误差最小, 即

$$\min \frac{1}{2} \sum_{i=1}^n \|a_i - (Rb_i - t)\|^2$$

定义两组点集合的质心位置 a, b

$$a = \frac{1}{n} \sum_{i=1}^n a_i \quad b = \frac{1}{n} \sum_{i=1}^n b_i$$

计算每个点的去质心坐标

$$q_1 = a_1 - a \quad q'_1 = b_1 - b$$

根据以下优化问题计算旋转矩阵

$$R^* = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^n \|q_i - Rq'_i\|^2$$

$$W = \sum_{i=1}^n q'_i q_i^T \quad W = USV^T$$

$$R^* = VU^T$$

$$t^* = a - R^*b$$

3.3. 机械臂位姿解算与运动规划

我们采用的大象机器人为六轴机械臂, 配有相应的控制柜。第一代机器人中, 我们采用官方的 python 函数来对大象机器人进行控制。其中我们采用了点对点的控制函数 `set_coords(self, coords_array, speed)` 来进行点到点的机械臂运动, 通过输入视觉识别得到的目标点的笛卡尔坐标和设定的运动速度来使机械臂移动。

而在二代机器人中, 由于加入了手眼相机, 因此当机械臂回到初始位置后, 算法确定下一个待抓

取的俄罗斯方块, 全局相机得出该目标的类型、位置、姿态信息, 并计算出抓取点的 $[x_0, y_0, z_0, \theta_0]$, 即 pos_0 。

之后, 将 pos_{target} 设置为 pos_0 目标点, 六轴机械臂当前末端位置设置为 pos_i , 循环计算手眼相机的视觉标识符与抓取点之间的误差, 并基于该误差在尔空间对机械臂末端进行速度控制, 使误差趋近于 0。当误差达到可接受阈值范围时, 结束循环, 机械臂末端吸盘静止在目标方块抓取点上方。之后启动气泵, 吸起该俄罗斯方块, 并将其移动到已经得到的最终目标位置。

而在改进后的机器人中, 为了加快机械臂大范围移动的速度, 我们利用机器人建模的相关知识, 对于长距离的空间移动采用了角度规划, 通过 IK 算法先将目标位置的空间笛卡尔坐标转换成对应的六个关节角, 再通过直接转动角度实现长距离的快速移动。而对于抓取目标方块的垂直移动我们依然采用之前的笛卡尔空间规划的策略, 由控制柜来规划路径。

3.3.1. 基于雅克比迭代的 IK Solver

我们的 IK 求解器通过机械臂的 DH 参数表, 由雅克比迭代求解机器人的逆运动学, 通过目标的设定位姿来获得六个对应关节角的设置值。

Table 1. 机械臂的 DH 参数表

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	111	$q_1 - \frac{\pi}{2}$
2	$\frac{\pi}{2}$	0	123.8	$q_2 + \frac{\pi}{2}$
3	0	264	-111.8	q_3
4	0	236	101.8	$q_4 - \frac{\pi}{2}$
5	$-\frac{\pi}{2}$	0	101.8	q_5
6	$\frac{\pi}{2}$	0	90.238	q_6

借助正运动学, 我们用算法6通过数值微分的方法求解雅克比:

**Algorithm 6:** IK 迭代求解算法

Input: 目标位姿的转换矩阵 T_{tar} , 初始位姿的六个关节角 $joints_init$

Output: 提供一个初始的关节位置估计值 q_0 , 按以下步骤一直迭代到 δp 很小:

$$p_{current} = f_k(q_{current})$$

$$\delta p = p_{target} - p_{current}$$

$$J = J_{accobia}(q_{current})$$

$$\delta q = J^{-1} \cdot \delta p$$

3.4. 俄罗斯方块方案生成

俄罗斯方块放置需要通过当前盘面情况和目标方块的种类，确定当前块放置的位置和姿态，从而输出给控制模块进行装配。根据已知信息的不同，将俄罗斯方块放置的方案分为两种。

3.4.1. 已知所有块情况的生成方案

针对已知所有要放置的所有俄罗斯方块的类型、数量的情况，采用离线算法，用搜索的方法确定最佳的放置方案，保存得到的结果，在视觉处理部分输入当前方块信息后，输出最优的放置位置和姿态。

