Hu_Safe_Local_Motion_Planning_With_Self-Supervised_Freespace_Forecasting_CVPR_2021_paper

开源实现： here

Background

常规的轨迹预测的技术栈为了表示一个动态场景，通常采用object-centric的方式。模块包括：物体检测、物体追踪还有物体预测模型。这些模型的缺点是需要大量的人力对场景进行逐帧的数据标注（包括3D轨迹和语义分类）。

为了解决常规场景的人力代价昂贵的缺点，作者提出了一个freespace-centric的动态场景表示方式，基于该表达方式来完成motion planning.这种表达方式有两个好处：

- 对于motion planning这项工作而言，freespace-centric的表达方式更加符合直觉。我们只需要知道哪些路是可以走的，对于不能走的路，我们只需要知道路被挡住了，而不在乎是被什么东西挡住了。
- 在freespace-centric的语境下，freespace forecasting工作的训练数据是不需要人工标注的，可以直接由LiDAR提供(can be readily obtained by raycasting measurements from a depth sensor)。

Contributions

- 采用self-supervised方式训练了一个freespace predictor.
- 给定一个黑盒montion-planner，可以通过这个predictor鉴别哪些方案可能会在未来引发碰撞。
- 在训练planner的过程中，predictor预测出的future freespace可以作为planner的监督来源之一。

Method

Definition

(X, Y)是一个pair，X表示的是一系列aligned sensor data,Y表示一系列ego-vehicle trajectory.

$u = (x, y, t) \in U$，U是spacetime voxel grid. (x, y)表示俯瞰图坐标，t表示时刻。

$ray(u; X, Y) \in \{-1, 0, 1\}$表示在$(X, Y)$下$u$的状态。-1表示free, 0表示unknown, 1表示occupied.

不过事实上，这个unknown状态没有任何用处，后面我们会提到。

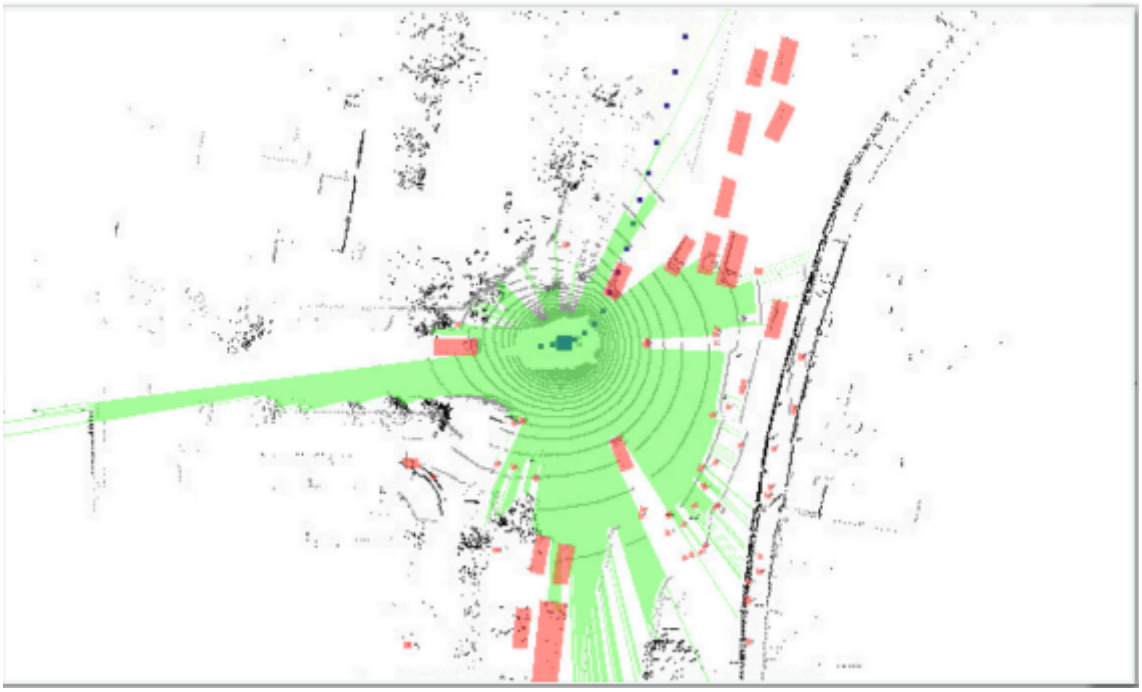
Compute freespace

由于LiDAR的输出是3D点云，与我们表示的2D-freespace有一定差距，所以需要一个转换算法。转换的方式如下：

- 利用robust ground segmentation algorithm识别出LiDAR的在地面的返回。
- 将ground丢弃，然后进行wall tracking算法，得到该帧的freespace表示。

注：wall tracking的直观理解here

freespace computation的结果如下：



图中的绿色部分是freespace，红色部分来源于常规的obstacle detecting.

