MS001-B.03.001DD.1.0

MS-001

导引软件详细设计说明书

编制/日期：

审核/日期：

批准/日期：

杭州三坛医疗科技有限公司

文档更改履历

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 发布/实施日期 | 更改内容概述 | 更改者 |
| V1.0 |  | 文件新编 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**目录**

[1. 引言 1](#_Toc11963)

[1.1. 编写目的 1](#_Toc12148)

[1.2. 项目背景 1](#_Toc16007)

[1.3. 术语及缩写词 1](#_Toc1160)

[1.4. 参考资料 2](#_Toc9225)

[2. 总体设计 2](#_Toc20329)

[2.1. 需求概述 2](#_Toc21587)

[2.2. 软件结构 3](#_Toc609)

[3. 校准功能设计 4](#_Toc3073)

[3.1. 功能描述 4](#_Toc5428)

[3.2. 功能 4](#_Toc8979)

[3.3. 性能 4](#_Toc27139)

[3.4. 输入项目 4](#_Toc15549)

[3.5. 输出项目 5](#_Toc8801)

[3.6. 算法 5](#_Toc6833)

[3.7. 程序逻辑 7](#_Toc11769)

[3.8. 接口 8](#_Toc22441)

[3.9. 存储分配 12](#_Toc26483)

[3.10. 限制条件 12](#_Toc11061)

[3.11. 测试要点 13](#_Toc31103)

[4. 定位功能设计 13](#_Toc10530)

[4.1. 功能描述 13](#_Toc21800)

[4.2. 功能 14](#_Toc16627)

[4.3. 性能 14](#_Toc3046)

[4.4. 输入项目 14](#_Toc9795)

[4.5. 输出项目 15](#_Toc13055)

[4.6. 算法 15](#_Toc20739)

[4.7. 程序逻辑 15](#_Toc25402)

[4.8. 接口 16](#_Toc31995)

[4.9. 存储分配 18](#_Toc16120)

[4.10. 限制条件 18](#_Toc2546)

[4.11. 测试要点 18](#_Toc31267)

[5. 仿真功能设计 18](#_Toc27235)

[5.1. 功能描述 18](#_Toc31387)

[5.2. 功能 19](#_Toc31927)

[5.3. 性能 19](#_Toc7433)

[5.4. 输入项目 19](#_Toc19279)

[5.5. 输出项目 19](#_Toc18747)

[5.6. 算法 20](#_Toc20969)

[5.7. 程序逻辑 22](#_Toc6779)

[5.8. 接口 23](#_Toc6948)

[5.9. 存储分配 27](#_Toc17276)

[5.10. 限制条件 27](#_Toc24367)

[5.11. 测试要点 27](#_Toc21202)

[6. 登录功能设计 27](#_Toc21737)

[6.1. 功能描述 27](#_Toc25619)

[6.2. 功能 28](#_Toc12535)

[6.3. 性能 28](#_Toc27409)

[6.4. 输入项目 28](#_Toc27296)

[6.5. 输出项目 28](#_Toc29133)

[6.6. 算法 28](#_Toc23659)

[6.7. 程序逻辑 29](#_Toc10000)

[6.8. 接口 30](#_Toc18134)

[6.9. 存储分配 31](#_Toc19155)

[6.10. 限制条件 31](#_Toc11680)

[6.11. 测试要点 31](#_Toc11994)

[7. 软件信息功能设计 31](#_Toc3193)

[7.1. 功能描述 31](#_Toc29996)

[7.2. 功能 31](#_Toc1553)

[7.3. 性能 31](#_Toc17235)

[7.4. 输入项目 32](#_Toc2282)

[7.5. 输出项目 32](#_Toc6275)

[7.6. 算法 32](#_Toc32589)

[7.7. 程序逻辑 32](#_Toc9758)

[7.8. 接口 33](#_Toc3155)

[7.9. 存储分配 33](#_Toc17707)

[7.10. 限制条件 33](#_Toc9708)

[7.11. 测试要点 33](#_Toc14525)

[8. 配置功能设计 33](#_Toc7227)

[8.1. 功能描述 33](#_Toc11360)

[8.2. 功能 33](#_Toc24775)

[8.3. 性能 34](#_Toc5882)

[8.4. 输入项目 34](#_Toc13399)

[8.5. 输出项目 34](#_Toc11728)

[8.6. 算法 34](#_Toc22633)

[8.7. 程序逻辑 34](#_Toc12422)

[8.8. 接口 36](#_Toc2467)

[8.9. 存储分配 37](#_Toc24154)

[8.10. 限制条件 37](#_Toc29376)

[8.11. 测试要点 37](#_Toc9697)

[9. 日志功能设计 37](#_Toc946)

[9.1. 功能描述 37](#_Toc27669)

[9.2. 功能 38](#_Toc14104)

[9.3. 性能 38](#_Toc27769)

[9.4. 输入项目 38](#_Toc24642)

[9.5. 输出项目 38](#_Toc916)

[9.6. 算法 38](#_Toc11851)

[9.7. 程序逻辑 38](#_Toc32278)

[9.8. 接口 40](#_Toc13588)

[9.9. 存储分配 41](#_Toc6760)

[9.10. 限制条件 41](#_Toc4913)

[9.11. 测试要点 41](#_Toc6005)

[10. 系统退出功能设计 41](#_Toc17403)

[10.1. 功能描述 41](#_Toc17127)

[10.2. 功能 41](#_Toc26462)

[10.3. 性能 41](#_Toc18995)

[10.4. 输入项目 42](#_Toc21810)

[10.5. 输出项目 42](#_Toc27948)

[10.6. 算法 42](#_Toc16397)

[10.7. 程序逻辑 42](#_Toc26094)

[10.8. 接口 43](#_Toc3058)

[10.9. 存储分配 43](#_Toc14687)

[10.10. 限制条件 43](#_Toc24407)

[10.11. 测试要点 43](#_Toc31910)

[11. 软件注册功能设计 44](#_Toc26948)

[11.1. 功能描述 44](#_Toc16183)

[11.2. 功能 44](#_Toc28073)

[11.3. 性能 44](#_Toc1641)

[11.4. 输入项目 45](#_Toc28949)

[11.5. 输出项目 45](#_Toc28348)

[11.6. 算法 45](#_Toc26484)

[11.7. 程序逻辑 46](#_Toc26671)

[11.8. 接口 47](#_Toc9608)

[11.9. 存储分配 49](#_Toc28242)

[11.10. 限制条件 49](#_Toc4902)

[11.11. 测试要点 49](#_Toc25091)

[12. 网络功能设计 50](#_Toc19600)

[12.1. 功能描述 50](#_Toc6643)

[12.2. 功能 50](#_Toc4954)

[12.3. 性能 50](#_Toc10339)

[12.4. 输入项目 51](#_Toc13245)

[12.5. 输出项目 51](#_Toc1980)

[12.6. 算法 51](#_Toc26955)

[12.7. 程序逻辑 51](#_Toc11498)

[12.8. 接口 51](#_Toc26198)

[12.9. 存储分配 57](#_Toc13739)

[12.10. 限制条件 57](#_Toc31631)

[12.11. 测试要点 57](#_Toc23084)

[13. 机械臂控制功能设计 57](#_Toc2473)

[13.1. 功能描述 57](#_Toc17575)

[13.2. 功能 57](#_Toc29954)

[13.3. 性能 58](#_Toc18146)

[13.4. 输入项目 58](#_Toc21407)

[13.5. 输出项目 59](#_Toc25792)

[13.6. 算法 59](#_Toc4096)

[13.7. 程序逻辑 59](#_Toc9287)

[13.8. 接口 61](#_Toc20371)

[13.9. 存储分配 64](#_Toc22201)

[13.10. 限制条件 64](#_Toc8663)

[13.11. 测试要点 64](#_Toc27585)

# 引言

## 编写目的

本详细设计说明书，是在概要设计说明书的基础上进一步明确系统结构，详细的介绍系统的各个功能，为进行后面的编码和测试做准备。

本系统设计说明书主要面向：软件设计人员、UI设计人员、测试人员。

## 项目背景

 传统的骨科手术非常依赖有经验的医生，手术过程也较复杂且漫长。为了能更加精准的定位病灶，缩短手术的时间，并尽可能减轻患者的痛苦，现开发导引软件，作为模块化导引系统的重要组成部分，主要用于控制机械臂运动、规划通道的实际定位。

## 术语及缩写词

* 仿真：根据机械臂的初始位置和目标位置，模拟机械臂的运动过程、并显示。
* 定位：根据规划软件下发的定位目标位置，控制机械臂运动定位。
* C臂机: C型臂X光机，因其外形类似英文字母 C，故被称为 C 臂机，是用于骨科手术的 X射 线影像设备。
* 机械臂：机械臂是指高精度，多输入多输出、高度非线性、强耦合的复杂系统。因其独特的操作灵活性, 已在工业装配, 安全防爆等领域得到广泛应用。
* AP（Anteroposterior）：正位，从胸前指向背部的方向，Anterior为前，Posterior为后，因此PA表示反方向。
* LAT（lateral）：侧位，左和右。
* DICOM：一种医疗图像格式。
* C臂机校准：通过术前大量采集透视图像来计算C臂机内参及畸变参数，用于C臂机影像的校正及透视变换。
* 正侧位校准：即通过正位、侧位透视，获取正位、侧位透视图，然后根据已知的2D、3D数据和C臂机内参及畸变参数，计算出正位（或侧位）标定板与Ｃ臂机间位置的变换关系。

## 参考资料

1. 《模块化手术导引系统技术需求说明书》
2. 《模块化手术导引系统-导引软件概要设计说明书》
3. 《医疗器械网络安全注册技术审查指导原则》
4. YY/T 0664-2008 《医疗器械软件软件生存周期过程》
5. YY/T 1406.1-2016 《医疗器械软件 第1部分：YY/T 0316 应用于医疗器械软件的指南》
6. GB/T 8567-2006《计算机软件文档编制规范》

# 总体设计

## 需求概述

导引软件包括登录功能、校准功能、定位功能、辅助设备功能、帮助功能五大功能以及校准子功能、定位子功能、仿真子功能、登录子功能、软件信息子功能、配置子功能、日志子功能、系统退出子功能、软件注册子功能块、网络子功能、机械臂控制子功能十一个子功能。

