基于边缘图的快速物体定位

答辩人：周航

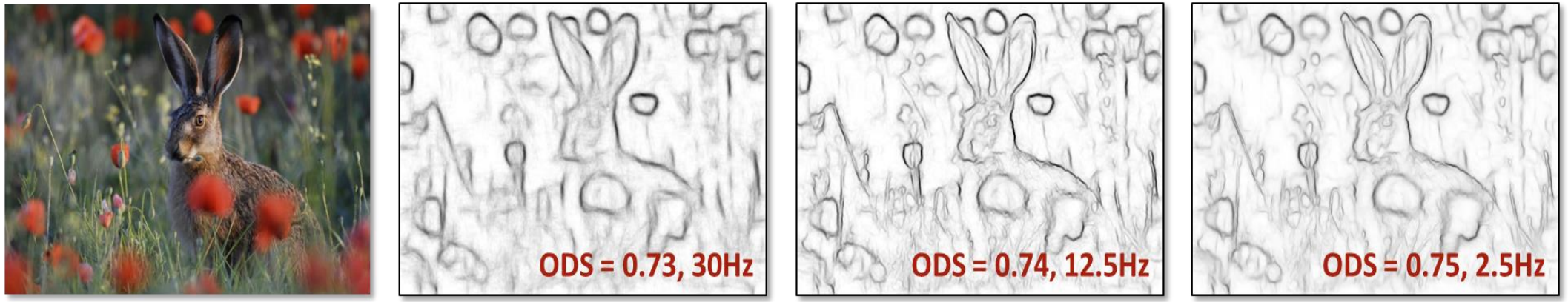
指导教师：沈为

概论

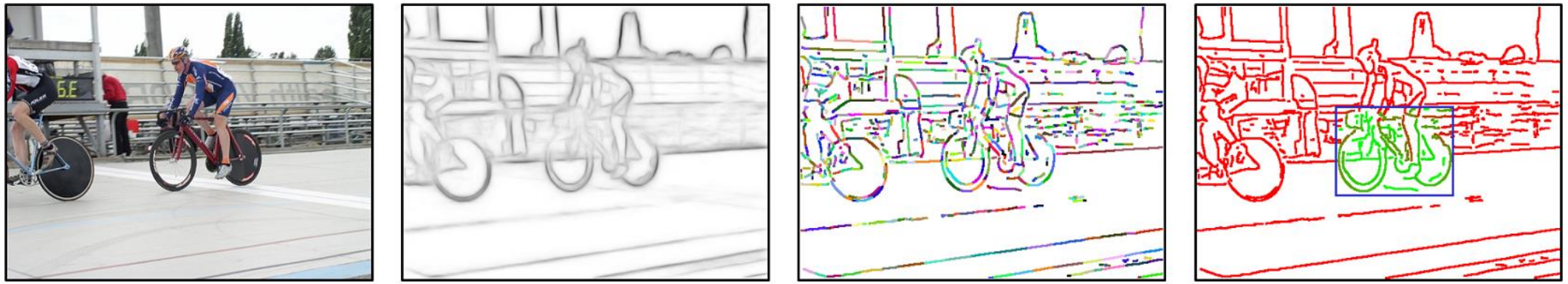
- 物体检测的应用：
- 交通（车辆归类/车牌识别）
- 人脸识别/区分
- 大规模图像分类
- 物体定位
- ...

概要

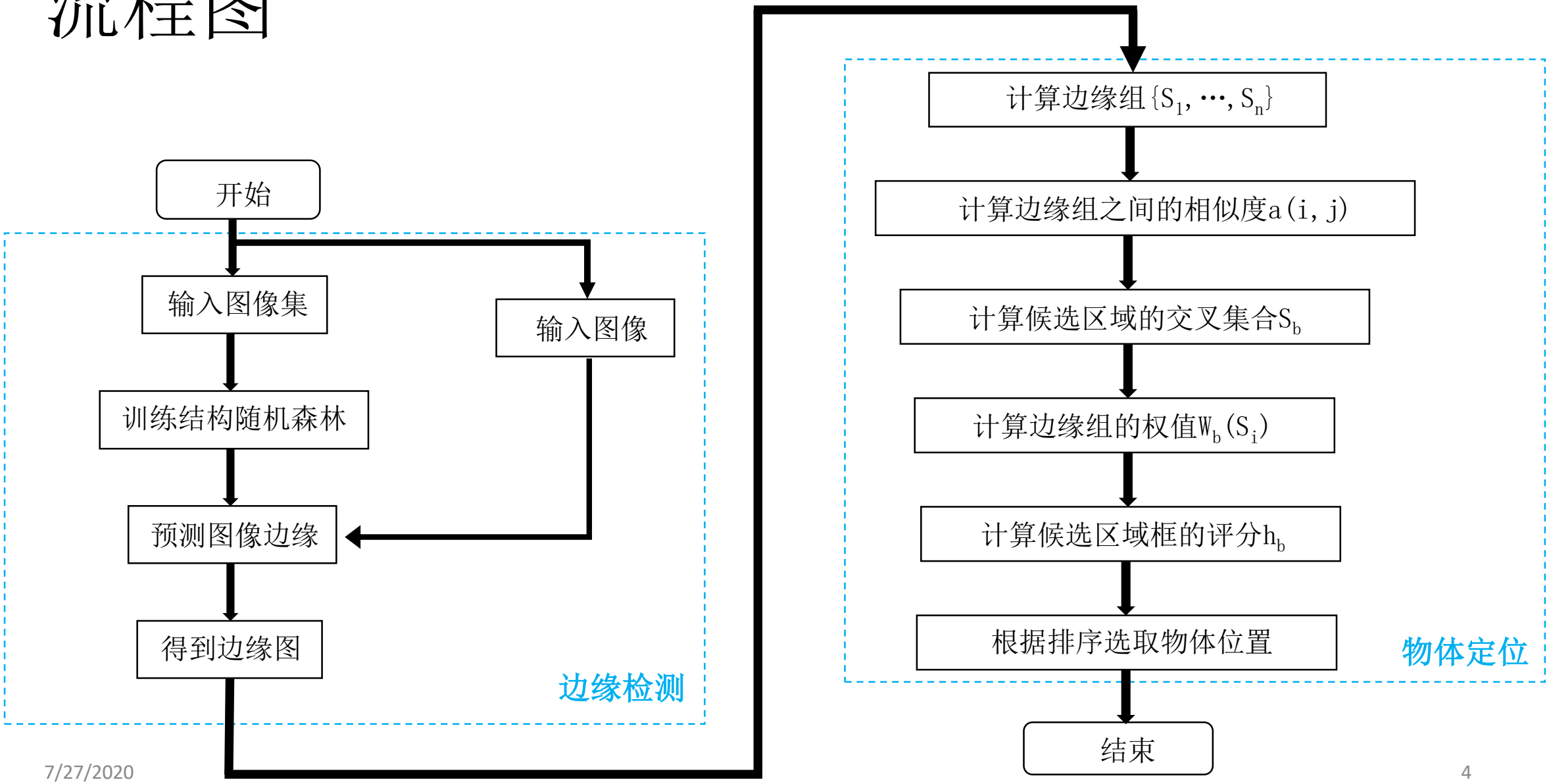
structured edge prediction——结构化边缘检测



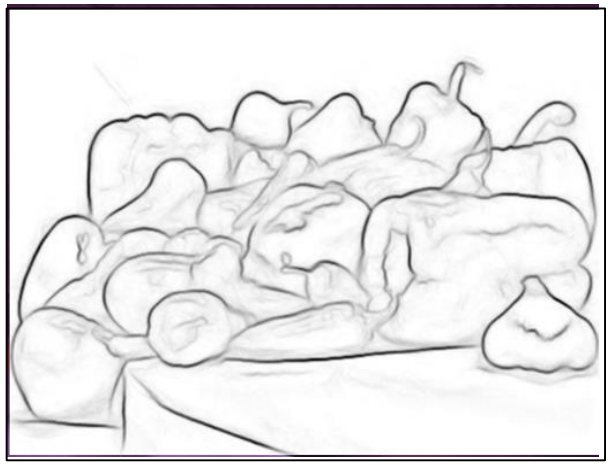
from edges to objects——由边缘直接获得物体位置

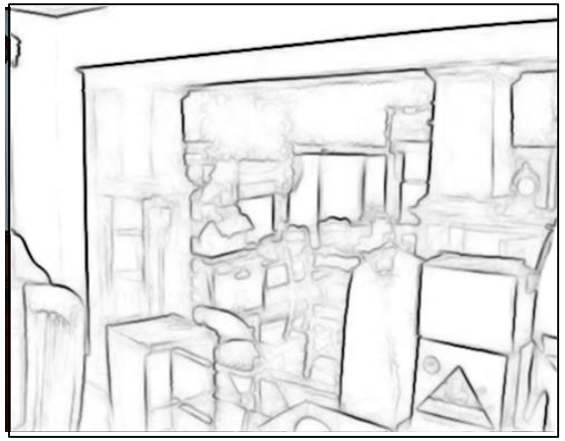


流程图


开始
输入图像集
输入图像
训练结构随机森林
预测图像边缘
得到边缘图
边缘检测
计算边缘组{S1, …, Sn}
计算边缘组之间的相似度a(i, j)
计算候选区域的交叉集合Sb
计算边缘组的权值Wb(Si)
计算候选区域框的评分hb
根据排序选取物体位置
物体定位
结束

