

# BEV车位自动化标注算法设计文档

草稿区

反转车位的调转时间点

## 1 设计功能

本模块为使用感知车位检测模型检测结果、EM融合结果、定位输出信息，进行BEV车位检测模型训练数据的自动化标注功能。

## 2 设计思路及折衷

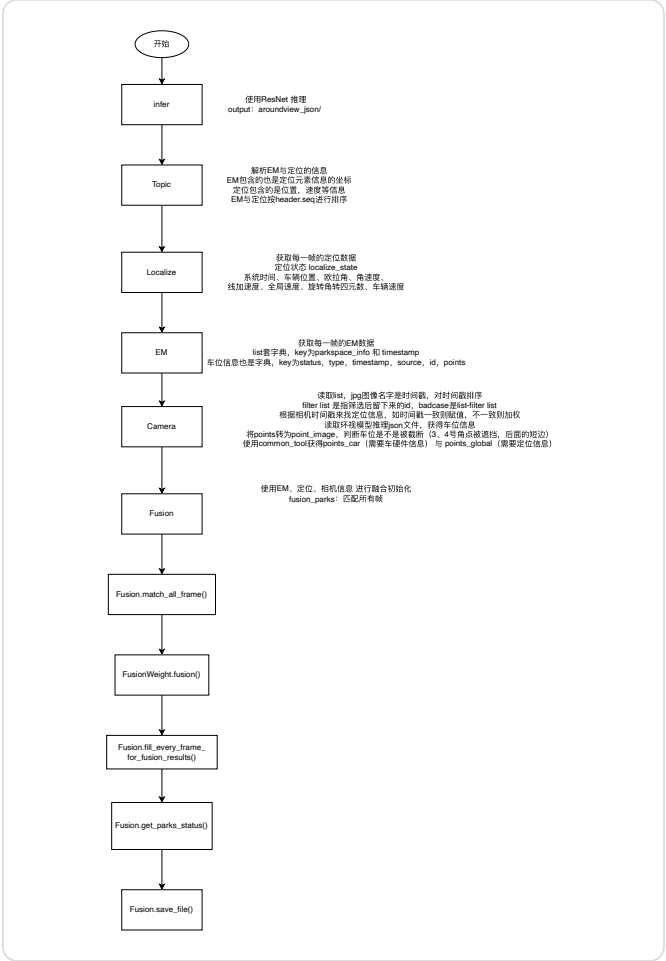
### 2.1 总体设计思路

自动化标注算法采用了EM车位融合策略，基于视频流中的单帧图像检测结果，进行全局车位融合，最后进行标注。总体分为三个部分：

- 1) 单帧车位检测部分：采用感知车位检测模型获得检测结果。
- 2) 多帧车位匹配部分：基于定位信息与IOU的规则将多帧中同一车位匹配。
- 3) 车位坐标融合部分：匹配后的同一车位角点坐标融合以加权为主。

融合车位使用的坐标建立在3D世界坐标系，3D坐标的来源是基于图像检测的2D图像坐标位置+定位信息，进行坐标变换，转到3D世界坐标系。融合车位对截断车位进行填补，默认补齐到5m。





## 2.2 设计场景覆盖

与BEV车位检测模型一致

## 2.3 数据分布

与BEV车位检测模型一致

## 3 总体设计

### 3.1 融合依赖数据源

自动化标注数据依赖源有三条，分别是定位信息、EM融合信息、感知车位检测结果。

1	数据源	获取方式	筛选规则	信息	含义	
2	定位topic	实车录制， ebag包解析	1.定位状态为Tracking  2.包含系统时间  3.包含位置、欧拉角、 角加速度、线加速度、 速度、四元数	timestamp	时间戳	16
3				pos	车身位置 (世界坐标)	ar
4				ypr	车身姿态	ar