## 软件结构

模块化手术导引系统

-

导引软件

校准子功能

定位子功能

仿真子功能

配置子功能

系统退出子功能

软件注册子功能

网络子功能

机械臂控制子功能

登录子功能

软件信息子功能

日志子功能

登录功能

校准功能

定位功能

辅助设备功能

帮助功能

机械臂控制子功能功能

网络子功能

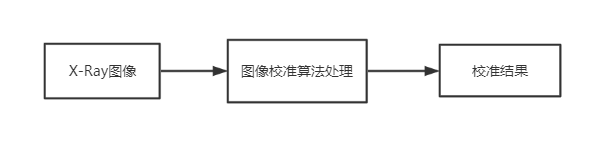
# 校准功能设计

## 功能描述

1. 校准功能主要用来对X-Ray图像进行校准，根据当前机械臂的姿态分为正位校准与侧位校准；
2. 校准功能在软件中属于常驻内存、不可重入，属于顺序处理且代码没有覆盖要求。本功能与其他功能耦合较小；登录成功后，进入校准界面；用户也可以通过点击校准按钮手动进入校准界面；软件默认校准顺序为正位校准-侧位校准，调用算法校准图片，弹出对话框显示校准结果；软件自动对机械臂标定板的姿态进行检测，以防止正侧位标定的误操作。
3. 功能内部代码增加边缘性检测，详见接口部分。

## 功能

1. 当正位校准时，软件检测标定板的姿态，符合正位要求，进行校准操作。完成后发送正位校准图片至规划软件，并在接收到规划软件的回复信息后，确认正位校准成功，并在提示框内显示校准状态，同时进入侧位校准模式；
2. 当侧位校准时，软件检测标定板的姿态，符合侧位要求，进行校准操作，完成后发送侧位校准图片和正侧位相对位置关系矩阵至规划软件，并在接收到规划软件的回复信息后，确认侧位校准成功。

校准功能功能IPO图

## 性能

校准所用时间应当≤10s。

## 输入项目

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| **1** | X-Ray图像 | 无 | dcm | 有足够的小球 | 自动输入 | 无 |
| **2** | 机械臂姿态 | 无 | string | 正位姿态或侧位姿态 | 自动输入 | 无 |

## 输出项目

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输出方式** | **备注** |
| **1** | 校准结果 | 无 | bool | 1或0 | 界面显示 | 校准成功将激活定位功能 |
| **2** | 校准精度 | 无 | double | 不为空 | 界面显示 | 校准失败不显示校准精度 |

## 算法

3.6.1. 图像校准算法

算法目的：

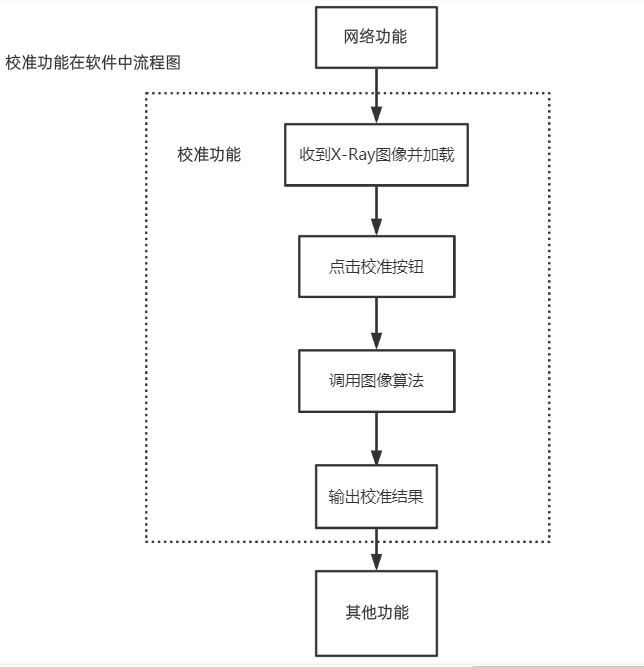
得到X光图像和机械臂之间的变换关系。

算法流程：

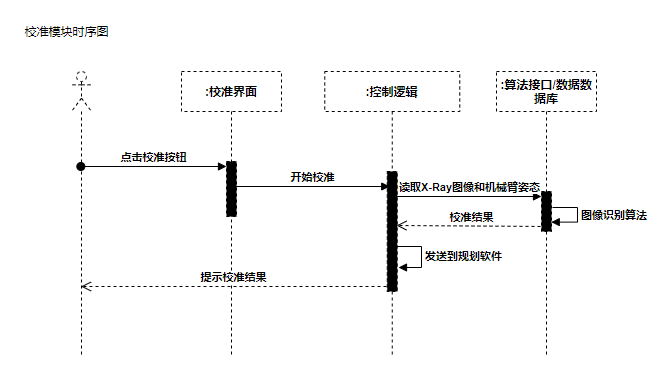
1. 获取X光图像，进行畸变矫正处理，并将图像归一化到0-255范围内；
2. 识别图像中的小球圆心坐标，圆心坐标减去图像中心再乘以像素间隔，得到用于配准的二维小球坐标；
3. 读取配置文件中的C臂机内参，对上述二维坐标与配置文件中的三维小球坐标进行配准，得到X光图像和机械臂之间的变换关系；
4. 正位图像校准得到正位图像与机械臂的关系A，侧位图像校准得到侧位图像与机械臂的关系B，则正位图像和侧位图像的变换关系C = B-1\*A。



## 程序逻辑



校准功能在软件内的流程图



校准功能时序图

## 接口

校准功能包含MarkerRegistration、calibrationProcessor 、RobotRegistrationProcessor 三个主要类。其中重要接口详细说明如下：

3.8.1. MarkerRegistration

/\*\*

\* @brief The MarkerRegistration class

\* Maker球的校准

\*/

class MarkerRegistration:

/\*\* Process function is used for registration between two sets of points

\*without one-to-one correspondence

\* @param pts\_2d输入1

\* @param pts\_3d输入2

\*/

Result Process(const Marker2DListType & pts\_2d, Marker3DListType & pts\_3d);

/\*\* Registration function is used for registration between two sets of \*points with one-to-one correspondence

\* @param pts\_2d输入1

\* @param pts\_3d输入2

\*/

Result Registration(const Marker2DListType & pts\_2d, Marker3DListType & pts\_3d);

void PoseUpdate(const Marker2DListType & fixed\_pts, Marker3DListType & moving\_pts, Matrix2DType & mjk,

std::vector<double> &pose\_para, double & cost);

Matrix2DType Qjk(const Marker2DListType & fixed\_pts, Marker3DListType & moving\_pts, std::vector<double> &para);

Matrix2DType SoftAssign(Matrix2DType & q\_mat, double & beta);

MatchType FindMatch(Matrix2DType & mat);

itk::Matrix<double, 4, 4> Pos2Mat(std::vector<double> & para);

/\*\*

\* @brief SetNormalizeRotationPara

\* 设置标准化旋转参数

\*@param para旋转参数

\*/

void SetNormalizeRotationPara(double & para);

/\*\*

\* @brief SetNormalizeTransformPara

\* 设置标准化转换参数

\*@param para转换参数

\*/

void SetNormalizeTransformPara(double & para);

/\*\*

\* @brief SetInitialPosePara

\* 设置初始化位置参数

\*@param para位置参数

\*/

void SetInitialPosePara(std::vector<double> &para);

/\*\*

\* @brief SetFocalLength

\* 设置焦距参数

\*@param para焦距

\*/

void SetFocalLength(double &para);

/\*\*

\* @brief SetCostValue

\* 设置消耗值

\*@param para 消耗值

\*/

void SetCostValue(double &para);

3.8.2. calibrationProcessor

/\*\*

\* @brief The calibrationProcessor class

\* 校准处理类

\*/

class calibrationProcessor:

/\*\*

\* @brief initialize

\* 初始化函数

\*@param name 配置文件目录

\*/

void initialize(const QString & name);

/\*\*

\* @brief prepared 是否准备就绪

\* @return

\*/

bool prepared();

/\*\*

\* @brief CalibrateImage

\* 图像校准

\*@param img待校准图像

\*/

cv::Mat CalibrateImage(cv::Mat & img);

/\*\*

\* @brief UndistortPoint

\* 校正各个点

\*@param pts待校正的点

\*/

ptType UndistortPoint(ptType & pts);

3.8.3. RobotRegistrationProcessor

/\*\*

\* @brief The RobotRegistrationProcessor class

\* 机械臂校准处理类

\*/

class RobotRegistrationProcessor :

/\*\*

\* @brief isCalibrated

\* 是否已校准

\*@return

\*/

bool isCalibrated();

/\*\*

\* @brief isRegsitered

\* 是否已注册

\*@return

\*/

bool isRegsitered();

/\*\*

\* @brief isFrontalRegsitered

\* 是否已正位注册

\*@return

\*/

bool isFrontalRegsitered();

/\*\*

\* @brief isLateralRegsitered

\* 是否已侧位注册

\*@return

\*/

bool isLateralRegsitered();

/\*\*

\* @brief lastLocation

\* 获取当前位置

\*@return

\*/

LocateResult lastLocation();

/\*\*

\* @brief lastError

\* 获取当前错误

\*@return

\*/

ArmRegistionError lastError();

/\*\*

\* @brief armPose

\* 获取机械臂当前姿态

\*@pose 位置信息

\*@return

\*/

ArmPoseType armPose(const std::vector<double>& pose);

/\*\*

\* @brief calibrateFrontalXRay

\* 正位X-Ray校准

\*@pose imageData 图像信息

\*@pose ret函数返回

\*@return

\*/

ImageData calibrateFrontalXRay(const ImageData& imageData, int& ret);

/\*\*

\* @brief calibrateLateralXRay

\* 侧位X-Ray校准

\*@pose imageData 图像信息

\*@pose ret函数返回

\*@return

\*/

ImageData calibrateLateralXRay(const ImageData& imageData, int& ret);

/\*\*

\* @brief saveFrontalCalibratedXRay

\* 保存正位X-Ray校准结果

\*@pose imageData 图像信息

\*/

void saveFrontalCalibratedXRay(const ImageData& imageData);

/\*\*

\* @brief saveLateralCalibratedXRay

\* 保存侧位X-Ray校准结果

\*@pose imageData 图像信息

\*/

void saveLateralCalibratedXRay(const ImageData& imageData);

