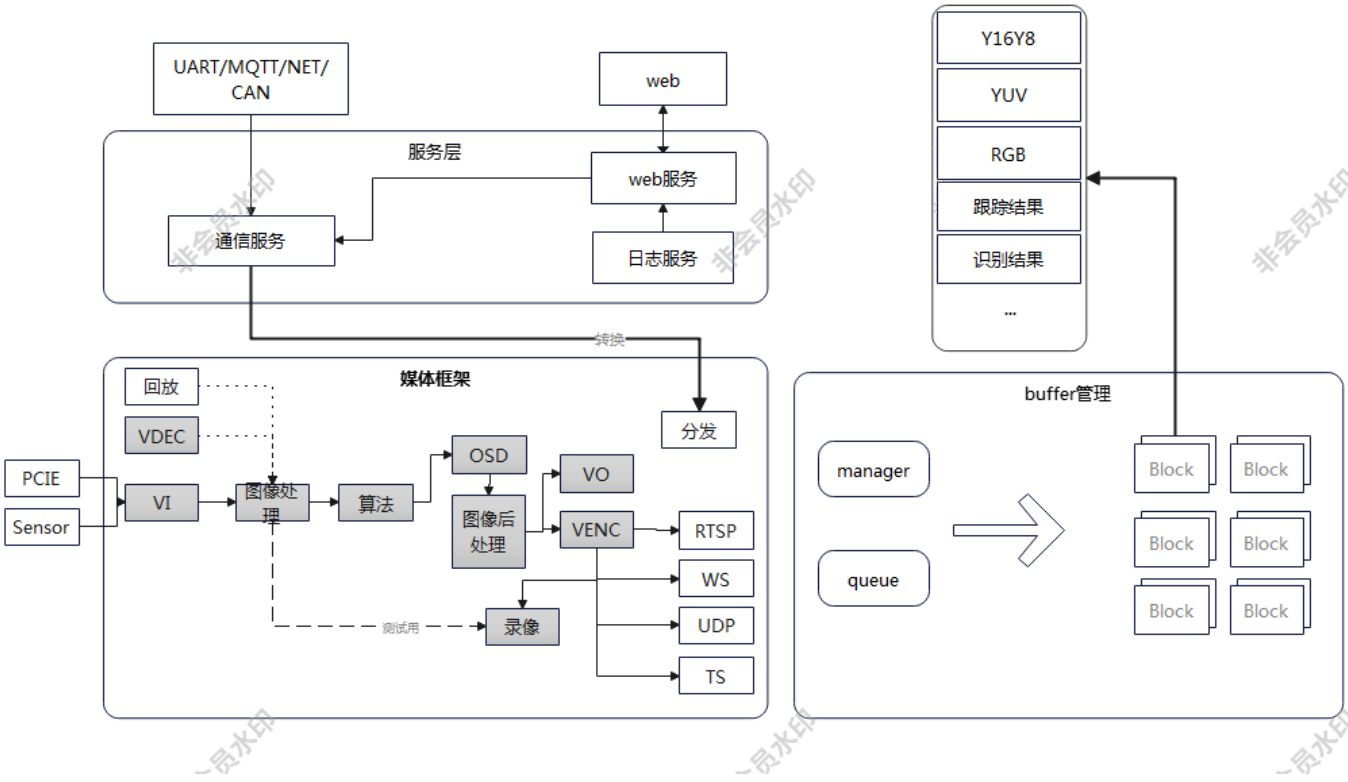
跟踪器框架

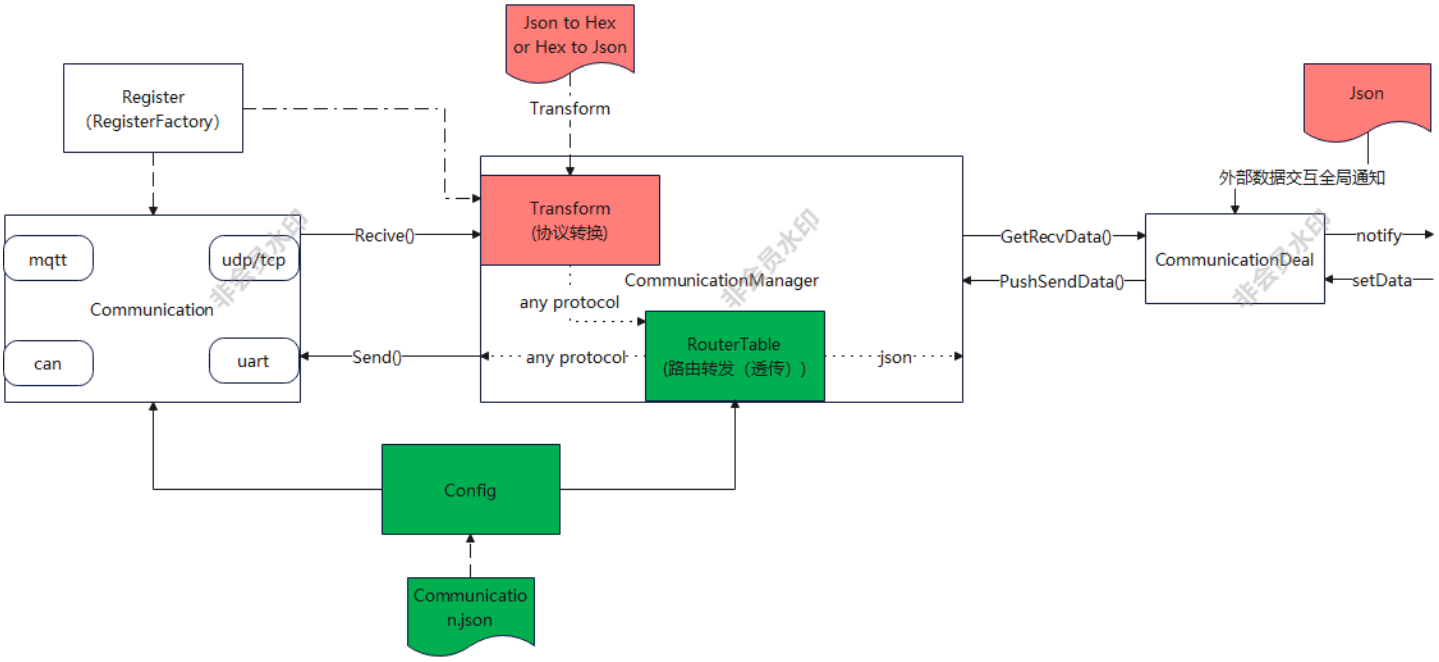
为实现跟踪器的通用框架，媒体层不应涉及任何与平台相关调用，只负责维护模块之间的绑定关系及数据传递，达到与平台解耦。平台按自身能力实现各模块的统一接口，媒体层通过调用统一的接口来与硬件进行交互。

1、模块划分

媒体框架可分为管理模块和业务模块，其中管理模块包括通信模块、配置管理模块、日志模块、buffer模块、web服务模块；业务模块指各图像处理模块，主要定义了采集输入、输出、图像处理、编码、解码、信息叠加、图像后处理、录像和算法9个模块。框架结构图如下图：



通信模块详细设计：



1.1 管理模块

1.1.1 配置管理模块

主要负责各级配置文件的解析。当前的配置文件包括框架模块配置、采集输入配置、编码输出配置、算法配置、OSD叠加配置、通信路由配置等。配置文件存储的是静态信息，修改需要重启软件。动态参数存在数据库中。

1.1.2 buffer模块

