1.30 更新日志

目录

计算引擎算法更新 3
Efficient Graph-Based Image Segmentation 3
Minimum Barrier Salient Object Detection at 80 FPS 4
Segmentation as Selective Search for Object Recognition 5
Unmixing-Based Soft Color Segmentation for Image Manipulation 6
Poisson Image Editing 7
GrabCut — Interactive Foreground Extraction using Iterated Graph Cuts 8
Tesseract OCR 9
计算引擎支持架构更新 10
基础库更新 11
CoreClasses.pas 库更新(编译器与平台兼容) 11
DoStatusIO.pas 库更新(编译器与平台兼容) 11
Geometry2Dunit.pas 库更新(编译器与平台兼容) 11
ListEngine.pas 库更新(编译器与平台兼容) 11
MemoryStream64.pas 库更新(编译器与平台兼容) 12
NotifyObjectBase.pas 库更新(编译器与平台兼容) 12
UnicodeMixedLib.pas 库更新(编译器与平台兼容) 12
UpascalString.pas/PascalString.pas 库更新(编译器与平台兼容) 12
TextParsing.pas 库更新(编译器与平台兼容) 12
OpCode.pas 库更新(编译器与平台兼容) 12
zExpression.pas 库更新(编译器与平台兼容) 13
ZDB 相关库更新 13
ItemStream.pas 库更新(编译器与平台兼容) 13
ObjectData.pas 库更新(编译器与平台兼容) 13
ObjectDataManager.pas 库更新(编译器与平台兼容) 13
光栅库更新 14
MemoryRaster.pas 库更新(编译器与平台兼容) 14
MemoryRaster_DocumentTextDetector.pas 库更新(编译器与平台兼容) 14
PictureViewerInterface.pas 库更新(编译器与平台兼容) 15
zDrawEngine.pas 库更新(编译器与平台兼容) 15
MorphologyExpression.pas 库更新(编译器与平台兼容) 15
PyramidSpace.pas 库更新(编译器与平台兼容) 15
FastHistogramSpace.pas 库更新(编译器与平台兼容) 15
FMXCharacterMapBuilder.pas 库更新(只能在 delphi 使用) 15
视频支持库更新 16
FFMPEG_Reader.pas 库更新(需要外部库支持,兼容 FPC+IOT,Delphi+移动平台) 16
FFMPEG_Writer.pas 库更新(需要外部库支持,兼容 FPC+IOT,Delphi+移动平台) 16
FFMPEG.pas 库更新(需要外部库支持,兼容 FPC+IOT,Delphi+移动平台) 16

自然语言支持库更新 16
GBKMediaCenter.pas 库更新 16
GBK.pas 库更新 16
FastGBK.pas 库更新 16
Z-AI 支持库更新 17
zAI.Pas 库更新 17
zAI_Common.pas 库更新 17
zAI_Editor_Common.pas 库更新 17
Z-AI Free Demo 更新 18
Rasterizatio Format Demo 更新 18
CS 响应式人脸入库/人脸识别 Demo 更新 18
DNN_OD demo 更新 19
FFMPEG_ReaderDemo 更新 19
memoryRasterProjection demo 更新 20
对齐线检测 demo 更新 21
sigmaGaussian demo 更新 21
TrainingTool.exe 更新 22
DrawEngineFontEdge demo 更新 22
RectRotationProjection demo 更新 22
Wildcard demo 更新 23
ParallelDemo 更新 23
ApproximatePolygon demo 更新 24
RotationDemo 更新 24
PolygonRegion demo 更新 25
MorphGalaxiesDetection demo 更新 25
DocumentFilter demo 更新 26
授权 Demo 更新 27
RoadCameraDetector demo 更新 27
ScheduleDrawingExtract demo 更新 27
SemanticSegmentationTraining demo 更新 28
OCRHelloWorld demo 更新 29
视频推流组合 demo 30
实时识别视频组合 demo 32
人证识别 demo 34
引擎工具链更新(省略,大规模更新,参考独立文档) 35

计算引擎算法更新

Efficient Graph-Based Image Segmentation

paper url https://wenku.baidu.com/view/f21203d726fff705cc170a33.html
post by.2004
author:

- Pedro F. Felzenszwalb, Artificial Intelligence Lab, Massachusetts Institute of Technology,pff@ai.mit.edu
- Daniel P. Huttenlocher,Computer Science Department, Cornell University,dph@cs.cornell.edu

该算法实现了广义上按颜色进行梯度分割,效果不理想,远不如 MemoryRaster 内置的 TMorphologySegmentation

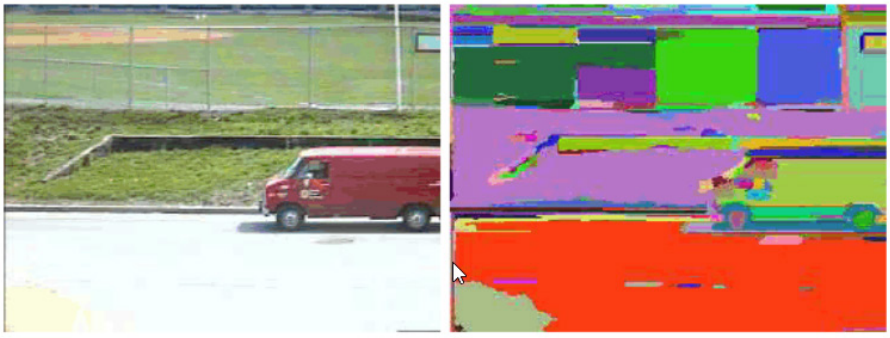


Figure 2: A street scene (320×240 color image), and the segmentation results produced by our algorithm ($\sigma = 0.8$, $k = 300$).

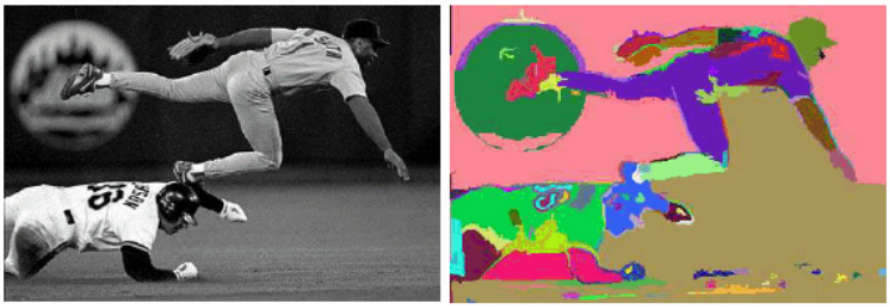


Figure 3: A baseball scene (432×294 grey image), and the segmentation results produced by our algorithm ($\sigma = 0.8$, $k = 300$).

Minimum Barrier Salient Object Detection at 80 FPS

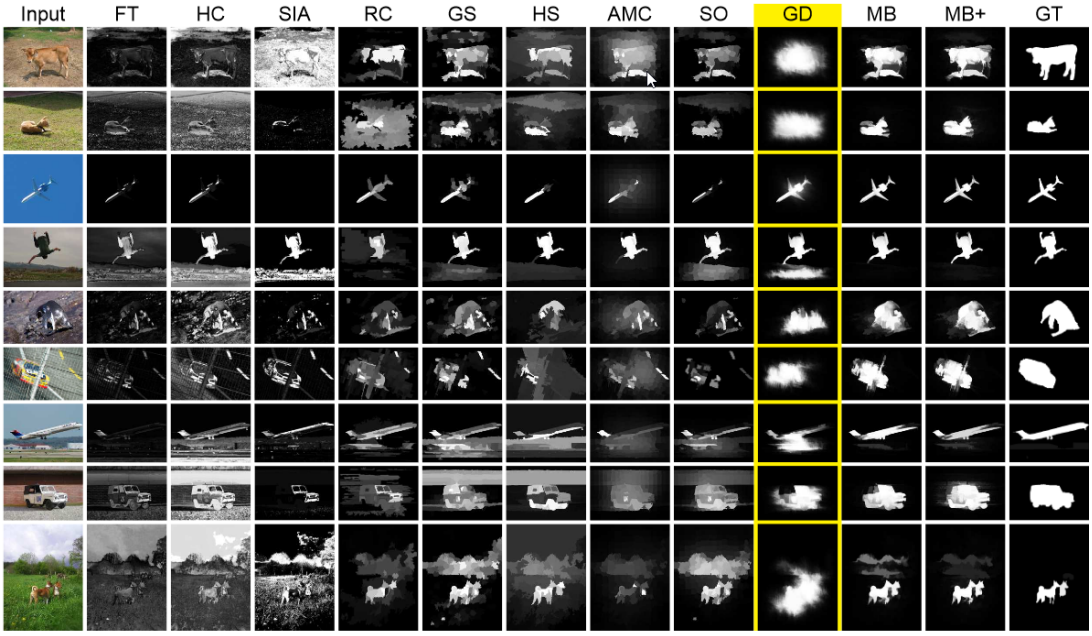
Paper url http://openaccess.thecvf.com/content_iccv_2015/papers/Zhang_Minimum_Barrier_Salient_ICCV_2015_paper.pdf

post by 2015

author

- Boston University: Jianming Zhang1
- Boston University: Stan Sclaroff1
- Adobe Research:Zhe Lin2
- Adobe Research:Xiaohui Shen2
- Adobe Research:Brian Price2
- Adobe Research:Radomir Mech2

该算法实现了目标显著性的灰度转换,通过结合形态学,可以增强样本提炼的效果,目前尚未集成到引擎与工具链中,只可以通过 ZAI 的 api 来使用.在后续的发行版本中会应用到它.



Segmentation as Selective Search for Object Recognition

Paper url https://www.koen.me/research/selectivesearch/

post by 2011

Abstract

For object recognition, the current state-of-the-art is based on exhaustive search. However,
to enable the use of more expensive features and classifiers and thereby progress beyond the state-of-the-art,
a selective search strategy is needed. Therefore,

该算法实现了候选分类框,它会将所有可能出现的分类框全部罗列出来.然后再依靠 SVM 对框内容进行图像分类.
该算法我只验证过它的可提取过程,并没有按照后续思路执行它的分类计算.我感觉这是一种不错的候选检测器实现思路.于是在 1.3 将它引入计算引擎,在后面版本来应用它.

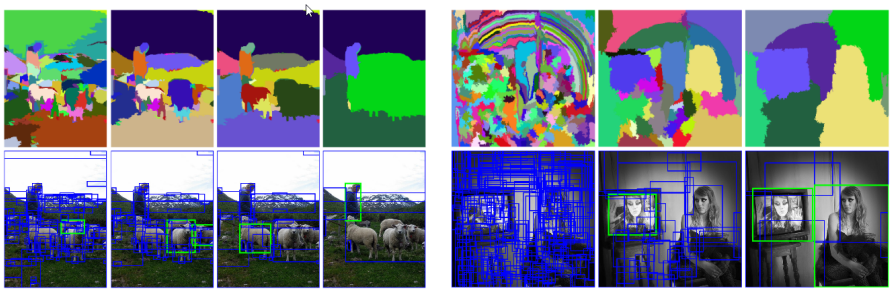


Figure 3. Two examples of our hierarchical grouping algorithm showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.

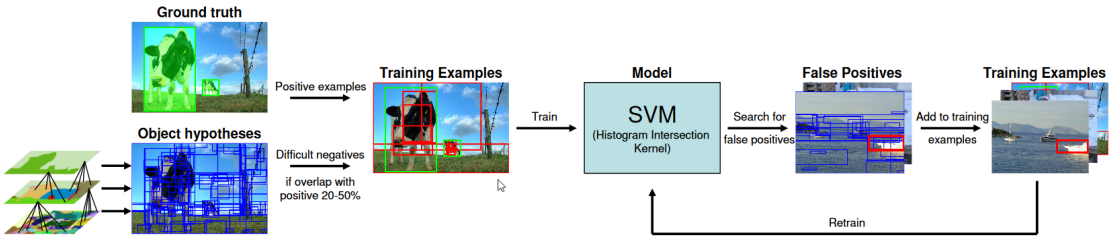


Figure 4. The training procedure of our object recognition pipeline. As positive learning examples we use the ground truth. As negatives we use examples that have a 20-50% overlap with the positive examples. We iteratively add hard negatives using a retraining phase.

Unmixing-Based Soft Color Segmentation for Image Manipulation

url https://cvg.ethz.ch/research/soft-color-segmentation/

post by 2017

reference material http://staff.ustc.edu.cn/~zhuang/acg/SIGGRAPH-2017-papers.pdf

该算法实现 alpha 的反混合解算,等同于用猜测法将照片反解算成带有分层效果的.psd 数据(photoshop 文件).

该算法对计算机的运算能力要求非常高,需要超级算力支持.

2 • Y. Aksoy et al.

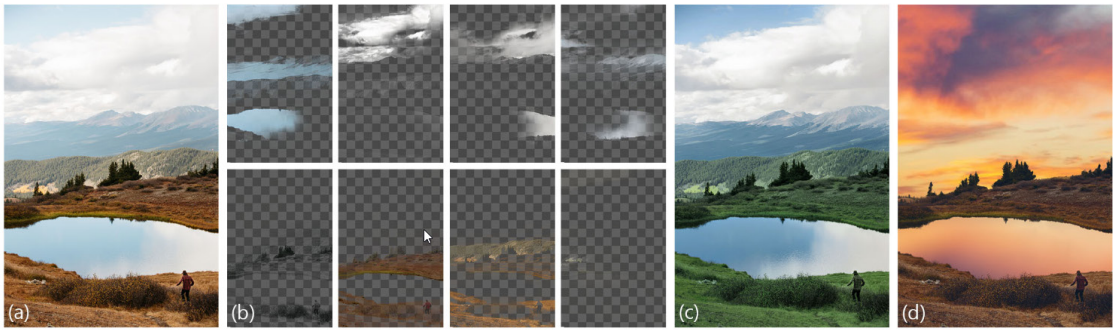


Fig. 1. Our method automatically decomposes an input image (a) into a set of soft segments (b). In practice, these soft segments can be treated as layers that are commonly utilized in image manipulation software. Using this relation, we achieve compelling results in color editing (c), compositing (d), and many other image manipulation applications conveniently under a unified framework.

4 • Y. Aksoy et al.

