(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 112242002 A
(43)申请公布日 2021.01.19

(21)申请号 202011075418.0
(22)申请日 2020.10.09
(71)申请人 同济大学
地址 200092 上海市杨浦区四平路1239号
(72)发明人 刘儿兀 陈铭毅
(74)专利代理机构 上海科律专利代理事务所(特殊普通合伙) 31290
代理人 叶凤
(51)Int.Cl.
G06T 17/05(2011.01)
G06T 15/00(2011.01)
G06T 3/40(2006.01)
G06N 3/04(2006.01)
G06N 3/08(2006.01)
G06K 9/32(2006.01)
G06F 16/583(2019.01)
G06F 16/587(2019.01)

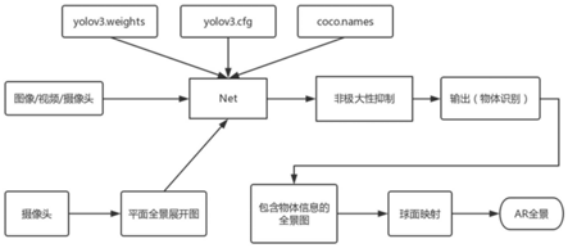
权利要求书3页 说明书7页 附图2页

(54)发明名称

基于深度学习的物体识别和全景漫游方法

(57)摘要

一种基于深度学习的物体识别和全景漫游方法，涉及计算机视觉技术和全景漫游领域。本发明考虑解决现有技术中所存在技术方面的问题，尤其是信息采集与全景漫游构建的相互分离问题。采用高效的物体检测模型，同时结合可用于多平台部署插件来构建具有丰富物体信息的全景漫游模型。采用端到端的设计方案，只要输入所拍摄的全景图，就可以直接得到最终模型，省去了采集、标定信息的繁琐步骤。由于目前智能手机自带全景拍摄功能，本发明可以让大众参与全景漫游模型的构建。结合地图和定位，人们可以分享自己所构建的全景模型，打造一个共享的三维地图模型。



CN 112242002 A

1.一种基于深度学习的物体识别和全景漫游方法,其特征是,包括步骤

步骤1、应用场景数据可通过人工采集或者使用开源的数据集;

步骤2、构建物体识别特征提取网络,利用步骤1中所采集的数据集或者所用开源的数据集来训练物体识别网络;

首先对所采集的数据集都缩放到416*416的大小,然后进行人工的标注,标记图片中所含物体的四个边界框的坐标信息和所属类别,同时记录数据集的总类别数为classes;然后把数据集输入到DarkNet-53模型中;对DarkNet-53进行改造:将网络末尾的全连接层去掉,同时建立三个尺度的输出,最大的尺度是原始图像的32倍下采样,中等尺度是原始图像的16倍下采样,小尺度是原始图像的8倍下采样;三个尺度是不独立的,为了融合高低层网络的特征,将大尺度特征上采样后与中尺度进行张量拼接(concat),接着对中尺度特征进行上采样后与小尺度特征进行张量拼接;将三个尺度的输出最后送入检测层进行识别;

步骤3、构建最终检测层:

步骤3.1,要先对边界框进行预测;

边界框预测算法借鉴了Faster R-CNN中锚框机制;锚框的大小与比例是人为选定的,采用的锚框的大小和比例是根据数据集的边界框数据进行维度聚类得到的;

对步骤2中标定过的数据集进行K-mean聚类来得到最接近真实边界框的锚框;对于边界框,采用如式(2.1)所示的距离度量,其中box为锚框,centroid为真实边界框,d为锚框和真实边界框的距离度量,IOU为交并比;

$$d(box,centroid)=1-IOU(box,centroid) \quad (2.1)$$

将要预测的宽和高分别与锚框的宽和高绑定,经过多次训练后,每个边界框就学习到了如何选择合适形状的边界框;上述所提到的绑定关系为:

$$b_w = a_w e^{t_w} \quad (2.2)$$

$$b_h = a_h e^{t_h} \quad (2.3)$$

其中a_w和a_h分别为锚框的宽和高,t_w和t_h为边框回归直接预测出来的宽与高,b_w和b_h为经过公式转换后所预测的宽和高,也就是网络最后输出的宽和高;从公式(2.2)和(2.3)中可以看到,损失函数经过求导后还保留有t_w以及t_h这两个参数;

接下来就是最后输出的边界框的中心位置(b_x,b_y)的计算公式,如公式(2.4)和(2.5)所示:

$$b_x=\sigma(t_x)+c_x \quad (2.4)$$

$$b_y=\sigma(t_y)+c_y \quad (2.5)$$

其中c_x和c_y是网格左上角点相对整张图片的坐标,$\sigma(t)$是sigmoid激活函数;边框回归预测的t_x和t_y是相对网格而言的,是相对坐标,为了得到绝对坐标,使用式(2.4)和(2.5)来转化;sigmoid激活函数是为了把t_x和t_y映射到(0,1)区间,让模型更快收敛;得到了边框回归所输出的四个值b_x,b_y,b_w,b_h;

