1. 摘要
   1. VR项目，需要做到带上pro眼镜后双目鱼眼相机识别到手并且生成相机坐标系下的3d keypoints坐标，用球棍模型构造出一个手的模型（双手）。并且要求根据手势动作分类完成场景交互。场景是由UE/WebGL 构建的。常见交互动作包括“pinch”捏合，“ok”确认，“zero”表示取消等以及一些复合手势“[explode](javascript:;)”，“flip”等。
   2. 后来需要支持，NVIDIA jetson 系列设备，进行了一系列轻量优化。（系统流程，网络轻量化等），以及Linux系统的适配。
   3. Slam 6Dof 功能的优化与合并。
2. Pipline
   1. 2 fisheye frames -> yolo detector -> 0~2 bbox 手势frames。
   2. 手势bbox frames -> Knet -> n(bbox num) x [21 \* (2,1) , 7 ] 每只手输出21个2d keypoints，以及gestrue；
   3. 将每只手21个2d keypoints 作为g2o优化的edges观测值优化的对象是手的20个angles， 以及手腕节点position rotation。
   4. Kalman filter smooth Hand angles。
   5. 根据 手腕RP信息 + 20个angles + 手部每段骨骼长度 生成手3d keypoints。
   6. 3d keypoints -> UE,WEBGL (球棍模型/手模)。
3. HandTracking （torch deeplearning）
   1. Dataset interHand数据集（70w+，{2d keppoints，gestrue，bbox，distance，3d keypoints}）。
   2. Yolov5 ：
      1. Yolov5 结构， input（1x640x480）->output（nx(4(bbox)+1(obj)+2(cls))）

->nms ->(0~2) bbox。

* + 1. Loss : BCEcks,BCEobj, Giouloss
    2. Metrics : ap：98.6%
  1. Knet ：
     1. 倒残差block堆叠 input（1x96x96）-> output(21x18x18)heartmap-> Regression ->(21,1,2) 21keypoints,(1x2)2d points
     2. Loss : keypoints MSE loss + classifer focal loss
     3. Metrics :mpe 每只手21keypoints 点平均欧氏距离差。
        1. Group by gestrue。每种手势mpe做出统计。
        2. Total MPE：2.1（px）low “five” 1.3（px）~ height “grab” 2.7(px)
        3. Class acc 92.7%

1. Cpp项目系统模块
   1. Trt ：class Engine\_runner 基于CUDA。（读取模型文件，create GPU buffers on device，Inference）
   2. Detecor tensorRT load trt engine， img -> result(bbox,handness,21keypoints)
   3. Skeleton：g2o 优化双手重投影误差，优化参数。
      1. Edges:
         1. EdgesHand Transfrom2（frame）\*1(wrist)+5（finger）\*2(frames)\*1（angle）。
         2. edgeFinger Joint 2（frame）\* (5(fingers)\*3(keypoints))。
         3. EdgeConstraint 角度约束(20 angles) 。
         4. Smooth 前后帧（20 angles）。
      2. Vertexs（1(hand\_node) + 20(angles)）。
   4. Smooth：kalmanfilter 对handangles进行smooth。
   5. Utils：存放数据结构与一些运算方法。
   6. Camera 鱼眼相机（基于dshow的videoCaptrue），获取相机信息（pid，calibration，get\_sync\_frames，status等信息）
2. 附带项目：
   1. 双目鱼眼相机标定（基于opencv 张氏标定法）。
3. 后续优化/工作
   1. 功能扩展：
      1. 深度学习：hand gestrues （Knet基础上增加了一个branch来进行手势分类）目的是在VR场景用手势触发事件。
         1. 使用hand angles 数据设计复合手势分类。
      2. 硬件辅助：基于dshow 编写videoCaptrue模块，效果（起因）：
         1. 根据vr眼镜里fisheye pid准确获取数据（opencv 不同设备每次根据id来获取有时是不同的）。
         2. 获取图像间隔大幅提升20+ms ->14ms 同时也带来流程的帧数提升，smooth的准确度也在提升（时间差缩小）。（体验效果大幅提升帧数在55-60fps）。
         3. 视频同步问题，拍摄秒表左右相机曝光时间差距（0-5ms）opencv的videoCaptrue不能做到。
   2. Loss优化
      1. 深度学习:
         1. 周期循环学习率。
         2. 分类loss。
            1. 优化分类头。
            2. 人为降低分类难度。
            3. loss权重调试 + focalloss。
      2. 系统流程：
         1. 基于每一帧的手势重投影误差来修改kalman filter 测量值协方差矩阵从而修改卡尔曼增益。简单来说当重投影误差较大时g2o优化的结果会强行将手势3d关节点扭成 2d图下那个样子。表现形式就是手部会扭曲。这时需要让最终生成出来的手势尽可能贴合预测出来的手势而不是测量出的2d点。
         2. Scale 优化：
            1. 每个人手掌大小不同，手掌过大过小都会极大影响手势效果。
            2. 程序正常运行时，收集50张，gestrue是“five”（手掌伸展，便于优化）的图片以及Hand信息（3dkeypoint，angles）
            3. 分布优化g2o setlevel 先优化手掌，再优化所有指头。
            4. G2o参数

Edges（21（keypoints）\* 2（camera）\*50（frames））

Vertexs（20（angles）\* 50（frames）+1（scale））

* 1. 轻量优化
     1. 深度学习：yolov5n，float8。
     2. 系统：隔帧检测（yolo time 75%->45%）。
  2. 工程化
     1. Jetson nano 移植。
     2. Api编写接口。
     3. 文档与so/dll。
     4. 异常状态划分与反馈log。