中国科学技术大学

面向移动机器人的行人追踪跟随的研究与实现

报告人：韦清

学号：PB15000027

院系：少年班学院

指导老师：陈小平 教授

目　录

PART ONE

研究背景与目标

研究背景与目标

行人跟随技术在移动服务机器人上的应用场景：

个人/家庭服务机器人提供家政服务

在医院等公共场所对用户进行指引等服务

研究目标：

实现一个通过移动机器人的传感器得到目标用户的位置信息并对其进行跟随的系统

- 实时性、稳定性、准确性
- 能够长时间运行

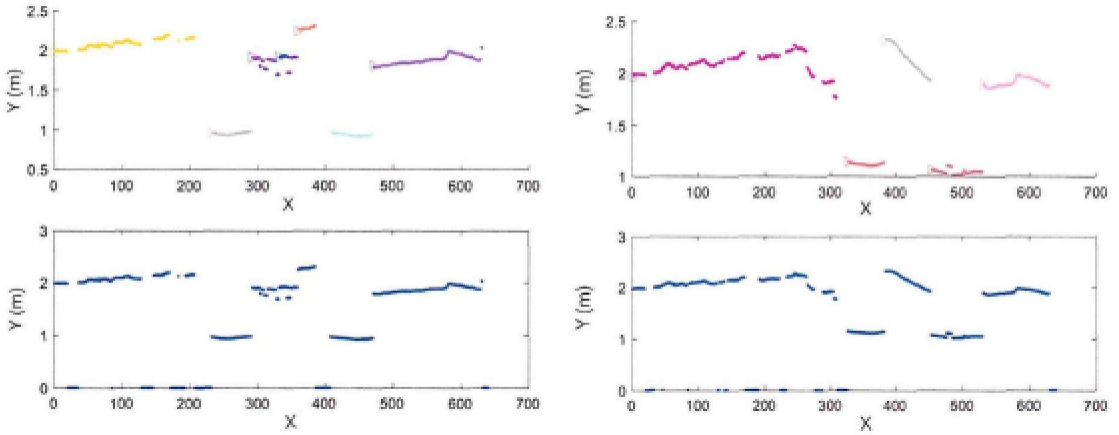
2

PART TWO

行人追踪算法的比较与选择

机器人行人追踪实现

- 基于激光传感器信息的行人追踪
  - 常通过提取行人的腿部信息实现行人追踪
- 基于视觉信息的行人追踪
  - 利用Kinect深度相机获得RGB-D图像
  - 利用计算机视觉算法得到行人在RGB图像上的位置
  - 通过将RGB图像与深度图像对齐，得到行人的3维坐标



激光行人腿部识别

Kinect相机得到的深度图像和RGB图像

算法名称	简介	优点	缺点
光流法	通过计算图像序列中像素的瞬时速度计算相邻帧之间物体运动信息	在比较理想的情况下，它能够检测独立运动的对象，不需要预先知道场景的任何信息，可以很精确地计算出运动物体的速度，并且可用于摄像机运动的情况	光流场计算量大，处理速度慢，难以实时；噪声、多光源阴影和遮挡等因素会对光流场分布的计算结果造成严重影响；只能检测运动目标
卡尔曼滤波	对目标的运动模型建模，估计目标在下一帧的位置	建模了目标的运动模型，易于实现	只适用于线性且呈高斯分布的系统，不对目标的特征进行建模，判别力差
粒子滤波	首先在全图撒大量粒子，统计这些粒子与目标的匹配程度来确定目标位置并调整下一帧计算时的粒子分布	可以有效应对遮挡情况，易于实现；相对于卡尔曼滤波，可以适用于非线性、非高斯系统	需要采样大量粒子，耗时较高
均值漂移	基于概率密度分布的跟踪方法，使目标的搜索一直沿着概率梯度上升的方向，迭代收敛到概率密度分布的局部峰值上。	算法复杂度小，是无参数算法	精度低，缺乏模板更新算法；没有利用目标在空间中的运动方向和速度等信息，当周围环境存在干扰时易丢失目标
Struck	把跟踪的目标作为前景，利用在线学习或离线训练的检测器来区分前景目标和背景，从而得到前景目标的位置	考虑了背景信息，判别力强，可以应对遮挡	在线更新容易引入噪声样本，从而导致跟踪器漂移，在线更新分类器时间开销较大
KCF	基于循环矩阵的核跟踪方法，利用傅立叶变换快速实现了检测的过程。在训练分类器时，一般认为离目标位置较近的是正样本，而离目标较远的认为是负样本。	速度快，精度高，可以高效的融合多种特征	尺度不变，滤波器在局部空间中进行搜索，完全遮挡后无法恢复，有边界效应，影响追踪效果
★ **CSR-DCF**	在KCF的基础上加入空间和通道置信度以及尺度估计。	精度高，可以适应不同尺度，速度较快	滤波器在局部空间中进行搜索，完全遮挡后无法恢复
TLD	由追踪、学习、检测三部分组成。追踪器逐帧追踪目标，检测器定位所有当前为止观察到的外观，在必要时纠正追踪器；学习器会评估检测器的错误并更新。	可以应对大尺度变换以及遮挡问题，失败后恢复及时	准确度较差，结果经常跳跃
GOTURN	神经网络算法，以视频当前帧和上一帧作为输入，输出当前帧的目标所在区域的位置。	速度快，可达到100FPS；对于视角的变化、光照、变形等具有鲁棒性。	准确度不高，不能很好地处理遮挡
MIL	使用在线学习分类器，在上一帧目标位置周围搜索。分类器将当前位置周围的小邻域都看作是潜在的正样本	可以应对部分遮挡	不能可靠地报告跟踪失败，无法从完全遮挡中恢复