什么是边缘？





亮度
色彩
纹理
相似性
连贯性
对称性
…

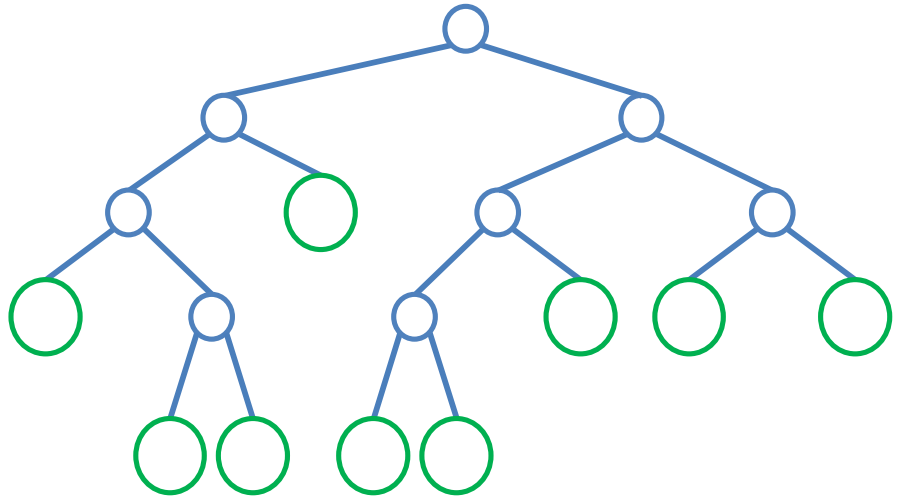
原图像

基于像素点检测边缘 ☹

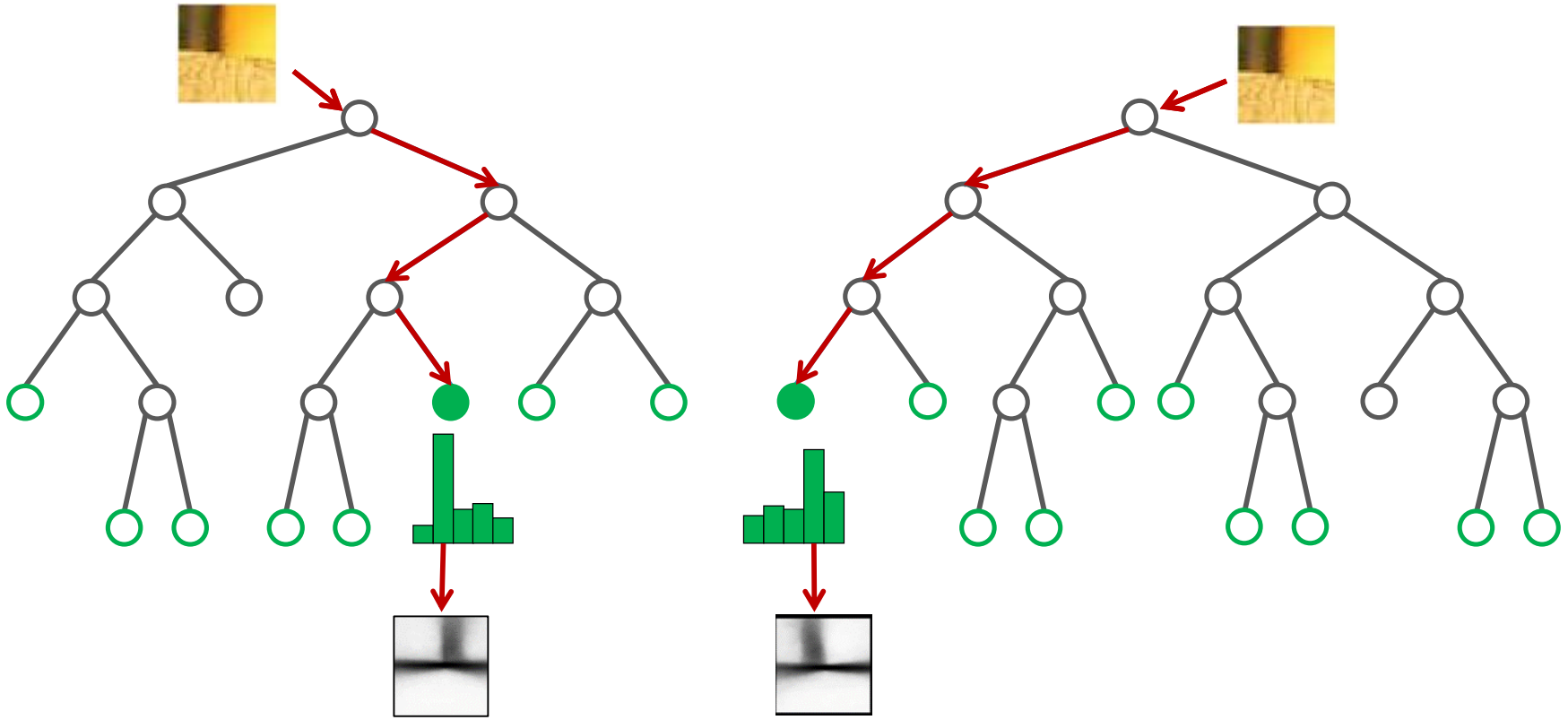
基于结构检测边缘 ☺

随机森林

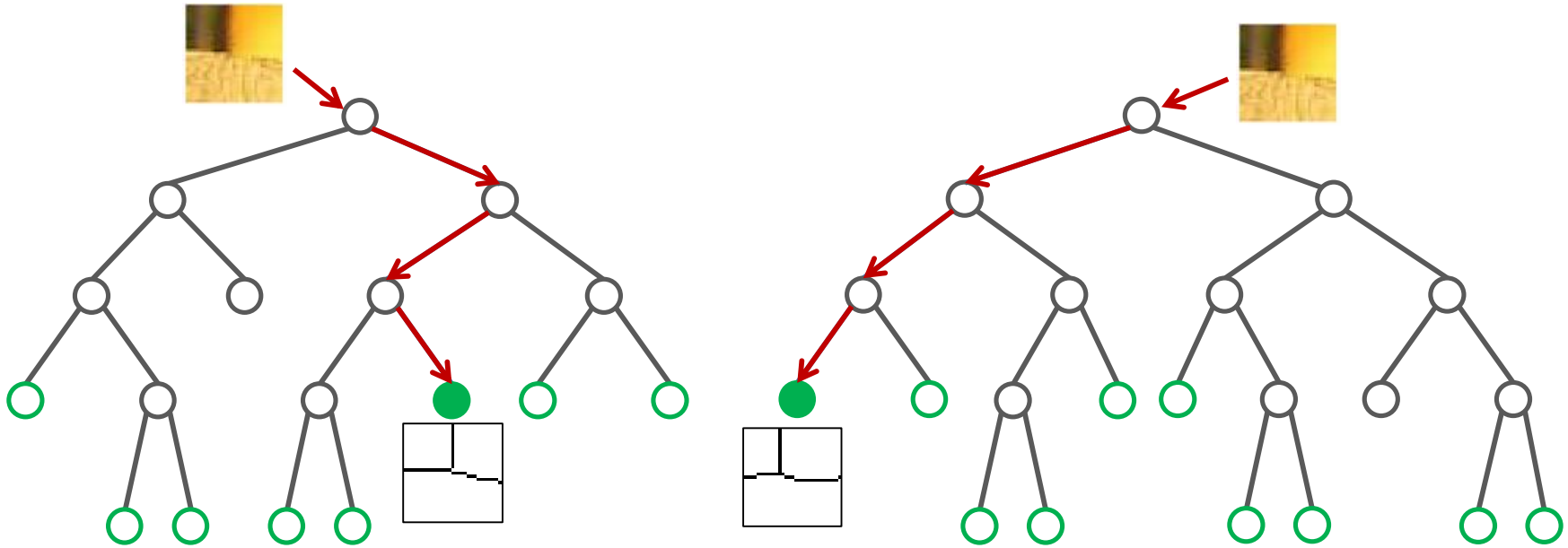
- 决策树
- 随机化学习
- 决策树组合模型——随机森林



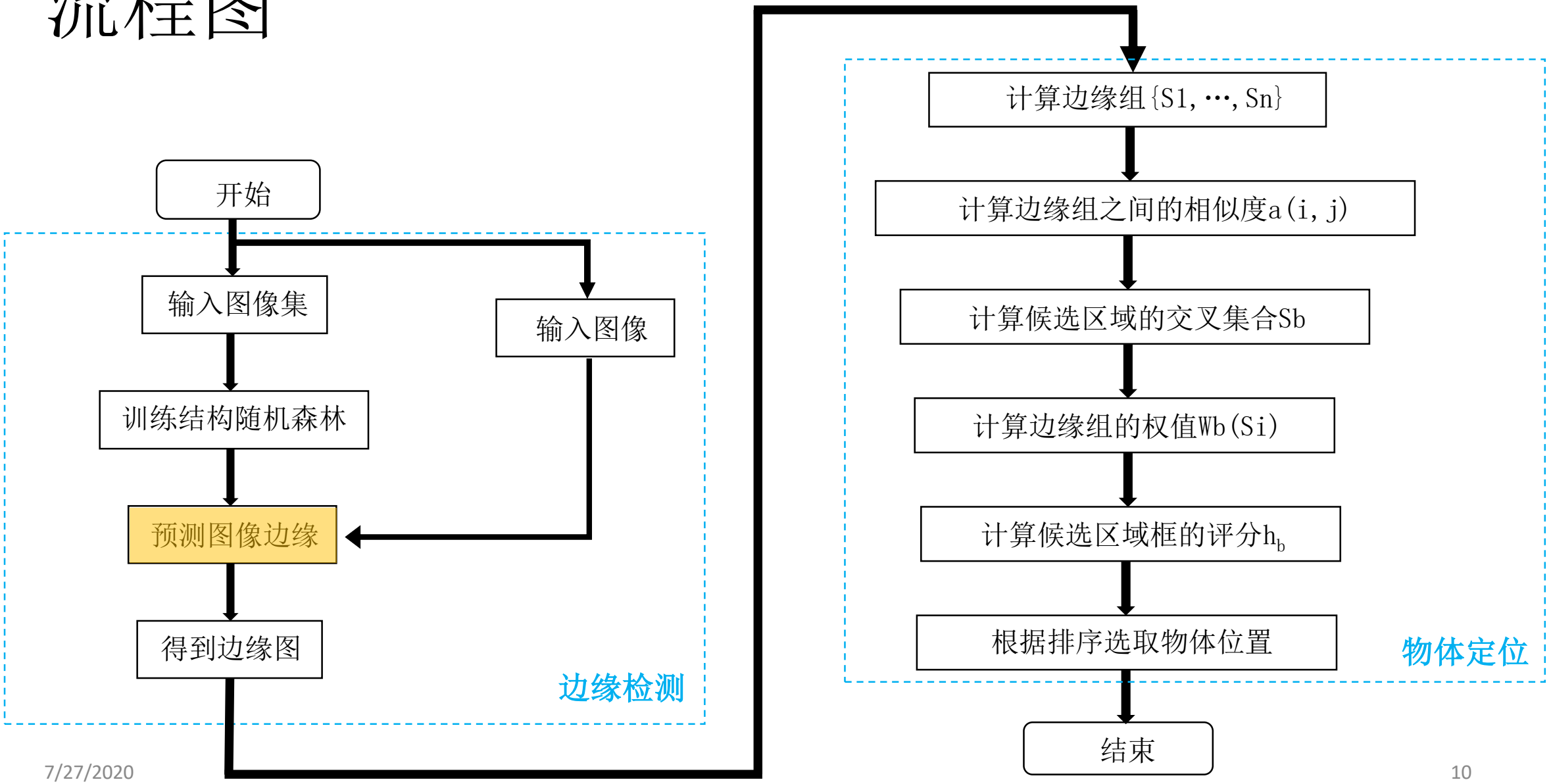
随机森林



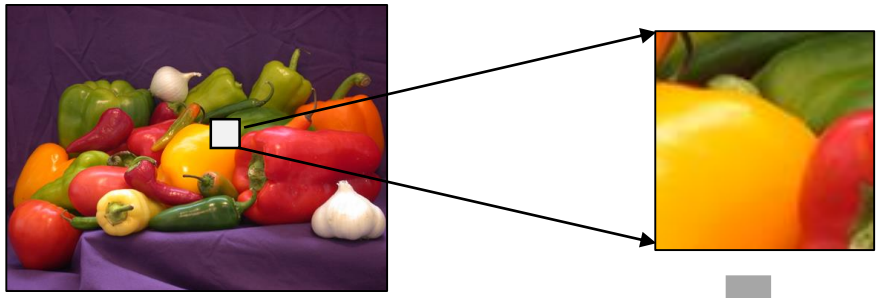
结构随机森林



流程图


开始
输入图像集
输入图像
训练结构随机森林
预测图像边缘
得到边缘图
边缘检测
计算边缘组{S1, …, Sn}
计算边缘组之间的相似度a(i, j)
计算候选区域的交叉集合Sb
计算边缘组的权值Wb(Si)
计算候选区域框的评分h_b
根据排序选取物体位置
物体定位
结束

边缘检测过程

