《计算机视觉-openCV应用技术》

第七章 目标检测与识别

李策

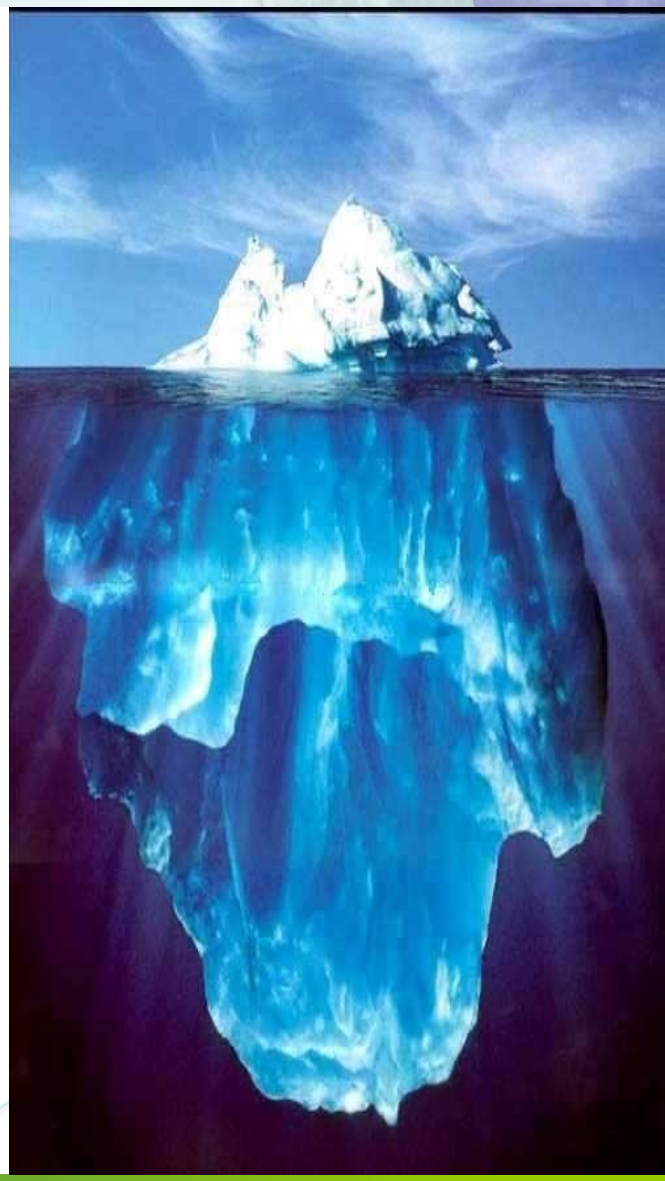
中国矿业大学（北京）计算机科学与技术系

E-mail: celi@cumtb.edu.cn

提纲

第七章 目标检测与识别

- 7.1 目标检测与识别技术
  - 7.1.1 HOG描述符
  - 7.1.2 检测人
  - 7.1.3 创建和训练目标检测器
- 7.2 汽车检测
  - 7.2.1 代码的功能
  - 7.2.2 SVM 和滑动窗口
- 7.3 总结



7.1 目标检测与识别技术

在计算机视觉中有很多目标检测和识别的技术，本章会用到：

- 梯度直方图（Histogram of Oriented Gradient， HOG）
- 图像金字塔（image pyramid）
- 滑动窗口（sliding window）

