

介绍

Nidaqmx网站主要用于获取数据并实时画图. 同时有静态位置判断等功能

界面

1. 小组选择按钮
 - a. Group 1
 - i. 获取第一组数据, 所有功能将使用第一组数据计算
 - b. Group 2
 - i. 获取第二组数据, 所有功能将使用第二组数据计算
2. 数据展示界面
 - a. 获取数据
 - i. 从nidaqmx机器上获取数据, 在process上储存publisher_interval(1.5, 在utils里定义)秒的数据
 - ii. 通过zmq publisher/subscriber方式发送给画图工具, 使用时间戳为x轴, 读取的数据为y轴.
 - b. 画图
 - i. 将nidaqmx上发送的数据进行画图, 共16个subplot
 - ii. 将最新的50个数据点进行画图
 - iii. 共有4种展示方式
 1. Basic Mode: 4x4排列, 前16组数据
 2. Group Mode: 同个点的xyz为一列排序, 前16组数据
 - c. 存储数据
 - i. 使用一个process读取publisher发送的数据, 并实时写入到json内, 每次读取为新的一行, 最后使用read_data.py将json转换成csv文件
 - d. 使用方法:
 - i. 点击Store按钮开始收集, 点击Stop按钮停止收集, 文件将被储存在stored_data文件夹下stored_data_(加上当前时间).csv
3. 数据收集界面
 - a. 收集数据
 - i. 共18个接口数据, 每组收集16个接口数据, 更换小组选择按钮可以保留第一组的2个接口并合并到第二组数据中
 - ii. 可收集多组数据
 - iii. 各读取2秒钟publisher发送的数据
 - b. 画图
 - i. 将收集到的数据转换成图表
 - c. 比较数据
 - i. 将读取的数据与数据标准std对比, 如果相差过大则不采用
 - d. 计算静态位置
 - i. 使用两组数据进行计算静态位置并在结果页面展示
 - e. 导出数据
 - i. 将收集到的数据导出为csv文件

- f. 清空列表
 - i. 将收集的数据和图表清空
- g. 使用方法
 - i. 点击Collect收集数据
 - ii. 点击Calculate进行计算
 - iii. 点击Output将收集到的数据写入stored_data_(加上当前时间).csv文件
 - iv. 点击Clear清空收集到的数据和图表
- 4. 结果界面
 - a. 显示结果
 - i. 计算静态结果
 - 1. 展示计算的点与接收器的位置3d图
 - ii. 标定结果
 - 1. 5组结果, 每组由标定结果矩阵, 原始标定数据的3d图, 校正数据的3d图组成
 - 2. 应用标定结果, 将标定结果矩阵与实时数据进行计算得出新的数据
显示在展示页面上

工具

- 1. Zmq
 - a. 使用publisher发送数据
 - b. 使用subscriber接收数据
 - c. port: 5556 组成网址(localhost + port)在utils中定义
 - d. topic: 100 定义发送信息的标签 在utils中定义
- 2. Multiprocessing
 - a. Process 1 get_data 不断获取nidaqmx数据并使用zmq进行发送
 - b. Process 2 dash plotly 网页
 - i. Process 3 store data, 不断接收get_data发送的数据并将他们写入csv文件
 - 1. 使用multiprocessing value(finish)为while loop的判断条件
 - 2. 每次点击store_data按钮, 将生成一个新的process, 并将finish传进去
 - 3. 点击stop按钮时将改变finish, 停止loop, 并开始read_data转换
 - ii. Process 4 calibrate, 标定计算
- 3. Dash网页
 - a. 利用callback机制来实现自动更新, 按钮, 储存数据等功能
 - b. Components
 - i. group-slider: 切换小组, 0为Group 1, 1为Group 2
 - ii. tabs: 切换页面按钮, "index-tab"为数据展示页面, "collect-tab"为数据收集页面, "result-tab"为结果页面
 - iii. mean-lists: 储存平均数, 由mean-list组成, mean-list是长度为18的list
 - iv. collected-data: 储存收集到的数据
 - v. display-lists-content: 储存收集页面展示文字
 - vi. index_layout: 界面
 - vii. static-data: 储存calculate的结果用于不同界面间的交流

- viii. store-button-state: store功能是否在运行
- ix. result-number: 区分计算类型, 0为未进行任何计算, 1为calculate, 2为calibrate
- x. apply-state: 储存是否将标定结果应用到展示界面上, 同时用来存储按钮文字 "Apply Calibration" 为否, "Stop" 为是
- xi. calibration-type: 区分标定的小组, 0为Group 1, 1为Group 2
- c. 数据展示界面
 - i. Components
 1. basic-button: 普通排序按钮 4行4列, 隐藏legend
 2. group-button: 组队排序按钮, 将相似的4组使用相同颜色排成一行, 显示legend
 3. store-button: 储存按钮
 4. live-update-graph: 图像
 5. data-list: 保存y数据
 6. number-list: 保存x数据
 7. figure_number: 保存图像类型
 8. interval-component: 持续更新页面, 每500ms更新一次
- d. 数据收集界面
 - i. Components
 1. collect-button: 收集按钮1, 收集数据, 根据group-slider收集不同组数据, 收集第二组数据时将保留两组第一组的数据, 然后替换其他数据
 2. calculate-button: 计算按钮, 使用收集到的所有两组数据进行算法计算
 3. calibrate-button: 标定按钮, 使用第一组或第二组数据进行标定计算
 4. output-button: 导出按钮, 将收集到的数据导出为cvs
 5. clear-button: 清除按钮, 清除收集到的数据
 6. collect-counter: string, 统计收集到的组数
 7. collect-graph: 画图
 8. display-lists: 显示平均数
- e. 结果界面
 - i. Components
 1. apply-button: 将矩阵应用到实时数据上
 2. matrix-value: 显示计算得出的矩阵
 3. result-graph: 画图

文件

1. collected_data 收集数据的文件夹
2. stored_data 储存数据的文件夹
3. algorithm.py 静态位置计算算法
4. algorithm_utils.py 静态算法参数
5. main.py 网页主体
6. utils.py 网页参数