



第十三章：目标跟踪

中国科学技术大学
电子工程与信息科学系

主讲教师：李厚强 (lihq@ustc.edu.cn)
周文罡 (zhwg@ustc.edu.cn)
李 礼 (li11@ustc.edu.cn)
胡 洋 (eeychu@ustc.edu.cn)



目标跟踪

- ☐ 单目标跟踪
- ☐ 多目标跟踪
- ☐ 主动目标跟踪



单目标跟踪

- 目标跟踪概述
- 贝叶斯跟踪框架
 - 卡尔曼滤波
 - 粒子滤波
- 均值漂移算法
- 相关滤波器

概述

□ 目标跟踪：

目标跟踪是计算机视觉领域的重要研究方向之一，其目标是在连续的图像中对感兴趣物体进行检测、提取、识别和跟踪，从而获得目标物体的相关参数，如位置、速度、尺度、轨迹等，并对其进行进一步处理和分析，实现对目标物体的行为理解，或完成更高一级的任务。



初始化第1帧目标状态

.....



预测第N帧目标状态

典型应用

□ 视觉跟踪技术典型应用

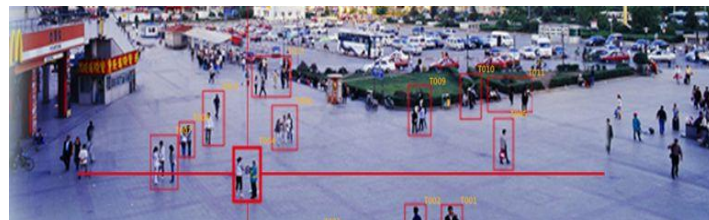
■ 安全监控

- ✓ 车站、机场、银行及超市等公共场所的实时监控。



■ 交通检测

- ✓ 对行人及车辆行为进行判定，完善智能交通系统。



■ 军事领域

- ✓ 导弹制导、武器观测瞄准、敌方目标定位及跟踪。



■ 医学应用

- ✓ 标记、增强及跟踪生物特征来帮助医生诊断疾病。



主要挑战

□ 主要技术挑战

- 形变
- 光照变化
- 快速运动
- 相机抖动
- 运动模糊
- 背景杂乱
- 尺寸变化
- 遮挡
- 超出视野
-



deformation



illumination variation



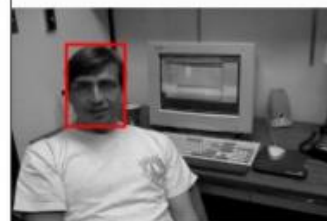
blur & fast motion



background clutter



scale variation



occlusion



out-of-view

视频目标跟踪基本框架

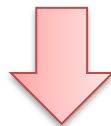


外观模型



视觉目标表征

1. 生成式模型
2. 判别式模型



运动更新模型

跟踪算法通常同时依赖外观建模和运动信息建模。

- 通过利用初始帧的目标外观信息进行建模后，跟踪器具有了目标和背景的辨别能力。
- 在后续帧中，跟踪算法首先通过运动模型，如粒子滤波，粗略地估计目标位置并得到一系列的候选样本，进一步结合外观模型进行目标的精准定位。
- 准确的定位用于更新运动模型和外观模型



生成模型 (Generative Model)

□ 特点:

- 通过前景信息拟合目标的外观表达

□ 缺点:

- 缺少背景信息的利用, 容易漂移

□ 经典算法:

- 均值漂移目标跟踪算法 (MeanShift)
- 基于颜色信息的粒子滤波
- 基于目标局部稀疏表达的粒子滤波
-



判别模型 (Discriminative Model)

□ 特点:

- 通过前景和背景信息去拟合区分前景和背景的二分类器

□ 缺点:

- 过分依赖训练样本及样本标签

□ 经典算法:

- 基于支持向量机 (SVM) 的跟踪算法
- 基于相关滤波器的跟踪方法
- 基于深度学习的跟踪方法
-



目标跟踪

- 目标跟踪概述
- 贝叶斯跟踪框架
 - 卡尔曼滤波
 - 粒子滤波
- 均值漂移算法
- 相关滤波器

贝叶斯跟踪框架

□ 基于贝叶斯估计的跟踪框架

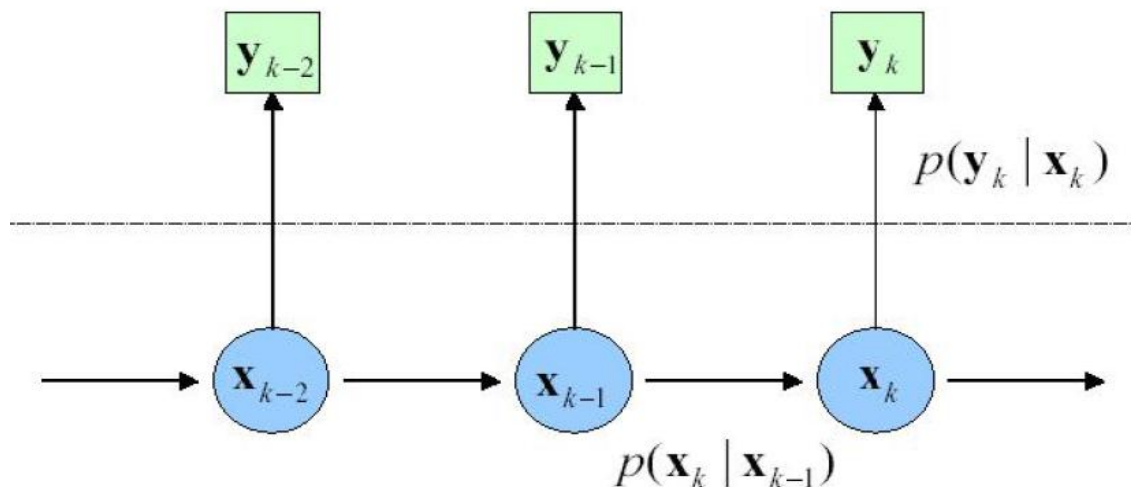
- 状态方程: $x_k = f(x_{k-1}) + u_k$
- 观测方程: $y_k = h(x_k) + v_k$

在目标跟踪中, x 和 y 代表候选样本的状态 (位置、尺度等信息)

x_k 为系统状态, y_k 为观测值, u_k 为过程噪声, v_k 为观测噪声。

□ 目标跟踪需要解决的问题

- 估计后验概率密度 $p(x_k | y_{1:k})$





贝叶斯跟踪算法

- 贝叶斯滤波将状态估计视为概率推理过程，即利用贝叶斯求解后验概率密度

假设已知状态变量的初始概率密度函数： $p(x_0|y_0) = p(x_0)$ ，则估计后验概率密度 $p(x_k|y_{1:k})$ 可以通过预测和更新两步递推得到。

预测：
$$p(x_k|y_{1:k-1}) = \int \boxed{p(x_k|x_{k-1})} p(x_{k-1}|y_{1:k-1}) dx_{k-1}$$

状态转移概率

更新：
$$p(x_k|y_{1:k}) = \frac{\boxed{p(y_k|x_k)} p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})}$$

观测似然度

卡尔曼滤波通过线性、高斯假设条件可得到上述过程解析解。

粒子滤波算法通过数值逼近方法求近似解。



贝叶斯跟踪的通俗理解

- 贝叶斯跟踪滤波的核心：**预测+测量反馈**！
- 一片绿油油的草地上有一条曲折的小径，通向一棵大树，一个要求被提出：蒙着眼从起点沿着小径走到树下。
- “不难，我当过特种兵。” A说，于是他歪歪扭扭地走到了树旁。
“唉，好久不练，生疏了。” （只凭自己的**预测**能力）
- “看我的，我有 DIY 的 GPS！” B说，于是他像个醉汉似地歪歪扭扭的走到了树旁。“唉，这个 GPS 没做好，漂移太大。” （只依靠外界的**测量**）
- “我来试试。” 旁边一也当过特种兵的拿过 GPS，蒙上眼，居然沿着小径很顺滑的走到了树下。（自己能**预测+测量**结果的反馈）
- “这么厉害！你是什么人？”
- “卡尔曼！”
- “卡尔曼？！你就是卡尔曼？” 众人大吃一惊。
- “我是说这个 GPS 卡而慢。”



卡尔曼滤波跟踪算法

□ 卡尔曼滤波基本思想：

- 以**最小均方误差**为最佳估计准则，采用**信号**与**噪声**的状态空间模型，利用前一时刻的**估计值**和当前时刻的**观测值**来更新对**状态变量**的估计，求出当前时刻的估计值
- 卡尔曼滤波融合估计和观测的结果，利用两者的不确定性来得到更加准确的估计

□ 卡尔曼滤波的重要假设：

- 被建模的系统是**线性**的： k 时刻的系统状态可以用某个矩阵与 $(k - 1)$ 时刻的系统状态的乘积表示。
- 影响测量的噪声属于**高斯分布**的**白噪声**，噪声与时间不相关，且只用**均值**和**协方差**就可以准确地建模。