与特征检测算法不同，这些算法是互补的。如在梯度直方图（HOG）中会使用滑动窗口技术。

7.1.1 HOG描述符

HOG 是一个特征描述符，因此 HOG 与 SIFT、SURF 和 ORB 属于同一类型的描述符。

人脸识别的 LBPH 描述符：

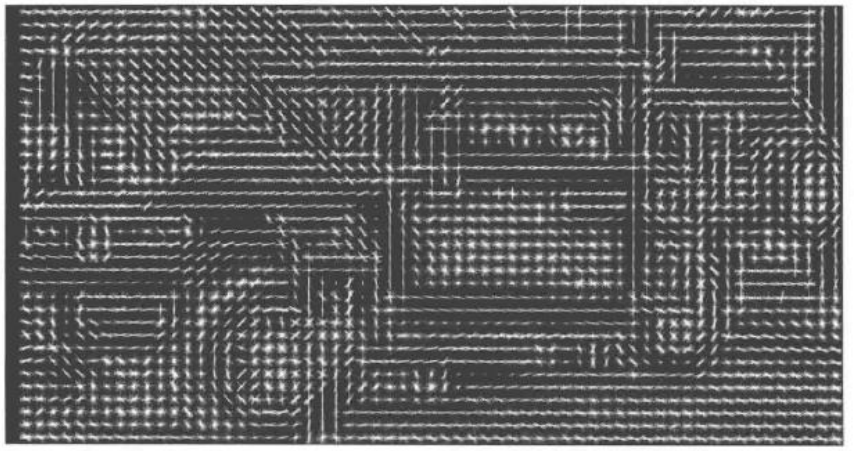
- 第一步：将图像划分成多个部分
- 第二步：计算各个部分的梯度

HOG 所得到的特征描述符能够为特征匹配和目标检测提供非常重要的信息。

7.1.1 HOG描述符

下图是一个卡车图像：

下图为HOG对卡车图像所提取的特征：



7.1.1 HOG描述符

将直方图外推成描述符过程：

- 计算每个单元的局部直方图，这些单元会合成较大的区域，也称为块（block）
- 按块构成特征向量是为了便于归一化

两个主要的问题：

- 位置
- 尺度

7.1.1 HOG描述符

1. 尺度问题

对于这个问题，可以通过两个例子来说明：若要检测的目标（比如自行车）是较大图像中的一部分，要对两幅图像进行比较。如果在比较过程中找不到一组相同的梯度，则检测就会失败（即使两幅图像都有自行车）。

7.1.1 HOG描述符

2. 位置问题

待检测的目标可能位于图像的任何位置，所以需要扫描图像的各个部分，以确保能找到感兴趣的区域，且在这些区域中去尝试检测目标。

两个技术：

A. 图像金字塔

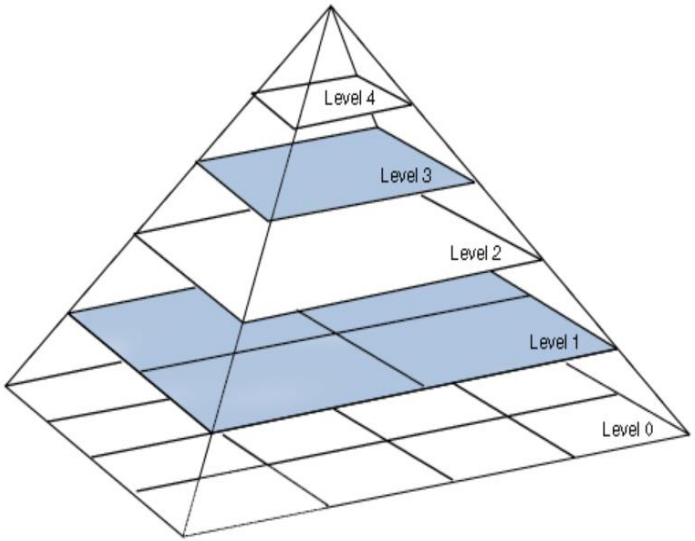
B. 滑动窗口

7.1.1 HOG描述符

A. 图像金字塔

如何构建图像金字塔：

- 获取图像
- 使用任意尺度的参数来调整（缩小）图像大小
- 平滑图像（使用高斯模糊）
- 如果图像比最小尺寸还大，则从第一步重复该过程



7.1.1 HOG描述符

B.滑动窗口

滑动窗口通过扫描较大图像的较小区域来解决定位问题，进而在同一图像的不同尺度下重复扫描。

- 图像分解成多个部分
- 丢掉那些不太可能包含对象的部分
- 对可能区域进行分类

7.1.1 HOG描述符

存在有一个问题：区域重叠（overlapping region）

窗口每次都会丢掉几个像素，滑动窗口可以对同一张人脸的四个不同位置进行正匹配；只需要一个匹配结果，而不是四个，这里对有良好评分的图像区域不感兴趣，而是对有最高评分的图像区域感兴趣。

