

XXX（校名） XXX（队名） 编制

YYYY年MM月 发布

目录

[1. 规则技术点复盘 2](#_Toc109729267)

[1.1 规则技术点分析回顾 2](#_Toc109729268)

[1.2 实际实现技术点 2](#_Toc109729269)

[1.3 异同原因分析 2](#_Toc109729270)

[1.4 解决方法分析和经验总结 2](#_Toc109729271)

[2. 技术方案复盘 3](#_Toc109729272)

[2.1 机械结构方案 3](#_Toc109729273)

[2.2 硬件方案 3](#_Toc109729274)

[2.3 软件方案 3](#_Toc109729275)

[2.4 算法方案 3](#_Toc109729276)

[2.5 测试方案 3](#_Toc109729277)

[3. 项目进度复盘 4](#_Toc109729278)

[4. 赛季人力安排复盘 5](#_Toc109729279)

[5. 预算复盘 6](#_Toc109729280)

[6. 技术方案参考文献 7](#_Toc109729281)

[附录 学术创新成果 8](#_Toc109729282)

# 规则技术点复盘

填写这里

## 规则技术点分析回顾

填写这里

## 实际实现技术点

填写这里

## 异同原因分析

填写这里

## 解决方法分析和经验总结

填写这里

# 技术方案复盘

## 机械结构方案

填写这里

## 硬件方案

填写这里

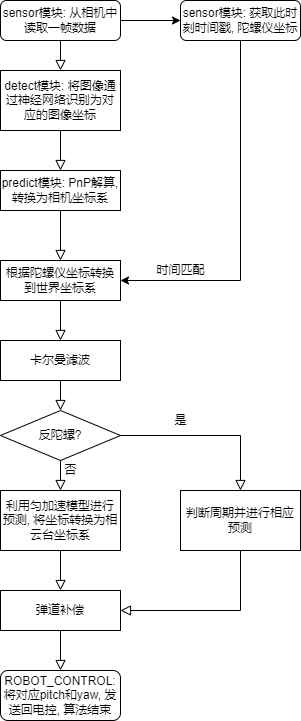
## 软件方案

填写这里

## 算法方案

**2.4.1 整体架构**

整个算法流程主要分为三个部分, sensor, detect和predict

* Sensor主要处理传感器的数据
* Detect负责识别装甲板
* Predict进行解算, 预测以及弹道计算等后处理步骤

**2.4.2 多线程并发库**

**简要介绍:** 采用ROS中消息模型的思想, 实现了Publisher/Subscriber信息模型, 用于各个模块之间的管道通信, 并且经过包装, 使得调用十分简便, 且线程锁的加入也使整个并发过程具有良好的线程安全性, 可以从容处理各个模块之间来回传输的数据

**原理解释:**

摘要: 采用了STL中的队列std::queue<T>实现了核心的信息存储, 以字符串作为每个收发管道的唯一标识符(以哈希表保存), 从Publisher中通过push(const T&)方法传入数据, 然后在需要数据的地方声明具有相同标识符的Subscriber并调用pop()方法获取数据

**详细描述:**

 创建管道

当使用构造函数Publisher<T>(const string&)或Subscriber<T>(const string&)时, 查找静态的无序表(unordered\_map<string, weak\_ptr>), 若没有结果, 则创建强指针然后返回(指针指向的类型是一个辅助类ObjManager)

 绑定到管道

与上一条情况相同, 但是如果此时在表中查找到已经存在的管道, 则直接将弱指针转换为强指针, 然后获得该管道

 发布信息

Publisher::push方法

如果管道不为空, 使用unique\_lock锁住, 然后将信息写入管道, 完成之后唤醒其他的锁(在Subscriber里面的condition\_variable)

 获取信息

Subscriber::pop方法

使用condition\_variable等待队列不为空, 收到唤醒之后使用std::queue::pop()获取队列的首元素

 析构

使用了智能指针中强指针和弱指针并用的方式, 在记录进哈希表中时保存弱指针, 而建立起与管道连接时使用的是强指针, 可以既保留哈希表中对应的地址, 又可以保证析构时所有的队列都能够成功释放

清空时会自动销毁队列中的元素

**使用示例:**

// in the first part...

Publisher<std::string> pub(“sensor\_data”);

pub.push(“hello”);

//in the other part...