在训练中,将真实边界框的四个参数用公式(2.2),(2.3),(2.4),(2.5)的逆运算转化为与t_x,t_y,t_w,t_h相对应的g_x,g_y,g_w,g_h,然后进行误差计算,最后计算经过sigmoid激活函数后的值;详细步骤如公式(2.6)到(2.9)所示:

$$\sigma(\hat{t}_x) = g_x - c_x \quad (2.6)$$

$$\sigma(\hat{t}_y) = g_y - c_y \tag{2.7}$$

$$\hat{t}_w = log\left(\frac{g_w}{a_w}\right) \tag{2.8}$$

$$\hat{t}_h = log\left(\frac{g_h}{a_h}\right) \tag{2.9}$$

其中σ(t)是sigmoid函数，a_w和a_h分别为锚框的宽和高，c_x和c_y分别为网格左上角点相对整张图片的坐标，g_x，g_y，g_w，g_h为t_x，t_y，t_w，t_h为逆运算所得到的结果；

步骤3.2，边界框定位后进行分类；

类别的数量取决于训练数据集的类别数量，每个边界框都要计算所有类别的条件类别概率；分类器采用的是逻辑回归(logistic regression)方法；

有了边框信息和类别概率后需要进行反向传播，采用和方差(sum-squared error，SSE)作为损失函数进行反向传播；损失函数如公式(2.10)所示，其中s^2为特征图中网格的数量，B为每个网格负责预测的锚框数量，σ(t)是sigmoid函数，a_w和a_h分别为锚框的宽和高，t_x，t_y，t_w，t_h为公式(2.6)，(2.7)，(2.8)，(2.9)进行运算所得到的结果，C_i^j表示第i个网格的第j个预测边界框的置信度水平，不等同于公式(2.10)中同名参数，$p_i^j(c)$表示第i个网格的第j个预测边界框c类别的条件概率值，Π_{ij}^{obj}和G_{ij}为控制参数；

$$\begin{aligned} Loss = & \sum_{i=0}^{S^2}\sum_{j=0}^{B} \Pi_{ij}^{obj}\left[\left(\sigma(t_x)_i^j - \sigma(\hat{t}_x)_i^j\right)^2 + \left(\sigma(t_y)_i^j - \sigma(\hat{t}_y)_i^j\right)^2\right] + \\ & \sum_{i=0}^{S^2}\sum_{j=0}^{B} \Pi_{ij}^{obj}\left[\left(t_{w_i}^{\ j} - \hat{t}_{w_i}^{\ j}\right)^2 + \left(t_{h_i}^{\ j} - \hat{t}_{h_i}^{\ j}\right)^2\right] + \\ & \sum_{i=0}^{S^2}\sum_{j=0}^{B} G_{ij}\left(C_i^j - \hat{C}_i^j\right)^2 + \\ & \sum_{i=0}^{S^2}\sum_{j=0}^{B}\sum_{c\in classes} \Pi_{ij}^{obj}\left(p_i^j(c) - \hat{p}_i^j(c)\right)^2 \end{aligned} \tag{2.10}$$

在公式(2.10)中看到每个锚框都有一对置信度和条件类别概率；$\hat{C}_i^j$是由某个网格的锚框是否负责预测某个对象决定的；如果负责预测某个对象，那么$\hat{C}_i^j$等于1，反之$\hat{C}_i^j$等于0；当某个边界框不负责预测真实边界框，但是又与真实边界框之间的交并比数值大于预先设定的阈值时，为了不让此边界框影响最终的损失值，需设置控制参数G_{ij}等于0，反之设置G_{ij}等于1；若某个网格的某个锚框没有负责预测某个对象，为了让这种锚框不影响最终的损失值，引进参数Π_{ij}^{obj}，当某个锚框负责预测某个真实边界框时，Π_{ij}^{obj}等于1，否则Π_{ij}^{obj}等于0；然后将数据集通过整个网络进行更新权重参数，得到收敛后的物体识别模型；

然后将数据集通过整个网络进行更新权重参数，得到收敛后的物体识别模型；

步骤4、构建全景模型：

首先创建一个html文件，添加three.js库依赖，建立一个场景，

然后在场景下放置透视摄像机(PerspectiveCamera),用来观测场景中的各种物体以及光源;

然后构建一个渲染器将内容渲染到页面上;

接着设立一个视角控制器来控制视角的移动;

上述准备工作完成后,导入经物体识别程序后的平面全景图,使用此全景图来创建纹理,创建一个球体用来映射,忽略场景中的光照,创建一个MeshBasicMaterial材质,生成网格,用MeshBasicMaterial材质来渲染mesh内表面,并显示在场景中,最后进行循环渲染,设置窗口监听来设置交互系统;这样得到全景漫游模型,使用浏览器进行解析就能观测到模型。

2.如权利要求1所述的方法,其特征是,步骤3中,边框回归算法会得到一堆候选框,需要进行非极大值抑制选择与真实边界框之间有着最大IOU的边框;与训练过程不同的是,在使用网络进行预测时,真实边界框是不存在的,也就无法使用IOU,此时需使用置信度来替代IOU进行筛选;