## 存储分配

校准功能本地文件存储内容：

1. 已校准成功的正位图片保存在相对目录: "./calibrated\_image/frontal.bmp"文件。
2. 已校准成功的侧位图片保存在相对目录下:"./calibrated\_image/lateral.bmp"文件。

## 限制条件

1. 校准前需保证规划软件已连接。
2. 校准前需拍摄正位和侧位的X光图像。
3. X光图像中需包含校准板，且校准板的位置尽量在图像中间。
4. 机械臂在校准时需处于相应的位置：正位或侧位。

## 测试要点

1. 直接点击开始校准；
2. 校准是否高亮；
3. Tool文件与前端器械数据一致，正侧位校准；
4. Tool文件与前端器械数据不一致，正侧位校准；
5. 畸变文件与C臂机一致，正侧位校准；
6. 畸变文件与C臂机不一致，正侧位校准；
7. 畸变文件未配置，正侧位校准；
8. 发送bmp，Dicom格式的图片校准；
9. 发送非bmp，Dicom格式的图片校准；
10. 发送长宽不一致的图片校准；
11. 校准后的图片弹框，退出或关闭按钮测试；
12. 跟校准相关参数修改，重新校准正侧位；
13. 更改工具，校准正侧位；
14. 正侧位校准后，查看校准进度条何校准精度及百分比；
15. 正位校准，侧位校准，校准成功；
16. 侧位校准，正位校准，校准失败。

# 定位功能设计

## 功能描述

1. 定位功能主要用来控制机械臂移动到目标点，包含2D定位、3D定位以及定位重置功能，并且UI中集成了仿真功能；
2. 定位功能在软件中属于常驻内存、不可重入，属于顺序处理且代码没有覆盖要求。本功能与其他功能耦合一般；本功能得到的机械臂目标位置通过网络功能下发至机械臂下位机，由下位机进一步控制机械臂运动。
3. 功能内部代码增加边缘性检测，详见接口部分。

## 功能

1. 2D定位功能区有居中按钮、姿态仿真按钮、通道定位按钮、重置按钮；点击居中按钮后，脚踩住踏板，机械臂运动至目标位置，提示框显示机械臂状态或错误信息；姿态仿真按钮功能详见第五章；在图像区用鼠标左键选择目标位置后点击通道定位按钮，脚踩住踏板，机械臂运动至目标位置，提示框显示机械臂状态或错误信息；定位过程中点击重置按钮，取消当前定位指令，非定位过程中点击重置按钮不做任何操作；
2. 3D定位功能区有通道定位按钮、姿态仿真按钮、重置按钮；点击通道定位按钮后，脚踩住踏板，机械臂运动至目标位置，提示框显示机械臂状态或错误信息；姿态仿真按钮功能详见第五章；定位过程中点击重置按钮，取消当前定位指令，非定位过程中点击重置按钮不做任何操作。



定位功能功能IPO图

## 性能

无。

## 输入项目

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| **1** | 目标点 | 无 | double | 不为空 | 自动同步 | 无 |
| **2** | 运动过程中的路径点 | 无 | double | 不为空 | 自动同步 | 无 |

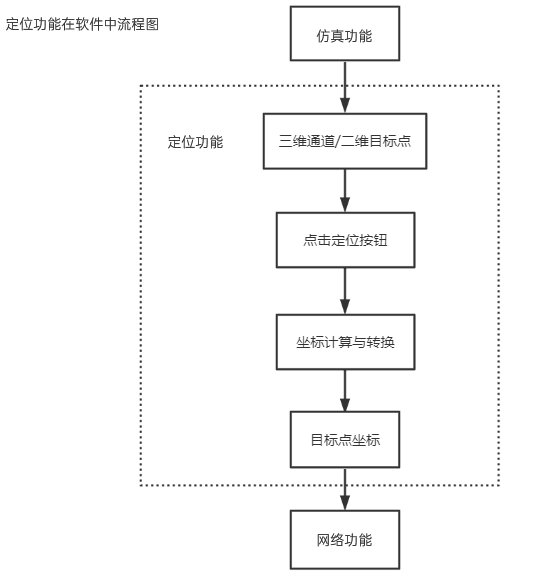
## 输出项目

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| **1** | 定位结果 | 无 | bool | 1或0 | 界面显示 | 机械臂是否运动到位 |

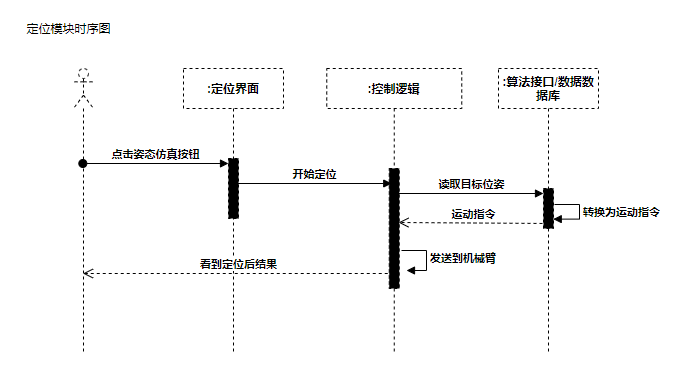
## 算法

无。

## 程序逻辑



定位功能在软件内的流程图



定位功能时序图

## 接口

定位功能包含ArmDrive、LaserLocator两个主要类。其中重要接口详细说明如下：

4.8.1. ArmDrive

/\*\*

\* @brief The ArmDriveclass

\* 机械臂驱动类

\*/

class ArmDrive:

/\*\*

\* @brief moveChannel

\* 通道定位

\*@param pos 位置

\*@param angle 角度

\*/

void moveChannel(const ArmPosition& pos, const int& angle);

/\*\*

\* @brief moveChannel

\* 按路径定位

\*@param angle 角度

\*/

void movePath(const int& angle);

/\*\*

\* @brief setFreeDrive

\* 开启/禁用自由拖动

\*@param freeDriveOn开关

\*/

void setFreeDrive(bool freeDriveOn);

/\*\*

\* @brief setSpeed

\* 设置定位的速度

\*@param speed 运动速度

\*/

void setSpeed(double speed);

4.8.2. LaserLocator

/\*\*

\* @brief The LaserLocator

\* 机械臂驱动类

\*/

class LaserLocator:

/\*\*

\* @brief updateHeight

\* 更新高度

\*@param height高度

\*/

void updateHeight(int height);

/\*\*

\* @brief updateStatus

\* 更新状态

\*@param s 状态

\*/

void updateStatus(Status s);

/\*\*

\* @brief resetStatus

\* 重置状态

\*/

void resetStatus();

/\*\*

\* @brief status

\* 获取状态

\*/

Status status();

## 存储分配

定位功能内存存储内容：

vector<Vector6d> pathall。

## 限制条件

1. 定位前需要先通过姿态仿真，仿真成功才能进行定位。
2. 定位时需踩住脚踏，机械臂才会运动。

## 测试要点

1. 定位页面图像显示区显示正位；
2. 图像旋转后定位，定位准确；
3. 定位高亮显示；
4. 接收定位指令提示信息正确性验证；
5. 3D姿态仿真后定位；
6. 3D未仿真，直接定位，提示请先进行姿态仿真；
7. 踩脚踏，点通道定位，定位失败；
8. 点通道定位，踩脚踏，定位成功；
9. 3D定位中，重置，指令清除；
10. 2D居中测试；
11. 2D姿态仿真定位测试；
12. 参数区调整机械臂6个关节角数据，机械臂姿态变化。

# 仿真功能设计

## 功能描述

1. 仿真功能主要用来模拟机械臂运动轨迹和目标位置；
2. 仿真功能在软件中属于常驻内存、不可重入，属于顺序处理且代码没有覆盖要求。本功能与其他功能耦合较小；本功能得到机械臂从当前点到目标点的一系列路径点，由定位功能处理后下发至机械臂下位机。
3. 功能内部代码增加边缘性检测，详见接口部分。

## 功能

1. 模型仿真区用于模拟机械臂的运动轨迹和当前位姿；用户可以通过按住鼠标左键来旋转机械臂模型，通过鼠标右键进行模型的缩放操作，通过按住中键来进行模型的移动操作；
2. 点击姿态仿真按钮后，功能根据3D通道或者2D目标点，自动规划机械臂从当前位置移动到目标位置的理想路径，规划成功后在仿真区显示；
3. 参数区显示机械臂每个关节当前位置角度信息，防止某个关节运动到极限位置，也为每个关节手动转至合适位置提供参考。参数区还提供控制按钮，用于帮助每个关节恢复至理想位置。



仿真功能功能IPO图

## 性能

仿真计算时间应当≤3s。

## 输入项目

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| **1** | 三维通道 | 无 | double | 不为空 | 网络功能输入 | 无 |
| **2** | 二维目标点 | 无 | double | 不为空 | 鼠标点击 | 无 |

## 输出项目

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输出方式** | **备注** |
| **1** | 仿真结果及提示 | 无 | bool | 1或0 | 界面显示 | 成功：仿真区的模型连续运动；  失败：提示错误信息 |
| **2** | 路径点 | 无 | double | 不为空 | 自动同步 | 仅在仿真成功时输出 |

