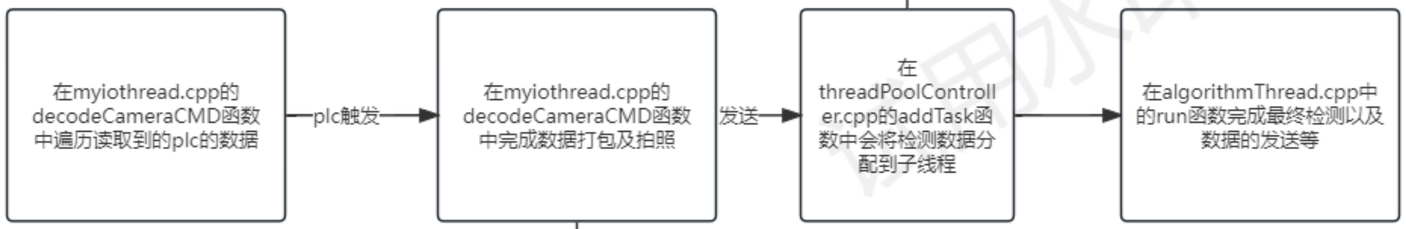
```cpp
void threadController::addTask(detData* data)
{
    cout << __LINE__ << "insert Data " << data ->timeCode<< endl;
    {
        algoThreadCls::datastruct _struct = bridge.back.json.algoThreads.getData(data->algoThreadName);
        data->algoThreadIndex = _struct.threadIndex;
        data->resShowIndex = _struct.showIndex;
        data->imgAmountMin = _struct.imgAmountMin;
    }
    {
        CameraListCls::datastruct _struct = bridge.back.json.camera.getData(data->cameraKeyName);
        data->oriShowIndex = _struct.showIndex;
    }
    insertQueueMutex.lock();
    //insertQueue.push_back(data);
    int threadId = data->algoThreadIndex;
    if (threadId == -1)
    {
        cout << "threadId return -1 ,algoName " << data->algoThreadName.toStdString() << endl;
        return;
    }
    algoThreadList[threadId]->addDetTask(data);
    insertQueueMutex.unlock();
    //this->start();
}
```

从json中获取数据
线程index
展示结果图的index
最小图片数量
原图展示index
发送检测数据到子线程
没啥问题不需要修改




```cpp
if (checkTrigger(8060, 1, byte) == true)//2D-1 8060正极2
{
    int snPos = 8700;
    bridge.back.sendFinsMsg("write_DM8060_0");//触发信号清零
    bridge.back.sendFinsMsg("write_DM8061_1");//开始检测信号
    ////////数据初始化///////
    detData* data = dController.insertDetData("2D-P-2", "2D-P-Thread", workThread->snDatas.readSnFromQByteArray(byte, snPos, true, true, 0), 0);
    //
    qDebug() << "[ Station 10 ] " << QString::fromUtf8("2D-1第一次拍照开始 timeCode:") << data->timeCode;

    ////////SN码读取操作//////
    if (!data->SNcode.compare(snErrorsnError))//查不到SN,送到snWaittingVec继续查询
        workThread->snDatas.snWaittingVec.push_back({ data->timeCode ,data->SNcode,snPos });
    ////////拍照///////////////
    threadlist[0]->takeImg(data->timeCode, MyThread::_2dSpecialP, "", data, "write_DM8062_1");
    workThread->addTask(data);//添加任务
    emit SendStationLabel(1);//播放动画
}
```

初始化检测数据
无用
线程名
sn
无用
固定
拍照模式
固定
取完图后发送的fins数据
将数据发往检测线程

```cpp
while (detQueue.size() > 0)
{

    detQueueMutex.lock();
    detData* data = detQueue.dequeue();//取出数据
    detQueueMutex.unlock();
    if (waitForImg(data) == false)
    {
        qDebug() << "取图失败";
        cout << "get img err" << endl;
        break;
    }
    oriSaveThread->saveOriImg(data);//保存原图
    bridge.front.showImg("Camera", data->imgs[0], data->oriShowIndex);//显示原图
    int threadId = data->algoThreadIndex;//获取线程id
    switch (threadId)
    {
    case 0:
        _2D_P_Algo(data);
        break;
    case 1:
        _2D_N_Algo(data);
        break;
    case 2:
        _3D_Algo(data);
        break;
    }
    if (data->resultImgs.size() > 0)
    {
        bridge.front.showImg("Algorithm", data->resultImgs[0], data->resShowIndex);//显示原图
        resSaveThread->saveResImg(data);//保存结果图
    }
    if (!data->SNcode.compare(snDatas->snErrorsnError))//无sn阻塞
        data->SNcode = snDatas->waitForSn(data->timeCode);
    //cout << "algoEnd" << data->algoThreadIndex;
    sqlDataStruct sqlDataStruct;//定义sql数据
    sqlDataStruct.SN = data->SNcode;//
    for (auto sqlData : data->sqlVec)//将算法中的数据导入到数据结构中
        sqlDataStruct.dataVec.push_back(sql->sqlDatasInit(sqlData.first, sqlData.second));
    sql->addData(sqlDataStruct); // ...
    detDataController* dController = data->dController;
    data->dController->deleteDetData(data->timeCode);
}
QThread::msleep(10);
```

根据线程id使用算法
等待图片
这段一般不需要改