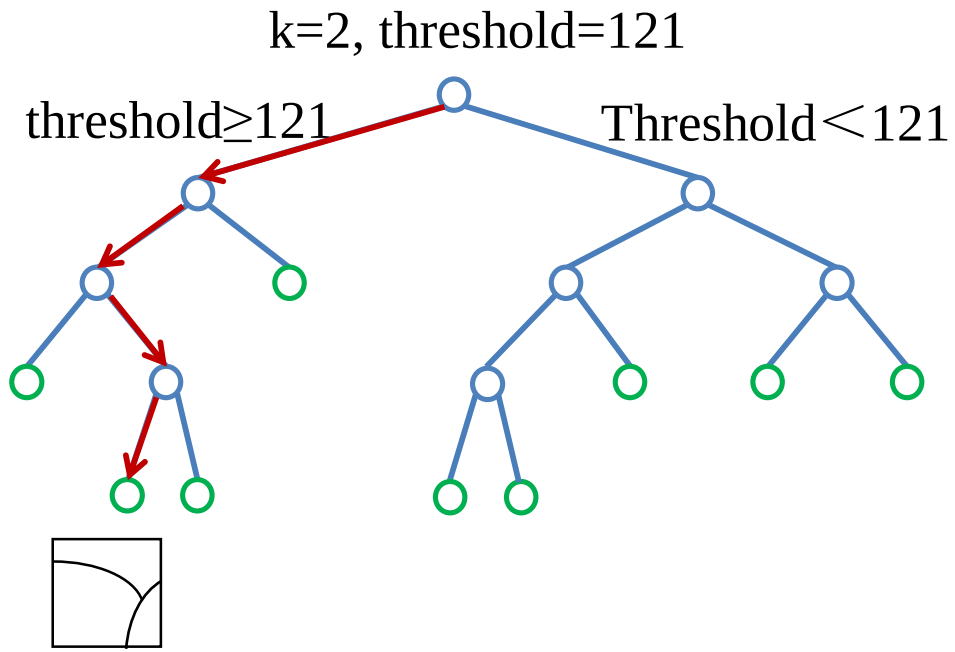


k=2

特征向量 v

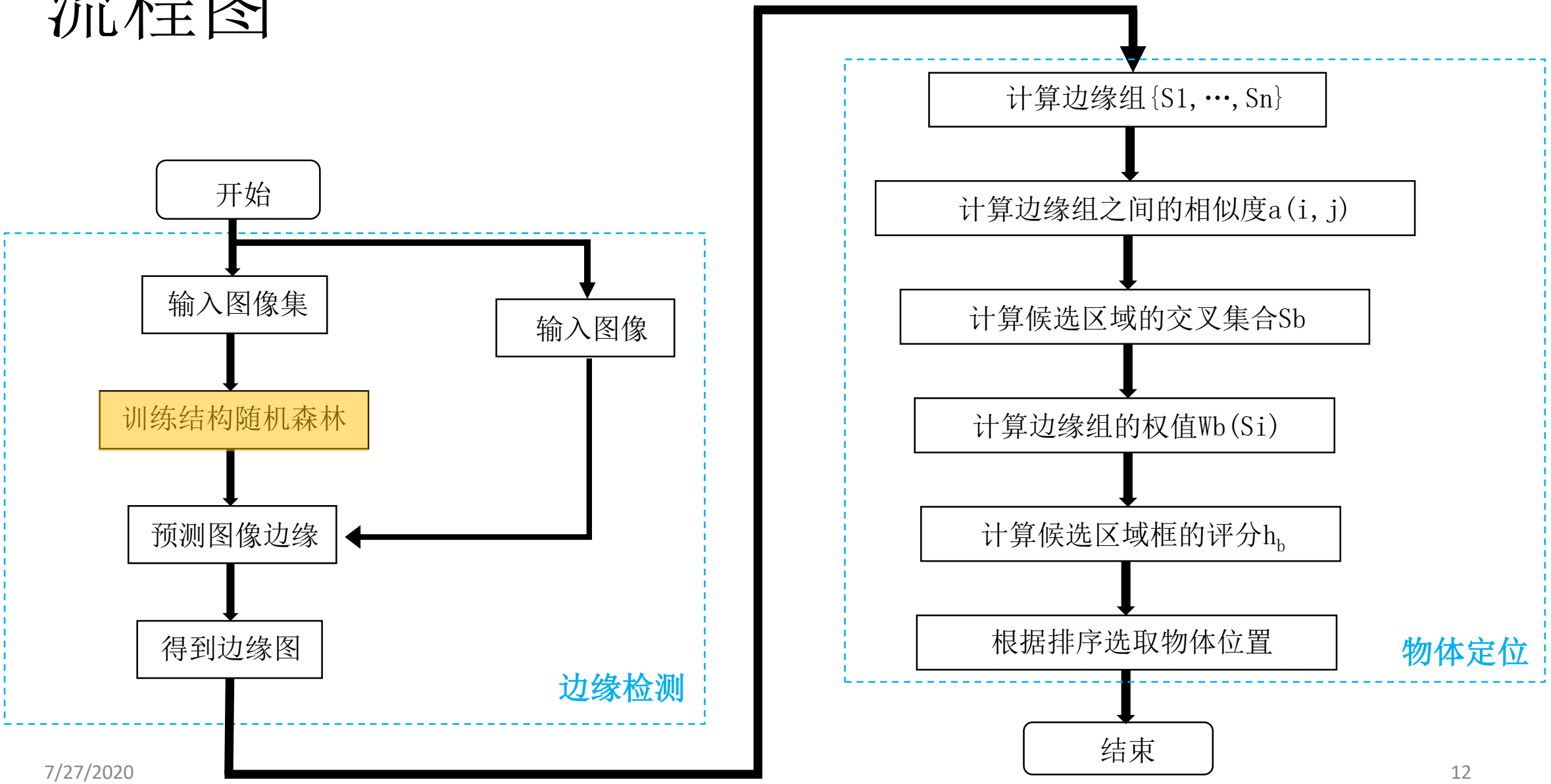
7228

value=145

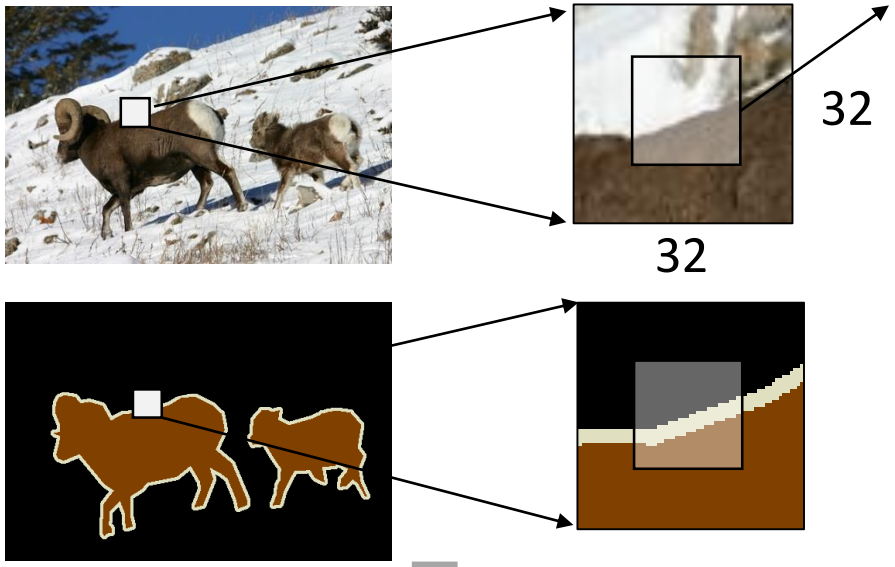


多棵树输出求平均，得到最后的mask

流程图


开始
输入图像集
输入图像
训练结构随机森林
预测图像边缘
得到边缘图
边缘检测
计算边缘组{S1, …, Sn}
计算边缘组之间的相似度a(i, j)
计算候选区域的交叉集合Sb
计算边缘组的权值Wb(Si)
计算候选区域框的评分h_b
根据排序选取物体位置
物体定位
结束

决策树训练过程



对第i维特征排序

	k维特征 1	2	3	4		k	标签	mask
8	7	253			…		0	
17	4	173			…		1	
25	3.7	121			…		0	
20	3.5	43			…		0	
n个patches	…	…	…	…	⋮	…	⋮	⋮
9	1.1	6			…		0	

