浙江大学

自动化综合实验结题报告

题目　基于机器视觉的质量检测与智能分拣

所在小组　　第 1 组

指导老师　　冯毅萍 赵久强 曹峥

所在学院　　控制科学与工程学院

结题时间　　2020 年 1 月

目 录

基于机器视觉的质量检测与智能分拣

邓家超 曾之宸 陈伽洛 徐博文 王镇林

（浙江大学控制科学与工程学院，浙江 杭州 310027）

摘 要：

工件分拣是工业生产环节的重要环节,相比于传统的人工分拣，基于机器视觉的机械臂分拣方案具有实时分析、高效准确以及稳定的优势。本项目关注于基于机器视觉的质量检测与分拣环节；通过图像处理与深度学习算法对工业铝材表面瑕疵情况进行辨别，并基于识别结果指导机械臂完成相应的抓取与分拣工作。

本文对项目中完成的工作进行汇报。首先介绍项目的研究背景、整体方案与技术路线。其次对项目涉及的图像处理算法与瑕疵检测方法进行介绍，主要包括支持向量机以及 InceptionV4 网络，并对算法的结果进行分析比较。之后针对智能分拣中的机械臂建模仿真、运动规划以及相应实体的 API 开发进行了阐述。在底层方案得以落实的基础上，本文后续介绍并展示了对应的上位机界面。最后将对项目进行总结与展望。

关键词：质量检测；智能分拣；支持向量机；InceptionV4；机械臂建模与仿真

1 问题背景与重述

1.1 研究背景

工件的质量检测与分拣是工业流程中的重要环节。当下实际工业生产流程中，大多数质量检测与工件分拣工作依赖于人工完成。相比于人工操作，近年兴起的基于机器视觉的技术有着不少优势。两种质量检测方法的比较如图 1.1 所示。人工检测方法有着显著的缺陷，一方面其受到检测员主观判断的影响，另一方面人工检测方法效率低下且检测精度有限；再则，此类检测任务繁重枯燥，且要求检测员长时间高度地集中精力，所以希望能开发相应的智能检测系统以代替人工检测方法。随着深度学习算法的推广，基于机器视觉的质量检测得到了业界的广泛认可。基于机器学习的质量检测算法所拥有的高效、准确以及稳定的特性是人工检验所不具备的；同时，机器视觉方法能很大程度上将劳动力从枯燥的劳动中解放出来。

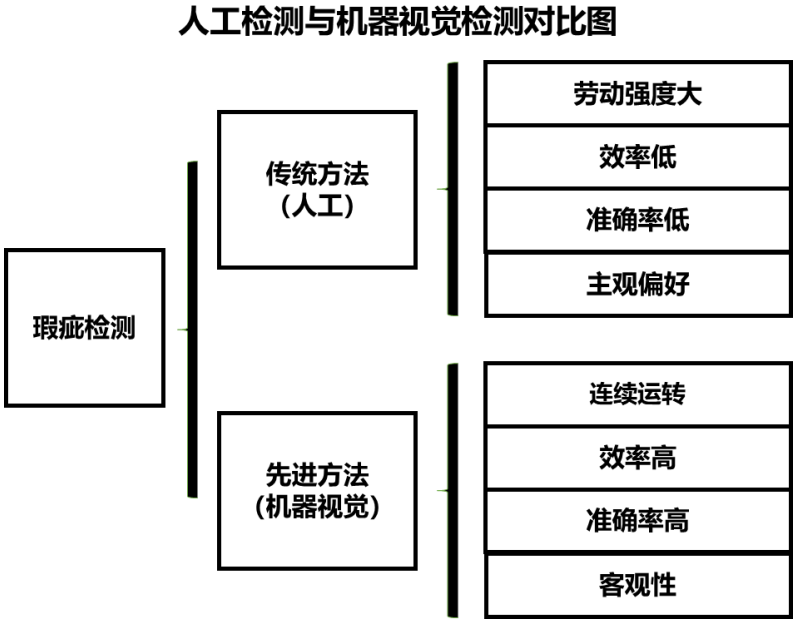

图 1.1　人工检测与机器视觉检测比较

而对于机械臂分拣系统，智能分拣系统通过有着不可比拟的优势，两种分拣方法的比较如图 1.2 所示。

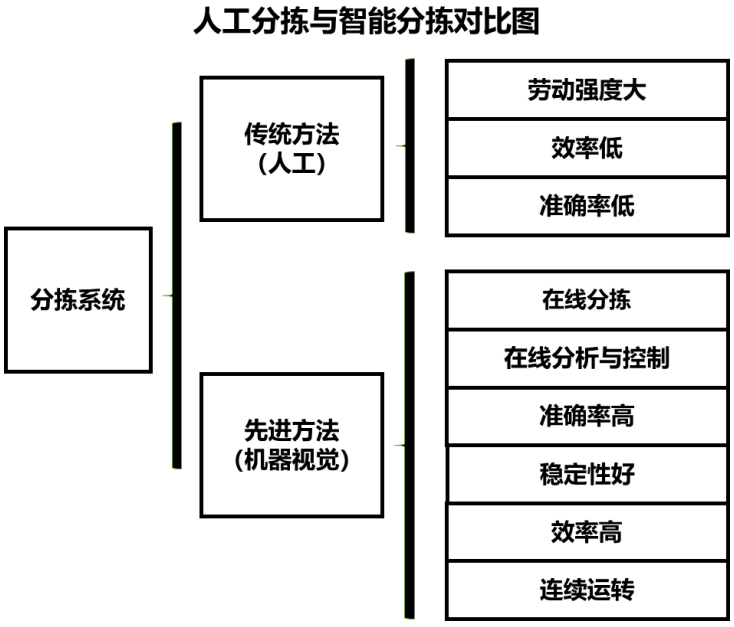

图 1.2　人工分拣与机器视觉分拣比较

当下人工分拣系统主要面临着劳动强度大、分拣准确率及效率低下的问题。而智能分拣系统不仅能够解决上述问题，并且还可以实现在线分拣、分析以及控制；同时，智能分拣系统相比于人工分拣具有更好的稳定性与更高的可靠性。基于以上几点，本项目所研究的“基于机器视觉的质量检测与智能分拣”项目是具有实际意义与应用价值的。

1.2 问题重述

本项目主要关注基于机器视觉的质量检测与智能分拣问题。主要可以将问题分为两个部分：①机器视觉质量检测；②机械臂分拣。机器视觉质量检测中主要关注于如何构建深度学习分类器，对摄像头读取的工件表面图像进行处理并识别相应的瑕疵类型；机械臂分拣中主要关注于如何在接收到质量检测算法的检测结果后，对检测后的工件进行对应的分拣。

本项目主要有以下几个问题需要考虑：

1. 如何构建高效的质量检测算法。鉴于摄像头获取 RGB 三通道图像涉及数据量庞大，再加之深度神经网络的复杂结构与参数，深度学习分类器的算法效率可能较为低下。而较低下的检测效率是不适用于实际的高速生产工业流程。
2. 如何构建准确的质量检测算法。鉴于工业过程对于产品的高要求，开发的质量检测算法要具有足够的检测准确度才能投入到实际生产中。而在工艺加工过程中，工件的瑕疵种类复杂、面积小，再加之摄像头捕获流水线上动态图像可能造成的图像降质问题，这给质量检测算法的准确度提出了很高的要求。
3. 如何构建鲁棒的质量检测算法。工业过程中的随机性因素很大，瑕疵种类繁多且表现差异大，再加之实际生产过程中人员流动等因素给摄像头捕捉图片所带来的影响；这要求质量检测算法具有较高的鲁棒性，以应对实际工业生产中的大量不确定因素。
4. 如何对机械臂进行合理的运动规划，包括运行轨迹中间点的衔接，速率及位置的差值，以及轨迹中可能存在的奇异点的判别与规避。

2 课题内容与技术路线

为了更好地阐述本课题的技术路线，本课题开发系统的架构如图 2.1 所示。

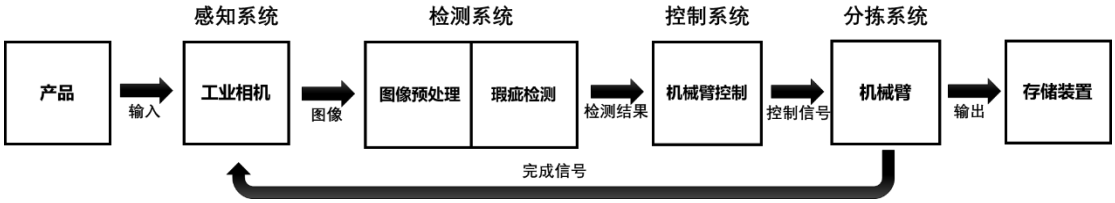

图 2.1 系统架构示意图

首先通过传感系统（工业摄像头）读取工件图像信息，并传输至检测系统中。检测系统中的预处理模块对图像和数据进行增强，并将增强后的图像送至瑕疵检测模块。瑕疵检测模块通过深度学习分类器对图像进行性分类，并将分类结果传

输至机械臂控制系统中。机械臂控制系统依据不同的检测系统执行不同情况下对应的分类动作，并在此模块中完成机械臂的运动规划。分拣系统（机械臂）接收来自控制系统的关节控制信号，将物件放入到不同的类别存储器中，并返回分拣完成信号至传感装置，为接下来的分拣工作做准备。

3 视觉组

3.1 视觉组技术路线

作为一个典型的机器视觉项目，可将瑕疵检测算法的开发分为预处理，特征提取以及瑕疵检测三个环节。技术路线如图 3.1 所示。

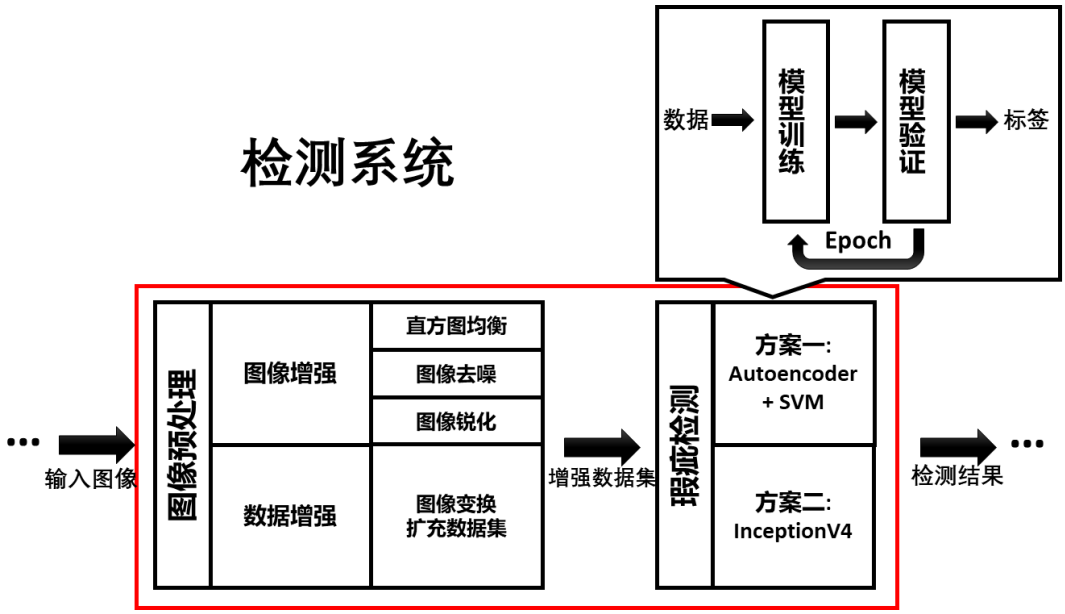

图 3.1　检测系统技术路线示意图

预处理环节通过图像处理方法，对摄像头捕获的图像进行强化，以保证后续环节中图像的质量。由于处理后的 RGB 图像往往就有庞大的数据量，直接将其放入到瑕疵检测环节必定会导致瑕疵检测算法效率低下，所以特征提取环节需要通过机器学习算法对图像进行处理，以提取出就有代表性的特征来降低输入数据量。在提取了有效特征后，可以通过深度学习算法构建分类器，对不同的工件图像进行瑕疵分类工作。

为了后文更好地阐述相关技术，现对本课题涉及的图像数据集进行阐述。本课题采用天池大数据平台提供[1]的 10000 份实际生产中的瑕疵铝型材料监测影像数据，图像数据均为 2560×1920 的 RGB 图像，其中包括“不导电”、“擦花”、“角位漏底”、“桔皮”、“漏底”、“漆泡”、“起坑”、“杂色”以及“脏点”等 10 个瑕疵类别，且数据已打上对应的类别标签。