为减少图像内存拷贝，定义一个通用的图像数据模版，用来存储图像数据，包括原始采集数据、拆分之后的y16、yuv数据等，同时还会携带当前图像帧的附加信息（如算法结果等）。在程序初始化时进行内存分配、由当前平台决定内存的申请方式。数据块指针会在各级模块之间流转，直至最后使用完毕后回收。

1.1.3 日志模块

日志服务，负责存储和检索跟踪器日志信息。

1.1.4 通信模块

负责跟踪器对外协议，以及转换内部协议。内部协议定为json格式，需要将其他协议转换成内部json格式，并下发给业务模块。

1.1.5 web模块

实现一个标准跟踪器的控制界面。包括图像点播、跟踪控制、参数下发等功能。在调试阶段能代替显控终端覆盖跟踪器的基本功能。

1.2 业务模块

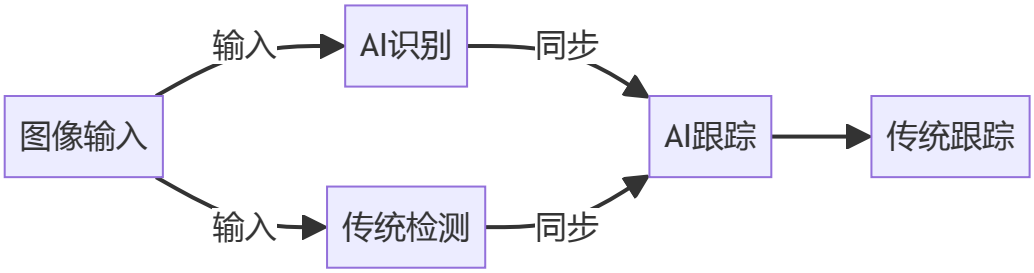
业务模块定义了主框架中的几个模块，各个模块由媒体管理统一创建，并根据模块的绑定关系进行绑定、绑定关系必须是流水线形式，不能形成环路。一帧图像从采集到输出的总时间取决于各个模块的处理耗时。