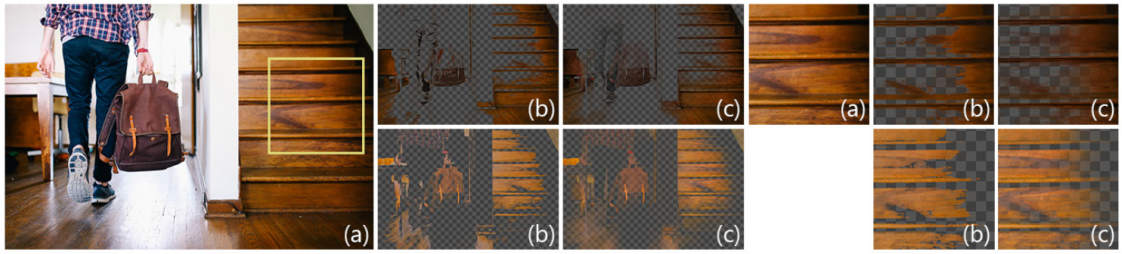


Fig. 3. Two layers corresponding to the dark (top) and light wood color in the original image (a) are shown before (b) and after (c) matte regularization and color refinement.

Poisson Image Editing

Paper url http://www.cs.virginia.edu/~connelly/class/2014/comp_photo/proj2/poisson.pdf

post by 2014

author

- Patrick Perez, Michel Gangnet, Andrew Blake
- Microsoft Research UK

该算法主要是由微软研究院针对老泊松混合公式重构而出的新的微分计算方式.目前应用在样本融合功能中.

我们也可以通过计算引擎直接调用 Poisson 进行图像融合.

在计算引擎中,Poisson 分为经典融合(opencv 内置的 Poisson 算法)与 Paper 融合(微软研究的新积分算法)

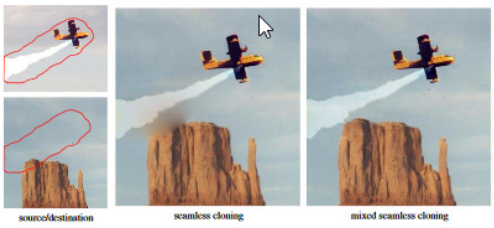


Figure 8: **Inserting one object close to another**. With seamless cloning, an object in the destination image touching the selected region Ω bleeds into it. Bleeding is inhibited by using mixed gradients as the guidance field.

Figure 9: **Texture flattening**. By retaining only the gradients at edge locations, before integrating with the Poisson solver, one washes out the texture of the selected region, giving its contents a flat aspect.

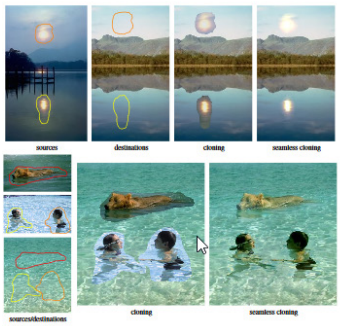


Figure 3: **Insertion**. The power of the method is fully expressed when inserting objects with complex outlines into a new background. Because of the drastic differences between the source and the destination, standard image cloning cannot be used in this case.

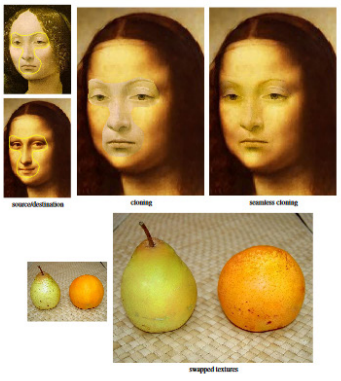


Figure 4: **Feature exchange**. Seamless cloning allows the user to replace easily certain features of one object by alternative features. In the second example of texture swapping multiple broad strokes (not shown) were used.

The discrete counterpart of this guidance field is:

Figure 5: **Monochrome transfer**. In some cases, such as texture transfer, the part of the source color remaining after seamless cloning might be undesirable. This is fixed by turning the source image monochrome beforehand.

Figs. 6 and 7.

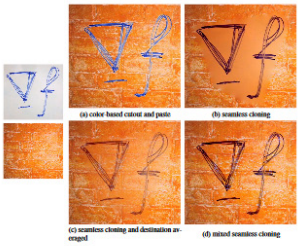


Figure 6: **Inserting objects with holes**. (a) The classic method, color-based selection and alpha masking might be time consuming and often leaves an undesirable halo; (b-c) seamless cloning, even averaged with the original image, is not effective; (d) mixed seamless cloning based on a loose selection proves effective.

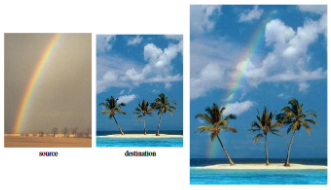


Figure 7: **Inserting transparent objects**. Mixed seamless cloning facilitates the transfer of partly transparent objects, such as the rainbow in this example. The non-linear mixing of gradient fields picks out whichever of source or destination structure is the more salient at each location.

GrabCut — Interactive Foreground Extraction using Iterated Graph Cuts

Paper url http://www.cvg.ethz.ch/teaching/cvl/2012/grabcut-siggraph04.pdf
post by 2004
Published in: international conference on computer graphics and interactive techniques · 2004
Authors: Carsten Rother · Vladimir Kolmogorov · Andrew Blake
Affiliation: Microsoft
该算法为 opencv 现代方式实现,已经在计算引擎中集成,目前应用于样本自动化分割建模功能.
该算法需要交互性质输入只能作用于工具,无法自动化完成工作,目前被自动化的语义分割全面取代.在工具领域该算法仍然由一席之地.

摘自知乎,关于 GrabCut
GrabCut 在 graph cut 基础之上的改进包括：将基于灰度分布的模型替换为高斯混合模型（Gaussian Mixture Model，GMM）以支持彩色图片；将能一次性得到结果的算法改成了『强大的』迭代流程；将用户的交互简化到只需要框选前景物体即可。我觉得，核心的改变只有一点，就是支持彩色图片，其他的调整都是顺势而为，是被动的调整。

出发点是大量的彩色图片的分割需求，那么，就需要把判断像素属于前景/背景的公式从简单的使用灰度 histogram 扩展为一种能支持彩色的模型。彩色像素值的稀疏问题比灰度图要严重得多（256 vs 17M），所以，继续使用 histogram 是不现实的，需要信息压缩得更好一点的模型，作者在这里参考前人，对前景和背景各建了 K=5 的高斯混合模型。

但 GMM 模型也需要一定量的数据才能得出有意义的概率。原始方法中，用户会用有宽度的笔刷初步标记前景/背景区域，要是刷得长一点粗一点，一刷子也能有几百像素，在灰度模型中已经足够用了，但要换成彩色图，需要把前景/背景中的代表色都给刷上，难度就高了些，用户估计也理解不了。也就是说，GMM 需要足够多的数据来得到靠谱的参数，画线是不够的，得标注成区域才能满足模型需要。

这也就是论文中『框住前景』处理的讨巧之处。『画框』看似在标记前景物体，但其实得到了大范围而且确定的背景区域。根据这些确定的背景区域，能够得到比较准确的背景 GMM。相比之下，前景的 GMM 要低效得多，因为用来出前景 GMM 的数据里包含了很多背景 GMM 的数据。

如果这样能得到比较好的结果，事情就结束了，但估计结果好不到哪儿去。主要原因推测是：GMM 模型本来就是压缩得比较厉害的模型，这样的模型还没有精确点的参数，以此推算出的概率实在不足为信。那按正常的思路，接下来就应该想方法得到更精确的 GMM 参数。论文中也是这样做的：既然前景 GMM 参数的主要问题是混杂了背景的信息，那就尽可能把背景去掉后再重算 GMM 参数，而区分前景背景本来就是我们的目的，这就成了论文中的『更新 GMM—更新分割』的迭代流程，由于计算 GMM 参数也还是有些计算量的（跟 k-means 的迭代更新类似，），所以是每完整更新一次最小割、更新一次参数。这样做也是有理论支持的，因为仍旧满足每次更新，能量减少的约束，结果可以收敛。

因为 GrabCut 是按颜色分布和边缘对比度来分割图片的，对一些常见的与此原则相悖的图片，效果确实不好。比如前景人物的帽子、鞋、墨镜，通常颜色跟前景主体有较大区别；再如前景中的孔，有可能由于颜色区分和边缘的对比度不足，导致边缘的惩罚占上风，而没有扣出来背景。所以，GrabCut 还是保留了人工修正的操作，定义了两种标记：绝对是背景和可能是前景。对分割错误人工修正后，分割还是可以比较准确的。

Tesseract OCR

Document url https://github.com/tesseract-ocr/docs
Project url https://github.com/tesseract-ocr/tesseract
内置集成的 tesseract 是重构版本,处于兼容性和易编译性,改动比较多.对授权用户提供标准 API 和入门 demo.
如果是 zOCR 用户,提供 c++端的傻瓜构建包,可自行构建 zOCR 的动态库.标准 ZAI 授权只有基本 API,不提供专业 OCR 解决方案.

Paper author Ray Smith

Ray Smith 是一个专注 OCR 检测识别 30 年的老头.下列文章是关于 Tesseract 在切字领域的算法方案.我在 MemoryRaster 光栅系统内置的 TextDetector 参考过这篇 paper.在经过深入思考和提炼以后,内置库中的 TextDetctor 简洁性和可读性目前优于这篇 paper.
Primary Paper url https://github.com/tesseract-ocr/docs/blob/master/tesseracticdar2007.pdf

paper 摘要
Abstract

The Tesseract OCR engine, as was the HP Research Prototype in the UNLV Fourth Annual Test of OCR Accuracy[1], is described in a comprehensive overview. Emphasis is placed on aspects that are novel or at least unusual in an OCR engine, including in particular the line finding, features/classification methods, and the adaptive classifier.

1. Introduction – Motivation and History

Tesseract is an open-source OCR engine that was developed at HP between 1984 and 1994. Like a supernova,it appeared from nowhere for the 1995 UNLV Annual Test of OCR Accuracy [1], shone brightly with its results, and then vanished back under the same cloak of secrecy under which it had been developed.

Now for the first time, details of the architecture and algorithms can be revealed.

Tesseract began as a PhD research project [2] in HP Labs, Bristol, and gained momentum as a possible software and/or hardware add-on for HP's line of flatbed scanners. Motivation was provided by the fact that the commercial OCR engines of the day were in their infancy, and failed miserably on anything but the best quality print.

After a joint project between HP Labs Bristol, and HP's scanner division in Colorado, Tesseract had a significant lead in accuracy over the commercial engines, but did not become a product. The next stage of its development was back in HP Labs Bristol as an investigation of OCR for compression.

Work concentrated more on improving rejection efficiency than on base-level accuracy. At the end of this project,at the end of 1994, development ceased entirely. The engine was sent to UNLV for the 1995 Annual Test of OCR Accuracy[1], where it proved its worth against the commercial engines of the time. In late 2005, HP released Tesseract for open source. It is now available at http://code.google.com/p/tesseract-ocr.

篇幅原因不做更多介绍,请在 Document url https://github.com/tesseract-ocr/docs 自行了解.

计算引擎支持架构更新

1.30 针对计算引擎全部做过重构,详情请参考下图

ZAI计算引擎支持表

引擎名称	CUDA	授权验证	SSE2	AVX2	AVX512	训练	并行训练	并行调用	线程支持	训练性能	64位	32位
zAI_Cuda10.2_Free.dll	Yes	No	Yes	Yes	No	No	No	No	No	n/a	Yes	No
zAI_Cuda10.2.dll	Yes	Yes	Yes	Yes	No	Yes	No	No	No	快	Yes	No
zAI_MKL64_Parallel.dll	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	慢	Yes	No
zAI_MKL32_Parallel.dll	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	慢	No	Yes
zAI_x64.dll	No	Yes	Yes	Yes	No	No	No	Yes	Yes	极慢	Yes	No
zAI_x64_Free.dll	No	No	Yes	Yes	No	No	No	Yes	Yes	n/a	Yes	No
zAI_x86_free.dll	No	No	Yes	Yes	No	No	No	Yes	Yes	n/a	No	Yes

32位由于不支持2G以上的内存分配，它顶天只能跑过程，无法运行建模，32位程序API提供建模功能，但是它受小内存限制。MKL支持建模，但是效率非常低，MKL是CUDA的移植支持。支持AVX512的CPU与CUDA性能差距10倍以上。建模程序只能用支持CUDA引擎计算。

授权验证:链接 licensed server 验证,从上图我们可以看大,只有两个计算引擎会连接服务器.
现在我们可以直接部署无验证的计算引擎,它没有任何限制.

基础库更新

CoreClasses.pas 库更新(编译器与平台兼容)

- 移除了对于 Delphi 标准库 Threading 的依赖性,改用 CoreThread 库来支持并行与线程池
- 移除了对于 FPC-LCL 标准库 Procs 的依赖性,改用 CoreThread 库来支持并行与线程池
- TCritical 硬件互斥接口,新增获取锁状态方法
- 新增原子锁的泛型变量操作 TatomVar
- 新增 FPC 编译器专属并行 API 支持 FPCParallelFor,回调接口为使用 nested 修饰的 Local Procedure
- 新增 Delphi 编译器专属并行 API 支持 DelphiParallelFor,回调接口为使用 reference 修饰的 anonymous Procedure
- 新增 MT19937 随机数函数体系,包括数十个 API 与专属 TMT19937Random 类
- 重做核心线程池调度机制
- 新增 FPC/Delphi 通用性泛型支持 TlineProcessor<Type>类, TlineProcessor 本来应该放 MemoryRaster,由于 delphi Formater 关系会将 FPC 的修饰符 generic 做换行处理,所以 TlineProcessor 在永远不需要做 Formater 的 CoreClasses 里面
- 新增 if_函数,用于代替 Math 库的 ifthen 函数,该函数主要用于优化计算型程序的可读性,因为计算过程复杂度是可维护性的重要指标.

DoStatusIO.pas 库更新(编译器与平台兼容)

- 新增 FPC 编译器专属回调支持 AddDoStatusHookP,回调接口为使用 nested 修饰的 Local Procedure
- 新增 Delphi 编译器专属回调支持 AddDoStatusHookP,回调接口为使用 reference 修饰的 anonymous Procedure
- 重做多线程/并行程序的 StatusIO 接口,现在输出 Status 打印方式为[线程 ID 前缀]+Status,主线程中的 StatusIO 不会打印线程 ID 前缀
- 重做 DoStatusNoLn 机制,在多线程会对齐当前行

Geometry2Dunit.pas 库更新(编译器与平台兼容)

- 新增点积拓扑学算法 Hausdorf
- 修正源正反投影的公式函数
- 新增正反转动投影公式函数
- 新增若干小函数

ListEngine.pas 库更新(编译器与平台兼容)

- 由于 hash 数据有特殊性,无法暴力 hash,因此将 hash 分为很多场景,导致了 ListEngine 库非常大,上万行代码量,本次更新为全部折叠
- 新增若干小方法

MemoryStream64.pas 库更新(编译器与平台兼容)

并行化压缩/解压机制更新:移除 FPC 的 Procs 并行机制
并行化压缩/解压机制更新:移除 Delphi 的 TParallel 并行机制
并行化压缩/解压机制更新:使用 CoreClasses.pas 库内置并行机制实现压缩/解压缩,稳定性和性能优于前两者.