5	EMtopic		4.车速由四元数与speed_global计算得到 ？为什么要单独算车速而不用自带的speed_global	angle_v		角加速度	ar
6				acc_v		线加速度	ar
7				speed		车速	ar
8				quaternion		四元数	ar
9			对于每一个车位 1.车位状态为FREE或OCCUPIED 2.车位类型为垂直车位，平行车位，斜车位 3.车位来源为视觉或超声波 4.车位已判断 5.包含全局id 6.包含环视时间戳	timestamp		时间戳	16
10				packspace_info -> list	status	车位状态	'F' 'O'
11					type	车位类型	'V' 'P' 'S'
12					source	车位来源	'V' 'U'
13					id	全局id	int
14					timestamp	时间戳	16
15					points	车位角点 (世界坐标系)	ar
16	Camera	大图模型 离线推理生成 (640*640环视图作为输入)	对于每个车位 1.环视推理结果包含车位信息 2.*有匹配的定位信息 (影响是否有全局车位)  ----- • 读取环视推理结果json • 读取车位角点（图像坐标系） • 判断车位是否截断 • 计算车位角点（车身坐标系） • 如果存在匹配定位信息，则计算车位角点（世界坐标系）；否则车位角点（世界坐标系） =None	timestamp		时间戳	16
17				is_get_loc		是否有定位数据	'T' 'F'
18				loc	timestamp	时间戳	16
19					pos	车身位置 (世界坐标系)	ar 单
20					speed	车速	a
21					acc_v	线加速度	ar
22					ypr	车身姿态	ar
23				parkspace_info -> list	points_image	车位角点 (四路环视坐标系)	ar
24					points_car	车位角点 (车身坐标系)	ar

2023/8/14 11:41

BEV车位自动化标注算法设计文档

25					points_global	车位角点 (世界坐标系)	ar
26					is_truncated	车位是否被截断	'Tr 'Fa
27				speedbump_info ->list			
28				footwalker_info->list			
29				arrow_info->list			
30				lane_info->list			

3.2 融合对象类

融合类对象参考EM融合车位规则，利用感知推理结果、EM、定位topic进行多帧融合，获得全局车位信息，再将全局车位信息回填至每一帧图像中。离线融合方式相比EM结果的优点：可以利用未来帧信息，感知模型使用离线大模型。

1	fusion.fusion_parks() 实现融合				
2	步骤	函数	目的	输出	实现方
3	全局车位匹配	fusion.fusion_weight()	获得全局车位	match_results	 BEV车位自 法设计文
4	车位坐标融合	fusion_weight.fusion()	提高全局车位精度	fusion_results	 BEV车位自 法设计文
5	填补融合车位缺失帧信息	fusion.fill_every_frame_for_fusion_results()	将全局车位回填至每一帧图像	fusion_results	 BEV车位自 法设计文 
6	获得EM车位信息	fusion.get_em_parks_info()	获取全局车位对应的EM车位信息	<ul style="list-style-type: none"><li>fusion_results[idx] ["em_timestamps"]</li><li>fusion_results[idx] ["em_parks"]</li></ul>	调用em.get_parks_info()遍历em_data中每一个车位，记录其IOU分数；若最高IOU分数[0.5]，则认为该帧融合成功，记录该帧车位信息

3.2.1 全局车位匹配策略

通过使用所有时间帧的车位信息 (camera.park\_data)，将单帧车位结果匹配，生成全局车位信息，为每一个车位关联一个唯一全局id。

逐帧遍历图像帧车位检测结果进行整理，对于每一帧，维护两个队列 未匹配车位列表（来自当前帧图像检测结果） 未更新车位列表（来自已匹配全局车位列表）

匹配思想：

- 将单帧无独立id的车位转为有时间序列信息的全局车位
- 匹配过程中维护两个队列，一个是当前帧检测车位的list，一个是全局检测车位的list
- 处理顺序：当前帧车位 -> 与已有全局车位匹配 -> 更新未匹配全局车位 -> 未匹配当前帧车位加入全局车位

### 3.2.1.1 当前车位与全局车位的匹配规则

```
iou_score, is_reverse = common_tool.cal_overlap(frame_parks[k]["points_global"], match_results[i]["filter"])
```

判断车位是否反向

判断规则：

同时满足以下条件则认为以下车位检测反向：

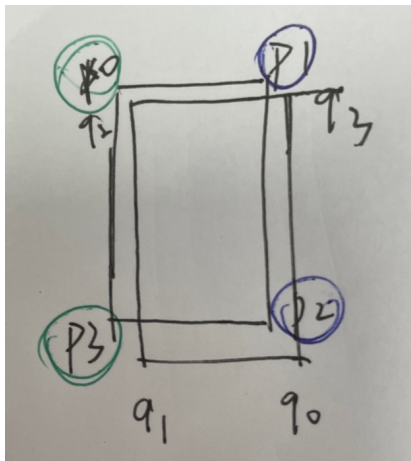
- 两个车位短边中心连线向量的夹角 > 135度 (180 - 45)
- 车位前短边的长度区间为 2~10 (m)

