

基于视觉与动力学的 ArmPi 写字机器人开发

张柏铭 *

日期：2024.11.14

1 设计背景

随着人类对物质需求的增加促进大规模工业生产的发展，人类开始探索自动化生产的方法。自1959年世界上第一台可编程工业机器人“尤尼梅特”(Unimate)诞生以来，机械臂已经经过了长足的发展。闭环控制理论，各类传感器的利用，现代电子信息技术和机械臂的结合，到当代人工智能的应用，机械臂的潜力正在不断释放，其生产效应亦在不断扩大。为此，我们基于树莓派4B和ArmPi FPV机械臂，研究并搭建了自己的机械臂控制系统，实现了仿人类书写的功能。

2 机械臂原有功能实现和优化

2.1 原有平台组成部分

- 机械臂平台：包含具有三个可动关节的机械框架，机械爪，可旋转的底座，硬件集成平台，摄像头以及相关结构固件。

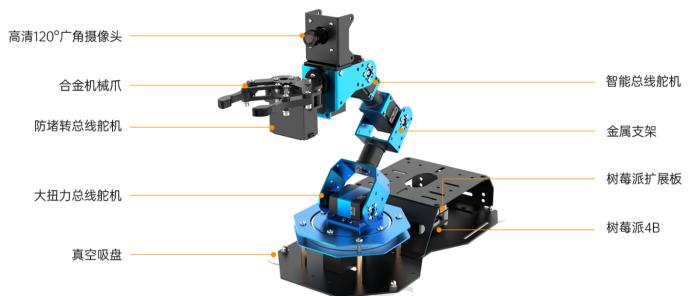


图 1：机械臂元件示意图

- 配套软件设施：NoMachine 图形化软件，可以实现对机械臂的信号输入和输出。
- 说明文件：包含机械臂说明书，软件使用教学书等。
- 模拟场地：配合机械臂平台原有功能实现。

2.2 原有平台功能的实现

- 人脸识别：机械臂水平转动，摄像头识别到人脸后，机械爪做旋转和夹取动作。

* baimingzhang@zju.edu.cn



图 2: 机械臂可实现功能示意图

- 物品分拣: 摄像头识别到目标颜色或标签方块后, 将其夹起并放置于指定区域。
- 智能码垛: 摄像头识别到方块后, 机械臂将会把方块按次序依次堆叠到指定区域。
- 智能入库: 摄像头识别到识别区内的方块后, 会将方块依次夹取并放置到对应的货架层内。
- 智能出库: 机械臂会依次夹取“货架”上的方块, 并将其移动至识别区。
- 智能仓转: 将目标方块从“货架”搬运到目标货架层位。

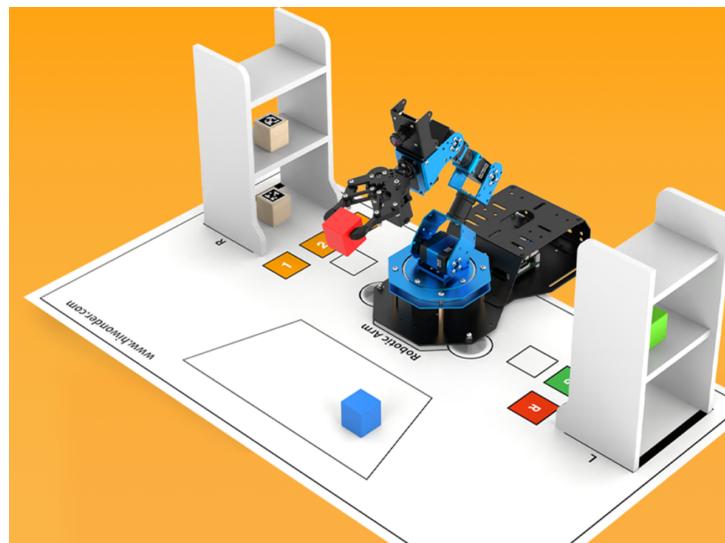


图 3: 智能码垛功能示意图

2.3 原有平台控制原理

机械臂每个关节对应一个数据, 共有五个可控制关节。通过在 Windows 或者 Ubuntu 平台安装配套控制软件 NoMachine, 并通过 WiFi 或者方有线式连接。在 NoMachine 控制界面中, 键入每个关节的数据, 以实现对机械臂总体姿态的控制。从控制软件中我们发现, 一组数据对应机械臂的一个姿态, 其工作过程本质上是不断在两个临近姿态进行转移。由此, 通过一系列元作业环节, 可以实现机械臂的复杂动作。