3

PART THREE

CSR-DCF算法的分析与优化

CSR-DCF算法

- 特征提取：HOG + ColorName + Gray
- 滤波器h_t：使用傅里叶变换提高运行速度。
  - 根据滤波器来计算相关响应 $\tilde{g}(h) = \sum_{d=1}^{N_c} f_d * h_d$
    - $*$ 表示循环相关运算（circular correlation）
    - d为特征通道个数，f_d为特征d在搜索窗口的特征向量，h_d为特征d的滤波器。
  - 滤波器的拟合：最小化 $\varepsilon(h) = \|\tilde{g}(h) - g\|^2 + \lambda\|h\|^2$
- 多特征通道计算优化：通道置信度w
  - 采用通道置信度来计算相关响应 $\tilde{g}(h) = \sum_{d=1}^{N_c} f_d * h_d \cdot \tilde{w}_d$
- 尺度和非矩形区域优化：空间置信度m
  - m为一个元素值为0或1的矩阵，由各像素点的颜色和它离上一帧目标位置的距离得到
  - 将m与滤波器h进行点乘，为相关滤波加上空域限制 $h_c - m \odot h \equiv 0$
- 尺度s_t：使用DSST算法进行估计和更新

CSR-DCF算法

定位和尺度估计

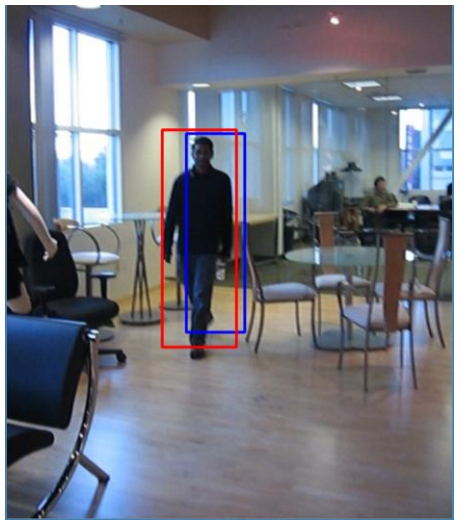
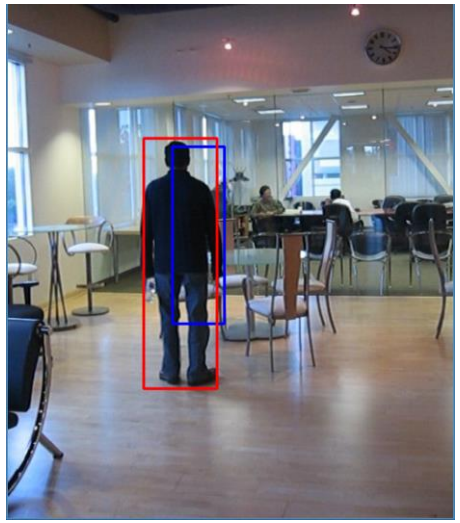
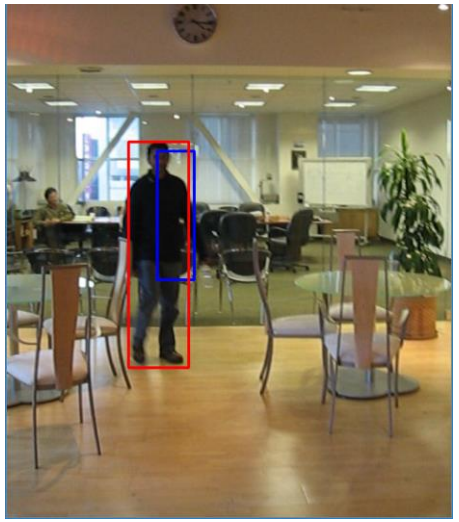
- 计算当前帧中目标位置P_t：在当前帧中，以上一帧的目标位置P_{t-1}为中心的搜索区域中提取特征f，并与得到的滤波器h_{t-1}得到相关响应，通道置信度为w_{t-1}。
- 计算h_{t-1}和当前图像在位置P_{t-1}特征f之间的相关响应，通道置信度为w；
- 根据各通道的响应，估计检测通道置信度$\tilde{\mathrm{w}}^{(det)}$；
- 根据目标所在位置P_t，使用DSST算法估计新的尺度s_t。