如果车位是反过来了，则调整3、4号角点的位置（因为默认的3、4号角点进行了补边，不一定是真实的坐标）。注意这里只调整3、4号角点位置，不将其与1、2号角点调换。

调整规则：

$$p2 = p1 + \text{车位长度} * \frac{\text{两点向量}(p2-p1)}{\text{两点长度}|p1p2|}$$

$$p3 = p0 + \text{车位长度} * \frac{\text{两点向量}(p3-p0)}{\text{两点长度}|p0p3|}$$



### 计算车位IOU

判断规则：车位IOU > 0.5 则认为匹配成功

计算公式：

$$IOU = \frac{inter\ area}{min(area1, area2)}$$

3.2.1.2 全局车位匹配成功后世界坐标更新公式

```
fusion.update_match_result(match_results, int(max_idx), frame_parks[k], frame_data,
is_reverse_flags[max_idx])
```

世界坐标的更新分两种情况：车位正常与车位反转，两种情况处理方式不同。⚠️ 不更新图像车位坐标。

1.若该帧车位检测反转 (**is\_reverse=True**)，不进行车位融合，将全局车位反转（12号角点为34号角点，34号角点为12号角点），同时12角点坐标作为全局车位的12号角点坐标。

```
全局坐标[2:] = 全局坐标[0:2]
全局坐标[0:2] = 当前帧坐标[0:2]
```

**考虑因素：**已有的融合车位后短线可能为脑补角点，置信度存疑，车位检测反转时相信前线角点的准确性，故不进行融合。

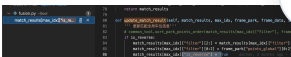
2.若该帧车位检测正常 (**is\_reverse=False**)，车位检测方向与历史帧一致，则进行车位融合。融合方式为加权融合，计算公式为：

$$GL_t = r * GL_{cur} + (1 - r) * GL_{t-1}$$

其中超参数r=0.4，通过 `config_param.py` 中的 `match_filter_ratio` 参数配置。

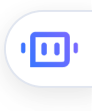
**考虑因素：**相信离线模型的检测性能，未采用复杂的融合策略，后续可评估更复杂融合策略性能

