机器视觉大作业汇报

组员： 颜铭，毛睿

分工情况：
颜铭——图像拼接
毛睿——目标检测

目　录

图像拼接

1. 1图像拼接的算法流程

基本算法框架

1.预处理等比例缩放原图像大小以提高运算速度

2.使用**ORB,SURF,SIFT算法进行特征检测，并生成特征描述子和兴趣点**

3.使用**Flann特征匹配算子结合kd树和knn算法**找最近邻和次近邻的匹配关系，再通过赋权几何距离分离，得到性质较好的匹配对期望拟合多边形

4.利用**随机抽样一致RANSC算法,**计算**单应矩阵**进行仿射变换，使用透视转换函数计算视角矩阵再利用视角变换函数将外图像坐标转移到子图视角上

具体拼接融合细节

1.导入图像数据集后根据图像大小二分分别标识左右标签并进入左右队列，对队首元素进行等处理**单批图像匹配连接，视角变换，弯曲拉伸，非连续局部遮罩，图像融合和裁切**后再将结果入队继续搜索符合要求的索引操作。最后得到左右端**分别处理的结果再将左右旋转对齐后多次拉伸平整拼接，**得到拼接图像

2.对于图像拉伸（弯曲）并生成初步连接，在计算单应矩阵并透视坐标变换后整形画布范围的四角投影目标区并利用x,y坐标的极大极小值和**构建模拟拉伸(弯曲)的左对齐3×3**矩阵**控制单应矩阵进行视角变换**。使用极小值约束，对于变换后极小值小于零的顶层优先级高图片分类为左标签反之为右标签。**对于左标签全景图宽由左对齐极小值修正确定而右标签全景图宽估计为原图坐标系宽变换后坐标**，并在已确定范围拼装剪切到初步重整全景图，进一步使用融合函数

3.对于图像融合和裁切结合**经验相关性非连续误差遮罩采用滑动窗口重复局部连续性适应方法并设置较小的先验比值**，再对输入输出的左右相对位置分开处理后旋转使图像拉伸中期望连续区域一致，在遮罩的尺度**加权约束**下进行**线性融合**得到全景图。使用期望的约束全景画布范围的四顶点坐标确定相框为方形区域得到最终拼接图

1.2图像拼接的初步结果和改进

上述拼接原理具有较好的对比融合特性但是也缺点明显，视角线性单一无法很好处理广角图片引起的视差，这就导致了较强的局部畸变和整体违和。

通过思考，这是由于图像拼接方式下的批处理图像截面性质决定的，最中间的视角决定因子最大，问题出在直接的线性投影和分割线不对称拉伸上，考虑在保留截面视角的基础上引进竖直分割对称坐标：利用柱坐标变换解决这一问题

```
while len(left) > 1:
    dst_img = left.pop()
    src_img = left.pop()
    left_pano, _, _, _ = warpTwoImages(src_img, dst_img)#warpTwoImages函数调用融合parnoramaBlending函数实现局部连接
    #left_pano=myutils.stitch_image_pair(src_img,dst_img,1)
    left_pano = left_pano.astype("uint8")
    left.append(left_pano)

while len(right) > 1:
    dst_img = right.pop()
    src_img = right.pop()
    right_pano, _, _, _ = warpTwoImages(src_img, dst_img)
    #right_pano=myutils.stitch_image_pair(src_img,dst_img,1)
    right_pano = right_pano.astype("uint8")
    right.append(right_pano)
```

```
def cylindrical_projection(image):
    I = image.copy()
    height = image.shape[0]
    width = image.shape[1]
    depth = image.shape[2]
    center_x = width / 2
    center_y = height / 2
    alpha = math.pi / 5
    f = width / (2 * math.tan(alpha / 2))
    for i in range(width):
        for j in range(height):
            theta = math.asin((i - center_x) / f)
            point_x = int(f * math.tan((i - center_x) / f) + center_x)
            point_y = int((j - center_y) / math.cos(theta) + center_y)
            for k in range(depth):
                if 0 <= point_x < width and 0 <= point_y < height:
                    I[j, i, k] = image[point_y, point_x, k]
```