这带来了另一个问题，非最大抑制，它是指给定一组重叠区域，可以用最大评分来抑制所有未分类区域。

7.1.1 HOG描述符

3. 非最大（或非极大）抑制

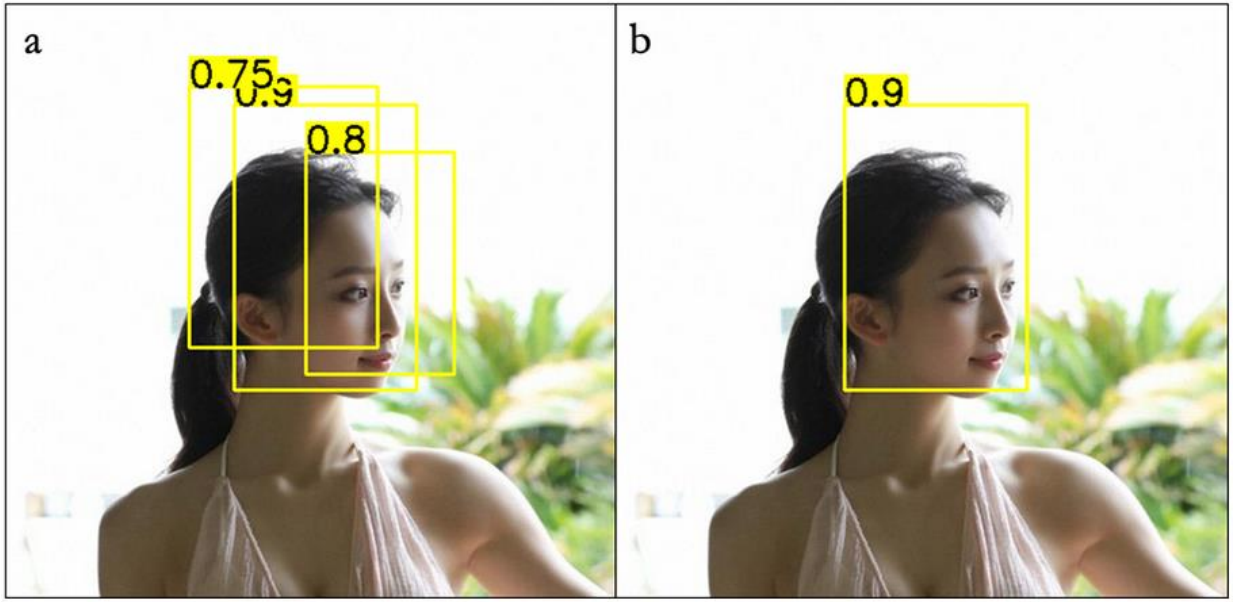
一种针对与图像同一区域相关的所有结果进行抑制的技术，使这些区域没有最大评分，利用该技术可以找到最佳的目标边界框，消除冗余的边界框。

