

计算机视觉与模式识别

《遥感图像目标检测》

2022 年 06 月 10 日

一、实验内容

1.1 题目选择

基于非深度学习方法的目标检测任务进行遥感目标检测。在遥感图像数据上进行检测，可能面临的挑战包括：光照变化、拍摄角度变化，多尺度目标等。

可能的解决方案不限于：使用具有光照不变、旋转不变性质的图像特征，通过多种增强手段提升算法鲁棒性。

1.2 采用方法

1.2.1 数据集

遥感领域数据集有很多个，我了解到的有 LEVIR 数据集包含 3 个类别，DOTA 数据集包含 15 个类别，RSOD 数据集包含 4 个类别，HRRSD 数据集共 13 个类别。考虑到自己机器的性能和传统方法的分类能力。采用了类别数较少的数据集，即拥有 4 个类别的 RSOD 数据集。

RSOD 是一个免费开放的目标检测数据集，主要用于遥感图像中的目标检测。数据集包含飞机，油箱，运动场和立交桥四种标注目标。

RSOD 数据集包括 4 个文件夹，每个文件夹包含一种检测目标，有两种标注文件，一种是 xml 格式，一种是简单的 txt 文本：

1. 飞机数据集，446 幅图像中的 4993 架飞机
2. 操场，189 副图像中的 191 个操场。
3. 立交桥，176 副图像中的 180 座立交桥。
4. 油箱，165 副图像中的 1586 个 油箱。

数据集下载地址：<https://github.com/RSIA-LIESMARS-WHU/RSOD-Dataset->

1.2.2 方法设计

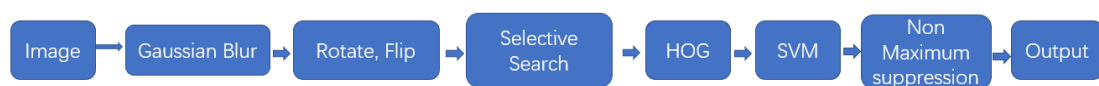
传统目标检测通常可以大致分为三个步骤：选择某一区域窗口、提取特征、

分类器分类。

区域窗口选择。较为简单常见的区域窗口选择方法是滑动窗口。即穷举各种比例尺度和区域，思想简单容易实现，缺点就是计算量非常大，也存在大量冗余的检测和计算。经过权衡，选用了速度较快效果较好的 **Selective Search** 进行窗口选择，该方法相比滑动窗口计算量大大减少，而且能获取到不同层次和大小的物体窗口，效果较好。

特征提取。传统目标检测的常用特征有 haar、HOG、SIFT 等特征提取方法。由于 haar 特征比较弱，所以不考虑用它。HOG 能较好的捕获局部形状信息，有一定的几何和光学不变性，不过不具有旋转不变性，而且由于需要计算梯度，所以对噪声会比较敏感。权衡考虑后采用 HOG 特征，考虑到其对噪声敏感，故在改进时，会加入例如高斯平滑等进行降噪，再提取特征，加上不具有旋转不变性，故改进时，会将训练图片进行旋转、镜像翻转等操作。

分类器。分类器有 Adaboost、cascade、SVM 等。由于 Adaboost 是弱分类器，考虑到我的遥感数据集有多个类别，故不太适合，所以采用 SVM 当分类器。数据集有 4 个类别，我还加入了第五类，即其他类别。



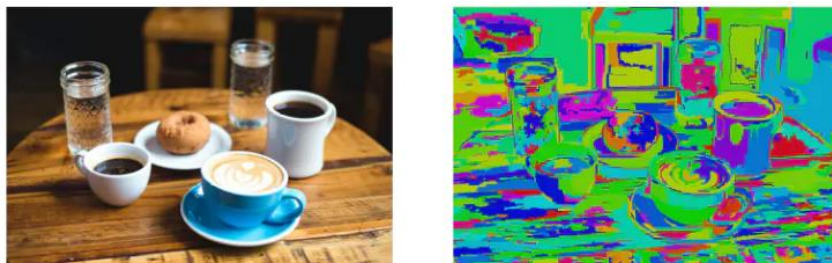
二、实验原理

2.1 原理解释

2.1.1 Selective Search

该算法有几个优势：捕捉不同尺度、多样化、计算速度快。

1. 分层分组算法（Hierarchical Grouping Algorithm）。
 - a) 使用论文《Efficient Graph-Based Image Segmentation》中的方法生成初始区域集合 region。