置信度一方面用来表征当前边界框是否存在对象的概率$P_r(Object)$;而另一个方面表征在边界框存在对象的情况下,最终所预测的边界框与真实边界框之间的IOU_{prep}^{truth},这里表征的是模型对于边界框框出了物体的一种自信程度;这样得到公式(2.11),其中C_i^j表示第i个网格的第j个预测边界框的总体置信度;在识别过程中采用C_i^j来筛选具有最高置信度的边界框,这样就解决了识别过程中无法计算IOU的问题;

$$C_i^j = P_r(Object) * IOU_{prep}^{truth} \qquad (2.11)$$

。

基于深度学习的物体识别和全景漫游方法

技术领域

[0001] 本发明涉及计算机视觉技术和全景漫游领域。

背景技术

[0002] 随着近几年深度学习算法的高速发展,计算机视觉得到了非常快速的发展。物体识别是计算机视觉领域中的一项基础研究,它的任务是识别出图像中有什么物体,并报告出这个物体在图像表示的场景中的位置和方向。现如今工业上的人机交互应用,如AR、机器人等,首要的问题是正确地认识所处的环境,而物体识别就是这些应用理解环境的核心关键。

[0003] 传统的物体识别方法通常是通过提取图像中的一些鲁棒性特征,例如Haar、SIFT、HOG等,使用DPM模型,用滑动窗口的方式来预测具有较高置信度的边界框,最后送入如SVM等类型的分类器中进行分类。这种方法的缺点是它采用单一的模板去定义物体,如果只专注于人脸检测的话,效果是不错的,但是对于多元的物体检测,尤其是背景较为复杂的情况下,精度急剧下降。

[0004] 深度学习算法让使得物体识别的技术得到了高速发展。传统的方法需要人为地根据场景和目标去设计合适的图像特征。例如对物体的外观进行建模,就需要基于梯度特征来描述轮廓,同时还需要对梯度信息进行筛选、量化,得到相对稳定的表达。而所有的这些工作,都需要有专门的领域知识去设计和调优。然而特征学习正是深度学习所擅长的部分,它把相关场景和目标的特征学习,转变为网络结构的定义和参数的学习,从而免去领域专家去设计特征这一环节。这样就极大简化了为目标设计合适特征的过程,只需要把原始图片和标注提供给网络,定义好网络结构,就可以自动学习出多层次的特征表达和分类器。

[0005] 目前物体检测分为两个类别,一类是两阶段检测器,将物体识别和物体定位分为两个步骤,分别完成,这一类的典型代表是R-CNN,Fast R-CNN,Faster-RCNN系列。其识别错误率低,漏识别率也较低,但速度较慢,不能满足实时检测场景。另一类为单阶段检测器,典型代表是YOLO系列,SSD等。他们识别速度很快,可以达到实时性要求,虽然早期的单阶段检测器准确度不够,尤其对于小目标的识别效果差,但是随着算法的进步,其准确率也接近了两阶段检测器的水平。而且由于单阶段检测器的耗费的资源低,在一些不那么追求高精度的应用场景下,使用单阶段检测器可以节省许多经费。

[0006] 同时全景技术在近几年高速发展,尤其全景漫游由于其高度可视化的优点,在各行各业都能看到全景的应用,尤其是在名胜古迹的展示上面,让人足不出户就能感受到身临其境的氛围。像基于Web的三维图形全景漫游技术由于其方便性和可交互性,迅速得到人们的认可。但是早期的三维图形技术,比如Flash等技术有着许多缺点,例如占用资源多、性能低且无法跨平台等问题。随着互联网三维技术的快速发展,诞生了WebGL这一技术。WebGL不依赖任何浏览器插件,它使用JavaScript脚本来绘制图像,利用底层硬件进行加速,并且拥有可以进行全平台部署的接口。因此解决了传统技术存在的占用资源多、性能低且无法跨平台等问题。WebGL有丰富的第三方开源框架,例如three.js、CubicVR等,这些框架对底

层的结构进行了很好的封装，通过简单地调用接口就能快速实现三维场景渲染。

[0007] “一种多场景漫游生成方法及装置”(专利申请号:201610520963.3)采用Krpano全景展示插件，结合线上3D加装平台来生成全景展示图，使用预设的平台生成家装全景图中的房间数据来精确定位不同场景中进行连接的热点。这个方法结合了平台数据来进行多全景漫游，虽然较手工标定数据有一定优势，但是仍然存在许多不足。比如所用数据取决于平台数据库，一旦在数据库外就得人为标定，延展性明显不足。

[0008] “一种基于全景影像的界桩信息管理方法”(专利申请号:201710372837.2)采用PTGui软件以及Krpano全景插件来构建全景图，采用photoshop软件将界桩信息处理为透明然后添加到全景图中，可在界桩全景图中表达界桩的属性信息。这个方法将Krpano插件中的动态热点系统与ps软件得到界桩信息相结合，可以动态展示界桩周围的地物信息，具有一定的应用性。但是其采用的全景构建方式需要PTGui软件来合成平面全景图，这对于非拍摄专业人士来说实用性很差。再者使用Photoshop来标定信息，这种人工的方式效率很低。