目标检测的过程中，同一目标位置上会产生大量的候选框，这些候选框相互之间可能会有重叠，此时我们需要利用非最大值抑制找到最佳的目标边界框，消除冗余的边界框。

7.1.1 HOG描述符

a图为人脸检测的候选框结果，并在标注了置信度得分（confidence score），如果不使用非极大值抑制，就会有多个候选框出现。

b图为使用非极大值抑制的结果，符合人脸检测的预期结果。



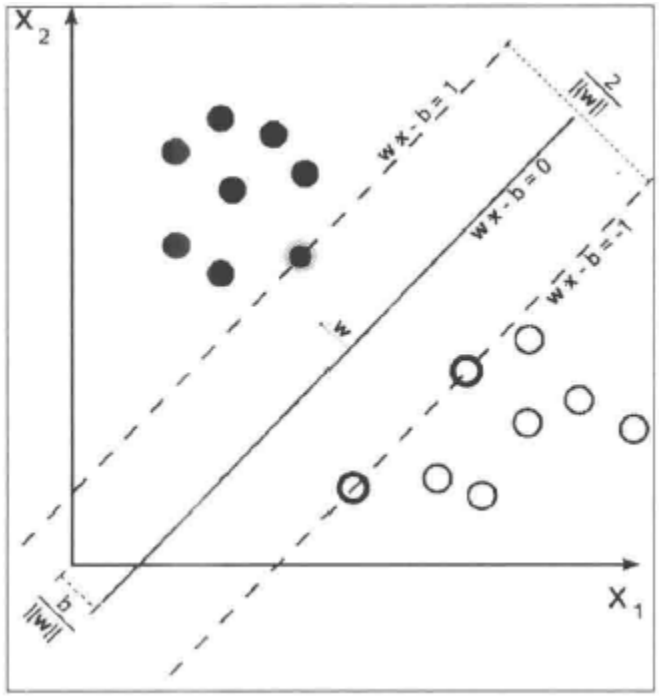
7.1.1 HOG描述符

对于上述如何确定窗口的评分，需要一个分类系统来确定某一特征是否存在，且对这种分类会有一个置信度评分，一般而言都是采用支持向量机（SVM）来分类。

7.1.1 HOG描述符

4. 支持向量机SVM

SVM是一种算法，对于带有标签的训练数据，通过一个优化的超平面来对这些数据进行分类。



7.1.2 检测人

OpenCV 自带的HOGDescriptor 函数可检测人。

实例如下：

A.确定某矩形是否完全包含在另一个矩形中

```
def is_inside(o, i):
```

B.绘制矩形来框住检测到的人

```
def draw_person(image, person):
```

C.导入图像

```
img = cv2.imread("face.jpg")
```

7.1.2 检测人

D.实例化HOGDescriptor对象，作为检测人的检测器

```
hog = cv2.HOGDescriptor()
```

E.设置线性SVM分类器的系数

```
hog.setSVMDetector
```

F.加载图像

```
found, w = hog.detectMultiScale(img)
```

7.1.3 创建和训练目标检测器

仅能实现人检测或人脸检测还远远不够，我们需要了解如何得到这些（用于人检测器的）特征，且知道如何修改这些特征。

在实际生活中，我们的检测更为具体，如汽车车牌等。

那么如何构建分类器？

- SVM
- 词袋（Bag-Of-Word，BOW）技术

7.1.3 创建和训练目标检测器

1. 词袋BOW

词袋BOW的概念起源于语言分析和信息检索领域，计算机视觉会使用词袋BOW的升级版本。

BOW 用来在一系列文档中计算每个词出现的次数，再用这些次数构成向量来重新表示该文档。

7.1.3 创建和训练目标检测器

应用示例：

Document 1 ： I like OpenCV and I like Python

Document 2 ： I like C++ and Python

Document 3 ： I don't like artichokes

对于上述三个文档，我们可以使用这些值来建立字典：

{ I:4,

like:4,

OpenCV:2,

and:2,

Python:2,

C++:1,

don't:1,

artichokes:1 }

7.1.3 创建和训练目标检测器

该字典有 8 个键值对，使用这 8 个键值对所构成的向量来重新表示原始文件，每个向量包含字典中的所有单词，向量中的每个元素表示文档中每个单词出现的次数，具体表示如下：

[2,2,1,1,1,0,0,0]

[1,1,0,1,1,1,0,0]

[1,1,0,0,0,0,1,1]

7.1.3 创建和训练目标检测器

2.计算机视觉中的BOW

回顾：

- 熟悉了图像特征的概念，且使用过图像特征提取方法（SIFT 和 SURF），这些特征可与其他图像特征进行匹配
- 理解了支持向量机的作用：给SVM提供一组特征数据，训练分类模型；然后可依据该模型预测其他数据属于哪一类别

