Z-AI1.4 版行为序列化建模

文档版本**:1.0**

最后修改时间**:2024-3-5**

修改日志

目录

什么是行为序列化

行为序列是把乱序数据按时间进行采集,行为数据是指识别结果,gpu 跑识别总是使用线程,当 gpu 按 123 顺序识别图片,输出识别结果顺序并不会是 123,这会是乱序,当流程按时间采集完成后进行排序,这时候就得到了行为序列.

当一张图片被识别后,图片中会有场景识别结果(许多场景分类结果值+标签),检测器识别结果(许多框),而框里面会有许多分类标签(许多目标分类结果值+标签),这一系列的数据构会成一种空间.

当多张图被识别,会形成多个空间,这时候再从整体来看,会是一种多维的时空数据,这些数据将成为统计程序展开拳脚的条件,因为统计思路总是多个维度相交的,但行为识别的统计并不是简单的统计方法.

饶了一圈回到正题,行为序列化就是把识别结果序列化,再用复杂的统计方法计算出结果,这就是行为识别,如果严谨的说,这是序列化行为.

在序列化行为中,行为的时空机制会按:**抬手->姿势运动->落手**,来运作.这并不是单指人类,而是针对目标行为的过程,在过程启动时,就是抬手,行为过程会有一系列动作,这一系列动作就是姿势运动,当行为处于非抬手和姿势运动时,就是落手,落手也可以归类成为无形为,无行为也就是无姿势.

当梳理出行为的运行规律,被技术面实现后的序列化行为时空机制会变成:**姿势运动->落手**,抬手机制被省略掉了,只要目标处于行为运动中,只会有处于可以被识别的姿势运动+无姿势运动两种状态.例如行人行走是无姿势,行走中举手向上是姿势 1,双手与肩膀平行张开是姿势 2.再例如,仍出一枚即将爆炸的炮竹,炮竹在地上滚动是无姿势,炮竹开始燃烧是姿势 1,炮竹燃尽是姿势 2.最后再以复杂的跳舞为例,跳舞会有很多很多的姿势分类,跳舞一系列的姿势变化都是姿势运动的过程,这里可以单独抽出许多姿势,例如只想识别抬左脚和下蹲,那么除了抬左脚和下蹲都以落手行为判定也就行了.

说到这里,序列化行为的建模思路也就清晰起来了:**从视频中采集出目标的姿势运动行为动作,处理它(细节省略,后面来细说),然后再采集出目标的落手行为动作.整个序列化的行为建模都会按照这种思路进行开展**.

聊个题外话题,国内市场有许多安防领域的 AI 盒子可以检测人流通行,它的实现机理是画一条线或则拉个框,当目标框(被检测到的人)与线条或则框发生重叠,那么就让程序启动计数+1,当没有相交或则重叠,就重置计数器.另外一方面,这些 AI 盒子还接入了大量目标检测器模型,例如垃圾,杂物,车辆,地摊,路面坑洼.国内的 AI 盒子圈几乎全都是堆检测器,然后用 RunTime 来解决行为识别,例如景区跳崖自杀,RunTime 只要检测到人在危险区域直接 Post 消息给 web,或则记录一条数据+图片,让三方程序进来查找.国内的 AI 盒子明显不是走的行为序列化路线,但国内的 AI 盒子销量很好,应用的项目也很多,主要是安防圈并没有特别多序列化行为识别需求,用的比较多的大都是区域计数,人流量计数,目标出现报警,做这些功能难点只有建模检测器,业务 RunTime 部分并不难,大多会是计算线框相交,记录数据,post web,UI 界面这些环节,目前这些 AI 盒子正在互相抄袭,也许未来会进行一波价格战,也许不久以后会出现 500 元的 AI 盒子.战胜国产盒子的策略可以走体系化路线:体系可以复制局部,但局部不能复制体系,剩下的交给时间,这对国产盒子是一种来自技术面的降维打击.

行为序列化的工作条件

稳定 10-60fps 视频采集速率的监控或定点拍摄,如果场景有高速运动目标,就 60fps,普通运动目标,例如步行+慢跑,就 25fps,如果目标大都处于缓慢运动或常处于静止,就 10fps.

需要保证检测器在每帧都可以找到被执行序列化行为的目标,例如要做人类行为检测,那么,检测器每帧都必须做到找到人类.检测器建模环节应仔细阅读检测器文档,因为检测器一旦漏掉目标,后面的行为序列化识别将无法工作.

在早期的监控项目中,Z-AI 使用 correlation tracker(锁定跟踪技术体系)来逐帧跟随目标,当监控技术发展到第六代,correlation tracker 体系被直接砍掉了,这是因为 correlation tracker 体系走的 fft 迭代计算路线,在单台服务器负担 50 路以上监控时,大量的 cpu 算力会资源被 tracker 环节消耗殆尽.这时候,以 YOLO-X+DNN-OD 这类对象检测器来代替 correlation tracker 体系,cpu 算力开销被 gpu 负担,性能问题迎刃而解.

需要完成目标分类器模型,这也是本文档主要描述的建模技术.行为序列化是针对数据的一种统计工程计算,当检测器找到目标,然后目标分类器识别出目标行为,形成多维时空数据,这时候行为序列化的计算才会派上用场.

在六代监控体系中,gpu 服务器只会负责对每帧做场景识别,用检测器找目标,再用目标分类器识别目标,最后把识别结果发送给数据中心.gpu 服务器并不会对识别结果做逻辑性质的处理,gpu 服务器的工作非常明确:尽可能多的识别监控内容,并且依赖于 gpu 算力优势实时重建监控视频(这一步就是做 nvr 的事,nvr 存在数据卡脖子问题).

行为序列化并不是工作于 gpu 服务器端(gpu 的 AI 识别体系已经很复杂了,不可以无限复杂下去).

行为序列化也不是工作于数据中心端(因为落地项目是 360 路高清监控,数据中心日流量已超过 1pb,尖端路线谢绝三方开发商自行集成) .

行为序列化是以一种独立的计算程序模块,它被数据中心的周边应用服务器使用,也就是,需要单独开个服务器来跑行为序列化计算,这类服务模型有 2 种工作模式

订阅模式:从数据中心订阅某一种监控,这时候 gpu 服务器所产生的识别结果,会被实时转发进来.这就为行为序列化识别提供了实时工作的数据条件了.例如目标在 9 分 21 秒做出了摸头动作 ,摸头动作的物理时序长度如果是 1 秒,也就是从抬头然后手掌移动到头部的过程是 1 秒,那么在 9 分 23 秒就会出现行为序列化识别结果,后面的流程走短信,post web,语音播报,记录到数据库,这些可以任意.在前一章节描述的替代国产 AI 盒子的工作,也是在这一步来干,当 AI 完成了识别,把识别结果实时的给出来了,AI 高技术门槛会瞬间变成开发商了的技术资源,在此基础上要做点机关单位,学校,工厂的智慧系统,都会是开发商的菜,并且,对手只会用 AI 盒子弄点简单的通用识别,可预见技术性的降维打击将会覆盖竞标,市场,投资这些环节.具体细节,大家可以等我编写完六代监控的文档.