[0009] 再如“一种企业三维数字地图管理系统”(专利申请号:201020554517.2)是将人工收集的地理位置相关的信息放入数据库，与地图相结合，构建三维地图，通过集合数据库中的各种信息，构造出多用途的三维数字地图。这个方法采集各种信息构建全景漫游图，且与地图相结合，可视化程度较高。但是该方案是将信息采集与全景漫游构建相分离，需要耗费大量人力物力来采集数据并标定在全景图上，显然这种方式是的效率较低。

发明内容

[0010] 发明目的

[0011] 本发明考虑解决现有技术中所存在技术方面的问题，尤其是信息采集与全景漫游构建的相互分离问题。采用高效的物体检测模型，同时结合可用于多平台部署插件来构建具有丰富物体信息的全景漫游模型。采用端到端的设计方案，只要输入所拍摄的全景图，就可以直接得到最终模型，省去了采集、标定信息的繁琐步骤。由于目前智能手机自带全景拍摄功能，本发明可以让大众参与全景漫游模型的构建。结合地图和定位，人们可以分享自己所构建的全景模型，打造一个共享的三维地图模型。

[0012] 技术方案

[0013] 一种基于深度学习的物体识别和全景漫游方法，其特征是，包括如下步骤：

[0014] 步骤1、应用场景数据采集

[0015] 采集数据集的过程可以通过人工来采集也直接使用开源的数据集。

[0016] 步骤2、构建物体识别特征提取网络(图3所示)

[0017] 利用步骤1中所采集的数据集或者所用开源的数据集来训练物体识别网络。

[0018] 首先对所采集的数据集都缩放到416*416的大小，然后进行人工的标注，标记图片中所含物体的四个边界框的坐标信息和所属类别，同时记录数据集的总类别数为classes。

[0019] 然后把数据集输入到DarkNet-53模型中，为了能在各个尺度都有着良好的性能，这里对DarkNet-53进行改造:将网络末尾的全连接层去掉，同时建立三个尺度的输出(如图2所示)，最大的尺度是原始图像的32倍下采样，中等尺度是原始图像的16倍下采样，小尺度是原始图像的8倍下采样。三个尺度是不独立的，为了融合高低层网络的特征，将大尺度特征上采样后与中尺度进行张量拼接(concat)，接着对中尺度特征进行上采样后与小尺度特

征进行张量拼接。将三个尺度的输出最后送入检测层进行识别。

[0020] 步骤3、构建最终检测层

[0021] 首先对边界框进行预测。这里使用的边界框预测算法借鉴了Faster R-CNN中锚框机制。锚框的思想就是反向思维，从顶层先假定每个网格负责的锚框是由原图中某些区域变换而来的，通过增加每个网格的锚框数量就可以解决传统算法中多个物体导致识别出错的问题。Faster R-CNN中的锚框的大小与比例是人为选定的，本发明所采用的锚框的大小和比例是根据数据集的边界框数据进行维度聚类得到的。将这些统计上的先验数据加入到模型中，可以让模型在学习的时候，加快模型的收敛速度。所以要先对步骤2中标定过的数据集进行K-mean聚类来得到最接近真实边界框的锚框。K-means聚类方法通常使用的是欧氏距离函数，会出现最终的聚类结果偏离实际的情况。对于边界框，要关注的是其与真实边界框之间的IOU，为了得到更好的聚类结果，本发明采用如式(2.1)所示的距离度量，其中box为锚框，centroid为真实边界框，d为锚框和真实边界框的距离度量，IOU为交并比。

[0022] d(box,centroid)＝1-IOU(box,centroid) (2.1)

[0023] 通常的边框回归算法是让边框回归算法直接预测实际边界框的宽和高，而本发明将要预测的宽和高分别与锚框的宽和高绑定，这样的话无论一开始边框回归算法所输出的宽和高是怎样的，只要经过转化后就会与锚框的宽和高相关。使用这种方法经过多次训练后，每个边界框就学习到了如何选择合适形状的边界框。上述所提到的绑定关系就是指：

[0024] $$b_w = a_w e^{t_w} \quad (2.2)$$

[0025] $$b_h = a_h e^{t_h} \quad (2.3)$$

[0026] 其中a_w和a_h分别为锚框的宽和高，t_w和t_h为边框回归直接预测出来的宽与高，b_w和b_h为经过公式转换后所预测的宽和高。

[0027] 最后输出边界框的中心位置(b_x,b_y)，如公式(2.4)和(2.5)所示：

[0028] $b_x＝\sigma(t_x)+c_x$ (2.4)

[0029] $b_y＝\sigma(t_y)+c_y$ (2.5)

[0030] 其中c_x和c_y是网格左上角点相对整张图片的坐标，$\sigma(t)$是sigmoid激活函数。边框回归预测的t_x和t_y是相对网格而言的，是相对坐标，为了得到绝对坐标，使用式(2.4)和(2.5)来转化。sigmoid激活函数是为了把t_x和t_y映射到(0,1)区间，让模型更快地收敛。

