**环视拼接模块说明文档**

**版本历史**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 日期 | 修订信息 |
| v1.0 | 李成 | 2018.7.8 | 创建 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

1. 模块介绍：

**1.1 功能描述：**

环视拼接模块(AVM\_stitch)作为360环视最后一个模块，主要用于完成各个查找表的生成。将各个试图的查找表生成指定的格式，供嵌入式下位机依靠这些查找表生成环视图，单视图，3D图。该模块可通过配置参数满足不同下位机查找表的格式。

**1.2 运行此程序需要的素材**

1．AVM\_stitch环视拼接模块并不包含求解内外参的程序，所以要运行此程序必须利用其它模块比如内参求解模块，外参求解模块来求解出内外参。

2．需要提供4幅鱼眼原图。

3．需要用户获得真实的车长，车宽，轮距，轮轴。

4．需要用户自己配置图像的长宽，物理视野范围等等

**1.3 运行环境**

AVM\_stitch环视拼接模块为裕兰环视系统组成部分，其模块函数由AVM\_stitch工程调用。要想 运行次工程需要配置如下

AVM\_stitch\config目录下存放各个视角的参数。目前整个工程支持TXT文件配置和SMC文件形式配置。TXT形式的配置文件直接在TXT文件中按照规则填写配置参数。如果需要使用SMC进行配置参数则需要在SMC文件.xlsm里填写配置参数。

目前整个工程支持3个项目HDMI，CVBS，T2。其中只有T2支持3D视角。

AVM\_stitch\config\CVBS\_config存放CVBS输出配置参数。包括相机的内参，外参，车长车宽，视野范围都在此填写存放。

AVM\_stitch\config\HDMI\_config存放HDMI输出高清显示的配置参数。包括相机的内参，外参，车长车宽，视野范围都在此填写存放。

AVM\_stitch\config\T2\_config存放T2项目的配置参数，其中T2项目又分为2D和3D配置。包括相机的内参，外参，车长车宽，视野范围都在此填写存放。

（上位验证的原图和上位机结果目录说明）

1．AVM\_stitch\image是存放的鱼眼原图像和生成的上位机结果图像。以便在移植到开发板上的之前验证拼接结果。

AVM\_stitch\image\source\CVBS存放CVBS项目鱼眼原图。

AVM\_stitch\image\source\HDMI存放HDMI项目鱼眼原图。

AVM\_stitch\image\source\T2存放T2项目的鱼眼原图。

2．AVM\_stitch\image\result存放生成的结果图像。

AVM\_stitch\image\result\ CVBS存放CVBS项目的上位机拼接结果

AVM\_stitch\image\result\ HDMI存放HDMI项目的上位机拼接结果

AVM\_stitch\image\result\T2存放T2板子项目的上位机拼接结果，又分为2D和3D拼接结果。

3.AVM\_stitch\ lut\_table\ T2中存放生T2板子的格式

AVM\_stitch\ lut\_table\ Amba\ CVBS中存放生成CVBS项目的安霸表calib17.bin与calib21.bin

AVM\_stitch\ lut\_table\ Amba\ HDMI中存放生成HDMI项目的安霸表calib17.bin与calib21.bin

AVM\_stitch\doc存放程序生成文档

AVM\_stitch\port存放工程配置文件

AVM\_stitch\source存放整个工程的源码

此处也支持直接读取SMC写成的BIN 文件。

2. 模块设计

**2.1 数据结构设计**

AVM\_stitch环视拼接模块的数据结构定义在Avm\_Param\_S。Avm\_Param\_S结构体具体细节在如下的xlsx文档里：

AVM\_stitch\doc\AVM\_Parameter.xlsx

**2.2.1 接口函数**

**(1) AVM\_Init函数为配置函数入口。**

/\*

\* Function Name: AVM\_Init

\* Function Function: load config param for AVM module

\* Input:

\* p\_avm\_handle: the structure is hided outside of this module

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Shuo.Yan on 2017/11/23.