3.2 图像预处理

3.2.1 直方图均衡

直方图均衡化是以累计分布函数变换法作为基础的，通过变换将像素值分布较为集中、对比度较低的图像进行直方图均衡，使其灰度分布范围更广、更为平均，从而提高图像的对比度。

工程上一般采取如（3.1）所示的变换函数来进行直方图均衡

$$s = T(r) = \int_0^r p_r(\omega)d\omega \tag{3.1}$$

其中 s 为变换后的像素值，r 为变换前的像素值，$p_r(\omega)$ 为变换前图像像素值的概率密度分布函数。对于数字图像，通常采用求和的方式替代连续函数中的积分操作。直方图均衡前后的图像及灰度分布如图 3.2~3.5 所示。

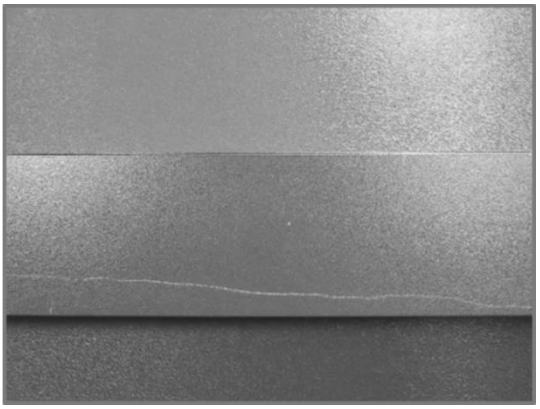

图 3.2 原始图像

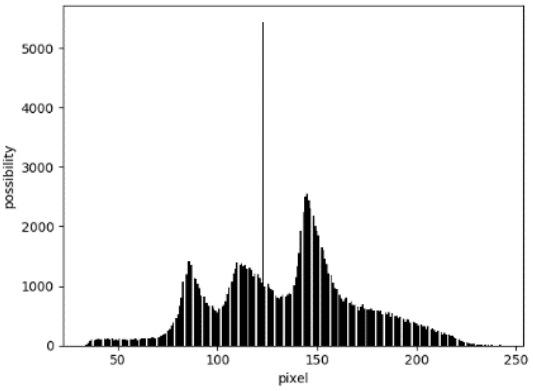

图 3.3 原图像灰度分布直方图

图 3.4 直方图均衡化后图像

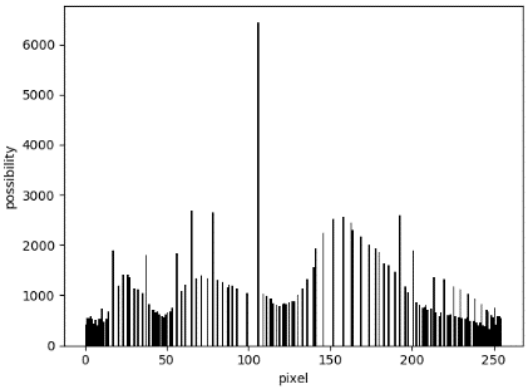

图 3.5 直方图均衡化图像灰度分布

比较变化前后的两张图像，可以明显地感觉到原始图像整体较亮，这导致亮色的瑕疵划痕不太明显；处理后的图像对比度整体提高，降低了图像的亮度，使得对应的瑕疵划痕相比更为突出，

根据两张灰度分布直方图可以看到，原图像像素值集中分布在[80,200]区间

内；经过直方图均衡化处理后，图像的像素值大致均匀地分布在了[0,255]区间内，使图片的像素值范围较大，图像的对比度增强。

3.2.2 图像去噪

由于图像采集过程中常常会受到各种噪声的干扰和影响，引起图像的降质，故而需要通过图像滤波操作来提升图像质量。均值去噪操作通过对一定范围内的图像像素进行卷积平均化处理，以起到降低噪声影响的作用。因为均值滤波的平均化处理，其适用于噪声分布较为均匀的噪声处理，例如高斯噪声。而中值去噪通过对某一像素周围一定范围内的像素点取中值作为该点像素值，其能够对较为明显的噪声有很好的处理效果，例如椒盐噪声。

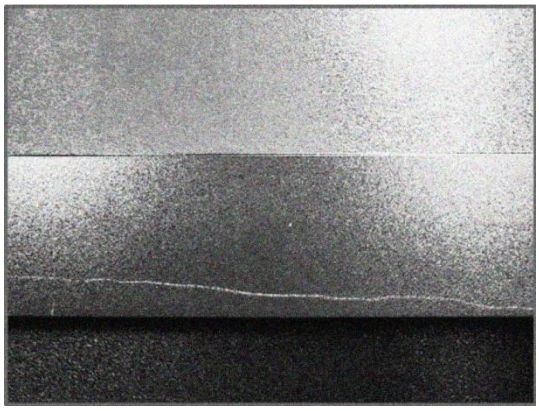

图 3.6 噪声图像

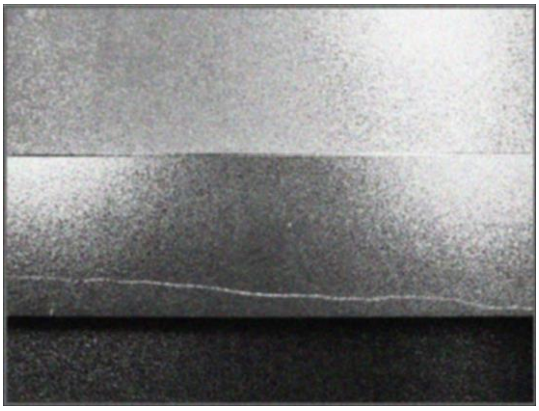

图 3.7 均值滤波后图像

图 3.8 噪声图像

图 3.9 中值滤波后图像

首先，我们采用了 3×3、5×5 和 7×7 的均值滤波卷积核对图像进行处理。对于卷积操作的边界，我们采取填零的方式进行扩充，以保证输出图像与输入图像尺寸相同。结果示意图如图 3.6、3.7 所示。可以看到，虽然均值滤波起到了一定的去噪效果，但是其同时导致了图像清晰度的显著降低，这对于后续的瑕疵检测是非常不利的。

其次，我们采用了 3×3、5×5 和 7×7 的中值滤波卷积核对图像进行处理。对于卷积操作的边界，我们采取填零的方式进行扩充，以保证输出图像与输入图像

尺寸相同。结果如图 3.8、3.9 所示。可以看到中值滤波对于椒盐类噪声有很好的过滤效果，但是其对于高斯噪声去噪能力较差。

综合考虑两种去噪算法的优劣，我们认为全局的去噪算法并不能对图像中局部的噪声特征进行针对性的处理，所以我们采用自适应滤波去噪算法进行处理。很多的自适应去噪算法是基于小波变换实现的：Li 和 Orchard[1]提出了基于小波变换的自适应滤波方法，其通过对变换后的高频小波分量进行处理，区分出其中的边缘特征分量与噪声分量，并采用最佳 MMSE 估计的方法对原图像进行估计，从而实现图像去噪；Zhang 和 Desai[2]提出了一种基于斯坦因无偏风险估计的自适应去噪算法，通过估计图像函数$\hat{f}$，并以最小化估计函数$\hat{f}$与真实函数f为目标，对原始图像进行估计；Scharcanski 和 Jung[3]通过对小波变换得到的分量进行概率建模，并引入空间和比例一致性的方法，以更好地区分边缘特征与噪声分量。由于小波变换的方法较为繁琐，这里我们采用的自适应滤波去噪仍是基于空间域进行的，该方法的基本思想是：对每次选定图像中的部分区域，判断该区域的噪声水平，并在噪声水平达到一定程度时才对该区域进行处理。这样的差别处理方法能够有效避免去噪算法对于高质量的图像区域造成的降质副作用。对于噪声水平的评定，因为相比于原始图像中的信息，噪声信息是很小的，故采用选定区域的最大均方差与最小均方差来进行评判，具体计算公式如（3.2）所示:

$$max\ \ MSE:\begin{cases} max\dfrac{1}{mn}\displaystyle\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-\bar{I}]^2 \\ s.t.\quad (0,0)\le(a,b)\le(p-m,n-q)\end{cases} \tag{3.2}$$

$$min\ \ MSE\begin{cases} min\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-\bar{I}]^2 \\ s.t.\quad (0,0)\le(a,b)\le(p-m,n-q)\end{cases}$$

可以认为，区域中的最大均方差能够代表该区域图像信息的大小，而最小均方差则是代表了该区域噪声信息的大小。之后，通过计算局部信噪比（Local SNR）即可表征该区域变得噪声强度，局部信噪比的计算公式如（3.3）所示:

$$Local\ SNR: 10\,log_{10}(\frac{Max\ MSE}{Min\ MSE}) \tag{3.3}$$

自适应滤波去噪算法框架如图 3.10 所示。

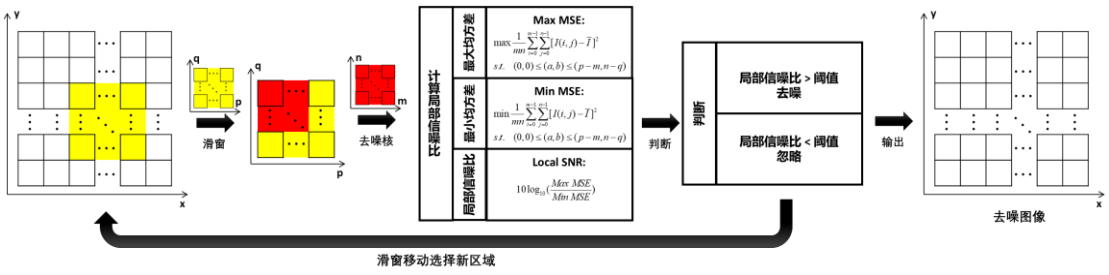

图 3.10　自适应滤波去噪框架

首先，对于输入图像$x \times y$，依据动态窗口选定图像中的某一区域$p \times q$进行处理；对于选定区域$p \times q$，采用大小为$m \times n$的核对其进行处理。其次，$m \times n$核遍历选定的$p \times q$区域，并依据（3.2）与（3.3）中的公式计算核覆盖区域内的局部信噪比，通过比较局部信噪比与设定阈值之间大小关系；当局部信噪低于阈值时，对选定区域进行去噪处理；当局部信噪比高于阈值时，则不进行处理。之后移动动态窗口，选定下一个区域$p \times q$，依次类推，直至图像$x \times y$的所有区域均被遍历时结束。

3.1.3 图像锐化

由于图像去噪后不可避免地会带来图像降质，所以往往在去噪后进行锐化处理以提升图像质量。通过图像锐化技术，能够将图像的纹理特征进行提取，进而使得图像边缘更为明显。在本项目中，我们主要采用了 Robert 算子对图像进行锐化，其基本原理如下。

Robert 算子充分利用了图像边缘特征灰度变化明显，非边缘特征灰度变化缓慢的特征，通过对邻近区域内的相应像素点做差分，从而能利用差分值来对边缘特征进行很好的描述。

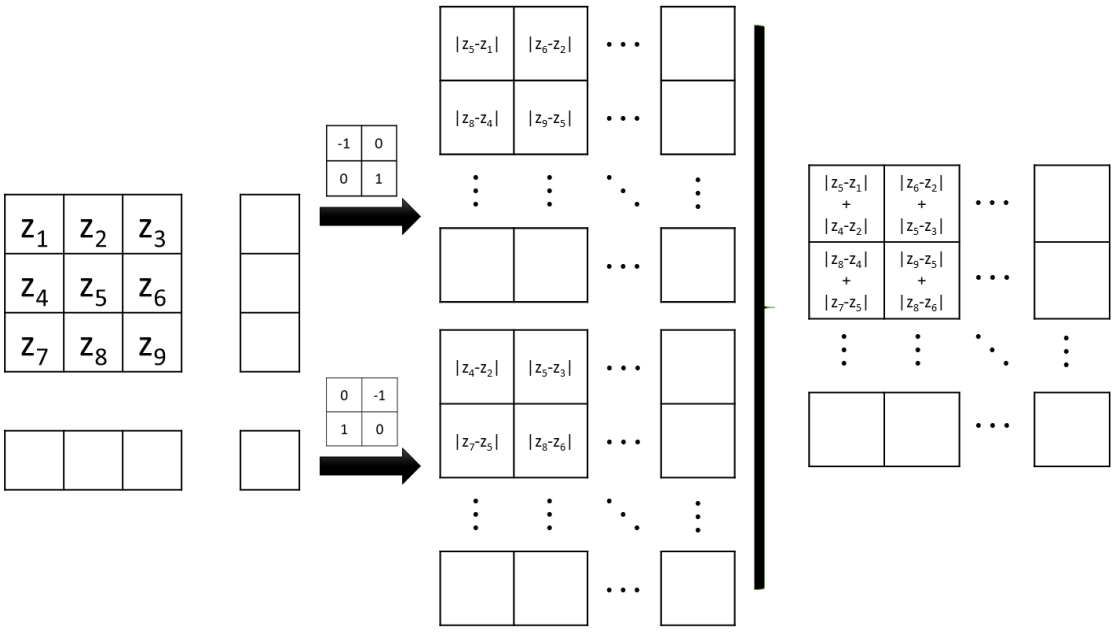

图 3.11 Robert 锐化算法框图

如图 3.11 所示，Robert 锐化首先通过两个 2×2 的卷积核对原图像进行卷积，并将两个卷积结果进行加和，得到的即为图像的边缘特诊。通过将图像的边缘特征按照预定的比例添加至原图像，即可对原图像边缘进行增强，从而起到图像锐化的作用。Robert 锐化的结果如图 3.12~3.14 所示。