图像拼接结果展示和评价

可见视角畸变问题得到了较好的改善，但同时由于中间图片对视角变换决定性因子（极差）大的性质未变，因此为了匹配融合函数的约束项，裁切函数会使得中间图融合对齐效果变差，局部融合损失增大直接导致中央图像的接缝失衡，但水平去抖连贯性较好，可以考虑动态规划计算能量损失函数的最佳接缝优化

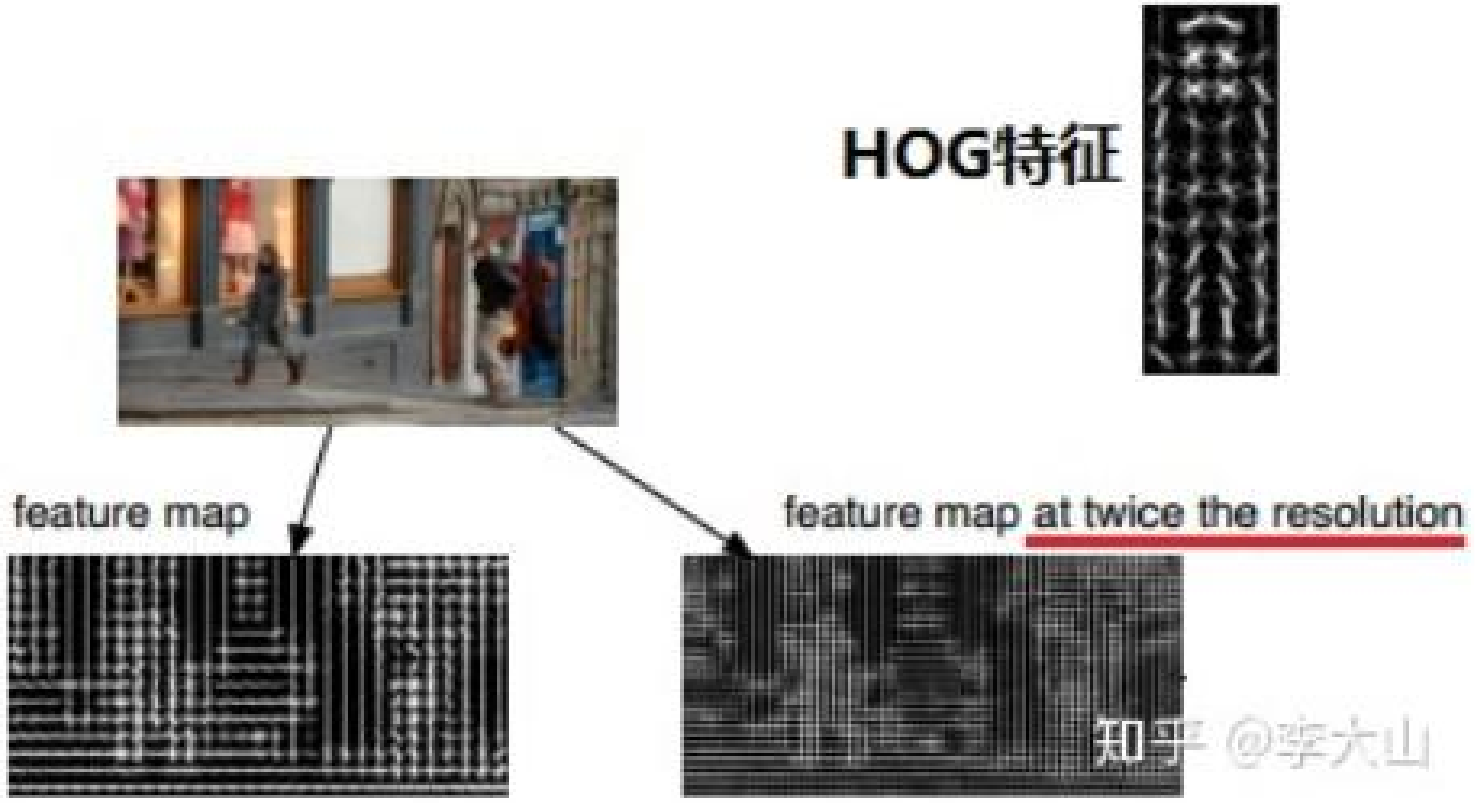
目标检测

2.1 理论：DPM算法

1.算法思想

输入一幅图像，对图像提取图像特征，针对某个物件制作出相应的激励模板，在原始的图像卷积运算，得到该激励效果图，根据激励的分布，确定目标位置。

制作激励模板就相当于 人为地设计一个卷积核，一个比较复杂的卷积核，拿这个卷积核与原图像进行卷积运算得到一幅特征图。比如拿一个静止站立的人的HOG特征形成的卷积核，与原图像的梯度图像进行一个卷积运算，那么目标区域就会被变密集。如右图所示：



DPM算法步骤

1、产生多个模板，整体模板以及不同的局部模板；

2、拿这些不同的模板同输入图像“卷积”产生特征图；

3、将这些特征图组合形成融合特征；

4、对融合特征进行传统分类，回归得到目标位置。

本次实验中采用了已经训练好的model对物体进行检测

算法局限性：首先要有根模型（root filter）和若干部件模型(part filter)和部件模型的偏离损失。这些东西就是通过已有的人体，四肢等样本提取HOG特征然后经过svm训练而来的。