[0031] 到这为止，得到了边框回归所输出的四个值b_x，b_y，b_w，b_h。在训练中并不直接是用这四个值直接与真实边界框的对应参数求误差。而是将真实边界框的四个参数用公式(2.2)，(2.3)，(2.4)，(2.5)的逆运算转化为与t_x，t_y，t_w，t_h相对应的g_x，g_y，g_w，g_h，然后再进行误差计算。由于sigmoid激活函数在数学上并不存在反函数，无法直接对t_x和t_y进行逆运算，所以最后计算其经过sigmoid激活函数后的值。详细步骤如公式(2.6)到(2.9)所示：

[0032] $$\sigma(\hat{t}_x) = g_x - c_x \quad (2.6)$$

[0033] $$\sigma(\hat{t}_y) = g_y - c_y \quad (2.7)$$

[0034] $$\hat{t}_w = log\left(\frac{g_w}{a_w}\right) \quad (2.8)$$

[0035] $$\hat{t}_h = log\left(\frac{g_h}{a_h}\right) \quad (2.9)$$

[0036] 其中σ(t)是sigmoid函数，a_w和a_h分别为锚框的宽和高，c_x和c_y分别为网格左上角点相对整张图片的坐标，g_x，g_y，g_w，g_h为t_x，t_y，t_w，t_h为逆运算所得到的结果。

[0037] 边框回归算法会得到一堆候选框，应进行非极大值抑制选择与真实边界框之间有着最大IOU的边框。为了解决实际应用过程中网络进行预测时并不存在真实边界框的问题，本发明使用置信度来表征。置信度一方面可以用来表征当前边界框是否存在对象的概率P_r(Object)；而另一个方面可以表征在边界框存在对象的情况下，最终所预测的边界框与真实边界框之间的IOU_{prep}^{truth}，在这里提到的真实边界框并不是客观存在的，这里表征的是模型对于边界框框出了物体的一种自信程度。这样可以得到公式(2.10)，其中C_i^j表示第i个网格的第j个预测边界框的总体置信度。所以本发明在识别过程中采用C_i^j来筛选具有最高置信度的边界框，解决了识别过程中无法计算IOU的问题。

[0038] $$C_i^j = P_r(Object) * IOU_{prep}^{truth} \tag{2.10}$$

[0039] 边界框定位后进行分类。类别的数量取决于训练数据集的类别数量，每个边界框都要计算所有类别的条件类别概率。传统的分类模型最后使用softmax分类器，使用softmax分类器来进行分类时，各个类别之间是互斥的，无法解决多标签的问题，也就是说无法对数据进行很好的拟合。而本发明最后的分类器采用的是逻辑回归(logistic regression)方法，而不是用softmax分类器。

[0040] 有了边框信息和类别概率后进行反向传播，本发明采用和方差(sum-squared error，SSE)来作为损失函数进行反向传播。损失函数如公式(2.11)所示，其中s^2为特征图中网格的数量，B为每个网格负责预测的锚框数量，σ(t)是sigmoid函数，a_w和a_h分别为锚框的宽和高，t_x，t_y，t_w，t_h为公式(2.6)，(2.7)，(2.8)，(2.9)进行运算所得到的结果，C_i^j表示第i个网格的第j个预测边界框的置信度水平，不等同于公式(2.10)中同名参数，$p_i^j(c)$表示第i个网格的第j个预测边界框c类别的条件概率值，Π_{ij}^{obj}和G_{ij}为控制参数。

[0041]
$$\begin{aligned} Loss = &\sum_{i=0}^{S^2}\sum_{j=0}^{B}\Pi_{ij}^{obj}\left[\left(\sigma(t_x)_i^j-\sigma(\hat{t}_x)_i^j\right)^2+\left(\sigma(t_y)_i^j-\sigma(\hat{t}_y)_i^j\right)^2\right]+\\ &\sum_{i=0}^{S^2}\sum_{j=0}^{B}\Pi_{ij}^{obj}\left[\left(t_{w_i}^{\ j}-\hat{t}_{w_i}^{\ j}\right)^2+\left(t_{h_i}^{\ j}-\hat{t}_{h_i}^{\ j}\right)^2\right]+\\ &\sum_{i=0}^{S^2}\sum_{j=0}^{B}G_{ij}\left(C_i^j-\hat{C}_i^j\right)^2+\\ &\sum_{i=0}^{S^2}\sum_{j=0}^{B}\sum_{c\in classes}\Pi_{ij}^{obj}\left(p_i^j(c)-\hat{p}_i^j(c)\right)^2 \end{aligned} \tag{2.11}$$

[0042] 在公式(2.11)中可以看到每个锚框都有一对置信度和条件类别概率。$\hat{C}_i^j$是由某个网格的锚框是否负责预测某个对象决定的；如果负责预测某个对象，那么$\hat{C}_i^j$等于1，反之$\hat{C}_i^j$

等于0;当某个边界框不负责预测真实边界框,但是又与真实边界框之间的交并比数值大于预先设定的阈值时,为了不让此边界框影响最终的损失值,需设置控制参数G_{ij}等于0,反之设置G_{ij}等于1;若某个网格的某个锚框没有负责预测某个对象,为了让这种锚框不影响最终的损失值,引进参数Π_{ij}^{obj},当某个锚框负责预测某个真实边界框时,Π_{ij}^{obj}等于1,否则Π_{ij}^{obj}等于0。然后将数据集通过整个网络进行更新权重参数,得到收敛后的物体识别模型。