1.2.1 图像后处理

图像后处理是将叠加osd之后的图像进行电子变倍、叠加画中画等操作。可能会涉及到图像数据拷贝。图像拼接需要缓存数据帧，需要单独创建内存来存储拼接图像，不能使用buffer模块的数据。

1.2.2 算法模块

算法模块是指与图像相关的算法，可以由多个算法组成，每个算法独立运行。各个算法的运行结果保存在数据帧的附加信息中。经跟踪模块统一向下级流转。图像算法流程图如下：



1.2.3 同步

- **算法结果同步**
  多个算法同时运行，并将结果作为跟踪算法的输入，由于各个算法耗时的波动，要等最后一个算法结果出来之后再将数据流转给跟踪模块（这个时间取决于各个算法之中的最大耗时，所以各个算法的耗时必须满足当前帧率），这样就能保证跟踪算法的输入帧和这帧图像的其他算法结果是同步的。
- **载荷参数信息同步**
  载荷参数信息包括伺服的方位、俯仰、横滚，以及载荷的经纬高等信息。以跟踪器接收到上述信息的时间戳和图像采集的时间戳进行匹配，取相近的判断是同一帧。

2、配置说明

2.1 配置文件

- 1、所有json配置文件，均为小写。

1、模块配置

media.json 在框架中定义了8个模块，模块定义如下：

模块名	**功能**
vi	采集输入：主要是硬件输入：pcie、sensor等
vdec	解码输入: 可以是网络流、文件
vpss	图像处理：裁剪、缩放、旋转等(优先使用硬件进行处理)
vo	图像输出：主要只硬件输出，mipi、bt1120等
arith	算法模块：图像算法模块
venc	编码模块：视频编码输出
osd	图像叠加：叠跟踪框、识别框，叠文字等功能
post-vpss	图像后处理：画中画、电子变倍、拼接融合等功能
record	录像模块：视频存储

各个变量的定义如下：

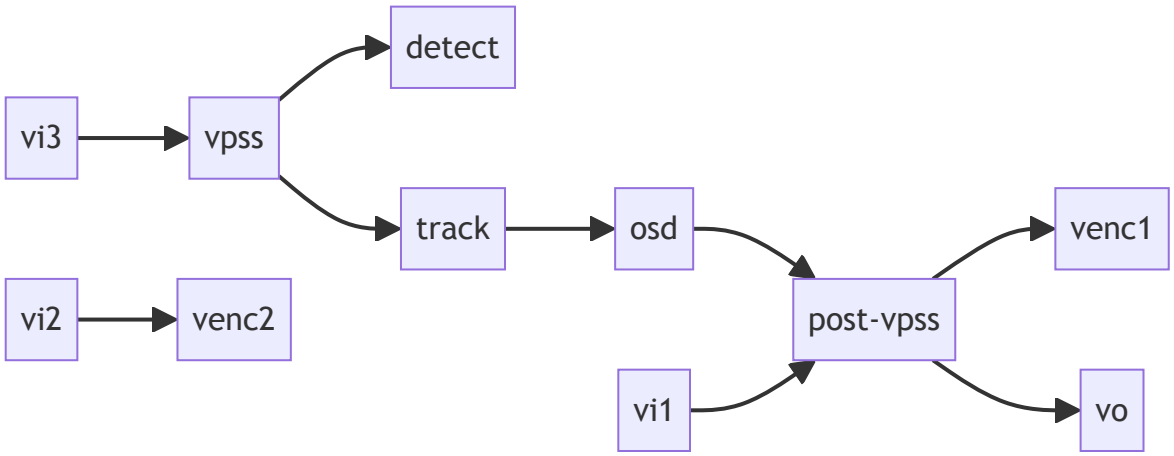
变量名	**变量说明**
enable	模块使能
name	模块名称
message	是否处理消息
bind	下级绑定

```
{
    "vi": [
        {
            "chn" : 0,
            "enable": true,
            "message" : true,
            "name" : "vi1",
            "bind" : [
                "post-vpss"
            ]
        },
        {
            "chn" : 1,
            "enable": true,
            "message" : true,
            "name" : "vi2",
            "bind" : [
                "venc2"
            ]
        }
    ],
    "vdec":{
        "enable": true,
        "message" : true,
        "name" : "vi3",
        "bind" : [
            "vpss"
        ]
    },
    "vpss" : {
        "enable": true,
        "name" : "vpss",
        "message" : true,
        "bind" : [
            "arith1"
        ]
    },
    "vo": {
        "enable": true,
        "name" : "vo",
        "message" : true
    },
    "osd": {
```

```
        "enable": true,
        "name" : "osd",
        "message" : true,
        "bind" : [
            "post-vpss"
        ]
    },
    "post-vpss": {
        "enable": true,
        "name" : "post-vpss",
        "message" : true,
        "bind" : [
            "venc1",
            "vo"
        ]
    },
    "venc": [
        {
            "enable": true,
            "name" : "venc1",
            "message" : true,
            "bind" : [
                "record"
            ]
        },
        {
            "enable": true,
            "name" : "venc2",
            "message" : true,
            "bind" : [
            ]
        },
        {
            "enable": false,
            "name" : "venc3",
            "message" : true,
            "bind" : [
            ]
        }
    ],
    "arith": [
        {
            "enable": true,
```

```
            "name" : "arith1",
            "message" : true,
            "bind" : [
                "osd"
            ]
        },
        {
            "enable": false,
            "name" : "arith2",
            "message" : true
        }
    ],
    "record": {
        "enable": true,
        "name" : "record1",
        "message" : true
    }
}
```