理想状态下有边缘的图像块 ⟶ 标签为1

理想状态下无边缘的图像块 ⟶ 标签为0

$C_{16\times16}^{2} = 32640$种
判断像素值是否相同

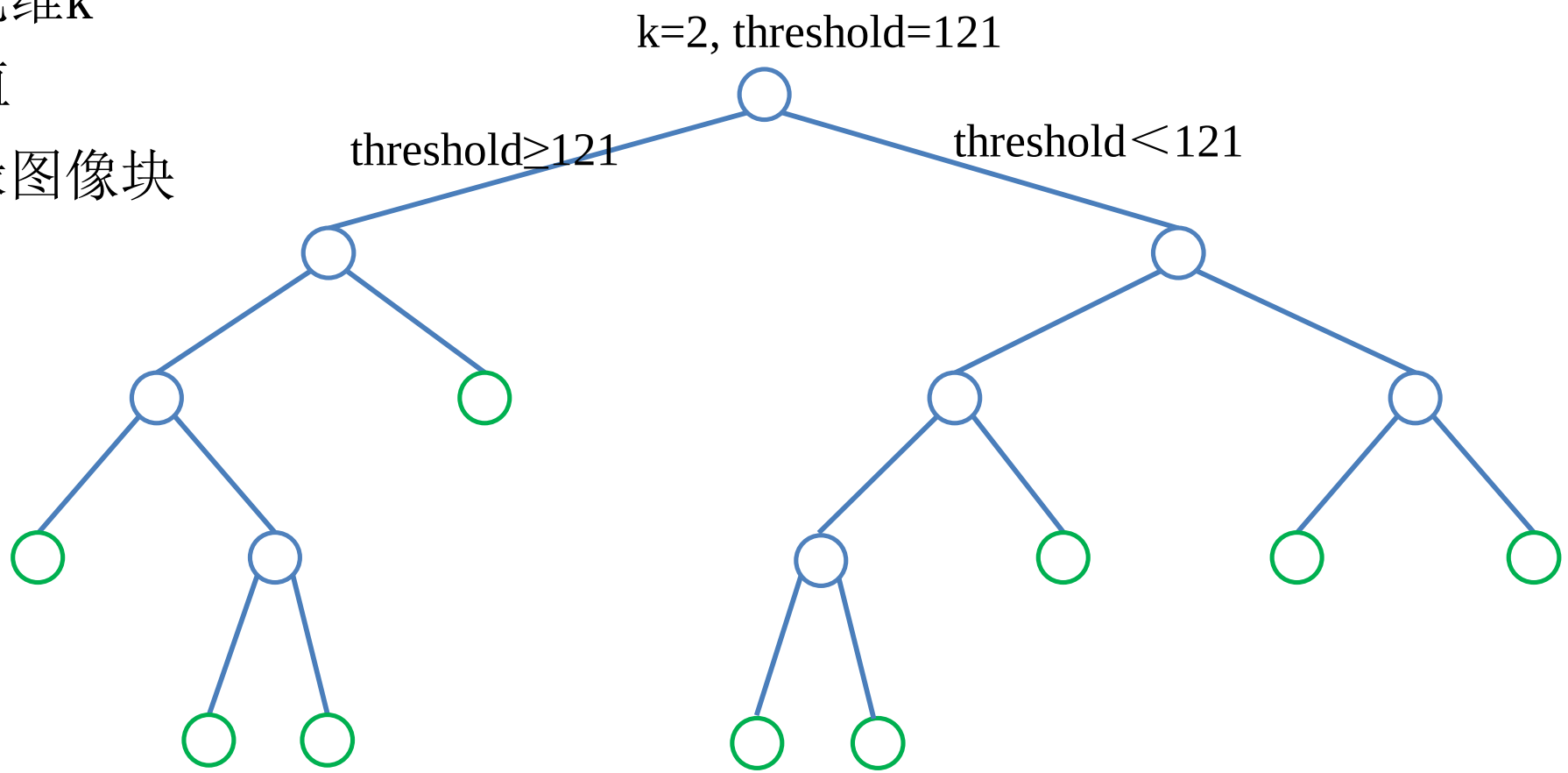
选取子集m=256

PCA降维，分为2类，划标签

大多数无边缘的标记为0，
大多数有边缘的标记为1。

决策树训练过程

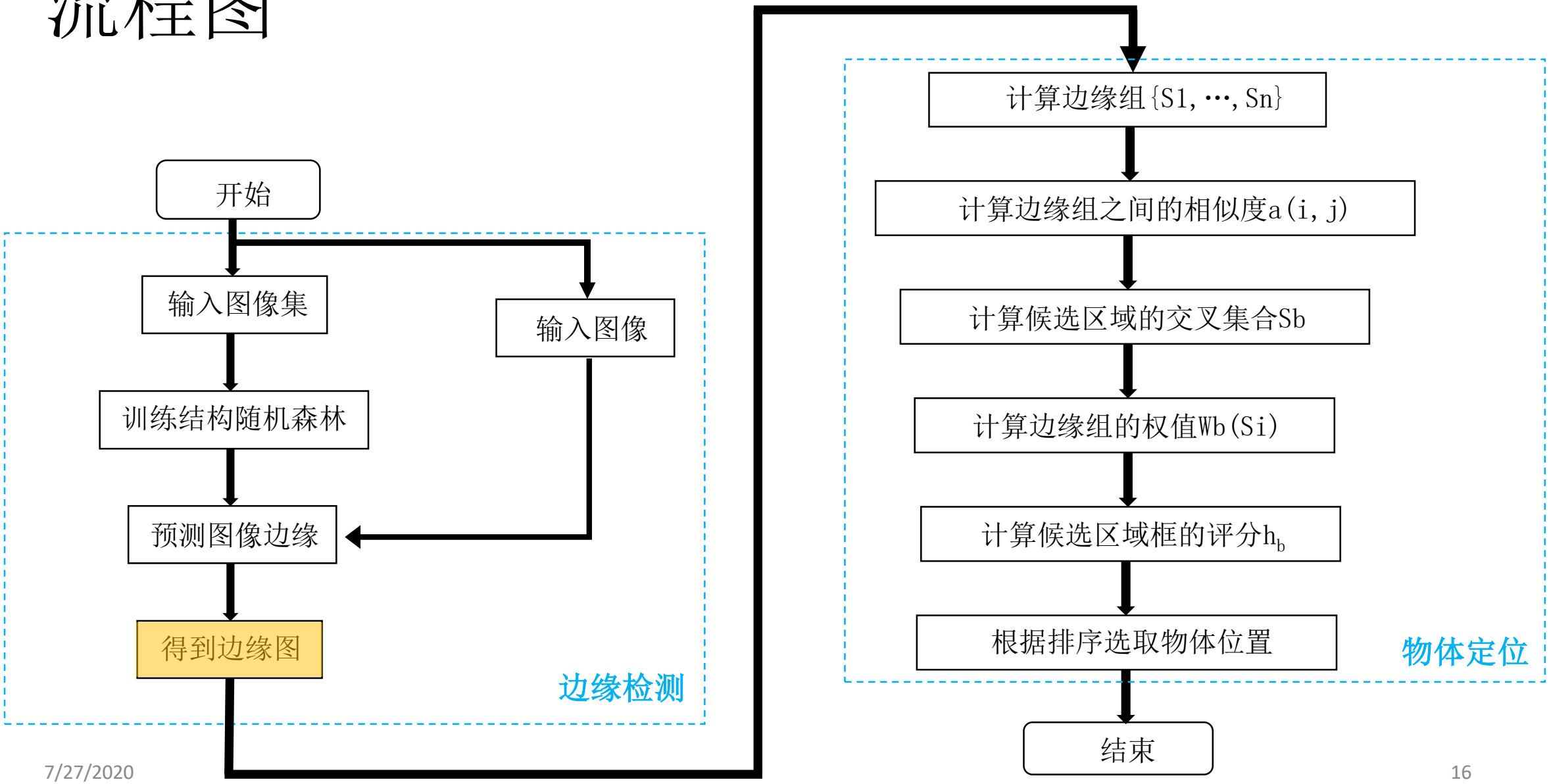
- 节点中放置
  - 第几维k
  - 阈值
  - 边缘图像块
- 迭代终止条件



