**俯视、透视生成接口**

**统一参数化接口：**

**在parameter\_\*\*\*.xml文件中设置车体及摄像头相关参数：**

**<?xml version="1.0"?>**

**<opencv\_storage>**

**//运行目录**

**<run\_directory> /home/chengguoqiang/wangkun\_git/tools/transform/</run\_directory>**

**<!-- 前相机内参和畸变 -->**

**<front\_K type\_id="opencv-matrix">**

**<rows>3</rows>**

**<cols>3</cols>**

**<dt>d</dt>**

**<data>**

**3.1078977195065829e+02 0. 6.3344839920094694e+02 0.**

**3.0968085232170705e+02 3.6349089096542718e+02 0. 0. 1.</data></front\_K>**

**<front\_D type\_id="opencv-matrix">**

**<rows>4</rows>**

**<cols>1</cols>**

**<dt>d</dt>**

**<data>**

**9.5340911744766776e-02 -9.5465014988361640e-03**

**8.3234674085381870e-03 -2.9988853067807321e-03 </data></front\_D>**

**<!-- 后相机内参和畸变 -->**

**<back\_K type\_id="opencv-matrix">**

**<rows>3</rows>**

**<cols>3</cols>**

**<dt>d</dt>**

**<data>**

**3.1082474080285789e+02 0. 6.3197430916786004e+02 0.**

**3.1033757066429246e+02 3.3587995224612575e+02 0. 0. 1.</data></back\_K>**

**<back\_D type\_id="opencv-matrix">**

**<rows>4</rows>**

**<cols>1</cols>**

**<dt>d</dt>**

**<data>**

**9.7825972530812252e-02 -9.2108999233784360e-03**

**7.1768654991110781e-03 -2.7100551438061496e-03</data></back\_D>**

**<!-- 左相机内参和畸变 -->**

**<left\_K type\_id="opencv-matrix">**

**<rows>3</rows>**

**<cols>3</cols>**

**<dt>d</dt>**

**<data>**

**3.1027573989574233e+02 0. 6.0634522155785749e+02 0.**

**3.0877244973194200e+02 3.5957934367374702e+02 0. 0. 1.</data></left\_K>**

**<left\_D type\_id="opencv-matrix">**

**<rows>4</rows>**

**<cols>1</cols>**

**<dt>d</dt>**

**<data>**

**8.4636588026297796e-02 1.7939636666712768e-03 2.3714315254197041e-03**

**-1.7544719354972824e-03</data></left\_D>**

**<!-- 右相机内参和畸变 -->**

**<right\_K type\_id="opencv-matrix">**

**<rows>3</rows>**

**<cols>3</cols>**

**<dt>d</dt>**

**<data>**

**313.789989100 0.000000000 630.193524682**

**0.000000000 311.561597329 336.182866986**

**0.000000000 0.000000000 1.000000000**

**</data></right\_K>**

**<right\_D type\_id="opencv-matrix">**

**<rows>4</rows>**

**<cols>1</cols>**

**<dt>d</dt>**

**<data>**

**0.083768200**

**0.000265363**

**0.002011781**

**-0.001565375</data></right\_D>**

**<!-- 前相机外参 -->**

**<front\_trans type\_id="opencv-matrix">**

**<rows>3</rows>**

**<cols>4</cols>**

**<dt>d</dt>**

**<data>**

**0.999250324 -0.024608571 0.029886602 79.388037189**

**0.012160461 -0.533395535 -0.845778533 2778.816196306**

**0.036754781 0.845507908 -0.532696409 -3201.870411153</data></front\_trans>**

**<!-- 后相机外参 -->**

**<back\_trans type\_id="opencv-matrix">**

**<rows>3</rows>**

**<cols>4</cols>**

**<dt>d</dt>**

**<data>**

**-0.998486875 -0.051605845 0.018994650 -65.497841335**

**-0.038555464 0.410684508 -0.910961970 1248.427071079**

**0.039210153 -0.910315919 -0.412052778 -530.614964118</data></back\_trans>**

**<!-- 左相机外参 -->**

**<left\_trans type\_id="opencv-matrix">**

**<rows>3</rows>**

**<cols>4</cols>**

**<dt>d</dt>**

**<data>**

**0.004755543 0.997934031 -0.064070697 -2364.883696406**

**0.616469004 -0.053373808 -0.785568077 1600.042880735**

**-0.787364815 -0.035761796 -0.615449220 -33.261139383</data></left\_trans>**

**<!-- 右相机外参 -->**

**<right\_trans type\_id="opencv-matrix">**

**<rows>3</rows>**

**<cols>4</cols>**

**<dt>d</dt>**

**<data>**

**0.027220919 -0.998529828 0.046874341 2347.335777751**

**-0.612429908 -0.053719560 -0.788697545 1621.577005458**

**0.790056093 -0.007238177 -0.612991826 -102.140392773**

**</data></right\_trans>**

***<!