OCR Text Detector 算法详解

什么是 Text Detector

文本检测器，顾名思义，用于发现，分割，图片中的文字。只有当我们成功分割好了文字，才能进行下一步的分割结果识别：图像相似性识别，以及，字字之间的正确性处理：通过语料数据库（zChiense），通过 LSTM+语料库等等。

文本检测器，目前有两种方向

1， 光学符号文本检测：这就是我们看到 Raster_DocumentTextDetector 库提供的支持，后面我会详细介绍它的工作原理。

2， 自然场景文本检测：这就是我们熟知的对象检测器+像素分割的组合技术，简称 OD+SS，在已训练好的模型基础上，遍历全图，找出文本框，多边形，几何形，顶点，原始分割图等等，方法非常多。时至 2019，目前最易于做中文自然场景文本检测算法，是由华中科技大学白翔团队提出的多种逆卷积检测法。

以下是白翔团队 2019 顶会摘录的更新 paper，请自行翻阅

Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, X. Bai. TANet: Robust 3D Object Detection from Point Clouds with Triple Attention. AAAI, 2020. (Oral) [code]

M. Liao, Z. Wan, C. Yao, K. Chen, X. Bai. Real-Time Scene Text Detection with Differentiable Binarization. AAAI, 2020. (Oral) [code]

M. Liao, P. Lyu, M. He, C. Yao, W. Hu, X. Bai. Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes. IEEE Trans. on PAMI, to appear. [code]

Z. Zhu, M. Xu, S. Bai, T. Huang, X. Bai. Asymmetric Non-local Neural Networks for Semantic Segmentation. ICCV, 2019. [code]

Z. Zhu, T. Huang, B. Shi, M. Yu, B. Wang, X. Bai. Progressive Pose Attention Transfer for Person Image Generation, CVPR, 2019. (Oral) [code]

B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, X. Bai. ASTER: An Attentional Scene Text Recognizer with Flexible Rectification. IEEE Trans. on PAMI, 41(9): 2035-2048, 2019. [code1][code2]

M. Liao, B. Shi, X. Bai. TextBoxes++: A Single-Shot Oriented Scene Text Detector. IEEE Trans. on Image Proc., 27(8): 3676-3690, 2018. [code]

B. Shi, X. Bai, C. Yao. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. IEEE Trans. on PAMI , 39(11): 2298-2304, 2017. [music score recognition datasets] [code1](Torch) [code2](Pytorch)

OCR Text Detector 的蝴蝶效应

因为 ZAI 的授权用户大多都在对我提出文本，文档，短字符，票据等等方向的文本识别需求，所以我用 Pascal 编写了一个现代化的文本检测->光学文本分割方案。在我编写该方案之前，我对授权用户提供了一个独立文本识别方案：zOCR，这是基于 Tesseract 重构而出文本识别方案，它需要使用 ZAI 自带的形态学+语义分割作为前置处理，同时需要保证 DPI 达到 200 以上（非模糊图 720p 以上）才能应用。在 ZAI 中已经内置了 ZOCR 的独立支持。

后来，用户反应发现识别率不高，且容易出错，这时候，我尝试性的帮助用户去训练自己的文本识别模型，发现 Tesseract 并不适合自由建模，于是，我才用 Pascal 语言编写现代化的文本检测->光学文本分割方案，也就是现在我们看到的这份文档。

本文的 OCR Text Detector 需要 4 个支持模块才做到到 end to end text recognition

1， 针对不同的场景前置化处理：Morphomatics + segmantic segmentation，2 选 1

2， 文本检测与分割：OCR Text Detector，Object Detector，2 选 1

3， 基于深度学习的光栅分类器：ResNet Metric+ GDCNIC+GNIC+Tesseract，4 选 1

4， 后置化处理：zChiense

以上方案组合均能应用，没有什么水分

前置处理：

把白纸黑字的图给抠出来，或则将图片转换成白纸黑字。

可以选择直接 Morphomatics，作用是降噪，配合 TMorphologySegmentation 也能抠图，如果场景噪音非常复杂，且不固定，我们可以选择使用语义分割 segmantic segmentation 来做前置处理