[0043] 然后将数据集通过整个网络进行更新权重参数,得到收敛后的物体识别模型。

[0044] 步骤4、构建全景模型

[0045] 在步骤3得到物体识别模型后,让平面全景图通过物体识别模型就得到了带有物体信息的全景图,然后需要构建全景漫游模型。首先创建一个html文件,添加three.js库依赖,先建立一个场景,然后在场景下放置透视摄像机(PerspectiveCamera),用来观测场景中的各种物体以及光源。然后构建一个渲染器将内容渲染到页面上,紧接着设立一个视角控制器来控制视角的移动。准备工作完成后开始导入通过物体识别程序后的平面全景图,使用此全景图来创建纹理,创建一个球体用来映射,忽略场景中的光照,创建一个MeshBasicMaterial材质,生成网格,并且用MeshBasicMaterial材质来渲染mesh内表面,并且显示在场景中,最后进行循环渲染,设置窗口监听来设置交互系统。这样就可以得到全景漫游模型,使用浏览器进行解析就能观测到模型。

[0046] 作为应用,进一步使用Tomcat来部署本地服务器供远端浏览器访问。在本地的全景漫游模型是HTML文件,所以本发明在本地搭建servlet服务程序,重写doPost方法,然后打开Tomcat服务器,供远端设备使用浏览器进行动态访问,实现了动态交互的功能。

[0047] 步骤1中根据不同应用场景,如果是用于建筑业,那么数据集可以专注家具,门窗等;如果是用于三维地图构建,那么数据集可以专注于各种公共物品,交通标识以及各种商家的logo。数据集采集具有很强的弹性,要注重多类型就多采集数据;要专注某些物品,就只需采集相关的数据集就行。

附图说明

[0048] 图1为本发明总体算法流程图

[0049] 图2为本发明网络的三种尺度输出

[0050] 图3为本发明物体识别模型架构(包含步骤2中详细的特征提取网络和步骤3的最终检测层)

[0051] 图4为本发明预参数模型

具体实施方式

[0052] 以下结合具体实例和附图对本发明技术方案进一步说明。

[0053] 步骤1、应用场景数据采集

[0054] 使用开源的COCO数据集,共有80个类别,也就是classes等于80,覆盖了动物,公共设备,电子设备,家具等各个类别,具有普适性。数据集的图像大小为416*416的RGB图像,无需进行缩放处理。数据集的所有类别信息存放到coco.name文件中,便于步骤2读取使用。

[0055] 步骤2、构建物体识别特征提取网络(如图3)

[0056] 如图4所示,网络的总体配置文件存放在yolov3.cfg中,所以首先读取网络的配置文件yolov3.cfg,重新来构建总体的网络结构。载入网络权重yolov3.weights。同时读取coco.name,把所有类别的类名放在一个列表里面。接着设置网络的输入,输入的大小会决定最终的识别精度和速度,本例将输入图片的高度和宽度都设置为416。到目前为止,初始化一些参数的任务就做完了,下面开始进行物体识别的构建。

[0057] 首先使用OpenCV读取视频流,图片视作是单帧的视频,而摄像头获取的则是源源不断的视频流。一帧一帧地对视频流进行处理,对于每一帧图像的长宽大小与比例进行缩放和裁剪处理使之与预设的网络输入相同。对图像中较长的一边缩放为416,对于较小的一边缩放后长度小于416,采用RGB元组为(128,128,128)的像素点来进行填充,得到网络的标准输入416*416。OpenCV读取图片的色域格式为BGR,将图片的最后一个维度,也就是通道(channel),让通道这维的矩阵从右到左重构矩阵,实现从BGR转化为RGB,满足了Pytorch所需的RGB格式。同时将图片的维度从(H,W,C),也就是(高,宽,通道数)变换为(C,H,W),也就是(通道数,高,宽),然后添加一个批次数量(batch)维度,这样就满足了Pytorch需要的输入形式(B,C,H,W),也就是(批次数量,通道数,高,宽)。最后将图片数值归一化,转化为Pytorch所用的张量格式,整个网络的输入就构建完毕。下一步需要构建网络的输出检测层。

[0058] 步骤3、构建最终检测层

[0059] 将步骤2所得的输入送入主体网络进行前向传播,到了检测层的时候,要在特征图上进行预测。对于大尺度其下采样为32倍,所以输出特征图设置为13*13;对于中尺度其下采样为16倍,所以输出特征图设置为26*26;同时对于小尺度,其下采样倍数为8,则将输出设置为52*52.三个尺度输出不一样,引入一个输出变换函数,来重构三个尺度的输出,将三个尺度的输出结合在一起计算。特征图维度为(批次数量,单个网格的锚框数量*边界框属性的数量,特征图单边网格数量,特征图单边网格数量)。需要将原来的维度变换成(批次数量,锚框总数量,边界框属性的数量)这种形式。有三个尺度所以要转换三次,然后把输出的锚框总数量拼接在一起,得到最终输出。