-- 汽车标志颜色 -->***

**<B>121</B>**

**<G>121</G>**

**<R>121</R>**

**<!-- 前后左右视野 -->**

**<front\_word\_view>8000</front\_word\_view>**

**<rear\_world\_view>8000</rear\_world\_view>**

**<left\_world\_view>8000</left\_world\_view>**

**<right\_world\_view>8000</right\_world\_view>**

**<!-- 汽车宽度 -->**

**<car\_world\_width>1878</car\_world\_width>**

**<!-- 汽车长度 -->**

**<car\_world\_height>5256</car\_world\_height>**

**<!-- 后轴中心到车前距离 -->**

**<car\_axle\_coord>4198</car\_axle\_coord>**

**<!-- 俯视展开左侧视图的高度-->**

**<left\_pixel\_height>720</left\_pixel\_height>**

**<!-- 透视标注用物理范围框,具体数值以相机为坐标原点,相机离坐标原点，车辆后轴中心在y轴上的距离 -->**

**<right\_cam\_height\_y>1928</right\_cam\_height\_y>**

**<left\_cam\_height\_y>1928</left\_cam\_height\_y>**

**<!-- front 以相机位置位坐标原点，光心往外为y正方向,相机往右前方向为x正方向 -->**

**<front\_projective\_lefttop\_x>-1970</front\_projective\_lefttop\_x>**

**<front\_projective\_lefttop\_y>965</front\_projective\_lefttop\_y>**

**<front\_projective\_leftbut\_x>-1970</front\_projective\_leftbut\_x>**

**<front\_projective\_leftbut\_y>365</front\_projective\_leftbut\_y>**

**<front\_projective\_righttop\_x>1970</front\_projective\_righttop\_x>**

**<front\_projective\_righttop\_y>965</front\_projective\_righttop\_y>**

**<front\_projective\_rightbut\_x>1970</front\_projective\_rightbut\_x>**

**<front\_projective\_rightbut\_y>365</front\_projective\_rightbut\_y>**

**<!-- rear 以相机位置位坐标原点，光心往外为y正方向,相机往右前方向为x正方向 -->**

**<rear\_projective\_lefttop\_x>-1970</rear\_projective\_lefttop\_x>**

**<rear\_projective\_lefttop\_y>965</rear\_projective\_lefttop\_y>**

**<rear\_projective\_leftbut\_x>-1970</rear\_projective\_leftbut\_x>**

**<rear\_projective\_leftbut\_y>365</rear\_projective\_leftbut\_y>**

**<rear\_projective\_righttop\_x>1970</rear\_projective\_righttop\_x>**

**<rear\_projective\_righttop\_y>965</rear\_projective\_righttop\_y>**

**<rear\_projective\_rightbut\_x>1970</rear\_projective\_rightbut\_x>**

**<rear\_projective\_rightbut\_y>365</rear\_projective\_rightbut\_y>**

**<!-- left 以相机位置位坐标原点，光心往外为x正方向,相机往车前方向为y正方向 -->**

**<left\_projective\_lefttop\_x>1050</left\_projective\_lefttop\_x>**

**<left\_projective\_lefttop\_y>-1050</left\_projective\_lefttop\_y>**

**<left\_projective\_leftbut\_x>450</left\_projective\_leftbut\_x>**

**<left\_projective\_leftbut\_y>-1050</left\_projective\_leftbut\_y>**

**<left\_projective\_righttop\_x>1050</left\_projective\_righttop\_x>**

**<left\_projective\_righttop\_y>1760</left\_projective\_righttop\_y>**

**<left\_projective\_rightbut\_x>450</left\_projective\_rightbut\_x>**

**<left\_projective\_rightbut\_y>1760</left\_projective\_rightbut\_y>**

**<!-- right 以相机位置坐标原点，光心往外为x正方向,相机往车前方向为y正方向 -->**

**<right\_projective\_lefttop\_x>1050</right\_projective\_lefttop\_x>**

**<right\_projective\_lefttop\_y>1070</right\_projective\_lefttop\_y>**

**<right\_projective\_leftbut\_x>450</right\_projective\_leftbut\_x>**

**<right\_projective\_leftbut\_y>1070</right\_projective\_leftbut\_y>**

**<right\_projective\_righttop\_x>1050</right\_projective\_righttop\_x>**

**<right\_projective\_righttop\_y>-1050</right\_projective\_righttop\_y>**

**<right\_projective\_rightbut\_x>450</right\_projective\_rightbut\_x>**

**<right\_projective\_rightbut\_y>-1050</right\_projective\_rightbut\_y>**

**</opencv\_storage>**

**变量：**

可执行文件运行路径：

**<run\_directory> /home/chengguoqiang/wangkun\_git/tools/transform/</run\_directory>**

B、G、R 中间区域汽车标志的三通道颜色，可根据实际需要做修改

front\_world\_view、rear\_world\_view、left\_world\_view、right\_world\_view 前/后/左/右物理视野，可根据实际需要做修改