2.4 原有功能优化

- 操纵机械臂平台需要通过 WiFi 或有线方式连接电脑, 导致平台的可移动性下降。为此, 团队利用安装了 Ubuntu 系统和 NoMachine 控制软件的数位板与 ArmPi FPV 机械臂组成集成平

台，即可实现在数位板上实现对机械臂的操控和编程。

- 机械臂的每个关节都由一个数据控制，该数值为 0-1000 的整数。但是，将现实的三维坐标映射至机械臂坐标的过程较为复杂。为此，团队使用 Python 编写了一个可以实现由三维坐标向机械臂各个关节数据转换功能的程序，使得团队对机械臂的控制效率大幅增加。

3 新增功能：仿人写字机器人的实现

3.1 视觉识别模块

我们调用了摄像头，训练了全新的关于笔的模型，并用于机械臂寻找笔的过程。

技术难点：

受环境高噪音和摄像头低像素的限制，置信度略低。另外摄像头摆放使得笔并不能一直识别全部，故要训练两套模型——一套用于寻找笔的位置，一套用于抓笔前的微调。

实现步骤：

1. 数据采集与标注

模型一采集了笔自由放置时的照片，共 112*50 epoch 个数据，使用 labelimg 完成标注

模型二采集了笔快被夹起时的照片，共 167*50 epoch 个数据，使用 labelimg 完成标注



(a) 模型一 (b) 模型二

图 4: 模型标注与训练照片示意图

2. 模型训练首先对标注的 label 进行坐标和格式变换，并分离训练集与验证集。

```
146 num = len(total_xml)
147 list_index = range(num)
148 tv = int(num * trainval_percent)
149 tr = int(tv * train_percent)
150 trainval = random.sample(list_index, tv)
151 train = random.sample(trainval, tr)
152
153 file_trainval = open(txtsavepath + '\\trainval.txt', 'w')
154 file_test = open(txtsavepath + '\\test.txt', 'w')
155 file_train = open(txtsavepath + '\\train.txt', 'w')
156 file_val = open(txtsavepath + '\\val.txt', 'w')
```

图 5: 分离数据集示意图

```

239     def convert_annotation(image_set, image_id):
240         with open('%s\\xml\\%s.xml' % (image_set, image_id), encoding='UTF-8') as in_file,
241             open('%s\\labels\\%s.txt' % (image_set, image_id), 'w') as out_file:
242             tree = ET.parse(in_file)
243             root = tree.getroot()
244             size = root.find('size')
245             w = int(size.find('width').text)
246             h = int(size.find('height').text)
247             for obj in root.iter('object'):
248                 # difficult = obj.find('difficult').text
249                 # difficult = obj.find('Difficult').text
250                 difficult = 0
251                 cls = obj.find('name').text
252                 if cls not in classes or int(difficult) == 1:
253                     continue
254                 cls_id = classes.index(cls)
255                 xmlbox = obj.find('bbox')
256                 b = (float(xmlbox.find('xmin').text), float(xmlbox.find('xmax').text), float(xmlbox.find('ymin').text),
257                         float(xmlbox.find('ymax').text))
258                 b1, b2, b3, b4 = b
259                 # 标注越界修正
260                 if b2 > w:
261                     b2 = w
262                 if b4 > h:
263                     b4 = h
264                 b = (b1, b2, b3, b4)
265                 b = convert(b)
266                 b = convert_size((w, h), b)
267                 # print(b)
268                 out_file.write(str(cls_id) + " " + " ".join([str(a) for a in b]) + '\n')