## 算法

5.6.1. 规划路径算法

算法目的：

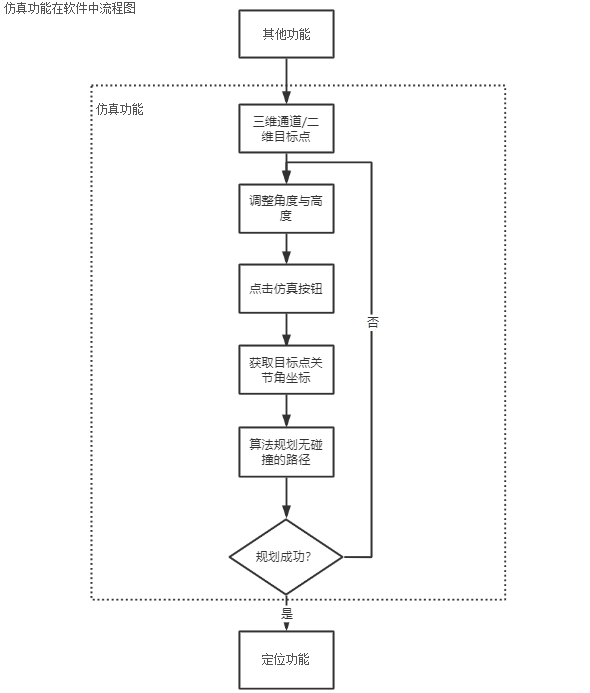
根据机械臂起点和目标点的信息，计算出最优路径，输出全部路径点。

算法流程：

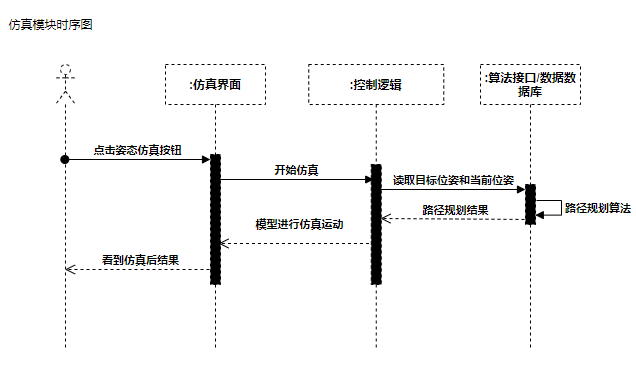
1. 获取机械臂起点以及需要定位目标点的信息，获取机械臂外形物理数据，获取手术环境信息。
2. 依据机械臂外形物理数据构建机械臂自碰撞模型，并建立自碰撞判断标准。
3. 在规划的空间中依据BI-RRT\*算法计算出一条无碰撞的路径，当该条路径点建立成功后，则对该条路径进行优化，使机械臂运动路径更加平稳和高效，随后将路径点下发给控制器完成控制机械臂移动的任务。
4. 若该条路径没有建立成功，则检索搜索次数是否在限定的范围内，在限定次数范围内则扩大搜索范围继续搜索，否则返回搜索失败的信息。



## 程序逻辑



仿真功能在软件内的流程图



登录功能时序图

## 接口

仿真功能包含RRT、RobotPath两个主要类。其中重要接口详细说明如下：

5.8.1. RRT

/\*\*

\* @brief The RRT

\* 路径规划类

\*/

class RRT:

/\*\*

\* @brief setStartEndPoint

\* 设置起始点、终点

\*@param start\_joint\_in起点

\*@param nd\_joint\_in 终点

\*/

int setStartEndPoint(const Vector6d &start\_joint\_in, const Vector6d &end\_joint\_in);

/\*\*

\* @brief setHigh

\* 设置地平面的限制高度

\*@param high高度

\*/

void setHigh(const double high);

/\*\*

\* @brief getTotalCount

\* 获取路点的个数

\*@return

\*/

int getTotalCount();

/\*\*

\* @brief toFindPath

\* 寻找路径点

\*@return

\*/

int toFindPath();

/\*\*

\* @brief setMode

\* 设置模式

\*@param mode模式

\*/

int setMode(const int mode);

/\*\*

\* @brief extendNewNode

\* 拓展新节点

\*@param rand\_joint随机后继点

\*@param is\_start\_tree 是否原始树

\*/

int extendNewNode(const Vector6d &rand\_joint, const bool &is\_start\_tree);

/\*\*

\* @brief findTrees

\* 寻找树

\*@return

\*/

int findTrees();

/\*\*

\* @brief creatNewRandJoint

\* 创建新节点

\*@param is\_start\_tree 是否原始树

\*@param new\_joint新后继点

\*/

void creatNewRandJoint(const bool &is\_start\_tree, Vector6d &new\_joint);

/\*\*

\* @brief getNewestAddNode

\* 获取新增节点

\*@param is\_start\_tree 是否原始树

\*/

TreeNode getNewestAddNode(const bool &is\_start\_tree);

/\*\*

\* @brief judgeReach

\* 判断是否到达

\*@param node1 节点1

\*@param node1 节点2

\*/

bool judgeReach(const TreeNode &node1, const TreeNode &node2);

5.8.2. RobotPath

/\*\*

\* @brief The RobotPath

\* 机械臂路径类

\*/

class RobotPath:

/\*\*

\* @brief getTransformMatrix

\* 获取转换矩阵

\*/

void getTransformMatrix(const double &joint, const double &dh\_d, const double &dh\_a, const double &dh\_alpha, Matrix44d &tran\_mat);

/\*\*

\* @brief fkine

\* 策略

\*/

void fkine(const Vector6d &joint, Matrix44d &tran\_mat);

/\*\*

\* @brief calModulePose

\* 计算功能位置

\*/

void calModulePose(const Vector6d &joints, Vector6d \* module\_pose);

/\*\*

\* @brief fkineTool

\* 策略工具

\*/

void fkineTool(const Vector6d &joint, Matrix44d &tran\_mat);

/\*\*

\* @brief calModule8TailPose

\* 计算功能末端位置

\*/

void calModule8TailPose(const Vector6d &joints, Vector3d \* module8\_tail\_pose);

/\*\*

\* @brief collision

\* 碰撞

\*/

bool collision(const Vector6d \*module\_pose, const Vector6d &current\_joint, Obstacles & obstacles);

/\*\*

\* @brief judgeLimit

\* 判断边界

\*/

bool judgeLimit(const Vector6d &joint, const double &high);

/\*\*

\* @brief getMinDisInPointBetweenLine

\* 获取最短距离

\*/

double getMinDisInPointBetweenLine(const Eigen::Vector3d &point, const Eigen::Vector3d &a\_line, const Eigen::Vector3d &b\_line);

/\*\*

\* @brief judgeChannel

\* 判断通道

\*/

bool judgeChannel(const Vector3d start\_pos, const Vector3d end\_pos, const Vector6d &joint);

/\*\*

\* @brief collisionLine

\* 碰撞线

\*/

bool collisionLine(const Vector6d &frist\_joint\_in, const Vector6d &second\_joint\_in, const double high, Obstacles & obstacles);

## 存储分配

仿真功能内存存储内容：

vector<Vector6d> pathall。

## 限制条件

1. 正位校准成功、侧位校准成功才能开始仿真。

## 测试要点

1. 未发送定位指令仿真，仿真失败；
2. 发送定位后，仿真，仿真姿态与实际姿态一致；
3. 起点或终点姿态低于基座，仿真失败；
4. 发送定位超限，仿真失败；
5. 定位中仿真，仿真成功；
6. 仿真中定位，定位失败；
7. 进入仿真查看仿真默认状态与发送定位状态一致；
8. 定位完成后仿真被重置；
9. 仿真模型鼠标交互测试：鼠标左键旋转，中键常按平移；
10. 仿真角度、高度调整，仿真实时变化；
11. 机械臂断开连接，仿真，仿真失败。

# 登录功能设计

## 功能描述

1. 登录功能是用户进入系统的唯一入口，保证了软件的安全；登录功能布局设计、风格合理；
2. 登录功能在软件中属于常驻内存、不可重入，不属于顺序处理且代码没有覆盖要求。本功能与其他功能耦合较小：登录成功可进入数据管理功能；
3. 用户信息以特定的内部加密方式保存在数据库中，防止数据泄露。
4. 用户登录进入软件之后30分钟内不操作会自动进入锁屏界面，解锁需要输入账户对应密码，保证软件具有网络安全的自动注销能力。
5. 不同账户拥有不同权限，保证软件网络安全的授权能力。
6. 用户拥有唯一账户，需要身份验证才能登陆，保证了软件网络安全的人员鉴别能力。

## 功能

1. 系统启动后自动开启软件，登录功能包含用户登录接口：输入正确的账号密码后可以成功离开登录功能进入校准功能；
2. 账号密码错误时，软件提示对应消息；
3. 退出登录则系统自动关机。

## 性能

无。

## 输入项目

登录功能-全功能输入项目如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| **1** | 账号 | user\_id | varchar | 满足规定正则 | 手动输入 | 无 |
| **2** | 密码 | user\_pwd | varchar | 满足规定正则 | 手动输入 | 无 |

## 输出项目

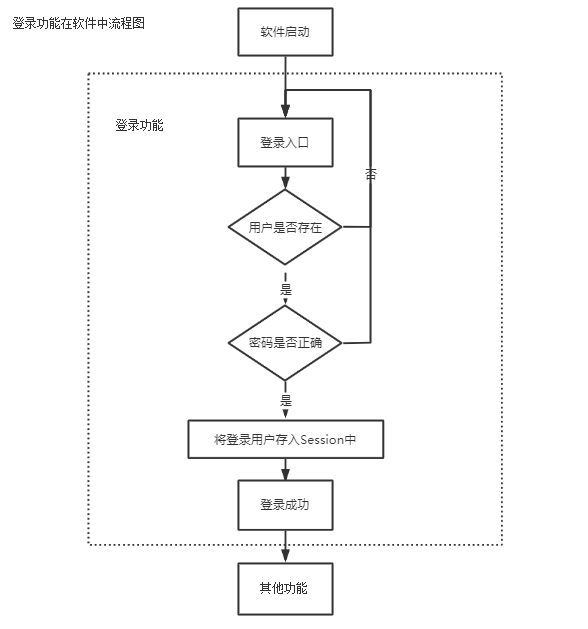
登录功能输出是从数据库中读取的数据或错误警告信息。

## 算法

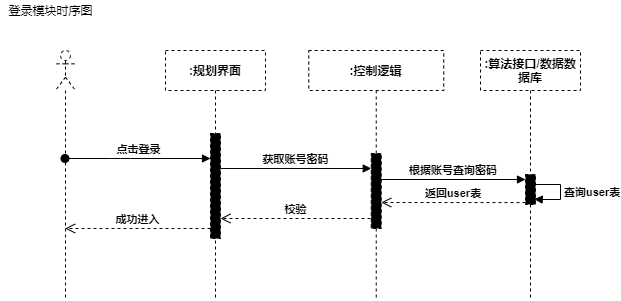
无。

## 程序逻辑

登录功能独立于其他功能，随软件启动而启动。



登录功能流程图



登录功能时序图

## 接口

登录功能主要包含ManualMainStore。其中重要接口详细说明如下：

/\*\*

\* @brief The ManualMainStore class

\* 登录功能ui和逻辑

\*/

*class* ManualMainStore：

/\*\*

\* @brief on\_pButton\_login\_clicked

\*/

void checkCalibrateBoardAndTool();