图表 1: 论文《Efficient Graph-Based Image Segmentation》得到的图像分割

- b) 计算邻近区域的相似度，包括颜色、纹理、吻合度、大小等相似度，把相似度最大的两个小区域合并成一个更大的区域，并加入 region 集合。
- c) 重复 b)，直到整个图像只剩下一个区域。那么得到的 region 集合就是生成的全部区域。

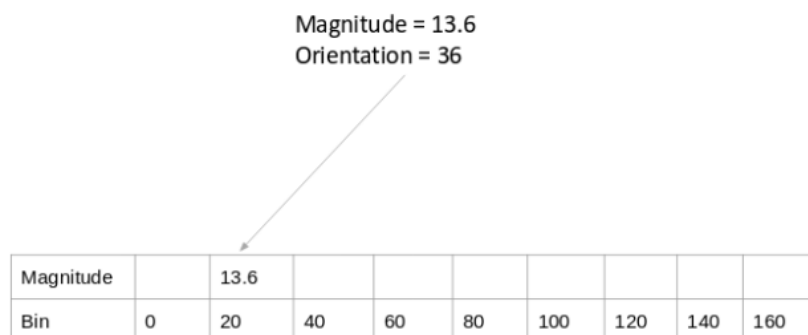
2. 多样性策略 (Diversification Strategies)

- a) 即计算相似度时, 考虑了颜色、纹理、大小、形状等多个相似度。然后通过权重加权得到总的相似度。

2.1.2 HOG

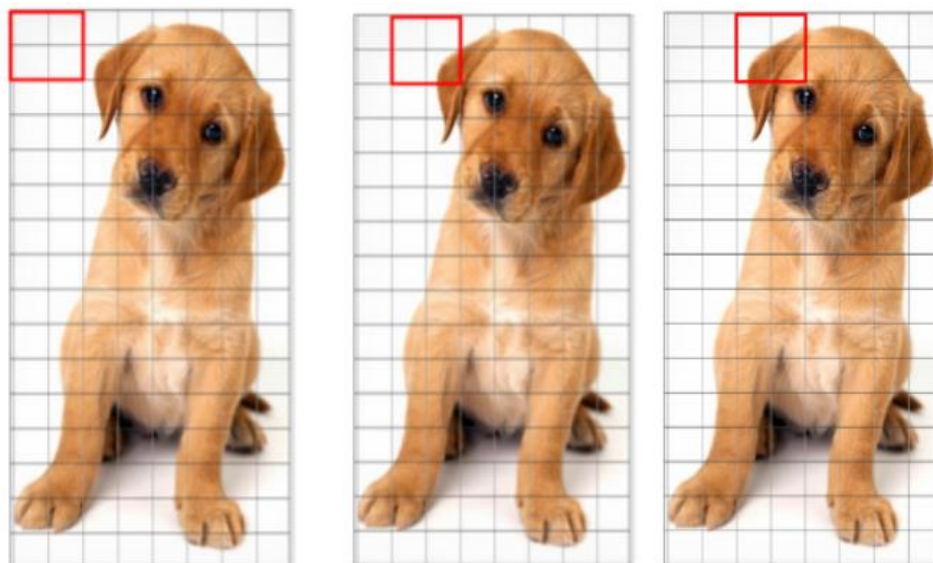
HOG 即方向梯度直方图的意思。主要思想是在一幅图像中, 局部区域的目标能够被梯度和梯度方向、密度较好的描述, 从而有较好的特征能让分类器取得不错的效果。

1. 将图像分割成 $n \times n$ 的 block 区域 (block 之间没有重叠), 例如 8×8 , 计算每个 block 中每个像素的水平 and 垂直梯度, 然后计算梯度方向 (得到的是度数)。
2. 把 180° 角度分箱成一个个的 bin, 例如 9 个 bins, 则每个 bin 有 $180^\circ/9=20^\circ$, 统计每个 block 中的像素点的梯度方向落在每个 bin 区间的像素点和梯度大小 (也有其他统计方法, 例如加权分配到相近的两个 bin), 这样的话, 一个 block 就有 9 个值。