图 3.12 锐化前图像

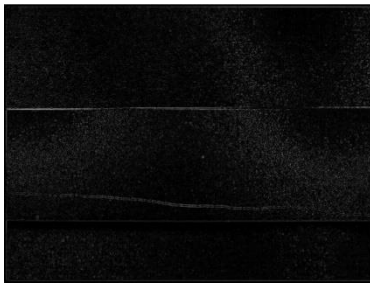

图 3.13 图像特征

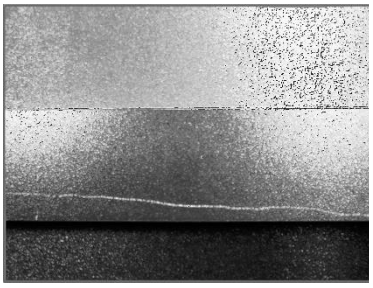

图 3.14 锐化后图像

3.1.3 数据增强

由于天池所提供的的数据样本有限，而深度学习算法需要大量的数据样本作为支持，所以在训练深度网络之前我们使用了一个小技巧。瑕疵是具有旋转不变性的，也即同样的一张瑕疵图片，通过旋转该图片我们可以得到一个对于深度学习算法而言全新的数据样本，然而无论图像如何旋转其终究是一张瑕疵图像。所以我们充分利用瑕疵的旋转不变性，通过对原数据集中的图像进行旋转，我们能够对数据集进行扩充，从而为之后深度学习算法的构建提供了数据条件。

3.2 Autoencoder + SVM

获取了增强图像之后，我们采用了 Autoencoder + SVM 的方法来对瑕疵进行检测，采用 Autoencoder 提取图像特征，采用 SVM 基于提取的特征进行瑕疵分类。由于增强后的图像仍然为 2560×1920×3 的数据矩阵，直接输入到 SVM 分类器一方面计算量庞大，不符合实际工业现场对于实时性的要求；另一方面，整张图像中的仅有较少的区域是瑕疵区域，也就意味着图片中很多输入是没有意义的，而这些冗余部分的输入不仅增加了计算量，还会降低其他代表性特征的表征程度，所以在分类器之前对特征进行提取是十分必要的。

3.2.1 Autoencoder 及其变种

Autoencoder 是一种无监督的特征选择降维方法，其广泛运用于深度学习领域。选择 Autoencoder 特征提取器的原因是其结构简单易实现、同时其能够通过网络结构提取出图像的非线性特征，是一种高效简洁的特征提取器。

Autoencoder 的结构非常特殊，如图 3.15 所示。

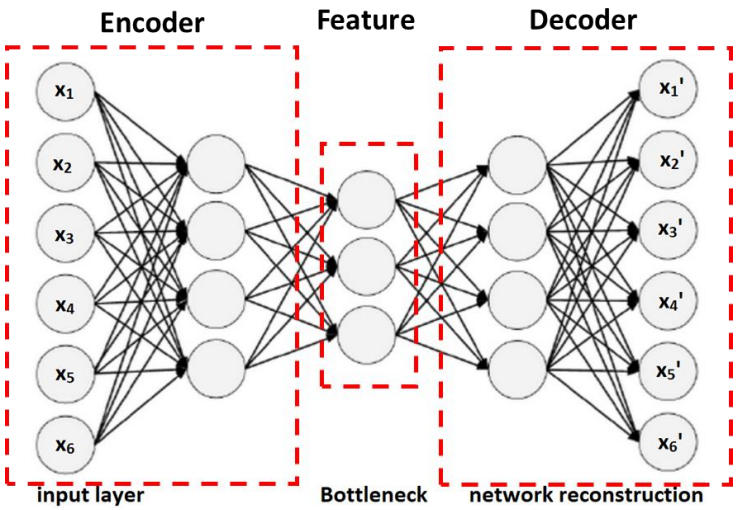

图 3.15 Autoencoder 结构示意图

其结构上可以分为输入层、中间层以及网络重构层三个部分，功能上可以分为编码器、特征层以及解码器三个部分。Autoencoder 的输入层与输出层节点个数相同，节点数由输入、输出两边向中间层递减，中间层节点个数最少[4]。如前面所述，Autoencoder 是一种无监督的特征提取方法，其仅需要输入数据而不需要数据标签。以图 3.15 为例，对于输入数据$[x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6]$与输出数据$[x_1' \quad x_2' \quad x_3' \quad x_4' \quad x_5' \quad x_6']$，网络的训练目标是希望输入与输出数据尽量匹配[4,5]。不难理解，编码器通过逐步递减的神经网络将纬度较高的输入数据编码为特征层，相应解码器依据维度较少的解码器解码出维度较高的输出数据。当输入数据与输出数据相似时，说明特征层的特征能够很好地表征输入数据中的信息[5]。通过衡量对应节点间误差，再依据反向传播算法传播误差值至整个网络，即可通过迭代更新逐步获得最佳的网络权重。

在后续的发展中，人们意识到上述的 Autoencoder 存在一些问题，故而提出了很多类似的变种。首先，因为 Autoencoder 的特征较为简单，对于高维数据（例如图像数据），一次特征提取很难提取出高度非线性的特征，并且提取出的特征一般难以很好地表征图像[6]。所以有两个解决方案，其一是将网络结构复杂化，提高网络层数以提取高度非线性特征；其二是则是 Stacked Autoencoder。通过同多个 Autoencoder 的堆叠，逐步简化特征，从而能够提取数量较少的代表性特征。Stacked Autoencoder 的结构如图 3.16 所示。

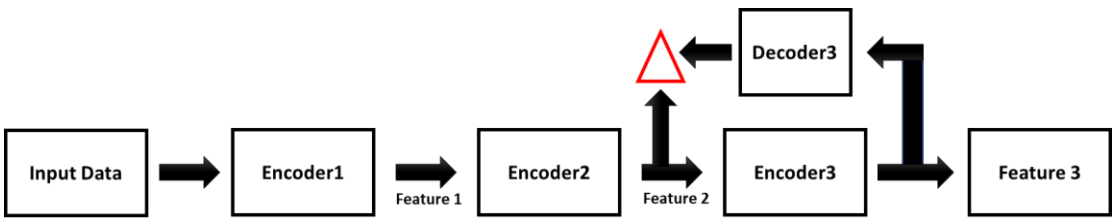

图 3.16 Stacked Autoencoder 结构示意图

Stacked Autoencoder 的好处在于其结构很能好地避免深层神经网络反向传播算法，相比于复杂的网络结构，其将一系列 Autoencoder 堆叠起来，从而能很大程度上减少反向传播算法所需耗费的大量时间与计算力[6]。以图 3.16 为例，其先依据传统 Autoencoder 的算法训练编码器 1，并提取出特征 1；之后将特征 1 所谓输入数据传输至编码器 2 中，并训练编码器 2，依次类推[6]。Stacked Autoencoder 能够通过一系列 Autoencoder 的堆叠提取出高度非线性的特征，并通过逐个 Autoencoder 减少特征数量，最终起到特征提取降维的作用。Autoencoder + SVM 方案中我们也是采用了 Stacked Autoencoder 作为特征提取器。

3.2.2 SVM

在 Autoencoder 提取出特征之后，可将特征集合与其对应的标签放入到 SVM 分类器中进行有监督训练。SVM 分类器是一种二分类算法，对于两个类别的数据，其优化目标是找到最佳的分类超平面，以使分类间隔（Margin）最大化。

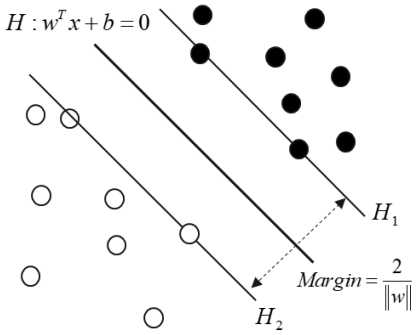

图 3.17 SVM 原理示意图

如图 3.17 所示，对于图中的两类样本（黑点与白点），SVM 的任务即是寻找出超平面 $H: w^T x + b = 0$，以使得分类间隔最大化。所谓分类间隔，即是两个类别中各自到分类超平面的 H 最小距离的和。为了更好地描述 SVM 算法，及输入图像特征及其标签为 (x_i, y_i)。最终的分类结果应如式（3.4）所示[7]。

$$y_i = \begin{cases} +1 & w \cdot x_i + b \geq 0 \\ -1 & w \cdot x_i + b \leq 0 \end{cases} \tag{3.4}$$

令 $g(x) = w \cdot x + b$ 则，空间中某点 x 到 $g(x)$ 的距离如式（3.5）所示：[7]

$$r = \frac{|w \cdot x + b|}{\|w\|} \tag{3.5}$$

假设常数 D 是最近点到判别平面 $g(x)$ 的距离，则对于所有样本点应当满足 $r \geq D$。一般规定 $D = 1$，所以可以将 SVM 求解描述为式（3.6）所示的优化问题：[7]

$$min \ \ \frac{1}{2}\|w\|^2 \tag{3.6}$$

$$s.t.\ y_i(w \cdot x_i + b) \geq 1,\ i = 1,2,\cdots,n$$

通过拉格朗日乘子法，求解过程如式（3.7）所示：[7]

$$L(w,b,a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i (y_i(w \bullet x_i + b) - 1)$$

$$\Rightarrow \begin{cases} \dfrac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \\ \dfrac{\partial L}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{3.7}$$

$$\Rightarrow \begin{cases} w = \sum_{i=1}^{n} \alpha_i y_i x_i \\ b = y_i - w \cdot x_i \end{cases}$$

传统的 SVM 对于线性可分的问题能够很好地找出对应的超平面，但是对于线性不可分问题，需要首先采用核函数将数据点映射到高维空间。通过映射到高维空间，线性不可分数据变得线性可分，从而通过高维空间的线性超平面即可解决分类问题，此处不再赘述。

3.3 InceptionV4

3.3.1 InceptionV4 网络介绍

图 3.18 展示了 InceptionV4 网络的基本结构。其主要由三种 Inception 网络模块组成，基本模块如图 3.19 所示，三个模块均是由一系列并行卷积网络组成。相比于之前的 InceptionV1~InceptionV3 网络，InceptionV4 网络加入了残差的思想，将之前的 Inception 网络模块与残差的 Resnet 模块进行结合，使得网络的训练速度与识别精度都得到了很大的提升。

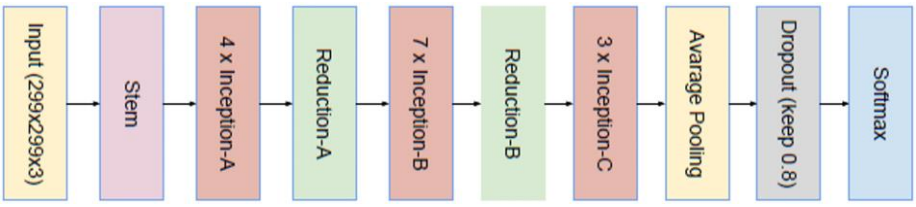

图 3.18 Inception V4 网络基本结构[8]

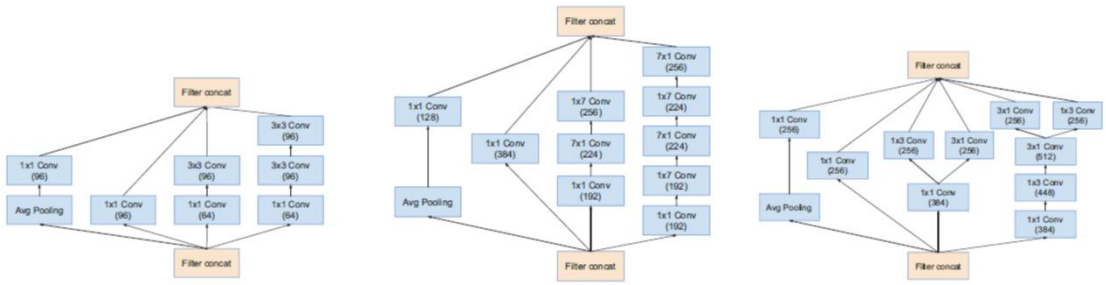

图 3.19　Inception V4 网络基本模块[8]