更新

- 提取前景和背景的色调直方图$\tilde{c}^f$和$\tilde{c}^h$；
- 更新c^f和c^{h}：
  - $c^f = (1-\eta_c)c_{t-1}^f + \eta_c\tilde{c}^f$
  - $c^b = (1-\eta_c)c_{t-1}^b + \eta_c\tilde{c}^b$
- 计算空间置信度图m；
- 根据空间置信度图拟合一个新的滤波器$\tilde{\mathrm{h}}$；
- 根据h估计通道学习置信度$\tilde{\mathrm{w}}^{(\mathrm{lrn})}$；
- 更新通道置信度：
  - $\tilde{w} = \tilde{\mathrm{w}}^{(det)} \odot \tilde{\mathrm{w}}^{(\mathrm{lrn})}$
  - $w_t = (1-\eta)w_{t-1} + \eta\tilde{w}$
- 更新滤波器$\mathrm{h_t} = (1-\eta)h_{t-1} + \eta\tilde{\mathrm{h}}$

CSR-DCF算法--存在的问题

注：蓝色框为CSR-DCF算法的输出结果，红色框为数据集的ground truth

误差累积导致bbox逐渐偏离目标



追踪过程中可能会因为遮挡等原因而误追踪到其他行人或物体，且不能可靠地报告追踪失败

解决方案

加入行人检测器来报告跟踪失败和辅助恢复

- 特征提取：HOG + 颜色直方图
  - HOG：梯度方向直方图
    - 用于描述物体的边缘和纹理，是行人检测中最为常用的特征描述子之一
  - 颜色直方图：图像区域中的颜色分布
    - 采用HSV作为颜色空间，分别表示颜色的色调(hue)、饱和度(saturation)和亮度(value)
- 分类器：SVM
  - 有监督学习：使用图像背景作为负样本，目标所在区域作为正样本进行训练
  - 输入图像区域的特征向量，可得到该区域是正样本或负样本的概率

解决方案

加入行人检测器来报告跟踪失败和辅助恢复

- 初始化分类器
  - 使用追踪器前若干帧的输出来对分类器进行训练
- 验证追踪结果，报告追踪失败
  - 为了减少对速度的影响，每10帧使用分类器对追踪器的结果进行一次验证
- 追踪器失败后辅助恢复
  - 当追踪器错误或失败后，对每个输入帧通过滑窗法得到一些不同尺度上的搜索窗口，由分类器判断每个搜索窗口是否属于目标，若属于，则追踪器使用该窗口继续进行追踪。

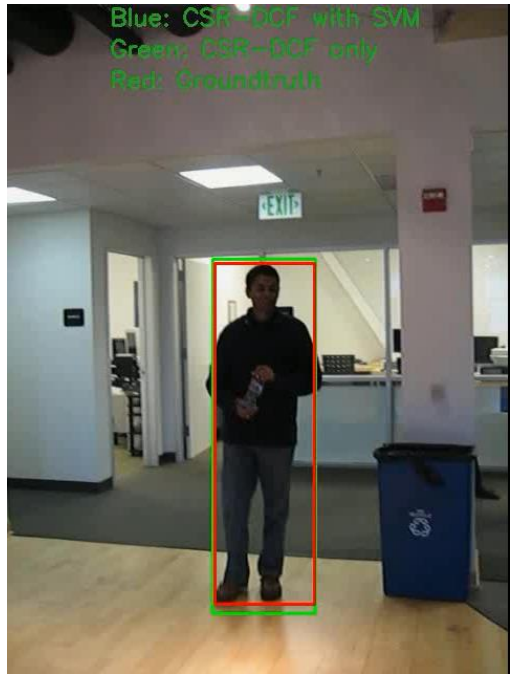
4

PART FOUR

系统实现与测试

视觉行人追踪系统－评估标准

- 重叠率
  - 两个界限框重叠部分的面积除以总面积
  - 衡量结果在尺度和位置上的准确程度
- 偏移率
  - 界限框中心点的偏移量除以其尺度
  - 衡量结果目标位置的准确程度
- 运行速度
  - 每秒能够处理的帧数



GroundTruth bbox1 = (198, 249, 95, 325)

系统输出 bbox2 = (197, 244, 97, 331)