DPM算法步骤

1、产生多个模板，整体模板以及不同的局部模板；

2、拿这些不同的模板同输入图像“卷积”产生特征图；

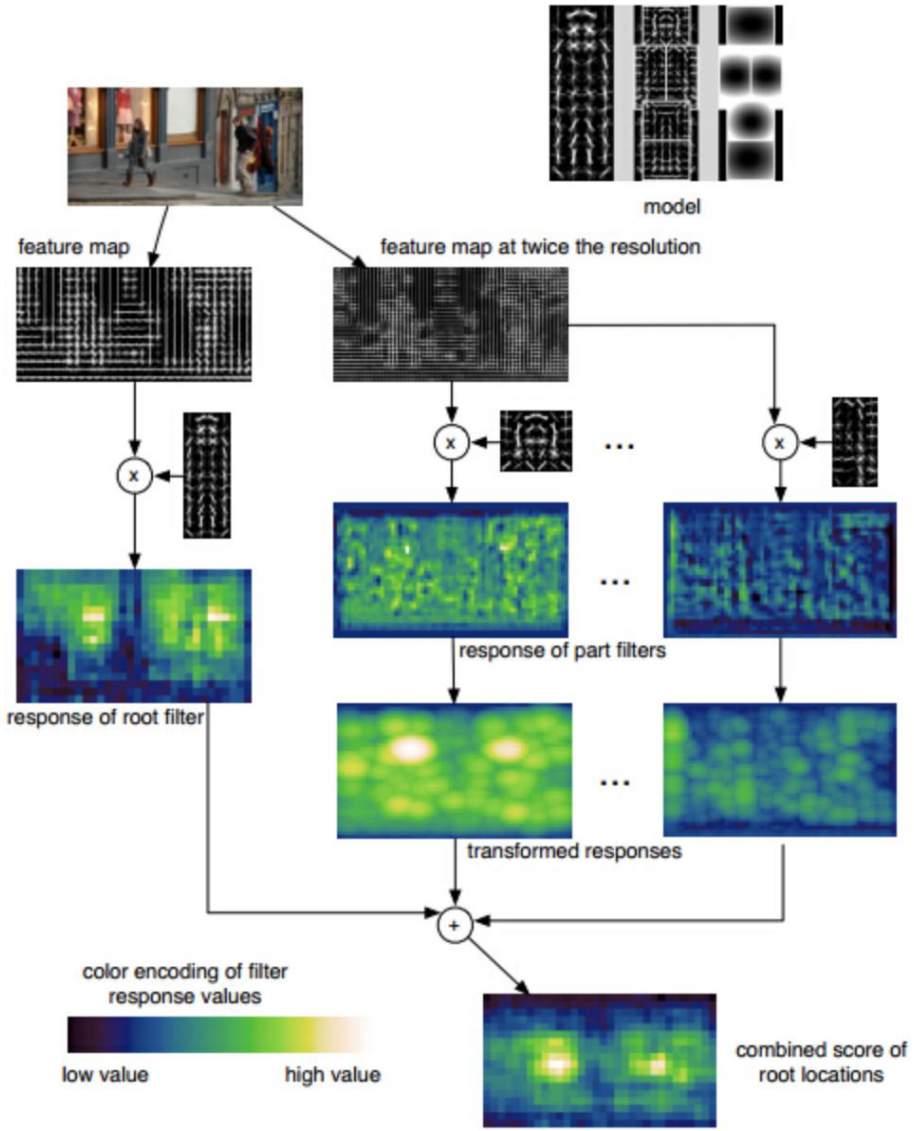
3、将这些特征图组合形成融合特征；

4、对融合特征进行传统分类，回归得到目标位置。

本次实验中采用了提前训练好的model对物体进行检测

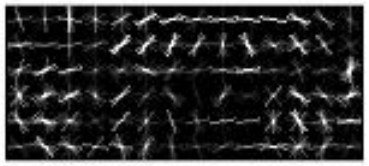
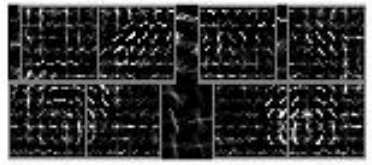
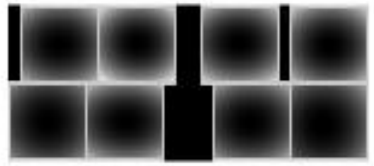
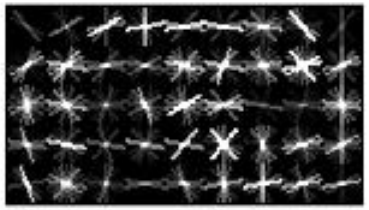