卡尔曼滤波方程

- 假设有一辆直线运动的小车

A diagram of a green open-wheel car with a driver wearing a green helmet. To the left of the car is the state vector $\mathbf{x}_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix}$. To the right of the car is a red arrow pointing right, labeled with the control input $\mathbf{u}_t = \frac{f_t}{m}$.
$$\mathbf{x}_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix}$$
$$\mathbf{u}_t = \frac{f_t}{m}$$

$$\begin{cases} x_t = x_{t-1} + \dot{x}_{t-1} \Delta t + \frac{1}{2} \frac{f_t}{m} \Delta t^2 \\ \dot{x}_t = \dot{x}_{t-1} + \frac{f_t}{m} \Delta t \end{cases}$$

- 矩阵形式

$$\begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ \dot{x}_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix} \frac{f_t}{m}$$



卡尔曼滤波方程

□ 状态方程
$$\mathbf{x}_t = \mathbf{A} * \mathbf{x}_{t-1} + \mathbf{B} * \mathbf{u}_t + \boldsymbol{\omega}_{t-1}$$

□ 观测方程
$$\mathbf{z}_t = \mathbf{H} * \mathbf{x}_t + \boldsymbol{\nu}_t$$

□ 状态预测方程

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{A} * \hat{\mathbf{x}}_{t-1} + \mathbf{B} * \mathbf{u}_t$$

□ 状态更新方程

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t \left(\mathbf{z}_t - \mathbf{H} * \hat{\mathbf{x}}_{t|t-1} \right)$$

□ 状态更新方程的主要问题： \mathbf{K}_t 未知

□ 卡尔曼滤波基于最小均方误差准则求解 \mathbf{K}_t

卡尔曼滤波方程

□ 误差

$$\begin{aligned} e_t &= x_t - \hat{x}_t = x_t - \left(\hat{x}_{t|t-1} + K_t (z_t - H * \hat{x}_{t|t-1}) \right) \\ &= x_t - \hat{x}_{t|t-1} - K_t (H * x_t + \nu_t - H * \hat{x}_{t|t-1}) \\ &= (I - K_t * H) * (x_t - \hat{x}_{t|t-1}) - K_t \nu_t \\ &= (I - K_t * H) * e_{t|t-1} - K_t \nu_t \end{aligned}$$

□ 均方误差

$$\begin{aligned} P_t &= E[e_t * e_t^T] \\ &= (I - K_t * H) * E[e_{t|t-1} * e_{t|t-1}^T] * (I - K_t * H)^T - K_t * E[\nu_t * \nu_t^T] * K_t^T \\ &= (I - K_t * H) * P_{t|t-1} * (I - K_t * H)^T - K_t * R_t * K_t^T \\ &= P_{t|t-1} - K_t * H * P_{t|t-1} - P_{t|t-1} * H^T * K_t^T + K_t (H * P_{t|t-1} * H^T + R_t) K_t^T \end{aligned}$$



卡尔曼滤波方程

□ 最小化均方误差

$$\frac{\partial P_t}{\partial K_t} = -2 * P_{t|t-1} * H^T + 2 * K_t * (H * P_{t|t-1} * H^T + R_t) = 0$$

$$K_t = P_{t|t-1} * H^T * (H * P_{t|t-1} * H^T + R_t)^{-1}$$

$$P_{t|t-1} = E \left[(x_t - \hat{x}_{t|t-1}) (x_t - \hat{x}_{t|t-1})^T \right]$$

$$= E \left[(A * x_{t-1} + B * u_t + \omega_{t-1} - (A * \hat{x}_{t-1} + B * u_t)) (A * x_{t-1} + B * u_t + \omega_{t-1} - (A * \hat{x}_{t-1} + B * u_t))^T \right]$$

$$= E \left[(A * (x_{t-1} - \hat{x}_{t-1}) + \omega_{t-1}) (A * (x_{t-1} - \hat{x}_{t-1}) + \omega_{t-1})^T \right]$$

$$= A P_{t-1} A^T + Q$$

$$P_t = P_{t|t-1} - K_t * H * P_{t|t-1} - P_{t|t-1} * H^T * K_t^T + K_t (H * P_{t|t-1} * H^T + R_t) K_t^T$$

$$= (I - K_t * H) * P_{t|t-1}$$

卡尔曼滤波跟踪算法

□ 卡尔曼滤波递推过程

(1) 建立空间模型（线性假设）：

$$\begin{aligned} x_{k+1} &= \Phi_k x_k + u_k \\ y_{k+1} &= H_{k+1} x_{k+1} + v_{k+1} \end{aligned}$$