## 存储分配

登录功能信息存在本地数据库中：软件所在目录下：“sqlDemo”文件。

## 限制条件

1. 账号格式必须是标准字符串且长度1-19位。
2. 密码格式必须是标准字符串且长度6-19位。

## 测试要点

1. 用户名密码正确，登录；
2. 用户名正确，密码错误；
3. 用户名不存在；
4. 用户名1-19位有效字符验证，无效字符长度验证；
5. 密码6-19位有效字符验证，无效字符长度验证。

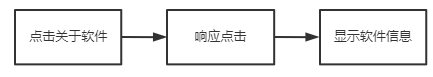
# 软件信息功能设计

## 功能描述

1. 软件信息功能管理软件基本信息，与其它功能耦合度较小。

## 功能

1. 点击软件顶部工具栏的关于按钮，弹出消息框显示软件基本信息（软件名称、软件版本、完整版本、注册证书编号、说明书编制日期）和软件厂商信息（公司名称、邮箱、生产许可证号、生产地址、注册地址）。



软件信息功能功能IPO图

## 性能

无。

## 输入项目

无。

## 输出项目

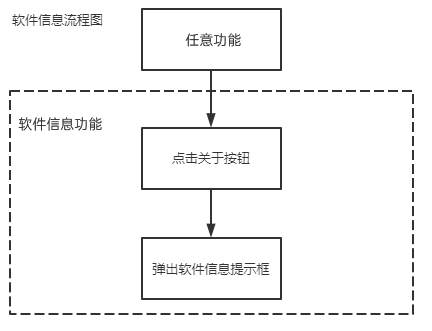
无。

## 算法

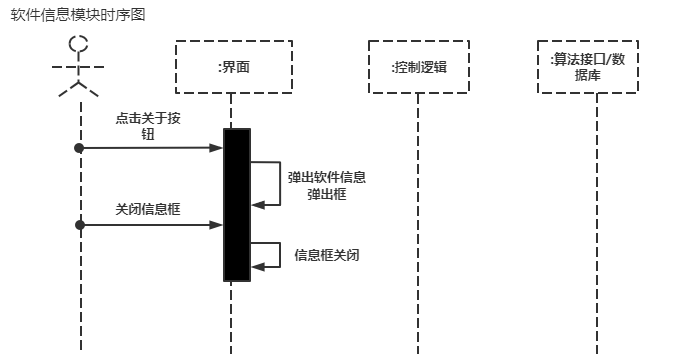
无。

## 程序逻辑

软件信息功能用流程图来展示本功能的程序逻辑。



软件信息功能在软件内的流程图



软件信息功能时序图

## 接口

规划功能包含Version.qml这个文件。其中重要接口详细说明如下：

/\*\*

\* @brief The version qml

\*

/\*\*

/\*\*

\* @brief onClicked

\* 初始化界面并加载软件信息

\*/

void onClicked()

## 存储分配

无。

## 限制条件

无。

## 测试要点

1. 关于页面正常显示与关闭；
2. 关于页面信息是否完整，版本、公司信息、软件名称等。

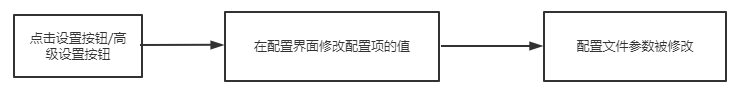
# 配置功能设计

## 功能描述

1. 配置功能通过管理软件配置文件实现对软件配置参数的管理与其它功能耦合度较小。

## 功能

1. 软件处于运行状态，点击软件顶部工具栏的设置/高级设置按钮，系统会弹出设置界面。修改设置界面上设置项对应的值，然后点击确定，配置文件会相应更新。



配置功能功能IPO图

## 性能

无。

## 输入项目

配置功能输入项目为系统配置文件和用户修改值。

## 输出项目

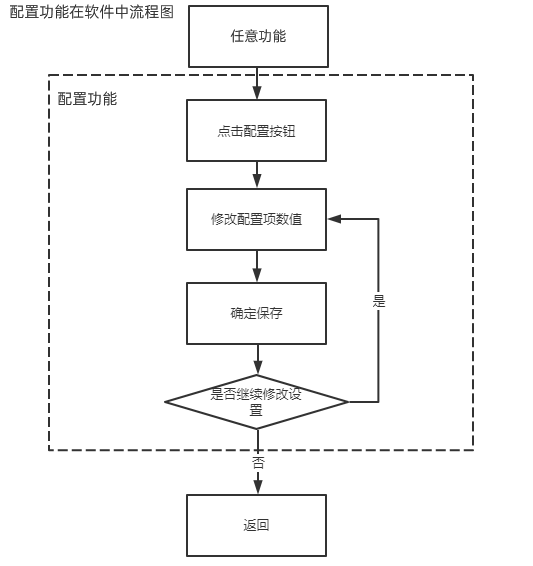
配置功能相关输出为对系统配置文件参数的修改。

## 算法

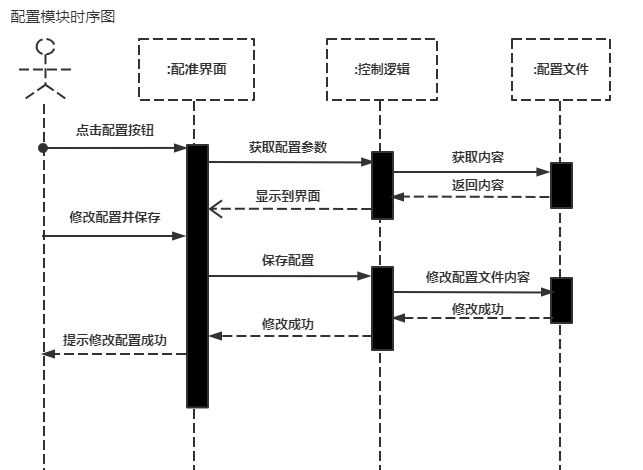
无。

## 程序逻辑

配置功能流程图来展示本功能的程序逻辑。



配置功能在软件内的流程图



配置功能时序图

## 接口

配置功能包含ParameterConfigDialog这个主要类。其中重要接口详细说明如下：

/\*\*

\* @brief The ParameterConfigDialogclass

\* 参数配置类

/\*\*

class ParameterConfigDialogclass：

/\*\*

\* @brief showEvent

\* 显示事件

\*/

void showEvent()

/\*\*

\* @brief readIniSettings

\* 读取配置信息

\*/

void readIniSettings()

/\*\*

\* @brief writeIniSettings

\* 将配置信息保存到ini文件

\*/

void writeIniSettings()

## 存储分配

配置功能本地文件存储内容：

配置文件路径："./arm\_config.ini","./para\_config.ini"文件。

## 限制条件

1. 对参数数值范围有限制；
2. 对设置的参数值类型有限制；

## 测试要点

1. 参数配置页面正常打开与关闭验证；
2. minThreshold maxThreshold minArea maxArea thresholdStep filter focalLength spacing radius normalize initialize Packet Head para2d para3d参数有效范围和无效范围输入测试；
3. 参数修改后，点确定，是否实时生效；
4. 参数修改后，点取消，参数修改失败。

# 日志功能设计

## 功能描述

1. 日志功能用来记录关键操作及关键结果，通过记录异常和警告来分析处理不可预见的错误，保证了软件网络安全的审核和网络节点鉴别能力；
2. 日志功能在软件中属于常驻内存、不可重入，不属于顺序处理且代码有覆盖要求。本功能与其他功能耦合性较高，软件所有功能均与日志功能有数据交换；
3. 日志可以导出到U盘，用来将记录的日志文件压缩并加密导出到U盘，保证了软件网络安全的健康数据存储保密性能力。

## 功能

1. 软件开启后记录关键操作步骤和异常信息；
2. 软件关闭时最后进入日志记录功能，保存关闭信息。

****

日志功能功能IPO图

## 性能

日志功能所有操作在后台进行，及时响应时间在毫秒级。

## 输入项目

日志记录功能输入项目如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| **1** | 操作流水 | 无 | string | 不为空 | 自动输入 | 软件关键操作步骤 |
| **2** | 警告、错误信息 | 无 | string | 不为空 | 自动输入 | 软件警告、错误信息 |

## 输出项目

日志记录功能输出项目如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输出方式** | **备注** |
| **1** | 日志条目 | 无 | log | 不为空 | 自动保存 | 软件运行后，日志实时更新 |

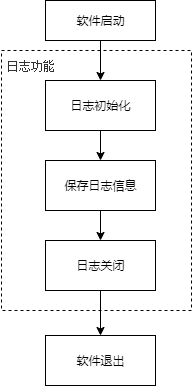
## 算法

无。

## 程序逻辑

日志功能独立于其他所有功能随软件启动而开启，随软件关闭而结束。

用日志功能在软件内的流程图来展示本功能的程序逻辑。



日志功能在软件内的流程图

## 接口

日志记录功能包含SystemLog、 MilestoneLog两个主要类。其中重要接口详细说明如下：

9.8.1. SystemLog

/\*\*

\* @brief The SystemLog class

\* 系统日志类

\*/

class SystemLog：

/\*\*

\* @brief writeLog 写日志

\* @param logStr 日志内容

\* @param \_flag日志等级

\*/

void writeLog(QString logStr, int \_flag);

/\*\*

\* @brief endLog 结束日志

\*/

void endLog();

9.8.2. MilestoneLog

/\*\*

\* @brief The MilestoneLog class

\* 里程碑日志类

\*/

class MilestoneLog ：

/\*\*

\* @brief writeLog 写日志

\* @param logStr 日志内容

\* @param \_flag日志等级

\*/

void writeLog(QString logStr, int \_flag);

/\*\*

\* @brief endLog 结束日志

\*/

void endLog();

## 存储分配

日志记录功能本地文件存储内容：日志文件保存在当前软件根目录下：“systemLog.html”和“milestoneLog.html”文件。

## 限制条件

无。

## 测试要点

1. 是否正常生成日志文件；
2. 是否正常输出日志文件；
3. 日志文件是否存在乱码；
4. 日志文件是否记录了用户所有操作记录；
5. 日志记录时间及内容准确性验证；
6. 日志记录用户行为可追溯性验证。