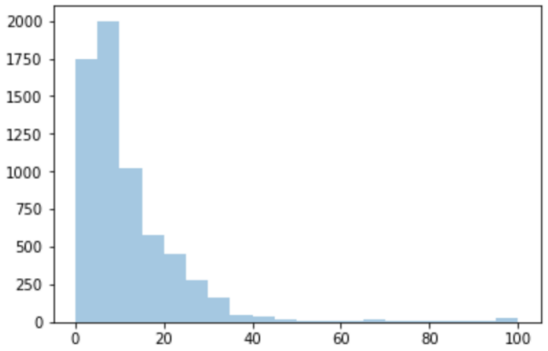
重叠率 = 96.12%

中心点偏移率 = (0.04%, 0.52%)

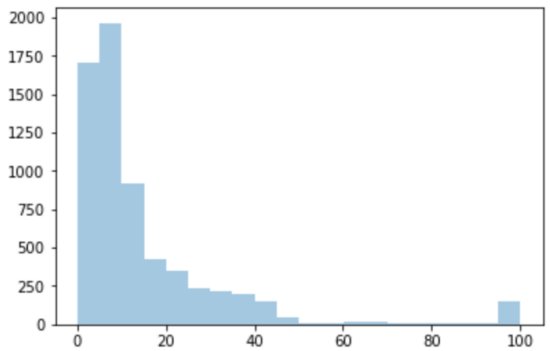
视觉行人追踪系统－测试结果

- 测试数据集：TB-100中的14个视频，共有6448帧。
- 为了验证加入检测器辅助追踪是否提高了结果准确度，另外测试了仅使用CSR-DCF的结果作为对比。

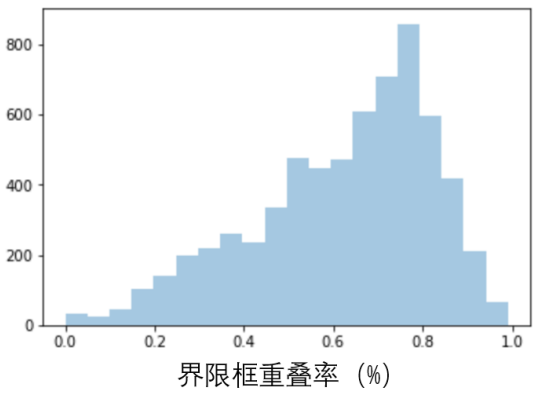
CSR-DCF+SVM方法中心点偏移率直方图



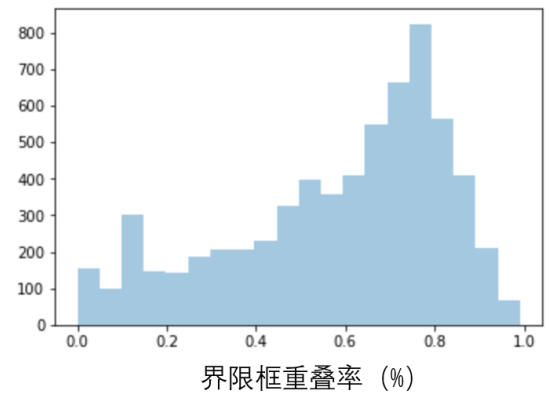
CSR-DCF方法中心点偏移率直方图



CSR-DCF+SVM方法界限框重叠率直方图



CSR-DCF方法界限框重叠率直方图



使用算法类型	CSR-DCF+SVM	CSR-DCF only
总帧数	6448	6448
追踪失败帧数	10	141
平均重叠率	62.16%	58.26%
重叠率小于50%的帧数	1625	2027
平均中心点偏移率	15.81%	27.32%
偏移率大于50%帧数	108	232
平均帧率	16.36 FPS	40.78 FPS

移动服务机器人行人跟随系统

01

系统初始化

使用OpenPose系统来检测人体骨骼。设置一个初始手势（如举起右手），当Open Pose系统检测到该初始手势时，认为该行人即为目标行人，给出其在该帧中的位置，传给追踪器进行初始化。

02

追踪器

使用CSR-DCF算法来进行追踪。利用OpenPose识别出的初始帧和目标位置来初始化追踪器。追踪器在每一帧上运行，是该系统的核心部分。

03

检测器

使用追踪器的结果来训练SVM分类器，使用HOG和色调直方图作为特征。每10帧验证一次追踪器的结果是否正确，如果连续验证失败，则认为追踪器结果错误。当追踪器失败时，使用滑窗法检测全图搜索窗口，判断目标是否在图像中，找到目标在图像中的位置。

