# EMplanner

为什么要进行轨迹规划？

因为为了加快运行速度，规划算法的周期和控制算法的周期并不同步，如果不进行轨迹规划，向控制层输出一系列带绝对时间的轨迹，那么就只能向控制层输出一个点，但是规划是100ms一次，控制是10ms一次，这样一来，控制在100ms以内都跟踪的是同一个点，这样的控制效果显然是不好的，因此要进行轨迹规划。

轨迹规划的起点怎么确定？

轨迹规划涉及到轨迹的拼接问题以及判断跟踪控制效果问题，首先是要确定规划起点的时间问题，规划起点的时间是当前时间加上100ms，设当前时间为current\_time，那么规划起点时间就是curren\_tiime + 0.1，为何？因为轨迹规划算法规划出来一条轨迹然后发送给控制层这个“动作”也是需要消耗时间的，如果我们就是从current\_time 开始进行轨迹规划，那么在current\_time + 0.1时刻传给控制层的时候，轨迹是断开的



显然从current\_time开始进行规划到current\_time+0.1规划结束后发给控制的话那么上一帧的规划轨迹与控制层接受的当前帧的控制轨迹是断开的，对于控制来说如果每一帧接受的轨迹都是割裂的不连续的、不平滑的，那么控制效果不会很好，甚至会导致车辆抖动。

控制接收的轨迹如下：



如果从current\_time+0.1为起点进行轨迹规划，那么效果如下：



控制接收的轨迹如下：



显然上一周期的规划结果与这一周期的规划是衔接的，从控制来看这些轨迹是连续的，会有比较好的控制效果。

在实际编写代码中不仅仅是进行了轨迹拼接，还将上一周期在current\_time之前的20个点与从plan\_start开始的规划轨迹结合在了一起，这是因为我们认为规划算法需要100ms才能将结果传入控制层，但是这个100ms不是精确的，而是大概的，计算机运行过程中不可能每一次都保证刚好100ms，假设在current\_time+0.1时刻提前了几ms发送结果，那么会出现几ms没有轨迹跟踪，因此需要适量的拼接上一周期在current\_time+0.1之前的点。可能有人会有一个误区，觉得我current\_time才发送，我还往前面拼接已经走过的点？是不是多次一举了，但是事实上我们在current\_time+0.1进行规划时，实际运行时间是current\_time ，他们之间的100ms还没走。

轨迹规划起点确定代码：