Subscriber<std::string> sub(“sensor\_data”);

assert(“hello”, sub.pop());

**优点:**

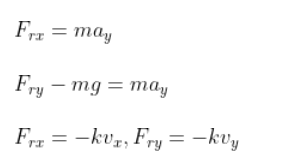
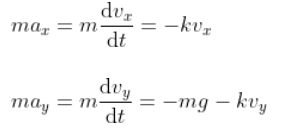
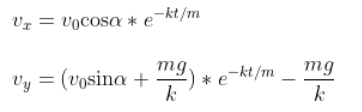
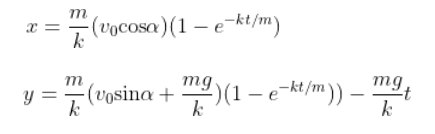
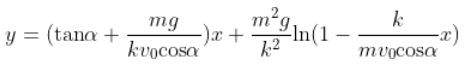
使用简单, 无需初始化即可使用, 语法也易于理解, 并且具有可靠的并发能力, 可以在多个线程中互相通讯, 而且也可以实现多个发布多个订阅

**缺点:**

Publisher必须在另一边Subscriber开始进行pop()之前声明, 对于采用主函数中开启多线程的操作会导致捕捉管道失败, 所以需要提前在开启其他线程之前开启发布器的线程

此外, 由于从内部取数据是move操作, 所以数据只能在进程间拿取一次, 对于需要多方接收统一数据的场景(比如赛场内录)就会需要再次发送，较为麻烦。

**2.4.3弹道模型**

* 为了让后续的能量机关自动发射时机解算更加精确，我们采用了带空气阻力的弹道模型
* 推导过程：
* 
* 
* 微分方程初始条件：
* 
* 解出初速度的x，y分量
* 
* 再通过位移与速度的微分关系
* 解出位移x，y
* 
* 最后联立获得轨迹方程
* 
* 考虑到通过确定的x,y来计算t与α会涉及到多次迭代，降低程序效率
* 我们在代码中通过枚举枪口抬起角度α与某范围的x计算y和子弹飞行时间t
* 计算出多条弹道轨迹后将α和t保存为一张二维表，考虑到弹道轨迹之间可能存在无法覆盖的空间，我们在两根弹道相邻轨迹之间的进行了二维线性插值，并且在查询弹道表的过程中也使用了二维线性插值。

**2.4.4 手眼标定**

**算法介绍：**

算法意在建立完善的机器人坐标系统，打通电控与视觉的联系，并标定相机光心的实际坐标，使得预测器EKF有坚实的算法和数据基础。具体表现为，能稳定解算目标在世界坐标系下的位置，机器人移动云台，相机从不同角度拍摄到目标，解算得到的目标在世界坐标系下的位置变换在可接受的范围内。

**算法原理：**

1. 坐标系定义：

Base系（世界坐标系）：原点在云台的旋转中心（yaw转轴与pitch转轴的近似交点）

云台系（gripper系）：原点与base系重合，到base系的旋转矩阵由陀螺仪imu解算获得。

相机系（cam系）：“固定”在云台系上，与云台系的旋转和平移都是定值，是待标定的内容。

1. 手眼标定：

手眼标定实际上是求解矩阵方程AX=XB，其中A是相机（cam系）前后两次空间变换的其次矩阵，B是机械臂末端（gripper）系前后两次变换的其次矩阵，通过多次求解该方程即可解出X。

在实现上采用opencv4的calibrateHandEye()函数进行求解。

图形用户界面

中度可信度描述已自动生成

其中:

R\_gripper2base，t\_gripper2base是云台系（gripper）相对于机器⼈基坐标系（base）的旋转矩阵与平移向量，平移向量为0，旋转矩阵由陀螺仪数据计算得到。R\_target2cam ， t\_target2cam 是标定板相对于相机的旋转矩阵和平移向量，由SolvePNP()运行PNP算法结合标定相机标定数据得到。

图示

描述已自动生成流程图：

1. 坐标转换：

通过PNP算法获得装甲板在相机坐标系（Cam）下的位姿（平移和旋转），通过旋转和平移变换解算到云台系（利用R\_cam2gripper和t\_cam2griper），再通过旋转变换解算到base系（通过R\_gripper2bass和t\_gripper2base）。这样，将在base系完成对目标的EKF跟踪，更具有鲁棒性和可解释性，同时使得目标的位置不收到云台自身运动状态的影响。