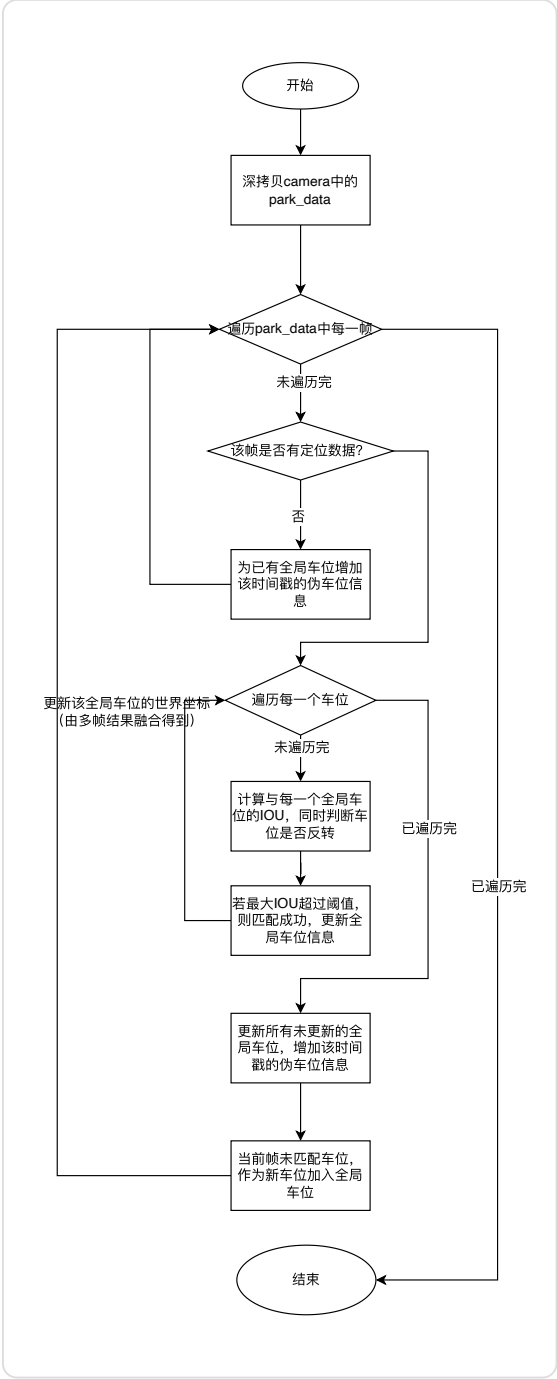
3.2.1.3 match\_results 数据结构与本阶段流程图

1	match_results->list				
2	信息		含义	格式	来源
3	filter		融合车位角点 (世界坐标系)	array( [x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4])	
4	is_reverse		车位是否存在过反转	'True' 'False'	 车位存在不小于一帧的检测反转时该变量为True
5	scene		车位场景	{scene_id: cnt}	统计每一帧的scene，在融合取最大次数对应scene进行
6	frames->list (list长度等于检出该车位的图像帧数)	timestamp	时间戳	16位	图像帧的时间戳
7		loc	定位信息	同CameraTopic-loc 'None'（无匹配定位信息时）	拷贝camera中信息
8		id	全局id编号	int 从0开始	? id和match_results idx—是否必要?

9		is_reverse	是否反向	'True' 'False'	计算iou时给出
10		is_truncated	是否截断	'True' 'False'	拷贝camera中信息
11		points_global	车位角点 (世界坐标系)	array( [x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4]) None (无匹配定位信息时)	拷贝camera中信息
12		points_car	车位角点 (车身坐标系)	array( [x1 y1 0 x2 y2 0 x3 y3 0 x4 y4 0])	拷贝camera中信息
13		points_image	车位角点 (四路环视坐标系)	array( [x1 y1 0 x2 y2 0 x3 y3 0 x4 y4 0])	拷贝camera中信息

本阶段流程图如下：





### 3.2.2 车位坐标融合策略

对于每帧的全局车位，基于其时间区间内的车位信息进行融合。若该区间内仅有当前帧为None，则会被填上坐标值；若该区间内少于5帧检出，则即使该帧原有检出，也会覆盖为None。





## 融合处理流程：

- 1.划定窗口，取历史m帧和未来的n帧；
- 2.在窗口内，计算每个点的平均值和标准差，利用  $\text{mean} \pm 3 * \text{std}$  的范围去剔除异常值；
- 3.加权平均，理论上距离当前时刻越近，那么标定误差、定位误差、检测误差都会与当前时刻更接近，更符合当前时刻真值的分布。其中权重赋值如下：

- a) 比如窗口size为7，当前帧的索引值为6，取历史4帧和未来2帧，那么窗口的索引值为  $a = [2, 3, 4, 5, 6, 7, 8]$ ；
- b) 每一帧的索引值与当前帧的索引值相减，取绝对值：  $b = [4, 3, 2, 1, 0, 1, 2]$ ；
- c) 用  $\max(b) - b + 1$ ，得到  $c = [1, 2, 3, 4, 5, 4, 3]$ ；
- d) 再取  $d = \text{power}(c, n)$ ，最后  $\text{weight} = c / \text{sum}(d)$ ，power是幂运算，n作为可调整参数，n越大，当前时刻的权重越大。

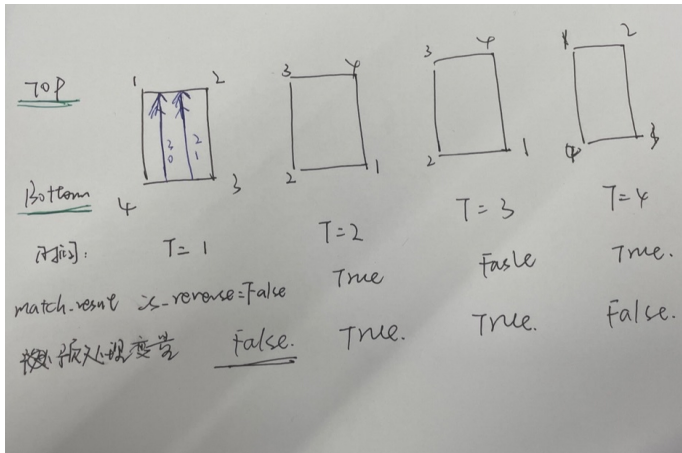
```
park_index = np.array(park_index, dtype=float)
step = np.absolute(park_index - idx)
step = np.max(step) - step + 1
weight = np.power(step, self.cfg.fwm_power)
weight = weight / np.sum(weight)
```