3.3.2 数据预处理

我们拿到的天池数据集是 Raw data，首先需要对训练数据集进行处理，从而在训练模型时能够直接使用。数据集的结构如图 3.20 所示，由于不同数据集的文件层次不同，故需要编写 python 脚本将读取子目录下的所有图片名称并保存在 CSV 文件下，并在后面加上对应的类别标签。类别标签如表 3.1 所示。

表 3.1　瑕疵类别标签汇总表

类别	编号	类别	编号	类别	编号
正常	0	不导电	1	擦花	2
横条压凹	3	桔皮	4	漏底	5
碰伤	6	起坑	7	凸粉	8
涂层开裂	9	脏点	10	其他	11

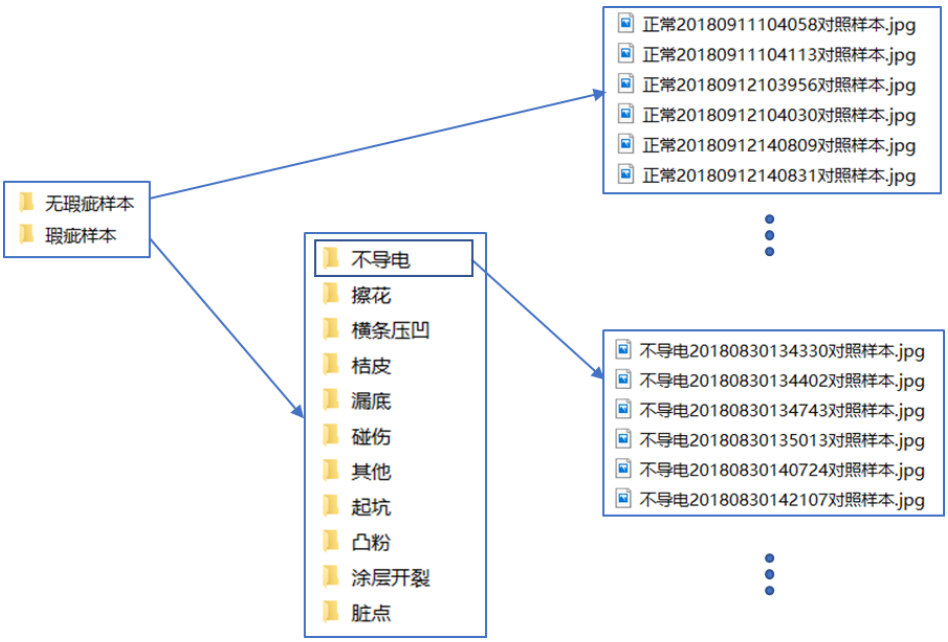

图 3.20　天池数据集原始数据文件分层示意图

在获得了处理后的 CSV 表格后，我们可以简洁地在 Python 脚本中读取图片。但是 InceptionV4 网络的图片输入尺寸要求为 299×299×3，即长宽为 299 像素，而数据集的分辨率并不满足，所以我们需要对数据集进行统一剪裁。借助 Pytorch

内置的 transform 模块，该问题能够优雅地解决。

当使用 PIL 模块将图片读入后，经过 transform 模块，就能够得到处理好的 Tensor 格式数据，Tensor 是 pytorch 框架的数据格式，能够自动求导并进行梯度下降，这样才能进行模型训练。需要注意的，因为基于 ImageNet 预训练的模型在训练的时候使用了归一化，所以我们同样需要对图像 Tensor 进行归一化，且归一化的参数（均值、方差）也要保持一样。

3.3.3 模型构建、训练与测试

Google 公司已将 InceptionV4 开源，因此能够比较容易地搭建我们的迁移学习模型。我们需要保持整个网络的前面部分不变，对最后一层全连接层神经元个数进行修改以对应 12 个分类类别。模型中包括 Stem、Inception-A-B-C、Reduction-A-B-C Pooling、Dropout、Softmax 单元，PyTorch 内置了封装好的 class，可以将其组合快速构建模型框架。

在实现了 InceptionV4 Class 之后，能够通过简单的构造函数生成一个模型，但此时模型的参数都是随机的，并不是预训练模型的参数。为此，我们需要采用 PyTorch 中提供的函数 load_state_dict 以将下载得到的模型参数装载到我们的模型中去。这样模型构建完成，因为原来的模型具有泛化特征，在我们针对铝材瑕疵数据集的训练中，能够加速收敛，节省训练时间。

模型训练首先定义交叉熵损失函数，然后使用 Adam 或者其他的算子对 Loss 函数进行梯度下降。训练初期 Loss 下降快，后期下降慢。为了更好地提升模型精度，在程序中根据训练的阶段动态调整学习率。随着训练的推进，逐渐减小学习率以在保证训练速度的前提下，让参数能够更快地收敛。

训练过程将经过 20 个 Epoch，每个 Epoch 完成后进行一次模型评测。根据准确度更新当前效果最好的模型，最后在应用的时候将使用这个 Best_model。模型训练环境表 3.2 所示，训练过程如图 3.21 所示，训练结果如图 3.22 所示。训练共耗时 3 小时。

表 3.2 模型训练硬件环境

CPU	AMD ryzen7 1700 eight cores
GPU	GTX1060 3g
RAM	2400MHZ 16g

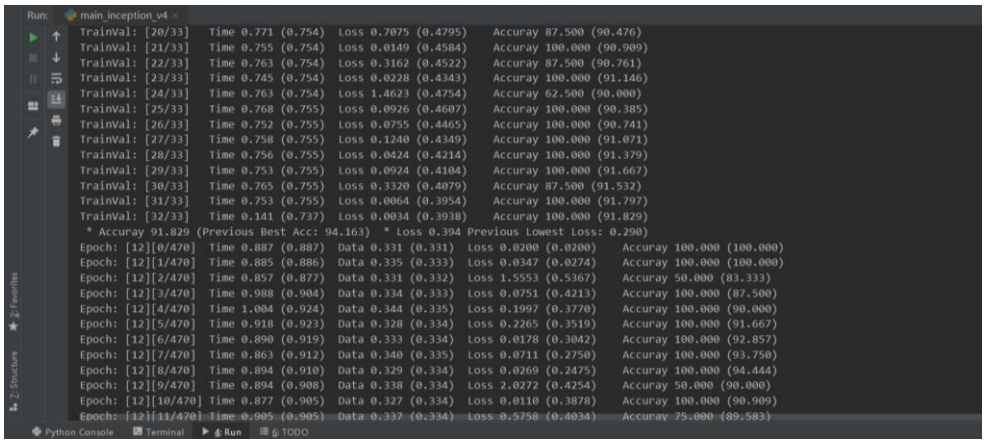

图 3.21　模型训练过程

```
Epoch:  0, Precision: 82.49027237, Loss: 0.60255025
Epoch:  1, Precision: 88.32684825, Loss: 0.49898946
Epoch:  2, Precision: 89.10505837, Loss: 0.48504822
Epoch:  3, Precision: 90.27237354, Loss: 0.39461614
Epoch:  4, Precision: 92.99610895, Loss: 0.33890654
Epoch:  5, Precision: 89.49416342, Loss: 0.42534446
Epoch:  6, Precision: 91.05058366, Loss: 0.33393112
Epoch:  7, Precision: 90.27237354, Loss: 0.39173879
Epoch:  8, Precision: 94.16342412, Loss: 0.28958592
Epoch:  9, Precision: 91.43968872, Loss: 0.45708952
Epoch: 10, Precision: 91.43968872, Loss: 0.33743074
```

图 3.22　模型训练结果

从训练结果可以发现，训练过程准确度并不是一直上升的，有一定的随机性。之后我们将训练好的保存为 pth 文件，方便之后调用训练好的模型。由于训练过程使用到了 GPU 和 CUDA 加速，而在之后的和机械臂的配合测试使用的笔记本电脑不具备该硬件条件，因此需要将在 GPU 上训练的模型保存为 CPU 能够读取的。该步骤完成后，只需要安装了 Python 和 PyTorch 的电脑可以直接读取我们的模型，并输入图像进行预测，输出对应的分类结果。

为了方便调用，编写了 python 脚本将我们的模型封装进函数，之后对图像进行预测只需要调用这个 API，减少最后集成机械臂实验部分的麻烦。在 Windows 命令行中使用该接口，即可对待检测图像进行瑕疵检测，示意图如图 3.23 所示。

```
E:\PycharmProjects\xiaci>test.py C:\Users\94955\Desktop\脏点2.jpg
有瑕疵
class:脏点, prob:95.80

E:\PycharmProjects\xiaci>test.py C:\Users\94955\Desktop\正常1.jpg
无瑕疵
class:正常, prob:90.35
```

图 3.23　模型检测示意图

4 机械臂组

4.1 Dobot 机械臂运动规划和仿真

机械臂是分拣系统中的执行机构，其运动控制的研究十分重要，而机械臂的运动学分析则是轨迹规划的重要基础。机械臂运动学中的位置研究包括正运动学分析和逆运动学求解，机械臂正运动学分析指的是根据各关节角的大小确定机械臂末端在笛卡尔空间的位姿表示，逆运动学求解则是通过某时刻机械臂末端的笛卡尔空间位姿去求解该时刻各关节角的大小[9]。

轨迹规划的任务是在给定机械臂末端起始位姿和终止位姿的条件下，确定机械臂末端到达目标位姿的准确路径、时间历程、速度曲线等。机械臂的轨迹规划方法可分为关节空间的规划方法和笛卡尔空间的规划方法。

正逆运动学分析是轨迹规划的基础，而建模和建系又是一切分析的前提。本小节包括以下内容：1) Dobot 机械臂建模；2) 正运动学分析；3) 逆运动学求解；4) 关节空间的轨迹规划；5) 笛卡尔空间的轨迹规划；6) MATLAB 仿真分析。

4.1.1 坐标系建立

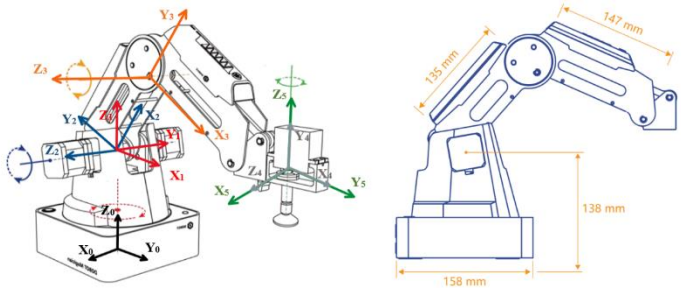

图 4.1 Dobot 机械臂空间尺寸示意图

建立图 4.1 所示坐标系，结合 Dobot Magician 机械臂对应尺寸参数，可得到其 DH 参数表如表 4.1 所示。

表 4.1 Dobot 机械臂的 DH 参数

	连杆转角α_{i-1}	连杆长度a_{i-1}	连杆偏距d_i	关节角θ_i
1	0	0	$d_1 = 138$	θ_1
2	90°	0	0	θ_2
3	0	$a_2 = 135$	0	θ_3
4	0	$a_3 = 147$	0	$\theta_4 = -\theta_3 - \theta_2$
5	−90°	0	0	θ_5

其中，$\theta_1 \in [-90°, 90°]$, $\theta_2 \in [0°, 85°]$, $\theta_3 \in [-90°, 10°]$, $\theta_5 \in [-135°, 135°]$。需要特别说明的一点是，第四个关节并不是真正的关节，而是被动关节，它保证了最后一个关节轴的方向不会因为大臂和小臂的旋转而发生改变，但关节 4 本身不是驱动结构，因此不是独立的关节变量，它的关节角由关节 3 和关节 4 的关节变量共同决定。

4.1.2 正运动学求解

根据 ${}_{i}^{i-1}T = \begin{bmatrix} cos\theta_i & -sin\theta_i & 0 & a_{i-1} \\ sin\theta_i cos\alpha_{i-1} & cos\theta_i cos\alpha_{i-1} & -sin\alpha_{i-1} & -sin\alpha_{i-1}d_i \\ sin\theta_i sin\alpha_{i-1} & cos\theta_i sin\alpha_{i-1} & cos\alpha_{i-1} & cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$，代入 DH 参数表后可逐一求得每对相邻坐标系的齐次变换矩阵，如式（4.1）所示：

$$
{}_{1}^{0}T(\theta_1) = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 & 0 \\ sin\theta_1 & cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}_{2}^{1}T(\theta_2) = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ sin\theta_2 & cos\theta_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},
$$

$$
{}_{3}^{2}T(\theta_3) = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & a_2 \\ sin\theta_3 & cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}_{4}^{3}T(\theta_4) = \begin{bmatrix} cos\theta_4 & -sin\theta_4 & 0 & a_3 \\ sin\theta_4 & cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (4.1)
$$

$$
{}_{5}^{4}T(\theta_5) = \begin{bmatrix} cos\theta_5 & -sin\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -sin\theta_5 & -cos\theta_5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\qquad {}_{6}^{5}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