04

定位和导航

根据RGB图像中目标的位置，从Kinect深度图像中获得目标在地图中的坐标作为导航目标，调用ROS导航或可佳导航模块，使机器人完成跟随动作。

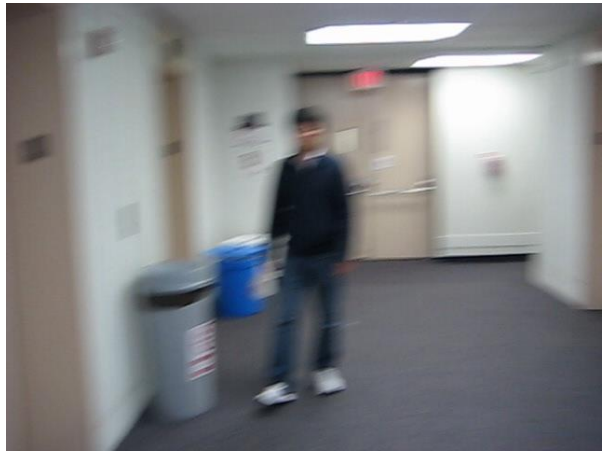
结论和展望

- 本项目设计了一个可以自动初始化的、长时间运行的、较为鲁棒的视觉行人追踪系统。该系统可以通过与深度图像和ROS导航结合，扩展为机器人上的行人跟随系统。
- 该视觉追踪系统以CSR-DCF算法为核心，加入行人检测器作为辅助，在原算法的基础上提高了行人追踪的准确度，降低了追踪失败率，且在大多情况下都可以保证原有的实时运行速度。
- 待改进之处：
  - 在检测器中加入在线学习机制，使用追踪过程中新得到的帧对分类器进行更新，可能会进一步改善分类的准确度
  - 利用深度信息来辅助行人追踪