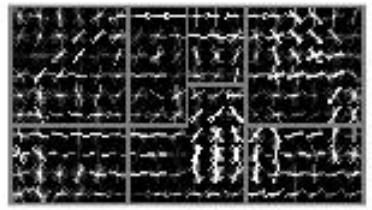
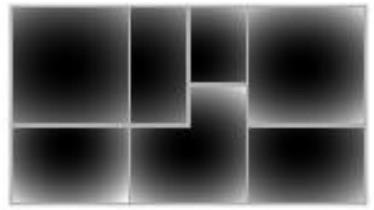
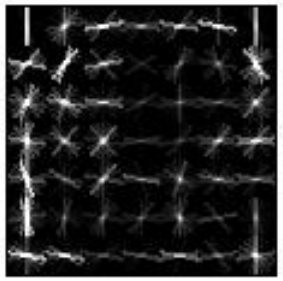
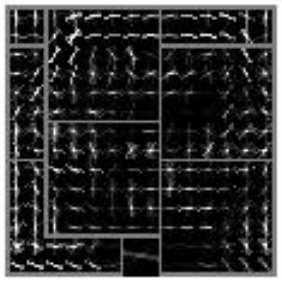
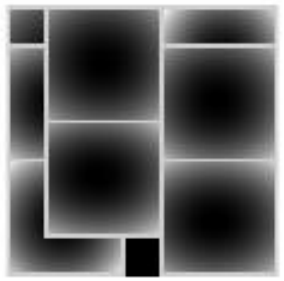
算法局限性：首先要有根模型（root filter）和若干部件模型(part filter)和部件模型的偏离损失。这些东西就是通过已有的人体，四肢，外轮廓等样本提取HOG特征然后经过svm训练而来的。