```

图 6: 坐标和格式示意图

利用神经网络的深度学习进行训练，文件夹架构如下：

ArmPi > ArmPi_new				
	名称	修改日期	类型	大小
📁	dataSet	2024/11/12 22:06	文件夹	
📁	Predict	2024/11/12 22:21	文件夹	
📁	runs	2024/11/12 22:07	文件夹	
📁	train	2024/11/12 22:07	文件夹	
📁	val	2024/11/12 22:07	文件夹	
🐍	AutoSplit.py	2024/11/13 0:57	PY 文件	8 KB
📄	mydata.yaml	2024/11/11 0:10	Yaml 源文件	1 KB
📄	Readme.md	2024/11/11 1:37	Markdown 源文件	1 KB
📄	Rename_Copy.bat	2024/11/11 1:34	Windows 批处理文件	1 KB
📄	Reset.bat	2024/11/11 1:05	Windows 批处理文件	3 KB
🐍	Split_videos_into_figures.py	2024/11/12 22:00	PY 文件	1 KB
🐍	train.py	2024/11/12 22:07	PY 文件	1 KB

图 7: 模型二文件夹示意图

3. 预测使用训练完成的模型进行预测。

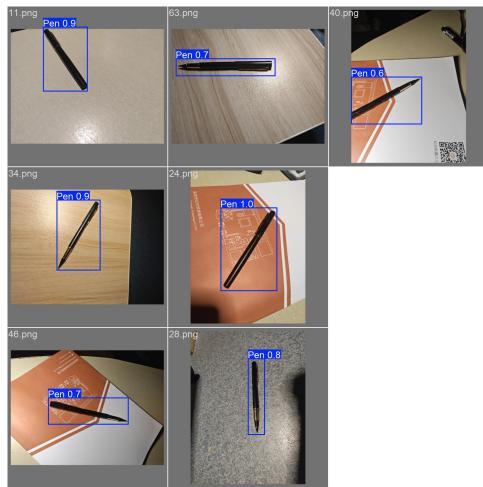


图 8: 模型一预测示意图

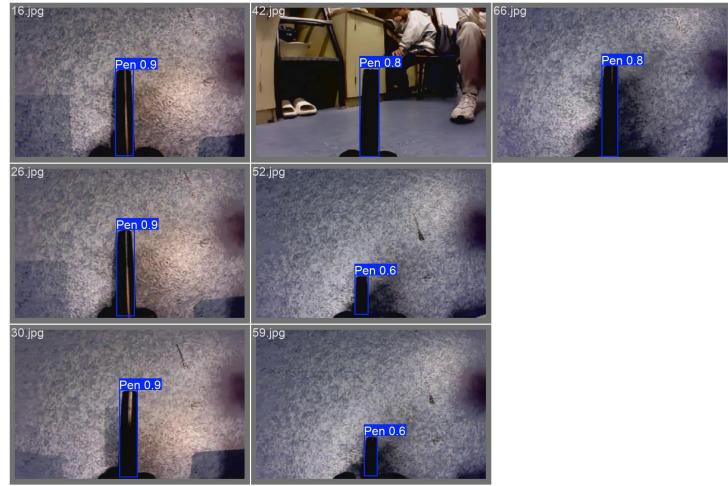


图 9: 模型二预测示意图

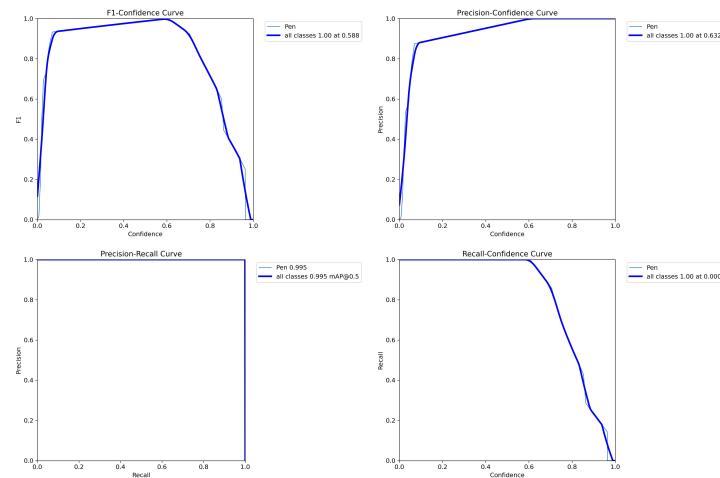


图 10: 置信度

另外，对于视频流还需调用 opencv 模块，此处不再赘述。

3.2 抓笔规划模块

抓取是根据视觉效果，先开环后闭环实现从桌上握起笔，以进行写字。

技术难点：

自由度非线性，难以对准实现精准控制微调。

实现步骤：

1. 找笔：先开环转到笔所在大概位置，低下摄像头
2. 靠近：附身到夹爪到达桌面高度
3. 根据识别图像定位笔，微调夹爪位置使其到达最中间（具体自由度函数及路径规划详见下节“写字系统模块”）