其中 ${}_{6}^{5}T$ 表示末端执行器尖端在坐标系{5}中的位姿描述，而$\theta_4 = -(\theta_2 + \theta_3)$，因此有式（4.2）

$$
\begin{aligned}
{}_{4}^{3}T(\theta_4) &= \begin{bmatrix} cos\theta_4 & -sin\theta_4 & 0 & a_3 \\ sin\theta_4 & cos\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} \cos(\theta_2+\theta_3) & \sin(\theta_2+\theta_3) & 0 & a_3 \\ -\sin(\theta_2+\theta_3) & \cos(\theta_2+\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (4.2) \\
&= {}_{4}^{3}T(\theta_2, \theta_3)
\end{aligned}
$$

为了简化表达，将$cos\theta_i$记为c_i，$cos(\theta_i + \theta_j)$记为c_{ij}，同理，s_i表示$sin\theta_i$，s_{ij}表示$sin(\theta_i + \theta_j)$。则

$$
{}_{6}^{0}T = {}_{1}^{0}T(\theta_1){}_{2}^{1}T(\theta_2){}_{3}^{2}T(\theta_3){}_{4}^{3}T(\theta_2, \theta_3){}_{5}^{4}T(\theta_5){}_{6}^{5}T
$$

$$= \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

其中，

$$\begin{cases} r_{11} = c_5c_1 - s_5s_1 = c_{15}, \\ r_{12} = -c_5s_1 - s_5c_1 = -s_{15}, \\ r_{13} = 0, \\ r_{21} = c_1s_5 + c_5s_1 = s_{15}, \\ r_{22} = c_1c_5 - s_5s_1 = c_{15}, \\ r_{23} = 0, \\ r_{31} = 0, \\ r_{32} = 0, \\ r_{33} = 1, \\ x = c_1(a_3c_{23} + a_2c_2), \\ y = s_1(a_3c_{23} + a_2c_2), \\ z = a_3s_{23} + a_2s_2 + d_1 - d. \end{cases} \tag{4.4}$$

将旋转矩阵用 X-Y-Z 固定角表示，令

$$\begin{aligned} {}^0_6R_{XYZ}(\gamma,\beta,\alpha) &= R_Z(\alpha)R_Y(\beta)R_X(\gamma) \\ &= \begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix} \\ &= \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \end{aligned} \tag{4.5}$$

解得

$$\begin{cases} \beta = atan2(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}) \\ \alpha = atan2(r_{21}/c\beta, r_{11}/c\beta) \\ \gamma = atan2(r_{32}/c\beta, r_{33}/c\beta) \end{cases} \tag{4.6}$$

将旋转矩阵转化为固定角表示有助于之后在笛卡尔空间内的运动规划求解。

4.1.3 逆运动学求解

对于本项目所使用的 Dobot 机械臂而言，由于各关节变量都有边界约束，逆运动学的解的存在情况有三种，即无解、有唯一解和有多解。

由于 Dobot 机械臂最后两个坐标系的原点重叠，且末端执行器尖端的位置确定时，可以很快得到坐标系原点重叠处的位置，可以根据该位置求解出前三个关节变量的值，即θ_1、θ_2和θ_3，采用 PIEPER 解法求解逆运动学[10]，具体解法如式（4.7），假设已知

$$
\begin{aligned}
{}_6^0T &= {}_1^0T(\theta_1){}_2^1T(\theta_2){}_3^2T(\theta_3){}_4^3T(\theta_2,\theta_3){}_5^4T(\theta_5){}_6^5T \\
&= \begin{bmatrix} r_{11} & r_{12} & r_{13} & x \\ r_{21} & r_{22} & r_{23} & y \\ r_{31} & r_{32} & r_{33} & z \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{4.7}
$$

则

$$
\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z+d \\ 1 \end{bmatrix} = \begin{bmatrix} {}^0P_{4ORG} \\ 1 \end{bmatrix} = {}_1^0T \cdot {}_2^1T \cdot {}_3^2T \begin{bmatrix} {}^3P_{4ORG} \\ 1 \end{bmatrix}
$$

$$
\begin{bmatrix} {}^2P_{4ORG} \\ 1 \end{bmatrix} = {}_3^2T \begin{bmatrix} {}^3P_{4ORG} \\ 1 \end{bmatrix} = \begin{bmatrix} c_3 & -s_3 & 0 & a_2 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_3 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_3c_3 + a_2 \\ a_3s_3 \\ 0 \\ 1 \end{bmatrix}
$$

$$
\begin{bmatrix} {}^1P_{4ORG} \\ 1 \end{bmatrix} = {}_2^1T \begin{bmatrix} {}^2P_{4ORG} \\ 1 \end{bmatrix} = \begin{bmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_3c_3 + a_2 \\ a_3s_3 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_3c_{23} + a_2c_2 \\ 0 \\ a_3s_{23} + a_2s_2 \\ 1 \end{bmatrix} \tag{4.8}
$$

$$
\begin{bmatrix} {}^0P_{4ORG} \\ 1 \end{bmatrix} = {}_1^0T \begin{bmatrix} {}^1P_{4ORG} \\ 1 \end{bmatrix} = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_3c_{23} + a_2c_2 \\ 0 \\ a_3s_{23} + a_2s_2 \\ 1 \end{bmatrix}
$$

$$
= \begin{bmatrix} c_1(a_3c_{23} + a_2c_2) \\ s_1(a_3c_{23} + a_2c_2) \\ a_3s_{23} + a_2s_2 + d_1 \\ 1 \end{bmatrix} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}
$$

则

$$
p_x^2 + p_y^2 + (p_z - d_1)^2 = a_2^2 + a_3^2 + 2a_2a_3cos\theta_3 \tag{4.9}
$$

$$
\theta_3 = -arccos\left(\frac{p_x^2 + p_y^2 + (p_z - d_1)^2 - a_2^2 - a_3^2}{2a_2a_3}\right) \tag{4.10}
$$

其中，$arccos(\cdot)$函数的值域是$[0, 180°]$，而由前面可知，逆运动学的解可能有多个，而$\theta_3 \in [-90°, 10°]$，也就是说，小臂的关节角θ_3在小于 0 的区间内有更大的活动空间，考虑到实际应用情况，在存在多个解时，我们更希望得到小臂的关节角处于负区间的解，因此取负值。如果这种做法会导致后面θ_2超出其范围，再尝试将θ_3取正值，若这样做后θ_2在$[0, 85°]$内有解，说明此时逆运动学的解唯一，若θ_2仍无解，说明对于给定的变换矩阵，不存在逆运动学的解。

$$
\begin{aligned}
a_3s_{23} + a_2s_2 &= (a_3c_3 + a_2)s_2 + a_3s_3c_2 \\
&= \sqrt{a_2^2 + a_3^2 + 2a_2a_3c_3}sin(\theta_2 + \varphi) = p_z - d_1 \\
\varphi &= atan2(a_3s_3, a_3c_3 + a_2)
\end{aligned}
$$

$$\theta_2 = arcsin\left(\frac{p_z - d_1}{\sqrt{a_2^2 + a_3^2 + 2a_2a_3c_3}}\right) - \varphi \tag{4.11}$$

而

$$\theta_1 = atan2(p_y, p_x) \tag{4.12}$$

确定θ_1、θ_2和θ_3后，很快可以得到

$$\theta_4 = -(\theta_2 + \theta_3) \tag{4.13}$$

对于姿态描述，若是采用旋转矩阵描述，可以根据正运动学求解部分最后的公式转化为固定角 XYZ 表示，由于 Dobot 机械臂的机械结构约束，末端坐标系的 Z 轴正向总是垂直地面向上，与基坐标系的 Z 轴平行，也就是说，末端坐标系的姿态可以通过基坐标系绕 Z 轴旋转一定角度得到，因此固定角 XYZ 表示形如(γ, β, α)，其中$\gamma = 0, \beta = 0$，对应的旋转矩阵形式是

$${}_6^0R_{XYZ}(\gamma, \beta, \alpha) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

而由正运动学的求解结果知：

$${}_6^0R = \begin{bmatrix} c_{15} & -s_{15} & 0 \\ s_{15} & c_{15} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

通过比较可知，

$$\theta_5 = \alpha - \theta_1 \tag{4.16}$$

4.1.4 基于关节空间的运动规划

关节空间规划方法，就是用关节角的函数来描述轨迹的运动规划方法。对于各关节而言，能够直接控制的是力矩大小，据此可以根据动力学确定关节角的角加速度，反过来说，知道了关节角的角加速度，也就能确定控制信号。关节角的角加速度也就是关节角的二阶导数，因此确定关节角函数，也就是能确定关节角的角加速度，进一步确定控制信号。由于本项目使用的魔术师机械臂的连杆质量、转动惯量等关键参数无法得知，因此这里只讨论如何确定关节角函数。在关节空间中，我们可以直接对关节角变量进行规划，考虑到角度、角速度的连续性，通常采用多项式插值的方法确定各关节角的函数，用三次多项式便可使加速度函数连续。

由前文可知，通过正运动学分析，已知各关节变量的值即可确定机械臂末端位姿；通过逆运动学求解，已知机械臂末端位姿也可反解出各关节变量的值。因此，无论初始条件和终值条件是以何种形式给定，我们最终都可以用关节角变量（也就是在关节空间中）进行表示。另一方面，各关节角变量都是独立的，因此

可以分别进行规划，而且在规划中通常要求所有关节同时到达每个路径点对应的关节角度[10,11][9,10][8,9][8]。

不失一般性，我们可以对第i个关节变量θ_i进行规划，确定其关于时间t的函数$\theta_i(t)$。假设

$$\theta_i(t) = a_3t^3 + a_2t^2 + a_1t + a_0, \dot{\theta}_i(t) = 3a_3t^2 + 2a_2t + a_1$$

$$\text{s.t.}\begin{cases} \theta_i(0) = \theta_{i0}, \\ \theta_i(t_f) = \theta_{if}, \\ \dot{\theta}_i(0) = 0\ , \\ \dot{\theta}_i(t_f) = 0\ . \end{cases} \tag{4.17}$$

解得

$$a_0 = \theta_{i0}, a_1 = 0, a_2 = \frac{3(\theta_{if}-\theta_{i0})}{t_f^2}, a_3 = -\frac{2(\theta_{if}-\theta_{i0})}{t_f^3} \tag{4.18}$$

至此，一旦确定初始条件、终值条件和运动时间t_f，便可确定关节角函数。

4.1.5 基于笛卡尔空间的运动规划

采用关节空间的运动规划时，关节角的变化曲线更加平滑，但机械臂的运动轨迹一般是不规则的。基于笛卡尔空间的运动规划，首先采用规则的曲线（如直线、圆弧等）进行插值确定机械臂末端的运动轨迹，按一定的步长离散化得到中间位姿，再通过逆运动学求解出相应的关节角。

值得一提的是，机械臂的末端位姿通常采用从基坐标系到末端坐标系的齐次变换矩阵表示，位置信息可以用三维坐标(x, y, z)表示，三个变量是独立的。以直线插值为例，中间位置坐标可以直接通过线性插值得到。而对于中间姿态，无法直接对旋转矩阵进行插值，因此需要转化为固定角（通常采用固定角 XYZ）表示，具体见正运动学求解的最后一部分内容。对于固定角便可通过线性插值，得到中间姿态。

对于 Dobot 机械臂，由于θ_4的引入，末端坐标系的 Z 轴始终正向朝上，与基坐标系的 Z 轴同向，也就是说，末端坐标系的姿态可以通过基坐标系绕 Z 轴旋转α得到。因此，机械臂末端姿态在笛卡尔空间中可以用$p = (x, y, z, \alpha)^T$的四维向量表示，四个变量分别独立。

以沿直线运动的笛卡尔坐标运动规划为例，不失一般性，假设初始位姿为p_0，终止位姿为p_f，运动时长为t_f，则

$$p(t) = p_0 + \frac{t}{t_f} \cdot (p_f - p_0), t \in [0, t_f] \tag{4.19}$$

再通过逆运动学求解，由$p(t)$可以得到 $[\theta_1(t), \theta_2(t), \theta_3(t), \theta_5(t)]$，也就是各关节角的变化轨迹。

