ZAI 1.4 更新日志

文档版本:1.1
文档更新时间:2023-7-26

1.4 相较 1.3 时隔 3 年,更新范围包括,基础算法类,基础程序类,基础技术体系类,基础平台类,AI 业务类.如按更新强度计算,**1.4 可以等同于一个全新版本的 ZAI**.

目录

基础算法类

基础算法涵盖,新增与修正的 AI 内部算法,以及某些问题的解决方法

优化大规模样本训练检测器

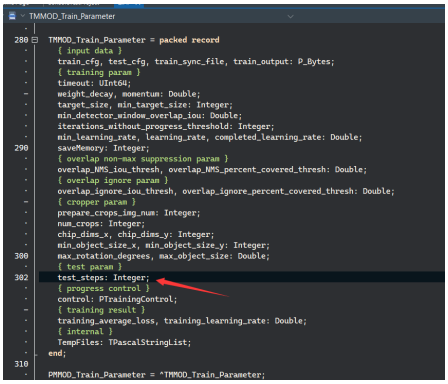
检测器是最常用的 AI 工具之一,当场景丰富性多了会出现大量标注,这些标注是否被算法有效使用取决于整个算法流程的设计.

当标注的目标数量>2000 个,而这些标注全都来自不同的场景.在计算 loss 过程,往往会进入近似无限循环,即使花上 2/3 天训练下来,模型也不会有很好收敛结果.因为底层在随机抽取样本进行输入迭代,加上某些人类难以发现的错误的标注,导致在漫长训练过程中无法收敛:**解决思路是对抗人类无法察觉的错误标注,给训练过程指一条明路,让它能达到最精确的结果**.

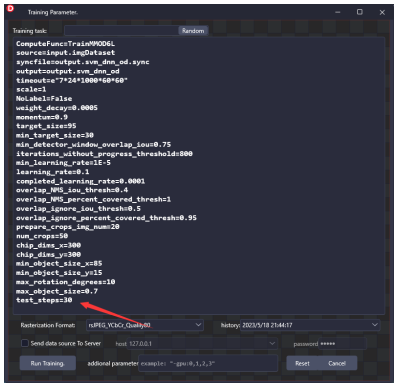
从 api 层来看,在检测器底层,多了一个叫做 test 的训练参数,test 在输入层会被高频率引用到数据样本中,例如,输入 10 个样本,其中 2 个会保证是 test,另外 8 个会有 1 个会是错误标注,这样来走训练流程,最后的收敛结果:错误的标注会慢慢被淡化,但不能忽略,正确的标注会慢慢被提升,从而避免了长时间训练无法提升精度.在大规模的检测器训练中,该机制立竿见影.

另外,约定了训练用的样本具备 test 属性,它的作用就是规范化检测器训练大规模样本.

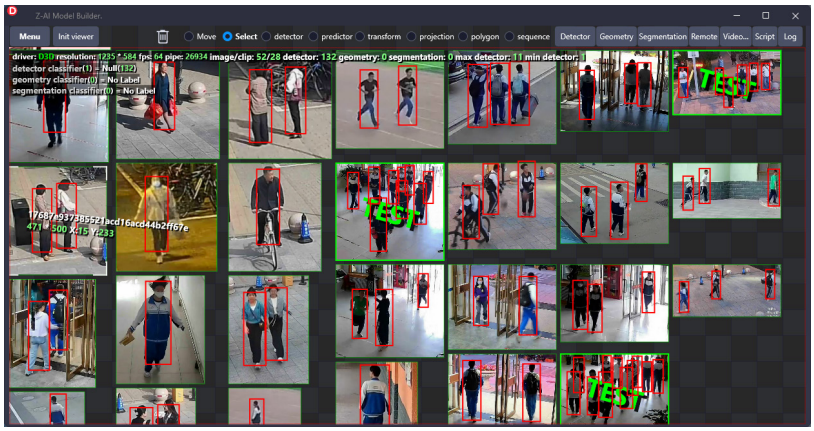
API 参数层:



建模工具参数:



数据样本在建模工具可以带有 test 属性,
Test 属性可以被各种接口操作,可以被脚本批量操作,
Test 属性只解决检测器框框太多不迭代问题,不影响训练其它模型.



提示:在几百个框框的标注规模下,不需要开启 test,建模流程继续按照老版本运作.

AI-Tech-2022

AI-Tech-2022 诞生于全国封锁+新冠肆虐期间,**初步构思接入一些生成式的技术体系,无奈 openai 这类开源项目的体系发展实在太快,外挂式的 ui,组件式的大流程,让 openai 整个体系非常接近于实际应用,这时候,Tech-2022 定位偏向了 CV 大数据和无监督方向**.

Z.AI.pas 单库逼近 25000 行,扩展修改,变得略微困难,未来只会对其做优化层面的维护,不会往里面堆新算法,新的 AI 算法和技术,都会往 Tech-2022 这套体系集中.待未来,Tech-2022,也是逼近 3 万行后,那就再开个新库来干.

Tech-2022 保留了 Z.AI.pas 的全部设计流程和机制,在实际使用中它会要求多支持一个计算引擎.例如,原有项目使用 Z.AI.pas 库中的计算引擎, 使用新技术时,会要求项目需要额外支持 Z.AI.Tech2022 引擎.它对于流程无影响,**对大型流程框架稍微影响:需要框架支持双引擎机制,但这并不难**.

Tech-2022 不是淘汰老的计算引擎,而是新增更多的算法和框架,在应用时,会出现新老并就的多计算引擎同时工作局面.新老并就含义是,未来的 AI 程序需要在原有的一套流程结构基础上新增一套 tech-2022 的流程结构,AI 程序内部将会出现双引擎局面.

Tech-2022 会延用老引擎的 gpu/cpu/操作系统的兼容架构,例如老引擎如果支持 cuda12,那么 tech-2022 也会支持 cuda12.

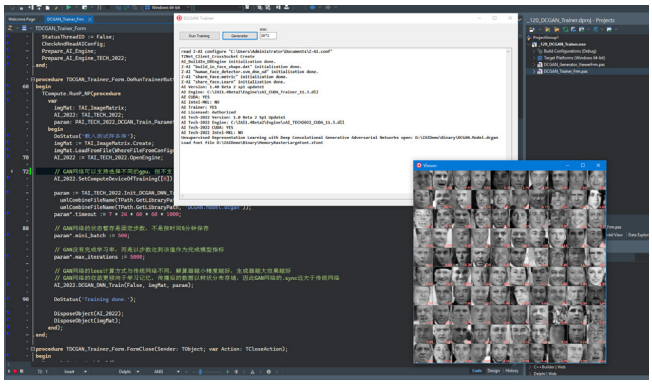
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

简称 DCGAN,这是一个小规模生成式模型,开发模型前 Stable-diffusion 还没有摆到应用层.同时 DCGAN 也是 Tech-2022 支持的第一个模型,它为 tech-2022 底层体系成型做了很大贡献.

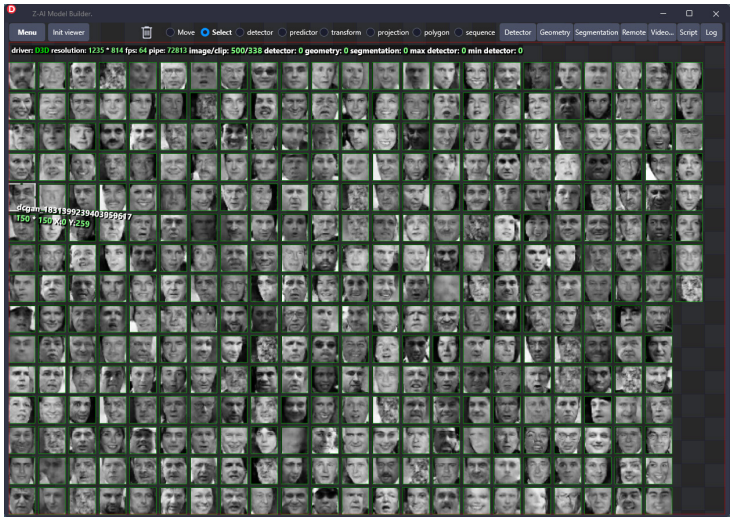
DCGAN 的内部技术体系这里不多做介绍,时间富裕时,我可以专门写个翻译式的 paper,这样干的意义并不大,复刻它就是一个新模型的试水行为.

- DCGAN 不支持多 GPU 训练,训练只可以使用一张 gpu,但可以使用 dnn-thread 技术实现多 GPU 生成.
- 生成目标分辨率极小:不能高于 64*64,大图操作,都是围绕 64*64 进行缩放.
- 可以生成手写字,仿真人脸等等.
- Stable-diffusion 非常强大,没必要在 DCGAN 下太多功夫.

DCGAN 有标准 demo,详细讲解了训练模型和 API 使用.demo 编号 120



在建模系统中,DCGAN 可以直接训练和快速生成小图



ZMetric-V2.0

简称 ZM2,这是 Z-AI 的自创 ZMetric 模型 V2.0 版本,该模型解决了许多个现实问题

- AI 识别有很多相似的错误判断,例如绕耳朵被识别成打电话
- 所有的识别结构都具备候选化机制,既,识别目标可以有多个,算法不再做唯一性返回,候选结果会交给业务层去处理.
- 底层分类器无法强推理,当目标有 1000 个样本时,现实中的识别率和 100 个样本相差不大
- 把 100 个的样本通过训练达到 1000 个样本的效果和准确度
- 反之推导,当我们有 2000 万个样本时,也许服务器无法负载了,而我们的服务器只能支持到 100 万样本,但能达到 2000 万样本的效果.

上面这一系列问题,就是 ZM2 的解决目标:更少样本+更高精度

ZM2 由一系列分支新算法重新匹配老的 ZMetric 网络组合而成

- (纯 pas 系)CV 方向抖动计算分支:抖动从算法设计到验证,再到独立 demo,经历了一系列正确性验证,因为这些抖动计算对于计算结果的要求很高:要求绝对符合尺度比,符合长宽比,符合多线性采样规则,并且保证宽松的抖动范围和尺度.外面看似简单,底层规则多多.
- (纯 pas 系)候选化识别计算分支:这项分支牵扯比较多,涉及到 KD 树优化(做了一堆测试程序,包含计算能力,精确性,api 合理性,另外还有 2 个 console demo),结构化流程(候选的前置是多采样,例如一个人脸的框框,需要从不同角度,尺度,旋转度进行采样,然后输入给 ZM2 做识别计算,完成后,计算结果多则上万条,这些数据需要用结构化处理流程进行过滤筛选,这样才能达到应用候选机制的作用),在应用候选机制时,项目的可选计算参数会列出一大堆(高级计算流程的参数永远都是很多的,未来 ZM3 也许会更多)
- (纯 C++翻 cudnn 文档)对 ZMetricV1.0 网络进行计算工程改造,因为 ZM2 定位就是偏大数据的,它与 1.0 相比必须凸显处理大数据的优异能力,例如输入 1000 张图给 gpu 计算 1.0 会等 gpu 计算完成再进行下一次 step,在 ZM2 这是直接进行的.它的现实意义是:ZM2 的计算机制就是无限抖动输入,耗时将比 1.0 多出几百倍,这时候,要运用多 gpu 的能力,就需要挖掘 gpu 的底层硬件机制.

ZM2 是 1.4 版本中 Tech-2022 体系的应用型模型,如果没有 ZM2, Tech-2022 将无存在意义.

ZM2 引入具备了候选机制,IO 过程会变成,输入图像,识别范围多个可能的结果,这对于现有项目是需要一定的改造工作的.对比原有的机制,输入一个,返回一个,改造将会变成,增加更多的可控计算参数,更改流程机制.

例如原本人脸输入完,直接给出这是谁,ZM2 给出的是谁更接近,意味着程序需要从单一结果适应性,变为多结果适应性,这将更符合现实应用,因为 AI 本就是一个计算接近度的机制.