DPM检测流程图

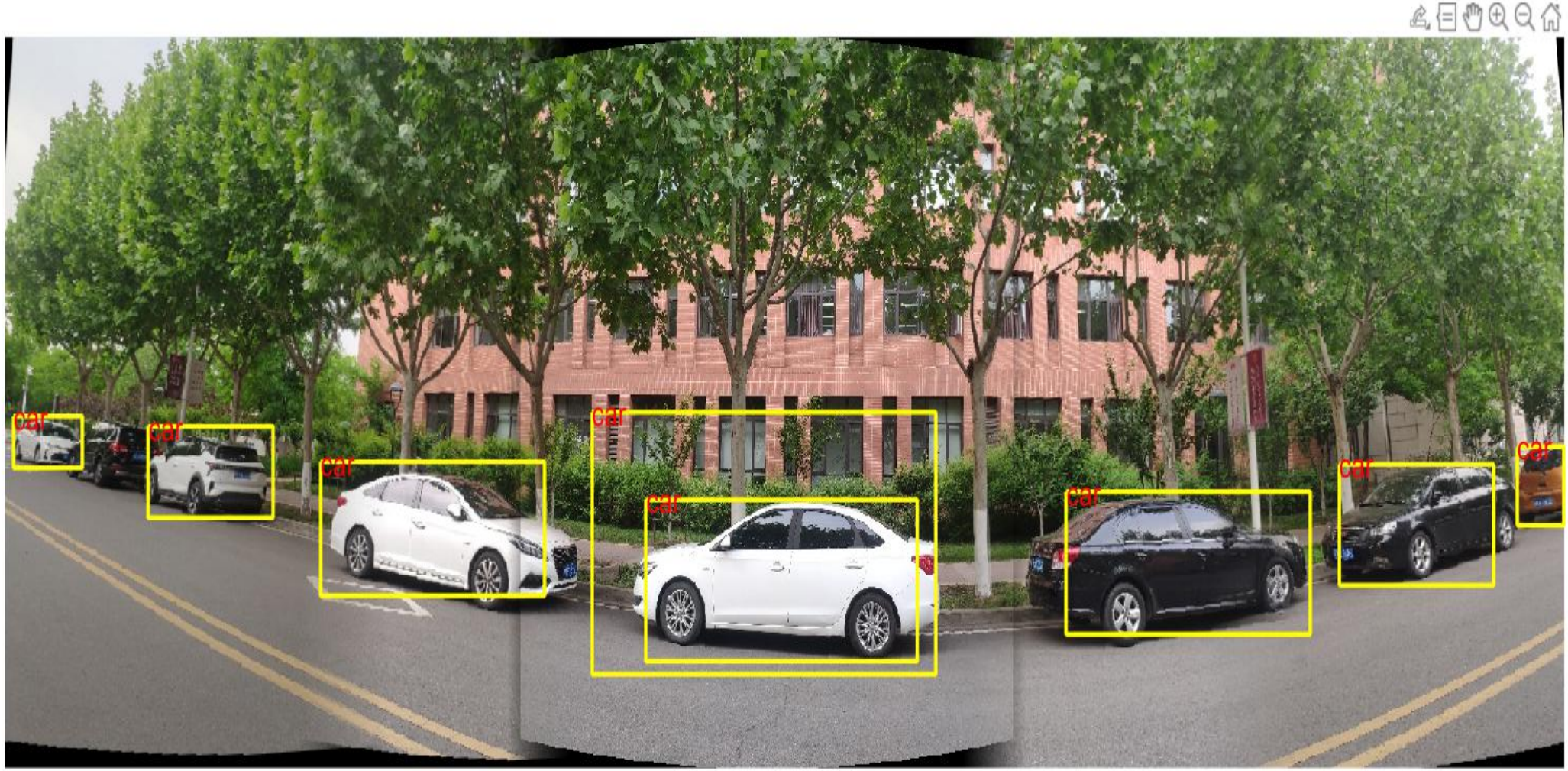


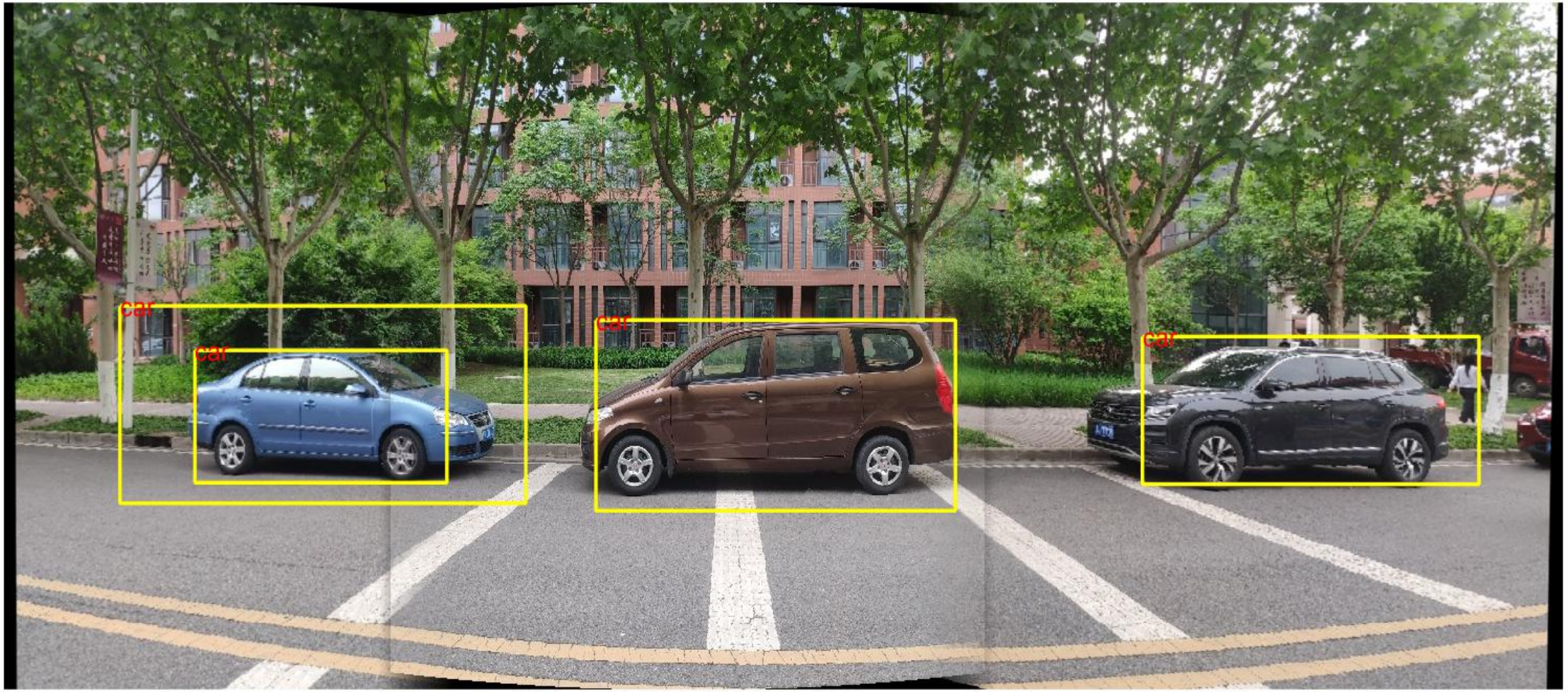
如上图所示，对于任意一张输入图像，提取其DPM特征图，然后将原始图像进行高斯金字塔上采样，然后提取其DPM特征图。对于原始图像的DPM特征图和训练好的Root filter做卷积操作，从而得到Root filter的响应图。对于2倍图像的DPM特征图，和训练好的Part filter做卷积操作，从而得到Part filter的响应图。然后对其精细高斯金字塔的下采样操作。这样Root filter的响应图和Part filter的响应图就具有相同的分辨率了。然后将其进行加权平均，得到最终的响应图。亮度越大表示响应值越大。