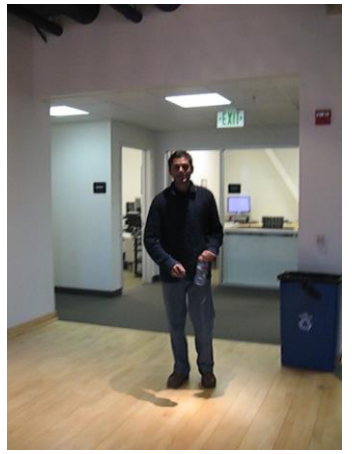
谢 谢 ！

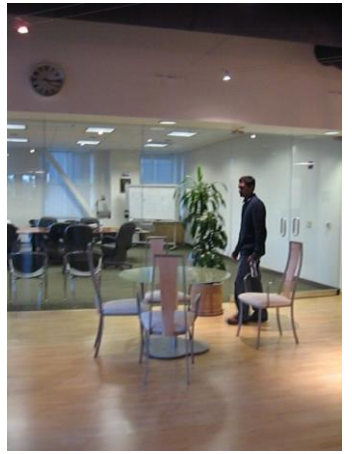
视觉行人检测和追踪中的难点

光照变化



运动模糊



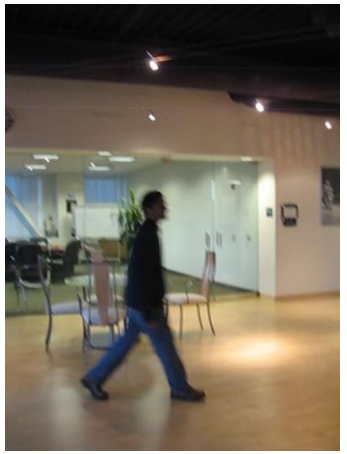


尺度变化

遮挡

形变

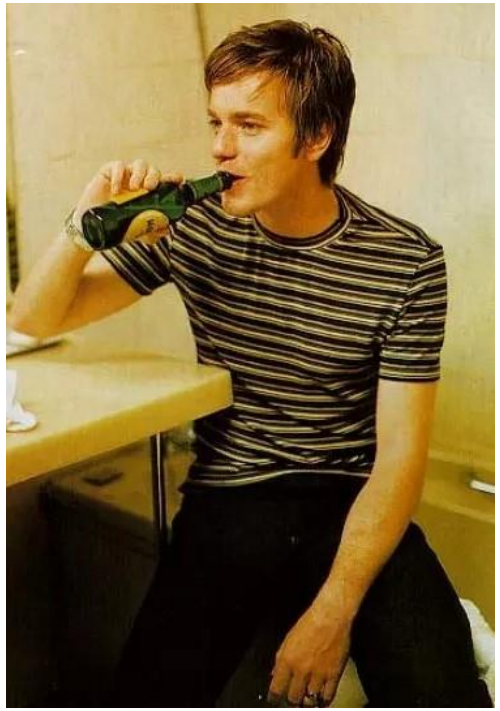
背景杂乱

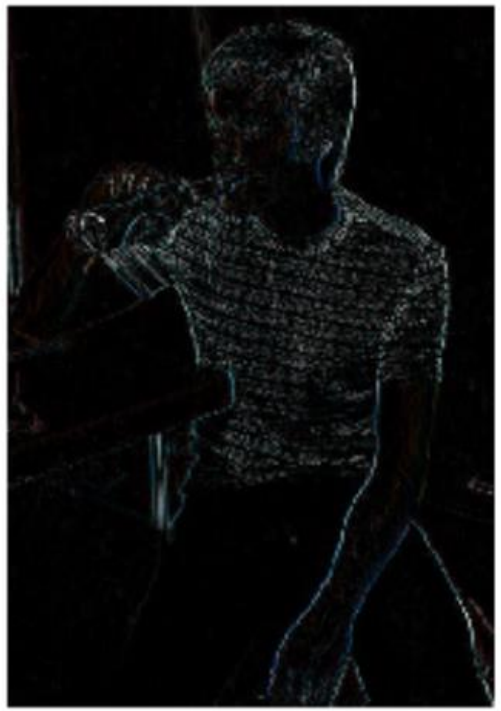


平面外旋转

梯度方向直方图 Histogram of Oriented Gradients, HOG

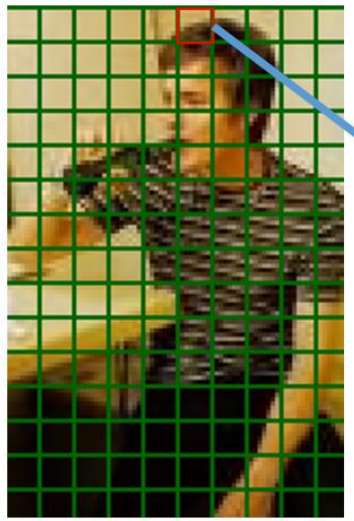
- 通过将图像矩阵与向量$[-1,0,1]$和$[-1,\ 0,\ 1]^T$分别作卷积得到各像素在横向和纵向的梯度，从而得到各点的梯度幅值与方向。
- 图像的梯度可以有效描述图像中各物体的边缘、角落和纹理。

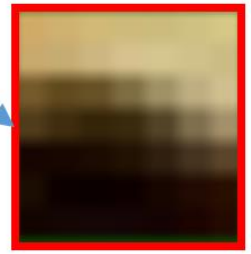




HOG + SVM 行人检测系统

分块





计算梯度

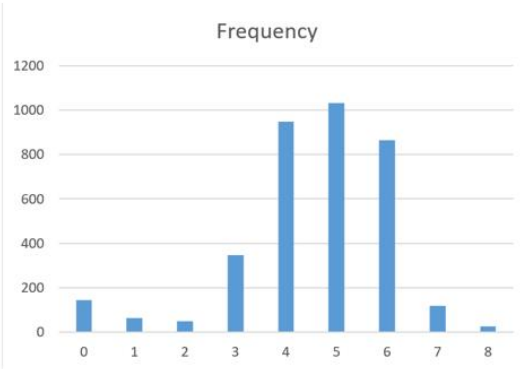
梯度大小

11	8	21	24	8	12	12	7
16	19	19	14	14	18	18	13
67	92	112	109	80	44	22	13
124	129	135	156	168	145	95	48
119	100	78	77	98	134	146	122
57	53	52	43	48	71	108	130
10	33	16	6	18	34	49	78
27	26	24	3	12	11	17	19

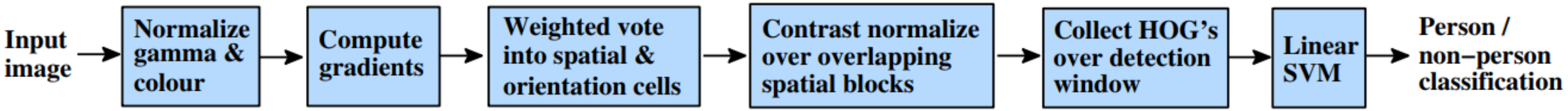
梯度方向

0	0	0	0	0	0	0	0
83	41	35	39	65	66	56	90
82	83	92	99	96	82	60	141
69	70	84	101	110	116	121	115
77	72	70	85	108	121	124	95
60	88	95	103	116	119	111	90
18	5	11	90	133	127	101	130
158	83	83	0	95	112	17	18