生成的绑定关系图



2、采集配置

模块配置中定义了输入模块，根据输入模块的名字，匹配对应的采集配置
capture.json定义了所有采集通道的信息 各个变量的定义如下：

变量名	变量说明
chn	显示通道：框架层次的通道定义，**不能相同**

变量名	变量说明
vi_chn	采集通道：适配层的通道定义，是硬件层的采集通道
type	数据格式：YUV Y16Y8 Y16 Y8等
width	采集宽度
height	采集高度
in_frame	采集帧率
out_frame	输出帧率(是否抽帧处理)
url	采集url(主要是解码模块使用)

```
{
    "vi2" :{
        "chn" : 0,
        "vi_chn": 1,
        "type" : "y16y8",
        "width" :1280,
        "height":1024,
        "in_frame": 50
    },
    "vi1" :{
        "chn" : 1,
        "vi_chn": 0,
        "type" : "yuv",
        "width" :1280,
        "height":1024,
        "in_frame": 50
    },
    "vi3" :{
        "chn" : 2,
        "type" : "yuv",
        "width" :1280,
        "height":1024,
        "out_frame":50,
        "url" : "./car_around.avi"
    }
}
```

根据上述采集配置，结合模块配置，知道当前输入模块是vi3，是解码输入。
chn: 表示应用层的视频通道，如视频源切换就与该通道对应。
vi_chn：表示硬件层采集通道，如光纤采集卡通道、sensor的vi通道等。

3、编码输出配置

编码配置流与模块配置中的venc进行匹配， 各个变量的定义如下：

变量名	**变量说明**
width	编码宽度
height	编码高度
fps	编码帧率
bitrate	比特率
ts	ts打包数据流
rtsp	rtsp数据流
websocket	ws数据流(web使用)
udp	私有组播流
record	录像流（编码后的流，如avi、MP4等）
enable	使能
dst	目的地址
port	端口

```
{
    "venc1" :{
        "width" : 1280,
        "height" :1024,
        "fps" : 25,
        "bitrate" : 2048000,
        "ts":{
            "enable" : true,
            "dst" : "udp://127.0.0.1:50003"
        },
        "rtsp" : {
            "enable" : true,
            "port": 8550,
            "dst" : "streanchn0"
        },
        "websocket":{
            "enable": true,
            "port" : 8081
        }
    },
    "venc2":{
        "width" :1280,
        "height": 1024,
        "fps" : 25,
        "gop" : 50,
        "bitrate" :2048000,
        "rtsp" : {
            "enable" : true,
            "port": 8550,
            "dst" : "streanchn1"
        },
        "websocket":{
            "enable": true,
            "port" : 8082
        },
        "udp":{
            "dst" : "224.1.0.12",
            "port": 50013
        },
        "record":{
            "split_mode":"time",
            "time":5,
            "size": 50
```