查询模式:使用定期,定时或则 UI 化,从数据中心拉去某个时段的监控识别内容,然后,进行查询分析.细节在这里就不多说了,直接去参考六代监控的使用文档.

回到主题,行为序列化是基于大数据的统计学工程,因为需要数据,而数据都是重量级的程序在产生,因此很难被做成独立小巧的方案,包括统计学的可行性验证,算法调试,这些都是建立在六代监控体系上.大家平时所看到行为序列的识别 demo,内部走是在监控产生的数据基础上做出来的,没有 AI 识别结果,计算方法再精妙也没用.

选择序列化行为样本

就地取材:**当监控摄像头安装在有人流的地方,并且目标人物经常会做出一些易于捕获的特定动作,例如闲庭信步(双手插兜,背着手小步悠悠走,拎包悠悠走),打电话(原地打转,慢悠悠边走边打),抽烟, 扫地,奔跑,正朝监控挥手求救,抬手在操作某些机器,这些行为都会出现明显的肢体动作,凡是有多种肢体动作,建模层面就都是走序列化行为**.但是需要注意视角与目标匹配,监控通常会自上而下呈斜角,人在不同位置所识别出的尺度会是不同的,偏远处距离人会呈现 1:3 的尺度,偏近距离会是 1:2,人也会有站立的角度差异,可以正朝向监控,也可以背朝向,或则侧朝向.只要确保这些不同站位和朝向的人会经常性重复动作就可以是样本,直接采集出来.

通常来说,在穿有制服的机关单位,建模检测器和跑序列化行为识别,是比较容易做到的.穿衣风格如果偏日常便装,会要求检测器建模环节有丰富样本.

指令型肢体动作:行为序列化并不是一定要针对目标监视,可以正对监控头发出某些指令,例如打开手机中的手电筒,然后左右挥手求救.摆出某些姿势识别以后,进入分支处理流程,例如挥一下手,识别以后播放一段音频.

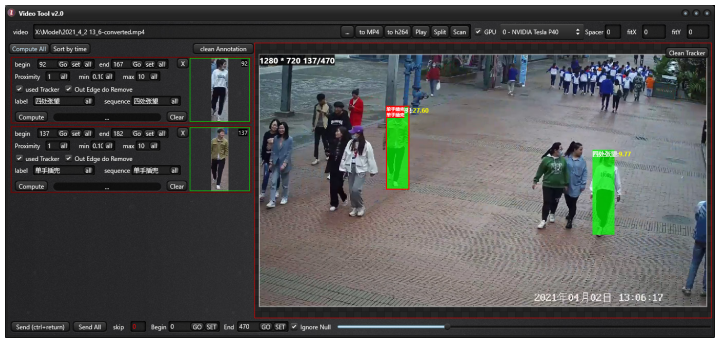
在实际监控项目中,场景状态需求会远远多于序列帧行为,例如闸口,走廊的人流量统计,目标场景中的活跃人数统计,性别统计,穿衣风格统计.做这些需求的建模只需要关心检测器和目标分类器就可以了,不需要折腾序列化行为.

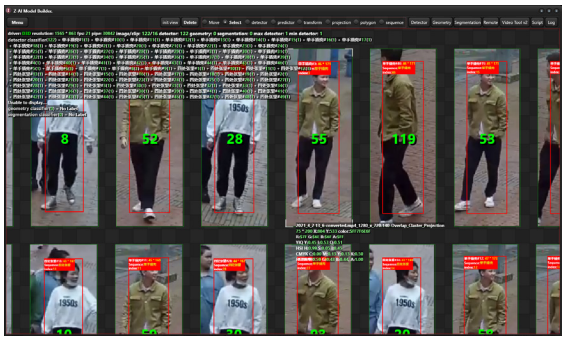
许多行为,用单帧就能识别出来,完全无需序列化识别技术,例如摔倒,躺在地上,教室上课时,学生们是否在听讲,写作,阅读,这些行为直接使用单帧的目标分类器完全够用.**单帧行为在建模层面也会走序列化行为采集路线,例如一个单手插兜行走的小伙子,会有许多状态,时而停下脚本,时而闲庭漫步,时而东瞧西看**.

在未来 sora 被开放使用权以后,也许这会是一种不错的序列化行为样本的再生成工具,**sora 空间系统内置了 frustum 结构**,可以把手机拍摄视角转换成为监控的 perspective 视角,希望能盘活抖音数据->监控视角.

当下 Stable Cascade 可以作为局部数据生成工具使用,例如举手动作从抬手到举手完毕,中间会有 10 帧是做出举手的动作过程,这 10 帧可以被 Stable Cascade 识别出来并且生成相似样本素材.可以丰富一下样本库.因为目标分类器不像检测那样对场景有所要求,目标分类器是走的相似性小图识别.提示一下,监控系统的内容大都是私有数据,偏机关单位,公共设施,**大模型处理私有化数据的能力很弱, StableAI 的大模型几乎没有学习过监控**.能有微辐度工作效率提升,不会有质变.

下图以采集走路姿势为主,分别为四处张望和单手插兜,在业务 runtime 层面这并不是序列化行为,在建模层面是走的序列化.





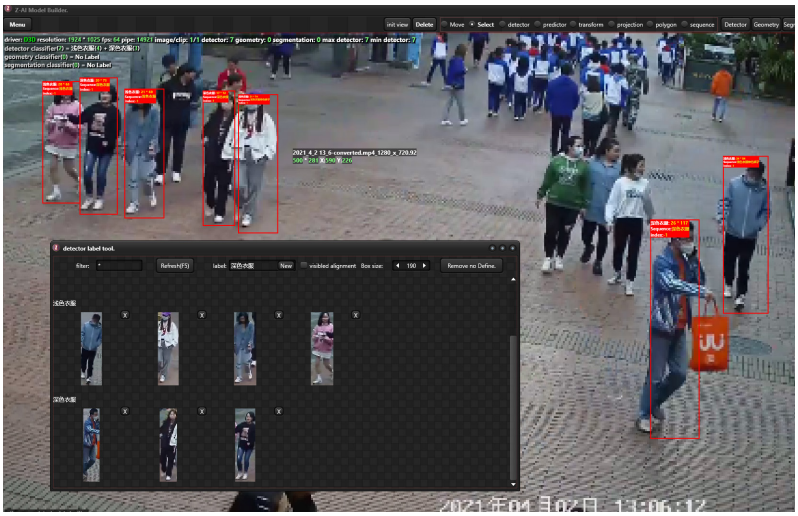
序列化行为建模在底层是生成目标分类器数据

序列化行为是在目标分类器标签加上序列后缀,另外,序列化行为建模是借助工具从视频采集连续性的运动姿势,而纯粹的目标分类器是以靠人工拉框一个一个来标注,这样的人工标注容易出错,并且规模都会有限.