**算法验证**

电脑显示屏

描述已自动生成为了验证算法的效果，我们将机器人底盘固定，并且在其正前方约1.6m处固定放置了一块标定板，在持续晃动云台（改变pitch和yaw）的过程中，观察解算的标定板坐标值。其结果如下：

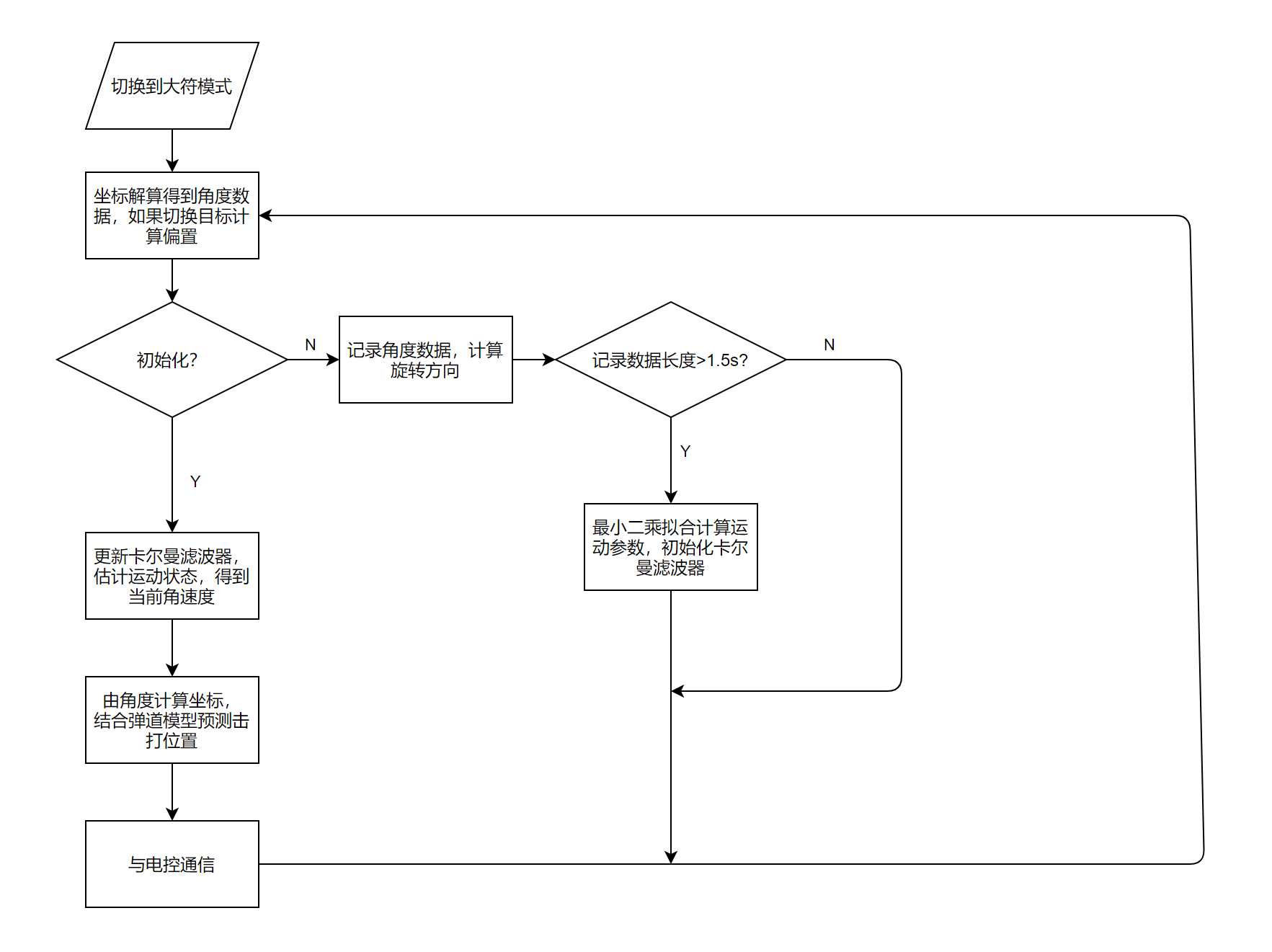
蓝色代表目标解算位置的距离（z方向位移），红色和绿色分别代表目标解算位置的x方向和y方向位移。（单位mm）