图表 2: 例如一个点的梯度方向计算得到为 36° , 梯度大小为 13.6。 36° 落在 $20^\circ \sim 40^\circ$ 之间, 所以划分给了第二个 bin 13.6 的大小

3. 规定每个 cell 包含多少个 blocks, 例如 4 个, cell 与 cell 之间是否有重叠 (可以重叠 n 个 block, 这里以重叠一个 block 为例)。所以一幅图像的特征点个数就是 $9 \times \text{blocks_per_cell} \times \text{cell}$ 。



图表 3: 红色框是一个 cell, cell 里面有 4 个 block

2.1.3 非极大值抑制算法流程

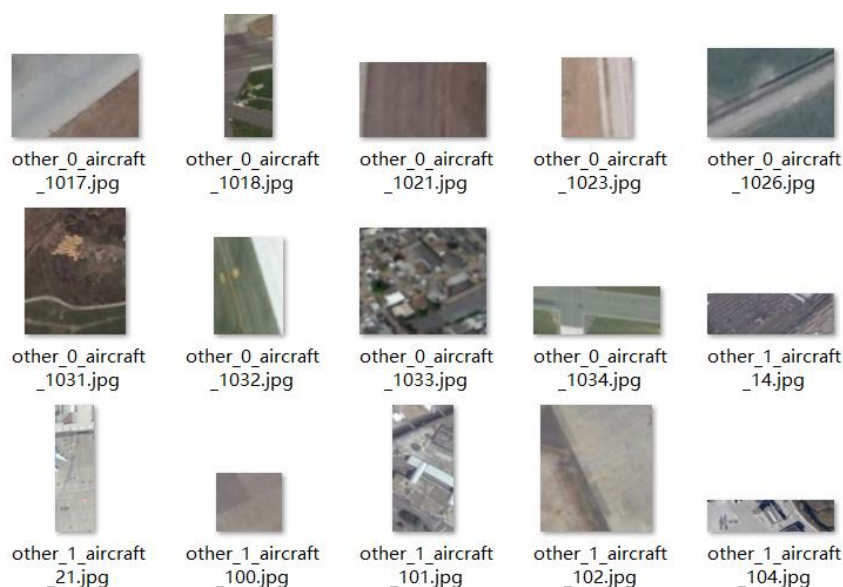
- 根据置信度得分进行排序
- 选择置信度最高的边界框添加到最终输出列表中，将其从边界框列表中删除
- 计算所有边界框的面积
- 计算置信度最高的边界框与其它候选框的 IoU。
- 删除 IoU 大于阈值的边界框
- 重复上述过程，直至边界框列表为空

2.1.4 其他类别数据集获取

RSOD 数据集包含飞机、油箱、立交桥和操场 4 种目标，为了训练 SVM 不仅能区分着 4 种类别，还要能判断是不是这 4 种之外的无关物品。故我设计了一个方法，从已有的数据集种提取某个不包含标注目标的子图作为第 5 类“其他”。

具体实现是：随机生成一个矩形坐标框，根据以下几个指标判断是否能作为“其他”类的子图。

- 计算 IOU，如果生成的框和标注的框 IOU 值大于设定的阈值，则不能作为“其他”类别
- 计算相交面积 S_0 与标注目标面积 S_1 之比，即 S_0/S_1 ，如果该结果大于某个阈值，说明生成的子图完全包含或包含了大部分的标注目标区域图片，这样是不符合“其他”类别的
- 随机生成的矩形框长宽比不能小于某个阈值，防止取到极端情形，入 1 像素宽，300 像素长的，没有意义



图表 4：“其他”类别图片，可以看到，经过以上几种策略，没有任何包含 4 种目标的图片作为“其他”类别的训练数据，也没有窗宽比例严重失调的样例

2.2 方法实现

2.2.1 数据集处理

数据集按照 8:1:1 的比例划分成训练集、验证集、测试集。训练集用来训练 SVM，验证集用来检验准确率与测试集差别是否很大，验证集用来做目标检测。对于已经有标注的 4 类样本，主要是提取 HOG 特征并存储起来给训练时用。对于每个标注的区域图片，resize 成 100*100 大小的图片后，提取 HOG 特征（采用 10 个 bin，每个 block 大小为 10*10），这样的话，每个图片的特征数就是 $\text{blocks} \times \text{cells} \times \text{bins} = (100/10 - 1)^2 \times 10 \times 4 = 3240$ 个。