行为序列化与传统的目标分类器

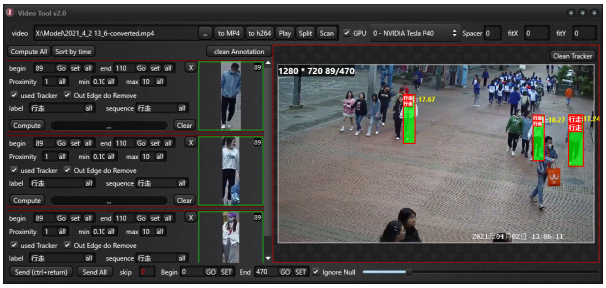
首先,一定一定,要区分目标分类器和检测器,它们,都会拉框,但这是完全不同的建模思路.

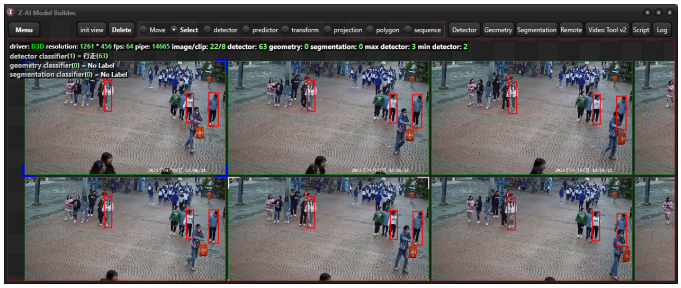
检测器需要能框尽框,并且要求场景,目标分类器只会计算框框,因此,目标分类器只需要框住,打上标签,那么样本就会可用,即使在几十人的场景只框一个人也是可以的.传统的目标分类器建模会先框目标,然后打标签,如下图.



行为序列化建模是以视频为样本,通过 tracker 技术跟踪目标,让框框形成时间维度,最后生成传统的目标分类器样本.例如框住 A 走路,这时候开启 tracker,那么 A 从迈出左脚到右脚,并且从监控远处到近处会进行序列化的样本采集+自动给它标注上框框+自动打上 label 标签.

左图为序列使用 tracker 的采集器,右图为是当 send all 以后被采集出的视频帧和标注信息,这时候样本已经出来了.





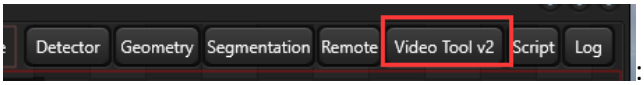
建模前需要规划行为的种类

行为建模可以是序列化的特殊行为,也可以是日常重复动作,在建模以前一定要做好规划.一般根据手上已有的视频资源数据种类进行规划.

再次重申关键:使用序列化行为功能的前提是检测器

行为识别的前提是检测到目标,如果检测不到目标,分类器是无法工作的.甚至连测试工作都无法开展!
Z-AI 的检测器有两套,分别是 DNN-OD 和 YOLO-X,检测器非常需要下功夫解决(检测器建模难度会偏高),一旦搞定检测器,行为识别和目标分类器会是平推的,按照文档指向去做就能出模型.

使用 Video Tool V2 工具抓取目标行为序列

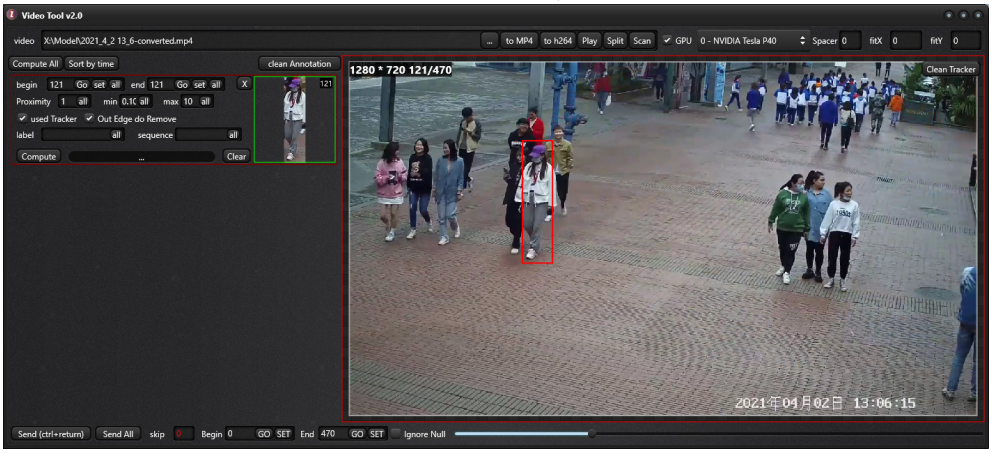


:通过标注工具打开 V2,这是一个视频分解工具,会把视频分解成连续性的图片,然后进行各种 tracker 操作,待操作完成,send 功能是把结果发送至标注工具.
序列化行为的样本必须是高帧率的,极好品质是 60 帧的物理视频,而不是通过外部工具把 25 帧强制转换成的 60 帧.最其次也需要 25 帧的视频,类似 10 帧的低帧率,几乎没有办法使用 tracker 跟踪运动目标的,这会导致 tracker 计算过出现相似度丢失.一般抖音和标准监控的视频资源大多为 20-25 帧,一定要注意帧率.

当准备好序列化行为的视频资源以后,就可以开始工作了,先指定文件,然后Scan, Spacer 0这里要给 0,采集不跳帧.

待采集完成后, 1280 * 720 121/470表示视频分辨率,当前帧和总帧数.

滚动条可以拉取快进倒退帧,每次操作以后当前帧都会变化



在视频画面中任意拉框,左边就会自动弹出标注工具

begin 121 Go set all end 121 Go set all :begin+end,表示围绕标注框的起始帧位置和结束帧位置,Go set all go 是当前帧直接跳转到 121 帧,set 是把当前帧赋值给输入条,all 是设置所有的输入栏,例如拉 3 个框,这时候 3 个框的 begin 都是 121,但是这时候我们希望从 150 帧开始,那么只需要修改其中一个,然后点 all,所有的 begin 都会变成 121.