NotifyObjectBase.pas 库更新(编译器与平台兼容)

- 新增 FPC 编译器专属回调支持,回调接口为使用 nested 修饰的 Local Procedure

UnicodeMixedLib.pas 库更新(编译器与平台兼容)

- 新增若干小方法

UpascalString.pas/PascalString.pas 库更新(编译器与平台兼容)

- 新增大小写字符转换
- CharIn 函数新增可见字符条件判断,字符范围在$20-$7E
- 新增 RandomString 函数,该函数只会生成可见字符

TextParsing.pas 库更新(编译器与平台兼容)

- 修复 C 语言字符表达式的'char'问题
- 新增若干小方法
- 新增左右探头缩写化支持

OpCode.pas 库更新(编译器与平台兼容)

- 新增 FPC 编译器专属回调支持,回调接口为使用 nested 修饰的 Local Procedure
- 注册函数时可以给定原型和分组
- 新增若干小方法

zExpression.pas 库更新(编译器与平台兼容)

折叠内部 api,并且以显著方式方式突出用户级 api
修复 c 语言常量字符表达式 bug
修复-round(pi)这类负数表达式问题
修复 trunc(99.99)-1 后面的-1 如果不给空格被分割成独立符号的问题

ZDB 相关库更新

ItemStream.pas 库更新(编译器与平台兼容)

- 使用 CopyFrom64 代替 Tstream.CopyFrom,因为 Tstream.CopyFrom 不支持 64 位 copy,该方法不支持大于 2G 的文件 copy

ObjectData.pas 库更新(编译器与平台兼容)

- 使用折叠方式区分内部 api 与用户 api
- 重新规范化结构体命名

ObjectDataManager.pas 库更新(编译器与平台兼容)

- 修复断电后的数据复原问题,该功能需要打开 zDefine.inc 中的 ZDB_PHYSICAL_FLUSH 编译开关才能生效
- 优化 flush 机制稳定型

光栅库更新

MemoryRaster.pas 库更新(编译器与平台兼容)

MemoryRaster.pas 是 ZAI 的光栅支持核心库,也是 Core 级支持库,该库一旦更新,上面架设的应用,高级库都会受影响,1.30 的光栅库更新巨大,不亚于开发一批小项目,堪称史无前例,同时也非常谨慎.

1. 新增 png 存储支持,png 存储时采用动态扫描方式,自动分析灰度,黑白,真彩,透明真彩,然后计算交错线,压缩率
2. 新增 png 读取支持,自动匹配数据头,zlib 兼容方式解压缩
3. 新增大型形态学数学支持地基: Tmorphomatics,并行化与经典算法支持
4. 新增大型形态学二值化支持地基: TMorphologyBinaryzation,并行化与经典算法支持
5. 算法优化级重构形态学分割模块地基: TmorphologySegmentation,大幅度重构,等同于全新形态学分割算法
6. 重构规则线分割模块:TMorphologyRCLines
7. 新增三套像素形态学体系支持:
   a) CMYK
   b) HIS
   c) YIQ
8. 新增近似像素形态数学支持: BuildApproximateMorphomatics,这是一个自定义近似像素抽取的方法
9. 新增 21 种固定形态像素支持,包括 CMYK,HIS,YIQ,RGBA,以及 5 种近似色彩模型.
10. 算法优化级重构顶点着色器: TrasterVertex,着色器现在是以并行方式着色,受着色过程长宽粒度参数影响,简单来说,我们投影时,面积越大, 长宽粒度也就越大,并行化会自动化根据粒度调整成最佳的并行计算模型
11. 修复大面积投影问题:取消了 32 位整型在大规模投影的前置计算方式,改用 64 位整数作为前置计算
12. 投影机制可以直接应用于:像素投影,形态数学投影,形态学二值化投影
13. 重构霍夫线支持方式:新的霍夫线变换支持方式全部依赖于形态学地基
14. 重做旋转检测体系:光栅倾斜校正,倾斜角度检测,形态数学倾斜检测,形态学二值化倾斜检测
15. 修复光栅字体的存储方式:当字符代码为$FFFF,同时打开 rangecheck 编译,会出现越界以及丢失$FFFF 符号
16. 优化读取函数:LoadFromStream,LoadFromFile,读取数据时会自动判断光栅格式,公共格式支持 jpg,jls,png,bmp,私有格式支持,序列化光栅,yuv 光栅,yuv half 光栅, yuv quart 光栅
17. 优化存储函数,SaveToFile,会自动根据扩展名决定存储格式
18. 新增 SigmaGaussian 处理机制,这是预处理技术体系中的经典平滑方法,该机制同时支持像素与形态数学两种计算模式
19. 原子线条支持形态数学,形态学二值化
20. 形态学二值化支持线条探头
21. Inbuild 内置字体光栅更新
22. 移除 FPC 的 Procs 并行机制
23. 移除 Delphi 的 TParallel 并行机制
24. 使用 CoreClasses.pas 库内置并行机制替代,新并行化机制稳定性和性能优于前两者.

MemoryRaster_DocumentTextDetector.pas 库更新(编译器与平台兼容)

- 这是 1.30 新增的文本检测于分割库
- 该库只有一个函数, DocumentTextDetector,从一张图片中检测文档,并返回每个文字的框体坐标

PictureViewerInterface.pas 库更新(编译器与平台兼容)

- 这是 1.30 新增的交互图片的预览库
- 我们看到的 h.264 encoder,morphology expression 等等工具,里面可拖动缩放,带有直方图信息的图片浏览器就是该库驱动

zDrawEngine.pas 库更新(编译器与平台兼容)

新增光栅纹理的高斯阴影,在 demo:DrawEngineFontEdge 中展示了平滑边缘的阴影技术,该技术只支持图片的高斯影子,不支持文本高斯影子
更新 progress 逻辑机制,TdrawEngine 现在可以直接 Progress(),它会自动填充时间碎片,而不必每次 progress(delta time)
新增 drawLabBox 绘图 api
新增 drawtile 图片 api
新增若干小方法

MorphologyExpression.pas 库更新(编译器与平台兼容)

- 这是 1.30 新增的形态学表达式的脚本库
- 该库有对应的 morphology expression editor 工具,可以独立在我们的程序中运行形态学脚本

PyramidSpace.pas 库更新(编译器与平台兼容)

- 移除 FPC 的 Procs 并行机制
- 移除 Delphi 的 TParallel 并行机制
- 使用 CoreClasses.pas 库内置并行机制替代,新并行化机制稳定性和性能优于前两者.

FastHistogramSpace.pas 库更新(编译器与平台兼容)

- 移除 FPC 的 Procs 并行机制
- 移除 Delphi 的 TParallel 并行机制
- 使用 CoreClasses.pas 库内置并行机制替代,新并行化机制稳定性和性能优于前两者.

FMXCharacterMapBuilder.pas 库更新(只能在 delphi 使用)

- 这是 1.30 新增用于截取字符光栅到 TMemoryRaster 的支持库
- 该库只能工作于 delphi 下,可以工作 delphi 构建的于移动平台
- 该库不支持 fpc,lazarus,lcl

视频支持库更新

FFMPEG_Reader.pas 库更新(需要外部库支持,兼容 FPC+IOT,Delphi+移动平台)

修复 rtsp/rtmp/https/http 等推流协议播放视频会出现马赛克的问题
新增 gpu 支持的解码器体系,主要使用 cuvid 解码器

FFMPEG_Writer.pas 库更新(需要外部库支持,兼容 FPC+IOT,Delphi+移动平台)

新增 gpu 支持的编码器体系,主要是 cuvid 的加速接口
编码器体系在普通民用 gpu 只能同时支持 2 路 gpu 编码器,如果超过 2 路编码器,必须使用 cpu 来编码
编码器体系在专业 gpu 可以支持高于 2 路的 gpu 编码器,据体支持细节需要参考 nvidia 给出硬件参数文档

FFMPEG.pas 库更新(需要外部库支持,兼容 FPC+IOT,Delphi+移动平台)

修正 FFMPEG 依赖库的引用目录,我们可以根据指定目录接口外部 FFMPEG 库
修正 FFMPEG 依赖库载入非指定外部库时发生异常的问题

自然语言支持库更新

GBKMediaCenter.pas 库更新

不在使用 code InBuild 编译方式构建语料库,改用了 ZDB 数据库方式载入 InBuild 语料库
InBuild 语料库可以使用 Package Tool 进行编辑和导出
优化 GBKMediaCenter 库在应用启动时语料库的加载性能
新增以外部 ZDB 数据库方式载入语料库的 api 接口

GBK.pas 库更新

新增函数, PyNoSpace,汉字转 ascii 方式的拼音字母,可以自动矫正多音字的拼音字母

FastGBK.pas 库更新

不在使用 code InBuild 编译方式构建语料库,改用了 ZDB 数据库方式载入 InBuild 语料库

Z-AI 支持库更新

zAI.Pas 库更新

- 新增算法:Efficient Graph-Based Image Segmentation
- 新增算法:Minimum Barrier Salient Object Detection at 80 FPS
- 新增算法:Segmentation as Selective Search for Object Recognition
- 新增算法:Unmixing-Based Soft Color Segmentation for Image Manipulation
- 新增算法:Poisson Image Editing
- 新增算法:GrabCut — Interactive Foreground Extraction using Iterated Graph Cuts
- 新增算法:Tesseract OCR

zAI_Common.pas 库更新

- 调整 Z-AI.conf 搜索机制,按下列顺序
  - 搜索**应用程序所在目录**,如果发现启动目录有 Z-AI.conf,载入它
  - 从"**文档**"目录搜索 Z-AI.conf, 如果发现启动目录有 Z-AI.conf,载入它
- 调整 WhereFileFromConfigure 函数搜索机制
  - 搜索**应用程序所在目录**,如果发现用户文件返回
  - 搜索 **Z-AI.conf 中的配置项 SearchDirectory 指定的目录**,如果发现用户文件返回
  - 搜索"**文档**"目录,如果发现用户文件返回
  - 搜索**应用程序当前目录**,如果发现用户文件返回
- 存储数据格式更新:新增 fileinfo 数据,这项数据来自工具链,大都保存文件名,图片分辨率,hash 代码这类数据,当导出成.imgdataset 格式后再导入到建模工具,fileinfo 就会清空,数据反复导出导入会影响合并效果,故此新增 fileinfo 数据解决该问题
- 增强语义分割数据的呈现效果
- 移除 FPC 的 Procs 并行机制
- 移除 Delphi 的 TParallel 并行机制
- 使用 CoreClasses.pas 库内置并行机制替代,新并行化机制稳定性和性能优于前两者.
- 增加若干小方法

zAI_Editor_Common.pas 库更新

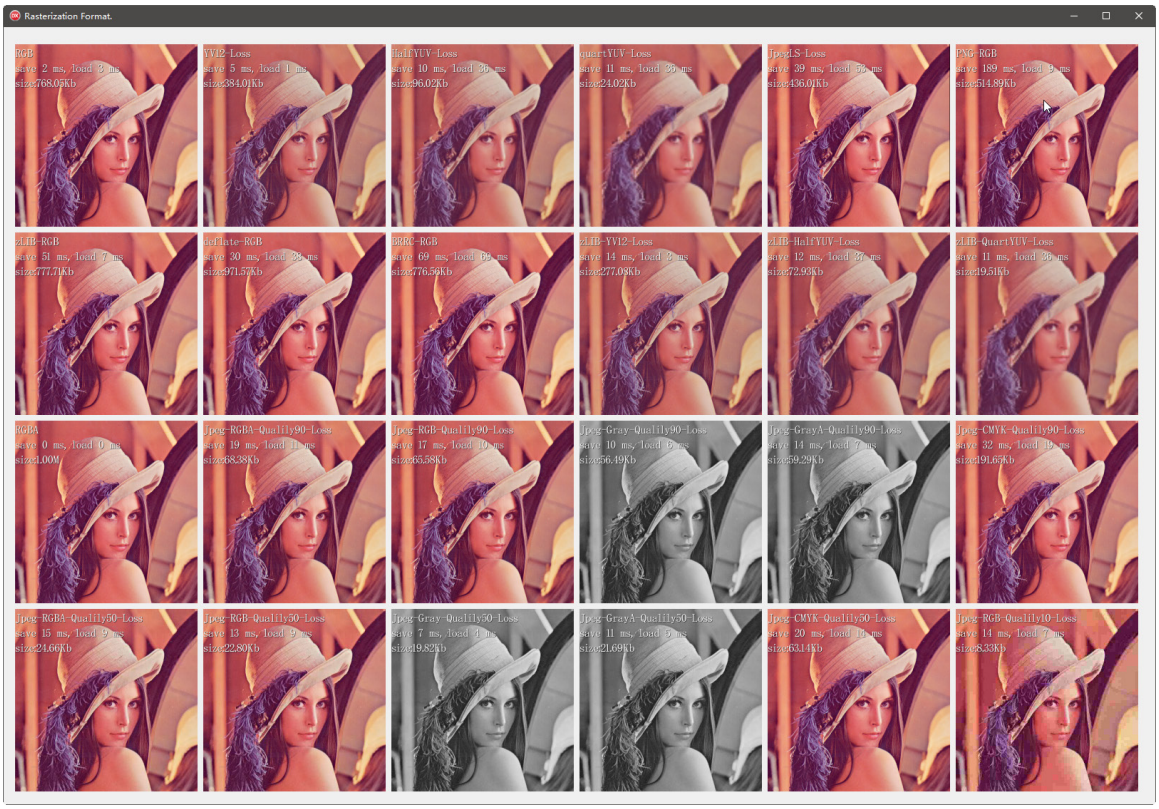
- 存储数据格式更新:新增 fileinfo 数据,这项数据来自工具链,大都保存文件名,图片分辨率,hash 代码这类数据,当导出成.imgdataset 格式后再导入到建模工具,fileinfo 就会清空,数据反复导出导入会影响合并效果,故此新增 fileinfo
- 移除 FPC 的 Procs 并行机制
- 移除 Delphi 的 TParallel 并行机制
- 使用 CoreClasses.pas 库内置并行机制替代,新并行化机制稳定性和性能优于前两者.
- 增加若干小方法

Z-AI Free Demo 更新

Free Demo 为开源方式的自由 Demo.