对于“其他”类型的样本，从原图中提取，具体原理见 2.1.4，这里根据 2.1.4 的原理给出实现方法（只要不满足条件就一直重新生成矩形框，直到满足所有约束条件）。

```
while IoU >= IoU_thres or contain or intersection_self >= intersection_self_thres:
    y0 = np.random.randint(0,height)
    x0 = np.random.randint(0,width)
    h = np.random.randint(0,height)
    w = np.random.randint(0,width)
    y0_h = y0 + h
    x0_w = x0 + w
    if y0_h >= height: # 超边界了
        # y0_h = height - 1
        continue
    if x0_w >= width: # 超边界了
        # x0_w = width - 1
        continue
    ratio = 0.25
    # 保证长宽有一定的比例，最小不会超过某个比例
    if y0_h <= y0 + int(w*ratio):
        continue
    if x0_w <= x0 + int(h*ratio):
        continue
    IoUs = []
    Contains = []
    intersections = []
    for i in range(len(labels)):
        y,y_h,x,x_w = int(labels[i][3]),int(labels[i][5]),int(labels[i][2]),int(labels[i][4])
        IoUs.append(getIoU([y, y_h, x, x_w], [y0, y0_h, x0, x0_w])) # 用来判断生成的负样本不会大面积包含标注的4类物体
        Contains.append(DoesBoxContain([y0, y0_h, x0, x0_w], [y, y_h, x, x_w])) # 用来判断生成的负样本不会包含标注的4类物体
        intersections.append(get_intersection_self([y0, y0_h, x0, x0_w], [y, y_h, x, x_w]))
    IoU = max(IoUs)
    contain = max(Contains)
    intersection_self = max(intersections)
return y0,y0_h,x0,x0_w
```

图表 5：提取“其他”类别图片的子图作为负例

2.2.2 SVM 分类器训练

SVM 的训练其实更简单。4 种目标标签为 0、1、2、3，“其他”类别标签为 5。为了保证训练的鲁棒性，还进行了洗牌 shuffle，减少一连串样本都是同一个类型的情况。进行训练，我分别用了不同的核函数进行测试，包括 linear、poly、rbf 三个核函数。并保存模型，方便后续检测使用。

```
HOG SVM validation set accuracy: 0.965954449401268 ==> Training set accuracy 0.9779448474347302
SVM model has been saved in './model/rotate_flip_gaussian_svm.model'!
Svm training and saving cost 0:07:40.904891 s
```

图表 6：poly 核函数 SVM 训练结果，准确率很高，但是实际测试仍然有大量不属于训练数据

集中任何类别的数据会被预测为 5 种的其中一种

2.2.3 Selective Search

这里采用 opencv 进行选择搜索。因为是基于库实现，所以比较简单。经过测试，一张 1000×900 的遥感照片，大概会生成 3000 到 8000 张不等的矩形框，生成速度远远快于滑动窗口。并且计算量也更小。得到候选框集合后，遍历整个集合，提取特征进行预测，得到待评估的预测结果。

```
##### selective search #####
# 创建 Selective Search Segmentation 对象
ss = cv2.ximgproc.segmentation.createSelectiveSearchSegmentation()

# 添加待处理的图片
ss.setBaseImage(img)

# 可以选择快速但是低 recall 的方式，这里的 recall 指的是选择出来的 region 是否包含
# ss.switchToSelectiveSearchFast()

# 也可以选择慢速但是高 recall 的方式
ss.switchToSelectiveSearchQuality()

# 进行 region 划分，输出得到的 region 数目
rects = ss.process()
print('Total Number of Region Proposals: {}'.format(len(rects)))
print('Selective search cost ' + str(datetime.datetime.now() - time0) + ' s')
```