文字检测与分割：

OCR Text Detector 要求标准输入必须是白底黑字，当我们前置处理成功以后，这一步的主要工作就是做文本检测，并且分割出每个字的框体坐标

Object Detector 则是针对非常复杂的场景，比如车牌，门牌号，手写便签，集装箱号这类自然场景，Object Detector 需要收集数据集，训练后我们才能使用。Object Detector 同样也能作为文档的检测器使用

光栅分类器，文字识别：

ResNet Metric 在训练一张光栅图片前，会假想式的拆分式重构图片，然后再告诉深度网络，这是同一个目标，这时候拟合计算会让网络慢慢知道目标的形状，像素构成，与同类图片区别在哪，与不同类类图片区别在哪，最后网络将学会目标图片的特征内容。一般来说，都是通过 cuda 大量反复服训练，也有用 opencl。ResNet Metric 不光是一个分类器，它还有度量化功能，简单来说，度量化可以取消掉分类器上限，达到数百万的分类效果，而中英文字和单词，加在一起也不到一万。训练文本的 Resnet metric 我们需要自己编程实现。

GDCNIC 是 Going Deeper with Convolutions 的缩写，与 Resnet 不同，它更追求训练的速度，当我们的样本达到上万时，GDCNIC 可以在 10 分钟训练出结果，它不会像 resnet 那样去反复重构输入图片，而是暴力的直接进行多尺度学习，每个中文字，都可以形成多种尺度

GNIC 是 Gradient-based learning applied to document recognition 缩写，这是 LeCun 大师在 90 年代设计的神经网络，专用于用于 OCR 的光栅分类处理，ZAI 内置的 GNIC 使用 cuda 重新赋予了它新的生命，它与前两个分类器不同，它在有很多干扰的情况中，仍然可以正确分类，前提是我们给它提供怎样的 input 数据，这取决于我们的文字检测与分割处理环节。

Tesseract 是 90 年代由 HP 领头开发，后续由 Google 负责维护，小规模升级。zOCR 已经内置了重构版本，重构规模不小。如果懒于自己建模，拿 Tesseract 来解决光栅级识别会是不错的选择

后置化处理：

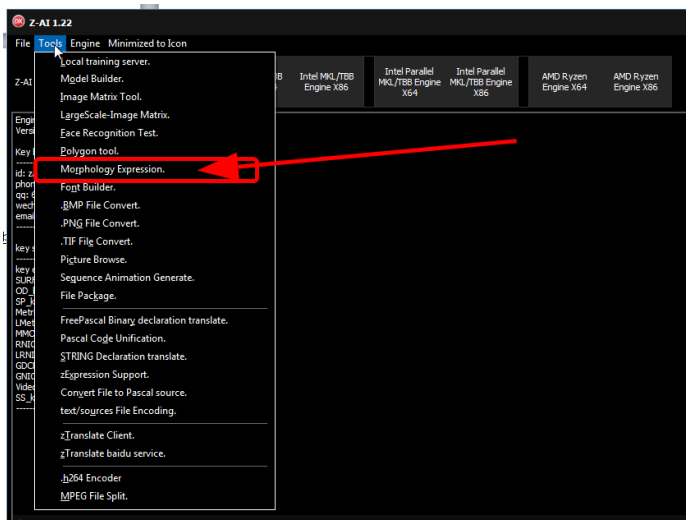
后置化处理在现代化 OCR 中，一般都和文字检测分割，光栅分类精密结合在一起，在前两步处理中就已经包含了分词候选方式这些处理了。我们考虑算法时，将后置化作为独立步骤思路可以更清晰。

