**相控阵三维声学摄像声纳系统**

**客户端详细设计**

**编 写 人：黄余格**

**编写时间：2014-11-20**

**浙江大学嵌入式系统研究中心**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | 章节名称 | 修订内容简述 | 修订日期 | 修订前  版本号 | 修订后  版本号 | 修订人 | 批准人 |
| 1 |  | 创建 | 2014-12-11 | V0.0 | V0.0 | 黄余格 |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**修订页**

目录

[1 引言 4](#_Toc406098144)

[1.1 需求与背景 4](#_Toc406098145)

[1.2 相关技术介绍 4](#_Toc406098146)

[1.3 参考资料 4](#_Toc406098147)

[2 总体设计 5](#_Toc406098148)

[2.1 系统应用架构 5](#_Toc406098149)

[2.2 程序模块架构 6](#_Toc406098150)

[2.3通信协议设计 6](#_Toc406098151)

[2.3.1控制命令协议 6](#_Toc406098152)

[2.3.2声纳数据传输协议 8](#_Toc406098153)

[2.4 配置文件设计 9](#_Toc406098154)

[3 业务流程 10](#_Toc406098155)

[3.1 程序初始化 10](#_Toc406098156)

[3.2在线状态工作流程 10](#_Toc406098157)

[3.2离线状态工作流程 11](#_Toc406098158)

[3.3自检状态工作流程（原始数据整合）？？？？ 12](#_Toc406098159)

[4 可维护性设计 13](#_Toc406098160)

[4.1 调试日志方案 13](#_Toc406098161)

[4.2调试手段 13](#_Toc406098162)

[5 问题与风险 13](#_Toc406098163)

# 1 引言

## 需求与背景

实时三维声纳成像是革命性的声纳成像技术，它能提供即时的全方位三维场景图像，并支持动静目标检测，可广泛用于还有石油及天然气开采，海洋生态检测及国防军事等多个领域。相控阵三维摄像声纳是一种新型的成像声纳，它基于相控阵技术实现被测目标的动态三维图像重建，它的研究与开发对我国海洋探测及军事布防能力的提高具有重要意义。

工控客户端软件需要完成控制声纳状态、网络实时数据接收、GPS&姿态传感器信息转换、3D数据显示/拼接和3D可视化显示等功能。基于以上需求，设计了一套相控阵三维声纳的客户端软件。

## 1.2 相关技术介绍

本客户端软件基于qt4.8.6，vtk6.2.0开源软件进行开发，采用C++语言。

Vtk是一个开放资源的免费软件系统，主要用于三维计算机图形学，图像处理及可视化。

Qt是跨平台C++图形用户界面应用程序开发框架。

关于各项技术的介绍参见相关技术手册

## 1.3 参考资料

《VTK user’s guide》

…..

# 总体设计

## 2.1 系统应用架构

根据相控阵三维声纳实时摄像系统的实际使用需求，可将系统抽象成如下的示意图。



图1相控阵三维声纳摄像系统应用架构示意图

如上图所示，相控阵三维声纳摄像系统主要由声纳相控阵下位机子系统，GPS，姿态仪，工控机组成。客户端软件运行在工控机上，通过以太网与下位机相连，通过两个232串口获取GPS和姿态仪数据。其中客户端软件主要完成控制下位机的运行状态；网络数据接收及与GPS、姿态仪数据的整合，经过运动和姿态补偿实时显示单帧数据；多帧拼接形成三维场景图像显示；对离线数据进行后处理场景重建并显示等功能。本文主要对客户端软件的详细设计进行描述。