car\_world\_width、 car\_world\_height、 car\_axle\_coord 汽车宽度、长度、后轴到车前距离，根据实际的车型而定

left\_pixel\_height 左侧展开像素的高度，因为需保证在俯视展开时，实际的长宽比与展开后的长宽比保持一致，只要确定了左侧像素高度，而车型参数、前、后、左、右物理视野确定，则前、后、左、右俯视展开图像的像素宽度和高度就会确定

**展开类型枚举：**俯视展开、透视展开、多视角透视展开

**typedef enum image\_view\_enum**

{

image\_view\_min = 0,

image\_view\_birdview\_e, //俯视图

image\_view\_projective\_e, //透视图

image\_view\_stitch\_e, //俯视拼接融合图

image\_view\_multiview\_e, //多视角透视展开图 image\_view\_max,

}IMAGE\_VIEW\_E;

相机类型枚举：front\rear\left\right,四个相机类型

**typedef enum** view\_enum

{

view\_min = 0,

view\_front\_e,

view\_rear\_e,

view\_left\_e,

view\_right\_e,

view\_max,

}VIEW\_E;

*//配合image\_view\_stitch\_e使用，其它情况无效*

//*融合区域枚举：*

**typedef enum** fusion\_enum

{

***fusion\_min*** = 0,

***fusion\_front\_e***, //前区域需要融合，后区域不需要融合

***fusion\_rear\_e***, //后区域需要融合，前区域不需要融合

***fusion\_left\_e***,//左区域需要融合，右区域不需要融合

***fusion\_right\_e***,//右区域需要融合，左区域不需要融合

***fusion\_left\_right\_e***, //左、右同时需要融合

***fusion\_front\_rear\_e***,//前、后同时需要融合

***fusion\_max***,

}FUSION\_E;

**读取xml格式的配置参数:**

int get\_parameter(std::string file\_name);

函数名：get\_parameter

功能：读取配置参数，初始化需要的变量

函数参数：std::string file\_name xml格式的配置文件名

返回：是否能成功读取配置文件

**透视去畸变展开函数接口：**

int undistort\_plane\_image(IN Mat raw\_image, INOUT Mat undistort\_image, IN float fov[2],IN VIEW\_E view\_index);

函数名： undistort\_plane\_image

功能：去畸变透视平面展开

函数参数：

IN Mat raw\_image：输入参数，原始鱼眼图

INOUT Mat undistort\_image：输入输出参数，最终目标去畸变 透视展开图，需初始化mat大小，即最终目标图的大小

IN float fov[2]：输入参数，fov大小，角度值，[0]水平方向 fov，[1]为垂直方向fov

IN VIEW\_E view\_index:输入参数，枚举类型， front\rear\left\right，对哪个相机进行展开

返回值：是否成功执行

函数接口运行时间统计：

测试环境：

Intel® Core™ i5-3470 CPU @ 3.20GHz × 4

7.7 GiB

64位

ubuntu 16.04

|  |  |  |
| --- | --- | --- |
| 函数名/commit | runtime(ms) | date |
| undistort\_plane\_image | 88 | 9.10 |
| 改为矩阵形式 | 110~130 | 9.27 |

**void** undistort\_generate\_matirx(**IN float** fov[2], **IN int** dst\_image\_size[2], **INOUT float** matrix[3][3])

*\* name: undistort\_generate\_matirx*

*\* function:产生从像素到相机坐标变换矩阵*

*\* parametes:*

*\* IN: float fov[2]*

*\* IN: int dst\_image\_size[2]*

*\* IN: int src\_image\_size[2]*

*\* OUT:*

*\* INOUT：float matirx[3][3]*

*\**

*\*return;*

*\* \*/*

|  |  |  |
| --- | --- | --- |
| 函数名/commit | runtime(ms) | date |
| undistort\_generate\_matirx | 0.12 | 9.27 |
|  |  |  |

**多视角去畸变展开函数接口：**

int multiview\_undistort\_plane\_image(IN Mat raw\_image, OUT Mat undistort\_image, IN float fov[2],IN VIEW\_E view\_index);