- 4.对于漏检帧和距离较远从而无法检测的帧，以车辆位移最近的融合帧投影到该时刻作为真值；
- 5.借鉴环视车位标注规则，针对某一时刻图像，只有当车位的两个前角点都在环视图像内或者超出图像边界不超过20cm，才会将这个车位作为该时刻环视图的真值车位。

- 1.在局部异常值过滤之前，增加了全局异常值， $\text{mean} \pm 3 * \text{std} \rightarrow \text{mean} \pm 2 * \text{std}$
- 2.增加最少检测帧数，目前整个数据段检测帧数不少于5帧的车位才会融合
- 3.优化图像截断点处理。利用全局所有检测帧信息，若有实际后角点的检测帧，则预处理时补长到实际长度再进行融合，否则默认补长到5米
- 4.完成车位判空关联
- 5.完成空间车位关联

### 3.2.2.1 预处理

预处理的目的是调整要进行融合的滤波车位 (`fusion_results[i]["filter"]`) 和每一帧的全局车位 (`fusion_results[i]["parks"][j]["points_gloabl"]`)。



### 滤波车位调整策略：

保留12号角点，对于34号角点，使用车位线长边长度进行调整。

&lt;/&gt; 3号角点的调整实现

Plain Text

收起 ^

```
1 parkline_12 = (park_filter[2] - park_filter[1]) / (np.linalg.norm(park_filter[2] -
    park_filter[1], ord=2) + 1e-12)
2 park_filter[2] = park_filter[1] + default_parkline_length * parkline_12
```

物理含义：计算单位方向向量，保持方向一致，长度使用默认长度

### 单帧全局车位调整策略：

单帧全局车位调整分两步，第一步针对被截断车位；第二步针对所有车位。

对于被截断车位，使用车位线长边长度调整34号角点位置。调整方式与滤波车位方式一致。

对于所有车位，计算角点均值与标准差，对于偏移 $2 \times \text{std}$ 的角点，使用平均值替代；此外对于反转车位，调换角点顺序，使得每一帧中同一车位的角点顺序保持一致。

### 车位线长边长度获取：

默认长度为5m（config\_param.py中default\_parkline\_length给定）

将车位两条短边分为top与bottom，top由1、2号角点构成线段，bottom由3、4号角点构成线段

如果车位底边bottom未截断地出现5次（config\_param.py中fwm\_truncated\_scale给定），则使用top与bottom的距离来极端、计算车位线长边长度。

&lt;/&gt; 车位线长边长度计算实现

Python

收起 ^

```
1 if len(parks_global_bottom) > self.cfg.fwm_truncated_scale:
2     bottom_mean = np.mean(parks_global_bottom, axis=0)
3     parkline_mean03 = bottom_mean[1] - top_mean[0]
4     parkline_mean12 = bottom_mean[0] - top_mean[1]
5     parkline_mean = (parkline_mean03 + parkline_mean12) / 2.0
6     default_parkline_length = np.linalg.norm(parkline_mean, ord=2)
```

### 3.2.2.2 融合处理

融合处理的目的是调整每一帧的全局车位（fusion\_results[i][j]["points\_gloabl"]）。

#### 处理思路：

对于第j帧的全局车位，使用[fwm\_old\_frame\_num, fwm\_future\_frame\_num]区间内所有非None的全局车位进行融合。（区间范围由config\_param.py进行配置，目前为 [j-20, j+15]）

若区间内可使用融合车位帧少于fwm\_need\_fusion\_frame\_num（目前为5帧），则该帧全局车位直接赋值None；否则使用区间内车位进行融合。