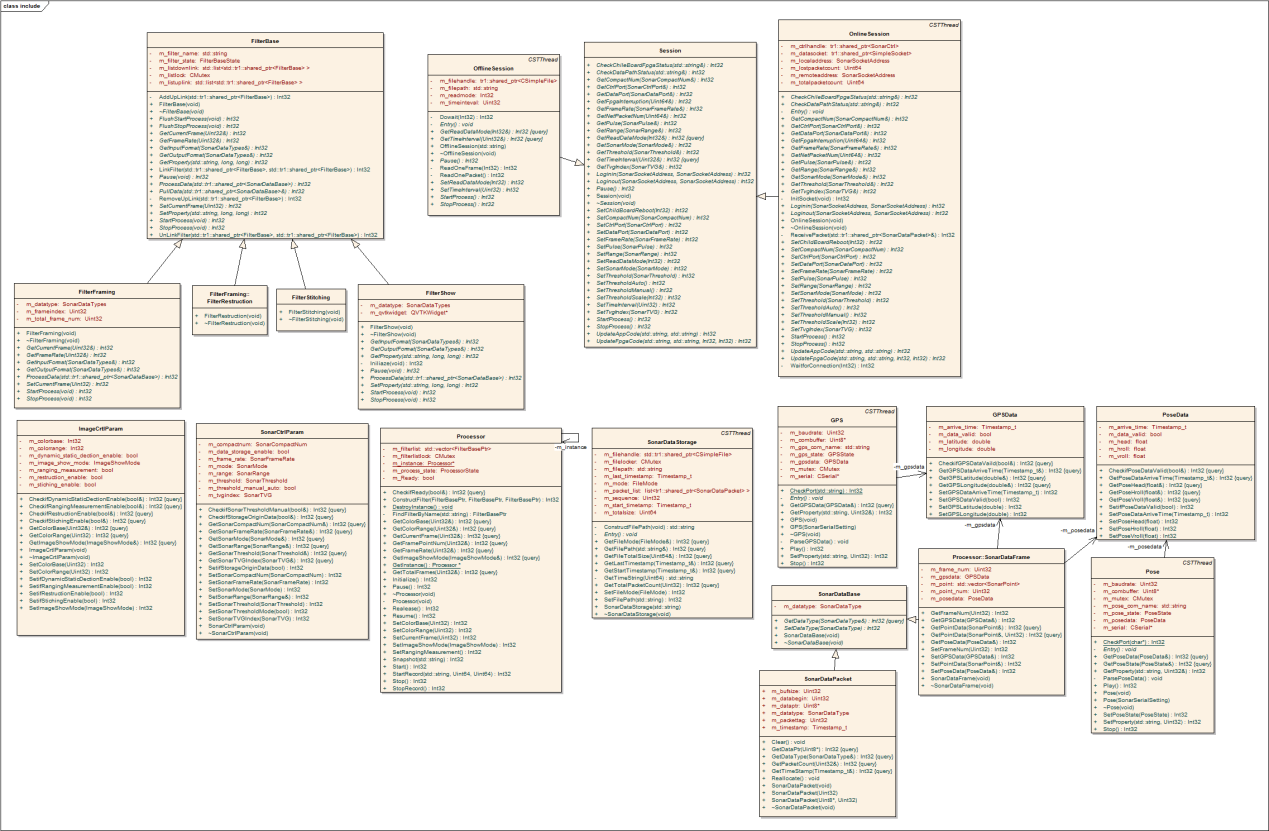
## 2.2 程序模块架构



图 2客户端程序模块架构

如图2所示为三维声纳客户端程序的模块架构示意图。根据系统的应用需求，业务类Sonar负责维护用户界面与实际处理之间的交互，业务类维护下方所有的模块初始化及其他必要的操作，将用户下达的指令转交给对应模块处理。其中三维数据处理模块现包含单帧重建，多帧拼接，动态检测三个子模块，根据用户要求确定是否开启，将相关算法处理过程抽出形成一个算法库，供以上三个模块调用。