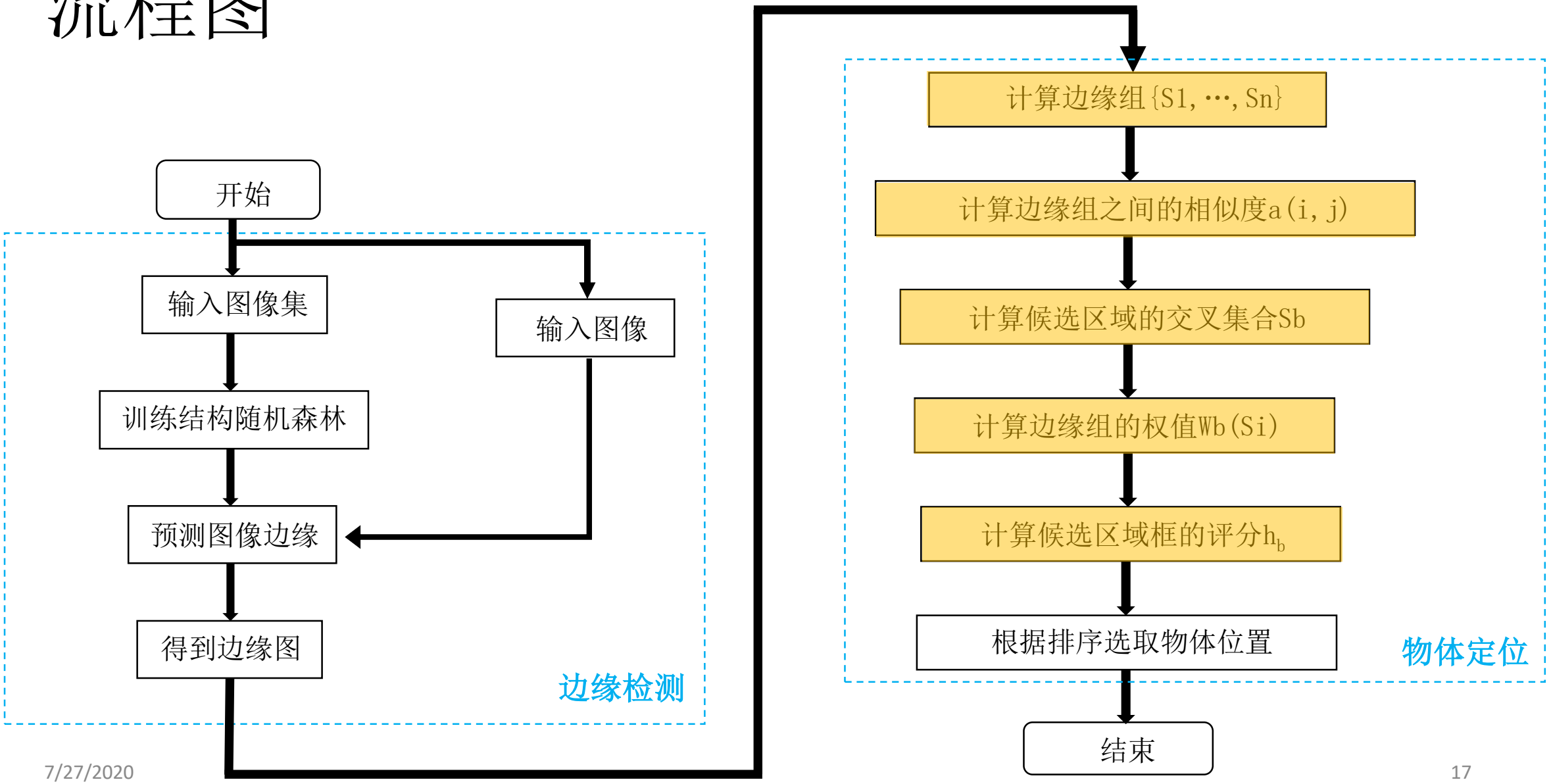
决策树训练过程

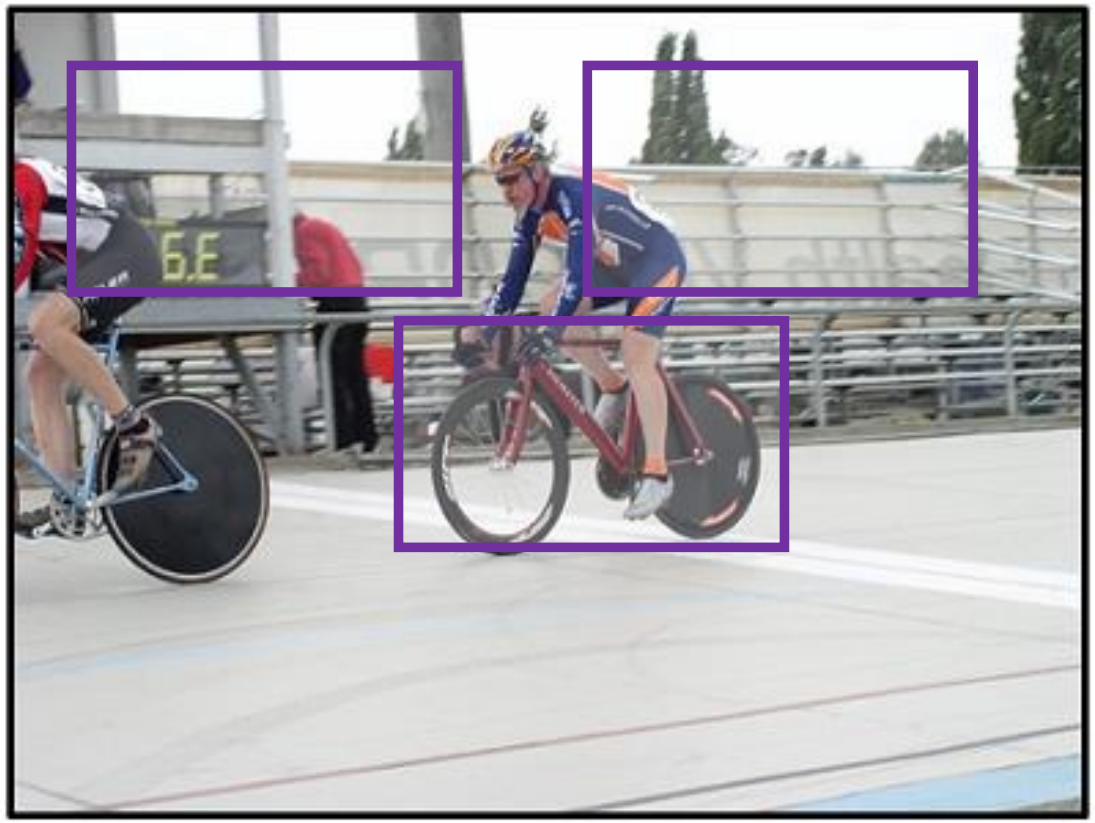
- 节点划分——信息增益
- $I(j) = H(S_j) - \sum_{k\in\{L,R\}} \frac{|S_j^k|}{|S_j|} H(S_j^k)$ $\quad\quad$ $H(s) = -\sum_y p_y log(p_y)$
- 特征向量v（每一个图像块7228个候选特征）

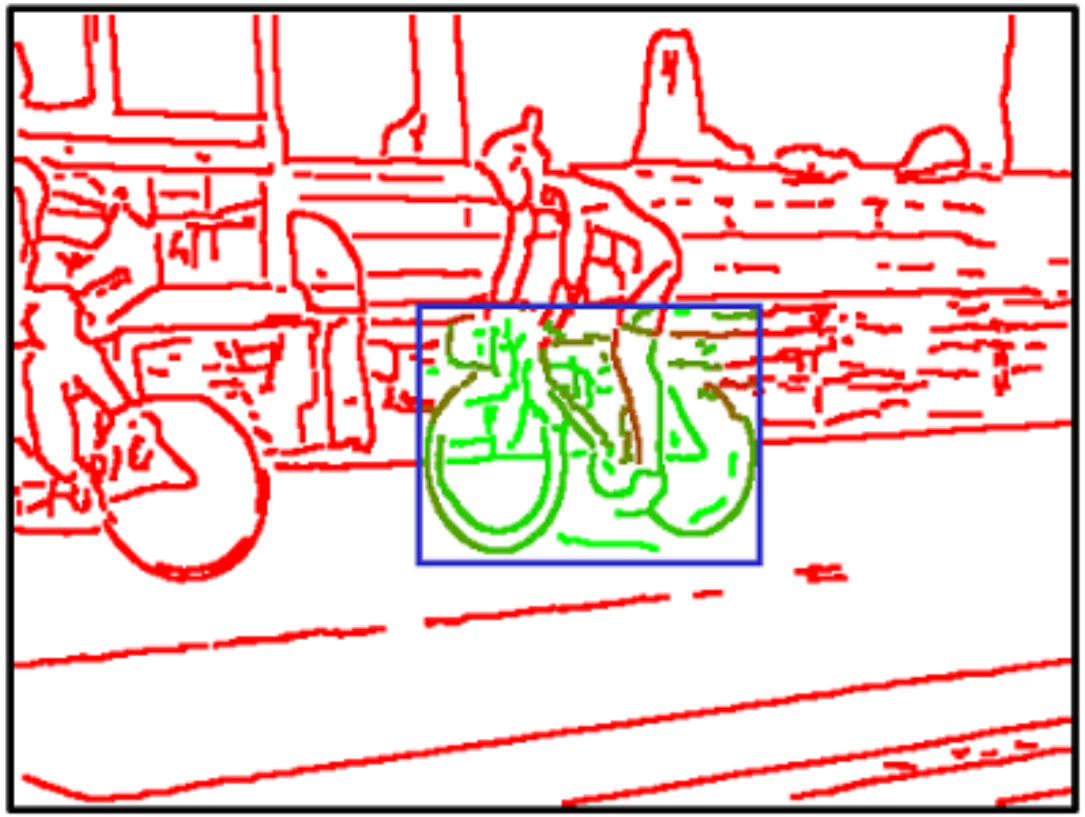
流程图


开始
输入图像集
输入图像
训练结构随机森林
预测图像边缘
得到边缘图
边缘检测
计算边缘组{S1, …, Sn}
计算边缘组之间的相似度a(i, j)
计算候选区域的交叉集合Sb
计算边缘组的权值Wb(Si)
计算候选区域框的评分h_b
根据排序选取物体位置
物体定位
结束

流程图


开始
输入图像集
输入图像
训练结构随机森林
预测图像边缘
得到边缘图
边缘检测
计算边缘组{S1, …, Sn}
计算边缘组之间的相似度a(i, j)
计算候选区域的交叉集合Sb
计算边缘组的权值Wb(Si)
计算候选区域框的评分h_b
根据排序选取物体位置
物体定位
结束





计算边缘组(edge groups)

- 边缘图→边缘修正→ 非最大值抑制
  →得到稀疏边缘图

- 每个像素点p
  - 边缘方向矢量向度m_p
  - 边缘方向θ_p

- $\sum_i^8 |\theta_a - \theta_i| < \frac{\pi}{2}$

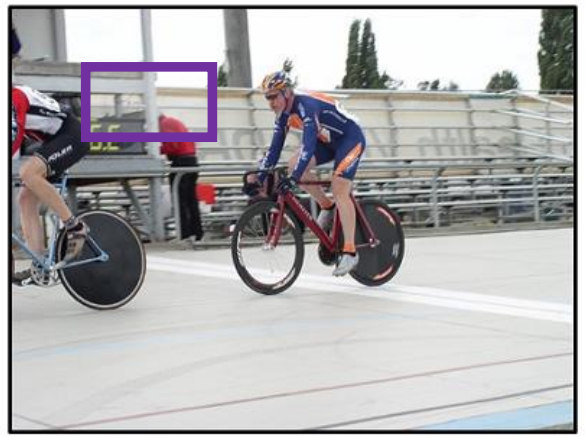
- 得到边缘组$\{S_1,\ldots,S_n\}$

θ_{x_1}	θ_{x_2}	θ_{x_3}
θ_{x_4}	θ_a	θ_{x_5}
θ_{x_6}	θ_{x_7}	θ_{x_8}

计算边缘组之间的相似度

- 相似度：$a(S_i, S_j) = |\cos(\theta_i - \theta_{ij})\cos(\theta_j - \theta_{ij})|^{\gamma}$
- 边缘组平均坐标x_i, x_y，平均方向θ_i, θ_j
- θ_{ij}：两坐标之间夹角；$\gamma = 2$
- 两个边缘组距离大于2时，$a(S_i, S_j) = 0$