ZM2 的缺陷

候选化机制

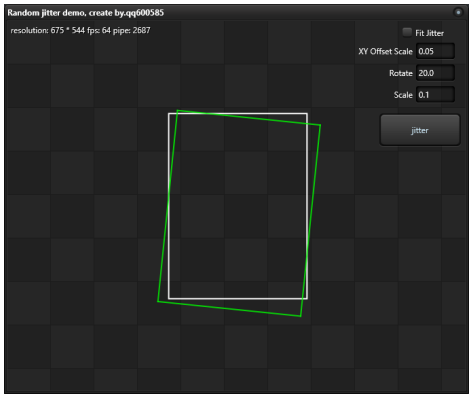
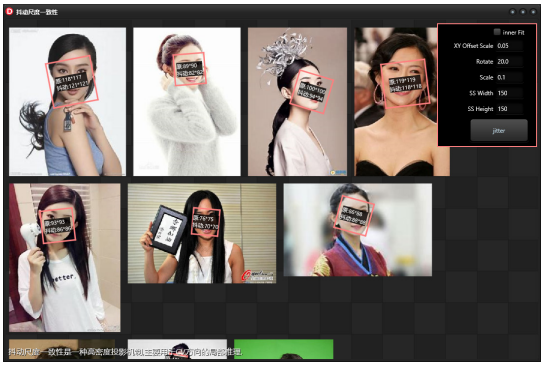
候选化机制对于程序改造,是最大的缺陷,需要从头到尾的考虑项目应用性,同时改造工作也非一蹴而就,同时,程序改造也会是算法优势,两者目前相持平,在克服了缺陷以后,优势将凸显.

ZM2 是基于 Tech-2022 体系的

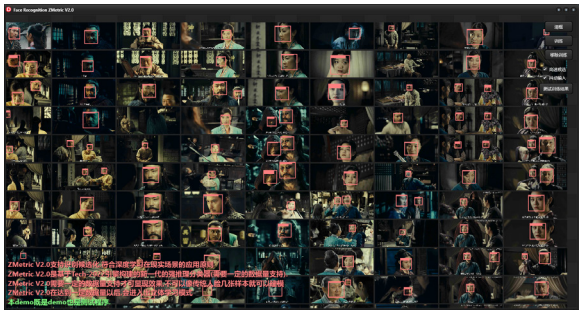
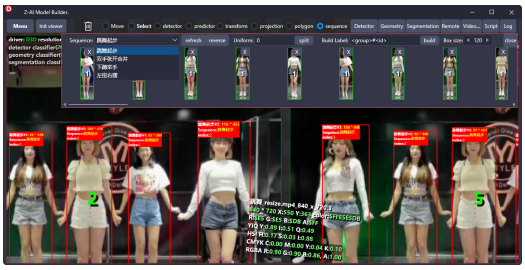
Tech-2022 有自己的 dnn-thread 系统,并不是与 Z.AI.pas 库使用同一套多线程引擎,因此改造 ZM2 还需要从新考虑在程序中接入新的 dnn-thread 系统.简单来说,使用 ZM2 必须是支持双引擎的程序模式,以前那种一个计算引擎的框架,是无法支持 ZM2 的.

- RealTimeTester 框架,无法支持 ZM2,改造工作巨大,只有待未来时间充裕后,支持 ZM2.
- Face3.0 框架,无法支持 ZM2,改造工作巨大,只有待未来时间充裕后,支持 ZM2.
- 监控 V1.0 框架,无法支持 ZM2,改造工作巨大,只有待未来时间充裕后,支持 ZM2.
- **已支持 ZM2 的应用型框架,只有 6 代监控系统**.
- **90%建模工具链已经支持 ZM2**
- **Large-scale 大规模训练系统已经支持 ZM2**

ZM2 的分支算法 demo

Jitter 初步算法原型,demo 编号 121	Jitter 尺度一致化验证,demo 编号 132
	

ZM2 的高级 demo

人脸识别候选化, demo 编号 133	放大后的候选化结果,demo 编号 133
	
基于 ZM2 模型的跳舞动作识别 demo,使用 6 代监控的驱动引擎	跳舞行为识别在建模系统中
	

ZM2 在第 6 代监控中的应用

AI Processor 服务器开预览工具	6 代监控管理中心端
	

优化 Multi Tracker 多轨道追踪算法

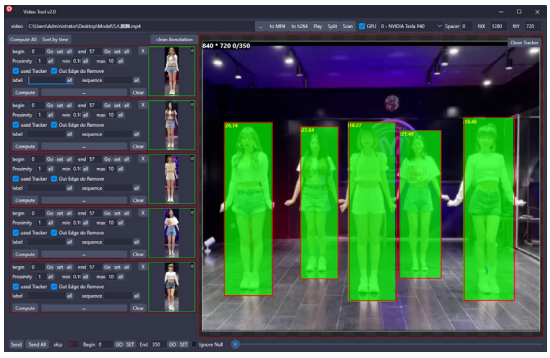
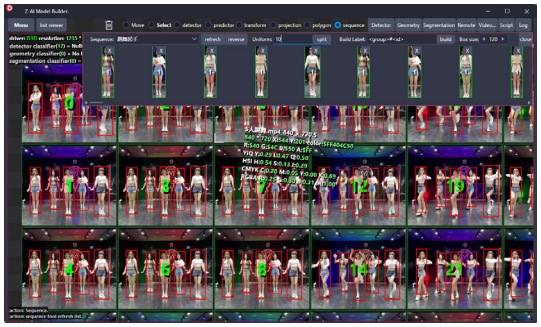
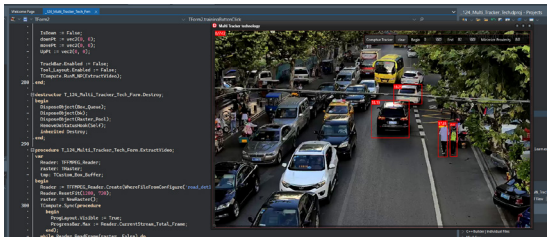
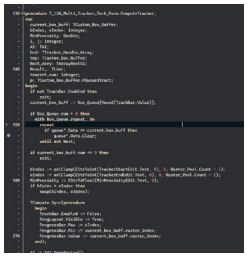
multi tracker 基于计算引擎 correlation tracker 体系做了一次应用层优化,主要解决 fft 加速问题,在解决过程中尝试了 sse2/avx2/avx512 等等加速手段.tracert 计算过程并不怎么消耗算力,对于 tracert 来说是把光栅内存转换到 tracert 结构在消耗 cpu 算力,延迟最大的也在这一步,试想一下,500*1920*1080*4=40G,500 帧 1080p 需要编码后转换到 tracert 结构.这对于 cpu 来说数据量非常大.总结:**优化非常+必须**

如果从连续视频帧采集样本,例如行为识别所需要姿态序列化,必须用到多轨道追踪,如果单帧采集,或则用单图标注,这种工作效率会非常低下,也许一个弯腰捡手机会把标注者弄的非常累.对于视频来说,弯腰捡手机,这只是一个 2 秒的行为数据.一个框选并且打开 tracert,2 秒内的人类姿态数据就生成出来了.

是的,multi tracert 是在行为识别中用于分解连续帧行为的关键技术之一,但并不意味从视频采集行为会变得很简单,其实行为采集的操作过程很复杂.例如,视频源有跳帧现象,很可能 tracert 出现丢失,例如,视频源的晃动非常厉害,且帧率只有 25fps/s,tracert 也会出现丢失,例如,部署一次 200 帧的 tracert 任务,运行中发现 tracert 丢失,标注者需要有效处理这些问题,否则行为建模将会无法持续.

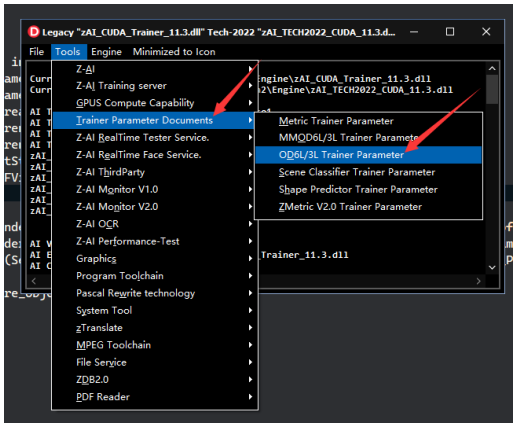
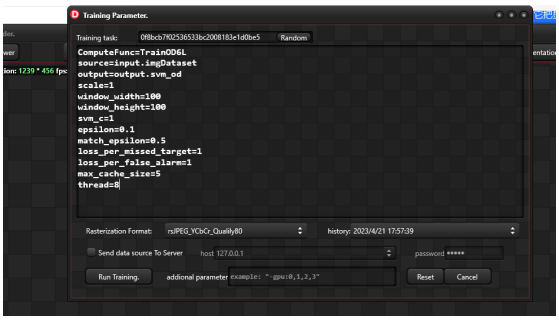
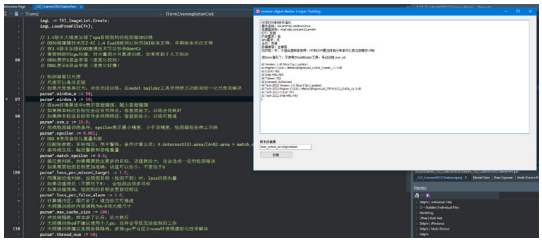
另外,在 ps,stable-diffusion 这些工具的协助下,姿态行为是可以产生仿真变体数据的.例如弯腰捡手机可以老太太,也可是小男孩,这些仿真的数据变体如何处理是一个重要问题.

总结:因为数据来源不可靠,multi tracert 不可以是简单解决连续性行为采集,但没有 multi tracert 支持,采集行为姿态数据将会变得更加困难.

在建模系统中对视频做行为采集,这些多框框就是 multi tracert 技术 	完成采集后的行为序列化数据,这些数据只能说贯通了行为识别流程,在实际系统中,行为识别对数据量要求非常大,这会上千人无限多的行为样本 
multi tracker 的 demo 编号 124 	124 的 demo 需要并入到主工具链,因此 demo 也是一种对可行性的验证,同时,并入主工具链的代码,相比纯粹 demo,复杂度会略高,这些 demo 中几乎没有备注说明,需要靠各自水平摸索 

优化经典 OD 检测器训练参数

在 1.4 版本中,OD 的训练参数被大幅增多
如果在项目中涉及到的经典 OD 检测器,本次更新将会是品质飞跃
如果要制作跨平台的通用检测器,并且不使用 GPU,例如人类,垃圾,汽车,汉字,1.4 版的 OD 将发挥作用

1.4 在工具链系统给出了训练参数的详细使用指南(需要安装引擎) 	相比老版本增加了许多可调整训练参数 
demo 编号 122 	

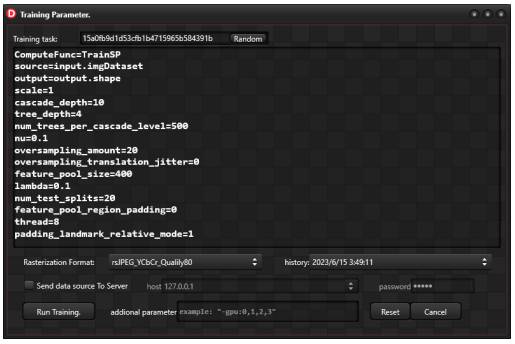
优化 Shape Predictor 训练参数

简称 SP,这是人脸对齐,矩形对齐的,的关键模型,sp 工作于 cpu 平台.

SP 通常处于中间层 检测->sp->识别,作用是对目标框做一次几何化识别,然后再根据几何坐标做矫正处理.