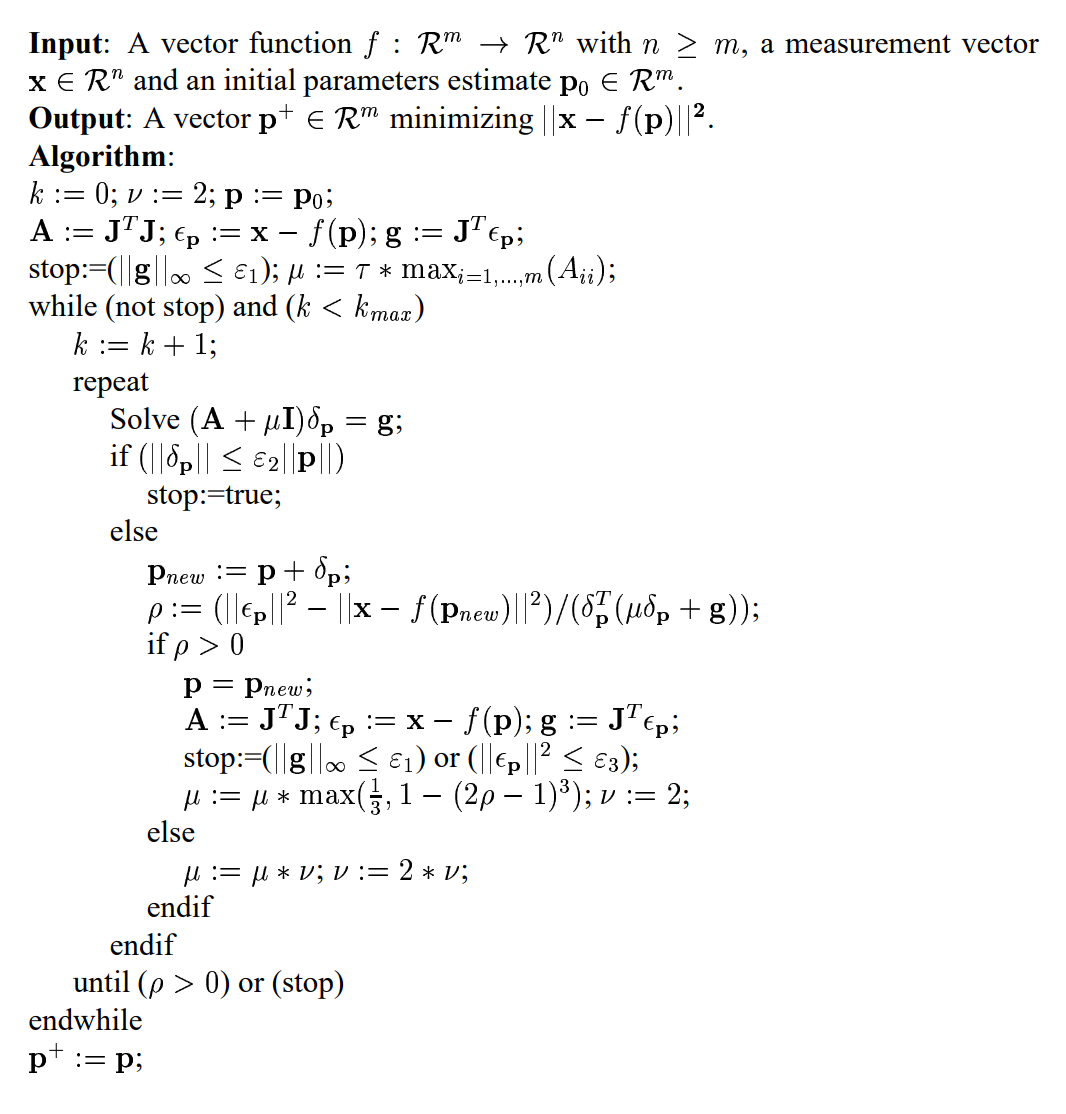
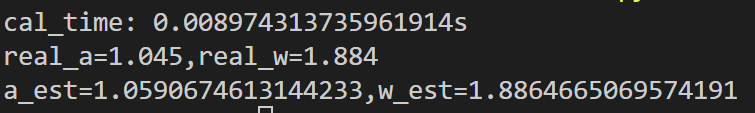