Proximity 1 all :最低接近度,从 A->B 追踪的 tracker 值越大,就表示越正确,越小则代表 ab 之间的接近度更低.

min 0.1C all :在 tracker 工作中,框框的尺度会自动变化,0.1 表示当框框由大变小时,面积不能低于标注框 10%

max 10 all :当 tracker 框框尺度由小变大时,面积不能高于标注框的 10 倍,如果高于,compute 将会中断

✔ used Tracker :勾上会启用 correlation tracker,否则将会使用锚定标注,从开始到结束,框框都在同一个位置,这种功能多用于锁定拍摄方式,例如人在中间的全身特写,亦或是在视频处理工具锁定了目标靶心,例如目标总是处于正中央.

✔ Out Edge do Remove :勾上以后,当 tracker 的框框超出视频边缘将会移除,即使发生边缘重叠也会移除,只会移除框框.

label all :tracker 框框的标签,也就是目标分类器的标签,这个值通常可以不用填写

sequence all :tracker 的序列标签,这是必须填写的值,序列标签用于重构目标分类器标签.非常重要!

Compute :启动 tracker 计算,算法内部会发生两次计算循环,例如标注框在 140 帧,begin 帧在 120,end 帧在 150,那么启动 tracker 计算以后,会先计算 140-150,然后再计算 120-140.

红框和绿方框的含义

凡是出现红框都是标注框,例如在 121 帧拉了一个红框,那么当进步到 122 帧红框就会消失,只有回到 121 帧红框才会出现,绿框为 tracker 框代表已经被计算完成.

V2 工具的标注思路就是时间线,在时间线某个位置画一个框,然后给 begin+end 值,这时再延时间线计算出时空框.

左下图中,空心红框表示有框但还没有被计算,**在绿框中会有个数字,这就是 tracker 帧间框接近度**,值越大可信度越高,**凡是没有绿框,都是没有计算**,右下图中,箭头所指的图片会是红框抠图,点击这张图片,当前帧会立即跳转值红框的标注帧.**使用时需要区分标注帧,begin 帧,end 帧**.

Tracker 循环中可能会出现的问题

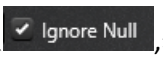
工具只是辅助,避免变成傻瓜,tracker 的作用是时空标注,这不是劳动解放,仍然需要不断的 tracker 和修正.

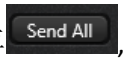
视频样本尽量使用物理高帧率,不可以转换工具 20 帧生成 60 帧,这是走像素化线性插值.

在人流涌动的视频样本中,tracker 目标很容易发生遮挡,因此 Tracker 长度不应该过大,建议 2-5 秒内小批量标注,否则 tracker 目标被遮挡,会导致框框变形.

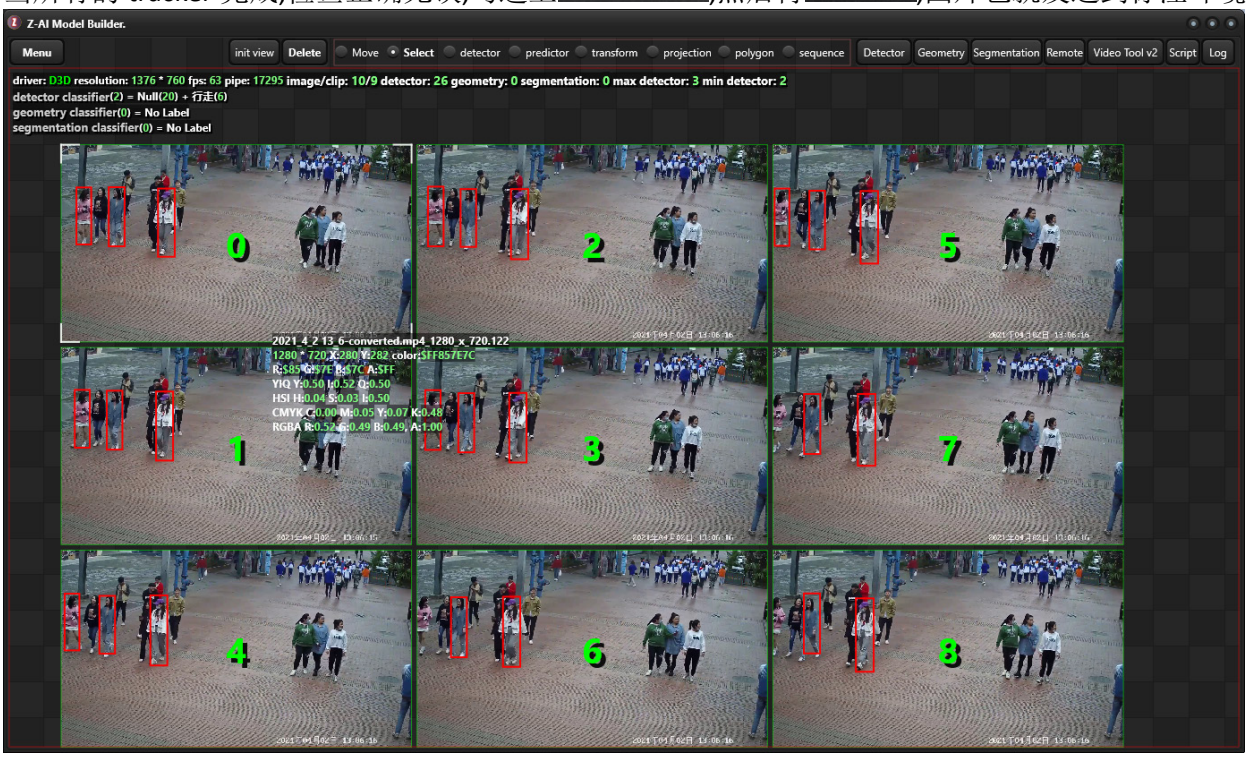
V2 工具会一次性把视频都解码到内存中,因此内存耗费很大,很容易动则上百 GB.后续章节会专门讲解视频剪裁.

如果 tracker 走坏,可以点 X 按钮删除,然后重新标注一下,再次 tracker.