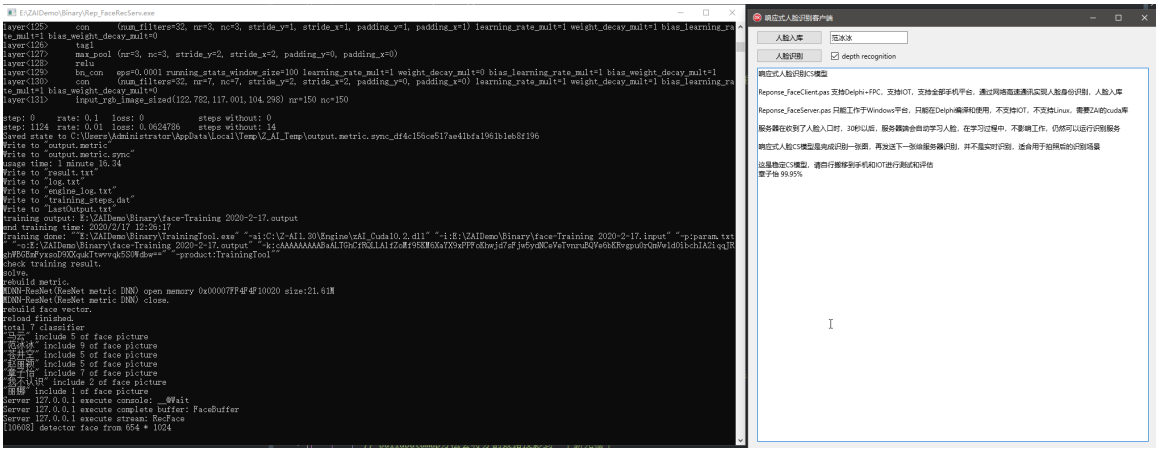
Rasterizatio Format Demo 更新

- 新增了 png 的 save/load 性能报告



CS 响应式人脸入库/人脸识别 Demo 更新

- 该 demo 示范了自动化的人脸入库与识别
- 修复了因为缺少 TrainingTool.exe 无法训练人脸的 bug

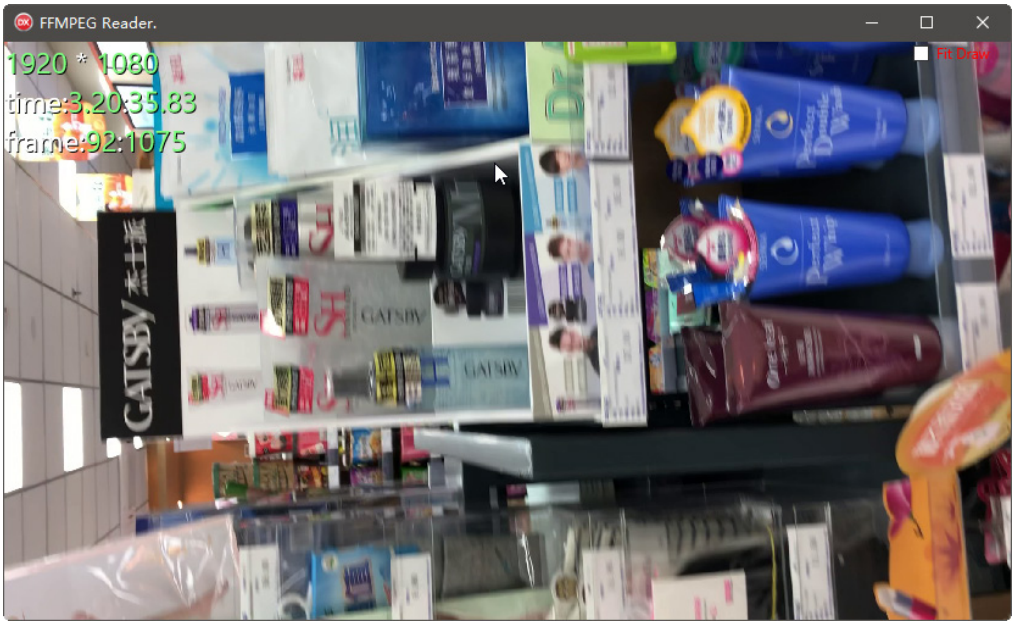


DNN_OD demo 更新

- 该 demo 被很多用户作为基础性能测试工具使用
- 已经在代码中备注了 cuda 对于线程调度的问题,详细说明了使用 cuda 必须在主线程调用识别计算 api
- 同时以此 demo 为例,在所有使用 cuda 计算的 demo 中,都采用了主线程调用识别 api 的方式.
- 过去在线程中调用 cuda 只能在 cuda9.x 使用,在 cuda10 版本之后,cuda 必须在主线程调用,因为 cuda 都是立即计算完成,所以不存在很卡很慢这种问题.
- 以监控识别为例: 在这类工作场景 cuda 会长时间循环工作,即使只有一点泄露,也会造成崩溃,请各位用户在使用 ZAI 做识别时尽量参考该 demo 的主线程 call cuda 计算的范例,主要是识别调用.
- 如果不使用 cuda 作为计算引擎,那就没有主线程要求,比如我们使用 MKL 计算识别,我们可以让它工作于线程