另一方面，后置化也可以多少做一些纠错，诸如我们识别的图片本身就有书写错误，这时候后置化的语料库可以帮助我们去纠错。

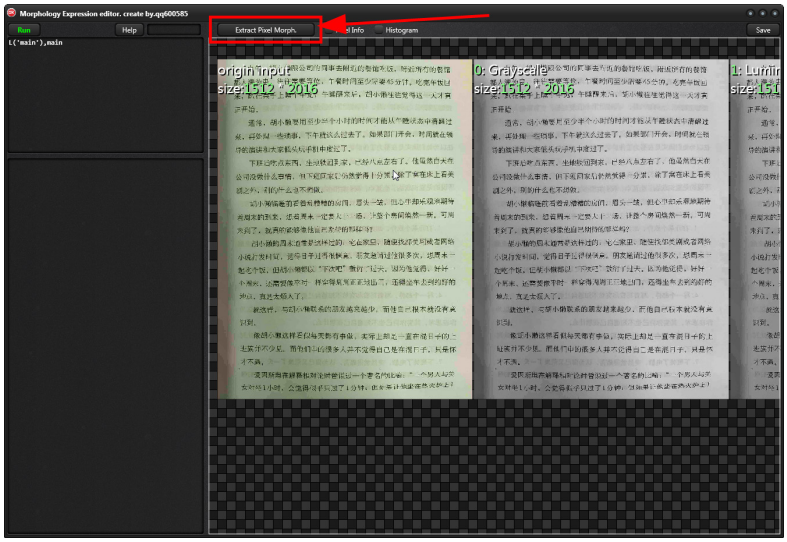
Text Detector 算法实现流程，与思路详解

形态学是在格论和拓扑学基础上衍生出来的数学形态学，它的作用的提取视觉形态，是早期，也是现代化 Compute Vision 的重要技术领域。

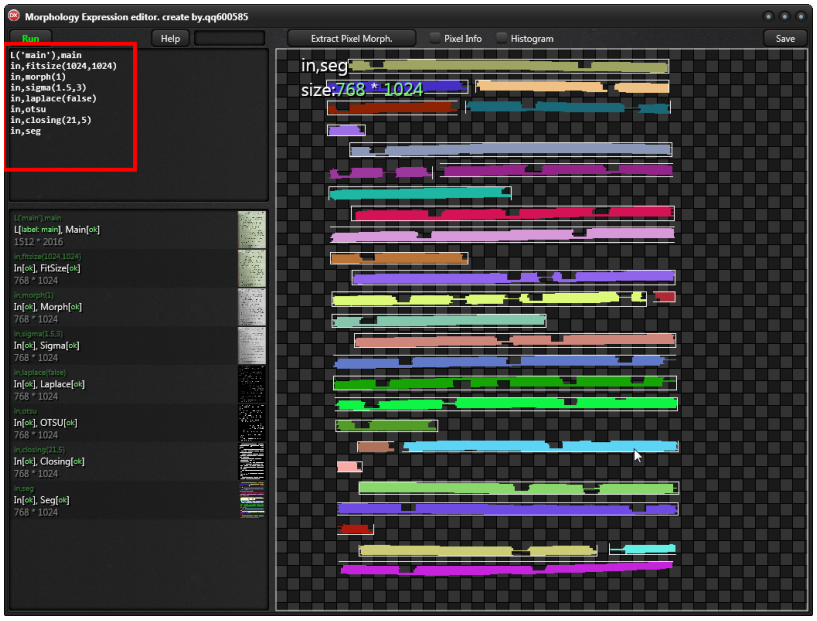
首先我们拿几张图通过试验工具来验证我们的文档内容提取思路
在 Z-AI 的工具链系统中，我们可以找到用于形态学试验的 Morphology Expression 工具，打开它