当所有的 tracker 完成,检查正确无误,勾选上 

,然后再 

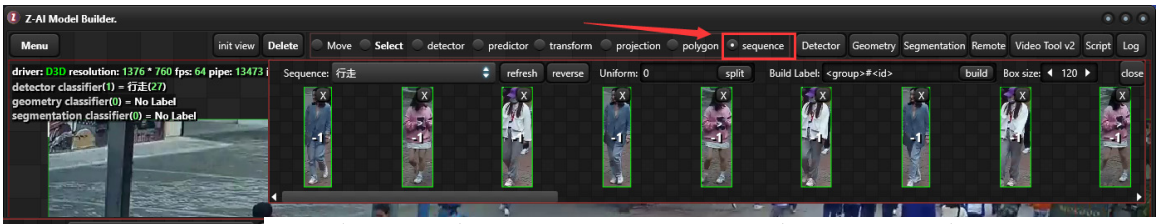
,图片也就发送到标注环境了,如下图



建议每次 Send All 前检查一下这几个地方

序列化行为分组

由于序列化行为是一种统计学工程,如果目标行为只有一种标签,会给统计代来数据匮乏的问题,因为统计不知道行为变化,例如行走有 10 帧样本,而分类器识别出来就是一个标签:行走,这时候,行走就应该分组,分组是让行走标签变成行走进度 1,行走进度 2,行走进度 3,当识别到同类行为,而行为在发生细微变化,就是体现在分组标签中.这样统计算法就会知道这是一个正在运动姿势的行为,而不是一个死板的目标分类标签.
使用分组前,必须先 tracker 出数据,并且在 tracker 数据中含有序列标签



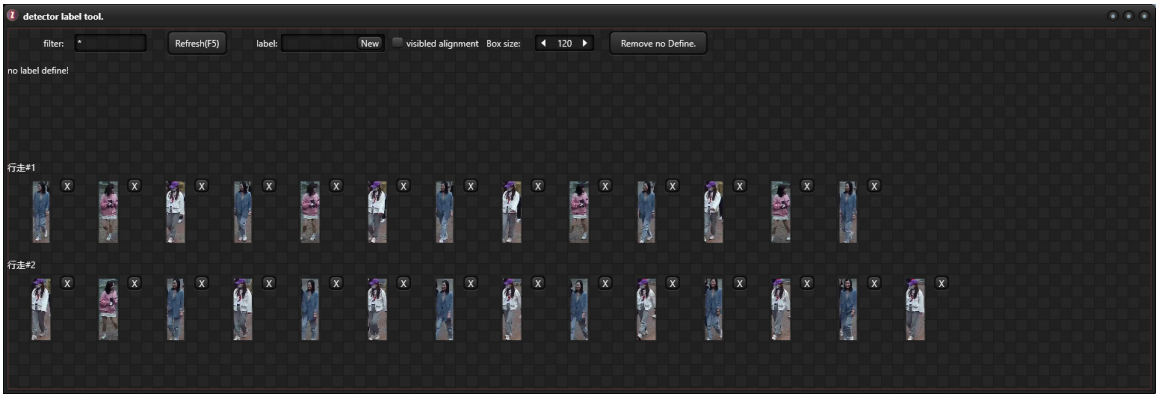
:这里会罗列出在之前 V2 的 tracker 工具中填写的序列标签.

:平均化切分成 2 个分组, split 按钮会立即对当前序列标签进行均化切分

:生成目标分类器标签,<group>是序列标签,#是分割符号,<id>是分组号

build :这里的 build 会把序列标签重建成目标分类器标签.

当重建完成后,在分类器标签工具会出现 2 个行走标签,行走#1,行走#2,在业务 runtime 框架中,会给流程返回”行走”,不会包含#符号的内容,#符号将作为运动变化姿势权重对待.在六代监控框架,标签分组符可自定义,例如@,&,$



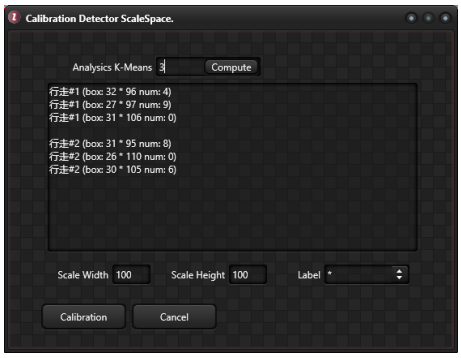
训练行为序列化行为之前需要了解尺度问题

这里的尺度问题就是,检测器的框框是一个尺度,序列化又是一个尺度,怎么来统一呢?其实,两者尺度只要是接近的都没问题!

如果不了解尺度概念,建议仔细阅读检测器建模文档关于框框尺度的章节.尺度是指宽高比例,而非宽高值.

标注体系使用了最小走样尺度机制,只要标注的尺度之间相接近,那么都是可以正常工作.一般按接近检测器尺度来标注序列行为就可以,不用管尺度问题,有差异它会自动适应.

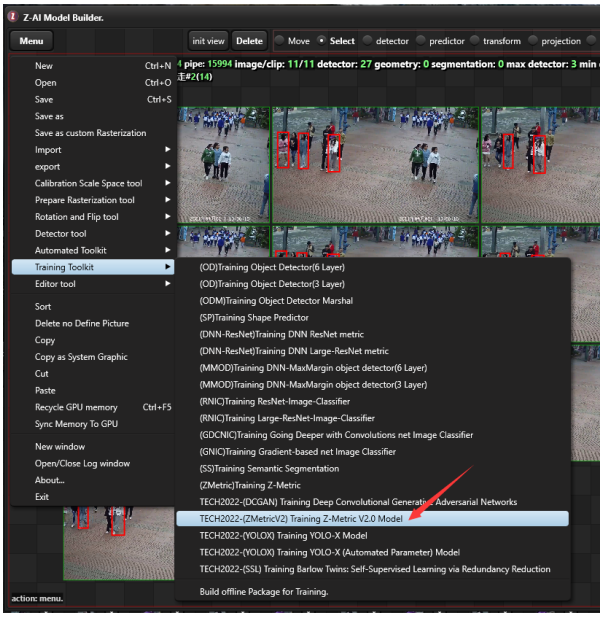
尺度有问题是指原本应该是 1:3,1:2 这类表示人类的尺度,突然变方或则变长了,这种尺度就是有问题的,可以直接把标注拉小,一眼就可以看哪里的尺度出现问题,如左下图,直接删除有问题的,第二种办法是通过尺度工具来检查尺度比问题.如右下图