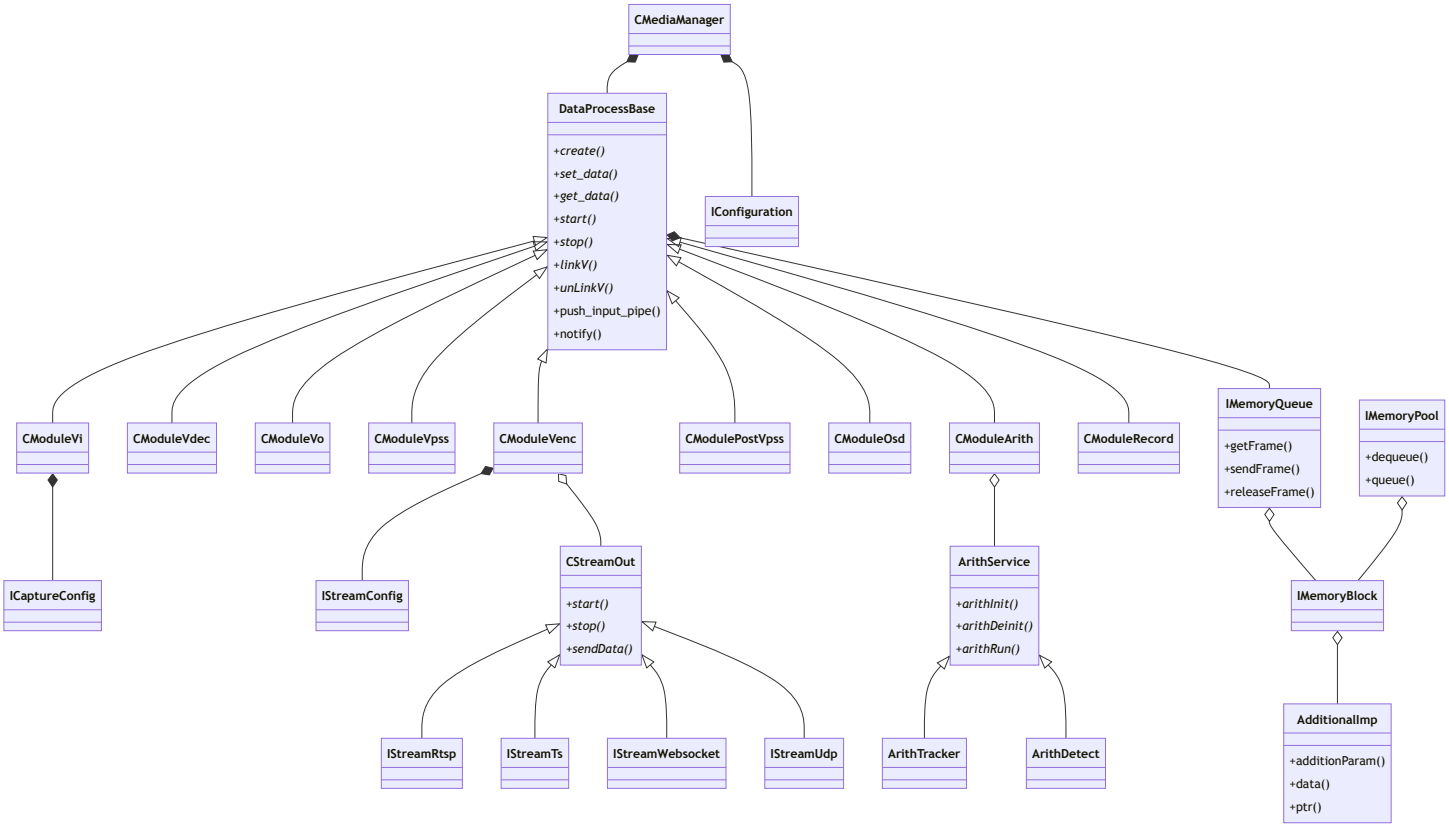
```
        }
    }
}
```

4、算法配置

3、开发流程

4、类图

框架类结构图：



- class MemoryManage :实现buffer块的填充，以及内存池初始化。如涉及到内存初始化的平台，需要指定图像宽高用于内存分配
- class CMediaManager :实现媒体模块之间的绑定，接收通信消息并分发处理。如需要新增模块，在这个类中进行匹配创建。
- class CModuleArith： 算法基类，用与各个算法模块的创建，根据指定的算法名字来匹配创建，如新增算法，在这个类中添加。并继承 **ArithService** 在Arith目录中实现该算法。
-

5、功能实现

功能划分