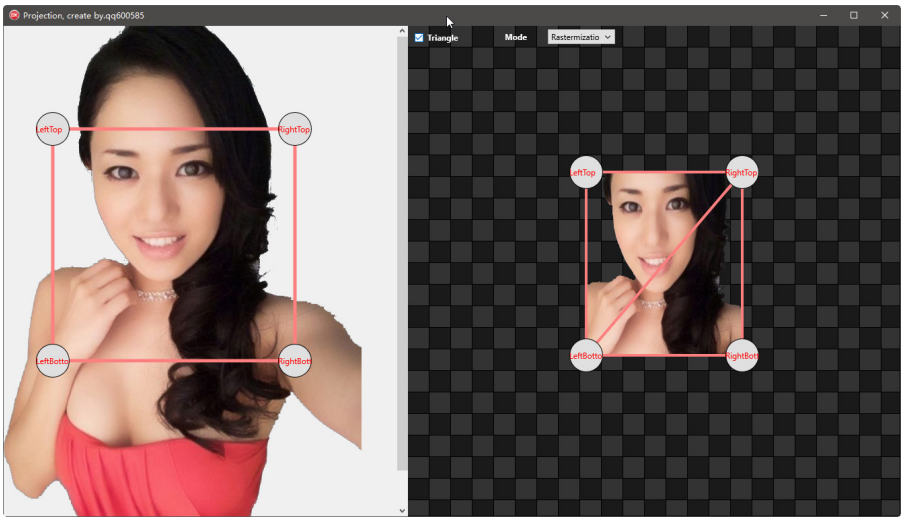
FFMPEG_ReaderDemo 更新

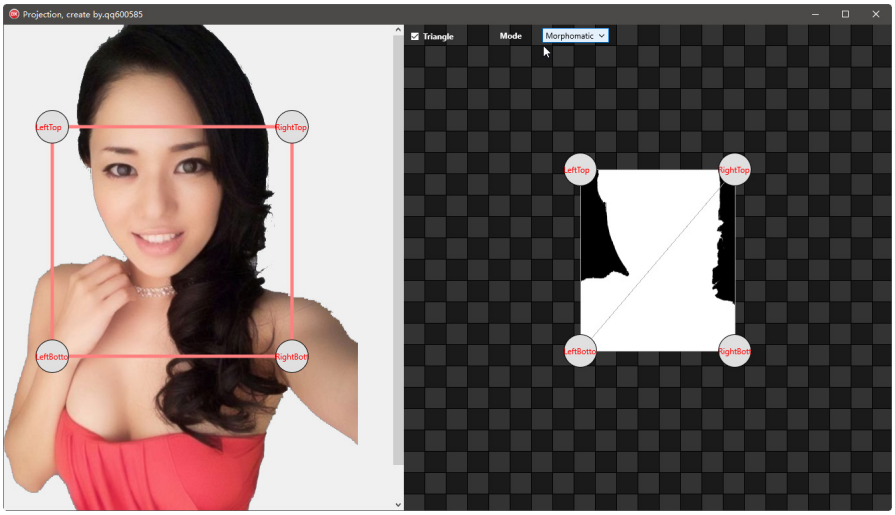
- 在该 demo 中备注了使用 rtsp/rtmp 等等协议的方法

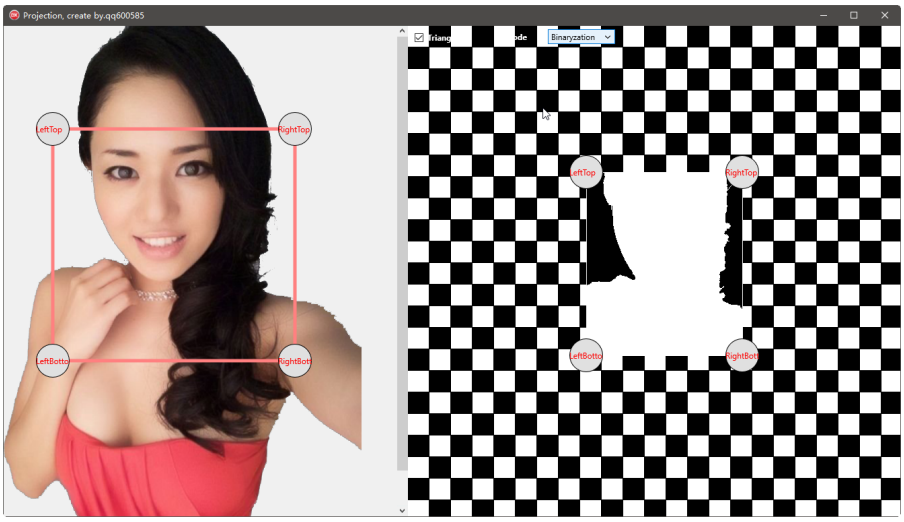


memoryRasterProjection demo 更新

在该 demo 中提供了三角投影的预览开关
提供了形态数学的三角投影预览效果
提供了形态学二值化的三角投影预览效果

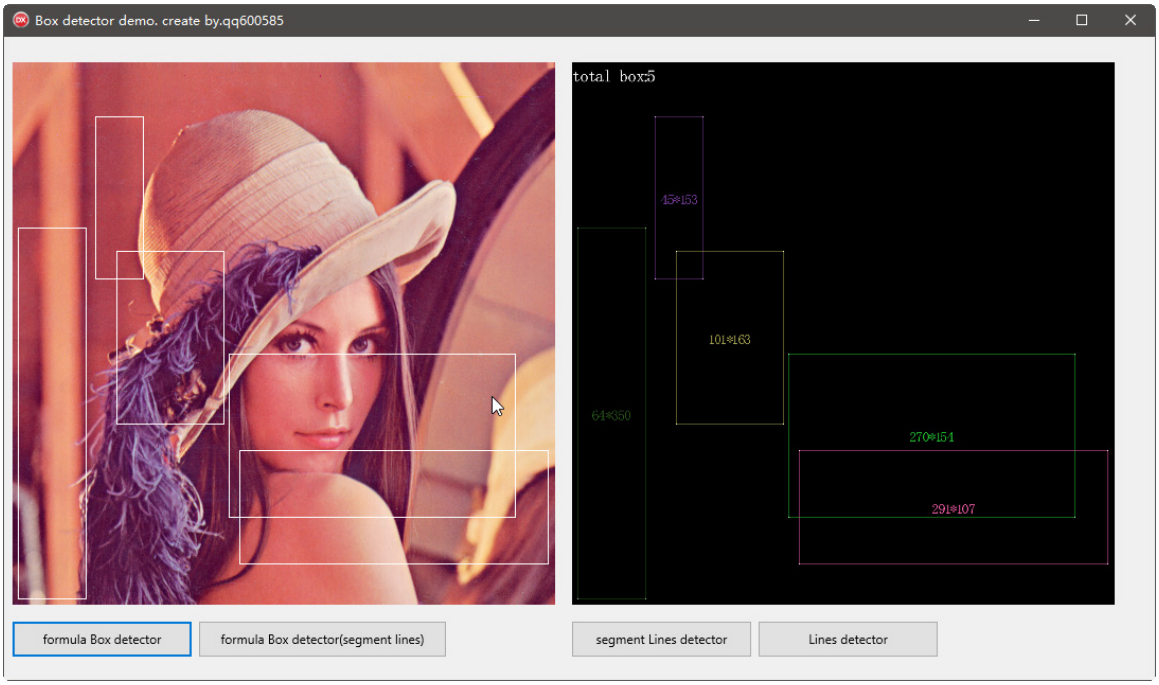






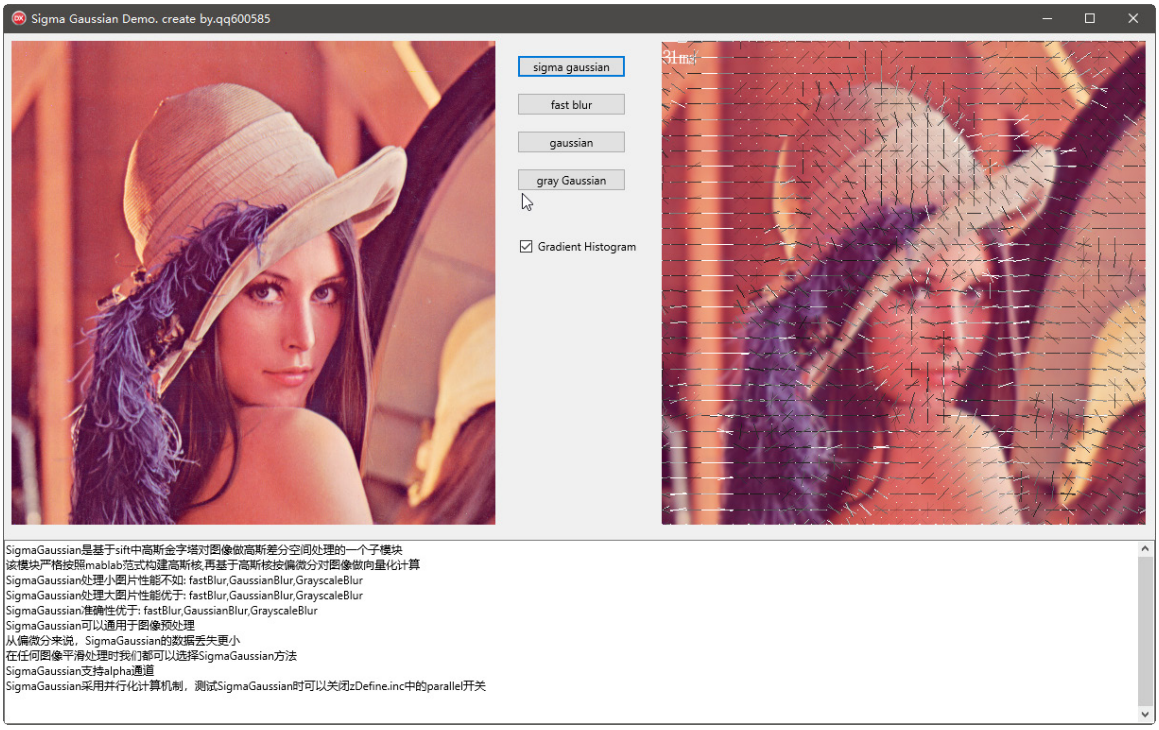
对齐线检测 demo 更新

- 该 demo 示范了怎样检测并计算对齐线
- 1.30 更新了形态学地基,该 demo 同步更新成:全部以形态学作为地基计算对齐线



sigmaGaussian demo 更新

- 1.30 新增 demo
- 该 demo 示范了高斯预处理的工作机制

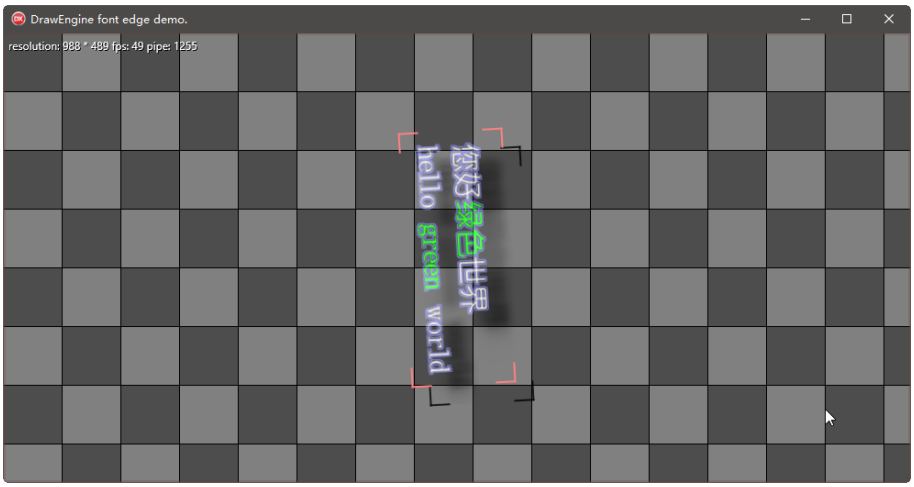


TrainingTool.exe 更新

- 这是 ZAI 的训练模型用的 shell 工具,由于部分 demo 需要使用到它,顾,将它集成到 demo 中来.
- 这只是一个命令行形式的 shell 工具.

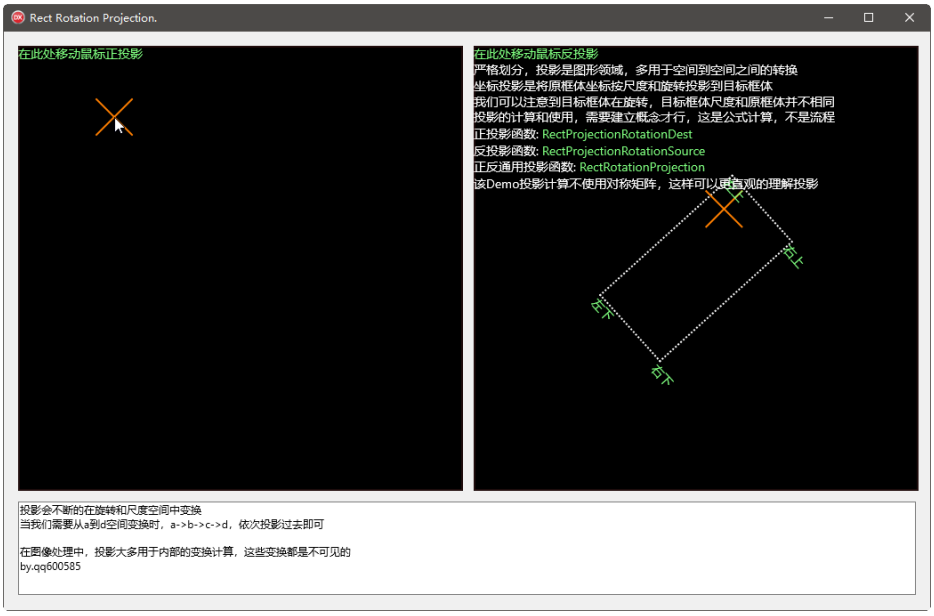
DrawEngineFontEdge demo 更新

- 1.30 新增 demo,它示范了怎样给光栅字体加边框
- 同时它也示范了,怎样使用高斯平滑的影子



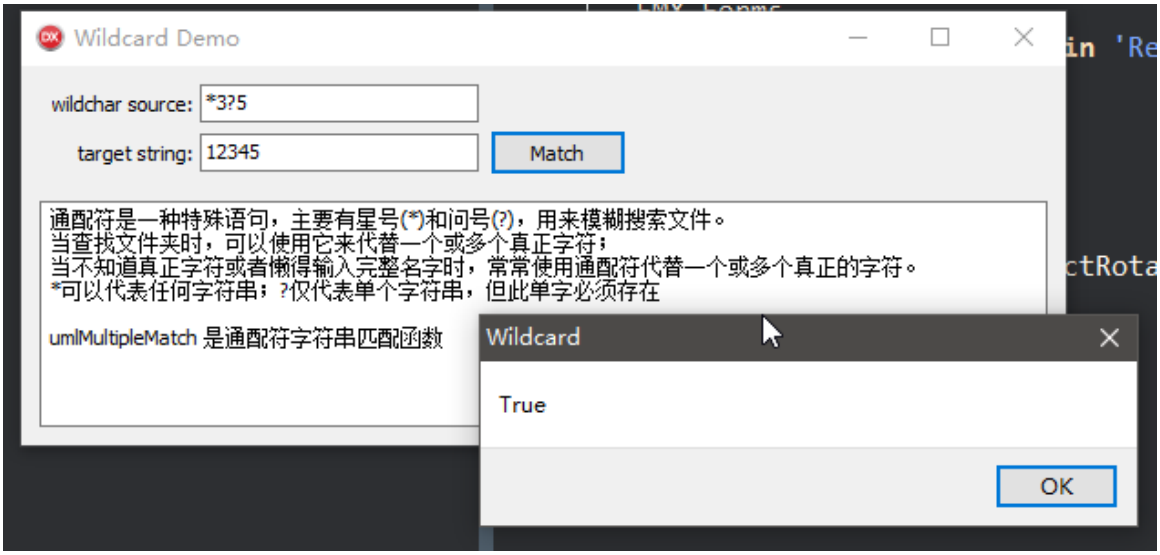
RectRotationProjection demo 更新

- 1.30 新增 demo,它示范了正反框体投影坐标系换算方法
- 包含框体旋转,非旋转状态,正投影坐标系计算,反投影坐标系计算



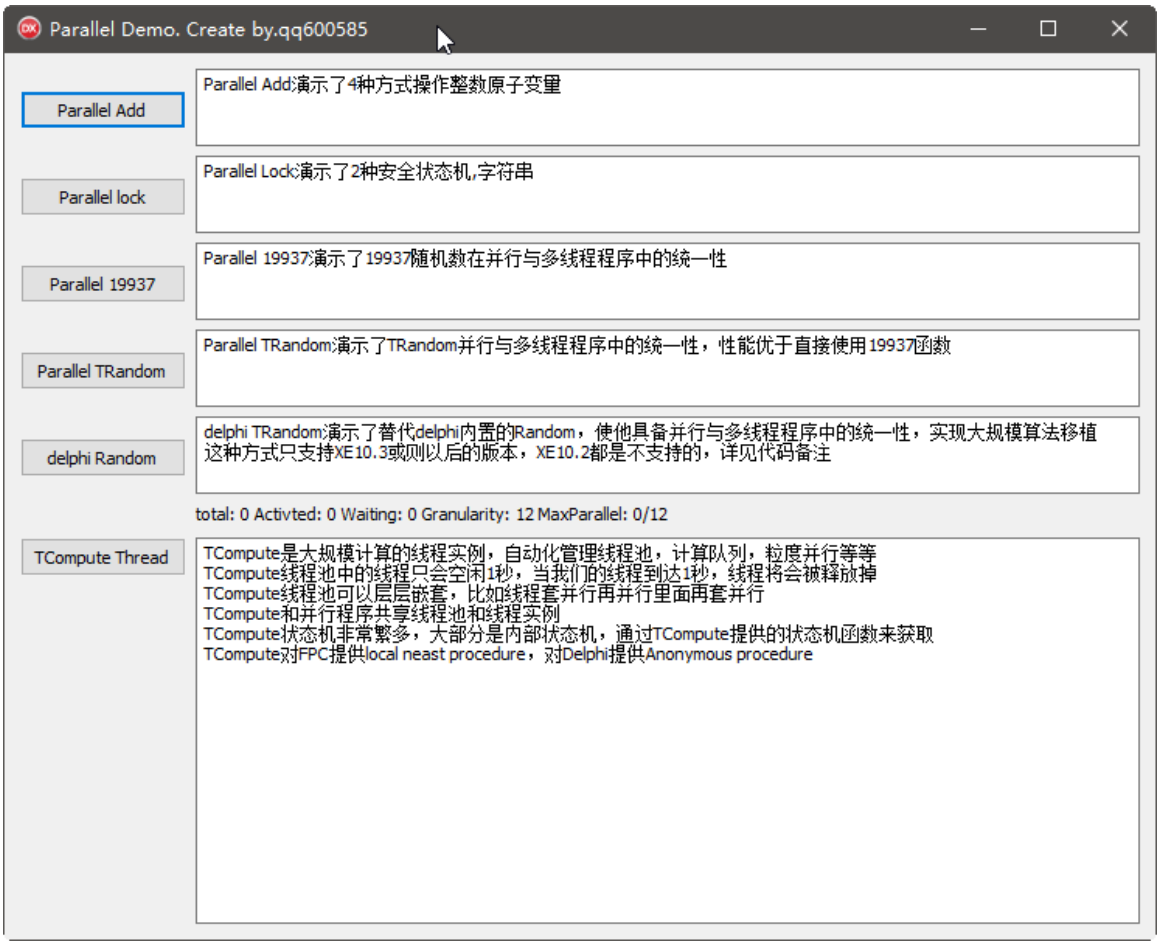
Wildcard demo 更新

- 1.30 新增 demo,该 demo 示范了通配符的傻瓜应用方式



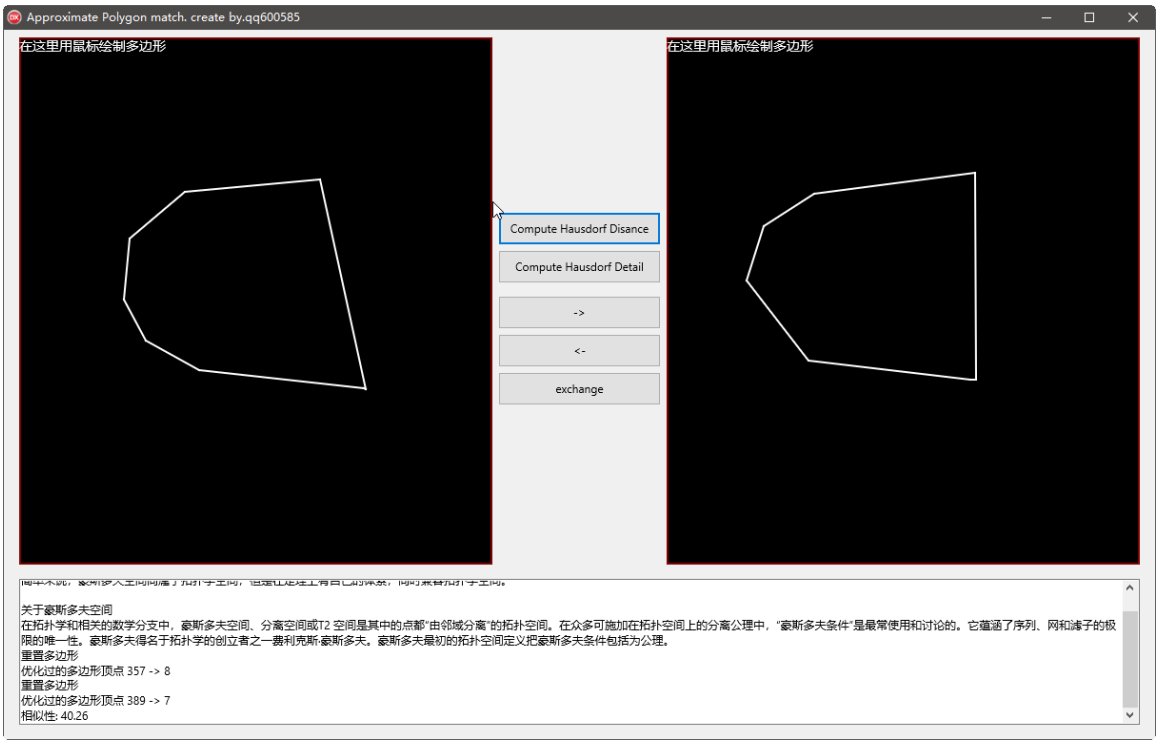
ParallelDemo 更新

- 1.30 新增 demo,它示范了内核中的并行程序,原子锁,支持多线程的统一随机数,并行计算等等基础
- 这是非常重要的基础 demo,它使用内核 api 支持