\*/

INTESIGHT\_API AVM\_Error\_Code\_E AVM\_Init(OUT Handle\* p\_avm\_handle)此函数完成的功能包括：

1) 为Avm\_Param\_S整个结构体在堆空间申请内存，此结构体在内存堆空间中将在整个程序运行期间有效。

2) 从配置文件中读取配置参数，包括读取各个相机的内参，外参视野范围等等

3) 为生成的查找表，动态线，静态线申请内存。

总之这个函数为了后续做了充分准备，并且返回大结构Avm\_Param\_S的指针，传入后续的处理函数使用

**(2) AVM\_Process函数为生成各种查找表的入口。**

/\*

\* Function Name: AVM\_Process

\* Function Function: the core process function of avm module

\* Input:

\* avm\_handle: the structure used in the module

\* Note:

\* the core interface function finish lut generation function

\* Revision:

\* Create by Shuo.Yan on 2017/11/23.

\*/

INTESIGHT\_API AVM\_Error\_Code\_E AVM\_Process(OUT IN Handle avm\_handle)

**(3)A**AVM\_Deinit**函数为各个内存释放堆空间的函数接口**

/\*

\* Function Name: AVM\_Deinit

\* Function Function: load config param for AVM module

\* Input:tf7

\* avm\_handle: the memory need to be freedes

\* Note:

\* the interface function should be called after the module finish its function

\* Revision:

\* Create by Shuo.Yan on 2017/11/23.

\*/

INTESIGHT\_API void AVM\_Deinit(IN Handle avm\_handle)

**2.2.2 核心函数**

**目前本工程支持三个项目的查找表：安霸HDMI，安霸CVBS，T2。为了区分这些项目在结构体**Avm\_Param\_S里加入项目支持标志位p\_avm\_param->project\_flag；三个项目的宏定义分别为

#define TEST 0//目前支持三个项目

#define CVBS 1//CVBS项目

#define HDMI 2//T2项目

**<3路，4路全景拼接部分>**

/\*

\* Function Name: handle\_3\_4\_view

\* Function Function: handle\_3\_4\_view

\* Input:

\* p\_avm\_param;

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/5/2.

\*/

AVM\_Error\_Code\_E handle\_3\_4\_view(IN P\_Avm\_Param\_S p\_avm\_param)；

此函数处理全景4路，前3路，后3路的查找表。

(1) generate\_4\_view\_lut函数主要完成生成4路环视图的查找表

/\*

\* Function Name: init\_lut

\* Function Function: this function is calculate source picture coordinate and zip data into LUT

\* Input:

\* image[4];

\* Note:

\*

\* Revision:

\* Create by Cheng.Li on 2018/1/12.

\*/

AVM\_Error\_Code\_E generate\_4\_view\_lut(IN P\_Avm\_Param\_S p\_avm\_param,

IN Result\_Size\_Param\_S result\_param, IN UInt32\_t\* p\_lut)

目前四路拼接支持两种形式：

一种是安霸形式的小车位置是写死的具体的小车位置自己求出来。自己写进配置文件中。

第二种是用户只需配置前后左右视野范围，小车的具体位置就可以自己计算出来。

代码的具体流程如下：

通过读取配置文档，获取标定好的内参，外参数据，用户配置的各种参数。

本程序的世界坐标系是建立在车辆的正中心，通过配置视野范围结合小车在环视图中的图像位置可以可以求出每个像素所代表的现实世界中实际距离。

计算环视图中每个像素点对应着鱼眼摄像图片的哪个像素的主要函数如下

此函数生成4路查找表的流程如下所示：



图2-1 4路环试图查找表流程图



图2-2 单个相机处理过程

calculate\_4\_view\_weight\_curve函数是计算融合区域的权重

/\*

\* Function Name: calculate\_front\_weight

\* Function Function: calculate front weight

\* Input:

\* i;

\* j;

\* world\_coordinate;

\* weight\_2\_dim;

\* Note:

\* 二次函数的曲线融合

\* Revision:

\* Create by Cheng.Li on 2018/1/12.