本次新增了大量 sp 的训练参数,可以自己建模复杂的几何匹配模型

在未来建模 sp 会被应用于很多项目

demo 编号 123	SP 在建模系统
	

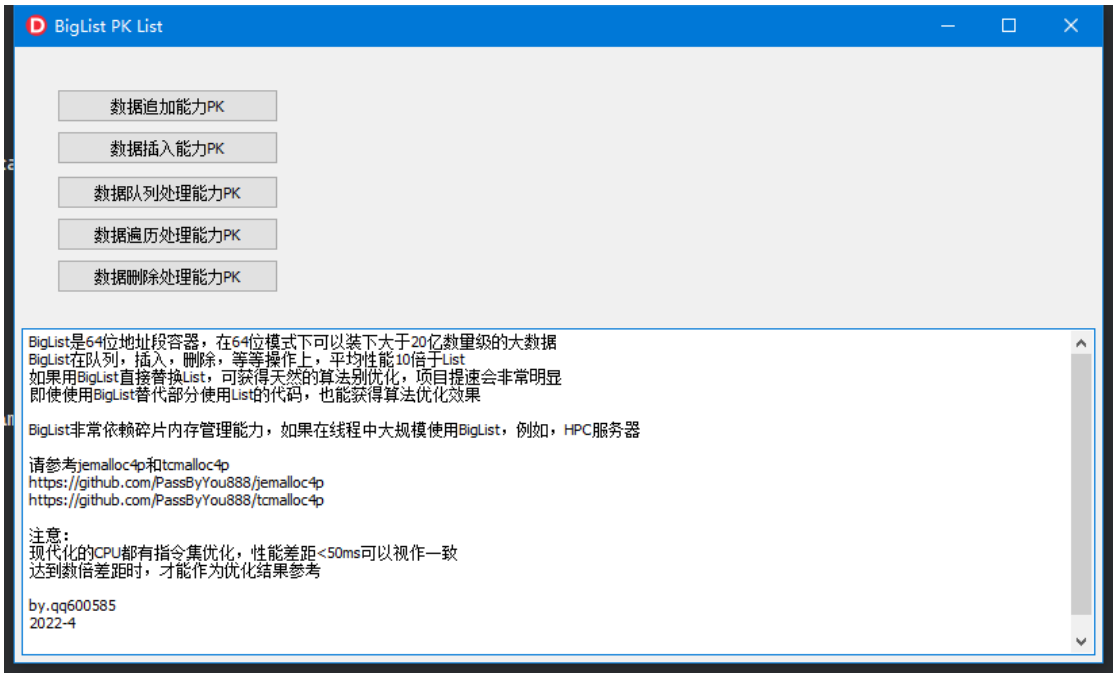
BigList,高速链表模板

- BigList 主要支持数据队列,大数据等等机制,BigList 具备 64 位最大数量限制,List 和 Arrry 模型是 32 位最大长度限制.
- BigList 在增删查改领域,性能几乎是 List 和 array 的 500 倍
- BigList 位于整个 1.4 的最内核区间,向上辐射全部使用链表的程序
- BigList 有 OrderStruct,BigList,Hash-Pair,三个分支算法,中心设计思路全部靠进 BigList

使用 BigList 的大型程序分支,小分支不计

- 全部核心并行程序,TCompute 线程池调度器,数据队列池调度器
- Z-AI 1.4 中 90%的链表结构
- ZDB2 全体系均从 BigList 架构,包含多线程模型的命令队列,ZDB2 数据集引擎
- DrawEngine 全体系,100%的链表结构均被改造成 BigList 体系
- Learn 统计学库 90%分类数据集程序改造成 BigList 体系
- ZNet 全体系均使用 BigList,为 100%涵盖率

关于 BigList 的使用和测试
BigList pk List 的 demo 编号 119



画音同步

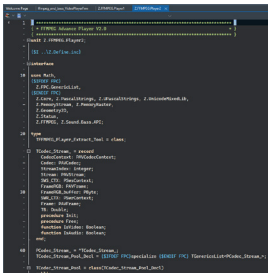
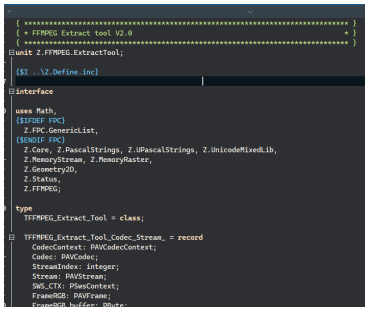
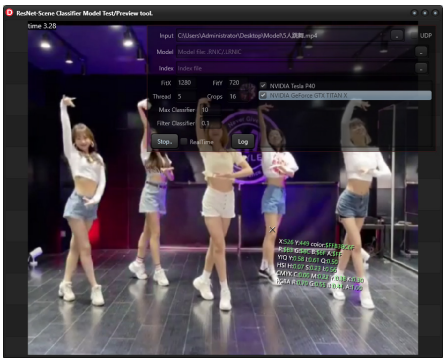
画音同步是指视频播放中,同步视频码流与音频码流,这一部分看起来是个计算时间和帧的方法,但其实,这会关系到许多结构化程序,这些程序使用高级链表,例如 OrderStruct,BigList,最后才是将各种缓冲上下串联起来.

首先,打开视频 or 音频,如果是封装格式,例如 rtsp,rtmp,mp4,mkv,这些封装就会包含码流,码流里面要区分多视频码流,多音频码流,多字幕码流.处理这一机制时,就需要开结构了.

在码流的下一级,是码流的数据工作机制,视频码流按帧进行切分,音频码流也是按帧进行切分,例如,当计算出视频帧在时间小数 4 位(用小数点 4 位是毫秒为千分位)后的帧时间戳,就需要匹配对应的音频帧位置,这时候就需要缓冲,缓冲是将未来时间的视频 or 音频缓暂存起来,然后匹配两者最接近的帧位置.而缓冲,也需要开结构.

当多码流结构,以及多码流中的帧结构,确定下来以后,是用 api 对数据进行逐帧的处理,封装之类,这一步主要是一个对接的过程,例如,把视频对接到渲染输入,把音频对接到音频数据帧输入.把整个流程串联一通后,是一套结构化结构设计+流程设计的算法.

结构化程序算法都是较贴近实际需求的,画音同步稍微修改就可以是一个直播端,或则是一个原生 pas 版 vlc,迅雷影音等等.

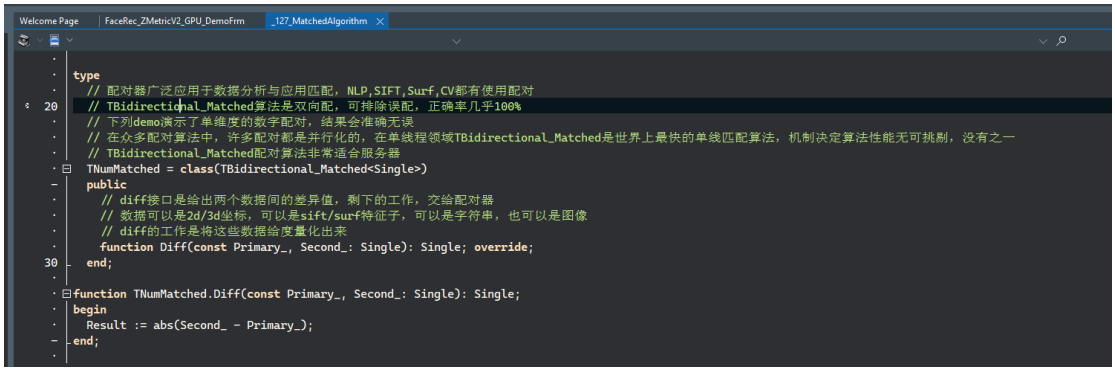
<table>
<tr>
<td>画音同步的 demo 编号 112,
  </td>
<td>支持画音同步有 2 个库
Z.FFMPEG.Player1,画音同步 1.0 版
Z.FFMPEG.Player2,画音同步 2.0 版
  </td>
</tr>
<tr>
<td>从画音同步框架衍出了视频帧解算器
Z.FFMPEG.ExtractTool,因为体系问题,目前 AI 系统并没有把 Z.FFMPEG.ExtractTool 摆到前台来,而是继续沿用老的视频 Reader+Writer 体系.
Z.FFMPEG.ExtractTool 是一个美好并来不及实践想法
  </td>
<td>基于画音同步的分类器测试系统
因为我们在 6 代监控考虑到许多摄像头设备支持音频采集,而我们还没有一个可以播放画音的工具,于是,顺手就把测试工具和画音合体了
  </td>
</tr>
</table>

小流程算法

小流程算法偏局部个体算法,没有衍生体系,大多拿来就用,不会要求建模系统,工具链去支持

Z.Matched.Templet,双向配对算法模板

在单线程模型中,基于 Z.Matched.Templet 构建的配对流程是地球上最快的框架
配对 Demo 编号 127



Z.BulletMovementEngine 弹道运动

弹道运动并不是新库,而是将老库重新梳理并给出规范的应用模型.将老库摆到了应用前台
当 Z.BulletMovementEngine 被绑定到渲染器以后衍生出了许多带运动属性的渲染能力

demo 编号 109

基础程序类

自 1.4 开始,Z 系开源项目,Z 商业交付项目,所有 Z 系的 code 全面使用机器化转译.

代码转译是一种 code 模型工具,可以大批量的对 pas 代码进行重定义:包括单元命名,各种申明的命名等等.

机器化转译就是将 Z.AI.pas 转译成 OS_Z.AI.pas,除此之外,这些单元内部的命名也会进行大规模更新,一切都使用机器完成.

在大规模转译系统中,会有许多的项目分支定义,例如以 M3C_开头的 zMonitor_3rd_Core 开源项目.由于未来的分支代码众多,母体需要特别说明:以 Z.开头的单元都是母体,母体表示发源地,未经过机器转译的原始代码.

基础程序更新的规模很大,近乎新项目,这里只挑一些特别的重要的程序更新

所有单元的申明环节都被大规模修改命名.如果现有项目需要升级,可以使用转译模型.具体操作方法到 ZNet 开源项目寻找 PRP 程序,命令行执行 prp.exe "-d:c:\my proj\"

基础程序类只讲诉宏观更新

Z.AI.pas 库更新(对应 zAI.pas 库)

- 增加 OD 建模参数
- 增加 SP 建模参数
- MMOD6L/MMOD3L 增加 test 标签支持
- 统一 Trainer-API 接口,不再由工具链自己局部规范输入参数,所有的训练参数均被统一化安置在 Z.AI.pas 库,未来增改训练参数都在这里干

新增 Z.AI.Tech2022.pas 库

- 新增 DCGAN
- 新增 ZMetricV2:主要解决大数据建模问题,V2 已经完善周边支持体系
- 使用自己的独立 DNN-Thread 子系统,它在 1.4 版本与 Z.AI.pas 库的 DNN-Thread 相差不大,未来随着各种 api 的堆砌,Tech2022 长大以后,会有许多优化+兼容性改动.

Z.Learn.pas 库更新(对应 Learn.pas 库)

- 优化 KDTree 查找流程
- 主要更新候选化体系支持机制,更改幅度相对大
- 其它地方都是小改动

Z.Net.Pas 库更新(对应 CommunicationFramework.pas)

- Z.Net 库的具体细节请参考开源项目, https://github.com/PassByYou888/ZNet
- 在不改变协议机制的前提下,几乎重做整个通讯协议栈,并且深度优化
- 开始建议所有用户逐步放弃直接使用物理 socket 通讯方式
- p2pVM 被提上前台,成为未来主要的通讯接口
- Z.Net.pas 定位倾向于规范化底层接口,作为未来后台程序支持地基来使用

新增 Z.Net.C4.pas 库

- Z.Net.C4 库的具体细节请参考开源项目, https://github.com/PassByYou888/ZNet
- 使用 Z.Net 直接裸堆项目是艰难的,C4 库提供了规范化的服务器模型,解决后台系统做大难题
- C4 是一个后台服务器体系,并且后续的升级更新力量很够

新增 Z.ZDB2.Thread.pas+Z.ZDB2.Thread.Queue 库

- ZDB2 是一套自主设计数据库引擎的技术体系,不是直接数据库
- ZDB2 也是未来的大数据库技术,在多线程模型被支持后,ZDB2 就长出翅膀了
- 未来计划针对 ZDB2 做一些开源支持工作,目前还没有把该计划摆上前台.