统计

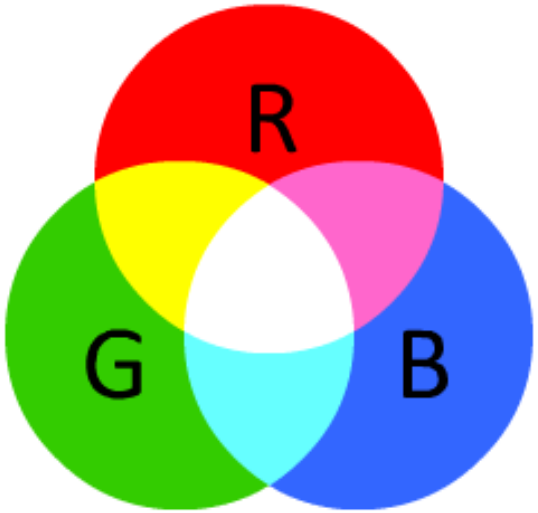


梯度方向直方图

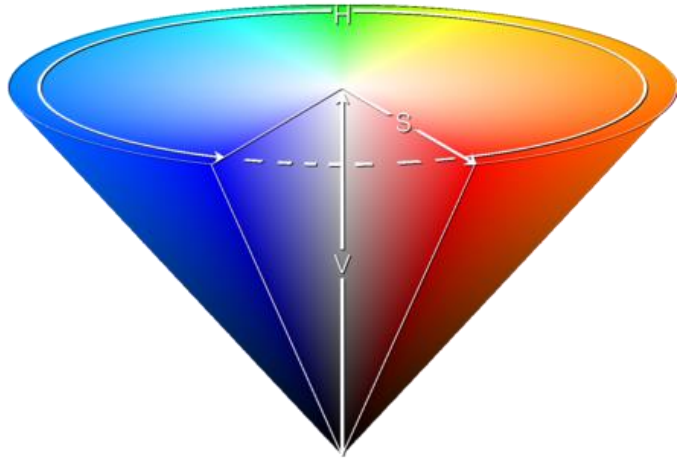


颜色直方图-颜色空间的选择

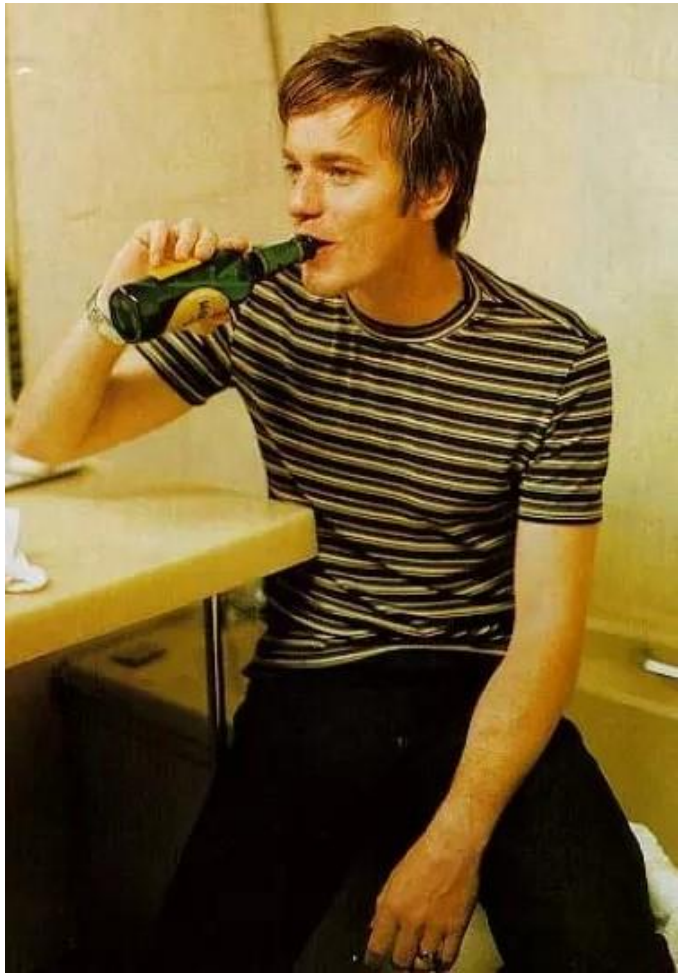
RGB颜色空间

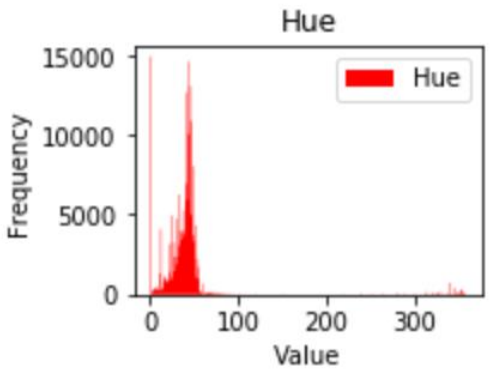


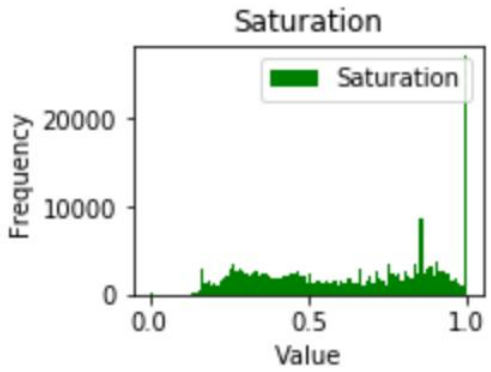
HSV颜色空间

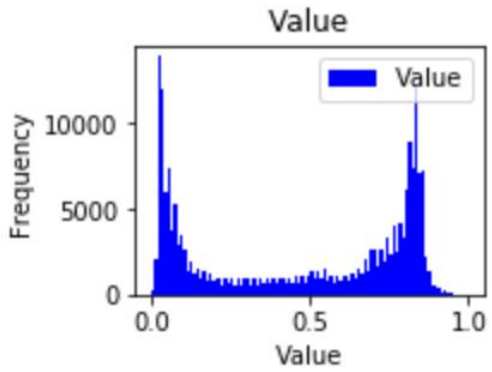


颜色直方图－HSV直方图