#### 融合策略：

[pp\_1, pp\_2, ..., pp\_n]为区间内的全局车位列表；mp为平均车位角点；diff为全局车位与平均车位的差值列表，如果 $\text{diff} > 2 \times \text{std}$ ，则 $\text{diff} = 0$ ；w为权重列表，距离当前帧越近，则权重越大；fp为融合车位角点



$$mp = mean([pp_1, ..., pp_n])$$

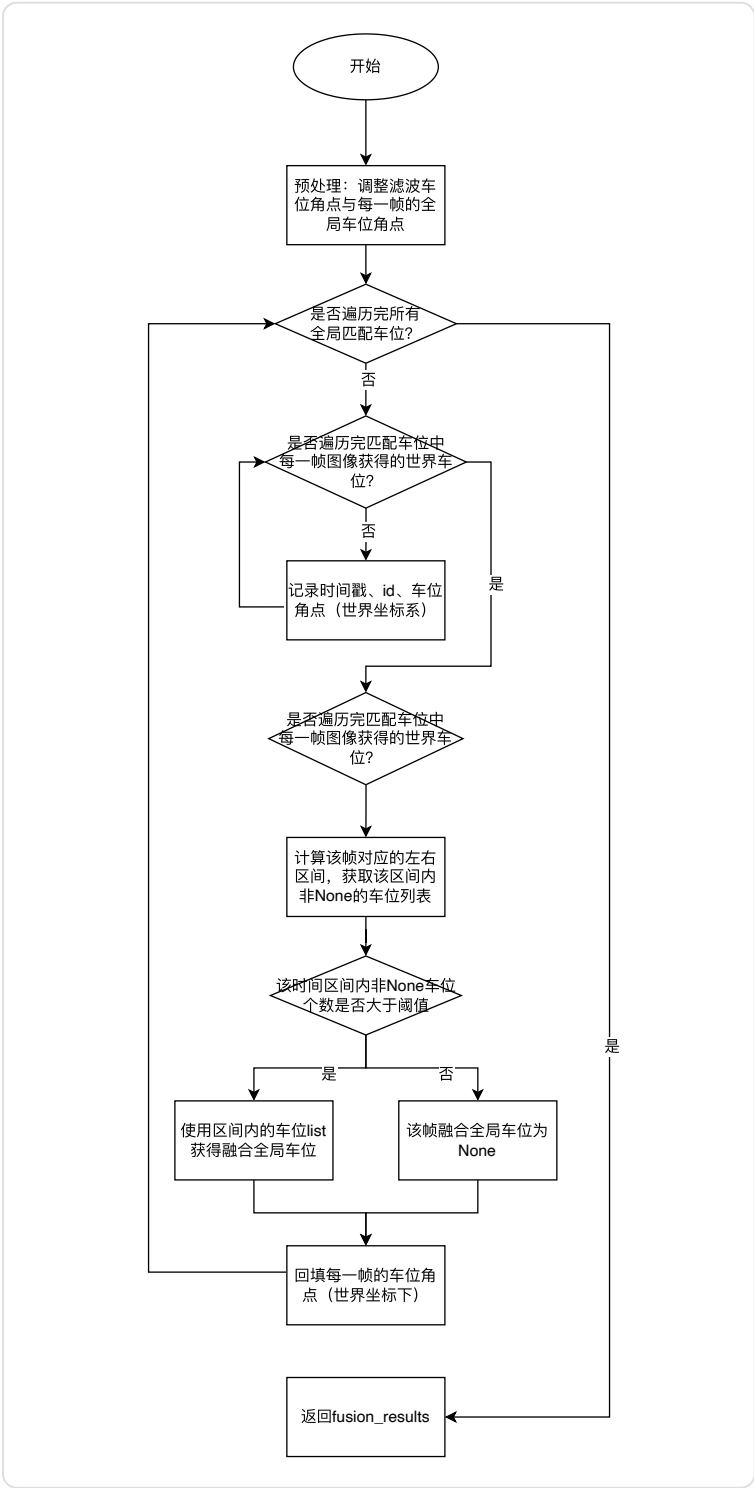
$$w = max(abs(n - idx)) - abs(n - idx) + 1; w = normalize(w)$$

