SyntText数据融合总结

1. 背景

参考论文《Synthetic Data for Text Localisation in Natural Images》，此论文提出了一种新的方法将文本自然的排列进自然场景图片中。此项融合方法的好处在于，它可以准确知道文字的信息以及位置，不需要人工取标记。这样自动生成的数据可用于训练自然场景文字的检测与识别。

1. 融合方法
2. 论文中提供的融合方法需要原始图片、深度图、分割

图以及分割域的label和area信息。



图1 融合准备

数据融合所需要的素材：

1）原始图片：自然场景图片，不能出现数字以及文字（英文字母和汉字等文字）。

2）原图所对应的深度图：获得方法见三。

3）原图所对应的分割图：获得方法见四。

4）文本：需要融合文字的文本。（注：需要相应的字体，本文默认为英文字体，如果需要融合其他文字，需要相应的文字字体，融合汉字方法详见《SynthText数据融合方法 - 融合汉字总结》）

融合流程如下图：

拟合平面

融合文本

分割数据

深度数据

原始图片

文本数据

计算平面矩阵调整大小

过滤分割区域

融合图片&H5文件

图2 融合流程

图2为融合流程，以下对流程进行详细说明：

1）过滤：根据分割数据中的分割区域信息，首先过滤掉面积小于100的区域，然后过滤掉长、宽小于30的区域，最后再进行过滤，过滤条件为长宽比为0.3~7，不满足此条件过滤掉。剩下为候选区域。

2）拟合平面：对候选区域和区域所对应的深度数据为原始数据，对每一个候选区域用RANSAC方法进行平面拟合，得到平面的发矢量。判断发矢量的Z轴分量，如果小于0.25，则表示该区域不适合填充文本。

3）计算：对通过筛选的区域进行计算，得到区域拟合平面的法矢量旋转到xy平面的旋转矩阵。以候选区域的矩形中心为中心，对该区域矩形框进行旋转，得到旋转后的平行四边形，再对其进行缩放，使其不超出该区域边界。得到缩放后的平行四边形，再进行长、宽筛选,小于20的平行四边形进行过滤。

4）文本融合：根据候选区域大小，随机挑选文本进行大小匹配，最后将嵌入的文本以平面矢量进行旋转，最终得到融合的文本。再次计算融合后文本的矩形框和每个字符的矩形框，并保存。

5）输出：最终输出文本融合的图片、嵌入文本的矩形框坐标、单个字符的矩形框坐标，以及文本信息，形成合成数据。

1. 程序运行：

github地址：<https://github.com/ankush-me/SynthText。作>者提供了融合准备数据dataset.h5,用户也可自己准备要融合的数据，以H5文件的形式替换上述文件。然后运行gen.py（注：由于程序中有些条件判断不完善，本文对其程序进行修改），结果会自动生成在results文件夹中，以H5格式保存。

运行结果：

图3 原图 图4 合成图

1. 深度图方法

根据FCRN得到图片的深度信息。github地址：

[https://github.com/iro-cp/FCRN-DepthPrediction。](https://github.com/iro-cp/FCRN-DepthPrediction。   运行)

[运行](https://github.com/iro-cp/FCRN-DepthPrediction。   运行)predict\_test.py，其中需要处理图片的路径和处理结果图片的路径都在脚本中有设置，用户可自行设置保存路径

注：运行结果的图片大小固定为160\*128；需将此结果resize调整到原始图片大小,此步骤已在运行脚本中处理。

运行结果：



图5 深度图

1. 分割图方法
2. 作者提供了两种分割方法，分别为Holistic Edge

Detection。github地址: https://github.com/s9xie/hed。运行结果发现，此方法为图像的边缘提取，不能满足本文融合方法所需要的分割图片，所以弃用此结果

1. 另一种方法为[Multiscale](https://github.com/jponttuset/mcg) Combinatorial

Grouping。

由于上述方法中的最终结果为将图片根据模型分割成很多区域（约2000个），其中存在大量重复区域，本次融合所需要的分割图为完全分割原图，不能有重复出现。所以考虑需求，本文按照要求，对上述方法进行修改，提取中间结果，再进行后续处理，最终得到符合融合要求的分割图。

1. 分割方法：

首先根据[Multiscale](https://github.com/jponttuset/mcg) Combinatorial Grouping方法对图片进行边缘提取，根据模型，生成一张全

零的灰度图，其中用0~1之间的数将图形区域进行分割，数值的大小代表相邻两个区域的不相似度，数值越大不相似程度越大，见下图。



图6 分割预处理图

本文对其结果进行后续处理，最终得到分割图。

首先，给定一个阈值thread对其进行合并分割(注：保证分割后的区域在255以内，如阈值不满足，程序会提示重新修改阈值)；

其次，对分割后的图片进行连通区域信息计算，得到区域面积，像素位置，像素个数，矩形框等信息。其中包括，一个区域完全包围另一个区域；

再次，处理上述完全包围区域，将其过滤掉；

最后，对过滤后的区域进行赋值，每个区域赋予相同的灰度值，赋值规律为图片从上到下、从左到右依次增大。

1. 运行程序：

github地址https://github.com/jponttuset/mcg。运行full/seg\_image文件中的Split\_image\_batch.m脚本，运行结果为未完全分割图片。本文对其结果进行了分割处理以及label和area信息的提取，最终生成达到要求的分割图片和相关信息。上述方法在Split\_image\_batch.m和seg\_image.m脚本中实现。

运行结果：

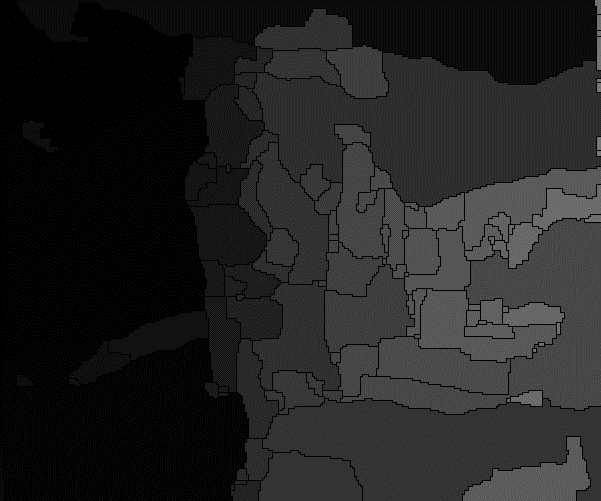


图 7 分割图