4. 对齐后握紧夹爪，起身，成功将笔夹起

```
sorting.py
286     x_dis = tag_x_dis = last_x_dis = target2[1]['servo6']
287     y_dis = tag_y_dis = 0
288
289     if state == 'color':
290         # 第四步：微调位置
291         if not adjust:
292             adjust = True
293             return True
294         else:
295             return True
296
297     # 第五步：对齐
298     bus_servo_control.set_servos(joints_pub, 500, ((2, 500 + int(F*gripper_rotation)), ))
299     rospy.sleep(0.8)
300     if not __isRunning:
301         servo_data = target4[1]
302         bus_servo_control.set_servos(joints_pub, 1000, ((1, 200), (3, servo_data['servo3']), (4, servo_data['servo4']), (5, servo_data['servo5'])))
303         rospy.sleep(1)
304         return False
305
306     # 第六步：夹取
307     bus_servo_control.set_servos(joints_pub, 500, ((1, grasps.grasp_posture - 80), ))
308     rospy.sleep(0.6)
309     bus_servo_control.set_servos(joints_pub, 500, ((1, grasps.grasp_posture), ))
310     rospy.sleep(0.8)
311     if not __isRunning:
312         bus_servo_control.set_servos(joints_pub, 500, ((1, grasps.pre_grasp_posture), ))
313         rospy.sleep(0.5)
314         servo_data = target4[1]
315         bus_servo_control.set_servos(joints_pub, 1000, ((1, 200), (3, servo_data['servo3']), (4, servo_data['servo4']), (5, servo_data['servo5'])))
316         rospy.sleep(1)
317         return False
318
319     # 第七步：抬升物体
320     if arasos.uo != 0:
```

图 11: 抓取程序设计示意图

3.3 写字系统模块

机械臂已经找到笔，夹起笔，现我们实现写字的自动化流程。

技术难点：

由于本个机械臂是仿人类书写，其自由度非线性，而是受角度影响。所以不同于直线模组类的以 xyz 坐标轴建立笛卡尔坐标系的打印机写字机器人，本系统功能实现需要协调三个旋转坐标系的关系，以在纸上写出平行的字，并在适时抬笔。（原装机械臂有五个自由度，但是由于三个自由度即可完成对三维空间各点的定义，本次开发仅调用其中三个自由度）

实现步骤：

1. 在 CAD 上对字体进行绘制

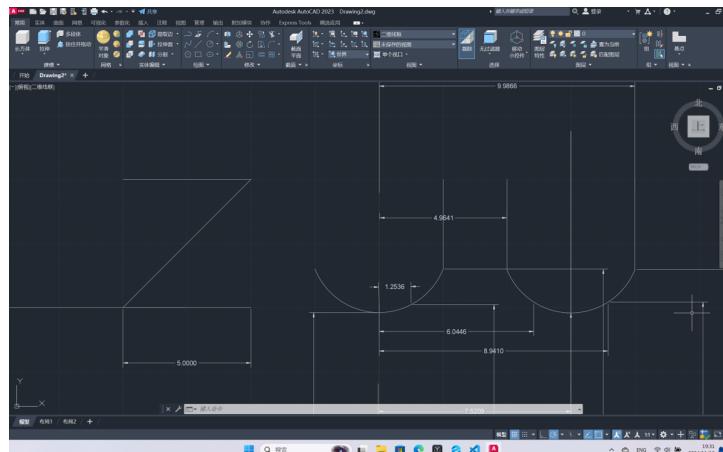


图 12: ZJU 的绘制

2. 对机械臂写字路径进行规划。例如书写“ZJU”，“Z”可以分解为三条直线，那么只需要四个坐标就可以令机械臂的运动轨迹和字体相同。而“U”下方的弯弧，则需要将弯弧分解多端直线，以绘制近似的弧线。