图表 7: Selective Search 实现

2.2.4 非极大值抑制

将刚刚得到的 Selective Search 框对应的子图，获取特征，输入训练好的 SVM 模型，预测是哪类目标。得到很多候选框，按概率从大到小进行排序，选出候选框概率最大的提出来，然后和剩下的框计算 IoU，将 IoU 大于阈值的去掉；反复计算，直到候选框为空。这样就得到了一个目标的全部结果，其他 4 种目标也是一样。下面代码就是对按概率排序后的框，取出最大的，和剩下的计算 IOU，将 IOU 大于阈值的去掉，重复此过程直到候选框为空。

```
IoU_threshold = 0.1
for i in range(4):
    # non maximum suppression
    while detect_boxes_sort[i] != []:
        # choose max probability and its corresponding box, then remove the max one
        final_boxes[i].append(detect_boxes_sort[i][0])
        final_proba[i].append(detect_proba_sort[i][0])
        box = detect_boxes_sort[i].pop(0)
        proba = detect_proba_sort[i].pop(0)

        # remove boxes whose IoU is larger than threshold
        removed_number = 0
        for j in range(len(detect_proba_sort[i])):
            # calculate IoU
            box_iou = getIoU(box, detect_boxes_sort[i][j-removed_number])
            if box_iou >= IoU_threshold:
                detect_boxes_sort[i].pop(j-removed_number)
                detect_proba_sort[i].pop(j-removed_number)
                removed_number += 1
```

图表 8: 非极大值抑制

2.3 尝试增加鲁棒性采用的措施

2.3.1 平滑去噪

因为 HOG 需要计算梯度，而噪声一般是高频的，会比较影响 HOG 的提取结果，所以对噪声比较敏感。基于此，在对图片进行特征提取之前，首先用高斯模糊进行平滑和降噪，然后再提取特征。高斯降噪的实现比较简单，不赘述。

2.3.2 数据增强、尽量增加旋转不变性

由于 HOG 不具有旋转不变性，为了使模型能更好的分辨各个目标，对一个区域的图片进行旋转 90°、180°、270° 的旋转，以及上下和左右的镜像翻转，这样能增加很多数据量，而且理论上将能增强整个系统处理遥感图像目标检测时的旋转不变性。

```
# 转化为PIL.JpegImagePlugin.JpegImageFile类型后旋转和翻转，再转化为numpy.array格式
img_90 = np.asarray((Image.fromarray(np.uint8(res_img0))).transpose(Image.ROTATE_90))
img_180 = np.asarray((Image.fromarray(np.uint8(res_img0))).transpose(Image.ROTATE_180))
img_270 = np.asarray((Image.fromarray(np.uint8(res_img0))).transpose(Image.ROTATE_270))
img_left_right = np.asarray((Image.fromarray(np.uint8(res_img0))).transpose(Image.FLIP_LEFT_RIGHT))
img_top_down = np.asarray((Image.fromarray(np.uint8(res_img0))).transpose(Image.FLIP_TOP_BOTTOM))
getHogAndSaveLine(writer=out, img=img_90)
getHogAndSaveLine(writer=out, img=img_180)
getHogAndSaveLine(writer=out, img=img_270)
getHogAndSaveLine(writer=out, img=img_left_right)
getHogAndSaveLine(writer=out, img=img_top_down)
```

图表 9：旋转和镜像增强，使得方法具有一定的旋转不变性

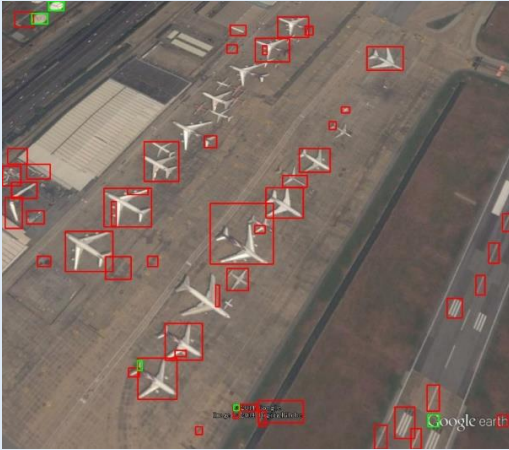
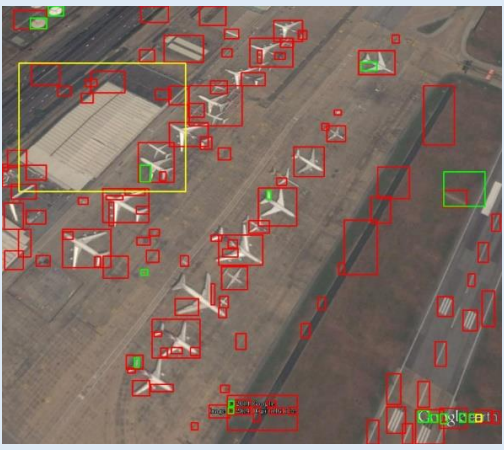