4.2 Dobot 机械臂实物测试与 Python API 开发

本项目使用的机械臂——Dobot 魔术师是一种桌面级机械臂，具有多功能，高精度，易控制，安全方便等特点，适合本项目中智能分拣的工作场景。硬件方面，该机械臂末端执行器可以根据实际需要进行切换，例如可以切换吸盘和手爪，能够很好地适应不同的工作任务；软件方面，该机械臂为二次开放提供了多种接口，可以使用多种语言进行编程例如 Python，C#，Java，也可以在不同平台下进行控制，例如 Windows，Mac，Linux，为机械臂模块与视觉模块的整合与对接提供便利。

4.2.1 机械臂物理结构

Dobot Magician 由底座、大臂、小臂、末端工具等组成，整体外观如图所示。该机械臂具有四个自由度，四个舵机分别分布在底座，大臂，小臂以及末端工具处，可以实现六个自由度的运动，工作空间较大。坐标系的建立支持关节坐标系与笛卡尔坐标系，在安装吸盘的情况下，笛卡尔坐标系包含四个关节 J1,J2,J3,J4,均为旋转关节，逆时针方向为正；笛卡尔坐标系以机械臂底座为参照，X 轴方向垂直于固定底座向前，Y 轴方向垂直于固定底座向左，Z 轴方向符合右手定则，垂直向上为正方向，R 轴为末端舵机中心相对于原点的姿态，逆时针为正。

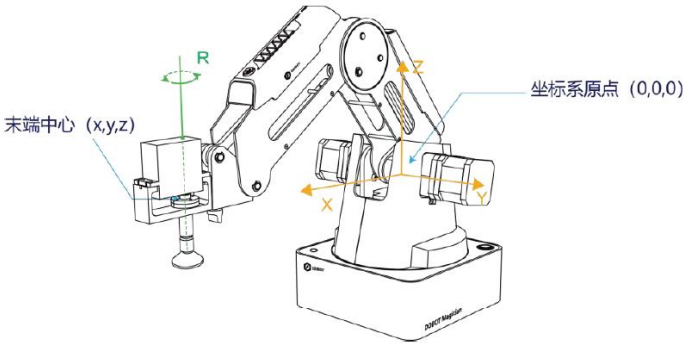

图 4.10 笛卡尔坐标系

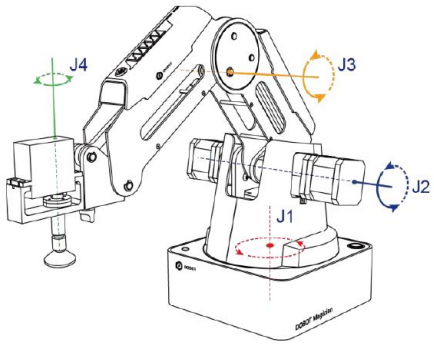

图 4.11 关节坐标系

末端工具有吸盘和手爪提供选择，然而手爪由于大小的限制，只能抓特定规格的物体。为了增强项目的适用性，本项目使用吸盘作为末端工具，吸盘的工作状态由气泵盒进行控制。机械臂整体的线缆连接如下：

1. 机械臂通过 USB 线缆与电脑连接，受电脑控制。
2. 机械臂通过电源适配器进行供电。
3. 气泵盒的信号线与电源线同机械臂的底座相连。
4. 气泵盒的气管与吸盘的气管接头相连，实现供气。

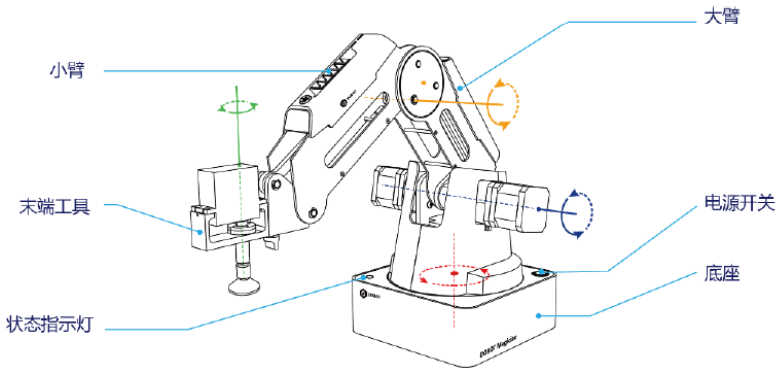

图 4.12 机械臂搭建示意图

4.2.2 Dobot 平台直接操作与仿真对比

在组装完成之后，可以通过提供的软件 Dobot Studio 对机械臂进行手动控制，完成简单的分拣任务。DobotStudio 平台提供了多种功能，包括示教与再现，脚本控制，鼠标控制等模块。其中示教与再现功能是一种先由人工引导机器人末端执行器运动，记忆中间点后，再由机器人不断重复运动的功能，通常用于不需要精确路径控制的场合,可以降低运算的时间成本，精简程序结构。

图 4.13 Dobot Magician 直接控制界面

Dobot Studio 中的运动控制，包括点动模式，点位模式，圆弧运动模式。点动模式，可以通过笛卡尔坐标下，改变末端 XYZR 的姿态；也可在关节坐标系下，直接控制电机的旋转角度，实现机械臂运动的控制。点位模式与圆弧模式通过存点进行轨迹的控制，其中点位模式的轨迹控制分三种：直线轨迹 MOVL，门型轨迹 HEIGHT 以及关节空间内的直线轨迹 MOVJ。在笛卡尔空间下进行轨迹控制，可以直接控制三维空间内的运动轨迹，但容易出现奇异解的情况；而关节空间内的轨迹控制，虽然一定有解，但三维空间下的轨迹往往比较杂乱[12]。

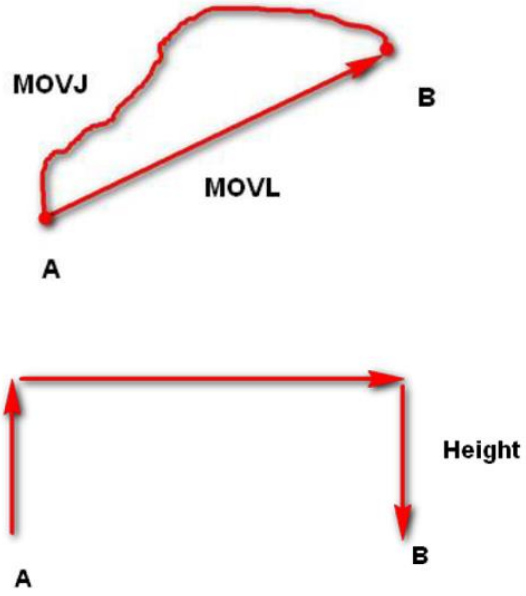

图 4.14 Dobot 路径规划方式对比图

由于本项目的应用场景为工厂内的智能分拣，因而任务实现的场景相对较为固定，为了提高安全性与准确性，避免意外的发生，可以先通过示教的方式定下中间点，采取直线轨迹的点位控制方式，完成运动轨迹的规划。而通过仿真模型规划得到的轨迹也可以采用再现的方式，放入中间点，通过点位控制，观察机械臂实际运动情况。

实际使用过程中，在场景构建完成后，先通过人工引导的方式选取中间点，在两个时间上连续的点之间采取点位控制，完成分拣工作。由于 DobotStudio 平台的运动控制采用点位控制，即两点之间的初速度和末速度为 0，不可更改，末端执行器在到达每一个中间点后会产生一个停顿，因此中间点的数量越多，末端执行器的运动效率也会随之降低。

在 MATLAB 模型与实物对比中可以发现，选取了合适的规划方式与坐标系之后，仿真模型与实物在相同动作下有比较高的一致性。仿真系统开发过程中遇到的特定变量如角加速度缺乏等问题，我们通过采取差分量等方法得到了解决，从而在 Dobot Studio 下得到了仿真系统与实物系统近似的表现。

4.2.3 Python API 开发

我们在 Dobot Studio 层面进行机械臂仿真与实物机械臂的对比，虽然能够满足机械臂路径规划等环节的探索，但是由于 Dobot Studio 较为封闭，无法与视觉识别环节，摄像头检测环节结合，因此需要整合在统一的代码框架下。为此，我们在 Python 上重新开发了 API，参考的 Dobot 通讯架构如下：

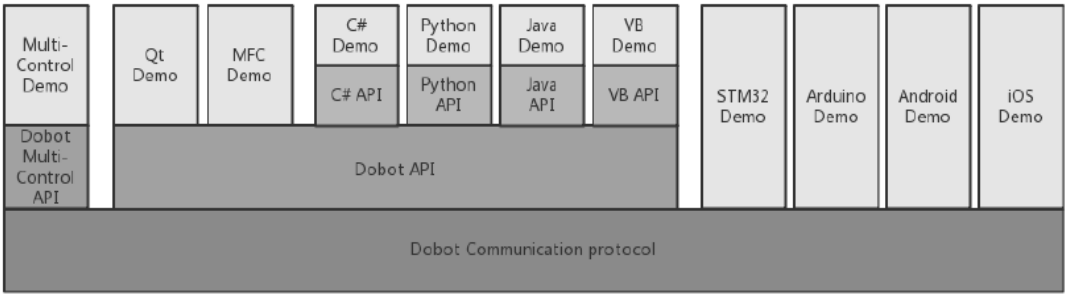

图 4.15 Dobot 机械臂通信架构示意图

可以看到，通过 Python 脚本，我们能够与机械臂进行直接通信，数据的传递和代码整合也将更为方便。该机械臂封装了底层的通讯协议，不需要再关注通讯控制工作，开发者可以专注于上层控制，包括路径控制，速度控制等。为了完成分拣任务，需要重点关注机械臂的运动部分，即每个动作的起点，终点以及运动速度，不同动作的时序关系等。

我们机械臂分拣 Python 项目建立在 Dobot 提供的通信规范基础上，项目结构如下:

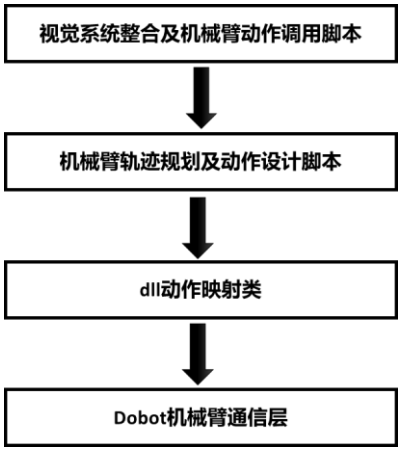

图 4.16 Python API 工程结构示意图

Dobot Magician 提供了结构比较清晰的通信协议文档和 DLL 文档，大大方便了我们Python接口项目的开发，也免去了在通信层进行繁琐的底层试错与调试。在 DLL 动作映射类文件上，我们按照 Dobot 提供的规范定义通信中用到的结构体和各类基础函数，由路径规划和动作设计脚本进行机械臂复合动作的控制，最

后由视觉整合脚本将实物机械臂与视觉部分的项目结合起来。在整个过程中，Python 语言友好的项目整合能力给我们的开发与调试带来了很多帮助。

一个典型的 Python 驱动机械臂 API 分拣动作如下所示：

图 4.17 典型分拣过程示例

在实物机械臂的开发过程中，Dobot 过度封装的问题还是给我们带来了一定的困难。为了避让摄像头，我们规划机械臂运动时选取的姿态通常比较极端，而 Dobot 虽然原则上在其状态空间里都可以实现点对点运动，但是在极端动作下求解还是会出现一些奇异性问题。很多姿态变化虽然在它的状态空间内，但是不能求解出一条合适的路径，会出现卡死并报错等问题，无法完成我们的分拣需求。

为了解决这一主要问题，我们采取的路线主要有两种：（1）进行更加精细的路径规划，在进行极端避让动作前选取尽可能友好的路线。（2）从 DLL 库中引入更多的底层函数，进行更细致的机械臂反馈信息处理，在故障发生之后能够通过归位，局部位移等方法进行自主排障，从而提高系统的容错性。