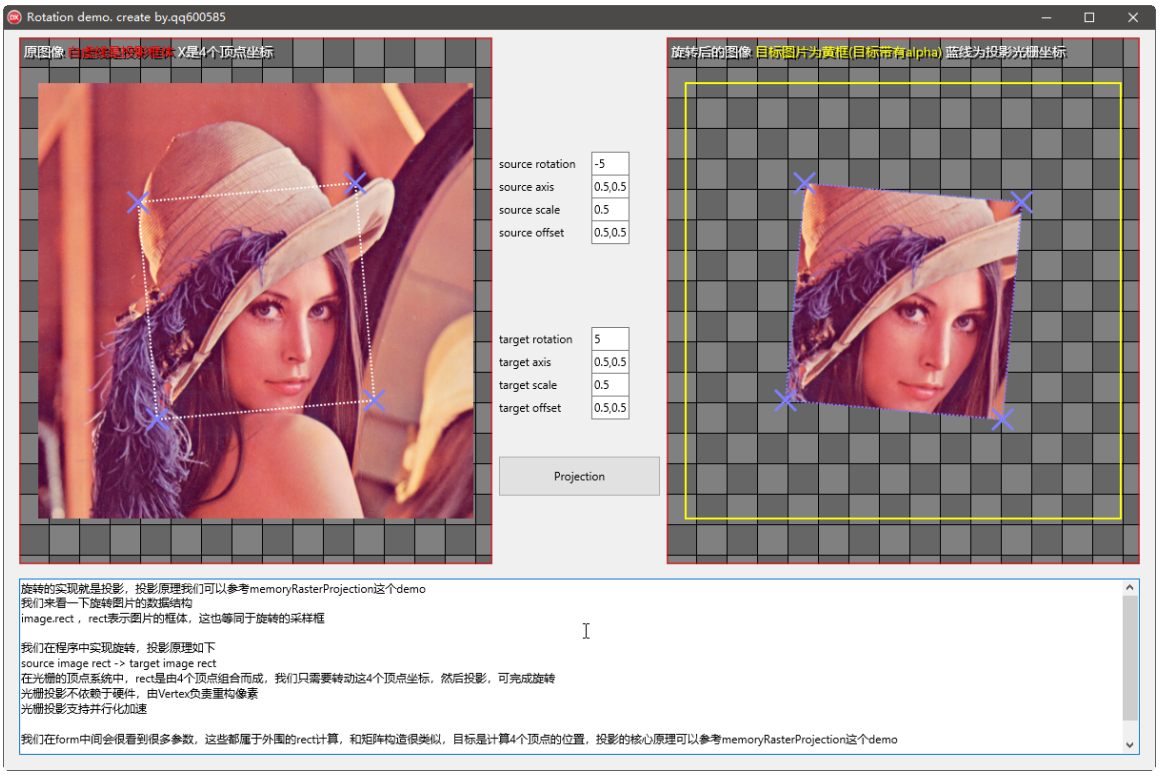
ApproximatePolygon demo 更新

- 1.30 新增 demo,该 demo 示范了使用 Hausdorf 算法求取两个多边形的距离,从而得到相似度



RotationDemo 更新

- 1.30 新增 demo,该 demo 示范了怎样用一行代码从坐标系模拟出常用矩阵的计算方式
- 同时也示范了光栅投影的功能和作用

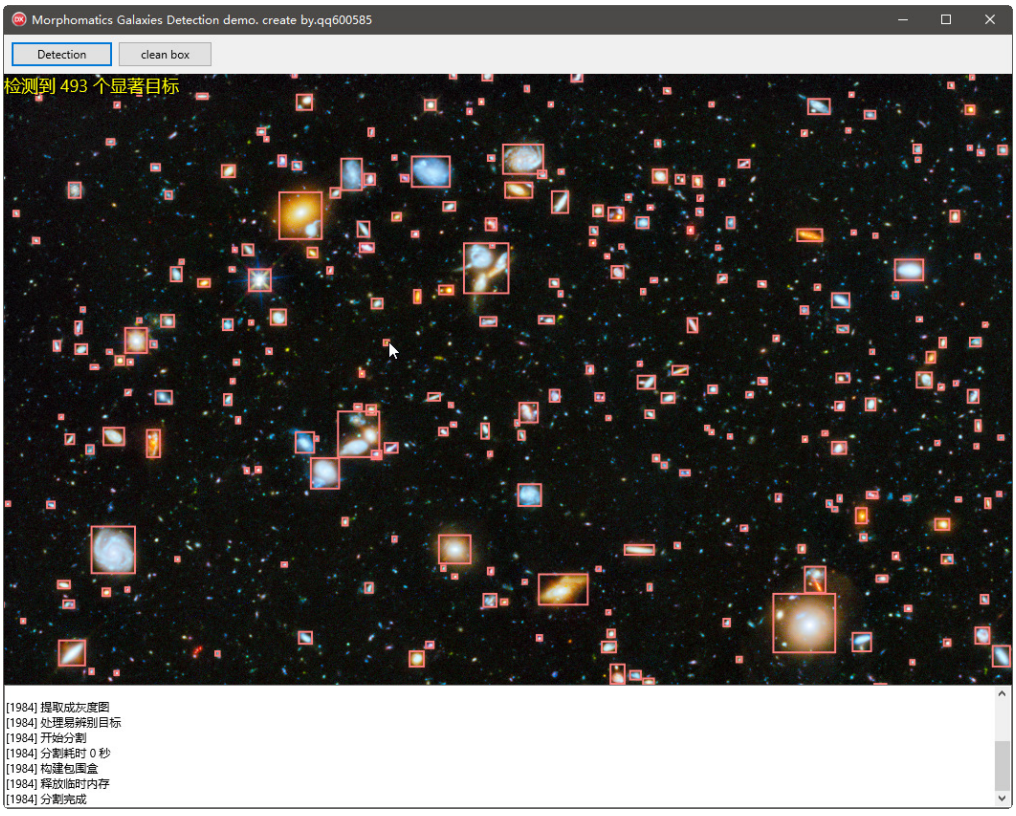


PolygonRegion demo 更新

- 1.30 新增 demo,该 demo 示范了监控视频中的一个特殊区域,并且追踪出现在该区域的物体

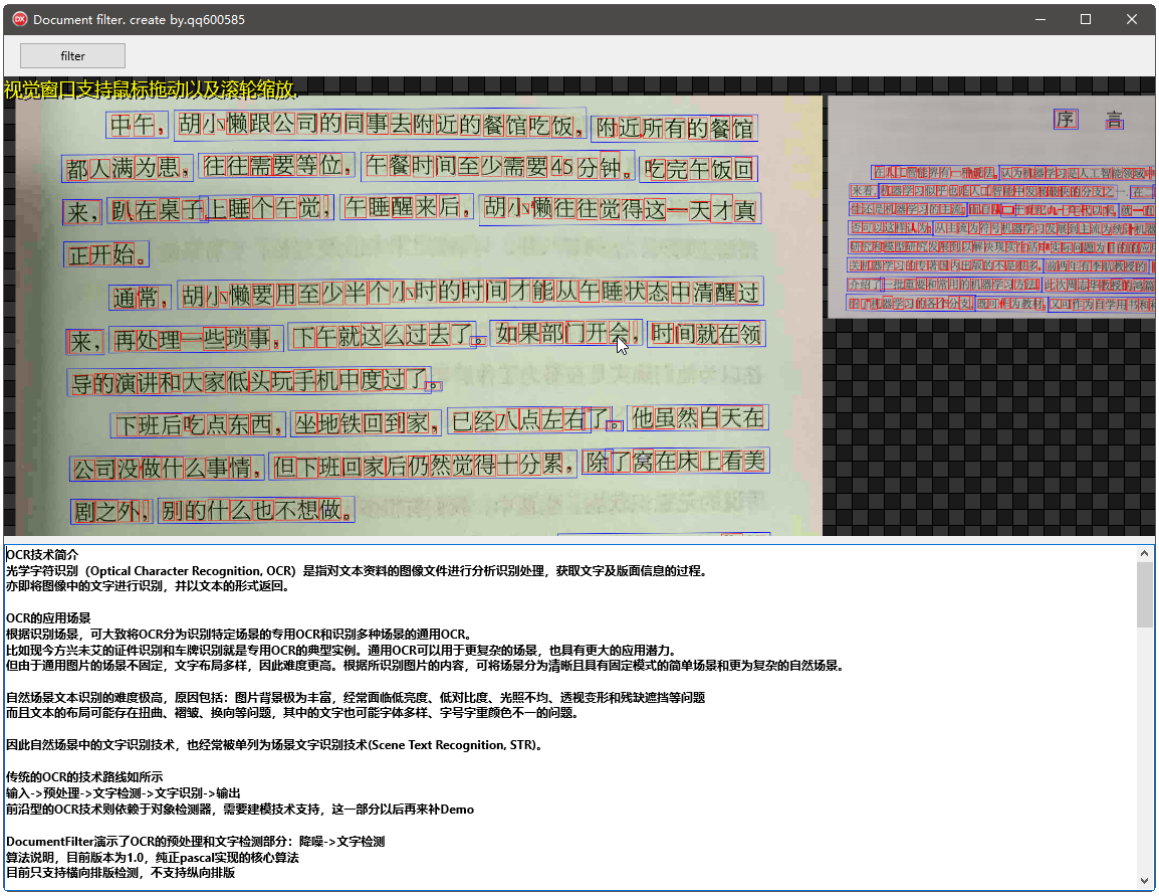
MorphGalaxiesDetection demo 更新

- 1.30 新增 demo,该 demo 示范了使用形态学做天文图像中的显著星系计数/检测



DocumentFilter demo 更新

- 1.30 新增 demo,该 demo 示范了检测文档图像中的文行,文字位置

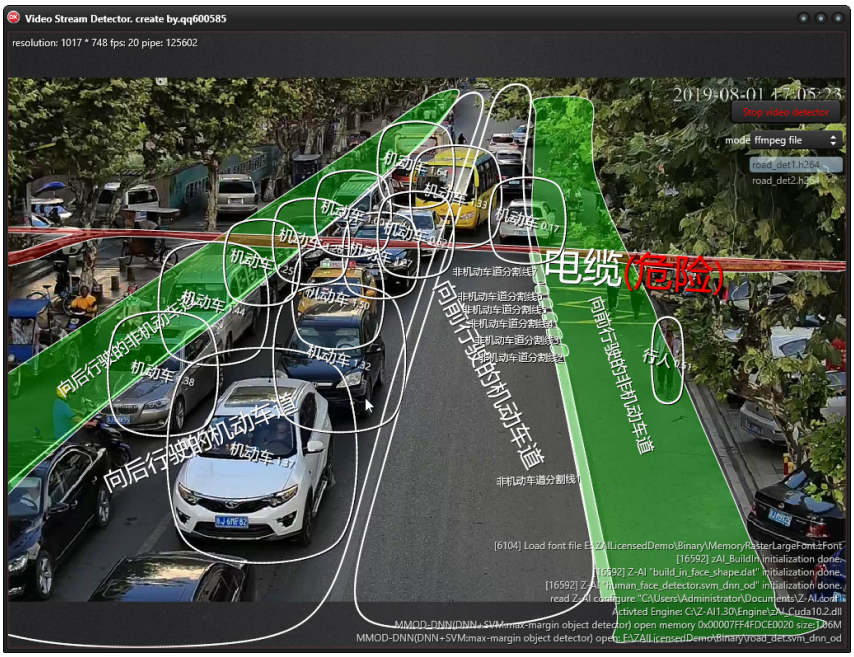


授权 Demo 更新

授权 demo 表示比较完善,成熟,可以应用的 demo,同时它也是不可轻易获取的解决方案

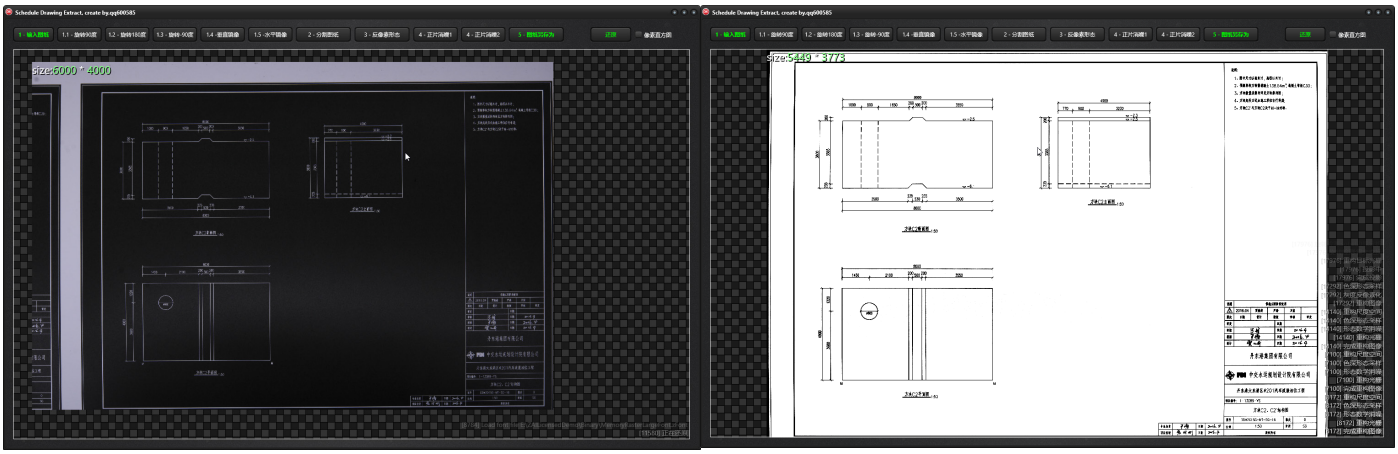
RoadCameraDetector demo 更新

- 1.21 的监控视频 demo,得益于 1.30 cuvid 新技术应用,多路推流视频监控性能有所提升
- 1.30 更新了主线程计算规则,长时间监控不会再发生显存泄露