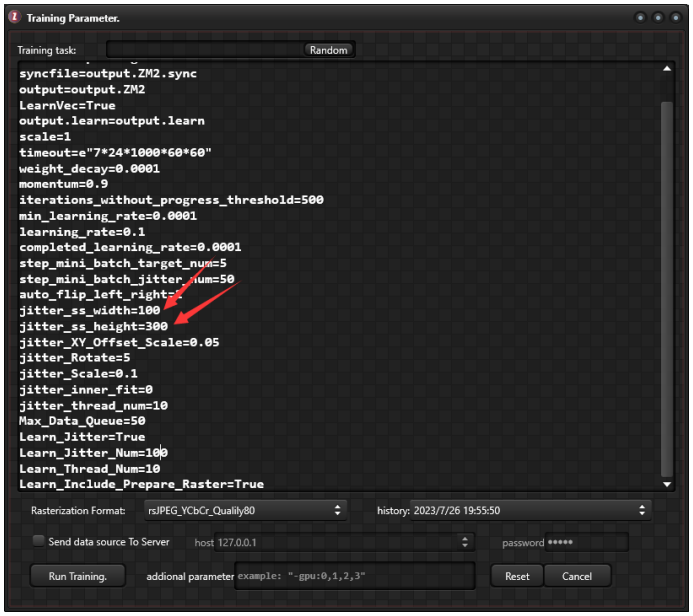


序列化行为主要使用 ZM2 模型体系(目标分类器)

关于 ZM2 模型体系在目标分类器建模文档中有非常详细的介绍,标注环节只要给出了序列化行为分组,RunTime 就会当成是行为识别来对待.

下图为启动 ZM2 的训练程序,在训练参数中,给出 1:3 的小图尺度



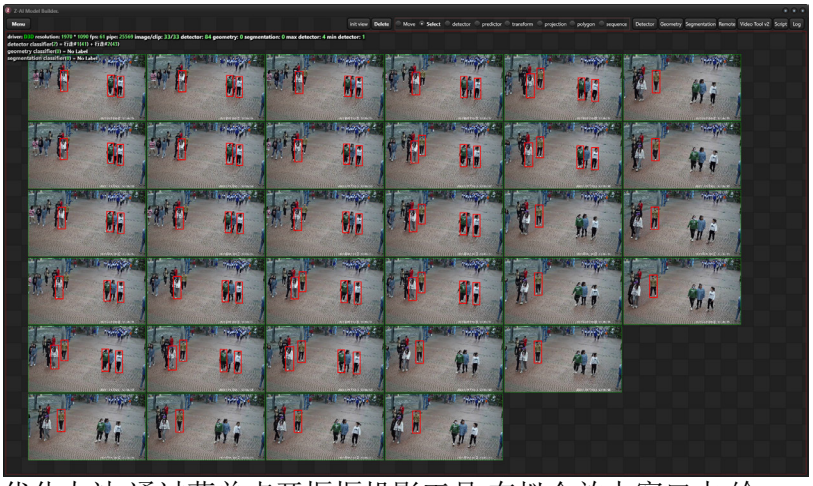


工程化的序列化行为建模

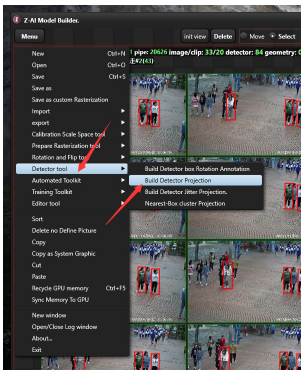
单独一个.AI_Set/.ImgDataset 样本库是无法堆大的,建议一个视频对应 1 个或 2 多个单样本库,通过导入到图矩工具来做训练整体.这样干会更方便维护修改,主要是工作模式会因此变成堆砌方式,从技术面来说,检测器不适合堆砌,而目标分类器完全可以走堆砌路线,即使一个行走行为,做 1000 个视频的序列化样本单库也是可以的.堆砌以时间推进不断的增加,1 天,2 天,3 天,这样下来,会拥有丰富的行走行为识别.

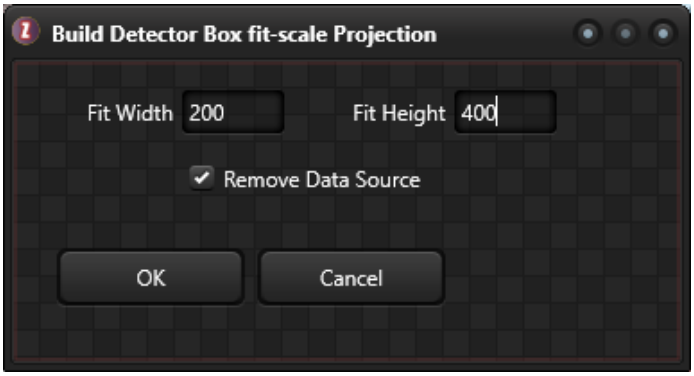
单库样本优化

即使一个视频的序列化样本也非常达到上百张图,样本库中都会是整图+标签,这些整图往往是 1080p/720p,既耗费内存,也不方便传播和拷贝,如下图,这时候,可以对样本做一些优化处理,让它看起来更美观,同时这在工程化建模中也能有效降低图矩的资源消耗.

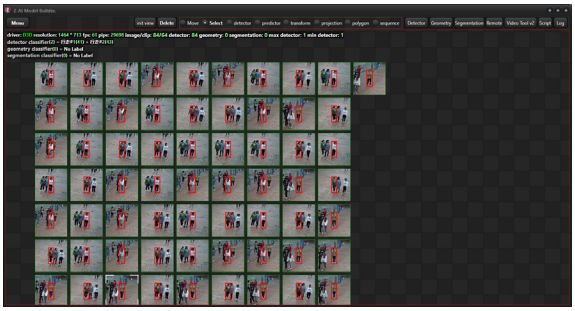


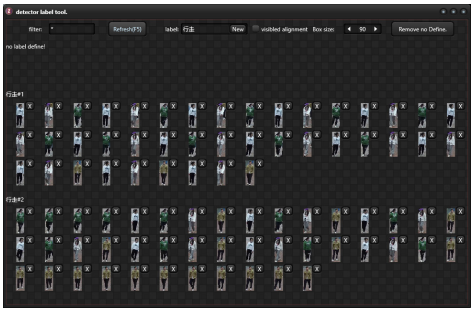
优化办法:通过菜单点开框框投影工具,在拟合放大窗口中,给 200+400 尺度比