$$w = max(abs(n - idx)) - abs(n - idx) + 1; w = normalize(w)$$

$$fp = mp + w * diff$$

3.2.2.3 本阶段流程图

本阶段fusion\_results结构同match\_results，流程图见下：



### 3.2.3 每帧缺失信息回填

#### 3.2.3.1 回填缺失的时间帧信息

在先前处理中，fusion\_results中parks list长度不固定，范围为首次检出帧到视频最终帧。为了统一信息，将对parks list填充至相同长度，首帧由首次检出帧变更为首次包含定位信息帧。填充通过双重遍历camera\_data与fusion\_results完成，此外还对每一个fusion\_results[i][“parks”][j]增加[“is\_fusion”]字段，用来描述fusion\_results[i][“parks”][j][“points\_global”]是否进行了[时间区间车位融合](#)。

#### 3.2.3.2 更新其余坐标系下的车位角点坐标

在先前处理中，仅对世界坐标系下的车位角点（point\_global)进行了融合，未更新其他坐标系下对应的车位角点。为了统一坐标信息，进行双重遍历fusion\_results中每一个parks[idx]以更新坐标，由于坐标系变换需要定位信息，若该帧没有匹配的定位信息则直接跳过。

对于已融合车位（fusion\_results[i][“parks”][idx][“is\_fusion”]=True)，直接更新其车身坐标系、图像坐标系下的车位角点坐标；对于未有融合车位（此时global\_points是None），则寻找该车位所有时间帧中最近欧式距离的已融合车位，将该融合车位信息赋值给global\_points，接着更新对应的车身坐标系、图像坐标系下的车位角点坐标。

#### 3.2.3.3 fusion\_results数据结构

1	fusion_results->list				
2	信息		含义	格式	来源
3	filter		融合车位角点 (世界坐标系)	array( [x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4])	
4	is_reverse		车位是否反转	'True' 'False'	
5	frames->list (list长度等于有定位信息的图像帧数)	timestamp	时间戳	<str>16位	图像帧的时间戳
6		loc	定位信息	同CameraTopic-loc 'None'（无匹配定位信息时）	拷贝camera中信息
7		id	全局id编号	int 从0开始	? id和match_results idx要?
8		is_reverse	是否反向	'True' 'False'	计算iou时给出 ⚠ 已在融合时将车位方
		is_truncated	是否截断	'True'	拷贝camera中信息

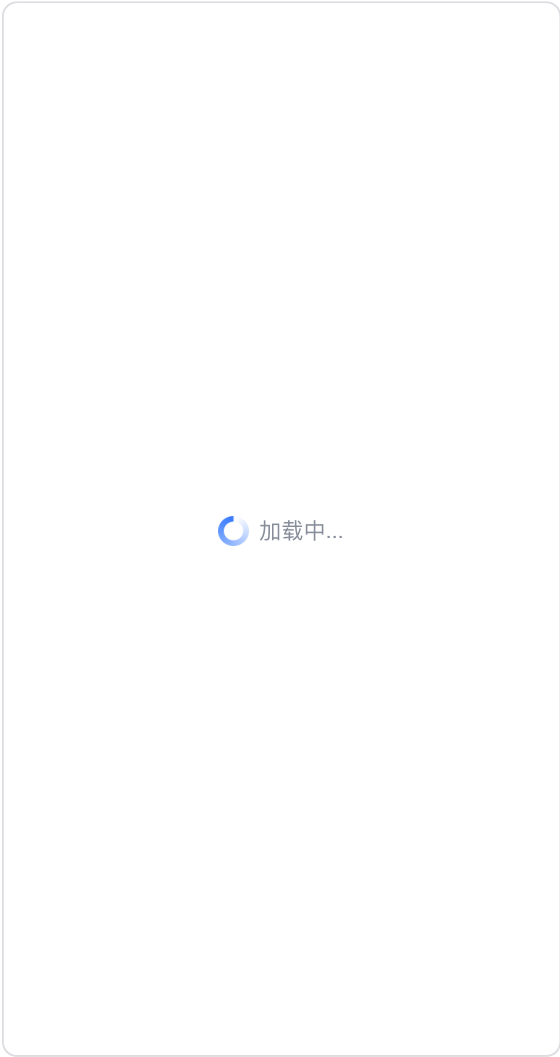
9					'False'	
10		is_fusion	是否进行车位融合		'True' 'False'	进行车位融合操作后，若points_global=None，则（可融合帧数<5帧）；融合  ⚠ is_fusion=False时points_global不为None（如果parks列表中的车位对该帧进行赋值）
11		points_global	车位角点 (世界坐标系)		array( [x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4]) None（无匹配定位信息时）	使用时间区间内的车位进行融合
12		points_car	车位角点 (车身坐标系)		array( [x1 y1 0 x2 y2 0 x3 y3 0 x4 y4 0])	基于融合后的points_global得到（依赖车身位姿信息）
13		points_image	车位角点 (四路环视坐标系)		array( [x1 y1 0 x2 y2 0 x3 y3 0 x4 y4 0])	基于points_car进行变换尺寸、分辨率、轮宽）
14		points_bev_image	车位角点 (五路环视坐标系)		array( [x1 y1 0 x2 y2 0 x3 y3 0 x4 y4 0])	基于points_image进行变换bev_offset_x、bev_offset_y
15	em_timestamps ->list		时间戳		16位	拷贝EMtopic
16	em_parks->list	status	车位状态		'FREE' 'OCCUPIED'	
17		type	车位类型		'VERTICAL' 'PARALLEL' 'SLANTED'	
18		source	车位来源		'VISION' 'ULTRASONIC'	
19		id	全局id		int 从0开始	
20		timestamp	时间戳		16位	
		points	车位角点 (世界坐标系)		array( [x1 y1 z1 x2 y2 z2 x3 y3 z3	