3.4.2. 仅知道当前盘面情况下的在线生成方案

针对仅知道当前盘面信息以及即将放置的方块种类的情况，通过对所有可放置位置进行评估，选择最优的位置进行放置，得到在线方案，具体步骤如下：

1. 预处理：标记每一类俄罗斯方块的中心方格，存储列举该方块的所有姿态
2. 获取可行解：遍历每个中心位置以及方块的姿态，判断若将方块放置在该位置上是否

会与已经放置的方块发生冲突，如果该位置可以放置，则将其加入到可行解中

3. 计算价值函数：为获得最优解，通过价值函数对每个由可行位置和放置姿态组成的元组进行价值评估。价值函数的影响因素如下，其中，记有方块放置的网格为满位，没有方块放置的位置为空位：

- 中心坐标位置 $[lh, lw]$: 为了保证俄罗斯方块机器人在装配过程中的可靠性，优先选择位于盘面左下角的位置
- 相邻网格中被占据的位置数 nx : 为充分利用盘面空间，优先选择与其他块靠近的位置
- 放置后新增被填满的行数 $epcm$: 越大越好
- 新盘面中空位组成的连通块个数 $empt$: 一般情况下，由于被满位包围而形成的空位连通块规格较小，且在后续装配的过程中难以被填充，因此，希望 $empt$ 值越小越好
- 新盘面中满位组成的连通块个数 $colo$: 由于希望满位尽量相邻，因此，希望 $colo$ 越小越好
- 新盘面中空位所在的“井”数 bw : 在这里，“井”指一个空位出现在两个满位之间的情况，在装配过程中，“井”的出现提高了对机器人装配精度的要求，容易导致装配过程中出现浮动等情况，不利于装配过程的连贯性和安全性。因此， bw 值越小越好