Z.MemoryRaster.pas 库更新(对应 MemoryRaster.pas)

- 光栅库更新很小,主要是一些命名规范化
- 1.4 没有出现技术体系级更新动作

Z.DrawEngine.pas 库更新(对应 DrawEngine.pas)

- 将内部使用的 List 链表模型全部更换成了 BigList 模型
- 新增弹道运动仿真和弹道运动渲染支持
- Z.DrawEngine 库可兼容多平台,但没有图形输出载体,只能在内存中仿真渲染,目前暂时没有接入 IOT/Linux 下某些渲染引擎的计划,曾经有过这种计划后被打消了.
- 目前 Z.DrawEngine 库暂时没有增强 2D 游戏方向渲染的计划,除非中国解除 IOS/Android 的版号限制(几乎不可能发生)
- Z.DrawEngine 周边支持性更新不小,例如 Picture Viewer 编辑器几乎重做,细节省略

Z.FFMPEG.pas 库更新(对应 FFMPEG.pas)

- 周边支持库,例如 Reader/Writer 都新增了,GPU,多线程,自定义参数类支持
- 规范化编码解码处理流程,Reader 更安全稳定
- 当前 FFMPEG 版本为 3.4.2,目前能很好兼容 gpu 设备,不用升级底层 api

Z.Core.pas 库更新(对应 CoreClasses.pas)

- 最底层的内核库因为新增了 BigList 之类的技术体系,内部改动很大,外部 api 仍沿用规范
- 围绕 Z.Core 库升级以后,外面的结构设计,流程设计,都得到了一次升级
- 使用 Z.Core 泛型模板搞 solve 程序更方便,BigList 体系可以有效降低问题 solve 复杂度

Z.Expression.pas 库更新(对应 zExpression.pas)

- 表达式是个技术体系,表达式无所不在,ini,json,xml,html,tedit,tmemo,debug 都可以表达式
- 新增一个执行热点接口,可以选择在当前线程或则主线程下执行 OpCode,此举是 C4 的 debug 必须功能,因为 C4 要求服务器运行期间随时监控内部运行状态,这些监控都是命令行形式,驱动程序就是用的 Z.Expression
- 其它地方几乎无改

新增 Z.Json.pas 库

- 解决 json 处理流程一致性问题,兼容 fpc/delphi,不用各写一套流程.

新增 Z.Pascal_Code_Tool.pas 库

- 解决 pas 代码机器化转译实现库

Z.Notify.pas 库更新(对应 NotifyObjectBase.pas)

- 延迟化,后置化,这些机制几乎能涵盖所有的应用场合
- 优化了延迟触发器的分离机制(传统链表被替代成 OrederStruct+BigList 模型)
- 给出 DelayFreeObject 机制:例如在线程中,类直接释放自己与延迟释放的安全性是不一样的(多线程环境底层库存在漏洞,不细说),不管怎样,延迟释放是一个很必要的 api

基础技术体系类

基础技术体系指技术方向性质,并且带有策略性,实用性,条件性,高完成度这些特点的技术路线规划

新增 ZDB2 体系

首先,明确 ZDB2 的未来使用路线:大数据底层地基

其次,明确大数据问题 solve:数据库引擎必须遵守计算机机理,数据库应用场景永远不确定,数据库不会是一套体系解决一切数据问题,**ZDB2 认为:解决数据库问题就要自己设计数据库引擎,在所有的项目中都要使用独特自主设计的数据库引擎**.

ZDB2 体系的技术路线推进回顾

1. 给出 ZDB2 底层空间表支持系统:解决数据块的增删存改问题
2. 在空间表支持系统基础上,尝试过小规模应用,例如 C4 系统中的 UserDB,FileDB,效果平平
3. 开始自我反思,之后,给出了线程化存储引擎体系,从底层机制来看待,ZDB2 此时已经具备大规模存储支持能力.
4. 支持了线程化存储引擎后,在落地项目应用,将数据库构建与数据中心服务器群集.
5. 在数据中心运营过程,ZDB2 被地狱级难度洗礼,同时也具备了新属性,包括可扩展性,数据安全性,增强自定义数据库设计.在 6 代监控系统,ZDB2 完成了单日 PB 级数据量驱动,在无人维护环境中持续 3 个月不重启,不关机.
6. 走出地狱,面向阳光,坚持走数据库=自定义数据引擎的底层逻辑.

ZDB2 的后续动作

- ZDB2 未来会持续更新
- 暂时没有放到前台推广计划,但会开源 ZDB2,也会提供一些相关 Demo.底层逻辑是因为我是程序员,我与世界的关系就是世界需要我的贡献,同时世界也在随时计算和剽窃我,但世界不能阻挡我的贡献行为,长久以来我和世界都是一种相互踩平衡木的关系,这也是只开源技术,而不开源项目的初衷:拥抱技术,建立开源过滤机制.
- 因为 ZDB2 具备了大数据支持能力,未来高几率会在某些条件成立时,在底层会给出非常棒的数据地基.只有自定义数据引擎才能胜任一切需求.

大幅优化 ZNet 体系(原 ZServer4D)

ZNet 代表整个后台服务器体系,以 ZNet 为基础的上层架构.

- 详情去看 git, https://github.com/PassByYou888/ZNet
- p2pVM 是新一代通讯底层:替代物理 socket
- C4 代表大型服务器:替代双通道交互,提供大规模后台地基

从宏观来看,C4 体系+ZDB2 体系+BigList 体系+Core 体系

这套组合拳只能在 1.4 或未来的版本使用.

ZDB2 后台系统的数据引擎是独特的,ZNet 后台服务器几乎全部工作于线程模型(凡处理延迟大于 100ms 的请求都用线程拉),BigList 体系为数据链带来大容量提升,C4 为服务器后台提供了高级架构.

pas 圈子有许多以设计模式为工作路线的程序员,他们为广大入门者带来了傻瓜化的后台框架,他们有非常深厚的计算机理论基础.这造成了一个局面:使用者和设计者是依赖关系,不是完全性互相依赖,当设计者不能自我提升时,使用者就会被动.反应在真实生活和工作中就是当世界出现新事物的征兆,有许多人发现了它,但设计者无法驾驭,导致位于设计者下游的使用者无从下手.

以 delphi 厂商的某些程序员+设计者为例,他们的工作是维护和开发 delphi.我在 2016 年时看见他们还在折腾 opengl 入门,显然错过了 open1.x(固管) -> open2.0(可编程管线)的历史性升级时代,delphi 通过商业路线收购 dxscene,fmx 被提上前台,这是被动行为,这并不是开创型做法.如果设计者在半路上出现自我局限,项目无法达标,使用者就会非常非常被动.

总结:**ZNet 代表未来整个后台服务器体系,保持开放,保持升级更新**.

ZNet 后续动作

- 非常不确定,也许会开源网盘系统
- 2 选 1,也许会给数据中心做个 filezilla 的 Z 系版本
- 2 选 1,也许只是给 https://github.com/PassByYou888/NetFileService 做个升级来完结
- 100%几率,增加一些关系到 ZDB2 的 demo
- 100%几率,增加一些关系到 http 的中转后台模型

计算机图形学体系(小规模性能优化)

DrawEngine 简称 DE,定位是轻量+标准化,它需要渲染输出接口.位于前端用户层.

图形 API 标准源自 Khronos,软硬厂商也会有自己的规范,这些规范,大都反应在像素排列,像素优化,api 优化这些机制上.

同时图形 api 也有许多标准,metal,opengl,gles,vulkan,d3d,这就造成了一种局面:渲染器要兼容各种平台和 api,需要大量接口,并且这些接口升级更新非常快,这让渲染引擎的开发和维护变得不容易了.DE 在设计定位直接避开了前面的标准接口和平台化,做中间件:渲染 api 差异大没事,渲染器能在目标平台把至少一种 api 体系支持到位那就没问题.

FMX 的底层是一种渲染器,对于多平台 api 支持程度,其实做的是可以的,用来设计游戏,VR 都是没问题的.DE 是把 FMX 当成输出接口来用,但不会局限于 FMX.DE 不是渲染器定位.

DrawEngine 的应用并不是直接让程序写完流程,因为渲染需要数据来源,例如,视频,图片,各种可视线框,大多时候,这要工具链,后台,内容生产这类体系来支撑.纯代码路线无法驾驭.

另一方面,DE 有一定计算机图形学的全局视野.它走了自己路线.

在 1.4 更新中,DrawEngine 的渲染调度能力被大幅优化.

计算机图形学 solve 组成部分

光栅系统

- Agg 分支:以 Agg 命名开头的体系库,给出了线条平滑性 solve
- 投影分支:由光栅字体分支,尺度化,api 支持分支共同组成
- CV 分支:形态学分支,像素分割分支,共同组成

渲染系统

- 文本渲染:彩字,倾斜字,自定义字,脚本化文本
- 图片渲染:没输出接口走光栅流程,有输出接口走渲染器流程
- 线框绘制和填充:以原子操作,点,线,园,扩展一堆高级 api
- 命令队列:就是把渲染 api 调用,和物理输出 api 绘制分离开,用至少两个线程提速
- 粒子:粒子是个重要分支,做特效,视觉方向使用
- 运动引擎:运动是渲染引擎必须支持子系统,作用是统一规范,减少实现运动化渲染代码工作,运动引擎是在 1.4 新引入的分支
- 场景支持:如果复杂项目,例如,2d 游戏,AI 建模系统,场景是渲染引擎的必须支持的标准规范,在 DE 中场景化渲染以 DrawInScene 这类 api 来使用

AI 体系(大规模更新)

AI 算法体系+ AI 数据库体系+AI 工具链体系+AI 平台体系=AI 体系
AI 体系非常庞大,更新文档不合适用于细说体系方面的结构

AI 算法 API

- 目标检测器,6 种目标检测器,全部包含训练 api
- 目标分类器,4 种度量化目标分类器,2 种直接分类器,全部包含训练 api
- 场景分类器,2 种场景分类器,全部包含训练 api
- 语义分割器,目前只支持了 U-Net 语义分割,包含训练 api
- 多轨道追踪,跟踪视频帧的框框,主要用于采集工具
- OCR,文档识别方向,不是自然场景
- 一堆 CV 性质的 API

AI 算法 API 底层框架

dnn-thread 框架是把驱动多 gpu 标准化,发挥硬件潜力,大型框架例如人脸,监控,都构建与 dnn-thread 框架

关于 AI 算法和 AI 支持体系

算法体系由算法 api+算法支持组成,算法支持是指运行算法条件,例如建模系统,算法应用思路,比如 dnn-thread 系统就是一种算法支持,样本数据库也是一种算法支持,基于样本数据库搭建的建模工具也是一种支持.

算法 api 这一部分就是大家在 git 接触各种 CV+识别性质的开源项目.ZAI 会有些自主创新,而算法思路和方向,与这些开源项目基本算一致,甚至一些开源项目比 ZAI 的算法更加先进,识别率也会更高.**看开源项目是看更新强度,持续的强更新才能真正推进到社会生产和应用,否则会被时间洗掉**.

在 ZAI 的早期版本(1.2/1.3),一直很注重算法体系的建设,基本每次更新,都会增加一大堆新算法.后来做完发行版本以后,这些算法加在一起,已经可以等同于数十个开源项目了,同时这些算法大部分都没有被应用.这就很尴尬!因为应用最多的只有,目标检测,目标分类,场景分类,其余算法基本不会应用,即使再增加几十种算法,也不会改变什么,例如把 yolov7 加进来也会一样.在计算引擎集成多个同性质算法无意义,这并不会改变应用模式,也不会改变建模成果和项目成败.

算法与实际项目的匹配性其实非常小,而我们通过百度,虹软,商汤,这类公司使用的 AI 方案都是挖空心思做出来的,这并不是算法层面的.这些公司,他们有自己的技术体系,以及,应用层思路,算法是一种资源,算法的命运是被技术体系筛选和重构.算法的方案性占比非常小.算法周边的体系,这才是 AI 类公司赖以生存的核心.换个角度来说,AI 公司靠时代红利生存,算法是 AI 公司吃下去的食物,框架和算法支持体系是 AI 公司的消化系统,最后,产出各种应用方案.

明确了算法的本质定位以后,回到 ZAI 的算法体系中,算法的核心是算法支持体系,在此之

后,才是建立应用层框架落地.

ZAI 的算法支持体系部分,经历了 3 年洗礼,现在,ZAI 正在走出自己的路线.

AI 数据库体系

AI 数据库体系只能适用建模体系,例如场景分类器的样本最少也会上万张 720p/1080p,稍微堆大一点,这些样本回答道百万级规模.这时候,大数据管理,高效存取必须使用服务器硬件设备,例如大内存,>20 核的 cpu.这对于数据库技术有一定要求.

AI 数据库体系并不是传统数据库,而是程序级的数据结构,这些数据结构一律支持多线程加速,以及 Large-Scale(TB 级大数据支持体系).传统 mis/erp 开发用的 pc 无法适应 AI 数据库:样本规模一旦大了,内存就不够了,必须精简 or 使用 Large-Scale 方式来管理数据库,这会导致浪费大把时间去干一些和硬件较劲的无意义工作.

AI 数据库结构=适应所有的模型的数据要求的+样本链结构+样本链矩阵结构

AI 数据库体系=Z.AI.Common(AI 数据库)+Z.AI.Editor(AI 工具链副本数据库)+*.dproj(整个建模工具链体系,大约 20-30 个建模支持工具),这些数据库在存储底层大量使用 DFE,ZDB1,ZDB2 工具形式支持库.