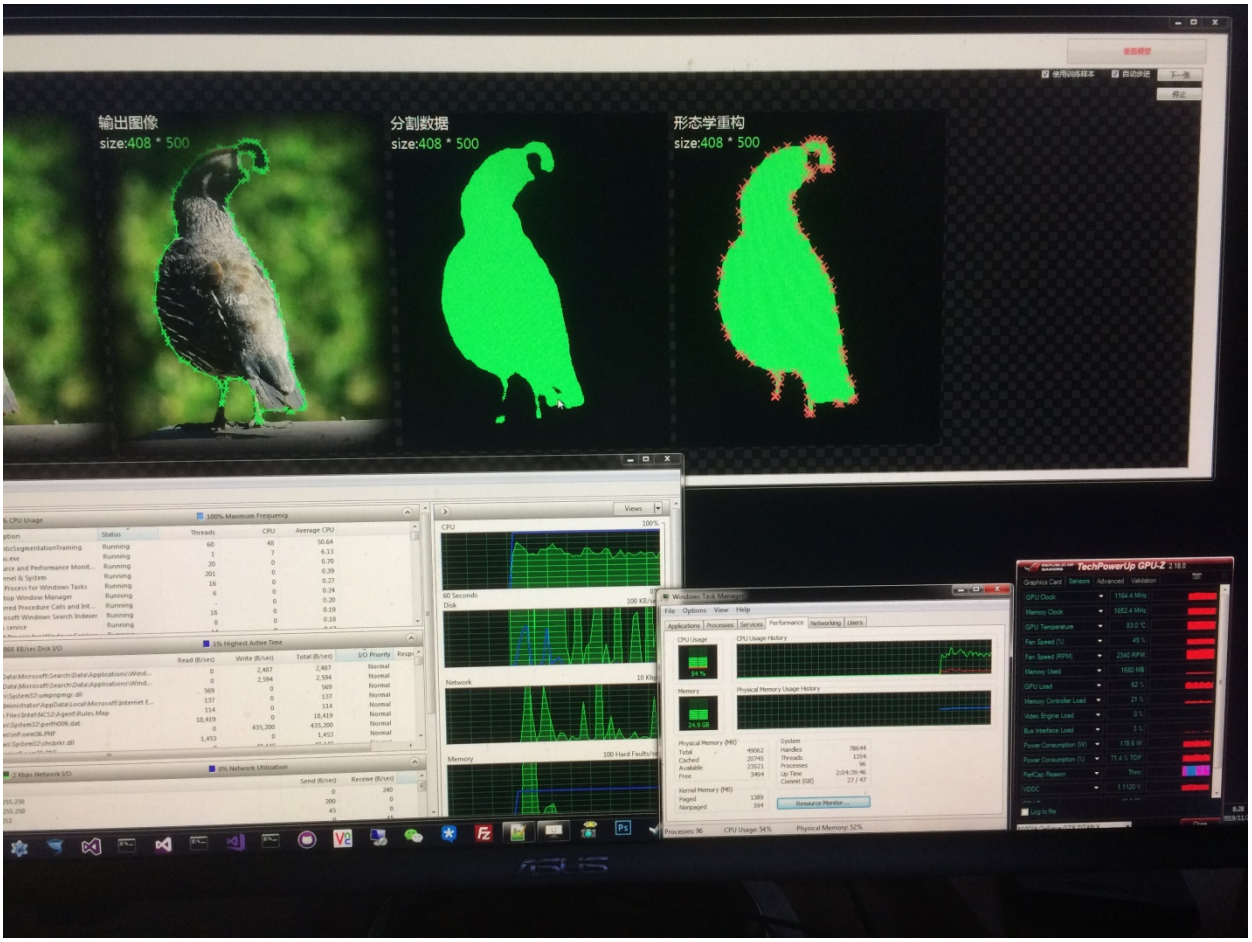
ScheduleDrawingExtract demo 更新

- 1.30 新增 demo,它示范了超高分辨率工程胶片图的切割与矫正,同时它也可以降噪,它是专属行业应用不对所有授权用户开放
- 它提供了 shell api 接口,可以直接支持在 c#工程中完成自动化切割,省却人工做图像处理的资源开销



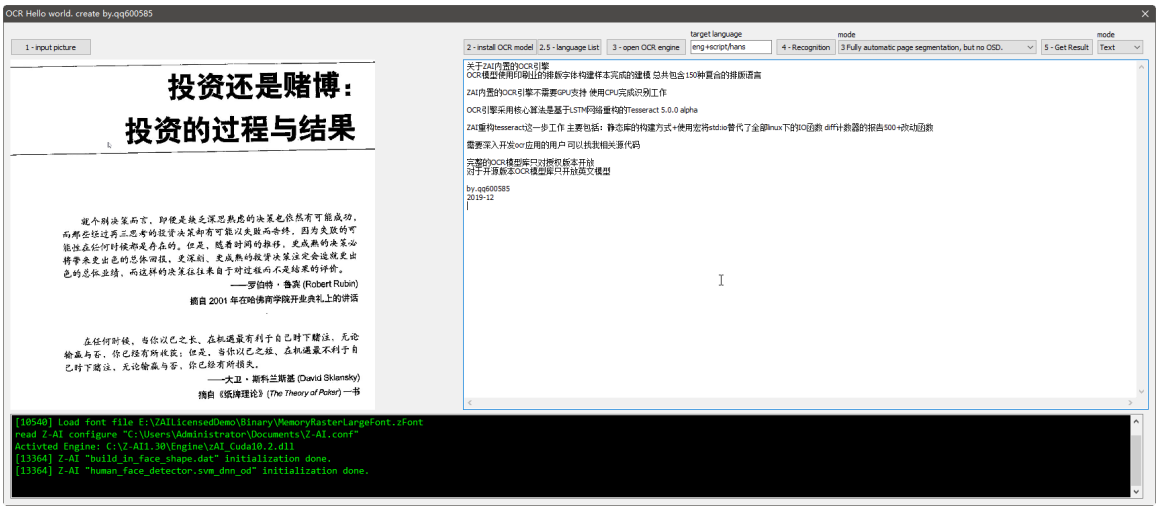
SemanticSegmentationTraining demo 更新

- 1.30 新增 demo,它示范了超大规模图像分割科学性与可行性,每秒可以计算完成 4 张精确分割图像
- 运行该 Demo 需要 32GB 内存以及 8G 以上显存,可长时间运行不泄露显存和内存,可部署于服务器
- 它标志了图像语义分割在 pascal 圈已经进一步走向成熟,不再是遥远的东西,
- 因为图像语义分割涉及到的计算规模非常大,形态学依赖,拓扑学依赖,欧几里几何学依赖,该 demo 可以算一种小规模的技术革命,它标志着后续新技术的更新应用.



OCRHelloWorld demo 更新

- 1.30 新增 demo,它示范了 OCR 识别的基本使用方法
- 它并没有详细讲解在工程中应用需要注意的问题,一般来说,我们需要做前置处理,同时我们也需要注意一点:OCR 不支持并行化的调用,绕过这些坑我们需要自己想办法解决,或则直接找作者购买 zOCR 的专业解决方案.



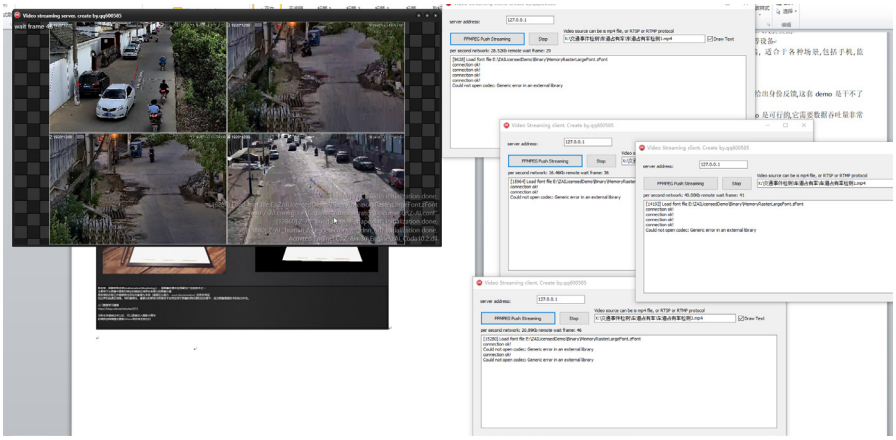
该 demo 主要预处理入门,它可以与 OCR 相配合,使用形态学提取图像中的纸张,再拿去识别

视频推流组合 demo

这套推流组合,都是基于推流技术,实现小数据的视频传输与处理
该 demo 由 4 个模块组合而成

1. **VidStreamingServ**:裸服务器,不做任何图像处理的推流服务器,适合自己在它上面实现逻辑业务处理
2. **VidStreamingFaceServ**:带有人脸检测的推流服务器,示范在裸服务器基础上加入人脸检测
3. **VideoStreamingClient**:推流客户端,适合于各种场景,包括手机,监控,IOT,PC 等等设备
4. **VideoStreamingClient_HighPerformanceMode**:高性能,低功耗的推流客户端, 适合于各种场景,包括手机,监控,IOT,PC 等等设备

- 视频推流都会有延迟,时间在 1-15 秒不等
- 这套 demo 不能作为实时识别需求的解决方案:比如我们站在摄像头前,需要马上给出身份反馈,这套 demo 是干不了这件事的,它只能分析,记录视频,因为网络传输视频延迟太大了.
- 假如我们需要做数据中心的视频分析,接收来自各个监控点的视频数据,这套 demo 是可行的,它需要数据吞吐量非常小.



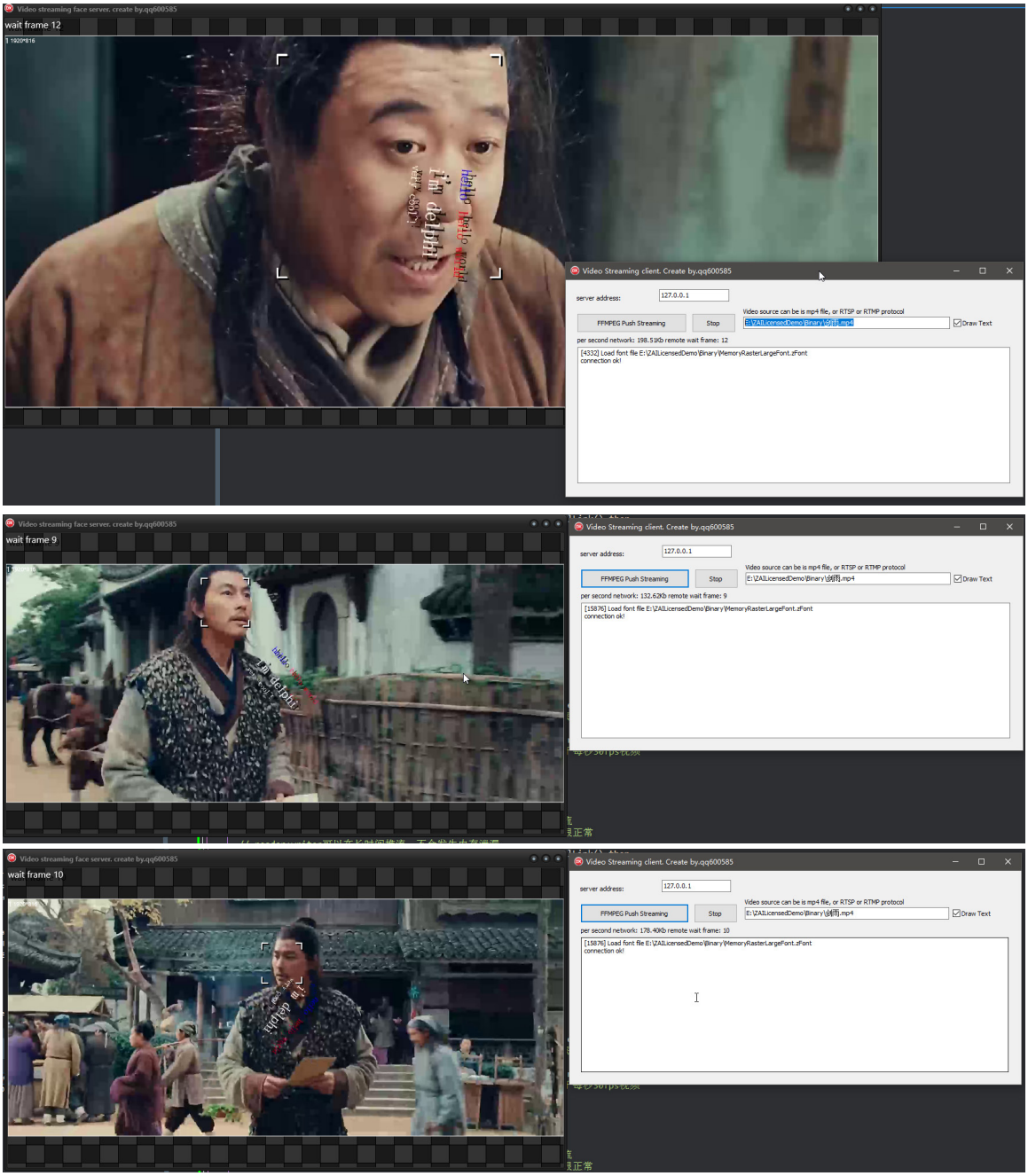
- 这套 demo 对于视频的编码解码均使用 cpu,如果在一台 pc 多路视频推流,会把 cpu 吃满,建议自行更改为 cuda 的解码器,修改方法为,在解码部分,用字符串指定解码器

```
  // TAI_VideoStream是基础FFMPEG技术的streaming reader实现类
  AIVideo := TAI_VideoStream.Create(Processor, nil);
  AIVideo.DiscardDelayBuffer := False;
194 AIVideo.OpenDecodec('h264_cuvid');
  buffQueue := TMemoryStream64List.Create;
```

- 如果需要使用 Cuda 的编码器请使用 **VideoStreamingClient_HighPerformanceMode**,程序里面我已详细备注 cuda 使用说明
- 我们可以开多少 gpu 的编码器,取决于我们 gpu 卡是民用还是专业卡,各个卡的硬件指标不同,并发视频编码加速支持度也不一样.

```
procedure TVideoStreamingClientForm_HighPeformanceMode.FFMPEGPushButtonClick(Sender: TObject);
begin
  if DoubleCli.Connect(servEdit.Text, 9289, 9288) and DoubleCli.TunnelLink() then
    begin
      DoStatus('connection ok!');

      WaitFrame := 0;
      MaxWaitFrame := 0;

      // reader是一个ffmpeg流式解码器实例
      reader := TFFMPEG_Reader.Create(VideoSourceEdit.Text);

      // writer是一个ffmpeg流式编码器实例
      writer := TFFMPEG_Writer.Create(TMemoryStream64.Create);
      writer.AutoFreeOutput := True;

      // 使用nvidia提供的h264 gpu编码器
      // 我自己实测下来，本机只能开两个gpu客户端做编码，多了会报错
      if not writer.OpenCodec('nvenc', reader.Width, reader.Height, reader.PSFRound, 5, 1,
        // 1024*1024是码率，表示每秒100k数据吞吐量，适合720p并且低于每秒30fps视频
        1024 * 1024) then
        if not writer.OpenCodec(AV_CODEC_ID_H264, reader.Width, reader.Height, reader.PSFRound, 5, 1,
          // 1024*1024是码率，表示每秒100k数据吞吐量，适合720p并且低于每秒30fps视频
          1024 * 1024) then
            RaiseInfo('error');

      pushStop.V := False;

      // 开启线程对VideoSource解码，然后在使用Writer编码，同时进行推流
      // 高清视频会需要很多内存，应用程序在工作期间，即使消耗掉16G也很正常
      // reader+writer可以在长时间推流，不会发生内存泄漏
      TComputeThread.RunP_NP(procedure
        var
          raster: TMemoryRaster;
          // TDrawEngine是一种渲染中间件，它可以做到在无任何内存copy下，实时在图像中做二次渲染
          d: TDrawEngine;
          angle: TGeoFloat;
        begin
```

- 如果使用 VidStreamingFaceServ 必须和 VideoStreamingClient_HighPerformanceMode 配套,且不能多开客户端
- VidStreamingFaceServ 演示了在服务器端实现人脸检测的程序范例