7.1.3 创建和训练目标检测器

2.计算机视觉中的BOW

BOW 方法的实现步骤如下：

A. 取一个待训练的样本数据集

B. 提取数据集中的（每幅图）描述符

C. 将每幅图对应的描述符添加到BOW训练器中

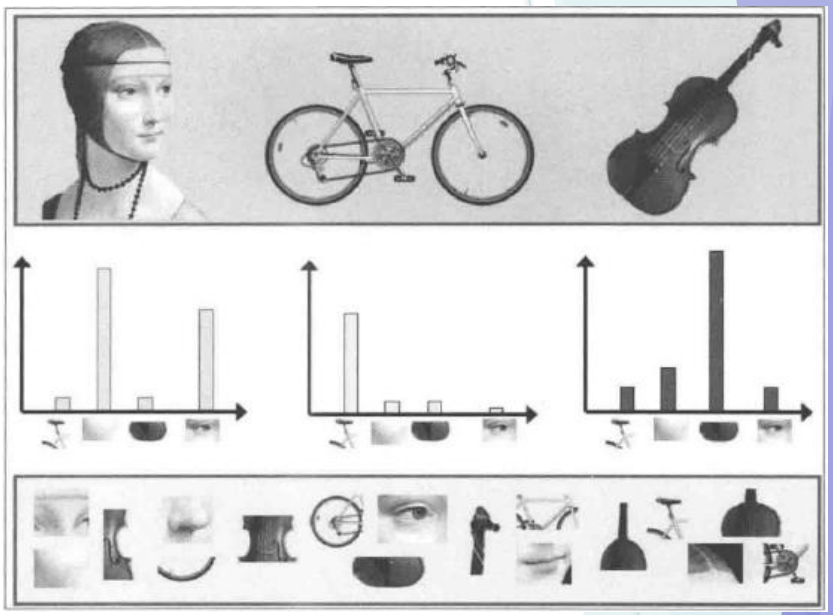
D. 将描述符聚类到 K 簇中（聚类的中心就是 " 视觉单词 " ）

7.1.3 创建和训练目标检测器

测试一下分类器，并尝试进行检测，具体过程为：

- 给定测试图像
- 提取特征
- 量化特征到最近簇心的距离，形成直方图

右图是BOW过程的可视化表现:



7.2 汽车检测

为了训练数据，需要有足够大的数据集，且训练图像的大小要一样。

可利用现成的数据集，下面的例子使用UIUC数据集

代码见书上P117

7.2.1 代码的功能

A. 定义训练图像的路径

```
datapath = "D:/project/opencv3/c7/CarData/TrainImages"
```

B. 定义函数完整路径path

```
def path(cls, i):
    return "%s/%s%d.pgm" % (datapath, cls, i+1)
pos, neg = "pos-", "neg-“
```

C. 创建SIFT实例

detect提取关键点，extract提取特征

```
detect = cv2.xfeatures2d.SIFT_create()
extract = cv2.xfeatures2d.SIFT_create()
```

7.2.1 代码的功能

D. 当看到SIFT时，就可断定会涉及一些特征匹配算法

范例接下来会创建基于FLANN匹配器的实例

```
flann_params = dict(algorithm=1, trees=5)
matcher = cv2.FlannBasedMatcher(flann_params, {})
```

E. 创建BOW训练器，并指定训练器簇数为40

```
bow_kmeans_trainer = cv2.BOWKMeansTrainer(40)
```

F. 初始化BOW提取器，视觉词汇将作为BOW类输入

```
extract_bow = cv2.BOWImgDescriptorExtractor(extract, flann)
```

7.2.1 代码的功能

G. 为了从图像中提取SIFT特征，需要通过一个方法来获取图像特征，并以灰度格式读取图像，然后返回描述符

```
def extract_sift(fn):
    im = cv2.imread(fn, 0)
    return extract.compute(im, detect.detect(im))[1]
```

7.2.1 代码的功能

以上是为训练BOW训练器做好了一切准备：

A. 每个类都从数据集中读取 8 张图像（8个正样本、8个负样本）

```
for i in range(8):
    bow_kmeans_trainer.add(extract_sift(path(pos, i)))
    bow_kmeans_trainer.add(extract_sift(path(neg, i)))
```

B. 创建视觉单词词汇需要调用训练器上的cluster函数，该函数执行k-means分类返回词汇

```
vocabulary = bow_kmeans_trainer.cluster()
```

将为BOWImgDescriptorExtractor指定返回的词汇，以便它能从测试图像中提取描述符

```
extract_bow.setVocabulary(vocabulary)
```