计算边缘组的权值$w_b(s_i)$



- 假设在图像中，一个窗口框出一个矩形框
- 令$w_b(s_i) \in [0,1]$表示：若s_i完全处于候选框b以内，则$w_b(s_i) = 1$；若不是，则$w_b(s_i) = 0$。
- 令S_b为横跨候选框b边界的边缘组的集合。
- $w_b(s_i) = 1 - \max_T \prod_j^{|T|-1} a(t_j, t_{j+1})$

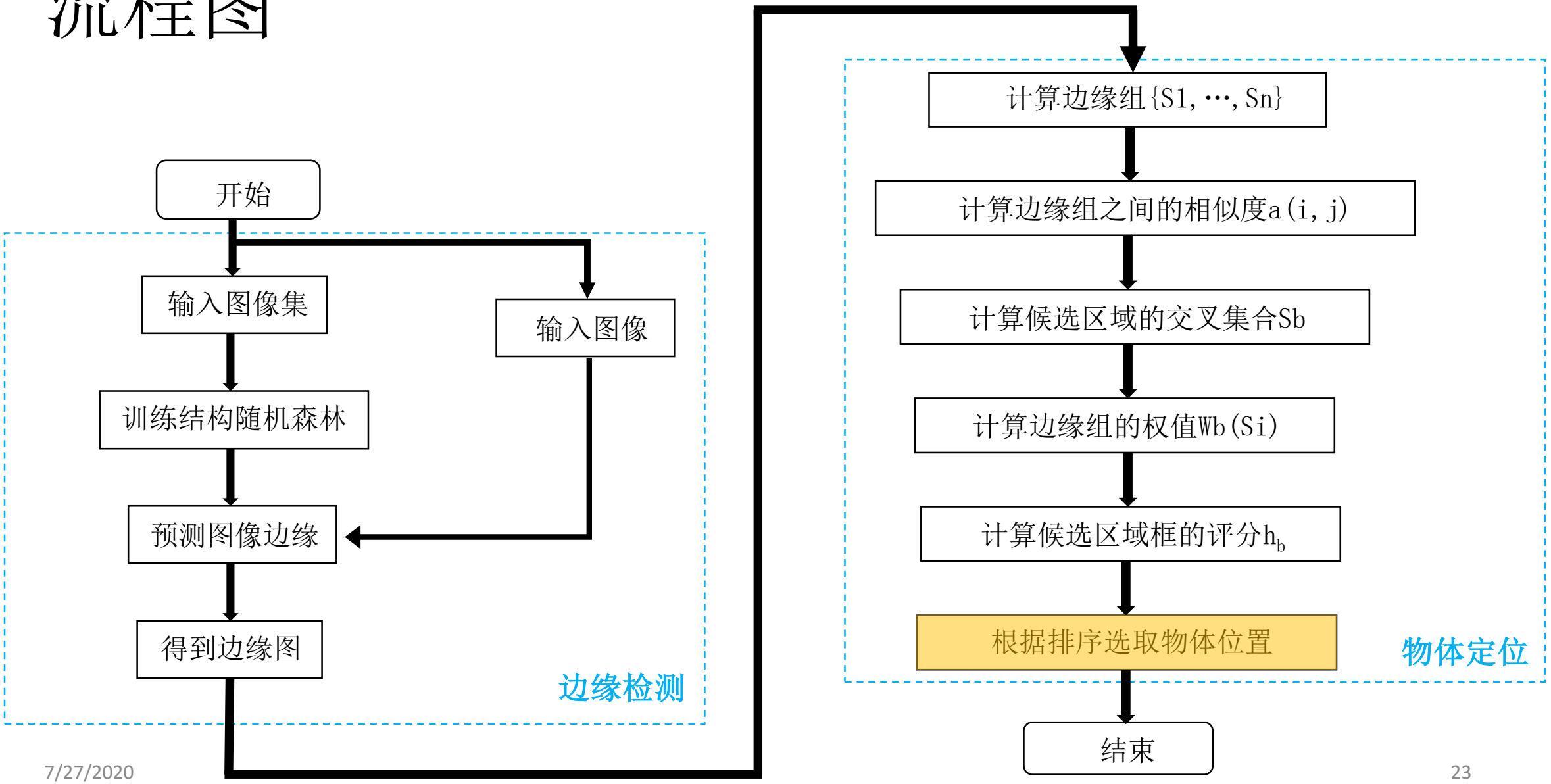
计算候选区域框的评分h_b

每个边缘组的权值

每个边缘内各个像素的梯度值

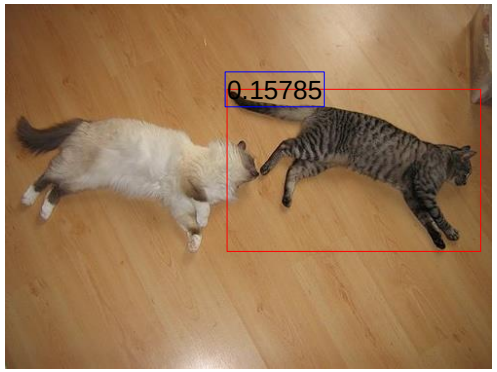
- 评分（特征值）：$h_b = \frac{\sum_i w_b(s_i) m_i}{2(b_w + b_h)^K}$
- 优化：$h_b^{in} = h_b - \frac{\sum_{p \in b^{in}} m_p}{2(b_w + b_h)^K}$

流程图

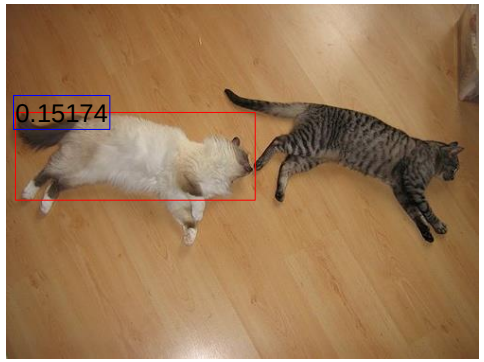

开始
输入图像集
输入图像
训练结构随机森林
预测图像边缘
得到边缘图
边缘检测
计算边缘组{S1, …, Sn}
计算边缘组之间的相似度a(i, j)
计算候选区域的交叉集合Sb
计算边缘组的权值Wb(Si)
计算候选区域框的评分h_b
根据排序选取物体位置
物体定位
结束

根据排序选取物体位置

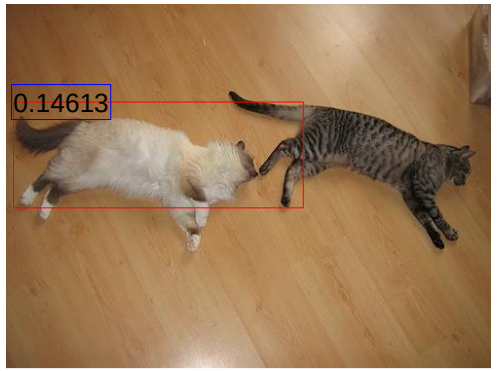
- 通过滑动窗口选取候选区域→对h_b^{in}的值（非零值）优化→进行排序
- → 得到物体候选区域