Forecasting freespace

为了计算的方便，我们丢弃了某个点的unknown状态，所以这个问题可以建模为一个二分类问题，即某一个点是不是freespace.

假设historical pair为(X_1, Y_1)，future pair为(X_2, Y_2)，那么损失函数为：

$$Loss = BCE(\sigma(f_\theta(u; X_1, y_1), ray(u; X_2, Y_2)))$$

f是一个卷积神经网络，θ是参数。

residual forcasting

事实上，在许多场景向future freespace和historical freespace并没有很大差异。所以我们可以通过对historical freespace进行线性差值来预测freespace大致是什么样的，然后再通过一个非线性的residual forecasting模块进行修正。形式化来说，上述的f可以分解为以下两个部分：

$$f_\theta(u; X_1, Y_1) = f_\alpha(u; X_1, Y_1)) + f_{\hat{\theta}}(u; X_1, Y_1)$$

前者表示线性插值的部分，后者表示residual forecasting的部分。

Planning with forcasted freespace

Behavior cloning

- **input:** (X_1, Y_1)
- **output:** Y_2，an expert-like trajectory.

前文中所产生的predictor可以对该结果进行评估，决定这个轨迹到底是不是一个合法的轨迹。形式地，合法轨迹的定义是：

Y_2是一个合法的轨迹，当且仅当$\forall u \in Y_2, ray(u; X_1, Y_1) = freespace$

但是事实上，我们的predictor得到的是一个soft的probability，我们如何将其转化为hard的decision呢?

$q = \wedge_{u \in Y_2}[f_\theta(u; X_1, Y_1) \leq \tau]$，$\tau$是自己限定的一个阈值，表示该voxel被占用的概率。

当且仅当q为真时，该轨迹合法。当这个轨迹不合法时，会采用fall-back option，例如急刹车。

Inverse optimal control

IOC方法为地图上的每个voxel都赋予了相应的权重，因此它所生成的轨迹的权重就是轨迹上的所有voxel的权重之和。形式表示如下：（这里的权重是越小越好的）

$C_\psi(Y_2; X_1, Y_1) = \sum_{u \in Y_2} cost_\psi(u; X_1, Y_1)$

可以注意到，我们的predictor也为每个voxel赋予了权值$c \in [-1, 1]$，越接近1表示这个voxel越有可能是障碍。因此，可以直接通过相加的方式将predictor加入到supervision当中。

$C_{\psi,\theta}(Y_2, X_1, Y_1) = \sum_{u \in Y_2}[(cost_\psi + \gamma f_\theta)(u; X_1, Y_1)]$

其中γ是超参数，如果希望predictor在loss计算中影响较大，可以将其增大。

Learning to plan with future freespace

得到predictor之后，我们相当于已经有了future freespace以及对应的cost map.我们将基于这个map进行trajectory planning.