选择我们的文档图片，展开像素形态，我们会看到 20+种像素形态，如果不理解像素形态，可以根据标注的 Grayscale,Luminace 这类关键字去搜索它的信号含义，我们也可以通过阅读代码来了解像素形态的信号含义。光栅内核的像素形态大致有：YIQ，HIS，CMYK，RGB，近似自定义色，一种 5 种像素形态范围，把这 5 种像素形态一一拆开以后就是我们看见的这 20 种可见的像素形态。这些像素形态可以进行数学操作。



红框中的脚本，都来自于光栅系统和形态学 API，Morphology Expression 工具是用于快速验证算法实现的思路实验工具，Morphology Expression 中的脚本全都来自 MemoryRaster.pas 库所提供的 API，我们可以通过阅读 MorphologyExpression.pas 库来了解脚本锁对应的具体 API。
另外，Morphology Expression 可以给出每一次迭代计算的结果值，一目了然。我们要实现强大的算法，必须先要有强大的支持工具，不过，理论还是需要的。Morphology Expression 只是帮助我们验证我们头脑中的构思的计算方法。



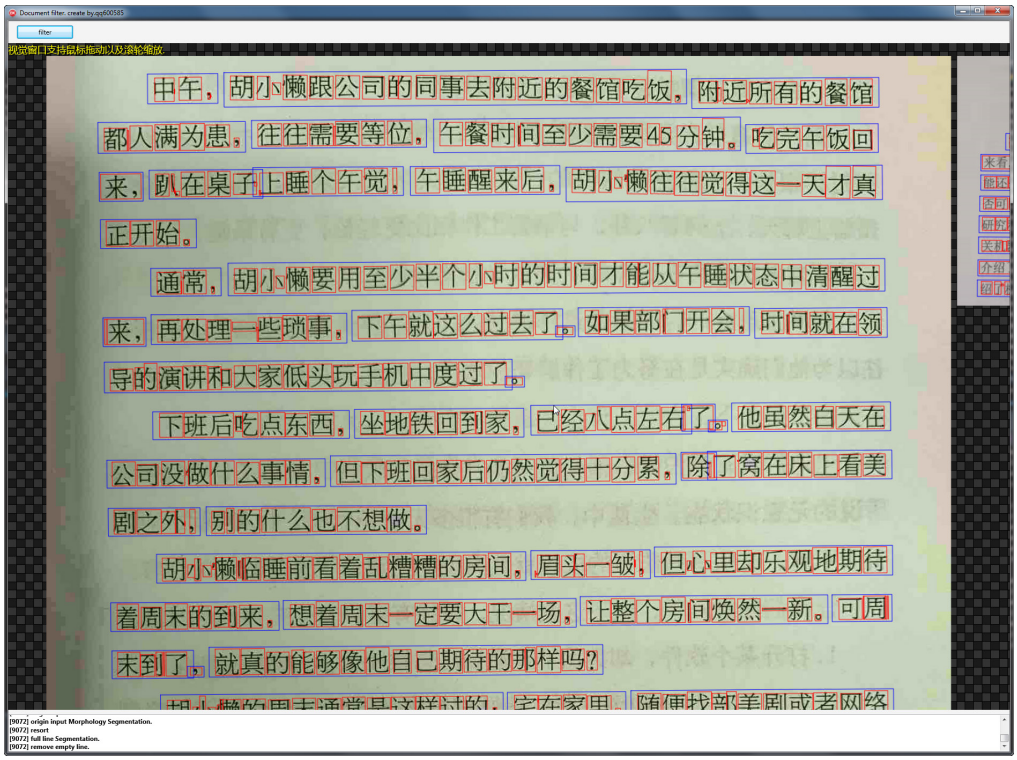
从上图，我们看到，尺度化，取灰度，高斯，拉普拉斯，大律法，分割，通过这几个步骤，我们成功提取出文档中的分割行。
现在，我们的算法思路经过论证可行，我们开始用 MemoryRaster.pas 库的 API 来实现上述算法
思路：先对齐文档，用形态学提取行，然后每行单独提取文字，最后输出各个文字的坐标
在实际工作中，我定义了很多应用级的数据结构，大环节总共 3 个，这 3 环节我均使用 Morphology Expression 来验证，然后再串联起来
下图红框的 pascal 代码就是上图的表达式脚本复现

```
// morphomatics
DoStatus('extract origin input morphomatics YIQ-Y');
with OriCalRaster.BuildMorphomatics(mpYIQ_Y) do
  begin
    DoStatus('origin input SigmaGaussian.');
    SigmaGaussian(opt^.LineSIGMA, opt^.LineSigmaGaussianKernelFactor);
    DoStatus('origin input Laplace.');
    Laplace(False);
    DoStatus('origin input Dilatation.');
    Closing(opt^.LineFinalClosingConvX, opt^.LineFinalClosingConvY);
    DoStatus('origin input Binarization.');
    with Binarization_OTSU do
      begin
        DoStatus('origin input Dilatation.');
        Dilatation(opt^.LineFinalDilatationConvX, opt^.LineFinalDilatationConvY);
        DoStatus('origin input Morphology Segmentation.');
        Seg := BuildMorphologySegmentation();
        free;
      end;
    free;
  end;

// extract line of pixel bounds
for i := 0 to Seg.PoolCount - 1 do
  begin
    r1 := Seg[i].BoundsRectV2;
    if OrientCalibrateRect4.InHere(r1) then
      begin
        r2 := RectProjection(OriCalRaster.BoundsRectV20, CalOri.BoundsRectV20, r1);
        if (RectWidth(r2) > opt^.LineSegmentationWidthSuppression) and (RectHeight(r2) > opt^.LineSegmentationHeightSuppression) then
          LDataList.AddLineData(r2, CalOri.BuildAreaCopy(r2));
      end;
  end;
DisposeObject(Seg);
```