7.2.1 代码的功能

C.函数返回基于BOW的描述符提取器计算得到的描述符

```
def bow_features(fn):
 im = cv2.imread(fn, 0)
 return extract_bow.compute(im, detect.detect(im))
```

D. 创建训练数据数组和标签数组

并用BOWImgDescriptorExtractor产生的描述符填充它们

接下来的方法生成相应的正负样本图像的标签

```
traindata, trainlabels = [], []
 for i in range(20):
     traindata.extend(bow_features(path(pos, i))); trainlabels.append(1)
     traindata.extend(bow_features(path(neg, i))); trainlabels.append(-1)
```

7.2.1 代码的功能

E.创建一个SVM实例

```
svm = cv2.ml.SVM_create()
```

F.通过将训练数据和标签放到Numpy数组中来进行训练

```
svm.train(np.array(traindata), cv2.ml.ROW_SAMPLE,
np.array(trainlabels)))
```

7.2.1 代码的功能

给SVM一些样本图像：

A.函数用来显示predict方法效果

```
def predict(fn):
    f = bow_features(fn)
    p = svm.predict(f)
    print(fn, "\t", p[1][0][0])
    return p
```

B.定义两个样本图像的路径，并将路径中的图像文件读取出来放到Numpy数组中

```
car,notcar = "D:/project/opencv3/c7/CarData/TestImages/test-0.pgm"
car_img = cv2.imread(car)
notcar_img = cv2.imread(notcar)
```

7.2.1 代码的功能

给SVM一些样本图像：

C. 将这些图像传给训练好的SVM，并获得预测结果

car_predict = predict(car)

not_car_predict = predict(notcar)

D. 最后在屏幕上展现图像，希望能看到每个图像都有正确的文字说明

font = cv2.FONT_HERSHEY_SIMPLEX

7.2.1 代码的功能

给SVM一些样本图像：

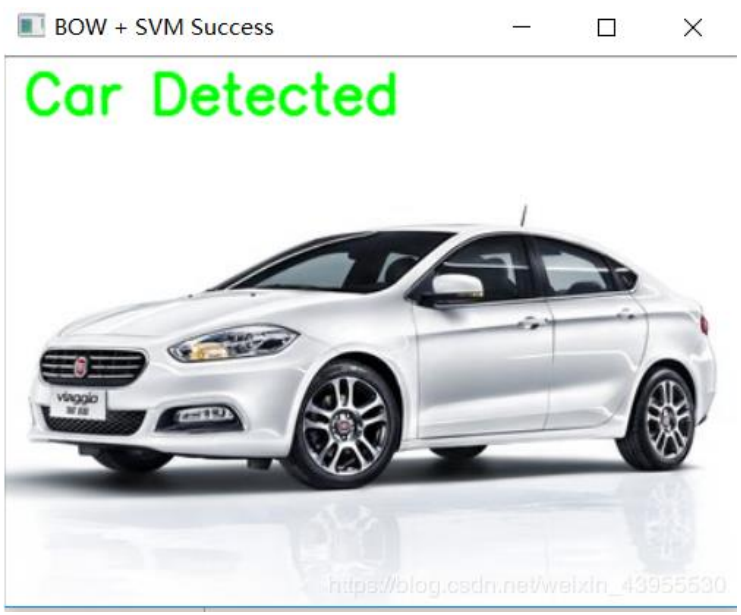
C. 将这些图像传给训练好的SVM，并获得预测结果

car_predict = predict(car)

not_car_predict = predict(notcar)