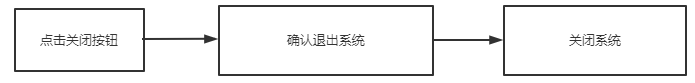
# 系统退出功能设计

## 功能描述

1. 系统退出功能用来退出系统并关机；
2. 系统退出功能在软件中属于常驻内存、可重入，不属于顺序处理且代码有覆盖要求。本功能与其他无耦合。

## 功能

1. 点击顶部工具栏的退出按钮，系统会弹出提示框提示是否退出系统，点击确认则系统关闭，点击取消按钮则可取消退出系统操作。



系统退出功能功能IPO图

## 性能

无。

## 输入项目

无。

## 输出项目

无。

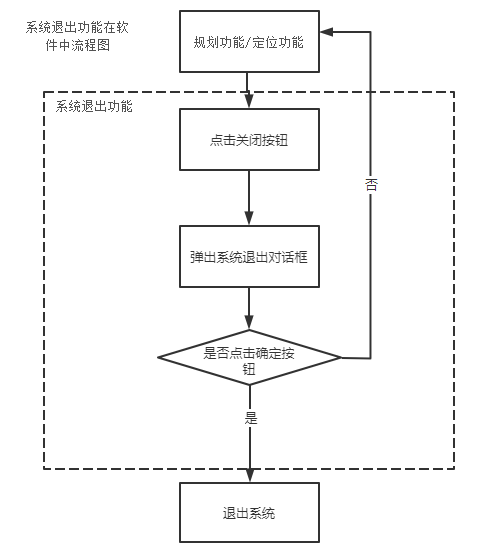
## 算法

无。

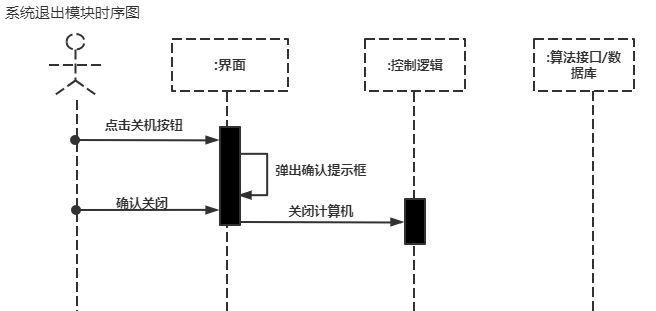
## 程序逻辑

系统退出功能独立于其他所有功能随软件登录而开启，随软件关闭而结束。

用系统退出功能在软件内的流程图来展示本功能的程序逻辑。



系统退出功能在软件内的流程图



系统退出功能时序图

## 接口

系统退出功能包含MainStore这个类。其中重要接口详细说明如下：

/\*\*

\* @brief The MainStore class

\* 主存储类

\*/

Class MainStore:

/\*\*

\* @brief invokeSystemExit

\* 调用系统退出

\*/

void invokeSystemExit()

## 存储分配

无。

## 限制条件

无。

## 测试要点

1. 关机按钮是否有效；
2. 是否关机页面内容是否合理；
3. 是否关机页面，点确认，系统关机；
4. 是否关机页面，点取消，返回原来页面。

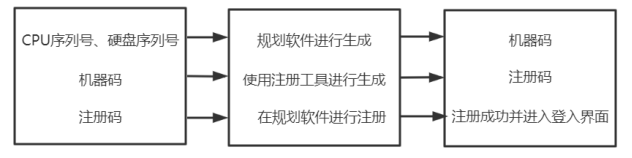
# 软件注册功能设计

## 功能描述

1. 为了防止软件被盗用，需要对软件进行保护；
2. 当系统软件在一台主机上安装后，会根据主机的硬件信息生成一串机器码；
3. 需要一个专用工具。针对机器码及设定的密钥，生成注册码；
4. 将注册码录入，注册成功后方能使用，此时也将注册码写入到配置文件中。

## 功能

1. 启动软件，读取配置文件中的注册码，验证是否正确；
2. 验证通过后，正常进入登录界面；
3. 首次安装等情况下，验证不成功，弹出注册对话框；其中机器码根据CPU系列号及硬盘序列号通过内部算法自动生成一串机器码；
4. 使用注册工具根据机器码，通过内部算法对应的注册码，将注册码输入注册界面进行注册，注册成功后将注册码写入配置文件中并进入登录界面。

软件注册功能功能IPO图

## 性能

无。

## 输入项目

1. 硬件信息：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| 1 | CPU序列号 | cpuId | QByteArray | 16~32位 | 自动获取 | / |
| 2 | 硬盘序列号 | diskId | QByteArray | 8~32位 | 自动获取 | / |

2.机器码：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| 1 | 机器码 | systemCode | QByteArray | 32位 | 手动输入 | / |

## 输出项目

1.机器码：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输出形式** | **备注** |
| 1 | 机器码 | systemCode | QByteArray | 32位 | 字符串 | / |

2.注册码：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输出形式** | **备注** |
| 1 | 注册码 | licensegenerated | QByteArray | 32位 | 字符串 | / |

## 算法

11.6.1. 注册算法

算法目的：

为了防止软件被盗用，需要对软件进行保护。

算法流程：

1. 规划软件启动后会检查配置文件license.ini中是否有32位的注册码，如果不存或注册码无效则被判断为未注册。

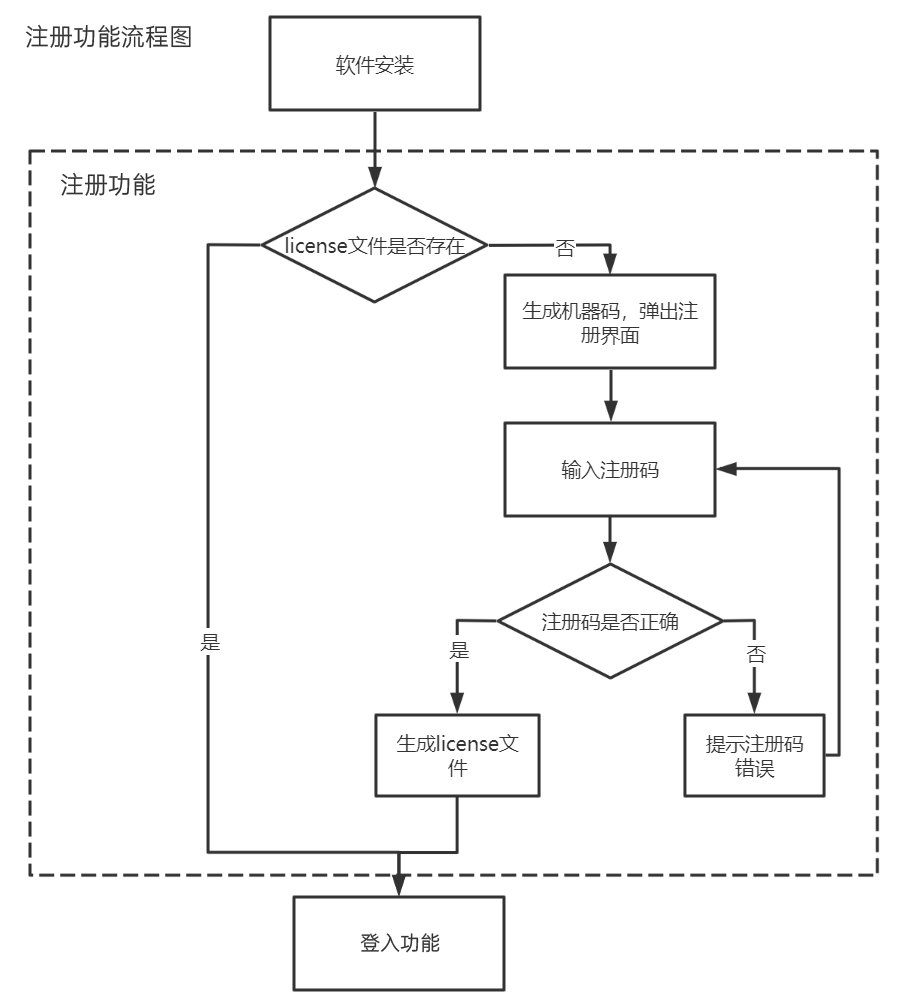
2. 如果规划软件未注册，则程序会取出本机cpu序列号和硬盘序列号拼接，然后经过MD5函数生成MD5码即机器码提供给用户，同时生成随机密钥。

3. 在注册工具中输入机器码，注册工具会将机器码和密钥字符串经过BKDR Hash生成32位的注册码提供给用户。

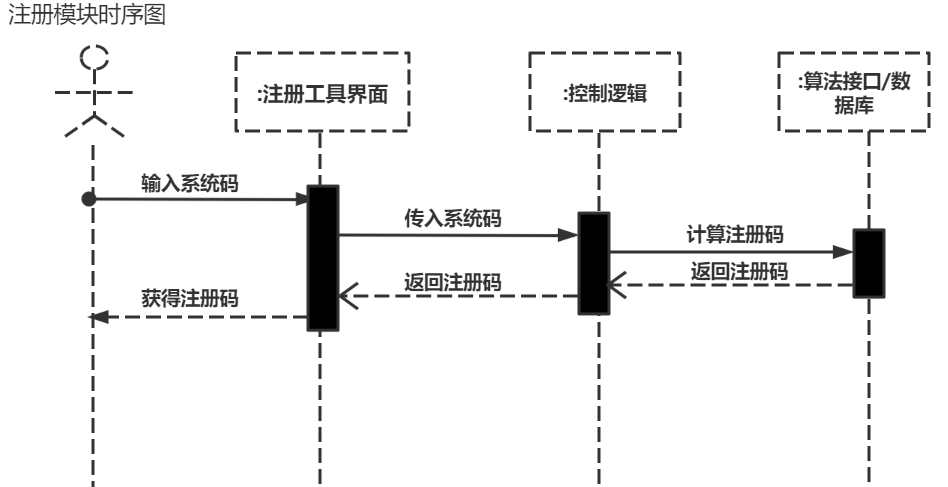
4. 将注册码输入到软件后，软件程序也会将机器码和密钥字符串经过BKDR Hash生成32位的注册码，然后与用户输入的注册码进行比对，如果相同，则注册成功。