\*/

Float32\_t calculate\_4\_view\_weight\_curve(int camera\_flag, int i, int j,

Float32\_t \*world\_coordinate, Float32\_t \*weight\_2\_dim, IN P\_Avm\_Param\_S p\_avm\_param,IN Result\_Size\_Param\_S result\_param, int view\_flag)；

图像融合区域代码实现说明：

全景图中融合区域的权重是利用了二次曲线融合的方式。

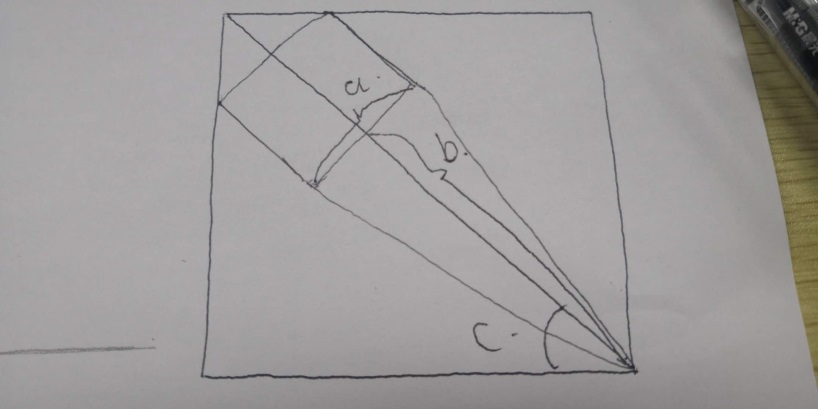


图2-3 融合区域的计算权重示意图

如图所示，配置文件中填写的fl\_fusion\_angle就是配置的c角度，parallel\_range与curve\_range的a与b的像素长度。在b范围内是曲线融合，这个曲线与a处的直线相切。a控制融合范围的大小，b控制曲线范围大小。

每一个像素的权重是由此像素位置到融合边界的距离与融合边界的总距离的比值求出来的。

world\_coordinate\_to\_pic\_coordinate函数是封装的世界坐标系转化成图像坐标系。

/\*

\* Function Name: world\_coordinate\_to\_pic\_coordinate

\* Function Function: world coordinate to pic\_coordinate

\* Input:

\* camera\_id; this is you shuold look for which picture

\* world\_coordinate;

\* image\_point; this is return the picture coordinate

\* Note:

\*

\* Revision:

\* Create by Cheng.Li on 2018/1/12.

\*/

//返回值image\_point[1]是x即列数，image\_point[0]是y即行数数，

AVM\_Error\_Code\_E world\_coordinate\_to\_pic\_coordinate(IN int camera\_id, IN Float32\_t \* world\_coordinate,

OUT Float32\_t \*image\_point, IN P\_Avm\_Param\_S p\_avm\_param)；

此函数主要包含两步，第一步通过外参将世界坐标系转换到相机坐标系中，第二步通过内参模型将相机坐标系转换到图像坐标系。

Cam\_Ray\_To\_Image\_Point函数将相机坐标系转换为图像坐标系

/\*

\* Function Name : Cam\_Ray\_To\_Image\_Point

\* Function Function : calculate image point's coordinate in camera coordinate system

\* Input :

\* point2D : image point's coordinate in image coordinate system(y to down(the 1st element, x to right(the 2nd element) ))

\* point3D : incline ray vector in camera coordinate system(x to up(the 1st element), y to right(the 2nd element), z to back(the 3rd element))

\* p\_camera\_model\_int : fisheye camera model

\* Note :

\* In camera coordinate system, to down is x positive direction, to right is y positive direction, to back is z positive z direction

\* Revision :

\* Rename by Shuo.Yan on 2017/11/23

\*///image\_point[0]

void Cam\_Ray\_To\_Image\_Point(OUT Float32\_t image\_point[2],

IN Float32\_t cam\_ray[3], IN P\_Camera\_Model\_Int\_S p\_camera\_model\_int)

zip\_data函数是将求解出的鱼眼图像，像素权重压缩存储到内存中

/\*

\* Function Name: zip\_data

\* Function Function: zip\_data

\* Input:

\* shift\_altogether; this is the position you should storage

\* image\_point; the picture coordinate

\* weight\_Float32\_t; the weight of pixel

\* Note:

\*

\* Revision:

\* Create by Cheng.Li on 2018/1/12.