AI 数据库体系不区分目标模型,但凡模型凡需要的数据都会在 AI 数据库体系对应支持,这些支持包括 API 支持,规范化支持,工具链支持.

1.4 更新对 OD 检测器数据结构增加了 Test 属性,生成模型训练的输入数据时会把带有 test 属性的样本放到一个新结构,然后一并输入给 gpu.

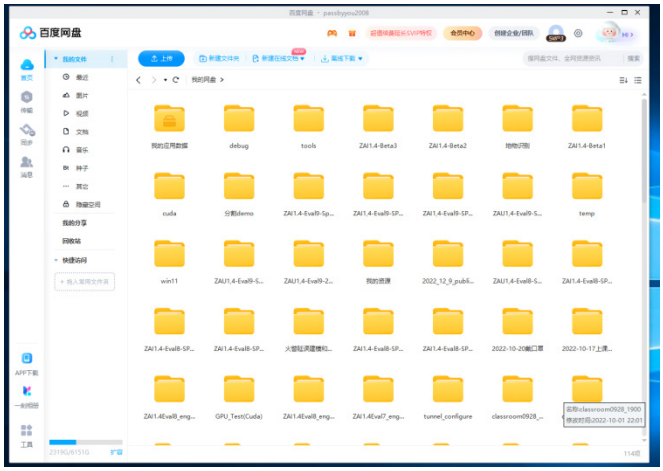
针对 AI 数据库操作方法只有一些非常简单的 demo,编号 75,主要演示数据导出.

制作三方样本与 AI 数据库的互相转换.需要用户自己编写高等级的 AI 数据库处理程序.

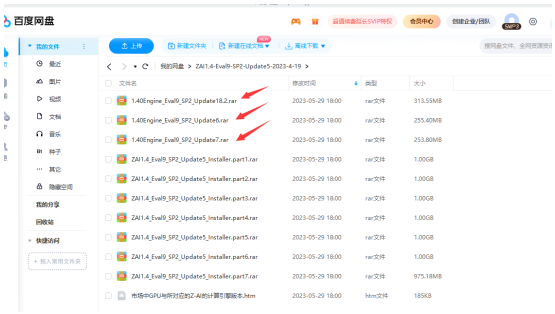
AI 工具链体系

由于项目的开发,AI 工具链一直作为内部版本在更新.
撰写本文时,工具链已经构建了接近 150 个版本,历经大量修复+更新.
版本计数历史自 1.4 Eval1..Eval9(这里走了 90%版本数量)
之后, Beta1..Beta3(这时候,已经逼近 2023-6 月出发行版本时间计划)

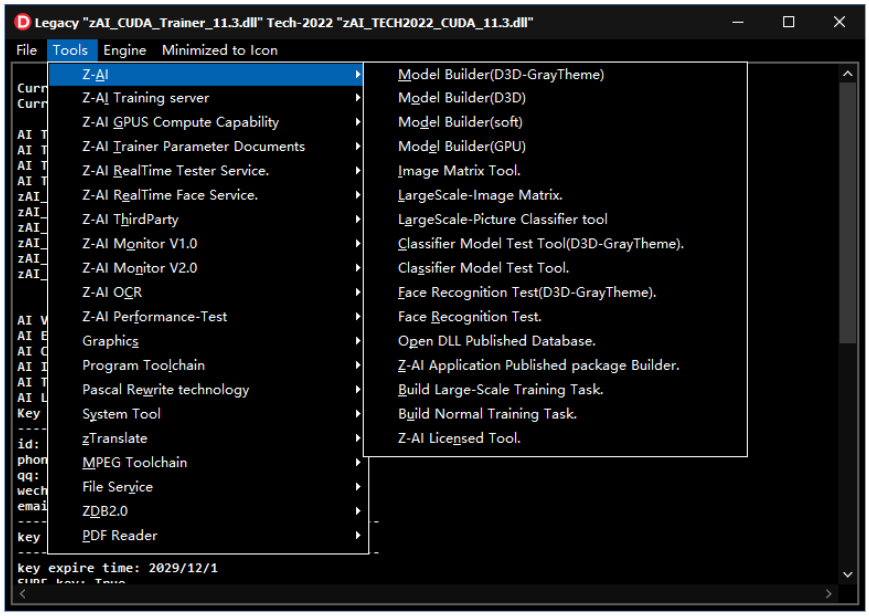
这些工具链版本一直作为内部发行在使用
时间跨度从 2021 至 2023



版本计数=大版本+小版本,例如发行了一个 Beta3,基于 Beta3 有 10 个更新版本=11 个版本
这些版本与开源项目不同,每一个版本都是庞然大物,在线更新几乎无法工作.只能通过打包工具或则安装程序来更新版本.



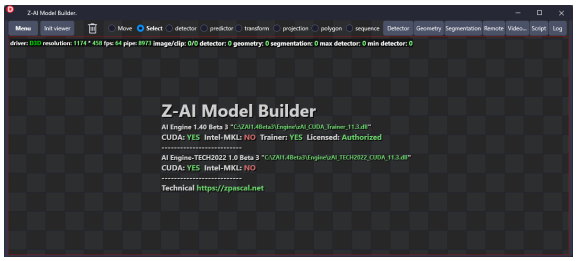
主工具链增加了一大批菜单选项



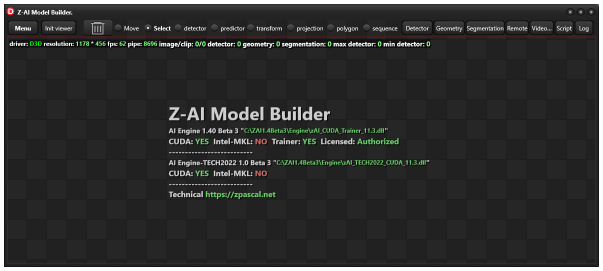
Model Builder 大幅更新

ModelBuilder 更新非常多,但不是 1.4 工具链的主要更新目标,以开发时间+更新频率计量,占比低于 1/10.

ThemeGray 界面



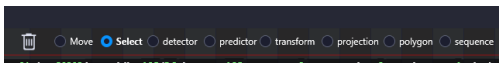
原界面



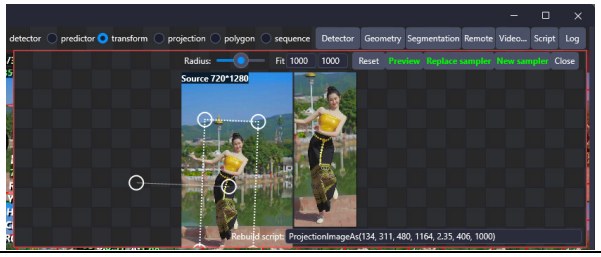
- 抬头会显示目标分类信息
- 明显 test 标签

按下 ctrl 键可显示序列号

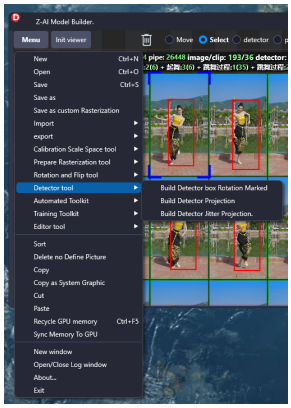
抬头编辑状态栏以常用性调整



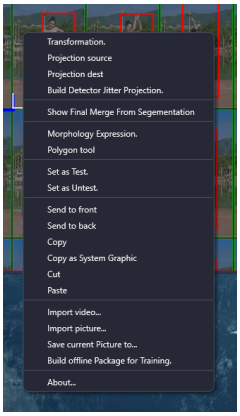
变换工具被摆到前台(热键 ctrl+5),变换几乎是做检测器框选的必须前置工作,另外,变换工具也支持了在脚本批量化变换



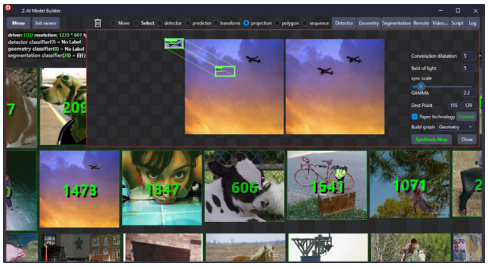
调整主菜单结构



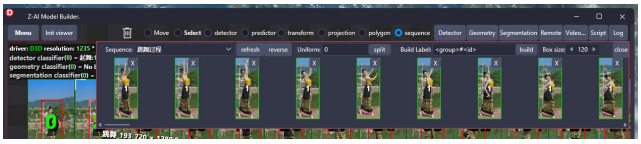
调整弹出菜单结构



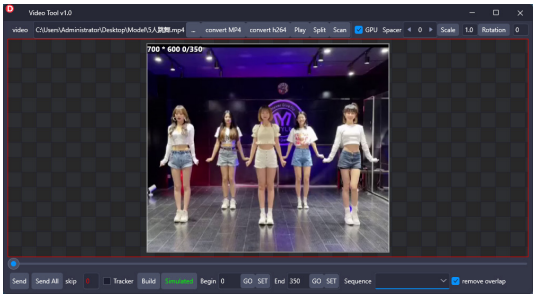
适当调大了投影工具



新增序行为列帧编辑工具

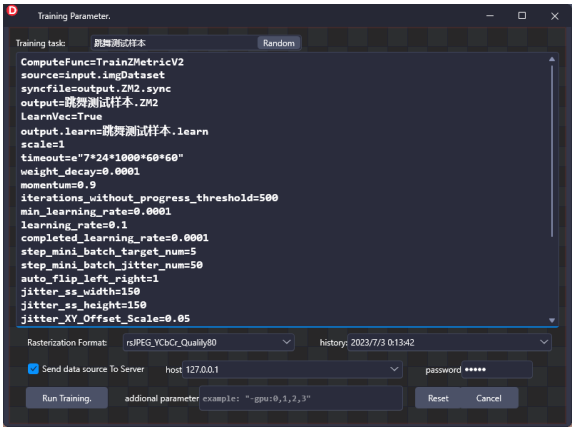


在不移除经典视频导入工具的前提下,新增第二代视频导入工具,两代工具共存



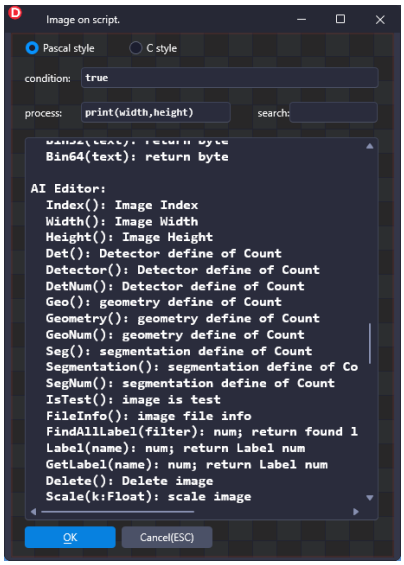
第二代视频导入工具:新增多轨迹追踪,fit 分辨率,指定 gpu.一代主要采集视频,二代工具可以采集视频+视频数据,两代工具共存

训练工具给出了训练参数历史,以及自动化命名训练任务



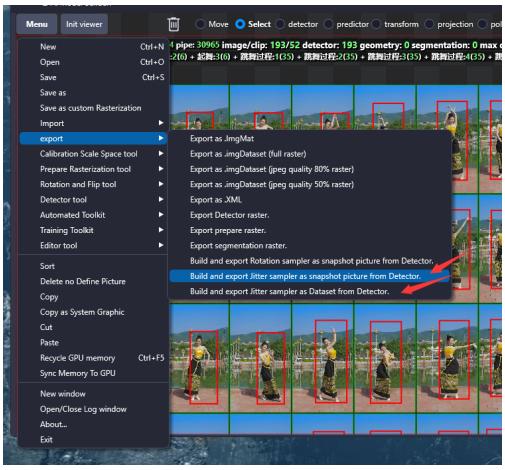
```
ComputeFunc=TrainZMetricV2
source=input.imgDataset
syncfile=output.ZM2.sync
output=跳舞测试样本.ZM2
LearnVec=True
output.learn=跳舞测试样本.learn
scale=1
timeout=e"7*24*1000*60*60"
weight_decay=0.0001
momentum=0.9
iterations_without_progress_threshold=500
min_learning_rate=0.0001
learning_rate=0.1
completed_learning_rate=0.0001
step_mini_batch_target_num=5
step_mini_batch_jitter_num=50
auto_flip_left_right=1
jitter_ss_width=150
jitter_ss_height=150
jitter_XY_Offset_Scale=0.05
```