在此基础上，对每个影响因素加以不同的权重，得到每个可行解的价值

4. 选取最优解：选取可行解中价值最大的位置和放置姿态，并用该最优解更新盘面信息

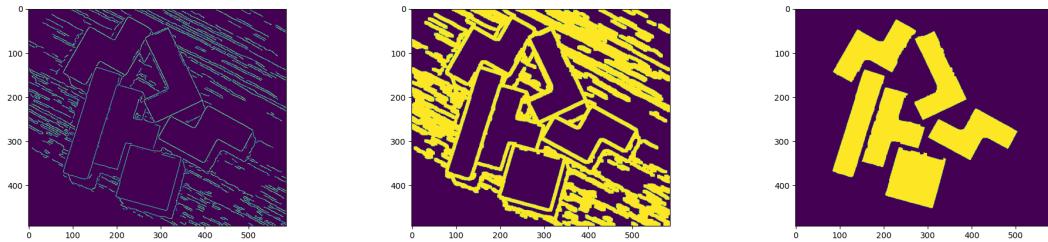


Figure 5. 俄罗斯方块分割

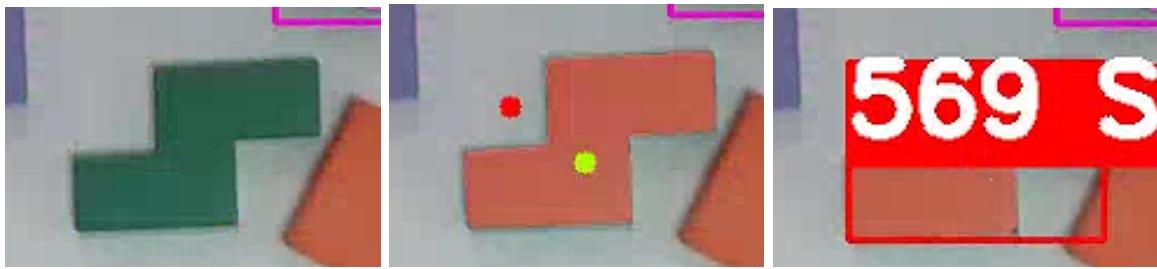


Figure 6. 数据标注流程

4. 结果与展示

4.1. 俄罗斯方块识别与位置与姿态检测

4.1.1. 基于传统机器视觉的俄罗斯方块分割、位姿估计与类别分析

如图5所示，先对输入图像进行 Canny 边缘检测得到左图，进行三次膨胀后得到中图，根据连通块面积处理后得到右图，黄色部分为俄罗斯方块的分割。

最终经过最小旋转矩阵与类别分析处理后得到结果如下，图像结果如图7

```
1 class= 1 theta= 164.74488067626953 position=
 [272.6576843261719, 381.0884704589844]
2 class= 4 theta= 342.4744338989258 position=
 [231.15615844726562, 277.83880615234375]
3 class= 0 theta= 162.29956817626953 position=
 [142.62265014648438, 260.3157043457031]
4 class= 3 theta= 149.53445434570312 position=
 [179.01541137695312, 96.77379608154297]
5 class= 2 theta= 241.32685470581055 position=
 [402.4243469238281, 284.1383056640625]
6 class= 6 theta= 384.8141975402832 position=
 [326.56313655088553, 137.86329131960156]
```

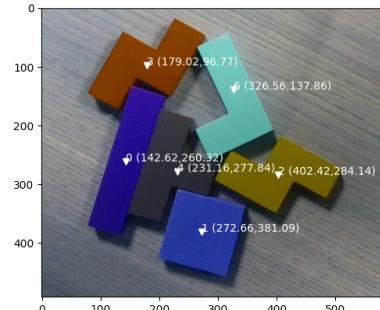


Figure 7. 方案一识别结果

4.1.2. 基于 YOLO 的俄罗斯方块类别、位置与姿态检测

如图8,9所示，通过 Yolo 可以很好的得到识别的结果。通过全局相机的视觉处理可以很好的得到各个方块的类别和位姿，而动态抓取的过程中手眼相机能够使机械臂精确的停在待抓取的俄罗斯方块上并完成抓取。

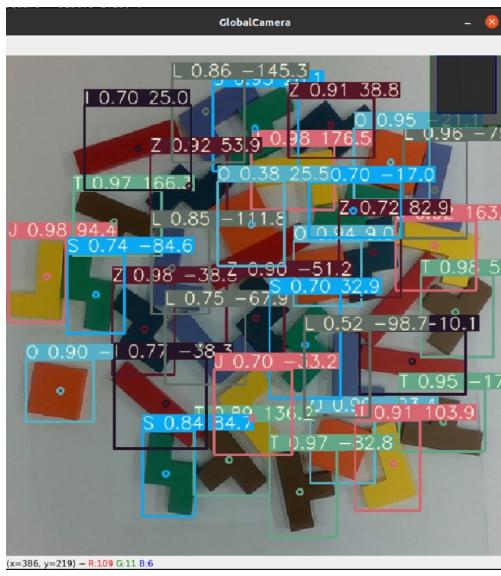


Figure 8. 全局相机视觉处理结果

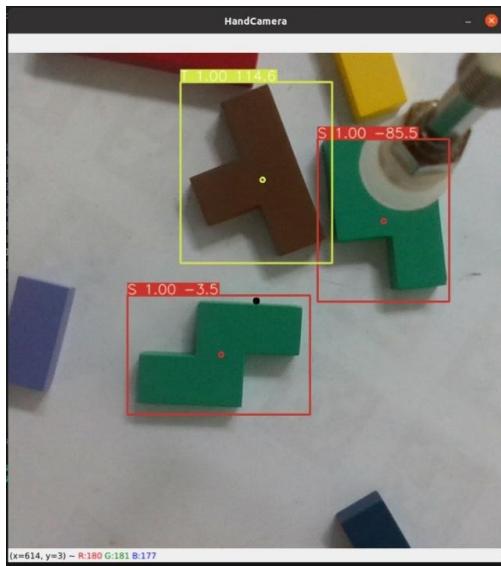


Figure 9. 手眼相机抓取过程

4.1.3. 基于 Segment Anything 的数据标注

数据标注平台的整体示意图如图10所示，右侧可以选择俄罗斯方块的类型，左键可添加正例点，右键可添加反例点，完成标注后点击上方的“添加对象”按钮即可完成一个实例的标注。具体的标注流程可见图6



Figure 10. 数据标注平台

4.1.4. 基于 Mask R-CNN 的俄罗斯方块智能分割方法

Mask R-CNN 的输出效果如图11所示。原本在 NMS=0.3 情况下会出现较多重复的检测框。本处通过后期再进行一次阈值为 0.7 的非极大值抑制，得到较好的效果。