\*/

AVM\_Error\_Code\_E zip\_data(IN int shift\_altogether, IN Float32\_t \*image\_point, IN Float32\_t weight\_Float32\_t,

IN UInt32\_t\*p\_lut, IN UInt32\_t camera\_flag, IN UInt32\_t left\_right\_flag)

**（2）**generate\_3\_front\_view\_lut与generate\_3\_rear\_view\_lut函数主要完成生成前3路和后3路环视图的查找表

/\*

\* Function Name: generate\_3\_front\_view\_lut

\* Function Function: generate 3 front\_view lut

\* Input:

\* image[4];

\* Note:

\*

\* Revision:

\* Create by Cheng.Li on 2018/1/17.

\*/

AVM\_Error\_Code\_E generate\_3\_front\_view\_lut(IN P\_Avm\_Param\_S p\_avm\_param)

/\*

\* Function Name: generate\_3\_rear\_view\_lut

\* Function Function: generate\_3\_rear\_view\_lut

\* Input:

\* image[4];

p\_avm\_param;

\* Note:

\*

\* Revision:

\* Create by Cheng.Li on 2018/1/17.

\*/

AVM\_Error\_Code\_E generate\_3\_rear\_view\_lut(IN P\_Avm\_Param\_S p\_avm\_param)

3路图的拼接程序流程图基本与4 路图基本一样，三路拼接图车体漏出来的长度是用户在配置文件里修改的。前三路不需要生成后镜头的查找表，后三路不需要生成前镜头的查找表，其他部分完全一样。

**<前后单视图生成部分>**

handle\_front\_rear\_single\_view函数主要是处理前后单视图，目前支持的类型有

TYPE\_CROP = 0,//原图裁剪

TYPE\_PLANE\_EXPAND,//平面展开

TYPE\_BOX\_EXPAND,//盒子展开

/\*

\* Function Name: handle\_front\_rear\_single\_view

\* Function Function: handle\_front\_rear\_single\_view

\* Input:

\* p\_avm\_param;

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/5/2.

\*/

AVM\_Error\_Code\_E handle\_front\_rear\_single\_view(IN P\_Avm\_Param\_S p\_avm\_param)

crop\_front\_rear\_single\_view函数是直接裁剪鱼眼原图显示到屏幕函数，

/\*

\* Function Name: crop\_front\_rear\_single\_view

\* Function Function: crop\_front\_rear\_single\_view

\* Input:

\* p\_avm\_param;

\* cam\_flag; this is picture's flag is front or back

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/1/30.

\*/

AVM\_Error\_Code\_E crop\_front\_rear\_single\_view(IN P\_Avm\_Param\_S p\_avm\_param, IN int cam\_flag)

plane\_front\_rear\_single\_view采用平面展开方式生成前单视图

/\*

\* Function Name: plane\_front\_rear\_single\_view

\* Function Function: plane\_front\_rear\_single\_view

\* Input:

\* p\_avm\_param;

\* cam\_flag; this is picture's flag is front or back

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/1/30.

\*/

AVM\_Error\_Code\_E plane\_front\_rear\_single\_view(IN P\_Avm\_Param\_S p\_avm\_param, IN int cam\_flag)

box\_front\_rear\_single\_view是采用盒子展开的方式生成前单视图，BOX展开方式可以在代码中修改虚拟相机的俯仰角的角度，更改左右事业范围

/\*

\* Function Name: box\_front\_rear\_single\_view

\* Function Function: box\_front\_rear\_single\_view

\* Input:

\* p\_avm\_param;

\* cam\_flag; this is picture's flag is front or back

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/1/30.

\*/

AVM\_Error\_Code\_E box\_front\_rear\_single\_view(IN P\_Avm\_Param\_S p\_avm\_param,IN int cam\_flag)

**<左右单视图生成部分>**

handle\_left\_right\_single\_view函数主要是处理左右单视图，目前支持的类型有

TYPE\_CROP = 0,//原图裁剪

TYPE\_PLANE\_EXPAND,//俯视图平面展开

TYPE\_BOX\_EXPAND:,//伪3D展开

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* handle\_left\_right\_single\_view\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Function Name: handle\_left\_right\_single\_view

\* Function Function: handle\_left\_right\_single\_view

\* p\_avm\_param;

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/5/2.

