- deep sort
  - Tracker
    - 实例变量
      - metric：NearestNeighborDistanceMetric类实例
        - 实例变量
          - _metric:距离计算方法
          - matching_threshold:匹配阈值
          - budget：samples缓存大小
          - samples{}：字典形式缓存每个轨迹的ID，历史特征数据
        - 方法
          - partical_fit: 创建新的目标轨迹缓存空间，更新轨迹缓存数据，筛选当前存在的目标轨迹缓存数据
          - distance:计算检测与目标轨迹的匹配代价
      - kf:KalmanFilter类实例
        - 实例变量
          - _motion_mat：状态转移矩阵
          - _update_mat：观测矩阵
        - 方法
          - initiate：卡尔曼滤波器初始化函数，创建状态矩阵，协方差矩阵
          - predict: 预测位置， 更新状态矩阵，协方差矩阵
          - project: 状态分布到测量空间
          - update： 矫正卡尔曼状态矩阵，协方差矩阵
          - gating_distance: 计算预测与观测误差
      - max_age
      - n_init
      - max_iou_distance
      - _next_id
      - tracks[]: Track类实例的集合，追踪器
        - 实例变量
          - mean:卡尔曼状态矩阵
          - covariance：卡尔曼协方差矩阵
          - track_id：轨迹ID
          - hits:追踪帧数
          - age:总的帧数
          - time_since_update:丢失帧数
          - state:状态标识
          - features[]: 特征数据缓存
          - _n_init：状态标识阈值
          - _max_age：最大可丢失帧数
        - 方法
          - pedict：调用kf.predict(), 更新age,time_since_update
          - update：调用kf.update(), 添加特征数据，更新hits，time_since_update， 状态标识
          - mark_missed()等，状态标记，查询函数
    - 方法
      - predict：执行每个追踪器的 predict()
      - update：将新的检测与历史轨迹匹配_match(), 根据匹配结果使用匹配检测更新对应的追踪器数据Track.update(), 更新丢失追踪器状态mark_missed()，_initiate_track()为新的检测创建新的追踪器。取出每一个当前帧存在的追踪器的特征数据并清空Track.features,对应的轨迹ID，调用metric.partial_fit(),更新samples{}中每一个轨迹的历史特征数据。
      - _match：将新的检测与历史轨迹匹配。1，首先调用matching_cascade()将新的检测数据与状态标识为Confirmed的轨迹匹配，在匹配过程中根据追踪器的time_since_update数值从小到大分批匹配，每一批使用metric.distance()计算的特征距离代价矩阵，然后调用gate_cost_matrix(),根据卡尔曼运动修正代价矩阵，再使用匈牙利匹配算法匹配，并根据metric.matching_threshold决定是否可以形成匹配。2，将状态标识不是Confirmed的轨迹与步骤1中未能匹配到新的检测且time_since_update==1的轨迹使用IOU匹配代价与剩余的检测进行匹配。3，将前两步剩余的检测与追踪器判定为未能匹配的追踪器与检测。
      - _initiate_track：对新的检测kf.initiate初始化卡尔曼参数，Track类实例化创建追踪器