5. 软件注册成功，程序同时也会将注册码写入到license.ini，这样下次登入程序时，执行步骤 1）判断为已注册，程序直接进入登入界面。

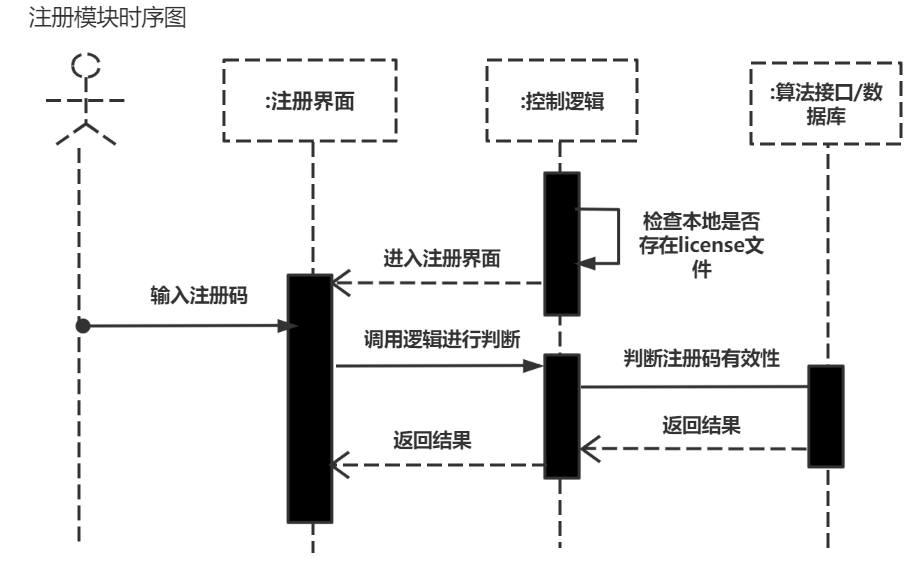
## 程序逻辑



软件注册功能在软件内的流程图



软件注册功能获取注册码时序图



注册功能使用注册码注册时序图

## 接口

注册功能包含SystemInfo、SoftLicense、Register三个主要类。其中重要接口详细说明如下：

11.8.1. SystemInfo

/\*\*

\* @brief The SystemInfo class

\*

/\*\*

/\*\*

\* @brief diskId

\* 获取硬盘序列号

\*/

void diskId()

/\*\*

\* @brief cpuId

\* 获取CPU序列号

\*/

void cpuId()

11.8.2. SoftLicense

/\*\*

\* @brief The SoftLicense class

\*

/\*\*

/\*\*

\* @brief systemCode

\* 是否生成机器码测试

\*/

void systemCode()

/\*\*

\* @brief isValid

\* 判断注册码的有效性测试

\*/

void isValid()

/\*\*

\* @brief generateLicense

\* 是否生成注册码测试

\*/

void generateLicense()

11.8.3. Register

/\*\*

\* @brief The Register class

\*

/\*\*

/\*\*

\* @brief registerLicense

\* 是否生成机器码测试

\*/

void registerLicense()

/\*\*

\* @brief cancelClicked

\* 点击取消注册测试

\*/

void cancelClicked()

/\*\*

\* @brief shutdownClicked

\* 点击关机测试

\*/

void shutdownClicked()

## 存储分配

软件注册功能本地文件存储内容：保存在当前软件根目录下：“./ license.ini”文件。

## 限制条件

无。

## 测试要点

1. 硬件正常，机器码可以生成；
2. 硬件去除，机器码生成失败；
3. 注册码正确，注册成功；
4. 注册码错误，注册失败；
5. 注册码格式正确，非本机生成，注册失败。

# 网络功能设计

## 功能描述

1. 网络通讯功能是本软件重要组成部分，主要用于和规划软件通讯。网络通讯功能基于自定义的协议实现了一套内部的客户端；
2. 网络通讯功能在软件中属于常驻内存、不可重入，不属于顺序处理且代码没有覆盖要求。本功能与其他功能耦合较小：收到导引软件的正侧位数据和将数据解码并发送给X-Ray分区功能，将定位信息加密发送给导引软件；
3. 网络通讯功能遵循内部加密协议，保证了软件网络安全的传输保密性和完整性能力。
4. 功能内部代码增加边缘性检测，详见接口部分。

## 功能

网络通讯功能独立于软件顺序流程，登录成功后客户端后台静默运行。

网络通讯功能详细功能：

1. DICOM图像传输：接收从C臂机发来的DICOM图像并显示；
2. 普通图像传输：将处理后的自定义格式图像发送到规划软件；
3. UR机械臂实时反馈端口(30003): UR机械臂关节角信息实时上传；
4. UR机械臂控制端口(30001): 下发机械臂关节角相关的控制命令；
5. 自定义端口(30000): 下发机械臂运动控制命令；
6. Dashboard端口(29999): 故障检测和解除；远程切换、加载程序，程序暂停和启动控制等。

## 性能

网络通讯功能性能如下：

1. 与C臂机传输Dicom影像时间不超过5s。
2. 与规划软件数据传递不超过1s。
3. 与机械臂通讯响应时间需做到毫秒级。

## 输入项目

网络通讯功能输入包括Dicom影像、C臂机参数、机械臂状态、规划螺钉信息。 相应内容详见对应功能。

## 输出项目

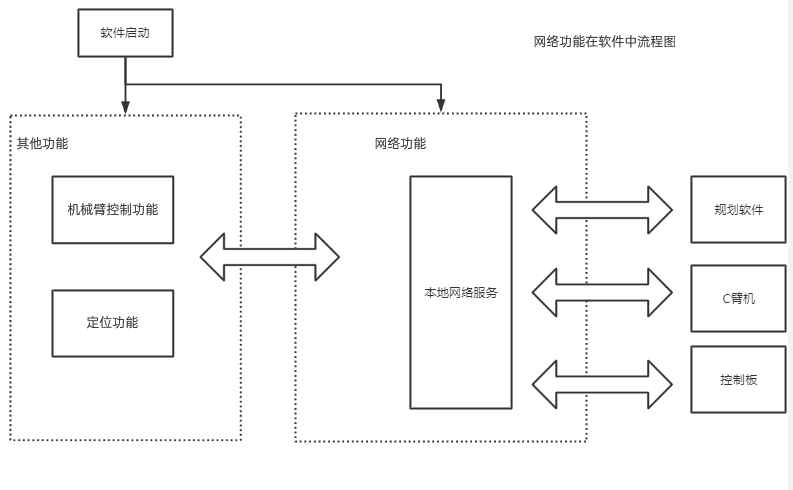
网络通讯功能输出包括X-Ray图像、图像对应相机信息、机械臂设置参数、机械臂控制参数。详细内容详见对应功能。

## 算法

无。

## 程序逻辑

软件登录成功后网络通讯功能自动启动，根据配置在后台静默运行。



网络功能在软件中的流程图

## 接口

网络通讯功能主要包含ArmURDashboardProcessor，其中重要接口详细说明如下：

/\*\*

\* @brief The ArmURDashboardProcessor class

\* 网络功能管理类

\*/

*class* ArmURDashboardProcessor :

/\*\*

\* @brief disconn 断开链接

\* @return

\*/

int disconn();

/\*\*

\* @brief connected

\* 保持链接

\*/

void connected();

/\*\*

\* @brief received

\* 收到数据

\*/

void received();

/\*\*

\* @brief recvTcpRead

\* 解析tcp包头

\*/

void recvTcpRead();

/\*\*

\* @brief recvTcpData

\* 解析tcp数据

\*/

void recvTcpData();

/\*\*

\* @brief onDisconn

\* 断开链接

\*/

void onDisconn();

/\*\*

\* @brief init 功能初始化

\* @param hostname

\* @param port

\*/

void init(*const* QString& hostname, *const* quint16& port);

/\*\*

\* @brief sendData 发送数据

\* @param arr

\*/

void sendData(QByteArray\* arr);

/\*\*

\* @brief receivedData 收到数据

\* @param byteArray

\*/

void receivedData(*const* QByteArray& byteArray);

/\*\*

\* @brief connectState 链接状态

\* @param state

\*/

void connectState(*const* int& state);

/\*\*

\* @brief sendDataResult 发送收到结果

\* @param result

\*/

void sendDataResult(*const* bool& result);

/\*\*

\* @brief isConnected 是否链接

\* @return

\*/

bool isConnected();

/\*\*

\* @brief ipAddr获取IP

\* @return

\*/

QString ipAddr();

/\*\*

\* @brief sendCmdPackages 发送命令包

\* @return

\*/

int sendCmdPackages(const QList<URDashboardCommandType>& cmd);

/\*\*

\* @brief sendSingleCmd发送一条命令

\* @return

\*/

int sendSingleCmd(URDashboardCommandType cmd);

/\*\*

\* @brief robotState获取机械臂状态

\* @return

\*/

bool robotState();

/\*\*

\* @brief processCommand加工命令

\*/

void processCommand();

/\*\*

\* @brief connectDashboard 连接面板

\*/

void connectDashboard();

/\*\*

\* @brief checkStatus检查状态

\*/

void checkStatus();

/\*\*

\* @brief startProgram 启动程序

\*/

void startProgram();

/\*\*

\* @brief play执行

\*/

void play();

/\*\*

\* @brief pause暂停

\*/

void pause();

/\*\*

\* @brief stop停止

\*/

void stop();

/\*\*

\* @brief quit退出

\*/

void quit();

/\*\*

\* @brief shutDown关闭

\*/

void shutDown();

/\*\*

\* @brief restart重启

\*/

void restart();

/\*\*

\* @brief connectToServer 连接到服务

\*/

void connectToServer();

/\*\*

\* @brief disconnectFromServer 断开服务连接

\*/

void disconnectFromServer();

## 存储分配

网络功能无本地存储，收到数据解密成功会发数据发送给机械臂控制功能处理，解密失败将直接丢弃。

## 限制条件

1. 开放端口不能被占用。
2. 本软件需要与规划软件、机械臂、C臂机处于同一个网段，并且相互之间网络畅通。
3. 收发数据符合机械臂通讯协议

## 测试要点

1. 稳定性测试：网络长时间连接；
2. 可靠性测试：不同数据格式及不同图像尺寸传输测试软件运行是否正常；
3. ip与规划软件不在同一网段，图片传输测试；
4. 端口被占用后，无法正常通讯，图片传输测试；
5. 查看传输数据是否加密处理；
6. C臂机批量发送200张图片测试。