脚本系统增加一堆常用条件+常用操作



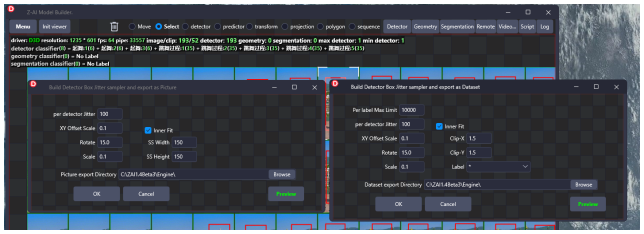
```
Bin64(text): return byte

AI Editor:
  Index(): Image Index
  Width(): Image Width
  Height(): Image Height
  Det(): Detector define of Count
  Detector(): Detector define of Count
  DetNum(): Detector define of Count
  Geo(): geometry define of Count
  Geometry(): geometry define of Count
  GeoNum(): geometry define of Count
  Seg(): segmentation define of Count
  Segmentation(): segmentation define of Co
  SegNum(): segmentation define of Count
  IsTest(): image is test
  FileInfo(): image file info
  FindAllLabel(filter): num; return found l
  Label(name): num; return Label num
  GetLabel(name): num; return Label num
  Delete(): Delete image
  Scale(k:Float): scale image
```

导出工具新增两个抖动导出工具

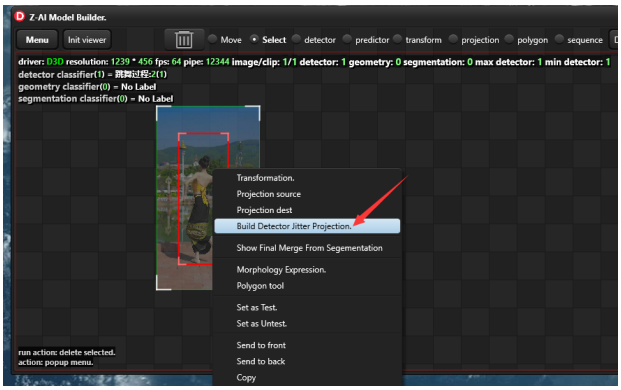


一个导出完整图片

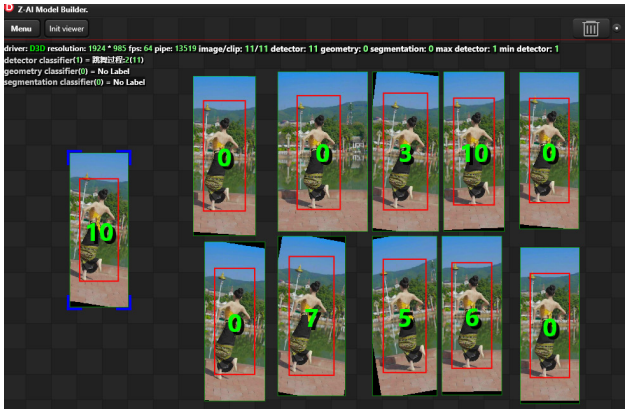
一个导出局部投影采样图片+抖动框,如果模型的训练算法不支持抖动,例如 OD/Metric,该功能可以丰富训练模型的准确率



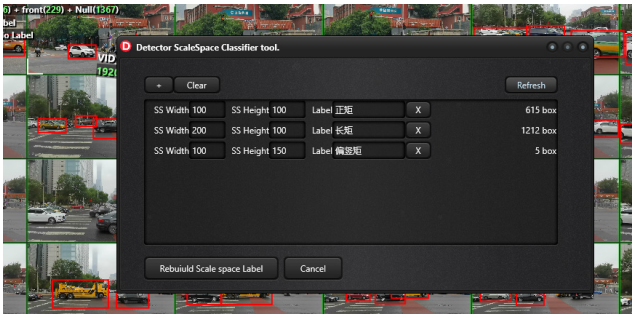
新增抖动样本投影功能,立即投影形式



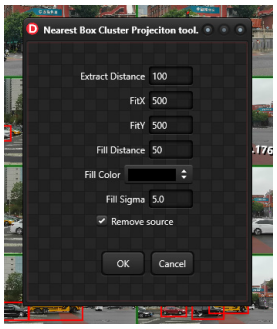
被投影以后,增加可变性,可用于不支持抖动的模型



新增相对尺度分类工具



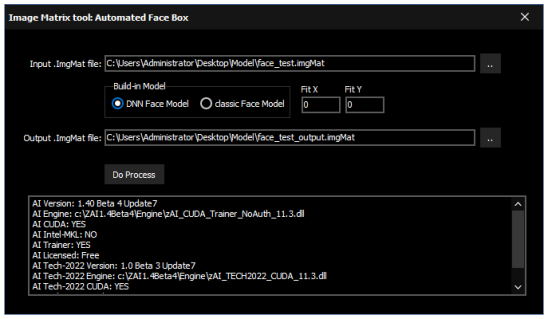
新增群聚投影

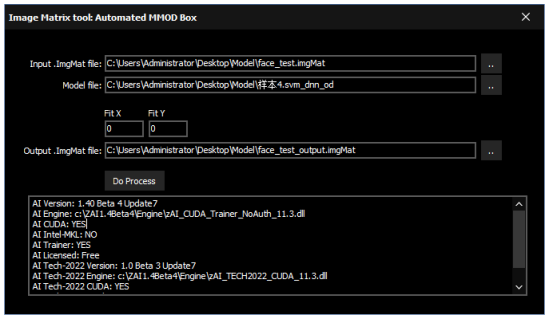


新增场景模型的测试工具,

从 **ImageMatrix tool** 移除内置标注,并新增若干外部标注工具

未来会新增更多针对.ImgMat 格式的标注工具

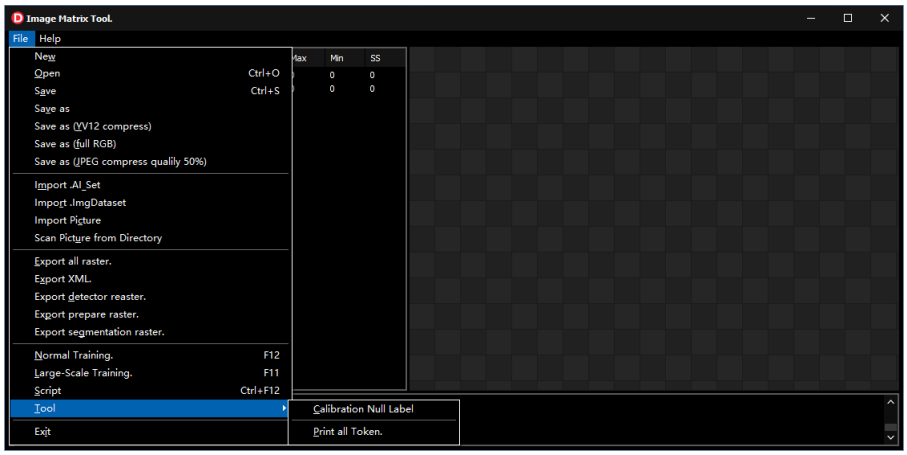




从 Image Matrix 移除所有自动化标注功能

自动化标注并不适用于大规模数据,标错根本找不到原因,另一方面,大规模数据标注必须考虑多 GPU 性能模式.最后,需要解决既能标注也能把三方的大数据导进来.

新的标注体系正在考虑设计.自动标注和大数据导入会在下个版本给出.



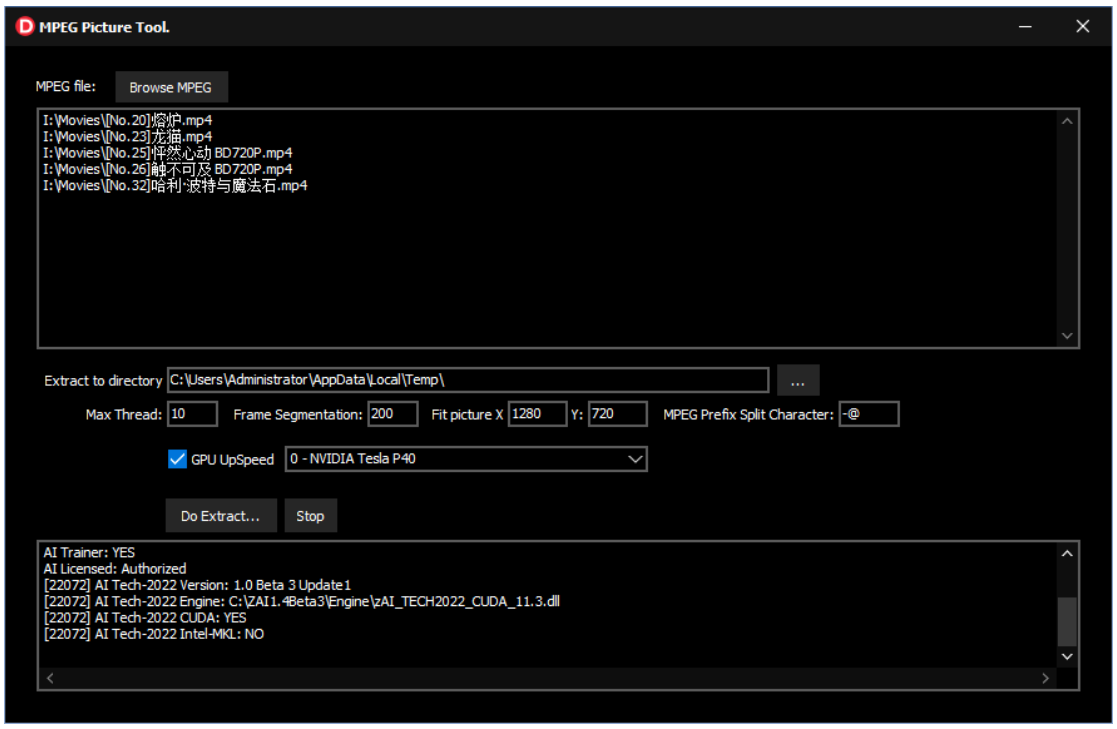
新增大规模图片分类工具

基于场景模型将紊乱的图片分类到各自目录,并且一致化图片的分辨率和图片编码格式.
建议使用大显存+多 gpu 运行

新增视频内容提取工具

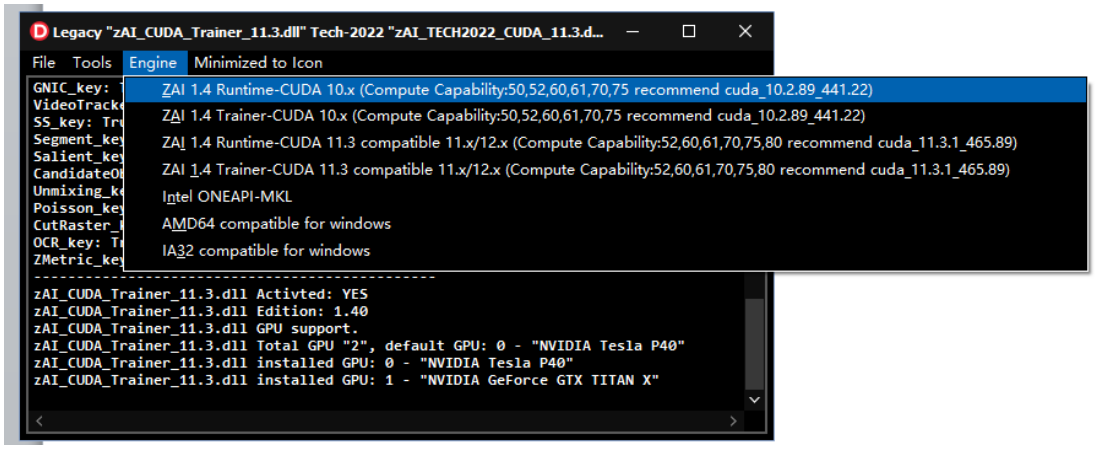
无论 4k/2k/1080p/720 它可以统一化尺度,并且在对应目录生成图片序列,可以作为场景模型的数据来源工具.

gpu 加速问题,注意:**某些游戏 gpu 有解码器会话限制,解决办法要么打 unlock 补丁,要么不使用 gpu 加速,M/T/V/P/A/H 系的 gpu 没有会话限制,某些 A/H 系 gpu 没有视频解码器**.