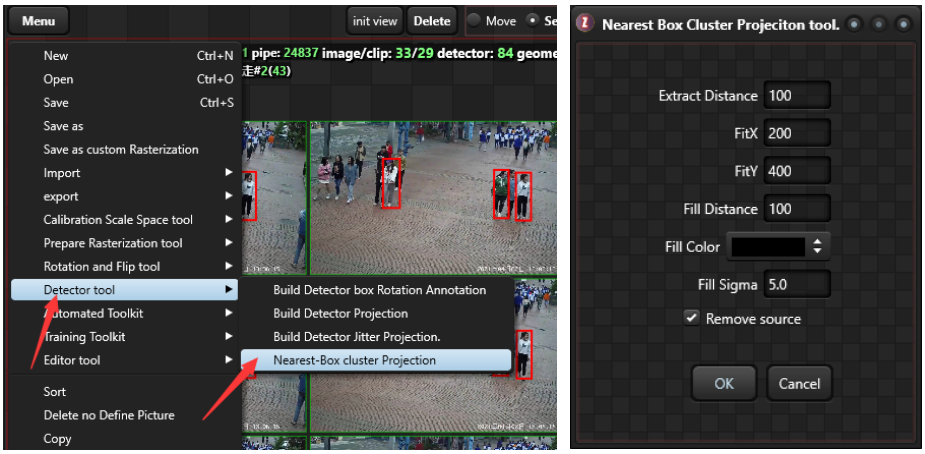
优化完成后目标分类器的标注数据不会改变,但样本库美观了很多,同时资源也更少了



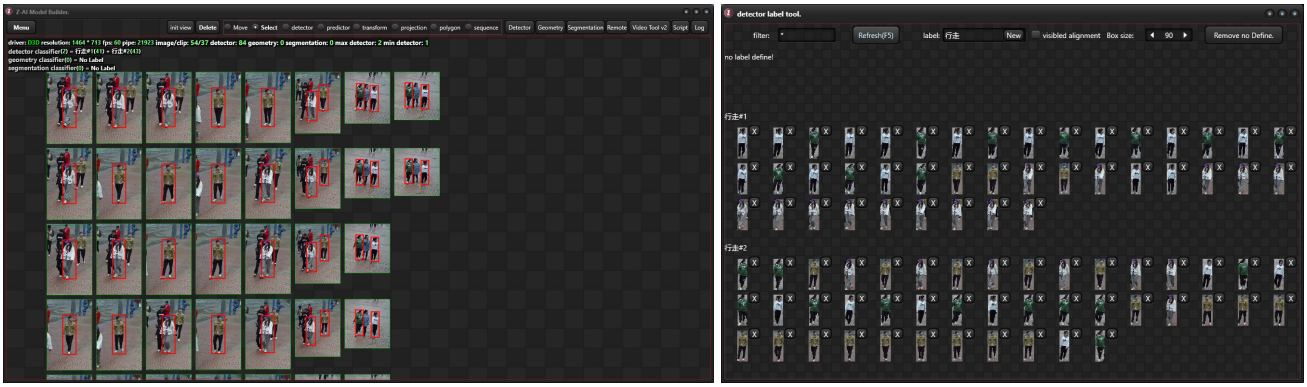


第二种样本优化手段,也是重构样本最优解

第二种优化办法是用群聚标注投影重新生成样本,通过菜单点出群聚投影工具,按右图参数投影



完成后会输出相比单图单框更为美观和节省资源的样本库,这是最优的方法,标注信息不会发生改变,这时候,直接导入图矩工具即可.

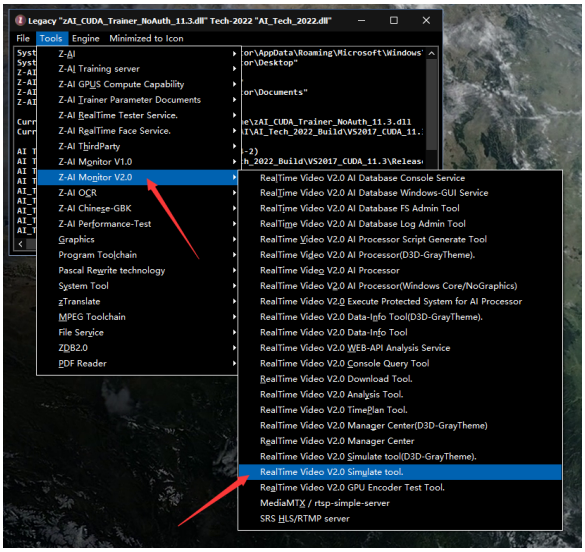


在图矩中汇集单样本库

直接把.AI_Set/.ImgDataset 导入到图矩中即可,序列化建模只要有样本,tracker 和标注这些操作还是很轻松的,一天下来随随便便搞完几百个人行为采集,这会是非常大的数据量,直接汇集到图矩,通过图矩来训练目标分类器模型.