[0060] 在得到每个边界框的输出之后,进行非极大值抑制。首先将网络的输出中的边界框的坐标属性转换为在原始输入图像中的坐标属性,使用上一段所得到的匹配特征图的锚框进行逆向操作来得到相对于原图的锚框,然后再进行非极大值抑制。在极大值抑制中只需要关注得分最高的那个类别,将边界框中得分低于阈值的边界框去除掉,提取置信度最高的那个类的得分,以及此类别相对应的序号。然后利用OpenCV把具有最大置信度的边界框描绘出来,并且根据具有最大类别概率得分相对应的序号来得到相应的类别。

[0061] 步骤4、构建全景模型

[0062] 首先添加three.js依赖库,添加依赖后先建立一个场景,然后在场景内放一个相机,起到类似人眼的作用,用来观测场景中的各种物体以及光源。本发明使用放置透视摄像机(PerspectiveCamera),实现了近大远小的3D效果,然后构建一个渲染器将内容渲染到页面上,接着设立一个视角控制器,来操控视角的移动,接着导入平面全景图,来创建纹理,紧接着创建一个球体用来映射,忽略场景中的光照,创建一个MeshBasicMaterial材质,生成网格,并且用MeshBasicMaterial材质来渲染mesh内表面,并且显示在场景中,然后进行循环渲染,最后设置窗口监听,用于交互。

[0063] 这样就建立好了一个完整的全景图展示器。

[0064] 步骤5、使用Tomcat来部署本地服务器。

[0065] 首先需要下载好Tomcat，使用IDE打开步骤1～4所构建的全景漫游模型，配置Tomcat环境，创建服务器主程序，继承HTTPServlet，重写一系列抽象方法，打开Tomcat，可让远端浏览器通过本地IP来访问到全景漫游模型。

[0066] 技术效果

[0067] 本发明的全景漫游模型是基于YOLO v3和three.js构建的。在COCO数据集的实例下，物体识别效果良好，在复杂背景下依旧有很好的性能。全景漫游模型生成时间不到一秒，且占用内存低。而且整个模型基于浏览器，可轻易移植到其他平台，在后续还可以与地图进行结合，构造三维语义地图。

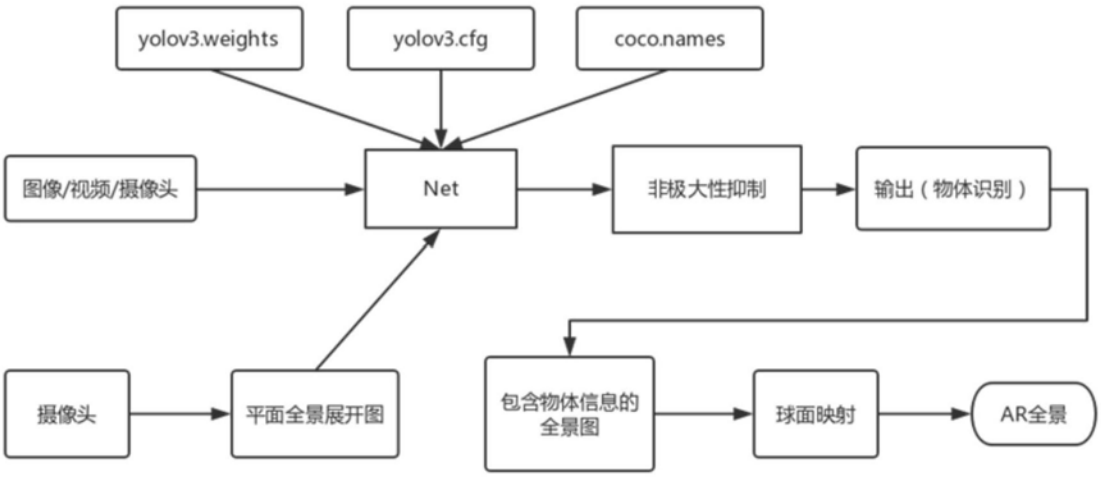

yolov3.weights
yolov3.cfg
coco.names
图像/视频/摄像头
Net
非极大性抑制
输出（物体识别）
摄像头
平面全景展开图
包含物体信息的全景图
球面映射
AR全景