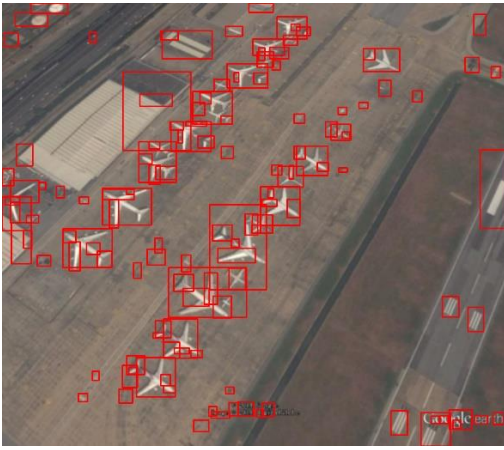
三、实验结果与分析

3.1 实验结果

3.1.1 检测飞机

取 Selective Search 前 500 个框效果

取 Selective Search 全部框效果

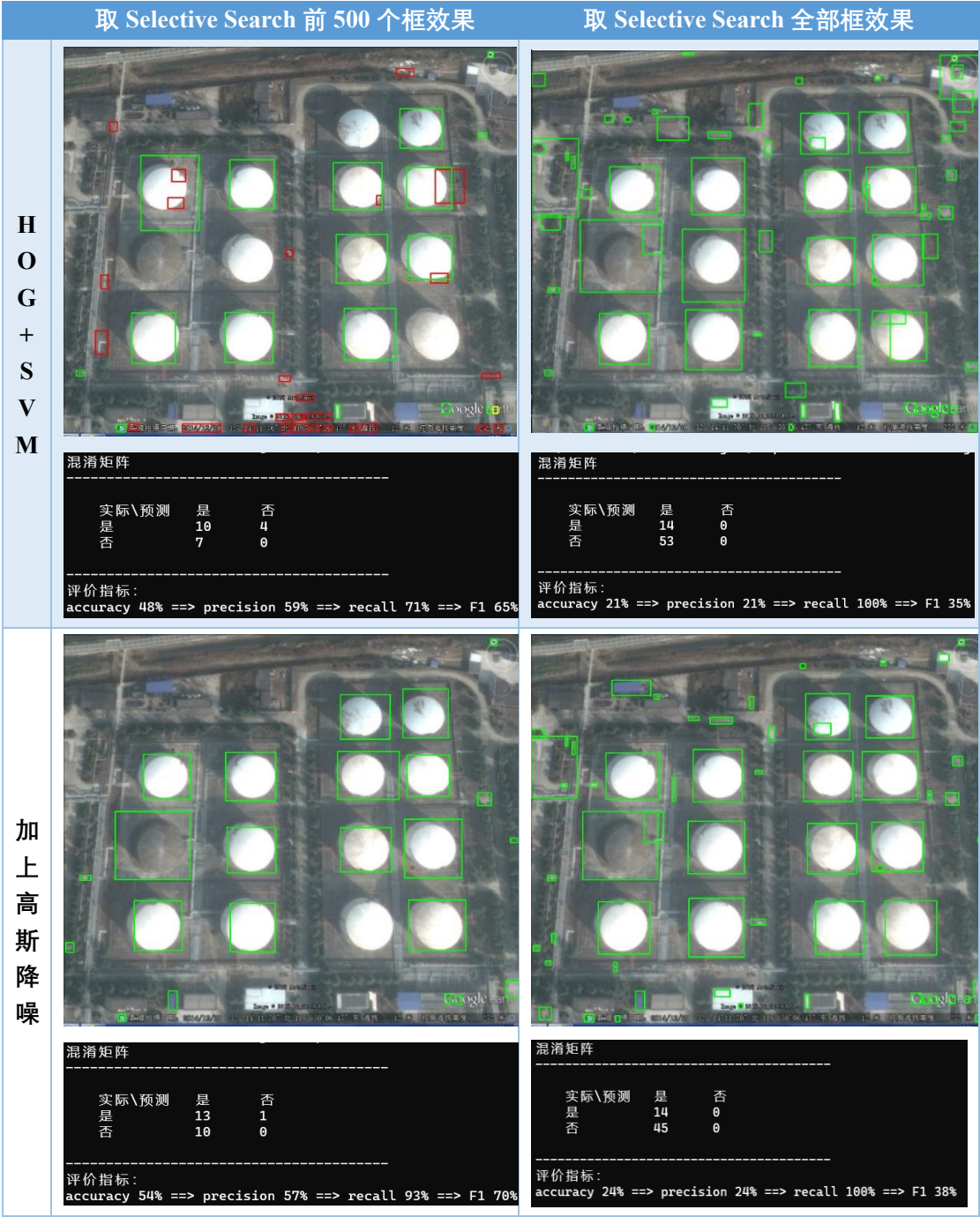
H O G + S V M																			
	混淆矩阵 ----- <table> <tr> <td>实际\预测</td><td>是</td><td>否</td></tr> <tr> <td>是</td><td>12</td><td>17</td></tr> <tr> <td>否</td><td>38</td><td>0</td></tr> </table> ----- 评价指标: accuracy 18% ==> precision 24% ==> recall 41% ==> F1 30%	实际\预测	是	否	是	12	17	否	38	0	混淆矩阵 ----- <table> <tr> <td>实际\预测</td><td>是</td><td>否</td></tr> <tr> <td>是</td><td>14</td><td>15</td></tr> <tr> <td>否</td><td>103</td><td>0</td></tr> </table> ----- 评价指标: accuracy 11% ==> precision 12% ==> recall 48% ==> F1 19%	实际\预测	是	否	是	14	15	否	103
实际\预测	是	否																	
是	12	17																	
否	38	0																	
实际\预测	是	否																	
是	14	15																	
否	103	0																	
高 斯 降 噪 + 翻 转 旋 转																			
	混淆矩阵 ----- <table> <tr> <td>实际\预测</td><td>是</td><td>否</td></tr> <tr> <td>是</td><td>18</td><td>11</td></tr> <tr> <td>否</td><td>68</td><td>0</td></tr> </table> ----- 评价指标: accuracy 19% ==> precision 21% ==> recall 62% ==> F1 31%	实际\预测	是	否	是	18	11	否	68	0	混淆矩阵 ----- <table> <tr> <td>实际\预测</td><td>是</td><td>否</td></tr> <tr> <td>是</td><td>18</td><td>11</td></tr> <tr> <td>否</td><td>117</td><td>0</td></tr> </table> ----- 评价指标: accuracy 12% ==> precision 13% ==> recall 62% ==> F1 22%	实际\预测	是	否	是	18	11	否	117
实际\预测	是	否																	
是	18	11																	
否	68	0																	
实际\预测	是	否																	
是	18	11																	
否	117	0																	