3. 通过运动学原理对三个自由度进行了计算，以实现笔的可控走动，计算过程如下。

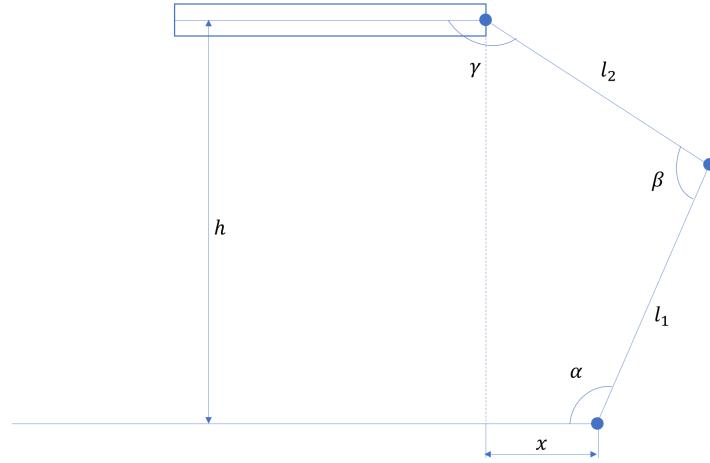


图 13: 三个自由度计算示意图

角度关系（保持夹爪水平）：

$$\alpha + \beta + \gamma = 2\pi$$

定位夹爪高度：

$$l_1 \sin \alpha + l_2 \sin \gamma = h$$

定位夹爪伸出距离：

$$x - l_1 \cos \alpha = -l_2 \cos \gamma$$

求解过程：

$$\begin{cases} l_1 \sin \alpha = h - l_2 \sin \gamma \\ x + l_2 \cos \gamma = l_1 \cos \alpha \end{cases}$$

消去 α ：

$$h^2 - 2hl_2 \sin \gamma + l_2^2 \sin^2 \gamma + x^2 + 2xl_2 \cos \gamma + l_2^2 \cos^2 \gamma = l_1^2$$

化简：

$$h^2 + x^2 + l_2^2 - 2hl_2 \sin \gamma + 2xl_2 \cos \gamma = l_1^2$$

移项：

$$l_1^2 - h^2 - x^2 - l_2^2 = 2xl_2 \cos \gamma - 2hl_2 \sin \gamma$$

同除 $2l_2$ ：

$$\frac{l_1^2 - h^2 - x^2 - l_2^2}{2l_2} = x \cos \gamma - h \sin \gamma$$

辅助角公式：

$$\frac{l_1^2 - h^2 - x^2 - l_2^2}{2l_2} = \sqrt{x^2 + h^2} \sin(\gamma + \theta) \quad \tan \theta = -\frac{x}{h}$$

最终解得：

$$\gamma = \arcsin \frac{h_1^2 - h^2 - x^2 - l_2^2}{2l_2\sqrt{x^2 + h^2}} - \arctan \left(-\frac{x}{h} \right)$$

回到原式：

$$\begin{cases} l_2 \sin \gamma = h - l_1 \sin \alpha \\ l_2 \cos \gamma = l_1 \cos \alpha - x \end{cases}$$

消去 γ :

$$\frac{l_2^2 - h^2 - x^2 - l_1^2}{-2l_1} = h \sin \alpha + x \cos \alpha = \sqrt{h^2 + x^2} \sin(\alpha + \phi) \quad \tan \phi = \frac{x}{h}$$

最终解得：

$$\alpha = \arcsin \frac{h_2^2 - h^2 - x^2 - h_1^2}{-2l_1\sqrt{h^2 + x^2}} - \arctan \left(\frac{x}{h} \right)$$

而：

$$\beta = 2\pi - \alpha - \gamma = 2\pi - \arcsin \frac{h_2^2 - h^2 - x^2 - h_1^2}{-2l_1\sqrt{h^2 + x^2}} - \arcsin \frac{h_1^2 - h^2 - x^2 - l_2^2}{2l_2\sqrt{x^2 + h^2}}$$

4. 利用 Python 转换程序，将规划路径的节点坐标转化为机械臂坐标，并在数位板的 NoMachine 中逐组输入并保存为 .6da 格式的执行文件

图 14: Python 转换程序示意图

5. 首先在 NoMachine 中实现夹笔的过程，在确保牢固之后，通过 Python 程序在命令行中运行，实现 .6da 的文件执行。机械臂则会按照计算得到的路径，带动笔绘制对应文字。