UML类图如下图所示：



## 2.3通信协议设计

本系统的通信主要发生在客户端与下位机交互之间，故本通信协议包括两部分，一种是下位机上传的声纳数据格式，另一种是客户端控制下位机状态的控制命令。

根据业务需求，通过TCP链路发送控制命令，通过UDP链路传输声纳数据。

### 2.3.1控制命令协议

客户端向下位机下发的声纳控制命令包分为两种，设置参数命令和获取参数命令。每种命令都是一个命令包。每个包有两部分组成，命令头和外带数据。

命令类定义如下：

|  |
| --- |
| class SonarCommand  {  public:  typedef enum  {  OrderBegin=0x9F,  SysMode=0xA0,  SysReset,  SysStart,  SysStop,  SysUpdate\_App,  SysUpdate\_Fpga, //No use  SysSetControlPort,  SysGetControlPort,  SysSetDataPort,  SysGetDataPort,  SysSetThresholdAuto,// new  SysSetThresholdManual,// new  SysSetThreshold,  SysGetThreshold,  SysSetTVG,  SysGetTVG,  SysSetTVGSingle,  SysGetTVGSingle,  SysSaveTVG, // new  SysSetTVGENUM, // new  SysGetTVGENUM,// new  SysSetRange,  SysGetRange,  SysSetChannel,  SysGetChannel,  SysSetFrustum,  SysGetFrustum,  SysSetPulse,// new  SysGetPulse,// new  SysUpdatePPCSparFpga, /\*update the Spartan on Mother Board\*/  SysUpdateVirtex5Fpga, /\*update the Virtex-5 fpga\*/  SysUpdateMAINSparFpga, /\*update the Spartan on Children Board\*/  SysVerifyVirtex5Fpga, /\*verify the program of Virtex5\*/  SysGetMAINSparStatus, /\*get the runnning status of 48 MAIN Spartan Fpga\*/  SysGetDataRecvStatus, /\*get the status of data receiving of Main Board\*/  SysSetWorkMode, /\* work on normal mode\*/  SysSetCompactNum, /\*set compact num\*/  SysSetFrame, /\*set data frame\*/  SysSetStandby1, /\*reserved....\*/  SysSetStandby2,  SysSetStandby3,  SysSetStandby4,  OrderEnd  }SonarCmdName;    SonarCommand(void);  ~SonarCommand(void);  //设置命令tag号  Int32 SetCmdtag(const Int32 cmdtag);  //获取命令tag号  Int32 GetCmdtag(Int32 &cmdtag)const;  //设置命令号  Int32 SetCmdName(const SonarCmdName cmdname);  //获取命令号  Int32 GetCmdName(SonarCmdName &cmdname)const;  //设置命令外带数据长度  Int32 SetCmdDataLength(const Int32 data\_length);  //获取命令回复数据长度  Int32 GetCmdDataLength(Int32 &data\_length)const;  //设置命令外带数据  Int32 SetCmdData(const Uint8 \*buffer,Int32 data\_length);  //获取命令回应外带数据  Int32 GetCmdData(Uint8 \*buffer,Int32 &data\_length)const;  //获取命令总长度  Int32 GetCmdLength(Int32 &cmdlength);    //命令转为字符串  Uint8 \* CmdToChar();  //字符串转成命令  Int32 CharToCmd(Uint8 \*buffer);  private:  Int32 m\_cmdtag;  union  {  SonarErrCode upstatus;  SonarCmdName cmdname;  }SonarDownUp;  Int32 m\_data\_length;  Uint8 \* m\_data;  }; |

其中各参数含义如下：

m\_cmdtag：表示为命令及回应序号，由1开始单增，越界后回卷到1。回应序号需要同命令序号对应。

SonarDownUp：联合体，分别表示命令或者回应。当设置/获取参数成功时返回SysSuccess，否则返回SysFailed。当系统当机时返回SysErr，SysErrAndRestart，SysErrAndShutdown，通知客户端目前状态及应对。