本次检测汽车所用到的model

目标检测样例展示

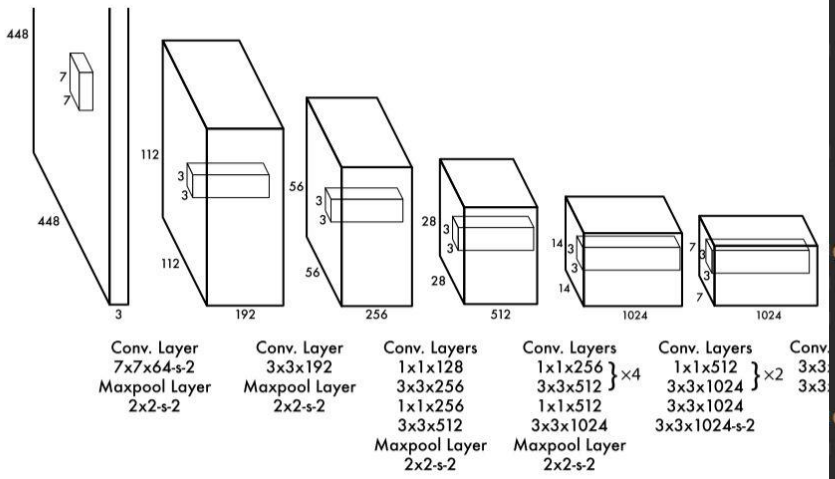

car
car
car
car
car
car
car
car


car
car
car
car

深度学习方法基于yolo预训练数据实现目标检测对比，基于coco和voc轻量级数据集

根据教程编写检测和接口文件：
网络接口文件 ————
darknet.py
预测文件————detect.py
预处理文件————
preprocess.py
外部函数集成通用文件----utils.py

外部依赖文件：
——yolov3.weight权重文件
——yolov3.cfg配置文件
对下图结构超参数具体配置



Darknet.py代码片断

```
class MaxPoolStride1(nn.Module):
    def __init__(self, kernel_size):
        super(MaxPoolStride1, self).__init__()
        self.kernel_size = kernel_size
        self.pad = kernel_size - 1

    def forward(self, x):
        padded_x = F.pad(x, (0,self.pad,0,self.pad), mode="replicate")
        pooled_x = nn.MaxPool2d(self.kernel_size, self.pad)(padded_x)
        return pooled_x


class EmptyLayer(nn.Module):
    def __init__(self):
        super(EmptyLayer, self).__init__()


class DetectionLayer(nn.Module):
    def __init__(self, anchors):
        super(DetectionLayer, self).__init__()
        self.anchors = anchors

    def forward(self, x, inp_dim, num_classes, confidence):
        x = x.data
        global CUDA
        prediction = x
```

Detect.py代码片断

```
num_classes = 80
classes = load_classes('data/coco.names')

#神经网络实例化并导入数据标签和权重文件
print("Loading network.....")
model = Darknet(args.cfgfile)
model.load_weights(args.weightsfile)
print("Network successfully loaded")

model.net_info["height"] = args.reso
inp_dim = int(model.net_info["height"])
assert inp_dim % 32 == 0
assert inp_dim > 32

#GPU装载提升预测速度
if CUDA:
    model.cuda()


#模型评估
model.eval()

read_dir = time.time()
```

深度学习yolo网络实现了全部的汽车检测，体现了对svm-dpm传统目标检测的优势

请老师批评指正

THANK YOU FOR WATCHING