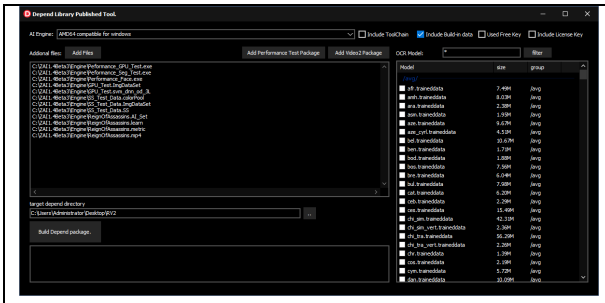
重构 AI 计算引擎

计算引擎从 gpu 到 cpu 总共 7 套,详见" GPUS Compute Capability.htm"

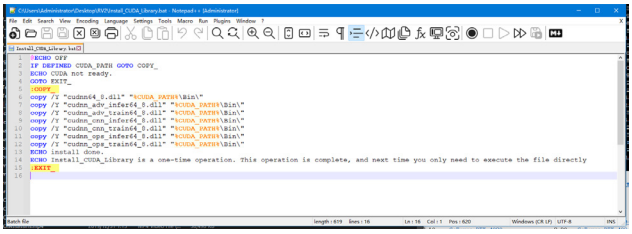


升级 AI 应用发行工具

针对 GPU 性能测试工具,监控包,新增一键功能
每次退出都会保存参数
生成的 AI 应用发行包支持 GPU 部署脚本:只要服务器或 pc 安装 cuda 它都能自动完成部署

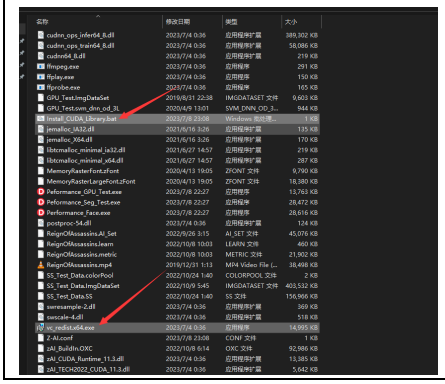


如果 AI 计算引擎是 GPU 类,它会生成一个安全部署脚本,该脚本是一次性的部署.

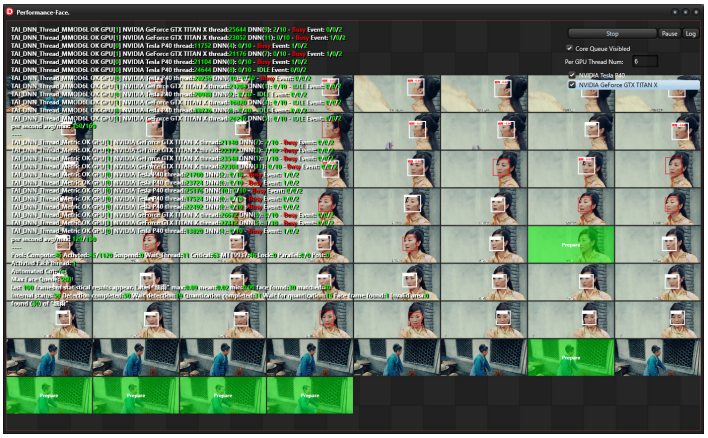


自动部署 vc 运行环境,如果服务器是全新安装,可以运行

- Install_CUDA_Library.bat:安全部署 gpu 依赖
- vc_redist.x64.exe:部署 vc 运行环境

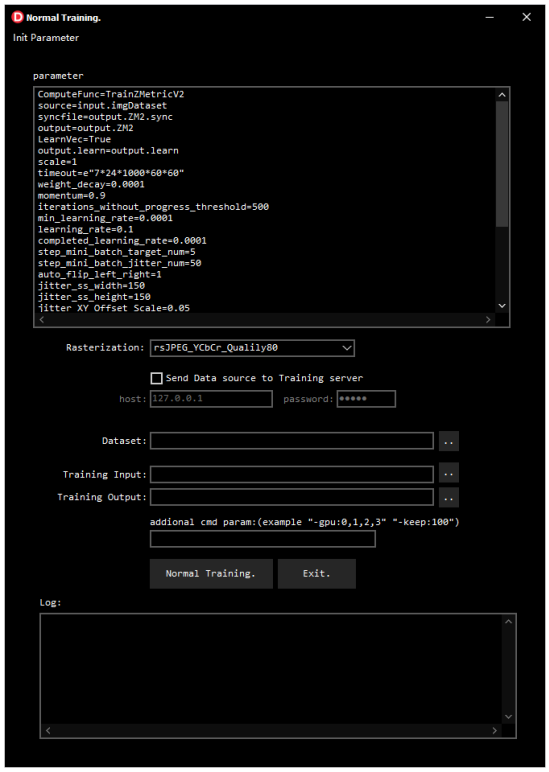
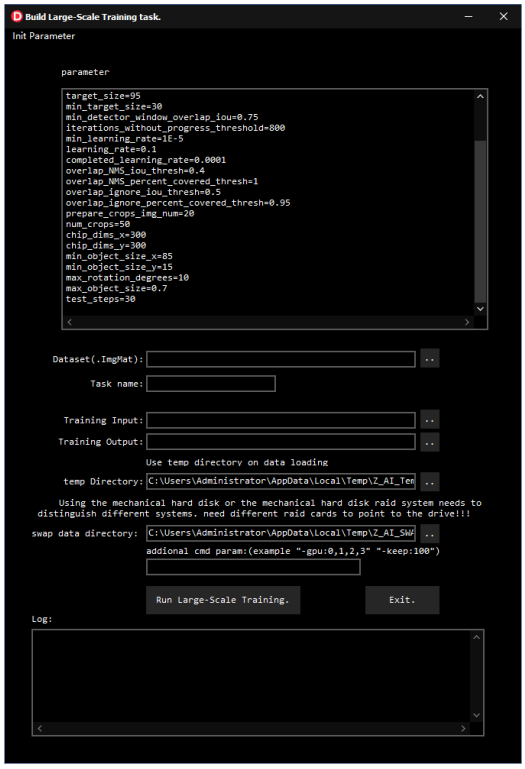


可以一键部署 gpu 测试工具,方便评估和测试服务器硬件



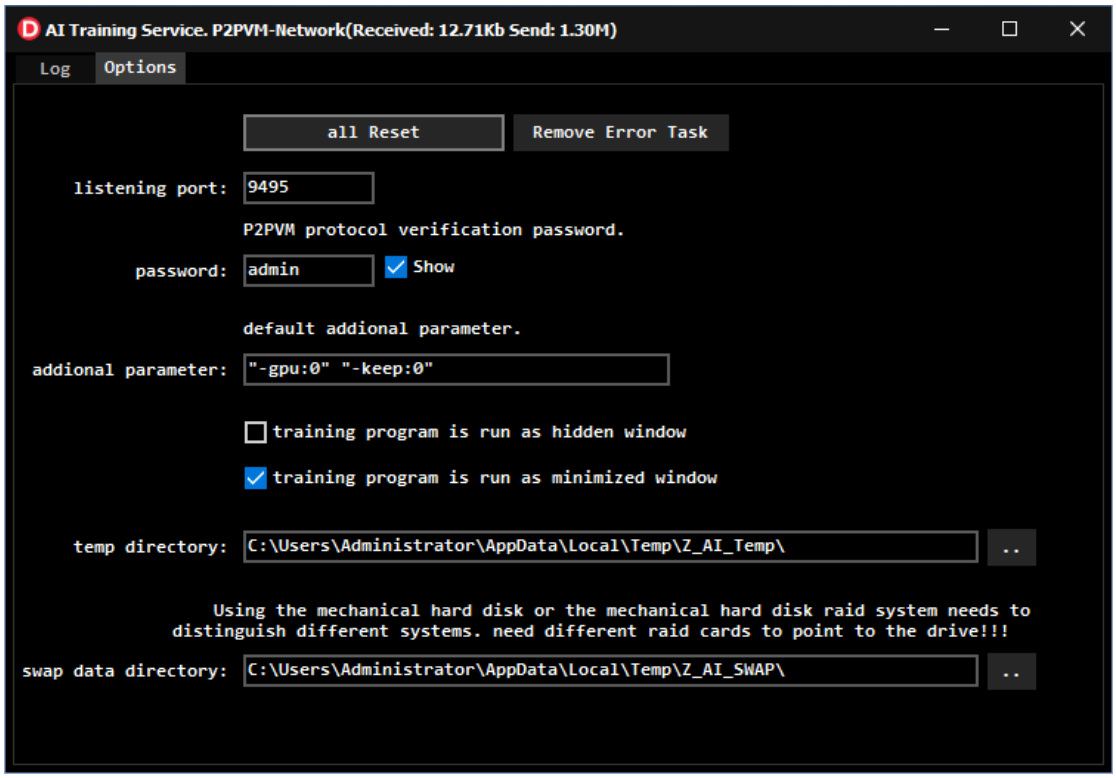
更新本地化训练启动器

启动器可以绕过训练服务器,直接启动训练任务,
启动器具备快速调度训练任务能力,可以快速调节参数,例如训练参数报警,这时候使用启动器来调节效率最高.提示:训练启动器不支持中继训练.

普通训练任务的启动器已经支持 ZMetricV2.0 模型体系.	Large-Scale 大数据训练器目前暂时不支持 ZMetricV2.0 模型.
	

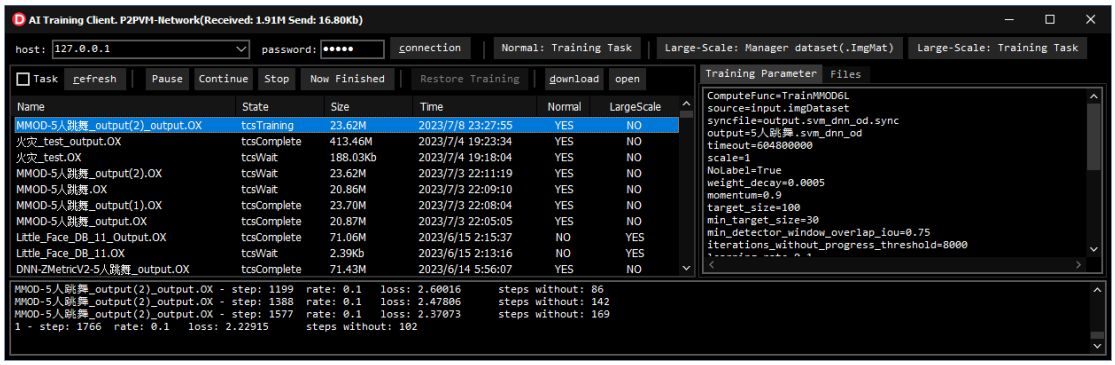
更新模型训练服务器

得益于底层 ZNet 体系升级,训练服务器数据传输能力提高.
服务器框架基本无改.