图1

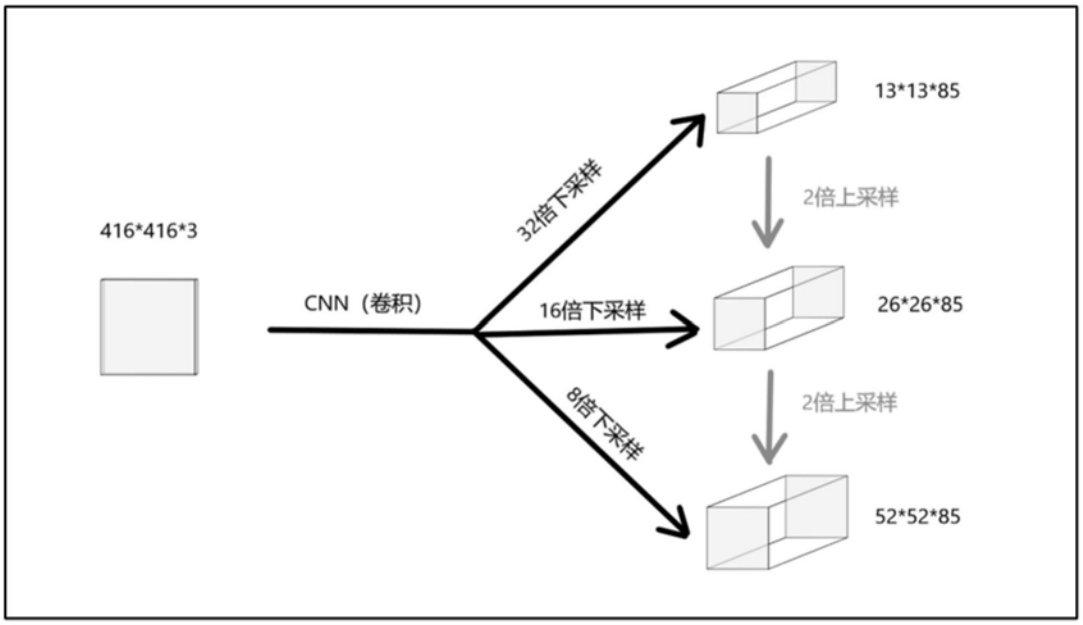

416*416*3
CNN（卷积）
32倍下采样
16倍下采样
8倍下采样
13*13*85
2倍上采样
26*26*85
2倍上采样
52*52*85

图2

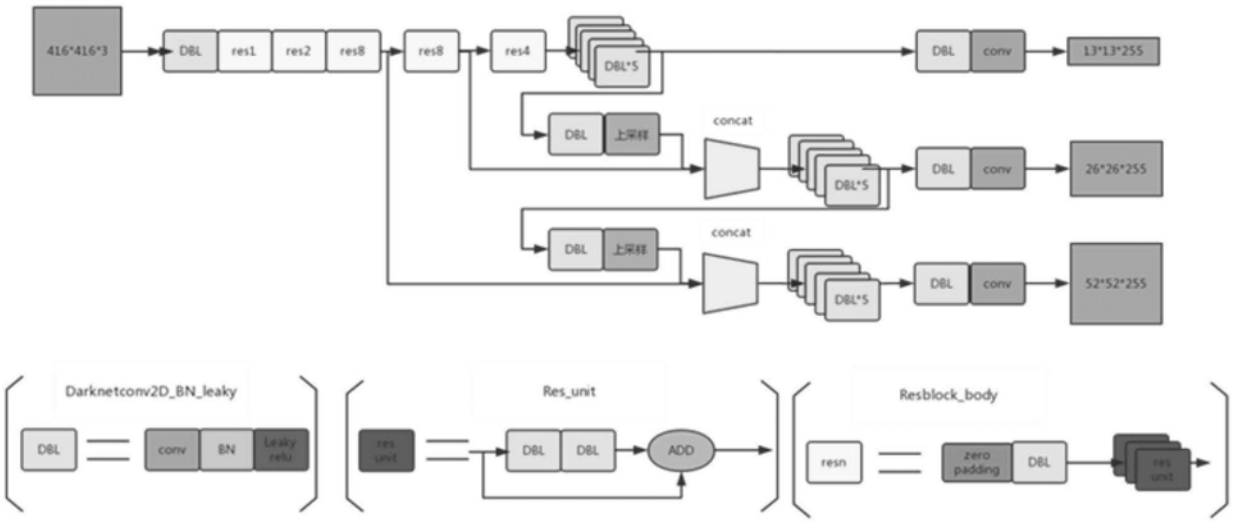

416*416*3
DBL
res1
res2
res8
res8
res4
DBL*5
DBL
conv
13*13*255
DBL
上采样
concat
DBL*5
DBL
conv
26*26*255
DBL
上采样
concat
DBL*5
DBL
conv
52*52*255
Darknetconv2D_BN_leaky
DBL
conv
BN
Leaky relu
Res_unit
res unit
DBL
DBL
ADD
Resblock_body
resn
zero padding
DBL
res unit

图3

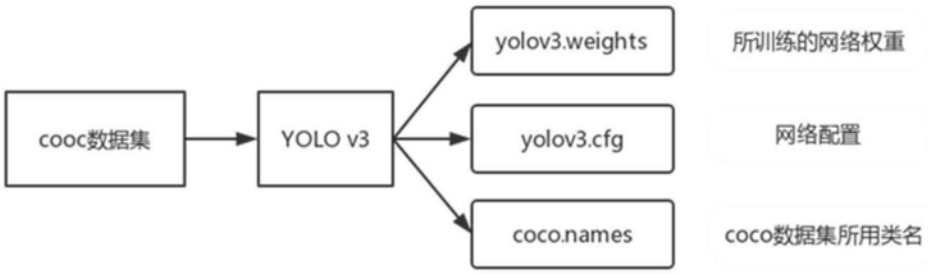

cooc数据集
YOLO v3
yolov3.weights
yolov3.cfg
coco.names
所训练的网络权重
网络配置
coco数据集所用类名

图4