函数名： multiview\_undistort\_plane\_image

功能：多视角去畸变平面透视展开，中间fov正视展开，上、下、左、右边缘部分斜视展开，达到无遗漏原始数据展开的功能

函数参数：

IN Mat raw\_image：输入参数，原始鱼眼图

INOUT Mat undistort\_image：输入输出参数，最终目标去畸变 透视展开图，需初始化mat大小，即最终目标图的大小

IN VIEW\_E view\_index:输入参数，枚举类型， front\rear\left\right，对哪个相机进行展开

返回值：是否成功执行

**俯视图像生成模块函数接口**：

Mat birdview\_image\_generate(IN Mat raw\_image, IN VIEW\_E view\_index, IN bool stitch\_select\_b);

函数名： birdview\_image\_generate

功能：生成俯视图像

函数参数：

IN Mat raw\_image：输入参数，原始鱼眼图

IN VIEW\_E view\_index:输入参数，枚举类型， front\rear\left\right，对哪个相机进行展开

IN bool stitch\_select\_b:输入参数，布尔变量

True: 生成俯视融合图

False:生成俯视图

返回值：返回生成的俯视效果图

**俯视非融合拼接图函数接口**：

Mat birdview\_image\_stitch(IN Mat front\_image, IN Mat rear\_image, IN Mat left\_image, IN Mat right\_image);

函数名： birdview\_image\_stitch

功能：前、后、左、右俯视图，生成一张俯视拼接非融合图片

函数参数：

IN Mat front\_image：输入参数，前相机原始鱼眼图

IN Mat rear\_image：输入参数，后相机原始鱼眼图

IN Mat left\_image：输入参数，左相机原始鱼眼图

IN Mat right\_image：输入参数，右相机原始鱼眼图

返回值：返回生成好的拼接图片

函数接口运行时间统计：

测试环境：

Intel® Core™ i5-3470 CPU @ 3.20GHz × 4

7.7 GiB

64位

ubuntu 16.04

|  |  |
| --- | --- |
| 函数名 | 平均运行时间 (ms) |
| birdview\_image\_stitch | 48 |

**LUT表生成图片函数接口**：

int lutTable\_generate\_image(IN Mat raw\_image, INOUT Mat dst\_image, IN IMAGE\_VIEW\_E image\_view\_e, IN VIEW\_E view\_index, **IN** FUSION\_E stitch\_fusion);

函数名： lutTable\_generate\_image

功能：根据已经生成好的lut表，生成对应的效果图

函数参数：

IN Mat raw\_image：输入参数，原始鱼眼图

INOUT Mat dst\_image：输入输出参数，目标图，需初始化对应 的目标图大小

IN IMAGE\_VIEW\_E image\_view\_e：输入参数，需生成的目标图 的类型，俯视、透视、多视角透视

IN VIEW\_E view\_index：输入参数，相机对应的枚举，前、后、 左、右相机

IN FUSION\_E stitch\_fusion:输入参数，选取需要融合的区域，配合 image\_view\_e = image\_view\_stitch\_e

使用，其它情况无效

返回值：是否成功执行

函数接口运行时间统计：

测试环境：

Intel® Core™ i5-3470 CPU @ 3.20GHz × 4

7.7 GiB

64位

ubuntu 16.04

测试图像: (640,480)

|  |  |  |
| --- | --- | --- |
| 函数名 | **展开类型** | 平均运行时间 (ms) |
| lutTable\_generate\_image | image\_view\_stitch\_e | 10 ~ 11 |
|  | image\_view\_birdview\_e | 10 ~ 11 |
|  | image\_view\_projective\_e | 18 |

**使用lut table，生成俯视融合拼接图**

Mat lutTable\_generate\_stitchFusion(IN Mat front\_raw\_image, IN Mat rear\_raw\_image, IN Mat left\_raw\_image, IN Mat right\_raw\_image, IN bool front\_use, IN bool rear\_use, IN bool left\_use, IN bool right)

函数名： lutTable\_generate\_stitchFusion

功能：根据已经生成好的lut表，生成对应环视拼接图

函数参数：

IN Mat front\_raw\_image 输入参数，前相机原始鱼眼图

IN Mat rear\_raw\_image 输入参数，后相机原始鱼眼图

IN Mat left\_raw\_image 输入参数，左相机原始鱼眼图

IN Mat right\_raw\_image 输入参数，右相机原始鱼眼图

IN bool front\_use 输入参数，是否使用前相机数据融合

IN bool rear\_use 输入参数，是否使用后相机数据融合

IN bool left\_use 输入参数，是否使用左相机数据融合

IN bool right\_use 输入参数，是否使用右相机数据融合

返回值： 返回拼接的图像

函数接口运行时间统计：

测试环境：

Intel® Core™ i5-3470 CPU @ 3.20GHz × 4

7.7 GiB

64位

ubuntu 16.04

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 函数名 | 平均运行时间 (ms) | 日期（备注） |
| lutTable\_generate\_stitchFusion | 55 ~ 60 | 9.10 |
| 200+ | 9.27(增加白平衡) |
| generate\_gain（每帧*1*次） | 10 | 9.27 |
| *image\_addGain* | 48 | 9.27*(1280\*720)* |
| 14 | 9.27*(279\*621\*4*路*)* |
| *pix2cam\_model*（） | 16 | 10.23 （运行10000次，多项式拟合求系数，误差小于一个像素） |
|  |  |  |