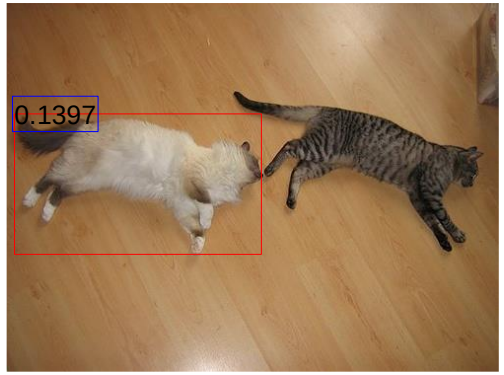
Rank 1



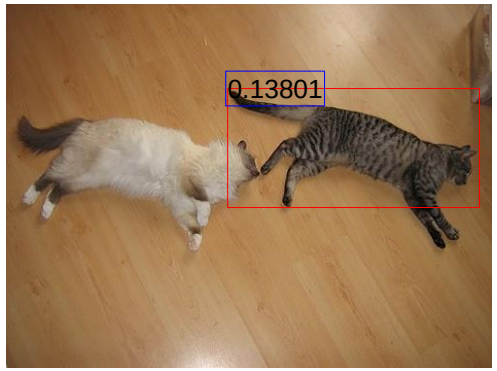
Rank 2



Rank 3



Rank 4

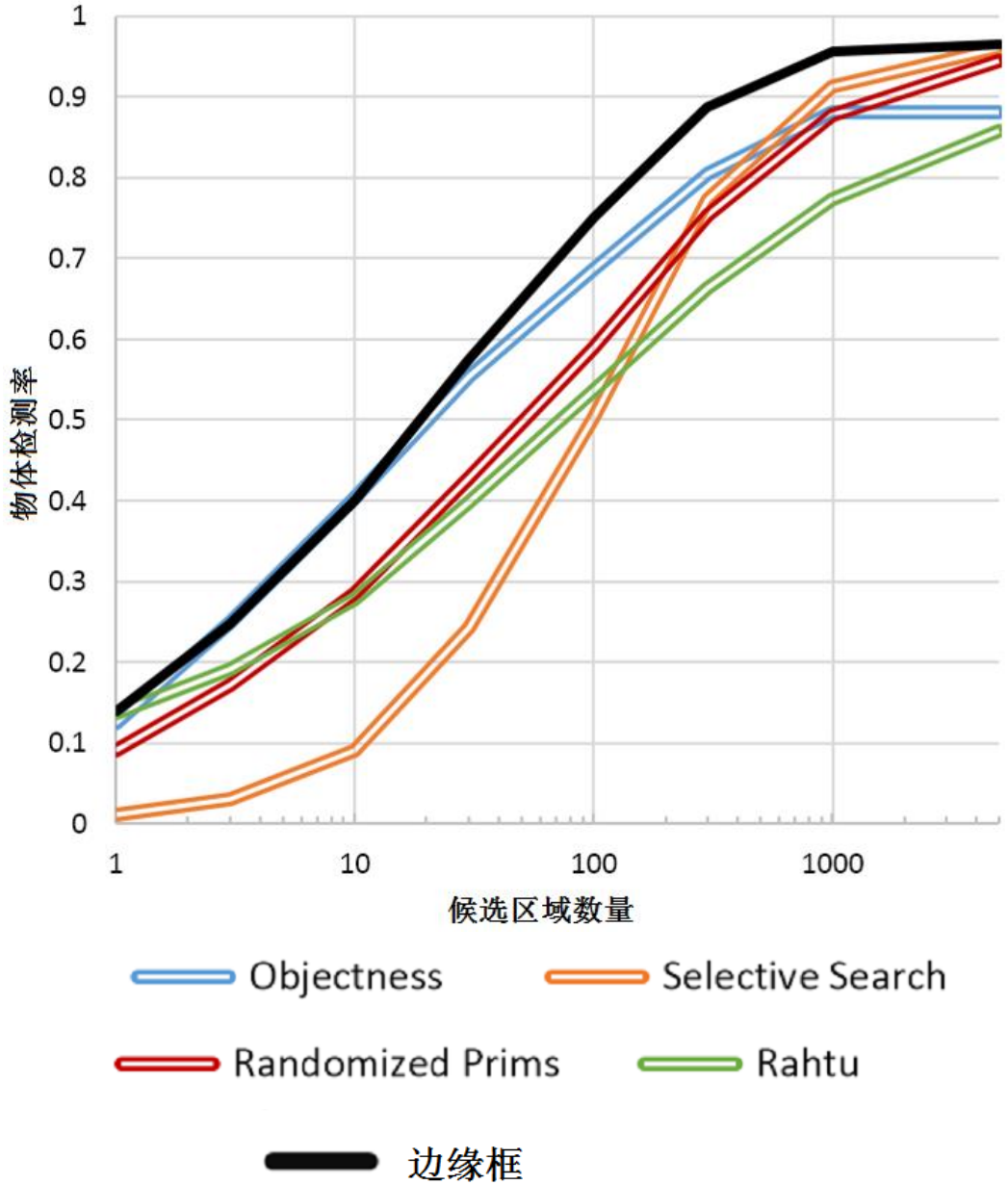


Rank 5

本算法的优点

原图像

去除边界轮廓图像


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
物体检测率
1
10
100
1000
候选区域数量
Objectness
Selective Search
Randomized Prims
Rahtu
边缘框

	曲线下面积	N@25%	N@50%	N@75%	召回率	运行时间
BING	0.20	292	-	-	29%	**0.2s**
Rantalankila	0.23	184	584	-	68%	10s
Objectness	0.27	27	-	-	39%	3s
Rand.Prim's	0.35	42	349	3023	80%	1s
Rahtu	0.37	29	307	-	70%	3s
Selective Search	0.40	28	199	1434	**87%**	10s
CPMC	0.41	15	111	-	65%	250s
边缘框	**0.46**	**12**	**108**	**800**	**87%**	**0.25s**

总结与展望

- 本算法中，基于结构随机森林的边缘检测算法效果非常好，可以应用到很多研究方向中（目标检测、图像分割、图像检索等）；
- 本算法中，直接从边缘图中进行物体检测的算法效果在一定的参数设置下，可以达到较高的物体检测率，同时物体检测的时间开销很少，与其他算法相比，有很高的优越性。

不足之处

- 相比于BING算法的检测速度300fps（C代码），边缘框的检测速度为4fps（Matlab混C代码）；
- 评分较高的候选区域均为近乎整幅图像，比如说，图像中有8个人，那么该算法评分最高的候选区域肯定是同时包含了这8个人的，这一点不足，使得这个算法的含金量大大缩水。在实际应用中，我们需要的效果往往是最高评分的几个候选区域最好是单独的人，而不是8个人一起。

不足之处

输入RGB图像

Idea： 将训练与学习的过程+本算法级联起来，应用到物体定位中。

thanks!

附录1

```
// 定位所有候选区域
  boxes.resize(0); int arRad, scNum; float minSize=sqrt(_minBoxArea);
  arRad = int(log(_maxAspectRatio)/log(_arStep*_arStep));
  scNum = int(ceil(log(max(w,h)/minSize)/log(_scStep)));
  for( int s=0; s<scNum; s++ ) {
    int a, r, c, bh, bw, kr, kc, bId=-1; float ar, sc;
    for( a=0; a<2*arRad+1; a++ ) {
      ar=pow(_arStep,float(a-arRad)); sc=minSize*pow(_scStep,float(s));
      bh=int(sc/ar); kr=max(2,int(bh*_rcStepRatio));
      bw=int(sc*ar); kc=max(2,int(bw*_rcStepRatio));
      for( c=0; c<w-bw+kc; c+=kc ) for( r=0; r<h-bh+kr; r+=kr ) {
        Box b; b.r=r; b.c=c; b.h=bh; b.w=bw; boxes.push_back(b);
      }
    }
```

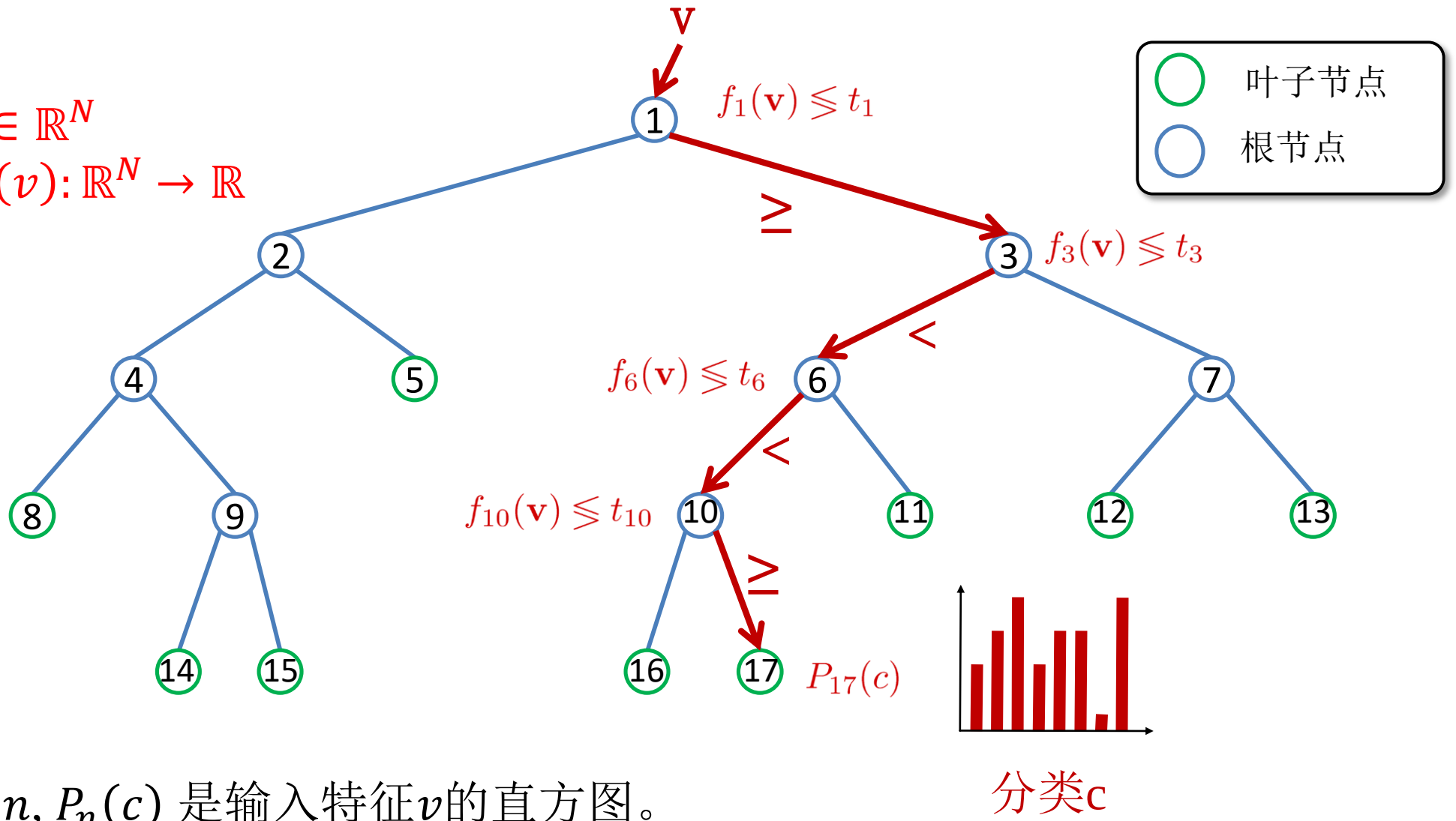
附录2

```
//进行NMS处理
void EdgeBoxGenerator::refineBox( Box &box )
{
  int rStep = int(box.h*_rcStepRatio);
  int cStep = int(box.w*_rcStepRatio);
  while( 1 ) {
    // prepare for iteration
    rStep/=2; cStep/=2; if( rStep<=2 && cStep<=2 ) break;
    rStep=max(1,rStep); cStep=max(1,cStep); Box B;
```

```
 // search over r start
    B=box; B.r=box.r-rStep; B.h=B.h+rStep; scoreBox(B);
    if(B.s<=box.s) { B=box; B.r=box.r+rStep; B.h=B.h-rStep; scoreBox(B); }
    if(B.s>box.s) box=B;
    // search over r end
    B=box; B.h=B.h+rStep; scoreBox(B);
    if(B.s<=box.s) { B=box; B.h=B.h-rStep; scoreBox(B); }
    if(B.s>box.s) box=B;
```