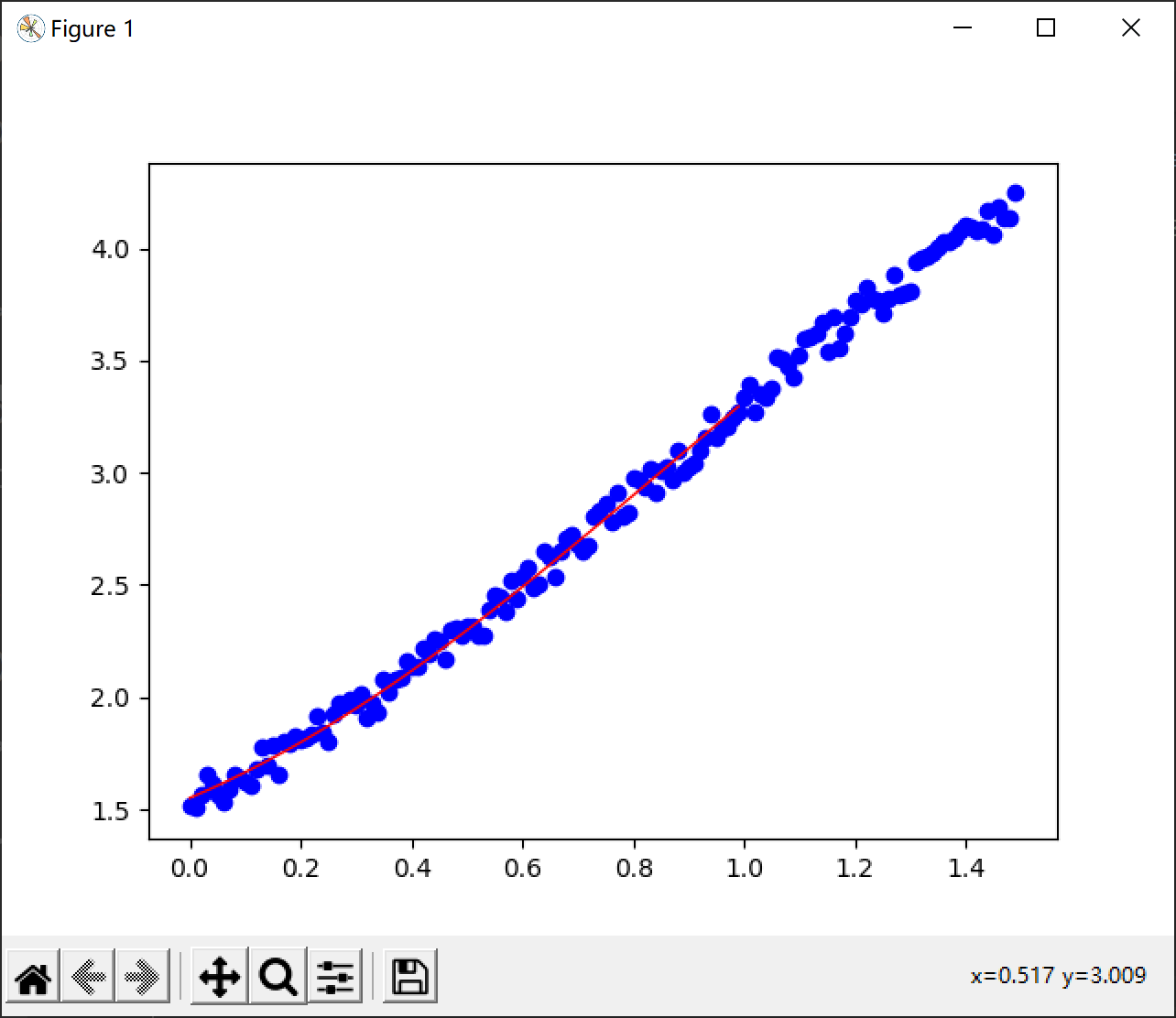
可以得出结论，我们的坐标系统和标定方法具有可行性和可以接受的准确性。