**构造斜车位相关说明：「2019/01/10」**

1. lut表生成部分

stretch\_coef = 0;

yCoef = 1;

对应无拉伸效果的正常车位lut表生成函数。

原理：在世界坐标上转换(world)，x坐标保持不变，y坐标加上x坐标的分量形成一定程度的拉伸。拉伸程度由x前的系数stretch\_coef控制， stretch\_coef越大，拉伸越严重，一般取0.4/0.5，当stretch\_coef取0时不对车位进行拉伸。为了保持拉伸后车位的长宽比，需要对拉伸后的车位进行拉宽恢复，由yCoef控制，一般取0.9/0.8，当yCoef取0时不对车位进行拉宽。

公式：x = x0;

y = y0 + stretch\_coef \* x;

y = yCoef \* y;

具体实现：在lut表生成函数birdview\_image\_generate() 中实现， world\_position[1] += stretch\_coef \* world\_position[0];

world\_position[1] = yCoef \* world\_position[1];

函数说明：

Mat birdview\_image\_generate(IN Mat raw\_image, IN VIEW\_E view\_index, IN bool stitch\_select\_b, IN float stretch\_coef, IN float yCoef);

*\* function:生成俯视图，并生成lut table*

*\* parametes:*

*\* IN: Mat raw\_image 原始鱼眼图*

*\* IN: VIEW\_E view\_index视图相机枚举,前、后、左、右相机*

*\* IN bool stitch\_select\_b 是否需要生成融合系数*

*\* IN float stretch\_coef 车位倾斜程度拉伸系数*

*\* IN float yCoef 车位纵向拉宽系数*

*\*return: Mat birdview\_image 俯视图像*

2.图像拼接部分

实现原理：对原矩形车位的倾斜角度进行量化，事先生成不同角度范围对应的lut表，存在不同的文件夹，并把路径存在.xml文件中；

<**run\_directory\_0\_20**> /home/liuli/test/calibration\_tool/transform/LUT/lut\_GL8/0.4\_0.9/</**run\_directory\_0\_20**>

<**stretch\_coef\_0\_20**> 0.4 </**stretch\_coef\_0\_20**>

<**yCoef\_0\_20**> 0.9 </**yCoef\_0\_20**>

<**run\_directory\_20\_40**> /home/liuli/test/calibration\_tool/transform/LUT/lut\_GL8/0.5\_1/</**run\_directory\_20\_40**>

<**stretch\_coef\_20\_40**> 0.5 </**stretch\_coef\_20\_40**>

<**yCoef\_20\_40**> 1 </**yCoef\_20\_40**>

<**run\_directory\_40\_60**> /home/liuli/test/calibration\_tool/transform/LUT/lut\_GL8/0.4\_0.8/</**run\_directory\_40\_60**>

<**stretch\_coef\_40\_60**> 0.4 </**stretch\_coef\_40\_60**>

<**yCoef\_40\_60**> 0.8 </**yCoef\_40\_60**>

<**run\_directory\_60\_90**> /home/liuli/test/calibration\_tool/transform/LUT/lut\_GL8/0.4\_0.9/</**run\_directory\_60\_90**>

<**stretch\_coef\_60\_90**> 0.4 </**stretch\_coef\_60\_90**>

<**yCoef\_60\_90**> 0.9 </**yCoef\_60\_90**>

拼接时利用lut表选择函数get\_stretchParms() 根据当前处理图像给定的角度选择对应的lut表路径。

量化及对应关系如下：

|  |  |
| --- | --- |
| 角度范围[0,90] | Lut表系数 stretch\_coef\_\_yCoef |
| [0, 20] | 0.4\_0.9 |
| (20, 40] | 0.5\_1 |
| (40, 60] | 0.4\_0.8 |
| (60, 90] | 0.4\_0.9 |

*拼接函数接口：*

**void** stitch\_thetaLot(**IN** Mat raw\_image, **INOUT** Mat &stitch\_fusion, **IN float** theta, **INOUT float** &stretch\_coef, **INOUT float** &yCoef);