			x4 y4 z4])	
21	park_dict->dict	key: parks[idx]["timestamp"]	value:parks[idx]	以parks中每一元素的时 元素为value，赋值构造

22

### 3.3 自动化标注标签生成

融合结果以两种形式进行存储，一是将融合后的全局车位存储，在多车型转换及车位调整中使用；二是将每一帧图像对应的标注信息保存位json格式，在模型训练时使用。



#### 3.3.1 全局车位字典格式

#### 3.3.2 json文件格式

</>JSON | 收起 ^

```
1 {
2     "image_id": "zzz.jpg",           // <str> -- 图像名，以16位时间戳命名
```

```

3      "preData": {                                // <dict> -- 标注真值
4          "loc": {                                // <dict> -- 定位信息
5              "acc_v": {"x":, "y":, "z":},        // <dict> <float> -- 车身线加速度
6              "pos": {"x":, "y":, "z":},          // <dict> <float> -- 车身位置
7              "speed": {"x":, "y":, "z":},        // <dict> <float> -- 车速
8              "ypr": {"x":, "y":, "z":}          // <dict> <float> -- 车身姿态
9          },
10         "parkingspace": [                        // <list> -- 车位信息 k parkspace in
'parkspace` list
11             {
12                 "id": ,                          // <int> -- 车位全局id
13                 "p": [                            // <List> --车位角点（四路环视图像坐标系）4
points in p
14                     {"x":, "y":, "z":},          // <dict> <float> -- 1\2\3\4号角点顺
次
15                     {},
16                     {},
17                     {},
18                 ],
19                 "p_bev": [{},{},{},{}],          // <List> --车位角点（五路环视图像坐标系）4
points in p
20                 "p_car": [{},{},{},{}],          // <List> --车位角点（车身坐标系）4 points
in p
21                 "p_global": [{},{},{},{}],        // <List> --车位角点（世界坐标系）4 points
in p
22                 "source":,                      // <str> -- 车位来源
23                 //      “VISION” 感知车位 “ULTRASONIC” 超
声波车位 “EM_VISION” 空间车位
24                 "status":                        // <str> -- 车位状态
25                 //      "UNKOWM" 未知 “OCCUPIED” 非空车位
“FREE” 空车位
26             },
27         ],
28         "timestamp": zzz                        // <int> -- 16位时间戳
29     }
30 }
```

### 3.3.3 可视化图像

黄色角点：1号角点

品红色：2、3、4号角点

## 3.4 多车型的标注结果统一

不同车型的两台车，在同一起点出发，到达同一终点，若每一时刻位置相同，则其定位信息得到的位姿是否相同？

### 3.5 结合环视检测 results 与 BEV 检测结果

只需更新 json 读取的信息，

aroundview\_json

infer\_result\_json

#### 测试样例