\*/

AVM\_Error\_Code\_E handle\_left\_right\_single\_view(IN P\_Avm\_Param\_S p\_avm\_param)

left\_right\_crop\_expend函数是直接裁剪鱼眼原图支持旋转角度的裁剪原图

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*left\_right\_crop\_expend\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Function Name: left\_right\_crop\_expend

\* Function Function: left\_right\_crop\_expend

\* Input:

\* p\_avm\_param;

\* cam\_flag; this is picture's flag is front or back

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/1/30.

\*/

AVM\_Error\_Code\_E left\_right\_crop\_expend(IN P\_Avm\_Param\_S p\_avm\_param, IN int cam\_flag)

left\_right\_overlook\_expend函数是左右展开，直接进行俯视图的展开

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*left\_right\_overlook\_expend\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Function Name: left\_right\_overlook\_expend

\* Function Function: left\_right\_overlook\_expend

\* Input:

\* p\_avm\_param;

\* cam\_flag; this is picture's flag is front or back

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/1/30.

\*/

AVM\_Error\_Code\_E left\_right\_overlook\_expend(IN P\_Avm\_Param\_S p\_avm\_param,IN int cam\_flag)

left\_right\_combine\_overlook\_expend函数为了实现左右的伪3D效果图，伪3D的实现是赋予世界坐标系一个估算的高度，不再是平面展开的高度为0。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\* left\_right\_combine\_overlook\_expend\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Function Name: left\_right\_combine\_overlook\_expend

\* Function Function: left\_right\_combine\_overlook\_expend

\* Input:

\* p\_avm\_param;

\* cam\_flag; this is picture's flag is front or back

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/1/30.

\*/

AVM\_Error\_Code\_E left\_right\_combine\_overlook\_expend(IN P\_Avm\_Param\_S p\_avm\_param,IN int cam\_flag)

<距离线模块>

handle\_static\_dynamic\_line是用来生成动态，静态距离线的。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*handle\_static\_dynamic\_line\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*

\* Function Name: handle\_static\_dynamic\_line

\* Function Function: handle\_static\_dynamic\_line

\* p\_avm\_param;

\* Note:

\* the interface function should be called at the beginning of this module

\* Revision:

\* Create by Cheng.Li on 2018/5/2.

\*/

AVM\_Error\_Code\_E handle\_static\_dynamic\_line(IN P\_Avm\_Param\_S p\_avm\_param)

original\_crop\_draw\_static\_plane\_front\_rear\_single\_view\_line函数用来生成静态距离线。生成的静态距离线距离车的距离可以用户自己在配置文件里配置。

/\*

\* Function Name: draw\_static\_plane\_front\_rear\_single\_view\_line

\* Function Function: draw\_static\_plane\_front\_rear\_single\_view\_line

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Input:

\* shift; p\_avm\_param

\* image; cam\_flag

p\_avm\_param;

\* Note:

\*

\* Revision:

\* Create by Cheng.Li on 2018/1/17.

\*/

AVM\_Error\_Code\_E original\_crop\_draw\_static\_plane\_front\_rear\_single\_view\_line(IN P\_Avm\_Param\_S p\_avm\_param, IN int cam\_flag)

draw\_dynamic\_plane\_front\_rear\_single\_view\_line是用来生成动态距离线的，动态距离线就是方向盘每转一个角度都描画出一个汽车 的运动轨迹。

\* Function Name: draw\_dynamic\_plane\_front\_rear\_single\_view\_line

\* Function Function: draw\_dynamic\_plane\_front\_rear\_single\_view\_line

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Input:

\* shift; p\_avm\_param

\* image; cam\_flag

p\_avm\_param;

\* Note:

\*

\* Revision:

\* Create by Cheng.Li on 2018/1/17.

\*/

AVM\_Error\_Code\_E draw\_dynamic\_plane\_front\_rear\_single\_view\_line(IN P\_Avm\_Param\_S p\_avm\_param,IN int cam\_flag)