# 机械臂控制功能设计

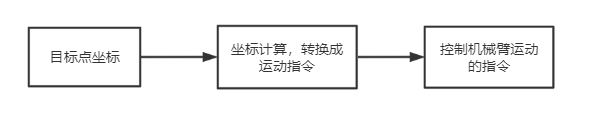
## 功能描述

1. 机械臂控制功能主要用来控制机械臂运动、停止，检测机械臂的状态；
2. 机械臂控制功能在软件中属于常驻内存、不可重入，属于顺序处理且代码没有覆盖要求。本功能与其他功能耦合较小。
3. 功能内部代码增加边缘性检测，详见接口部分。

## 功能

点击配置界面中的机械臂控制按钮进入机械臂控制界面，在此界面对机械臂各关节进行单独控制，此界面实时刷新机械臂各关节的角度信息；机械臂正位复位、机械臂侧位复位等功能，具体如下：

1. 关节控制：控制机械臂关节进行移动；
2. 启动机械臂：软件与机械臂建立通讯；
3. 关闭机械臂：软件与机械臂断开通讯；
4. 正位复位：控制机械臂运动到系统默认的正位姿态，保证软件网络安全的数据备份与灾难恢复能力；
5. 侧位复位：控制机械臂运动到系统默认的侧位姿态，保证软件网络安全的数据备份与灾难恢复能力；
6. 伸展机械臂：控制机械臂运动到系统默认的伸展姿态；
7. 收纳机械臂：控制机械臂运动到系统默认的收纳姿态；
8. 关节角信息获取：从机械臂获取每个关节的关节角信息，并展示；
9. 自由拖动：用户踩住脚踏开关的同时可以自由拖动机械臂，未踩住脚踏时无法对机械臂进行手动拖拽，如果点击其他机械臂功能按钮时会出现确认提示信息；
10. 禁止拖动：此状态下禁止用户手动拖拽机械臂。
11. 水平对齐：控制机械臂运动到水平对齐的姿态。

机械臂控制功能功能IPO图

## 性能

无。

## 输入项目

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输入方式** | **备注** |
| **1** | 目标点坐标 | 无 | double | 不为空 | 自动输入 | 无 |

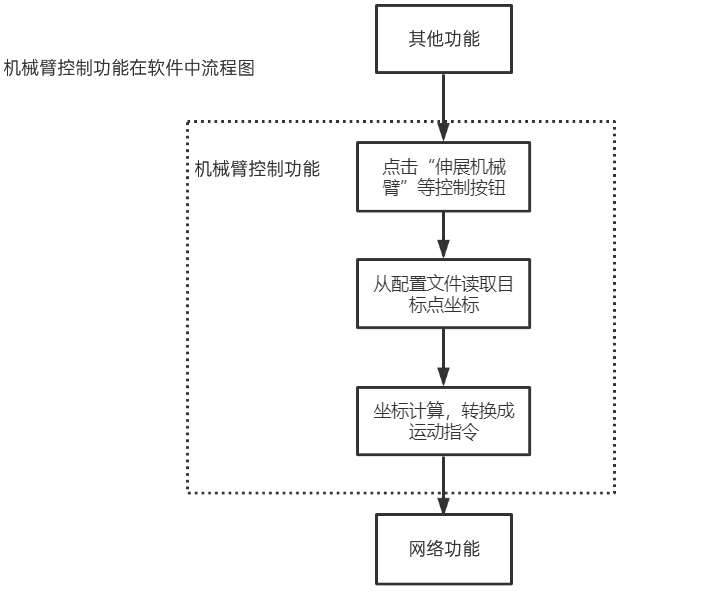
## 输出项目

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **序号** | **名称** | **标识** | **类型** | **有效范围** | **输出方式** | **备注** |
| **1** | 运动指令 | 无 | Qstring | 不为空 | 自动输出 | 供通信功能发送 |

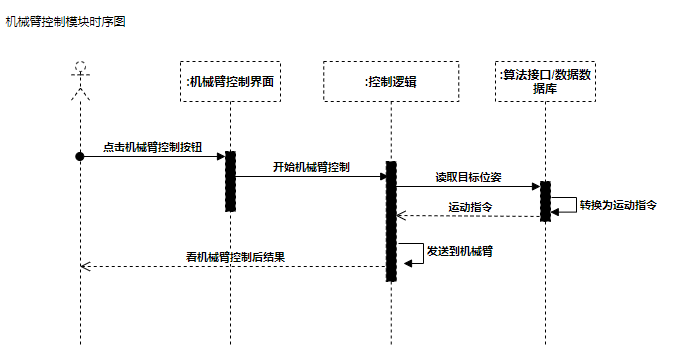
## 算法

无。

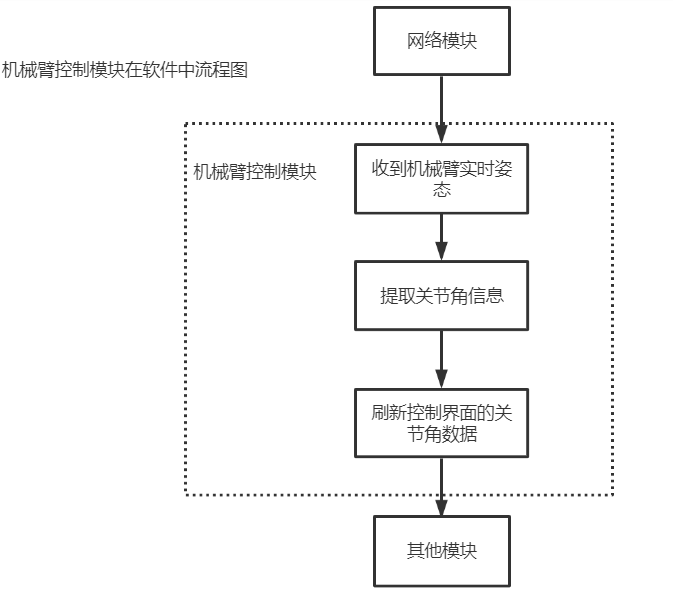
## 程序逻辑



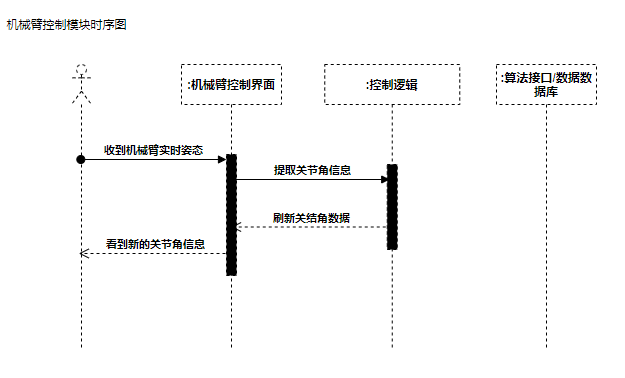
控制机械臂运动的流程图



控制机械臂运动时序图



获取关节角信息的流程图



获取关节角信息时序图

## 接口

校准功能包含ArmJointControl、GetArmJointProcessor两个主要类。其中重要接口详细说明如下：

13.8.1. ArmJointControl

/\*\*

\* @brief The ArmJointControl class

\* 机械臂关节控制类

\*/

class ArmJointControl:

/\*\*

\* @brief setPort

\* 设置端口

\*@param port端口

\*/

void setPort(int port);

/\*\*

\* @brief move

\* 运动

\* @return

\*/

void move(double x, double y, double z, double rx, double ry, double rz);

/\*\*

\* @brief setFetchStatus

\* 设置获取状态

\*@param isFetch 是否获取

\*/

void setFetchStatus(const bool& isFetch);

13.8.2. GetArmJointProcessor

/\*\*

\* @brief The GetArmJointProcessorclass

\* 获取机械臂关节信息类

\*/

class GetArmJointProcessor:

/\*\*

\* @brief initialize

\* 初始化函数

\*@param name 配置文件目录

\*/

void initialize(const QString & name);

/\*\*

\* @brief prepared 是否准备就绪

\* @return

\*/

bool prepared();

/\*\*

\* @brief startConnect 开始连接

\*/

void startConnect();

/\*\*

\* @brief disconnectUR断开连接

\*/

void disconnectUR();

/\*\*

\* @brief isConnected 是否连接

\* @return

\*/

bool isConnected();

/\*\*

\* @brief setPort

\* 设置端口

\*@param port端口

\*/

void setPort(int port);

/\*\*

\* @brief setIpaddress

\* 设置IP

\*@param ipaddress IP地址

\*/

void setIpaddress(const QString& ipaddress);

/\*\*

\* @brief connectUR

\* 连接UR

\*@param ipaddress IP地址

\*/

void connectUR(const QString& ipaddress);

/\*\*

\* @brief packageHeadUpdated

\* 更新包头

\*@param head 包头

\*/

void packageHeadUpdated(const QByteArray& head);

/\*\*

\* @brief initView

\* 初始化视图

\*@param show是否可见

\*/

void initView(bool show);

/\*\*

\* @brief setFetchStatus

\* 设置获取状态

\*@param isFetch 是否获取

\*/

void setFetchStatus(const bool& isFetch);

## 存储分配

机械臂控制功能本地文件存储内容：保存在当前软件根目录下：“./arm\_config.ini”文件

## 限制条件

1. 正位复位的前置条件是正位校准成功。
2. 侧位复位的前置条件是侧位校准成功。
3. 关节角信息获取成功的前置条件是报文包头格式正确。

## 测试要点

1. 非运动中，禁止拖动/自由拖动状态切换；
2. 运动中，禁止拖动/自由拖动状态切换；
3. 直接点击水平对齐，机械臂不运动；
4. 踩脚踏，点水平对齐，机械臂不运动；
5. 点水平对齐，踩脚踏，机械臂运动；
6. 机械臂别扭姿态，水平对齐，机械臂不运动；
7. 水平对齐中，松脚踏，再踩，继续运动；
8. 观察水平对齐完成姿态是否与期望结果一致；
9. 观察机械臂运动中是否发生自碰撞，路径是否是最短路径；
10. 机械臂伸展、收纳、正位复位、侧位复位是否到系统默认位置。