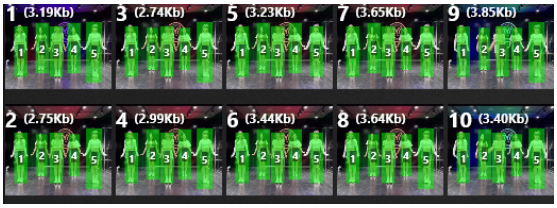
测试行为序列化识别结果

测试行为序列化识别使用 6 代监控的仿真工具来干,通过工具链菜单打开它

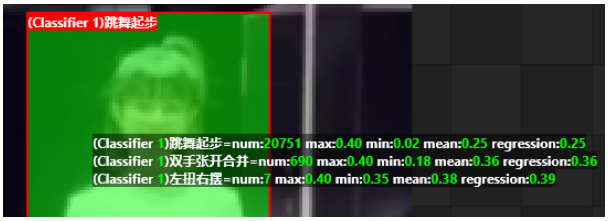


测试需要一个视频剪辑,一个检测器,以及目标分类器

下图为识别结果的视觉呈现,每个框框都有时空配对信息

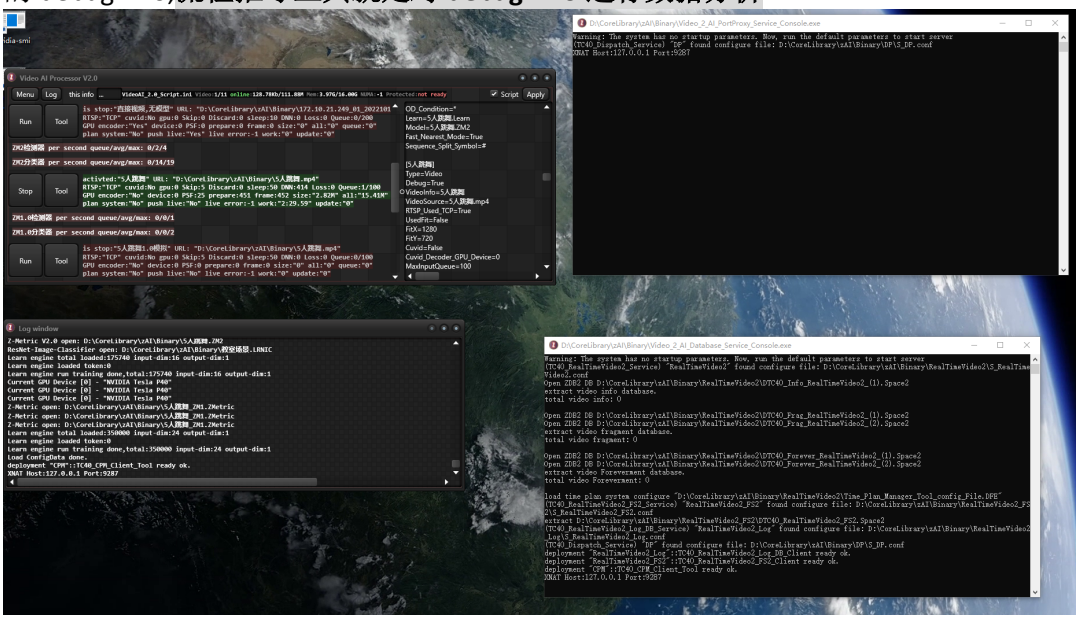


当鼠标移动至框框上,会出现目标分类器的候选信息,表示统计时的局部信息环节



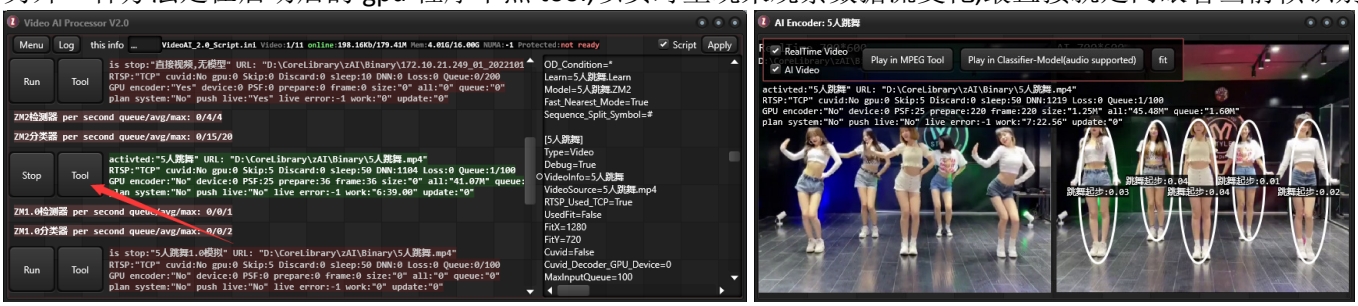
行为序列化流程推导工具

推导工具需要建立一个小型的 6 代监控环境,这里的脚本现在会如看天书,建议在阅读完 6 代监控体系的文档以后再来鼓捣这些东西,6 代监控分为 gpu 服务器,专门负责跑多线程识别,数据中心服务器,该服务器会包含行为序列化的 debug info,**流程推导工具就是对 debug info 进行数据分析**.



直接看 debug info 非常反人类,因此需要把数据以视觉方式呈现出来,把分割信息画出来,这里也就可以直接验证序列化行为的正确性了.**序列化行为算法面向数据,算法正确性是第一要素**,如果外面直接弄个 demo 和小工具来跑除了展示效果,几乎没有意义.

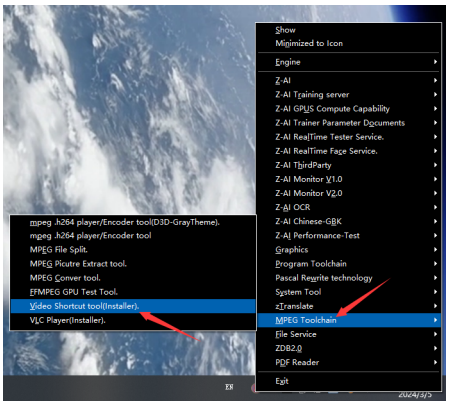
另外一种办法是在启动后的 gpu 程序中点 tool,以实时呈现来观察数据流变化,最直接就是肉眼看当前帧识别状态.



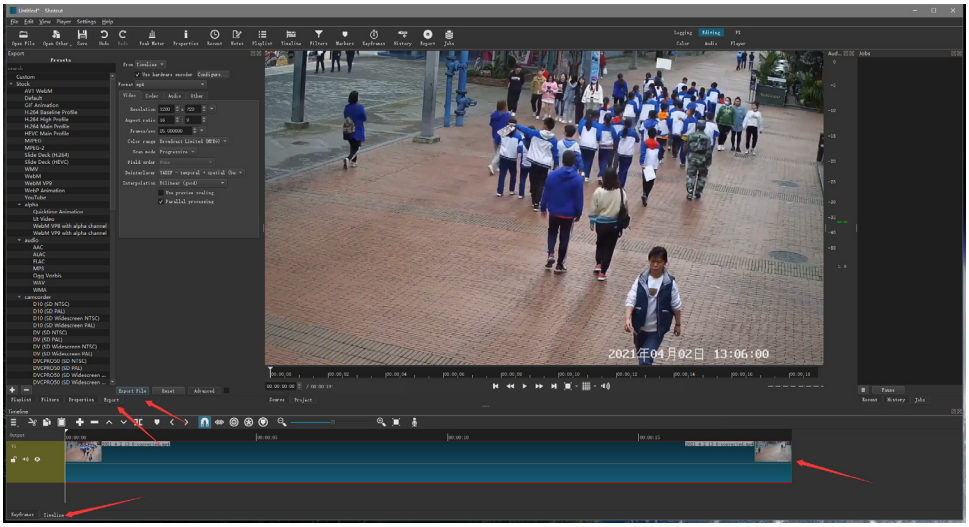
视频剪辑

从视频中提取样本时,以及测试序列化行为时,会遇到视频过大,或则是有行为数据的片段在视频的很后端.这样不仅仅是 V2 建模工具逐帧分解视频帧消耗内存,同时也非常浪费资源和时间,因此,就需要做视频剪辑.

在工具链菜单中,可以找到 shotcut 剪辑工具,这是一个小工具的安装程序.shotcut 是开源免费的,因此被直接集成在了 Z-AI 的视频工具体系中.



Shotcut 要直接切换英文界面,中文翻译很多我直接看不懂,把视频拖进来,切换时间线,拉时间线,重构导出,这时候,视频也就剪辑完成了,如下图.shotcut 也就是时间剪辑好用一点,其它功能像从视频里面区域性的抠图非常难用.具体细节,大家可以通过油管,抖音,去搜索一下关于 shotcut 的使用方法.



大家如果看不懂,尽可不必担心

在完成了文档编写工作后,会录制关于检测器,目把分类器,场景模型,序列化行为建模的相关操作视频.阅读文档+观看视频相结合,这会很容易理解繁琐的建模操作.

全文完.

By.qq600585

2024-3-5