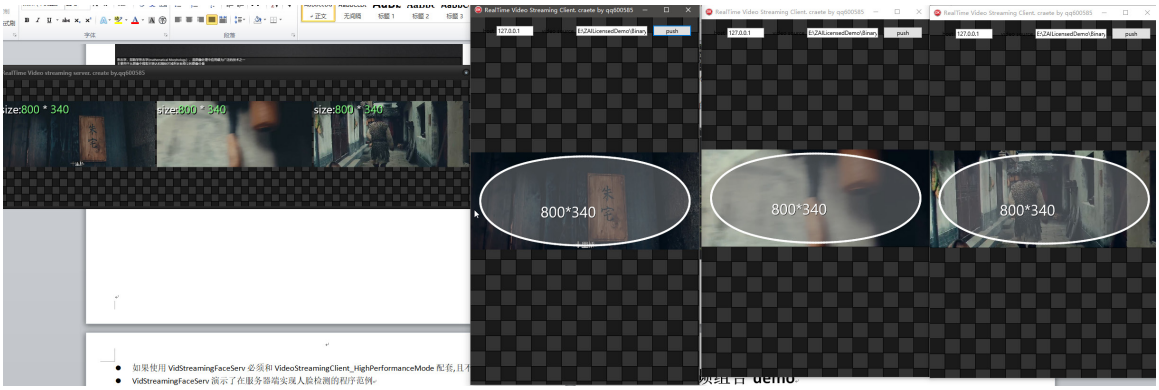
实时识别视频组合 demo

这套组合 demo,都是实时传输,延迟极小,低于 0.5s,可以用于实时识别
该 demo 由 4 个模块组合而成

1. **RealTimeVidStreamingServ**:裸服务器,不做任何图像处理的推流服务器,适合自己在它上面实现逻辑业务处理
2. **RealTimeVidStreamingFaceServ**:带有人脸检测+人脸身份识别的实时视频服务器,示范在裸服务器基础上加入人脸检测+身份识别
3. **RealTimeVideoStreamingClient**:高性能的实时视频客户端,适合于各种场景,包括手机,监控,IOT,PC 等等设备
4. **RealTimeVideoStreamingClient_Mobile**:手机客户端,需要我们自己配置一下手机端构建,自己改下程序才能用

- 这套组合 demo 适用于架设云服务器,让所有的端实时发送视频光栅,通过云服务器识别反馈来跑业务,我们可以假设一下:我们只需要站在摄像头前面,系统自动处理,不需要任何人工介入,因为这是实时识别框架
- 裸服务器可以同时支持多个开客户端连接推送光栅,它的电源功耗远低于推流服务器
- 裸服务器是一种框架,方便我们自行在上面架设各种识别的业务逻辑
- **RealTimeVideoStreamingClient:**是一种 pc 客户端,它只会传输数据给云服务器,不做计算

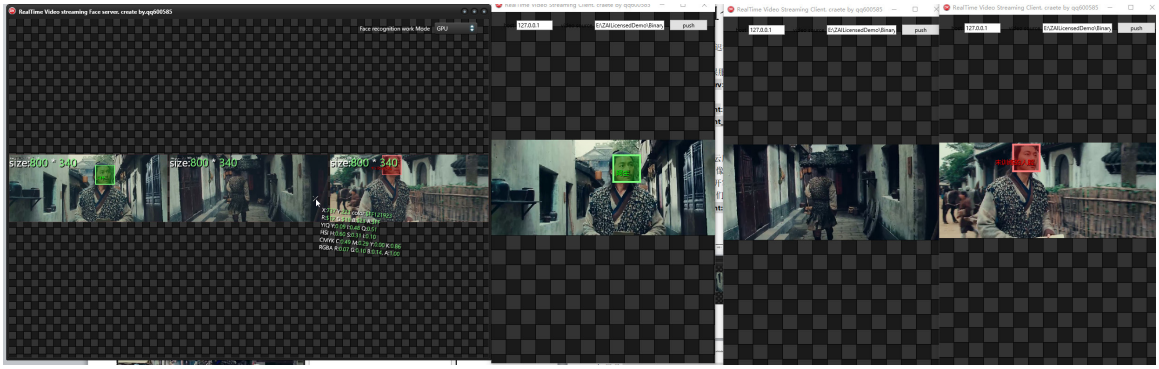
实时裸服务器运行截图



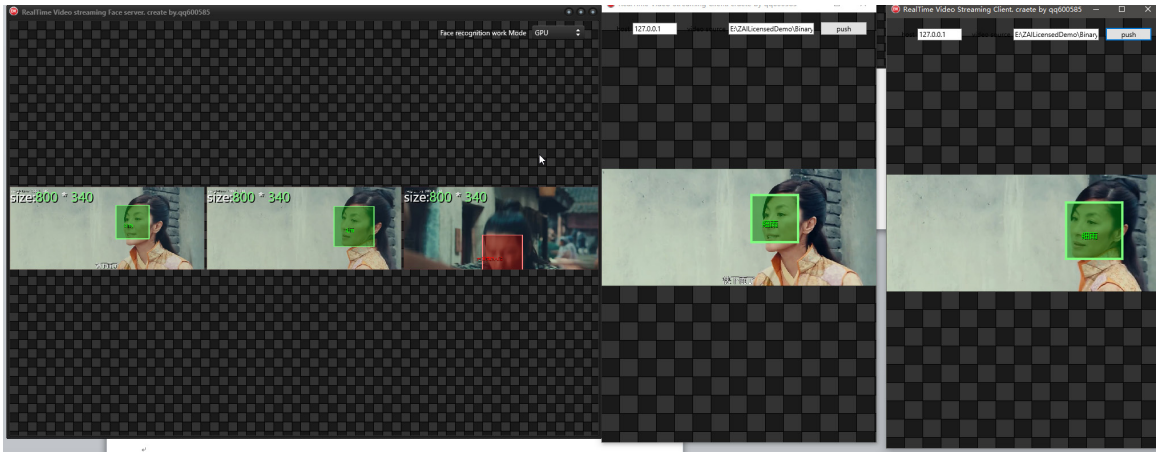
如果是 **RealTimeVidStreamingFaceServ**,我们在启动以后,需要指定一下 face 识别的工作框架,这里要选 gpu 才能做到实时

当我们同时打开多个客户端推送以后,我们可以看到客户端当前图像和服务器是一致的,它代表是实时性
另一方面,只要看到绿色框体,就表示人脸身份的成功匹配,这时候,客户端会收到反馈.做业务开发,以此类推

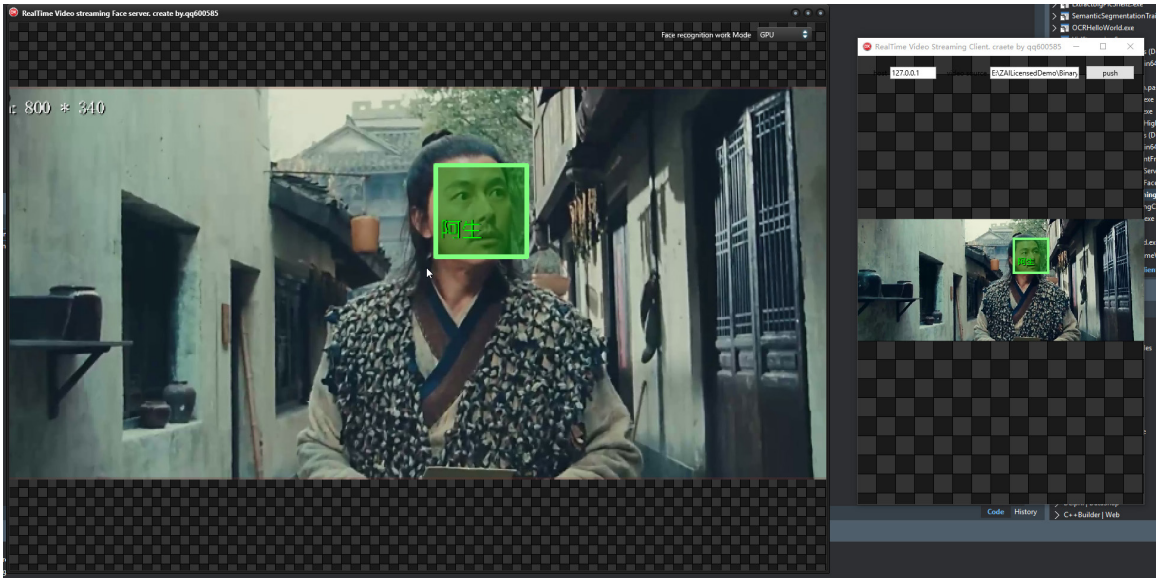
文档页面很小,下图可以通过放大 pdf 来查看细节效果



文档页面很小,下图可以通过放大 pdf 来查看细节效果

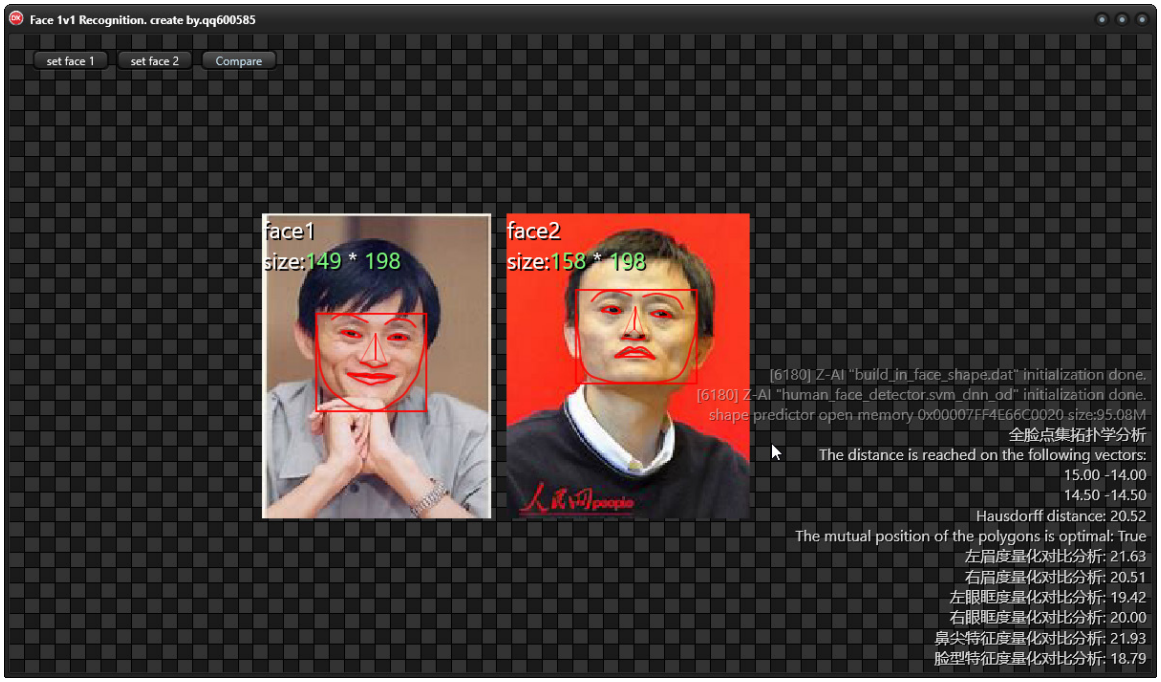


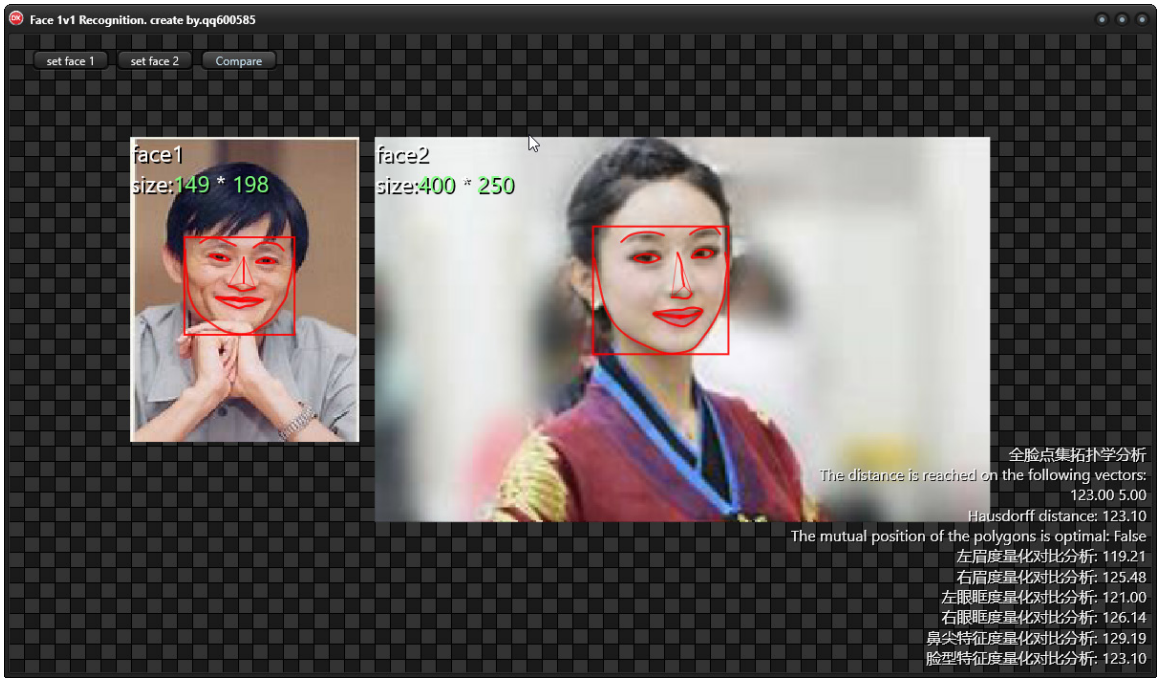
文档页面很小,下图可以通过放大 pdf 来查看细节效果



人证识别 demo

- 该 demo 使用暴力的面部轮廓方式来解决人证识别
- 同时该 demo 也给出了公安局要求人证识别的准确度解决方案预告:在下一版本将给出它的解决方案.
- 因为我手上的事情非常多,同时我一直没有购买小超算,构建模型需要时间,以及设备,我会在 2020 中旬争取购入小超算系统来解决应用模型和数据科学问题.
- 该 demo 作为展望,解决思路已详细备注在代码中.





引擎工具链更新(省略,大规模更新,参考独立文档)

省略.参考独立文档

完
2020-2
By qq600585