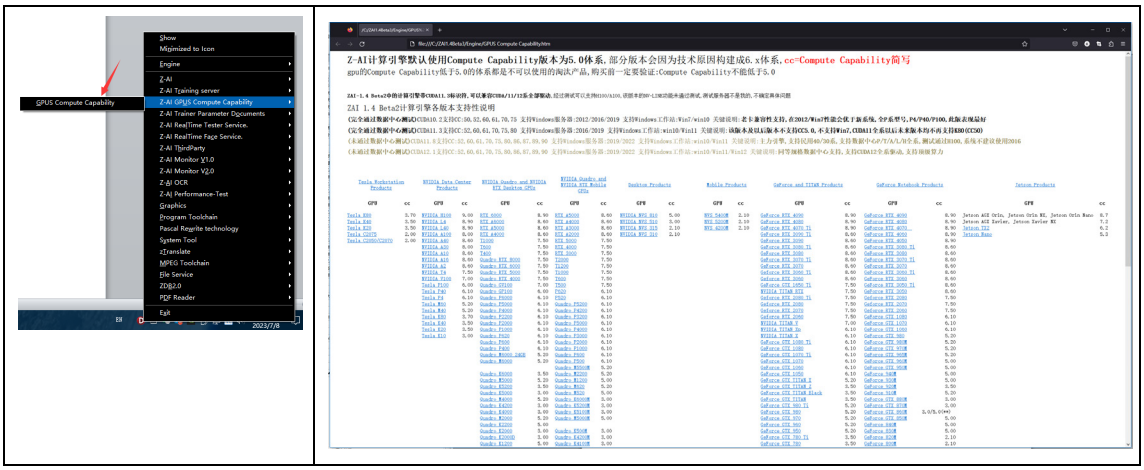


更新模型训练客户端

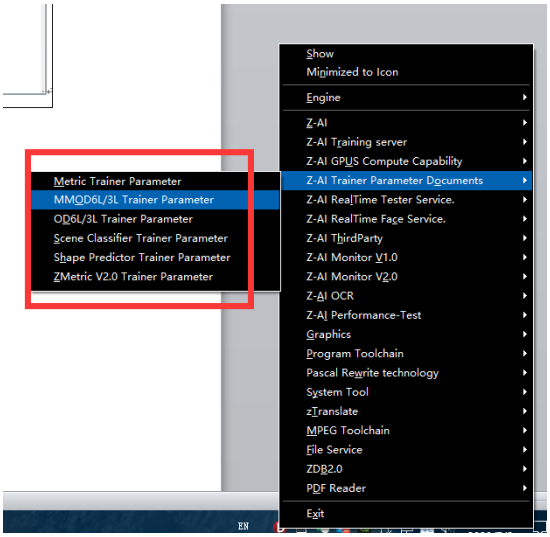
修复老版本一直遗留的自动连接问题:首次链接失败会尝试重连,持续 30 秒.
给了一个快捷链接历史 combo,以方便浏览多服务器



新增 GPUS Compute Capability 数据对照表(html 格式)

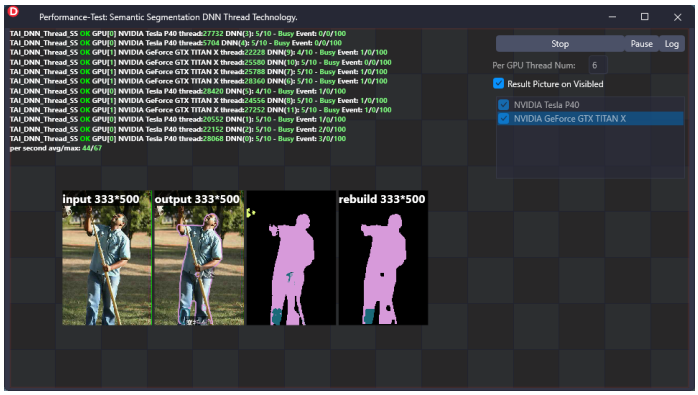


新增训练参数快捷参照文档

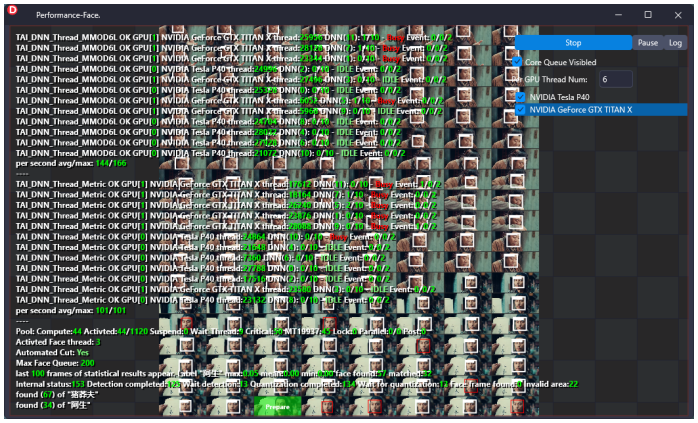


新增 3 套 gpu 服务器硬件测试程序

语义分割测试程序,综测 cpu+内存+gpu+显存,多路服务器可 numa 分路烤机测试



人脸测试程序，综测 cpu+gpu,可测 cpu+gpu 烤机



纯 gpu 测试程序,可测 gpu 烤机

AI 平台体系

基本未更新 RealTime Tester

简称 RTT,多用于观看 AI 效果,测试模型使用.

项目框架是按一个引擎设计,如果接入 Tech2022-ZMetricV2.0 需要框架支持双引擎, 完成框架改造前,RTT 不支持 Tech2022 方向的模型.

得益于底层 ZNet 大更新,享受网络支持优化.

基本未更新 Face3.0(人脸 3.0)

Face3.0 之前经历 3 次推翻重构型设计,2020 版本已具备候选+视频人脸识别能力,把 Tech2022-ZM2 引入并不会提升效果.ZM2 只能在人脸对齐失效的情况下做一些候选处理,例如大侧面.**在正脸领域,Face3.0 从算法到框架并不需要升级**.

得益于底层 ZNet 大更新,享受网络支持优化.

基本未更新 ThirdParty

ThirdParty 是作为三方 AI 识别接口直接支持 c++/java/c#/swift/web-js 的体系,目前为裸流 CS 方式通讯.需要接入端自行解决网络通讯的粘包机制.

底层视频光栅队列支持库(Z.AI.VideoRaster.DNNQueue 库,非 DNN-Thread)已完成 ZM2 接入,应用层未接入:包含 ThirdParty 服务器+前端辅助开发工具+api 接口

ThirdParty 在未来如果升级事件,会直接以 http 协议来替代底层粘包,降低接口复杂度.

得益于底层 ZNet 大更新,享受网络支持优化.

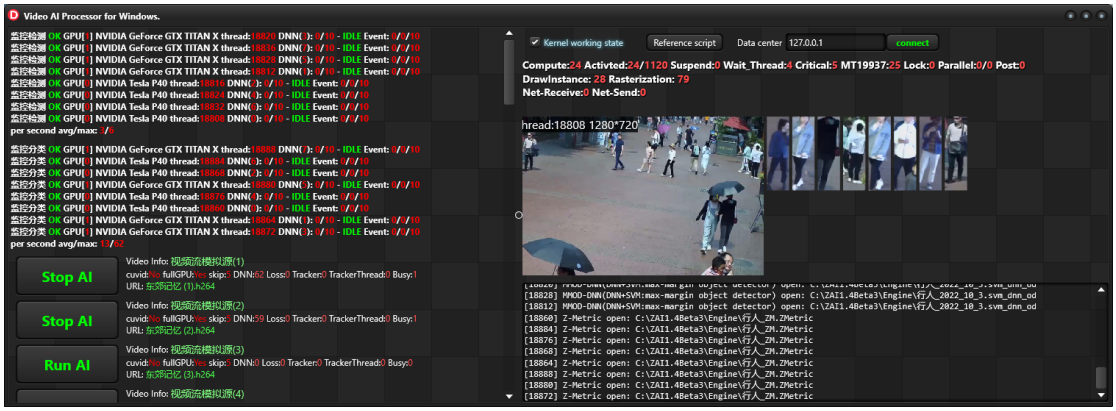
大幅更新 RealTimeVideo V1,非 6 代监控

简称 1 代,我们监控项目在开发之初就是打算在 1 代基础上做,大约 1 个月以后,6 代监控被开新.

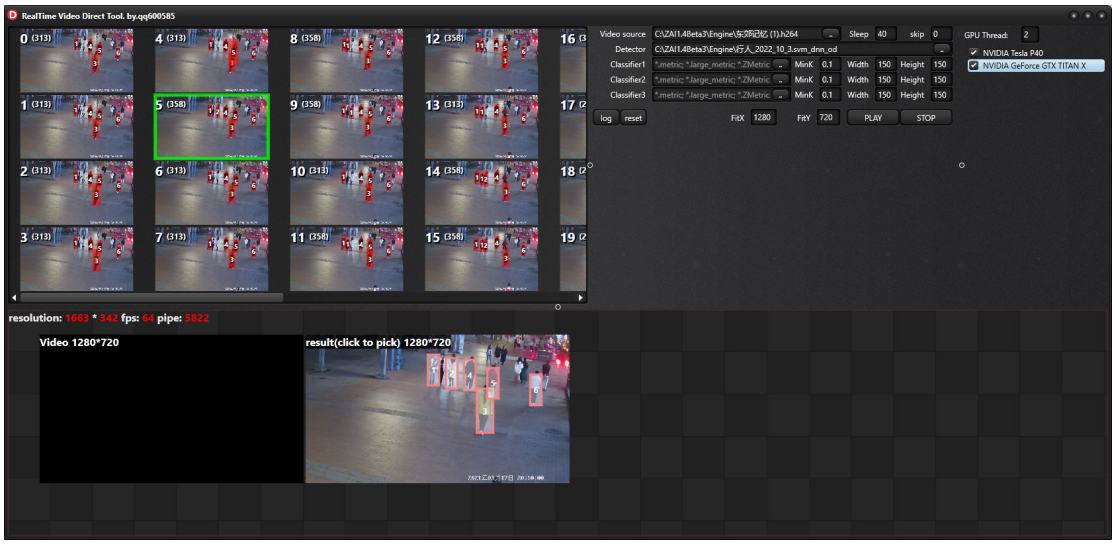
开新的原因是对 AI 监控的理解:AI 监控是一种应用,需要考虑 AI 识别与别的监控系统整合,然后体现在用户层,这会让你的站位很难确定,你到底是吃时代红利搞 AI 还是搞监控,如果搞 AI 那么 1 代已经足够了,开发方向就是让用户感受到识别,然后找他们收钱,如果搞监控那就另开一个项目并启动新的方向.

在 1 个月以后,我们做出了选择,这是显而易见的.

1 代的 ap 服务器



1 代的模型工具,可以从配对结果看出,1 代已经开始支持行为识别流程



第 6 代监控

6 代监控是我们在已建成数据中心(省市级单位)环境下设计出来的, 刚好 idc 成规模的运营了海康,大华,宇视的核心产品.这些公司的业务领域可以说遍布全世界,他们代表了世界上最强大的监控公司,这些监控的产品从设计到应用也都非常成熟.

我们的项目定位是对已有监控增加 AI 能力,在设计和思路层面,IDC 的全部监控系统从基建到后台再到前端,我们都认真研究过,并且,反复思考.刚好项目有时间条件+资源条件,于是,我们就做了自己的 6 代监控.

大公司只能使用已有人力资源进行研发,在时间推进作用下,这可以将一些领域做的很广阔,很深入,但这无法阻止通过技术+创新从另外一个维度切入.

海康,大华,宇视,非常注重核心嵌入式技术体系,系统和软件的建设略略偏外围,无限堆人模式不会有颠覆式创新.例如研究院方向 30 万,外围方向普遍 10 万(低于普遍,比外卖小哥略高,和滴滴持平).

颠覆式创新是种淡化金钱和权力并且影响世界变化的技术进步,如果用一种上市公司研究院做不到的方式,从技术面突破是可以有成功几率的.

6 监控的体系非常庞大,从 AI 生产技术,到后台数据库, gpu,再到 webapi,监控前端.

因为 6 代还没完整的竣工,此处细节现在暂时不便,未来会独立撰写.现在只是发 ZAI 版本,简单的提一下 6 代就够了.

1.4 的版本分类

标识符 OZAI=开源版本,LZAI=授权版本, ??ZAI=私有化版本
YES=明确开放,NO=不开放,n/a=定制化授予
注:开源版本会在计算引擎移除授权码,开源用户会无法授权版和私有化版.细节独撰文档.

种类	开源版	授权版	私有化版
AI 引擎源码	YES	YES	YES
AI 计算引擎源码	NO	NO	n/a
Model Builder 工具	YES	n/a	YES
Model Builder 源码	NO	YES	YES
Image Matrix 工具	YES	YES	YES
Image Matrix 源码	NO	n/a	YES
完整工具链	NO	YES	YES
完整工具链源码	NO	NO	n/a
程序员工具链	YES	YES	YES
程序员工具链源码	YES	YES	n/a
6 代监控	NO	NO	n/a
6 代监控源码	NO	NO	n/a
完整 Demo	YES	YES	YES
RealTime Tester	NO	YES	YES
RealTime Tester 源码	NO	YES	n/a
Face3.0	NO	YES	YES
Face3.0 源码	NO	n/a	n/a
1 代监控和源码	NO	YES	YES
ThirdParty	NO	YES	YES
ThirdParty 源码	YES	n/a	n/a

发行方式

重做工具链安装系统流程.
网站下载+百度网盘,百度网盘实测在美国已可直访
1.4 首发版不会使用 ZLaunch 平台.