4.3 京航工业摄像机

本项目使用的京航工业摄像机部分的开发较简单，京航工业提供了比较完善的项目结构示例，主要通过调用 OpenCV 框架进行图形拍摄和处理工作，开发过程较为顺利。

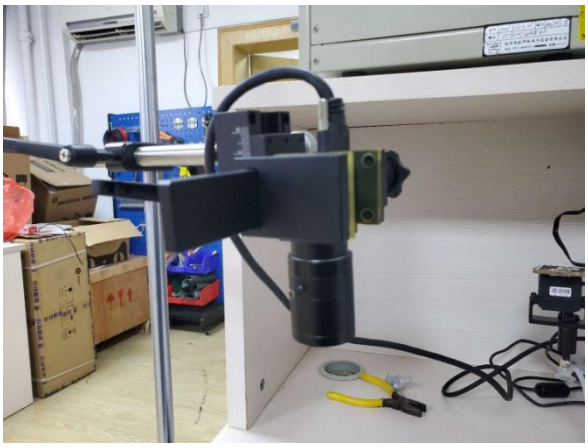

图 4.18 摄像头硬件

4.3.1 问题与解决思路分析

在摄像头部分的开发过程中，主要的问题是成像质量导致实物识别率下降。我们将仿真系统中使用的数据集打印后制作为工件物块并在视觉模块下拍照，这个过程中的二次成像导致了细节特征的进一步丢失。

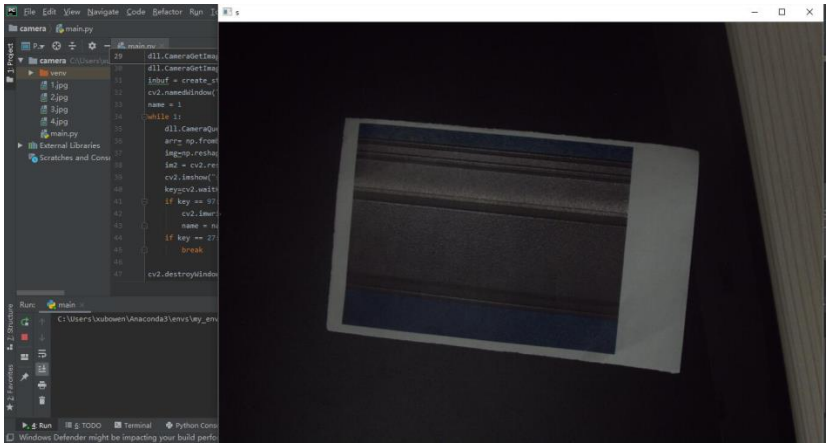

图 4.19 摄像头检测结果

在这个问题上，主要有两种解决思路：其一是换用感光元件更大的工业相机，加入外部光照改善曝光问题，使噪点数下降，信息丢失减少；其二是从软件层面解决目前的问题，直接使用工业相机数据构建训练集和测试集，从而避免我们开发路线中的场景迁移问题。

我们小组成功实现了实物机械臂与工业摄像机的整合系统，但是成像质量导致的劣化问题由于进度来不及解决，有待进一步优化。

5 上位机界面

5.1 瑕疵检测上位机界面

为了将机械臂控制、工业摄像头以及瑕疵检测进行整合并实现在线仿真，我们基于 Python3 编程环境使用 PyQt5 库编写上位机软件。该软件能够很方便地对瑕疵图片进行预测，并直观地显示结果。软件左上红框标出的为图片区域，软件支持拖拽加载、文件选择和工业摄像机拍摄三种图片加载方式。右边第一个文本框为当前图片的文件名，拖拽方式将图片直接拖入界面即可，点击“载入图片”按钮，将弹出文件选择框以供精确定位图片，点击“拍摄图片”按钮将调用摄像机 API 进行拍照并导入软件。软件界面如图 5.1 所示，软件框架构建如图 5.2 所示。

图 5.1 瑕疵检测上位机界面示意图

函数名	函数功能
dragEnterEvent	拖入图片以读取
dropEvent	释放鼠标
readImage	读入图像
loadimg	将图像显示到上位机界面
takephoto	摄像头采集图像
closephoto	关闭当前图像
loadmodel	载入瑕疵检测模型更
classify	对当前图像进行分类
action1	机械臂动作 1（连接）
action2	机械臂动作 2（复位）
action3	机械臂动作 3（左移）
action4	机械臂动作 4（右移）

图 5.2 瑕疵检测上位机程序架构

加载图片后需要加载训练好的神经网络模型，点击“载入模型”将会。点击“分类检测”按钮则将当前图片交给加载的模型进行分类预测，结果显示在下面三个

文本框中。第一个“瑕疵”文本框只有两个可能值，”有瑕疵“和”无瑕疵“，第二个”类别“文本框对应一共 12 个类别，会显示瑕疵的具体类别，如碰伤、擦花等，第三个”概率“文本框表示当前预测类别的概率，概率越高则可信度越高。需要注意，进行分类检测需要载入模型和图片，缺失任何一种会弹出警告窗口。

软件下半部分为魔术师机械臂控制按钮，预设了四种动作。在实验中使用前两个动作，对应我们有瑕疵和无瑕疵两个类别。点击按钮将控制机械臂进行对应的动作。组员基于 MATLAB 对机械臂成功进行仿真实验，由于对机械臂还不是很熟悉，未能获取机械臂的实施运动参数，且做到实时跟踪难度较大，最后放弃在软件里加入机械臂仿真的模块。

在实际测试中，软件易使用，预测响应速度较快。但与预料的结果一样，使用工业摄像头拍摄的打印图片效果不佳，和数据集的照片差距较大，从而导致分类效果较差。

5.2 机械臂仿真界面

Robotics Toolbox 是一款基于 MATLAB 编程平台的机器人仿真工具箱，其提供了可用于研究与模拟经典臂型机器人的功能。工具箱包含了可用于在二维或三维空间中机器人仿真的函数与库，可通过矩阵、四元数、矩阵指数等方式来表示机器人的位置与姿态。工具箱还提供了在数据类型之间进行操作和转换的函数，以便于在三维位置和方向不同表示法下的转换。对于机械臂，Robotics Toolbox 采用较通用的方法来表示串行链路机械臂的运动学和动力学[13]。依据机械臂相应的 DH 参数表，规定相应关节的类型，即可建立相应的关节 LINK，从而实现机械臂的仿真。

在本项目中，我们不仅用 MATLAB 实现 Dobot 机械臂的轨迹规划，还搭建了仿真平台，并采用图形化用户接口。整个仿真系统包括 7 个.m 文件，其中 DobotManipulator.m 是主程序，运行该文件便可进入仿真平台。具体如下：

首先是输入模式的选择，这里提供两种模式，分别对应机械臂位姿在关节空间内的表示方法和在笛卡尔空间内的表示方法。正如前面多次强调的，对于 Dobot 机械臂，θ_4并不是独立变量，可以由θ_2和θ_3确定，因此实际的关节角自由变量只有 4 个。而机械臂末端位姿在笛卡尔空间的表示方法，除了描述位置的(x, y, z)，还有描述姿态的绕 Z 轴旋转的角度α。

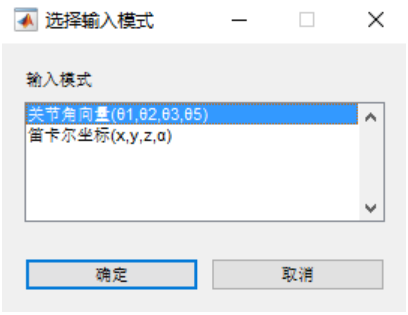

图 5.3 MATLAB 坐标系选择

选择输入模式之后，根据用户选择的结果，程序将跳转到不同的界面，分别如下：

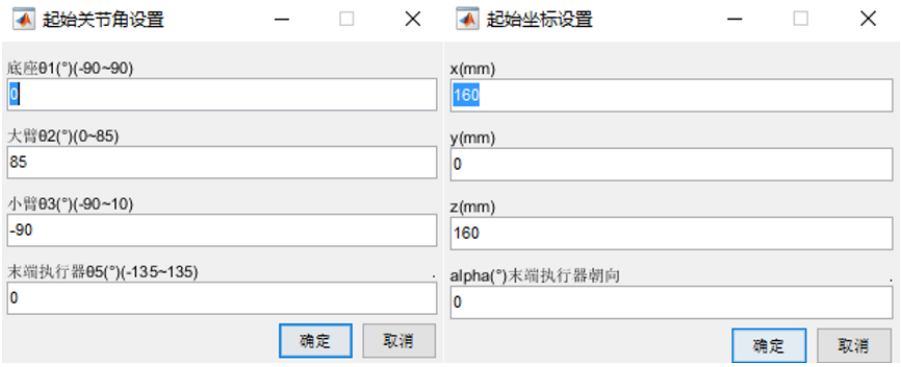

图 5.4 MATLAB 坐标系配置

输入初始和终止条件后，还需要选择运动规划的模式，本仿真平台共提供了四种运动规划模式，分别是在关节空间内的规划、笛卡尔空间内的直线运动规划、笛卡尔空间内的门型运动规划、示教模式。示教模式指的是，除了起始和终止位姿信息，用户还可以输入多个中间点的位姿信息，确定机械臂末端轨迹的轮廓，在后续规划中，机械臂末端执行器会陆续经过用户输入的中间点位姿，最后到达目标位置和姿态。

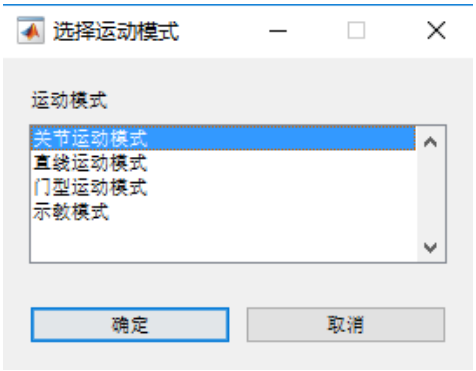

图 5.5 MATLAB 运动模式配置

选择不同运动模式，用户可能需要输入额外的信息，例如门型运动模式中，用户需要输入机械臂抬升的高度，示教模式中用户需要输入中间点的信息。完成

所有输入后，程序开始进行规划，并输出机械臂运动示意图以及各关节变量的变化曲线。如下：

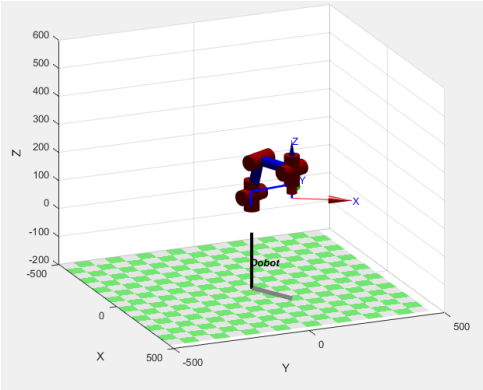

图 5.6 机械臂运动示意图

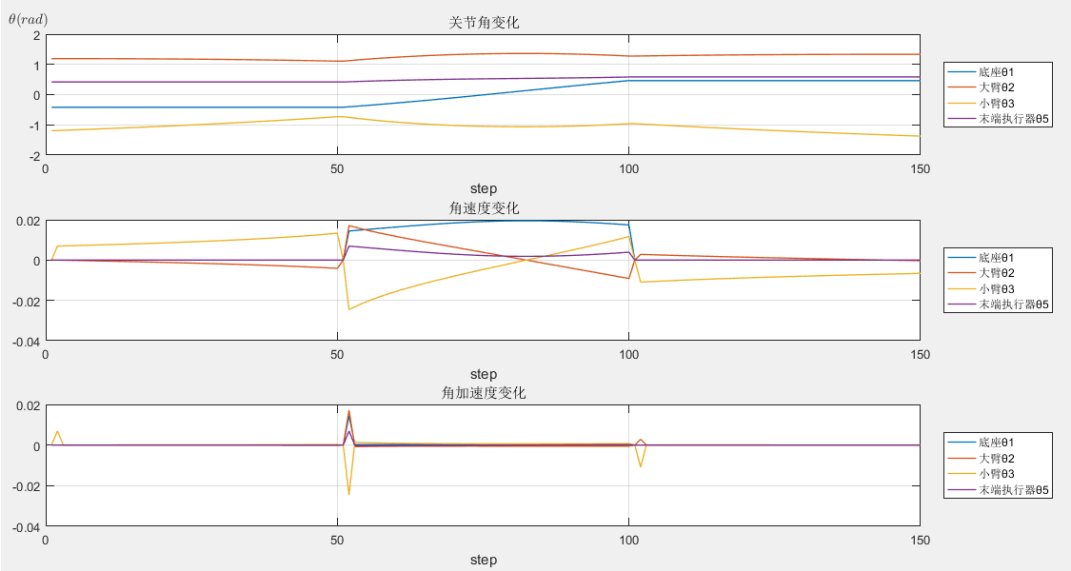

图 5.7 机械臂关节变量曲线

此外，当用户输入的末端执行器的位置没有落在工作空间内，也就是说逆运动学的解不存在时，程序会予以警告；当规划的轨迹会超出工作空间时，程序也会予以警告，并提醒重新输入。换句话说，该仿真平台并不是脱离实际进行仿真的，而是将各关节角变量的约束充分考虑在内，将机械臂的工作空间考虑在内，能够很好地模拟真实的 Dobot 机械臂。