训练得到的最佳权重对应 Box 和 Mask 的精度 Average Precision(AP) 和召回率 Average Recall(AR) 如表2,3所示。

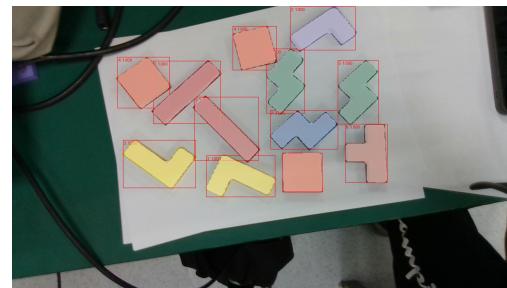


Figure 11. Mask R-CNN 输出结果

4.2. 机械臂位姿解算与运动规划

对于第一代机器人，我们通过 python 函数对其下达笛卡尔坐标系下的位置控制指令。控制指令集成在 python 文件 elephant_command.py 中，我们主要图12使用的 set_coords() 函数来控制机械臂运动到目标点。



Metric	IoU	Area	maxDets	Value	Metric	IoU	Area	maxDets	Value
AP	0.50:0.95	all	100	0.980	AP	0.50:0.95	all	100	0.980
	0.50	all	100	0.989		0.50	all	100	0.989
	0.75	all	100	0.989		0.75	all	100	0.989
	0.50:0.95	small	100	-1.000		0.50:0.95	small	100	-1.000
	0.50:0.95	medium	100	0.987		0.50:0.95	medium	100	0.982
	0.50:0.95	large	100	0.975		0.50:0.95	large	100	0.978
AR	0.50:0.95	all	1	0.681	AR	0.50:0.95	all	1	0.681
	0.50:0.95	all	10	0.986		0.50:0.95	all	10	0.985
	0.50:0.95	all	100	0.986		0.50:0.95	all	100	0.985
	0.50:0.95	small	100	-1.000		0.50:0.95	small	100	-1.000
	0.50:0.95	medium	100	0.992		0.50:0.95	medium	100	0.989
	0.50:0.95	large	100	0.978		0.50:0.95	large	100	0.980

Table 2. Bounding Box Metrics**Table 3.** Segmentation Metrics

```
def set_coords(self, coords_array, speed):
    '''设定机械臂目标点(毫米)和运动速度(毫米/秒)'''
    coords_msg = self.set_coords((1,1))
    coords_msg.x = coords_array[0]
    coords_msg.y = coords_array[1]
    coords_msg.z = coords_array[2]
    coords_msg.speed = speed
    self.sock.sendall(coords_msg, encoding='utf-8')
    back_msg = self.sock.recv(1024)
    print(back_msg)
```

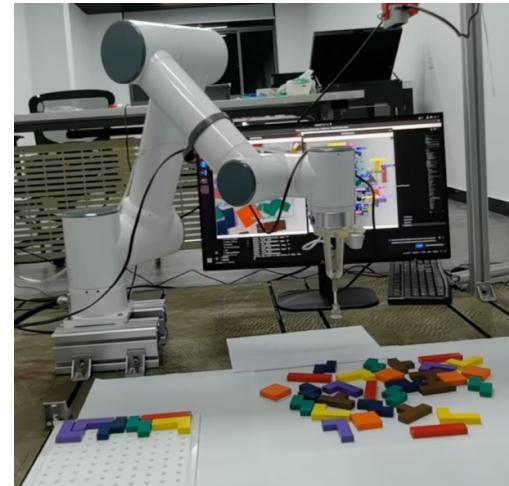
Figure 12. set_coords 函数

对于第二代机器人，加入手眼相机后增加了伺服控制环节，能够更加精确的抓取目标。工作状态见图13

而改进后的机器人对于大范围移动（如抓取完目标方块后移动到放置点）加入了关节规划。由于机械臂的故障，因此我们在 coppeliasim 中通过仿真实现了关节角的控制，从设定的目标抓取点通过关节角的规划移动到目标放置点。