实现算法中间过程省略，靠各自的理解来做。

最后，这是我们做好的可以应用的算法库运行结果，蓝色是行分割结果，红色是字分割结果



关于字分割，大致思路都是用表达式验证算法，唯一不同之处是使用了斜线命中率来计数，就是下图，它的作用是计算出各个的不规则包围盒，斜线命中计数，最后算出 document projection box。

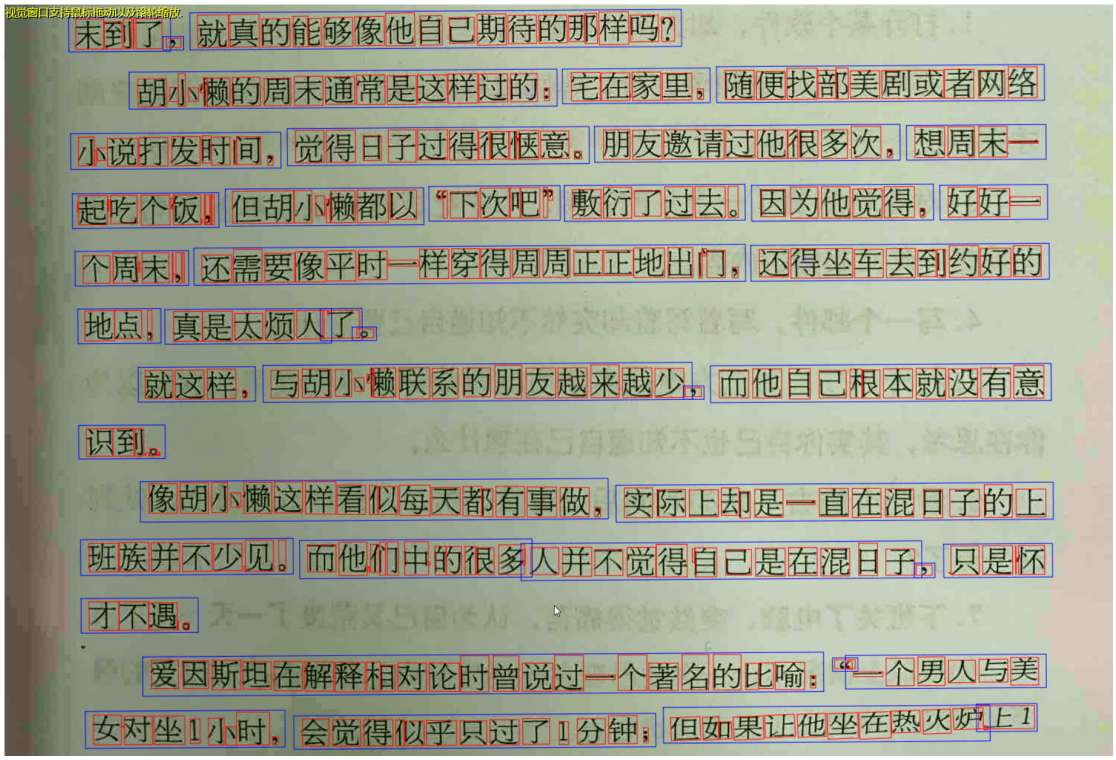
```
// extract as 1D space
SetLength(L1DBuff, calBin.Width);
for i := 0 to calBin.Width - 1 do
    L1DBuff[i] := calBin.LineHitSum(i, 0, i, calBin.Height - 1, True, True);
found := False;
R := RectV2(0, 0, 0, calBin.Height0);
for i := 0 to length(L1DBuff) - 1 do
  begin
    if (not found) and (L1DBuff[i] > 0) then
      begin
        R[0, 0] := if_(i > 0, i - 1, i);
        found := True;
      end
    else if found and (L1DBuff[i] = 0) then
      begin
        R[1, 0] := i;
        found := False;
        // compute top line
        for j := 1 to calBin.Height - 2 do
          if calBin.LineHitSum(Round(R[0, 0]), j, Round(R[1, 0]), j, True, True) > 0 then
            begin
              R[0, 1] := if_(j > 1, j - 1, j);
              break;
            end;
        // compute bottom line
        for j := calBin.Height - 2 downto 1 do