图表 10：检测飞机

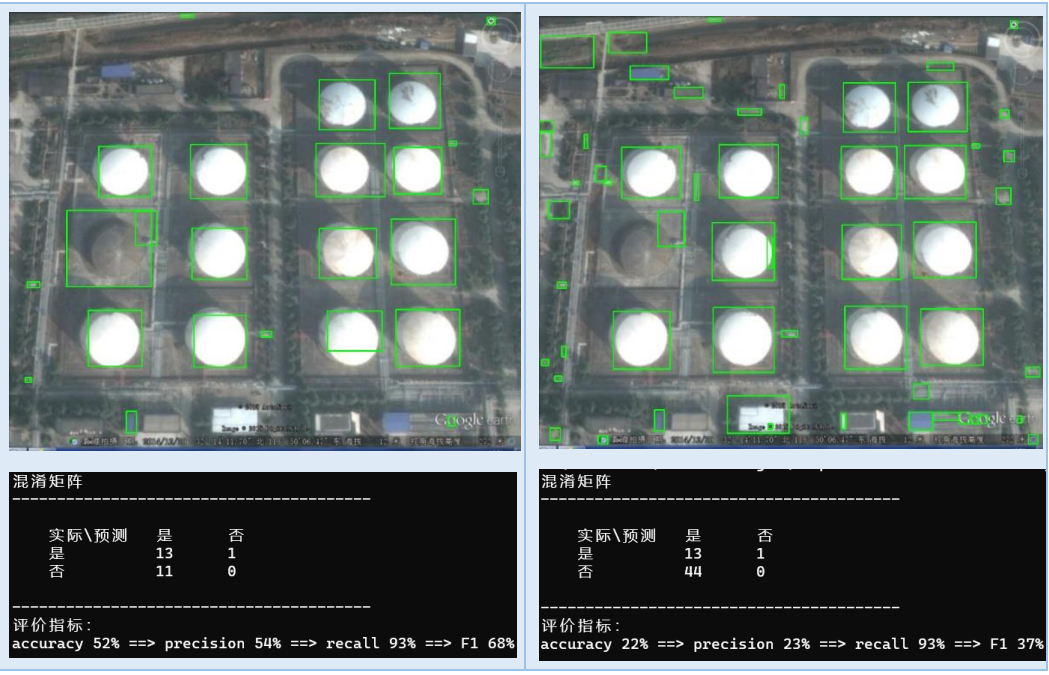
横向对比，也就是方法一样，不同框数目的对比。经过视觉上的观察，发现只取 Selective Search 一部分的框的时候，FP 的数量会少很多，也就是把不是飞机的预测成飞机的数量会少。然后用全部 Selective Search 都进行判断时，可以看到 TP 会有所提升，但是 FP 会增加的更多，所以除了召回率外，其他的都会变小，因为预测对的飞机增加数目不如预测错的飞机数目的增加。