图 5.8 仿真系统约束提示

该仿真系统能够很好地与用户进行交互，稍作调整也能和视觉系统进行配合。但仿真的主要目的是对两种运动规划方法进行比较分析。下面是相同起始和终止条件下，在关节空间内的运动规划和在笛卡尔空间内的运动规划（以门型运动模式为例）的结果比较和分析。

两种输入模式的结果可以相互转换，不影响运动规划的过程。这里不妨采用笛卡尔坐标的输入模式，取$p_0 = (180, -90, 170, 20°), p_f = (170, 80, 170, 60°)$。则关节空间运动规划和门型运动（取$h = 50$）的结果分别如下：

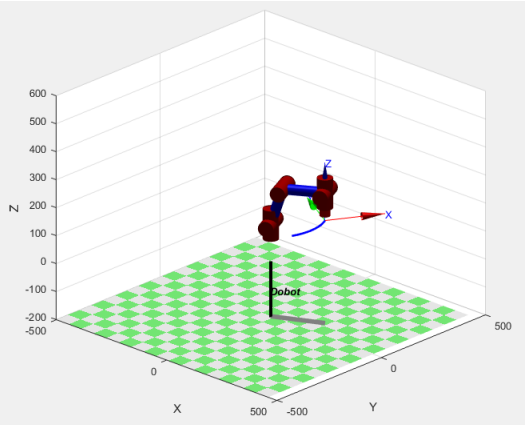

图 5.9(a) 关节空间运动规划结果

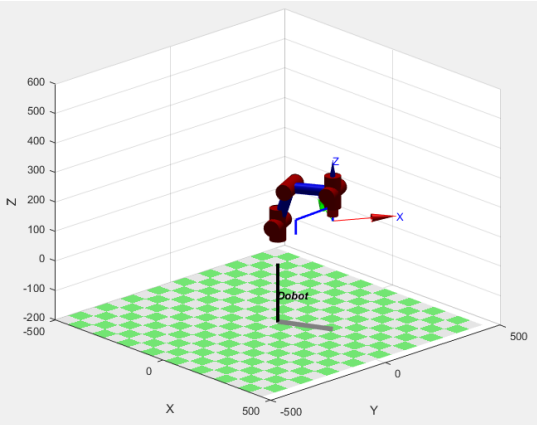

图 5.9(b) 笛卡尔空间门型运动规划结果

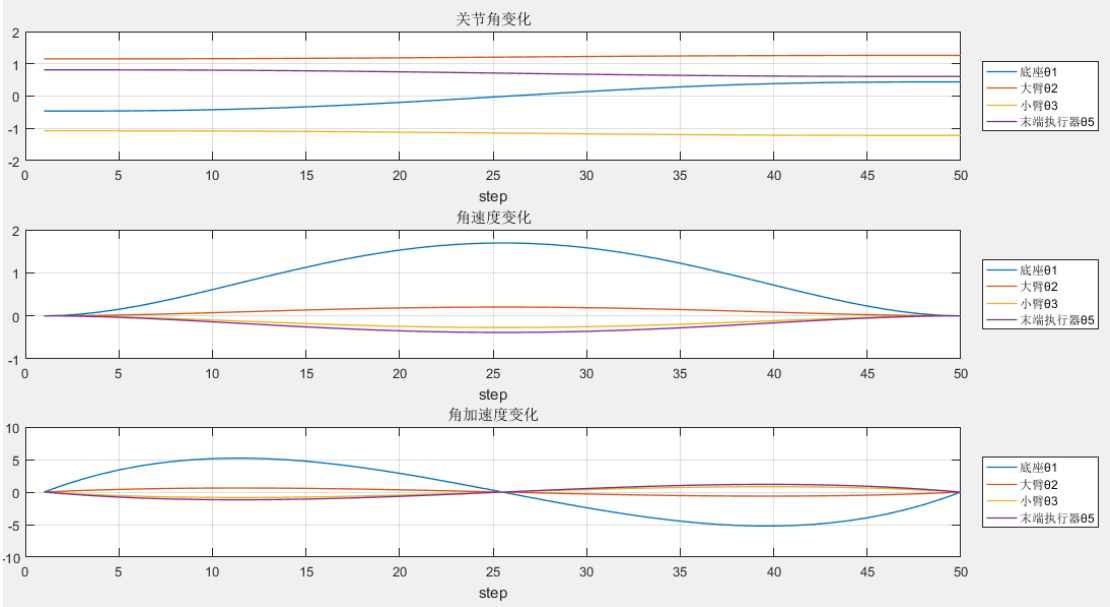

图 5.10(a) 关节空间运动规划各关节变量变化曲线

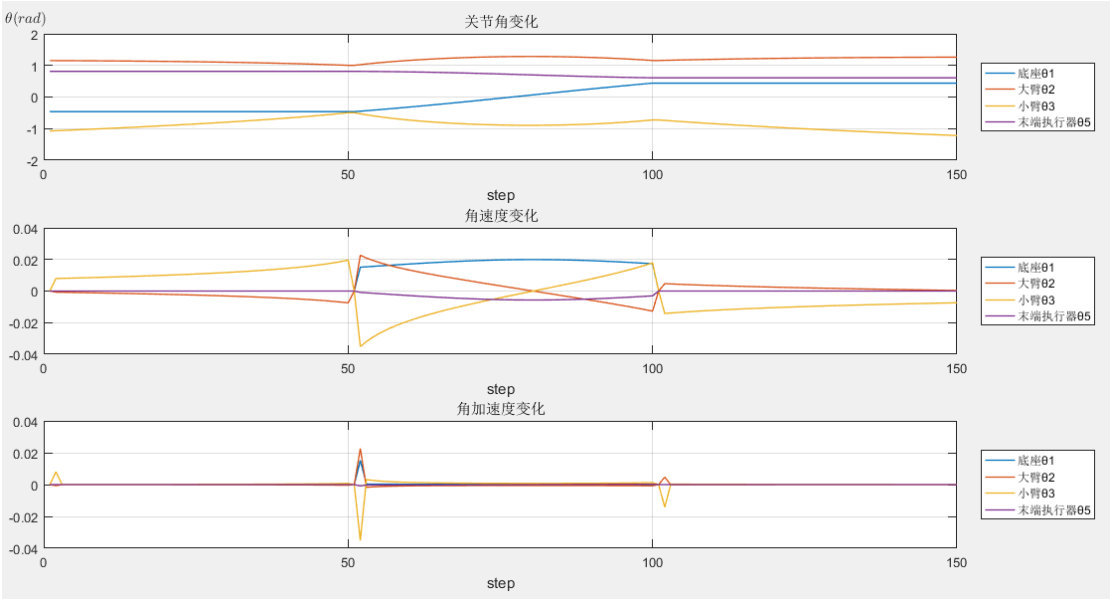

图 5.10(b) 笛卡尔空间门型运动规划各关节变量变化曲线

从上面的结果可以很直观看到两种规划方法的特点，关节空间运动规划中，机械臂末端执行器的运动轨迹通常是曲线，而在笛卡尔空间运动规划中，我们可以自定义机械臂末端执行器的轨迹。从运动过程中各关节变量的变化曲线来看，关节空间规划方法由于采用多项式插值（在仿真平台中采用五次多项式插值），因此无论是角度、角速度还是角加速度，其变化曲线都十分平滑；而笛卡尔空间规划方法是由末端执行器的位姿反过来确定关节角，因此相比之下曲线不是很平滑，存在多个转折点。

6 项目总结与展望

6.1 视觉组

6.1.1 项目成果总结

在本项目中，视觉组主要负责整个系统的检测系统的构建，包括图像预处理、SVM 分类器、InceptionV4 分类器以及瑕疵检测上位机的构建。

图像预处理中，通过对图像进行对比度提升、去噪以及锐化操作，以获取高质量的输入图像。

SVM 分类器中，首先 Stacked Autoencoder 对特征进行提取，提取出的特征再输入到 SVM 分类器中进行分类。但是由于 Autoencoder 的网络结构较简单，难以提取出复杂的图像特征，所以 SVM 分类器的分类结构相对不太理想。

在 SVM 分类器的基础上，我们又使用了 InceptionV4 网络进行瑕疵检测，通过对预训练模型的输出层进行迁移学习，从而能够见预训练的 ImageNet 模型迁

移到本项目的铝材瑕疵检测中。结果也显示 InceptionV4 网络有非常高的分类准确率。

基于 InceptionV4 网络的检测结果，我们依托于 PyQt5 进行了上位机的构建，并与最终的机械臂以及摄像头工作进行整合，实现了整个项目的运行状态可视化。

6.1.2 存在的问题及展望

在视觉组的研究过程中，我们也存在一些值得改进的地方。首先是摄像头读入的图像质量较差，通过摄像头读入的图像质量较差、并且图像的尺寸与网络的输入格式不复合，导致 InceptionV4 网络的检测效果较差。其次，摄像头读入的图像会随着外部光线变化以及人员流动等因素出现明暗程度等质量出现变化，从而对分类准确率造成影响。再则，SVM 分类器由于前面的 Autoencoder 提取的特征较简单，导致分类效果不理想。

6.2 机械臂组

6.2.1 项目成果总结

本项目种，机械臂组完成了机械臂的仿真与运动规划、实体机械臂的 API 编写、摄像头的 API 编写以及相应界面的构建。

机械臂仿真与运动规划中，我们从关节空间与笛卡尔空间中对机械臂进行了正运动学与逆运动学的建模，并对机械臂的速度、位姿进行了平滑规划。

机械臂与摄像头的 API 编写中，项目成员在熟悉 Dobot Studio 通讯架构的基础上采用 Python 脚本实现了对应 API 的开发。

6.2.2 存在的问题及展望

在机械臂组的研究中，我们的研究也存在一定的问题需要改进。首先在机械臂仿真中界面中，虽然系统提供两种仿真模式，但是对于用户仍然不够友好直观，希望能够通过直接操控按钮，控制机械臂运动到某个期望的姿态时选择确认完成初始条件和终止条件的输入。其次，仿真中由于采用的 Robotics 工具箱中的函数，但是该函数难以直接运用到 Dobot 函数，因为存在非独立变量不能任意操纵。再则，由于读取机械臂的运行参数较困难，本项目中没有实现实时的机械臂仿真，这也是较为遗憾的一点。

参考文献

[1] [飞粤云端 2018]广东工业智造大数据创新大赛 https://tianchi.aliyun.com/competition/entrance/231682/introduction

[2] Orchard M T. Spatially adaptive image denoising under overcomplete expansion[J], 2000.

[3] Xiao-Ping Zhang, Desai M D. Adaptive denoising based on SURE risk[J]. IEEE Signal Processing Letters, 5(10): 265-267.

[4] Scharcanski J, Jung C R, Clarke R T. Adaptive Image Denoising Using Scale and Space Consistency[J]. IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society, 2002.

[5] Baldi P. Autoencoders, unsupervised learning, and deep architectures[C]. Proceedings of ICML workshop on unsupervised and transfer learning, 2012: 37-49.

[6] Wetzel S J. Unsupervised learning of phase transitions: From principal component analysis to variational autoencoders[J]. Physical Review E, 2017, 96(2): 022140.

[7] Li W, Fu H, Yu L, et al. Stacked Autoencoder-based deep learning for remote-sensing image classification: a case study of African land-cover mapping[J]. International journal of remote sensing, 2016, 37(23): 5632-5646.

[8] Suykens J A, Vandewalle J. Least squares support vector machine classifiers[J]. Neural processing letters, 1999, 9(3): 293-300.

[9] Szegedy C, Ioffe S, Vanhoucke V, et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning[J].

[10] Wang T, Yao Y, Chen Y, et al. Auto-Sorting System Toward Smart Factory Based on Deep Learning for Image Segmentation[J]. IEEE Sensors Journal, 2018, 18(20): 8493-8501.

[11] 史先鹏, 刘士荣. 机械臂轨迹跟踪控制研究进展[J]. 控制工程, 2011, 18(1): 116-122.

[12] Chen Q, Zhang C, Ni H, et al. Trajectory planning method of robot sorting system based on S-shaped acceleration/deceleration algorithm[J]. International Journal of Advanced Robotic Systems, 2018, 15(6): 1729881418813805.

[13] 赵智勇, 王冬青. Dobot 机器人运动学分析及建模仿真[J]. 青岛大学学报: 工程技术版, 2017, 32(1): 52-57.

[14] 王彦璋. 基于 MATLAB/Robotics Toolbox 的六自由度机械臂仿真[J]. 陇东学院学报, 2016(2016 年 05): 22-26.