**2.4.5大能量机关预测**

1. **功能简介:**

预测器功能主要分为三部分：坐标解算、参数估计、滤波器预测

* 1. **坐标解算**
  2. 坐标解算部分在开始的时候通过对神经网络得到的特征点做pnp解算得到目标的相机坐标，结合从电控读取的imu数据转换到云台系，再通过预先标定好的相机系-世界系转换矩阵得到待击打目标在世界系下的坐标。计算旋转中心->装甲板中心的向量，并降维、归一化转化到单位圆上计算旋转角度，与上一次记录的数据比较判断是否切换目标，如果有记录偏置。
  3. 经过滤波器之后将预测出的角度量加上偏置，结合弹道模型计算待击打点。
  4. **参数估计**
  5. 考虑到滤波器收敛时间问题，先对运动参数进行估计，使用估计值初始化滤波器。通过对前1.5s采样得到的数据做非线性方程最小二乘拟合，计算旋转运动方程 spd=a\*sin(wt)+(2.090-a) 中的参数a，w和当前时间 t （实际上相当于三角函数的相位\phi =wt）。
  6. **滤波器预测**
  7. 对于非线性的运动方程，使用扩展卡尔曼滤波（EKF）算法跟踪和预测击打点。
  8. **预测器流程图**
  9. ****

1. **重要算法原理阐述**
2. **2.1 非线性方程最小二乘法拟合—LM算法**
3. 对于非线性的运动方程，使用扩展卡尔曼滤波（EKF）算法跟踪和预测击打点。
4. LM算法本质上就是在Gauss-Newton法的基础上添加了一个惩罚项，用信赖域法求解惩罚因子。该算法的主要思想是通过惩罚因子来调节步长。
5. 迭代公式如下：
6. 其中为多维向量的Hessian矩阵，近似计算公式为 ， 为雅可比矩阵，可以通过解析法或数值法求取；为多维向量的一阶梯度。
7. 当下降太快时使用较小的 ，使整个公式接近高斯牛顿法；当下降太慢时使用较大的 ，使整个公式接近梯度法。
8. 算法伪代码如下：
9. 
10. 在实现上，使用C++的矩阵库Eigen3完成基本的矩阵运算，使用数值法求解雅可比矩阵。
11. **2.2 拓展卡尔曼滤波**
12. 2.2.1 算法介绍
13. 卡尔曼滤波是一种已知系统数学模型，通过系统实际输入输出数据，将预测值和实际观测值进行数据融合，对系统状态做出最优估计的算法。
14. 扩展卡尔曼滤波（Extended Kalman Filter，EKF）是标准卡尔曼滤波在非线性情形下的一种扩展形式，它是一种高效率的递归滤波器（自回归滤波器）。EKF的基本思想是利用泰勒级数展开将非线性系统线性化，然后采用卡尔曼滤波框架对信号进行滤波，因此它只具有一阶精度，是一种次优滤波。
15. 2.2.2 算法流程
16. 状态转移方程：
17. 观测方程：
18. ,为过程噪声和观测噪声
19. 将方程线性化
20. , 为和的雅可比矩阵
21. 由此得到EKF算法预测和更新两个步骤：
22. 预测：
23. 更新：
24. 式中为状态转移误差协方差矩阵，为观测误差协方差矩阵，无下标量为观测量，下标表示预测量（predict），下标表示滤波后的估计量（estimate）。
25. 在实现上，使用C++的矩阵库Eigen3完成基本的矩阵运算，使用解析法求解雅可比矩阵。
26. 2.2.3 预测器使用的数学模型
27. 状态转移方程：
28. 观测方程:
29. re，im分别为装甲板在单位圆上的实部和虚部
30. 值得注意的是，由于EKF只具有一阶精度，所以在最后两个方程中角度变化用 来近似表示而没有采用积分的方式。经测试，所用状态转移方程与积分表示的方程滤波效果无明显差异，且雅可比求解更加简单
31. **算法性能与结果展示**
    1. **运动方程参数估计**
    2. 蓝色为采集到的角度数据，红色为最小二乘估计得到的函数曲线。
    3. 计算时间在0.01s以内；参数准确度：幅值误差±0.05，频率误差±0.01；使用拟合的参数初始化滤波器，收敛时间在0.1s以内。
    4. 
    5. **卡尔曼滤波跟踪和预测性能**

图表, 折线图

描述已自动生成蓝色为实际角度数据，黄色为使用滤波得到的状态量估计值预测1s以后的角度 ，误差在 ±2°以内，结果如下

可以得出结论，算法具有较好的预测性能。

## 测试方案

填写这里

# 项目进度复盘

填写这里

# 赛季人力安排复盘

填写这里

# 预算复盘

填写这里

# 技术方案参考文献

填写这里

附录 学术创新成果

填写这里