D. 最后在屏幕上展现图像，希望能看到每个图像都有正确的文字说明

font = cv2.FONT_HERSHEY_SIMPLEX

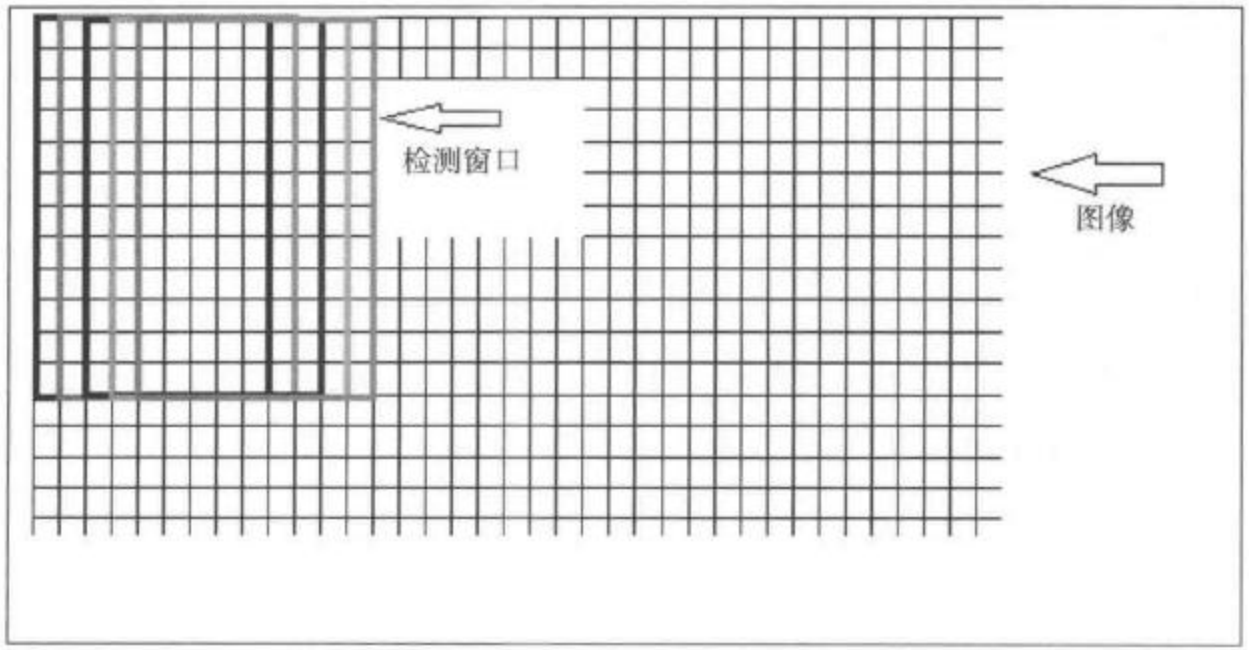


7.2.2 SVM和滑动窗口

完成目标检测后，想增加如下功能：

- 检测图像中同一种物体的多个目标
- 确定检测到的目标在图像中的位置

需要使用滑动窗口方法。



7.2.1 代码的功能

观察块的运动：

A. 选择图像区域，对其进行分类，向右移动固定大小的一步

B. 在每步中，使用通过BOW训练好的SVM进行分类

C. 将所有块传递给SVM进行预测，并保持结果

D. 完成整个图像的分类后，缩放图像并重复整个滑动窗口的过程

7.2.1 代码的功能

1.汽车检测的例子

- 获取一个训练数据集
- 创建BOW训练器并获得视觉词汇
- 采用词汇训练SVM
- 尝试对测试图像的图像金字塔采用滑动窗口进行检测
- 对重叠的矩形使用非最大抑制
- 输出结果

7.2.1 代码的功能

2. 测试汽车检测器

- 加载图像
- 利用滑动窗口和图像金字塔进行检测

代码见课本P131，最关键的部分为循环中的金字塔/滑动窗口：

7.2.1 代码的功能

```
bf = bow_features(roi, extractor, detect)
    _, result = svm.predict(bf)
    a, res = svm.predict(bf, flags=cv2.ml.STAT_MODEL_RAW_OUTPUT |
cv2.ml.STAT_MODEL_UPDATE_MODEL)
    print "Class: %d, Score: %f, a: %s" % (result[0][0], res[0][0], res)
    score = res[0][0]
    if result[0][0] == 1:
        if score < -1.0:
            rx, ry, rx2, ry2 = int(x * scale), int(y * scale), int((x+w) * scale),
int((y+h) * scale)
            rectangles.append([rx, ry, rx2, ry2, abs(score)])
```

7.3 总结

目标检测的概念：

- HOG
- BOW
- SVM
- 金字塔
- 滑动窗口
- 非最大抑制

下一章会以视频为基础，利用目标检测和分类技术学习如何跟踪目标。