          if calBin.LineHitSum(Round(R[0, 0]), j, Round(R[1, 0]), j, True, True) > 0 then
            begin
              R[1, 1] := if_(j < calBin.Height - 2, j + 1, j);
              break;
            end;
        // extract area
        if (RectWidth(R) > opt^.WordSegmentationMinWidth)
          and (RectHeight(R) > opt^.WordSegmentationMinHeight)
          and (calBin.BoxHitSum(R, True) > opt^.WordSegmentationHitSum) then
          begin
            R := RectDiv(R, Scale);
            LinePtr^.WordList.AddWordData(R, LinePtr^.LineCalibrateRaster.BuildAreaCopy(R));
          end;
      end;
  end;
```

由于忙于做版本，很多字分割以后，会有多个部分组合，这也是字分割的难点，或则说，传统检测技术瓶颈，关于这个问题，最好的解决办法是根据不同场景，去做不同的分割框抑制，我们通过 NMS 这种关键字搜索非最大值抑制，这类算法很简单，但是并不通用于所有文字检测。

这时候，我们任然可以做很多 OCR 组合搭配，我们可以把每行分割出来，然后用 zOCR+Tesseract 识别单独行，我们可以给 zOCR 输入无噪音图，这样可拥有超好的识别正确率。
同样的，我们也可以把每个字抠出来，交给 zAI 的分类器去识别这是哪个汉字。

总结：Text Detector 提供了将识别好的文字坐标逆投影到原始文档的功能，我们根据坐标，可以非常方便的重构出新的图片，然后再拿去做组合识别

在本文结束的时候，展示一下 Text Detector 对各种文档的分割效果，如果使用 PDF 阅读器，可以放大查看



By.qq600585