方法对比，也就是表格的纵向对比。可以看到加了高斯降噪和翻转、旋转等操作后，TP 数目分别从 12、14 升高到了 18，而 FP 也有一定程度的提升。除了第一列指标对比里精度 precision 下降外，其余都是增加。故整体上来说，这样的增强方法和高斯模糊是有一定效果的。

3.1.2 检测油罐



高斯
降噪
+
翻
转
旋
转



图表 11：检测油罐

可以看到，由于油罐的特征较为单一，类似于一个圆，所以检测效果比飞机好很多。计时只使用一部分候选框，也能有 54% 的准确率，相反，用了全部候选框后，虽然正确的油罐都被正确识别了，但是错误成油罐的数目大大增加，所以导致了精度和召回率等都是下降的。

此外我们看到，仅仅加了高斯模糊后，效果的提升就非常大了。而翻转相比之下就没有那么明显，没有表现出什么效果。而且由于镜像翻转、旋转等，导致数据量增加了很多，训练时甚至比较难收敛，故效果不明显。

3.1.3 检测操场

提取操场



图表 12：检测操场

来看操场的检测，这个目标的检测就比较困难了，首先由于操场一般比较大，所以会受到图片中很多小物体的干扰。经过我的测试，对于是操场的区域，它是一定能识别出是操场的，但是对于不是操场的区域，就会产生错误预测，把不是操场的当成操场，例如，图中很多房子就被当成了操场，因为他们的特征都是方形为主。这个原因有比较大的一部分是因为缺乏非目标样本的训练（即除了飞机、油罐、立交桥、操场 4 种外），我是对非目标样本进行随机采样，很难包含全部与 4 种目标类似但又不是目标的样本。

四、总结

我觉得此次在遥感目标检测上做的不好的地方，即对于操场的检测效果较差以及会误把非目标物体识别成目标物体情况。我目前心中的想法是：改进 SVM 分类器的方式。因为我目前是用 SVM 进行多分类，有 4 种目标，加上非目标样

本，共 5 种分类了，所以是 5 分类问题，而第 5 类样本很难概括大量的样例，如果量过于大，还会导致训练样本不均衡的问题，训练得到的效果反而会变差。如何改进呢，我觉得，可以采用多个 SVM 分类的方式，即一个 SVM 只识别一个目标，例如“飞机”为一类，“油罐、操场、立交桥”3 个为一个类别，然后 4 种类别就需要 4 个 SVM，这样的效果按理说是会比一个多分类的 SVM 效果要好。

另外，遥感目标检测感觉还是挺有难度的，因为是俯视视角，所以会要求有一定的旋转不变性，以及尺度的问题，都是切实需要解决的问题，我也通过去噪声进行平滑、镜像处理、翻转处理、对非目标进行大量采样（为了解决把非目标物体识别成目标物体的问题）。可惜的是效果只有比较小的提升，由于机器性能的限制、遥感数据集较为庞大，数据处理和训练、包括目标检测都需要大量的机器时间，受限于本人的机器以及时间，目前只能做到这一步。但我感觉还有的提升空间，仍然有可以尝试提升效果的地方，后面会继续探究这方面的内容。