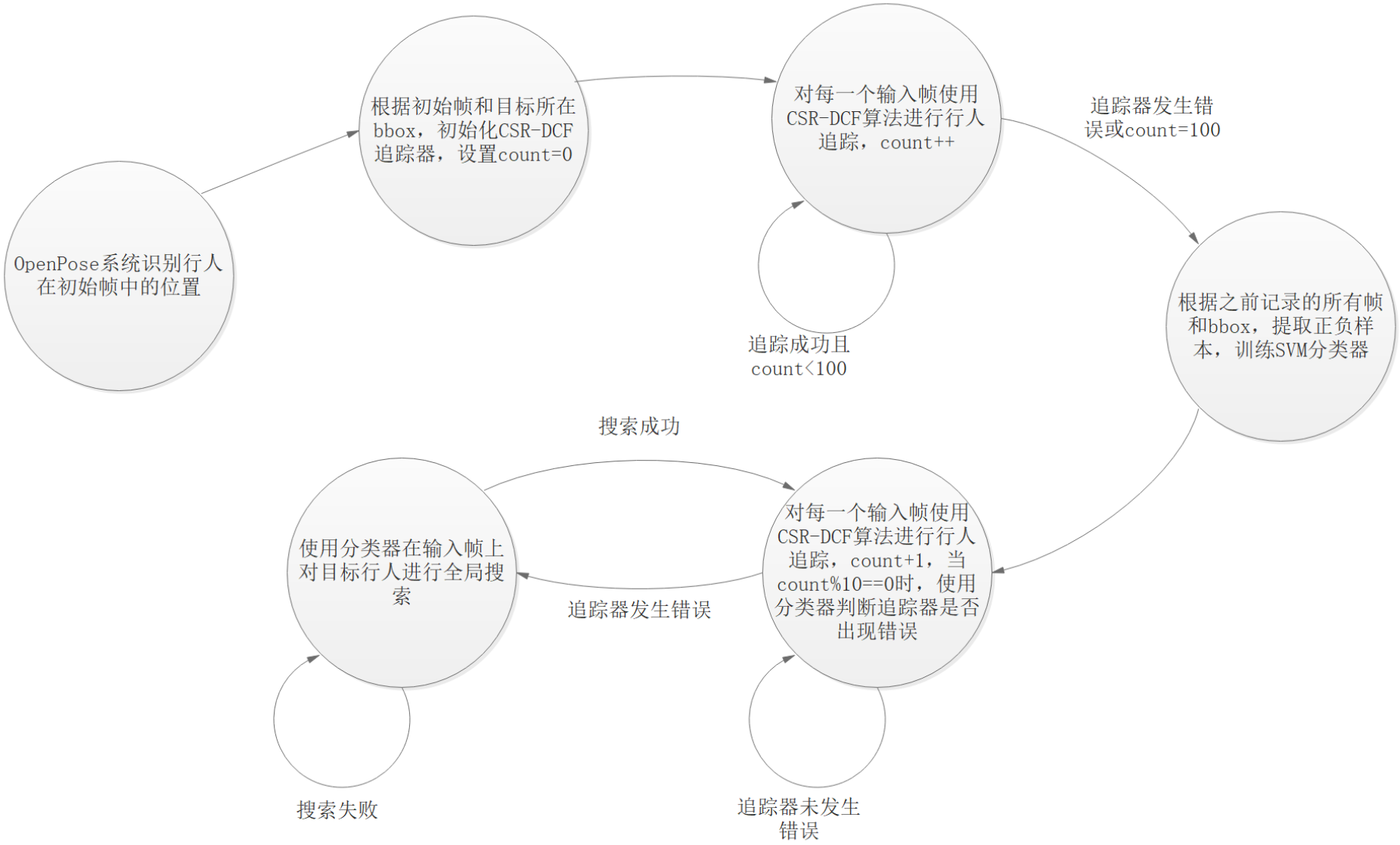








有限状态机


OpenPose系统识别行人在初始帧中的位置
根据初始帧和目标所在bbox，初始化CSR-DCF追踪器，设置count=0
对每一个输入帧使用CSR-DCF算法进行行人追踪，count++
追踪成功且count<100
追踪器发生错误或count=100
根据之前记录的所有帧和bbox，提取正负样本，训练SVM分类器
搜索成功
使用分类器在输入帧上对目标行人进行全局搜索
追踪器发生错误
对每一个输入帧使用CSR-DCF算法进行行人追踪，count+1，当count%10==0时，使用分类器判断追踪器是否出现错误
搜索失败
追踪器未发生错误