二叉树（决策树）

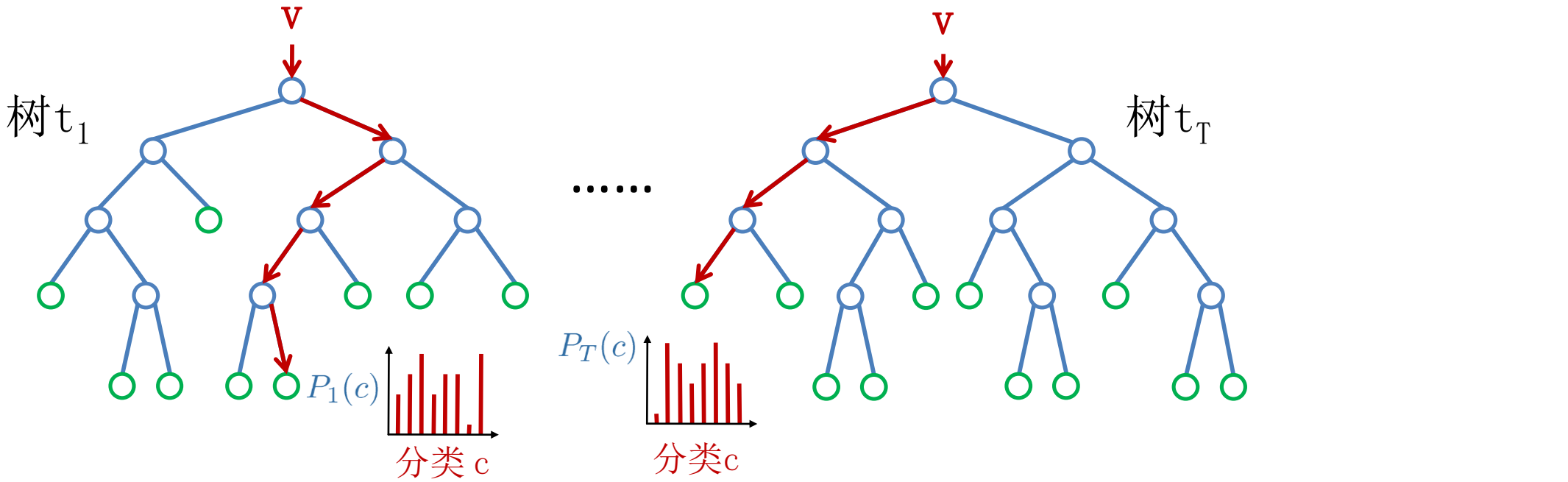
- 特征向量 $v \in \mathbb{R}^N$
- 分裂函数 $f_n(v): \mathbb{R}^N \rightarrow \mathbb{R}$
- 阈值 $t_n \in \mathbb{R}$
- 分类 $P_n(c)$



在节点n, $P_n(c)$ 是输入特征v的直方图。

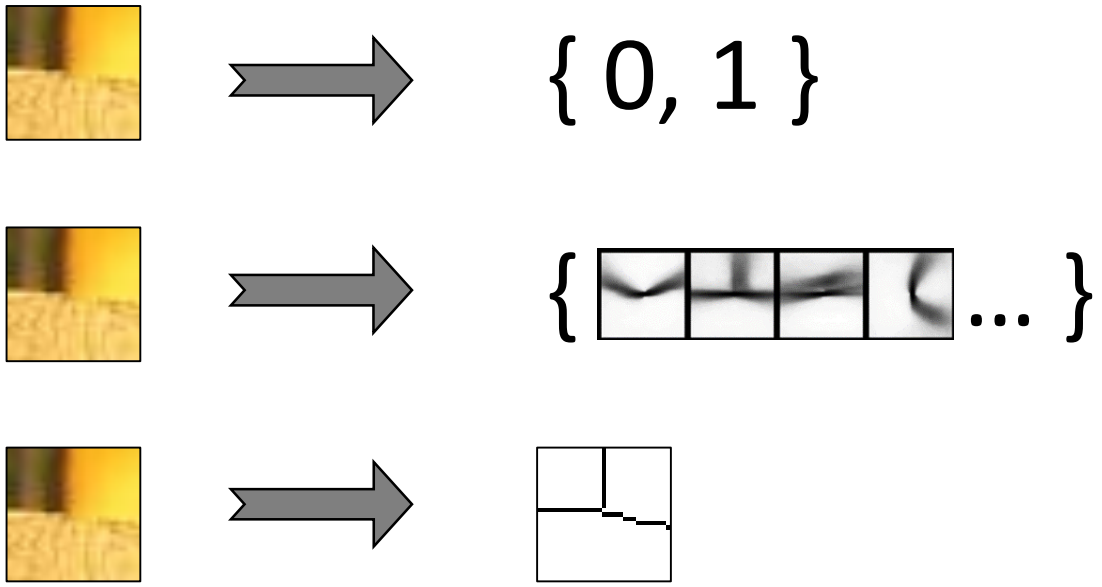
随机森林

- 森林由多棵决策树组成



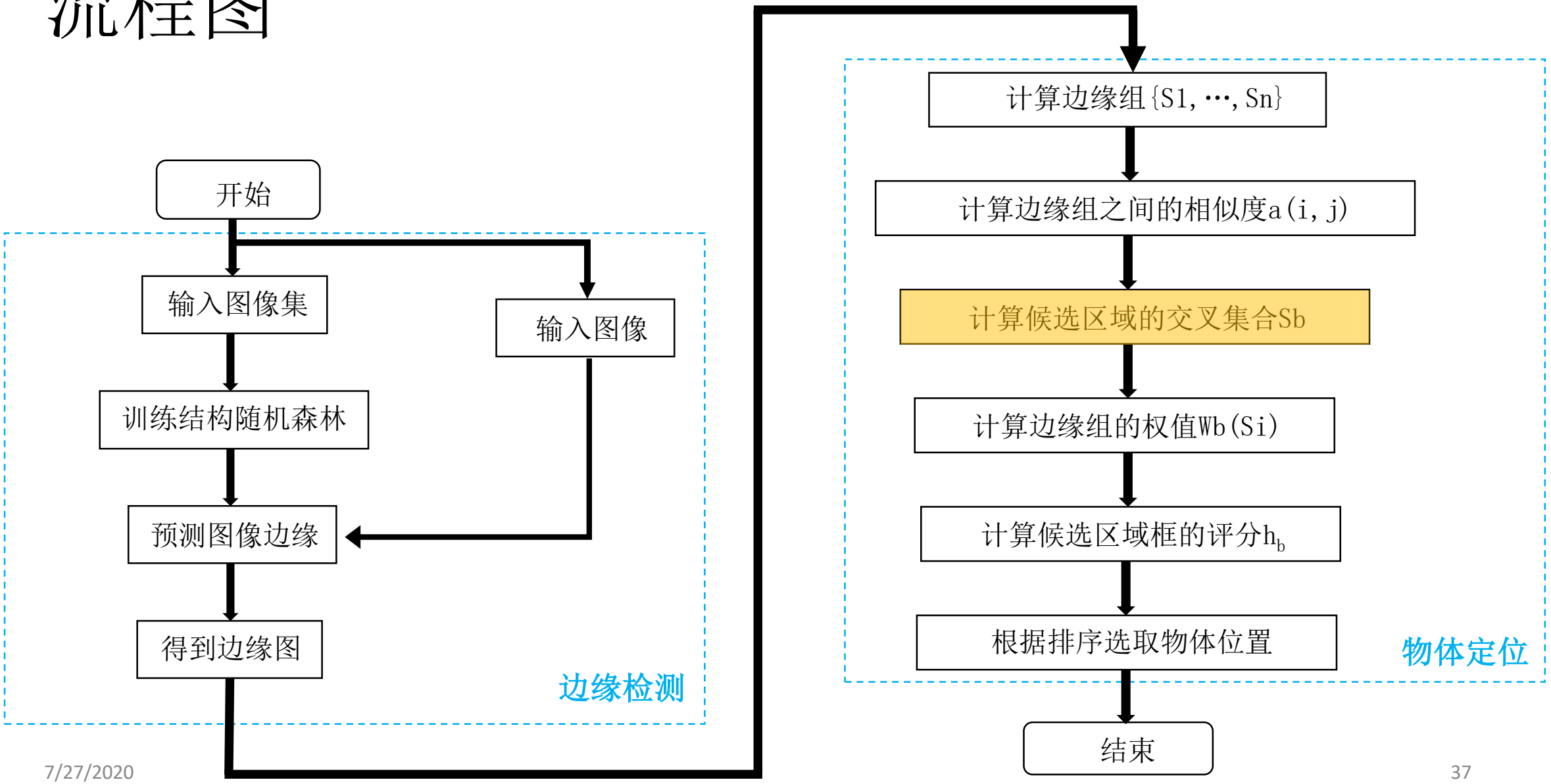
$$P(\mathrm{c}|\mathrm{v}) = \frac{1}{T}\sum_{t=1}^{T} P_t(\mathrm{c}\,|\,\mathrm{v})$$

提升输出空间



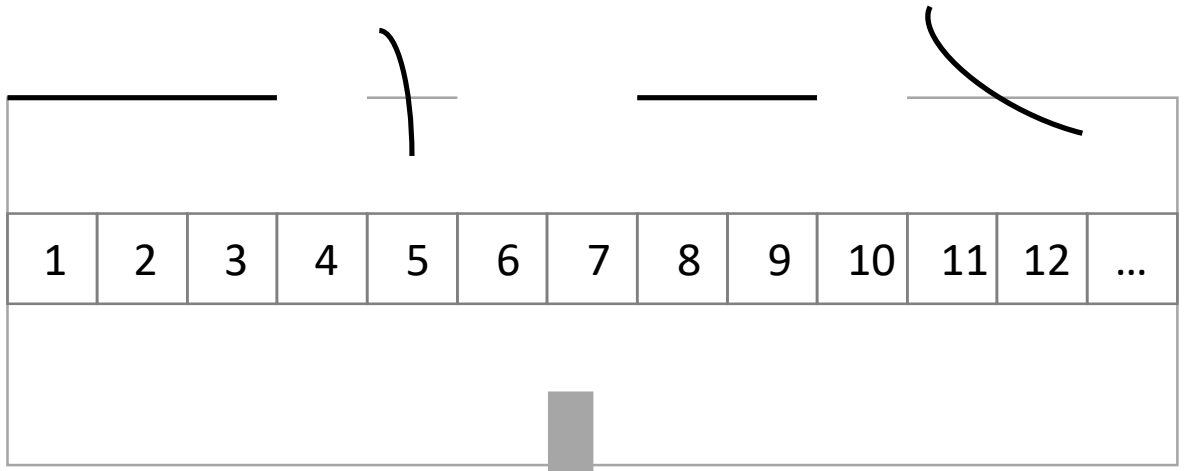
bits	year
1	CVPR06
8	CVPR13
256	ICCV13

流程图


开始
输入图像集
输入图像
训练结构随机森林
预测图像边缘
得到边缘图
边缘检测
计算边缘组{S1, …, Sn}
计算边缘组之间的相似度a(i, j)
计算候选区域的交叉集合Sb
计算边缘组的权值Wb(Si)
计算候选区域框的评分h_b
根据排序选取物体位置
物体定位
结束

计算候选区域的交叉集合S_b

- 候选区域框的每条边建立两个数组：L_r K_r



L_r	1	0	2	0	0	3	0	4

K_r	1	1	1	2	3	4	5	6	6	7	8	8

有序表L_r中非零值的下标代入K_r中查找得到的下标就是交叉的边缘组序列