一个朴素的想法是，ground-truth Y_2的cost之和是最小的，即：

$C_\psi(Y_2; X_1, Y_1) \leq min_{Y \in SY} C_\psi(y; X_1, Y_1)$

但是事实上，有些轨迹差的太远了，为了让ground-truth的cost明显小于这种轨迹，我们需要引入一个margin：

$C_\psi(Y_2; X_1, Y_1) \leq min_{Y \in SY}(C_\psi(y; X_1, Y_1) - l(Y, Y_2))$

其中l指的是两个轨迹间的距离。

以此，我们可以定义处planning的loss：

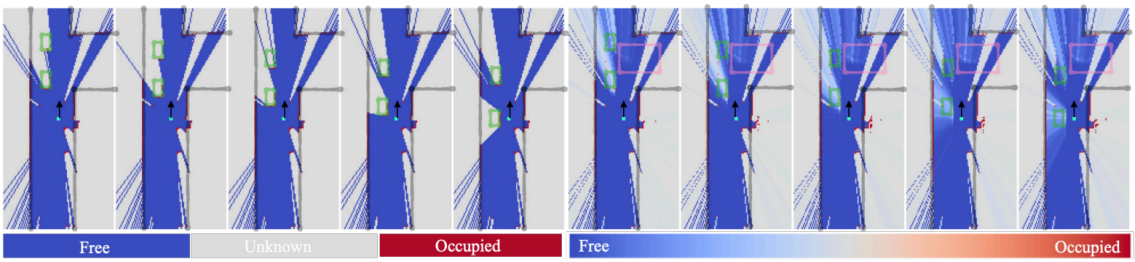
$loss(\psi) = max(0, C_\psi(Y_2; X_1, Y_1) - min_{Y \in SY}(C_\psi(Y; X_1, Y_1) - l(Y, Y_2)))$

其中具体地：

$l(Y, Y_2) = Dist(Y, Y_2) + \gamma \sum_{u \in Y} max(0, ray(u; X_2, Y_2))$

Effect

Predictor

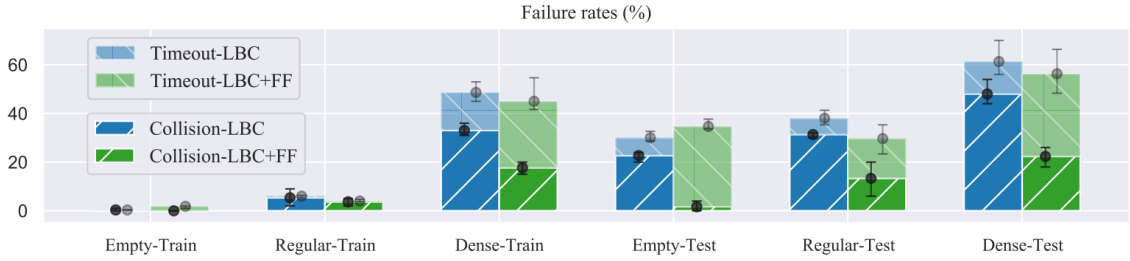


(a) Ground-truth future freespace　　(b) Predicted future freespace

Off-the-shelf Planner with predictor

Task	Weather	Success rate (%)			Jerk (m/s^3)		Route completion(%)	
		PV [6]	LBC [6]	LBC+FF	LBC [6]	LBC+FF	LBC [6]	LBC+FF
Empty	Train	100 ± 0	**100** ± 1	99 ± 1	**6.9** ± 0.0	**6.9** ± 0.1	**100** ± 0	99 ± 0
Regular	Train	95 ± 1	94 ± 3	**96** ± 2	7.8 ± 0.1	**7.5** ± 0.2	97 ± 2	**99** ± 1
Dense	Train	46 ± 8	51 ± 3	**57** ± 4	10.3 ± 0.3	**8.1** ± 0.2	79 ± 2	**80** ± 2
Empty	Test	100 ± 0	**70** ± 4	66 ± 3	**7.3** ± 0.1	7.5 ± 0.2	**86** ± 3	83 ± 2
Regular	Test	93 ± 1	62 ± 2	**73** ± 1	**8.4** ± 0.6	9.5 ± 0.7	82 ± 2	**87** ± 2
Dense	Test	45 ± 6	39 ± 6	**44** ± 5	**11.0** ± 0.8	12.0 ± 0.8	67 ± 3	**71** ± 4

在常规任务和密集任务中一定程度上提升了这些planner的决策质量。在testing中jerk偏高的原因可能是紧急情况下采用紧急制动的原因，或许换成缓慢制动会使结果更好。



Planning

Learn to plan	L2 (m)			Collision (%)		
	1s	2s	3s	1s	2s	3s
vanilla	**0.50**	**1.25**	**2.80**	0.68	0.98	2.76
+ object	0.61	1.44	3.18	**0.66**	0.90	2.34
+ object*	0.61	1.40	3.16	0.71	**0.81**	**1.45**
+ freespace	0.57	1.28	2.94	**0.66**	0.87	2.17
+ freespace*	0.59	1.35	3.07	0.74	0.93	1.65