4.2.1. 基于雅克比迭代的 IK Solver

通过调用 IK Solver 的 python 文件，我们在 coppeliasim 中实现了关节角的控制，并能通过输入目标抓取点的位姿和目标放置点的位姿，通过逆运动学求解出两个位姿处的关节角并进行机械臂控制。在仿真中，我们验证了我们 IK 求解算法的正确性。由于 coppeliasim 仿真中基坐标为 (x=0,y=0,z=25) (毫米)，因此减去基坐标的偏差，

**Figure 13.** 加入手眼相机后的抓取过程

IK 求解器的准确性得到了验证。以一组抓取过程为例：

通过图17中的计算结果与图18,19中的仿真结果对比可以看到，计算结果的 xyz 坐标和实际 coppeliasim 仿真出来的结果基本吻合（仿真中由于机械臂基坐标系的原因 z 轴坐标会低 25mm），由此验证了我们 IK 求解器的准确性。

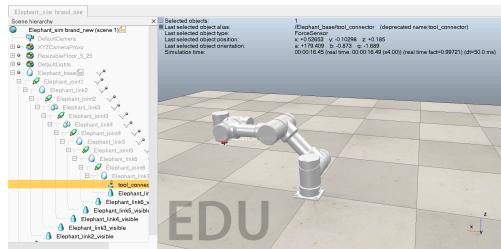


Figure 14. 目标抓取点

```

75 if name == "main__":
76     # joints = np.array([0, -np.pi/2, 0, np.pi/2, np.pi/2, np.pi])
77     joints = np.array([-math.pi/4, -math.pi/4, -math.pi/4, 0, math.pi/2, 0])
78     T_tar = fk(joints)
79     T_tar = np.array([
80         [1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0], [-113.8, 262.4762],
81         [0, 0, 0, 1], [0, 0, 0, 207.4382], [0, 0, 0, 1]
82     ])
83
84 joints_init = joints + np.random.uniform(-0.5, 0.5, 6)
85 joints_init=np.array([0,0,0,0,0,0])
86 print(T_tar)
87 q = ik(T_tar, joints_init)
88 for i in range(6):
89     while(q[i]<math.pi):
90         q[i]=q[i]+math.pi*2
91     while(q[i]>math.pi):
92         q[i]=q[i]-math.pi*2
93 print(q)
94
95 T_tar = np.array([
96     [-0.7071, -0.7071, -0.7071, -451.3204],
97     [-0.7071, 0.7071, 0.7071, -290.3919],
98     [0, 0, 0, 207.4382], [0, 0, 0, 1]
99 ])
100
101 joints_init = joints + np.random.uniform(-0.5, 0.5, 6)
102 joints_init=np.array([0,0,0,0,0,0])
103 print(T_tar)
104 q = ik(T_tar, joints_init)
105 for i in range(6):
106     while(q[i]<math.pi):
107         q[i]=q[i]+math.pi*2
108     while(q[i]>math.pi):
109         q[i]=q[i]-math.pi*2
110 print(q)