4 实现效果（具体演示过程详见 PPT）



图 15: 最终写字成功示意图

5 开发过程问题汇总及解决方案

在功能实现过程中，团队在各个环节遇到了很多问题，作如下汇总：

1. 在初步探索机械臂控制的时候，我们发现在 Windows 系统上安装 NoMachine 并通过 WiFi 方式无法连接上机械臂平台。解决方法：使用数位板连接并控制机械臂平台，实现了相同的功能。
2. 在数位板的控制软件上编写 6da 执行文件时，团队希望对 6da 文件进行直接编写以实现更快捷的控制。但团队并不了解 6da 格式文件的构成，常规打开方法显示的则是纯二进制文本无法阅读。团队放弃了这一想法。

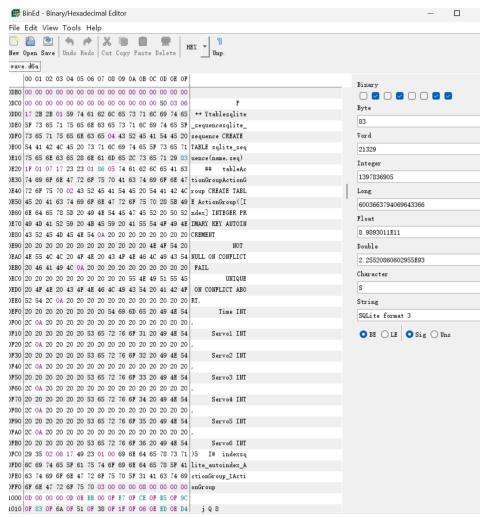


图 16: 二进制文件示意图

3. 在利用 python 程序实现 6da 动作文件的执行的过程中，由于源代码内对 6da 格式文件引用的绝对路径并不完整，且说明书上出现了错误，导致我们无法确认 6da 格式文件的确切应用路径。ArmPi 原始开发者并未把动作执行文件放在动作组文件夹，而是放在了传感器的文件夹内。团队利用 file 方法反向搜寻库的物理位置并通过源代码找到了 6da 文件的保存位置，并实现了利用 python 程序执行动作文件。

4. 机械臂抓手的路径难以保持横平竖直，在三维坐标上的偏移较大。在竖直约束平面内，机械臂共有三个可动关节，利用前这三个关节的配合保持了抓手及其关节高度的维持。团队通过运动学原理对三个自由度进行了精密计算，同时优化了 python 程序，结合人工微调，最终实现了笔尖在水平面内相对平稳的运行。
5. 由于机械臂在联动微调时存在抖动，所以有时候字会断断续续，如下图所示。我们增加了更多的数据点以提高系统的鲁棒性。

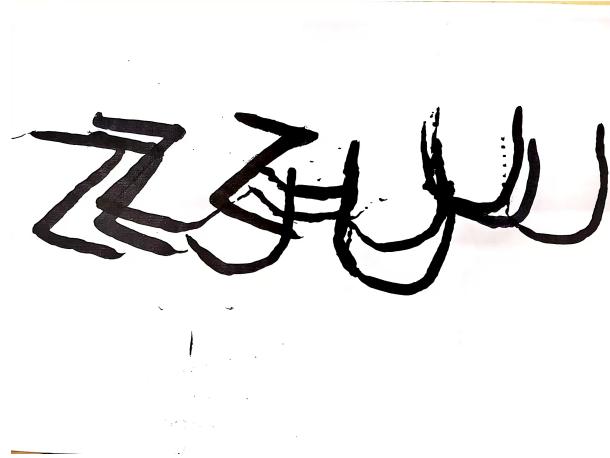


图 17: 调试书写示意图（为节省用纸才将多个 ZJU 写在一起）

6 后续改进与展望

- 通过摄像头识别图案或电子签名等，并进行仿绘。
- 通过联网调取字体的矢量模型数据库，并自动规划路线，生成动作执行文件，配合 UI 交互界面，实现任意输入文字的书写。
- 实现任意图案的绘制。
- 持续优化握笔的平稳性，改进握笔装置。

7 特别鸣谢

特别鸣谢浙江大学理论力学探究性实验项目组提供的资金支持。特别鸣谢理论力学课程提供的 ArmPi 硬件支持。感谢李华锋老师的指导和答疑。