*\* function: 给定车位与水平方向的夹角theta,得到拉伸后的斜车位*

*\* parametes:*

*\* IN：* *Mat raw\_image,原始四路鱼眼图*

*\* IN：* *float theta,原车位与水平方向的夹角*

*\* INOUT：* *Mat stitch\_fusion, 变换后的斜车位拼接图*

*\* INOUT：* *float stretch\_coef, 返回的车位拉伸系数*

*\* INOUT：* *float yCoef, 返回的车位展宽系数*

lut*表选择函数：*

*\* function:* *给定车位与水平方向的夹角theta,选择合适的lut表并返回所对应的拉伸系数stretch\_coef,yCoef*

**int** get\_stretchParms(string file\_name, **float** theta, **float** &stretch\_coef, **float** &yCoef)；

3. 坐标转换部分

实现原理：首先将原俯视图的点转换到世界坐标，根据之前的变换公式对世界坐标进行反变换，解出变换后的世界坐标对应的拼接图俯视坐标。

根据birdviewpix2worldpoint() 像素到世界坐标的反推，先考虑进拉伸系数，然后反解出俯视坐标：

worldpoint[0] = (stitch\_point\_x – stitch\_width/2)\*dx;

worldpoint[1] = (stitch\_height/2-stitch\_point\_y)\*dy + (car\_axle\_coord – car\_world\_height/2) + stretch\_coef \* worldpoint[0];

worldpoint[1] = yCoef \* worldpoint[1];

反解：

world\_position[1] = world\_position[1]/yCoef;

**float** stitch\_point\_x = world\_position[0]/dx + stitch\_width/2;

**float** stitch\_point\_y = stitch\_height/2 - (world\_position[1] - (car\_axle\_coord - car\_world\_height/2) - stretch\_coef \* (world\_position[0]))/dy;

坐标转换接口：

**void** birdview2birdview(**IN float** stitchpoint[2], **INOUT float** modifiedpoint[2], **IN float** stretch\_coef, **IN float** yCoef)

***\* function: 给定原俯视图某一点的像素坐标(u,v)，输出对应拉伸后的俯视图像素坐标(u',v')。***

*\* IN：* *float stitchpoint[2] 原俯视图下像素坐标，坐标,[0] u坐标; [1] v坐标*

*\* INOUT：* *float modifiedpoint[2] 拉伸后平行四边形像素坐标* *[0] u‘坐标; [1] v’坐标*

*\* IN: float stretch\_coef, IN float yCoef 逆向转换系数*

两个系数的获取依赖于坐标文件对应的角度,通过调用get\_stretchParms()得到。

4. 使用说明及调用示例

对给定角度的车位，拉伸成倾斜车位，并对相应车位点坐标进行转换

1）. 拼接的两种实现方式：

(1). 调用 stitch\_thetaLot(IN Mat raw\_image, INOUT Mat stitch\_fusion, IN float theta)接口,直接实现拼图(要求支持.xml相对路径和四路原始图排列顺序左上到右下为前后左右; 也可根据需要修改该函数内部代码，重新生成.so库调用);

(2). 直接调用 get\_stretchParms("./XML/parameter\_GL8\_08\_23.xml", theta, stretch\_coef, yCoef);

stitch\_fusion = lutTable\_generate\_stitchFusion(front\_image, back\_image, left\_image, right\_image, true, true, true, true);

两个函数，在get\_stretchParms()函数输入角度实现拼图。

2）. 标注文件的坐标转换：

(1). get\_stretchParms("./XML/parameter\_GL8\_08\_23.xml", theta, stretch\_coef, yCoef);

birdview2birdview(stitchpoint, modifiedpoint, stretch\_coef, yCoef);

3）.当前文件夹下的lut为720×621大小的，可根据需要自行生成512×512大小的(通过修改.xml文件实现)。

自行生成.so。

5. 不同尺寸拼接融合图对.xml文件参数修改说明：

对应参数：

*<!--MG 汽车宽度 -->*

<car\_world\_width>1804</car\_world\_width>

<!-- 汽车长度 -->

<car\_world\_height>4612</car\_world\_height>

<!-- 后轴中心到车前距离 -->

<car\_axle\_coord>3594</car\_axle\_coord>

*<!--GL8 汽车宽度 -->*

<car\_world\_width>1878</car\_world\_width>

<!-- 汽车长度 -->

<car\_world\_height>5256</car\_world\_height>

<!-- 后轴中心到车前距离 -->

<car\_axle\_coord>4198</car\_axle\_coord>

*<!--RW 汽车宽度 -->*

<car\_world\_width>1919</car\_world\_width>

*<!-- 汽车长度 -->*

<car\_world\_height>4678</car\_world\_height>

*<!-- 后轴中心到车前距离 -->*

<car\_axle\_coord>3752</car\_axle\_coord>

环视融合图尺寸：

**～～～621×720**

MG:

对应真实物理尺寸：

8000+1804+8000 × 8000+4612+8000

<!-- 前后左右视野 -->

<front\_word\_view>8000</front\_word\_view>

<rear\_world\_view>8000</rear\_world\_view>

<left\_world\_view>8000</left\_world\_view>

<right\_world\_view>8000</right\_world\_view>

GL8:

<!-- 前后左右视野 -->

<front\_word\_view>7678</front\_word\_view>

<rear\_world\_view>7678</rear\_world\_view>

<left\_world\_view>7964</left\_world\_view>

<right\_world\_view>7964</right\_world\_view>

**<!--融合尺寸512×512，.xml做如下修改：-->**

<!--俯视展开左侧像素的高度-->

<left\_pixel\_height>512</left\_pixel\_height>

<!--尺寸改为512×512\_40后： 目前采用的-->

<!-- MG.xml前后左右视野 -->

<front\_word\_view>8000</front\_word\_view>

<rear\_world\_view>8000</rear\_world\_view>

<left\_world\_view>9404</left\_world\_view>

<right\_world\_view>9404</right\_world\_view>

<!-- GL8.xml前后左右视野 -->

<front\_word\_view>7678</front\_word\_view>

<rear\_world\_view>7678</rear\_world\_view>

<left\_world\_view>9367</left\_world\_view>

<right\_world\_view>9367</right\_world\_view>

*<!--RW 前后左右视野 -->*

<front\_word\_view>7967</front\_word\_view>

<rear\_world\_view>7967</rear\_world\_view>

<left\_world\_view>9347</left\_world\_view>

<right\_world\_view>9347</right\_world\_view>

<!--尺寸改为512×512\_28后：-->

<!-- MG.xml前后左右视野 -->

<front\_word\_view>5023</front\_word\_view>

<rear\_world\_view>5023</rear\_world\_view>

<left\_world\_view>6427</left\_world\_view>

<right\_world\_view>6427</right\_world\_view>

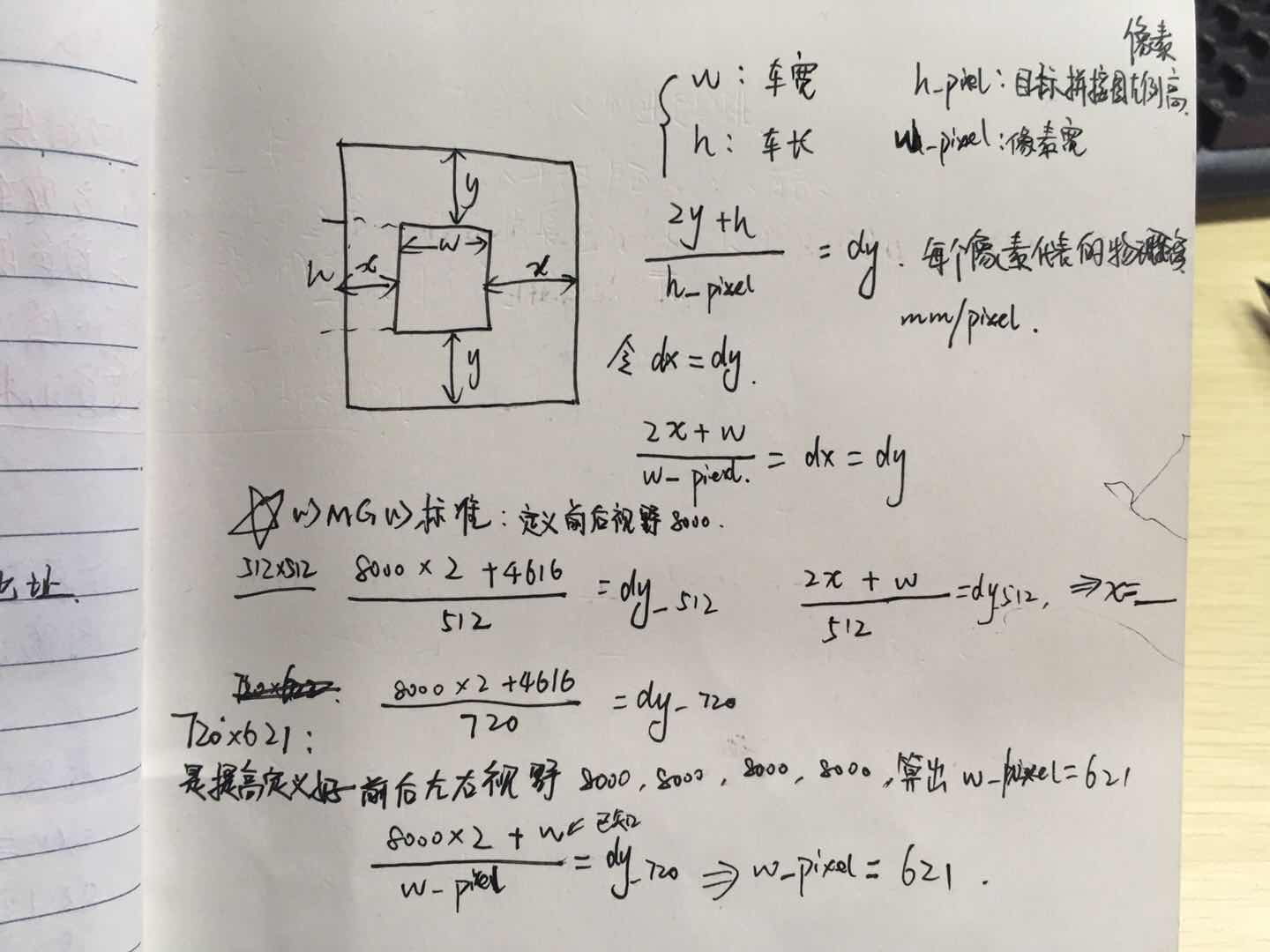