```

Figure 17. IK 代码以及两个位置逆运动学求解结果

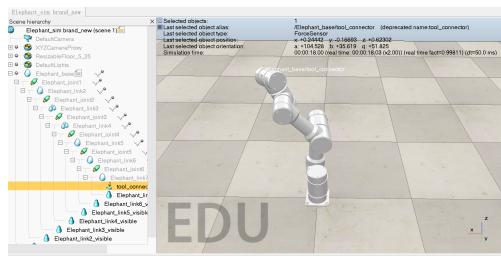


Figure 15. 关节角规划中间位姿

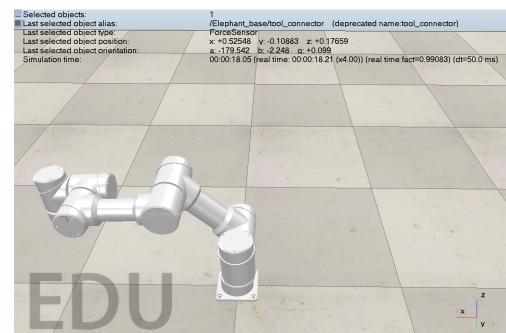


Figure 18. 目标抓取位置的位姿信息

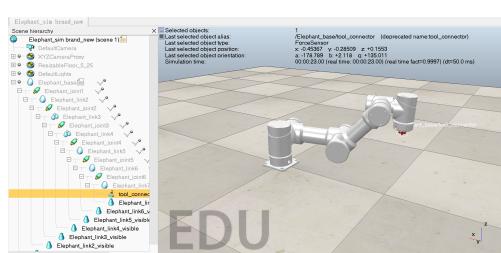


Figure 16. 目标放置点



Figure 19. 目标放置位置的位姿信息



4.3. 俄罗斯方块方案生成

俄罗斯方块放置需要通过当前盘面情况和目标方块的种类，确定当前块放置的位置和姿态，从而输出给控制模块进行装配。根据已知信息的不同，将俄罗斯方块放置的方案分为两种。

4.3.1. 已知所有块情况的生成方案

针对已知所有要放置的所有俄罗斯方块的类型、数量的情况，采用离线算法，用搜索的方法确定最佳的放置方案，保存得到的结果，在视觉处理部分输入当前方块信息后，输出最优的放置位置和姿态。对于七种俄罗斯方块各 5 个的情况，通过搜索，得到最佳的放置方案如图20左图所示

4.3.2. 仅知当前盘面情况下的在线生成方案

依次放置类型为 $T, Z, S, L, J, I, O, O, T$ 的方块，得到的放置方案如图20右图所示，其中，网格上的序号表示该处方块的序号

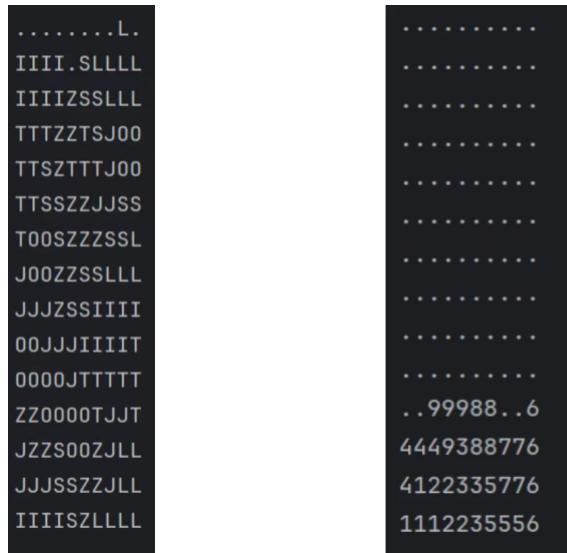


Figure 20. 左图：离线方案；右图：依次放置类型为 $T, Z, S, L, J, I, O, O, T$ 的方块方案

5. 总结与展望

本项目通过对俄罗斯方块拼接机器人的设计和改良了解了装配机器人的原理、控制和应用，完成了由基于传统机器视觉方法、PD 规划算法和大象机器人平台的第一代机器人到基于 MASK R-CNN 算法、改进后的在线规划算法以及 ROS 平台的第三代机器人的迭代，实现了俄罗斯方块机器人识别、规划、装配三个方面的改进，具有重要的意义。

通过本次项目，我们认识到了装配机器人在生产过程中的重要性以及其设计的复杂性，学习了视觉处理、运动规划等算法，获得了很大的收获。同时，基于对俄罗斯方块拼接机器人的研究，我们对装配机器人的改进提出了设想。首先，在目标抓取上，可以采用目标跟随等方法，提高抓取的精确度，并能更好地适应实际装配过程中零件移动的情况；其次，在方案设计上，考虑对装配顺序进行合理的设计，减小装配过程中受到其他零件的干扰；最后，在装配规划上，增加避障等功能，保证装配过程中的安全性以及装配机器人和工作人员的交互能力。



References

- [1] S. Algorta and Ö. Şimşek, “The game of tetris in machine learning,” *arXiv preprint arXiv:1905.01652*, 2019.
- [2] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
- [4] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.
- [5] D. Papadias and Y. Theodoridis, “Spatial relations, minimum bounding rectangles, and spatial data structures,” *International Journal of Geographical Information Science*, vol. 11, no. 2, pp. 111–138, 1997.
- [6] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, “Segment anything,” *arXiv:2304.02643*, 2023.
- [7] J. M. Willman, *Beginning PyQt*. Springer, 2020.
- [8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.
- [9] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016, pp. 3464–3468.
- [10] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.
- [11] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.