SonarCmdName：命令号，表示下发的命令具体表示含义。

m\_data\_length： 表示外带数据长度。

m\_data：外带的有效数据数据，格式如控制数据包数据内容

命令号定义如下：

typedef enum

{

Order\_Begin=0x9F,

Sys\_Mode=0xA0,

Sys\_Reset,

Sys\_Start,

Sys\_Stop,

Sys\_Update\_App,

Sys\_Update\_Fpga, //No use

Sys\_Set\_ControlPort,

Sys\_Get\_ControlPort,

Sys\_Set\_DataPort,

Sys\_Get\_DataPort,

Sys\_Set\_Threshold\_Auto,// new

Sys\_Set\_Threshold\_Manual,// new

Sys\_Set\_Threshold,

Sys\_Get\_Threshold,

Sys\_Set\_TVG,

Sys\_Get\_TVG,

Sys\_Set\_TVG\_Single,

Sys\_Get\_TVG\_Single,

Sys\_Save\_TVG, // new

Sys\_Set\_TVG\_ENUM, // new

Sys\_Get\_TVG\_ENUM,// new

Sys\_Set\_Range,

Sys\_Get\_Range,

Sys\_Set\_Channel,

Sys\_Get\_Channel,

Sys\_Set\_Frustum,

Sys\_Get\_Frustum,

Sys\_Set\_Pulse,// new

Sys\_Get\_Pulse,// new

Sys\_Update\_PPC\_Spar\_Fpga, /\*update the Spartan on Mother Board\*/

Sys\_Update\_Virtex5\_Fpga, /\*update the Virtex-5 fpga\*/

Sys\_Update\_MAIN\_Spar\_Fpga, /\*update the Spartan on Children Board\*/

Sys\_Verify\_Virtex5\_Fpga, /\*verify the program of Virtex5\*/

Sys\_Get\_MAIN\_Spar\_Status, /\*get the runnning status of 48 MAIN Spartan Fpga\*/

Sys\_Get\_Data\_Recv\_Status, /\*get the status of data receiving of Main Board\*/

Sys\_Set\_Work\_Mode, /\* work on normal mode\*/

Sys\_Set\_Compact\_Num, /\*set compact num\*/

Sys\_Set\_Frame, /\*set data frame\*/

Sys\_Set\_Standby1, /\*reserved....\*/

Sys\_Set\_Standby2,

Sys\_Set\_Standby3,

Sys\_Set\_Standby4,

Ctrl\_End,

}DownOrder;

### 2.3.2声纳数据传输协议

当PC向下位机发出的开始采样命令得到回应后，在数据端口接收声纳数据。每次接收SonaData结构体数据包，其中dataTag从0开始，单增回卷，dataLength表示SonaPoint数据的个数。point携带声纳数据。

当PC向下位机发出的停止采样命令得到回应后，停止在数据端口上接收。

原始声纳数据点类定义如下：

|  |
| --- |
| class SonarOriginPoint  {  public:  SonarOriginPoint(void);  ~SonarOriginPoint(void);  //获取i值  Int32 GetOriginI(Uint32 &i)const;  //获取j值  Int32 GetOriginJ(Uint32 &j)const;  //获取k值  Int32 GetOriginK(Uint32 &k)const;  //获取帧号  Int32 GetOriginFrameNum(Uint32 &framenum)const;  //获取强度值  Int32 GetOriginIntensity(float &intensity)const;  private:  float m\_intensity; //强度值  Uint32 m\_position; //包含i，j，k，帧号  }; |

## 2.4 配置文件设计

客户端程序维护一份其独占的配置文件，文件以XML格式书写，主要保存的配置项暂有：

1. 默认下位机地址:IP,端口
2. 自动连接选项：是，否

# 3 业务流程

本节将对系统的各个业务流程进行描述。

## 3.1 程序初始化

程序打开默认处于在线状态，重建拼接等选项不勾选。

程序初始化的主要任务有：

1初始化一个在线业务Sonar总体类对象。该类包含ImageControlParam及SonarControlParam对象；Process类对象，Process类对象初始化时创建FilterFraming，FilterShow对象，加入其维护的列表中；SonarDataStorage类指针；Session指针，初始化时指向OnlineSession对象，OnlineSession对象维护UDP连接，TCP命令连接； Pose对象；GPS对象。

2 根据默认参数初始化用户界面

3读取配置文件中的默认IP端口，根据是否自动连接，如果是则根据默认IP、端口创建SonarCtrl对象，将其指针赋给OnlineSession类中的SonarCtrl对象指针，进行连接，如果成功，标记连接状态为成功，获取下位机默认工作参数，送到显示界面显示，如果不成功，则提示自动连接失败，标记连接状态为未连接。

## 3.2在线状态工作流程

在线状态下，有以下任务可能进行。

1 如果自动连接失败，则无法进行其他操作，只能选择连接设备，根据输入IP和端口，创建SonarCtrl对象，尝试连接，连接成功，则将其指针赋给OnlineSession类，标记连接状态为已连接，解锁其他在线操作。