<!-- Gl8.xml前后左右视野 -->

<front\_word\_view>4700</front\_word\_view>

<rear\_world\_view>4700</rear\_world\_view>

<left\_world\_view>6389</left\_world\_view>

<right\_world\_view>6389</right\_world\_view>

计算原理：

5. 对任意角度的正车位，生成相应的拉伸系数

函数说明：

*/\* name: get\_stretch\_coef*

*\* function: 给定目标拉伸角度alpha和车位与水平方向的夹角theta,算出拉伸系数stretch\_coef*

*\* parametes:*

*\* IN：* *float alpha 给定目标拉伸角度; float theta 车位与水平方向的夹角*

*\* INOUT：* *float stretch\_coef 输出拉伸系数*

*\* std::cout << tan(45\*(2\*3.14)/360) << endl; //角度制算斜率*

*\* std::cout << atan(1) \* 360/(2\*3.14) << endl; //斜率算角度*

*\*/*

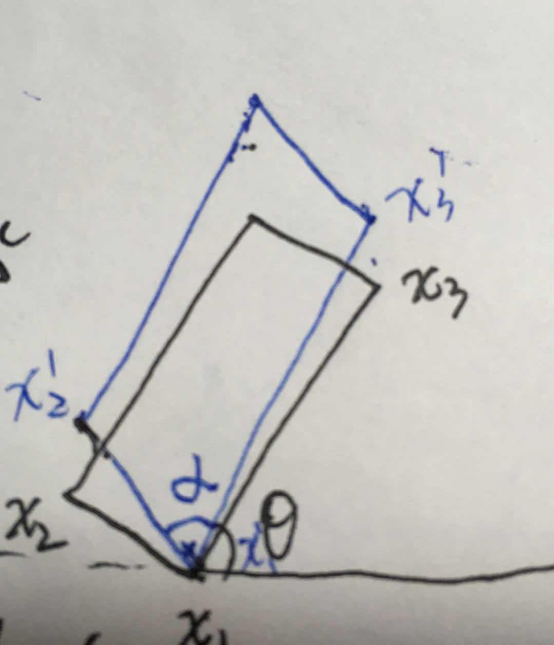
**float** get\_stretch\_coef(**IN float** alpha, **IN float** theta, **IN float** yCoef)

*//float get\_stretch\_coef(IN float cord[6], IN float alpha, IN float yCoef)*

参数说明:

alpha (0,180) : 目标拉伸角度。平行四边形斜车位的夹角，对应拉伸程度。

Theta [0,90]: 正车位与图像水平方向的夹角，由车位坐标点求得。



正车位到斜车位的转换公式：

x’ = x;

y’ = a\*x + b\*y; 其中：a = stretch\_coef; b = stretch\_coef \*yCoef = a \* yCoef;

正车位坐标：

x1(x1,y1)

x2(x2,y2)

x3(x3,y3)

斜车位坐标：

x1’(x1,a\*x1+b\*y1)

x2’(x2,a\*x2+b\*y2)

x3’(x3,a\*x3+b\*y3)

斜率：

k1 = Kx2x3 = tan(theta);

k2 = Kx1x2 = tan(theta + 90);

k1’ = Kx2’x3’ = (y3’ - y2’) / (x2 - x3 ) = a + b \* k1; （左上角为坐标原点）

同理： k2’ = Kx1’x2’ = a + b \* k2;

其中， b = a \* yCoef, yCoef 提前指定。

角度关系求解a:

atan(k1’) - atan(k2’) = alpha

整理关于a的一元二次方程为：

a^2\*(1 + yCoef \* (k1 + k2) + yCoef^2 \* k1\*k2) + a \* (yCoef\*(k2-k1) / tan(alpha)) + 1 = 0;

编程实现a的求解。