2 如果自动连接成功或者连接设备成功后可以选择断开设备连接，此时将连接状态标记为未连接，将OnlineSession中的类指针重置

3 在连接状态下，对声纳参数区的任何选项进行操作，例如量程选择，与原先创建的SonarControlParam类中对应参数进行比较，如发生变化，则通过已建立连接的SonarCtrl对象下发对应的命令，进行参数设置，并更新SonarControlParam对象。

4 在在线状态下，对图像输出区域内的选项进行操作时，同原先创建的ImageControlParam对应参数比较，如发生变化，则进行对应操作，并更新ImageControlParam对象。此处的操作与所选对象密切相关。

1. 如果选择存储，则根据选择的路径设置SonarDataStorage类中的路径，并将ImageControlParam对应项修改。
2. 如果选择颜色区域的基值和范围，则只需调用相应FilterShow类的对应接口即可，并修改ImageControlParam对应项。
3. 默认外观为点，如果想要选择网格、面及锥线显示，则需首先判断ImageContrlParam项中重建是否已经开启，如果已开启则调用ShowFilter类中的对应接口皆可，并修改ImageControl对应项。
4. 如果选择重建选项，则根据用户所选参数创建FilterRestruction，重新组织链表，将FilterRestruction加入到Process类中，将FilterShow加入到FilterRestruction中的list中。
5. 如果选择拼接选项，则首先根据ImageControlParam项中的重建选项是否开启判断是否允许开启拼接选项，如果已经开启重建选项，则根据用户设置参数设置GPS，POSE对象，将指针赋给Sonar对应指针，并修改ImageControlParam对应参数；创建FilterStitching对象，将其加入到FilterRestruction对象的list中，将FilterShow对象加入到它的list下。
6. 点击截图或录像功能，则调用对应操作即可。

各取消操作则与上述操作相反即可。

5 在连接状态下，可对设备状态进行查询，通过SonarCtrl对象下发命令。

6 在在线状态下，点击开始按钮，判断连接状态，如果已经连接，则根据对应IP则创建OnlineSession对象，该对象用于接收UDP网络数据包；通过SonarCtrl对象下发开始命令通知下位机开始发送数据。OnlineSession对象接收到数据后，判断ImageFrameParam对象中是否开启存储标志，调用SonarDataStorage中的对应接口，然后将数据包指针传递给Process对象中的ProcessData接口， ProcessData在ProcessData接口中调用自身维护的list中的各Filter开始流水线处理。

7在连接状态下，可以对下位机代码进行更新，通过SonarCtrl对象下发代码文件，等待回收状态即可。

## 3.2离线状态工作流程

点击离线模式，则首先创建OfflineSession业务类对象，将Sonar类中的Session指针指向OfflineSession对象，根据默认参数设置用户界面。

离线状态下，可能进行以下操作。

1. 打开文件，选择要进行离线处理的文件名，设置SonarDataStorage类中的路径。
2. 图像输出区域的操作同在线模式，不在赘述
3. 点击开始，OfflineSession业务类对象开始读取文件中的每个数据包，并传给Process类。流水线开始工作。
4. 点击暂停，OfflineSession对象停止读取文件。

## 3.3自检状态工作流程（原始数据整合）

自检状态可以包含在在线状态下，点击自检状态，则通过已连接的SonarCtrl对象下发自检命令。

1. 点击开始，则下发开始命令，显示数据。
2. 点击停止，则下发停止命令。
3. 点击存储原始数据，则下发上传原始数据命令，将Process类中的list清空，设置SonarDataStorage对象路径，存储数据。

# 4 可维护性设计

## 4.1 调试日志方案

本项目采用日志工具log，该工具采用配置文件控制日志输出路径与日志输出级别，使用方便。通过一层适配将工具API封装为系统适用的各个日志级别接口，以单例形式存在，可供整系统全局访问。

## 4.2调试手段

在各模块中加入统计模块，例如在各数据处理模块中加入处理时间统计，在网络接收模块中加入数据包状态统计等。

# 5 问题与风险

N/A