带噪声 u, v 的
状态方程
观测方程

(2) 设置初始化条件， $k=0$ 时：

$$\begin{aligned} \hat{x}_0 &= E\{x_0\} \\ C_0 &= Var\{x_0\} \end{aligned}$$

(3) 预测： $\hat{x}_{k+1}^- = \Phi_k \hat{x}_k$

(4) 计算预测误差协方差： $C_{k+1}^- = \Phi_k C_k \Phi_k^T + Q_k$

(5) 计算卡尔曼增益： $K_{k+1} = C_{k+1}^- H_{k+1}^T (H_{k+1} C_{k+1}^- H_{k+1}^T + R_{k+1})^{-1}$



卡尔曼滤波跟踪算法

□ 卡尔曼滤波递推过程

(6) 更新: $\hat{x}_{k+1} = \hat{x}_{k+1}^- + K_{k+1}(y_{k+1} - H_{k+1}\hat{x}_{k+1}^-)$

(7) 计算估计误差协方差: $C_{k+1} = (I - K_{k+1}H_{k+1})C_{k+1}^-$

(8) 计算下一帧, 令 $k = k + 1$, 重复预测和更新过程。

□ 回顾卡尔曼递推的前提假设:

- 观测噪声和模型噪声均服从高斯分布;
- 状态转移函数和观测函数都是线性的;
- 处理过程中各项均是时不变的;

以上假设严重限制了卡尔曼滤波跟踪在实际应用中的效果。



粒子滤波跟踪算法

□ 什么是粒子滤波？

- 粒子滤波 (PF: Particle Filter)的思想基于蒙特卡洛方法 (Monte Carlo methods)，利用粒子集来表示概率，可以用在任何形式的状态空间模型上。
- 其核心思想是通过从后验概率中抽取的随机状态粒子来表达其分布，是一种顺序重要性采样法 (Sequential Importance Sampling)。
- 粒子滤波的实质是根据一定规则(采样函数)采样一些随机粒子(样本)，观测粒子的相似度(似然)来确定粒子的权重，并利用粒子和权值来近似地表示后验概率。
- 可适用于非线性、非高斯系统，克服了卡尔曼滤波的缺点
- 弱点：粒子滤波的计算复杂度高
 - ✓ 随着计算机处理能力的不断增强，早期限制粒子滤波应用的硬件运算能力等障碍正逐渐消失。

粒子滤波跟踪算法

□ 粒子滤波基本思想

假设能从状态的后验概率分布 $p(x_{0:k}|y_{1:k})$ 中独立获取 N 个样本 $\{x_{0:k}^{(i)}\}_{i=1}^N$ ，则状态的后验概率分布可以通过如下离散近似表达式逼近，即使用 N 个样本通过蒙特卡洛的方法近似得到后验概率密度：

$$\hat{p}(x_{0:k}|y_{1:k}) = \frac{1}{N} \sum_{i=1}^N \delta(x_{0:k} - x_{0:k}^{(i)})$$

后验概率未知，怎么办？

通过引入已知的、容易采样的概率密度分布 $q(x_{0:k}|y_{1:k})$ 并从其中采样粒子。此时状态的后验概率可通过加权和的形式逼近：

在重要性概率密度函数中采样粒子，将采样的结果加权逼近后验概率

$$\hat{p}(x_{0:k}|y_{1:k}) = \sum_{i=1}^N w_k^{(i)} \delta(x_{0:k} - x_{0:k}^{(i)})$$

$$w_k^{(i)} = \tilde{w}_k^{(i)} / \sum_{j=1}^N \tilde{w}_k^{(j)} \quad \text{其中} \quad \tilde{w}_k^{(i)} = \frac{p(y_{1:k}|x_{0:k}^{(i)})p(x_{0:k}^{(i)})}{q(x_{0:k}^{(i)}|y_{1:k})}$$

观测似然度

粒子滤波跟踪算法

□ 权重推导过程

$$\begin{aligned}w_k^i &= \frac{p\left(x_{0:k}^{(i)} \middle| y_{1:k}\right)}{N \cdot q\left(x_{0:k}^{(i)} \middle| y_{1:k}\right)} = \frac{p\left(y_{1:k} \middle| x_{0:k}^{(i)}\right) \cdot p\left(x_{0:k}^{(i)}\right)}{N \cdot q\left(x_{0:k}^{(i)} \middle| y_{1:k}\right) \cdot p\left(y_{1:k}\right)} = \frac{\tilde{w}_k^{(i)}}{N \cdot p\left(y_{1:k}\right)} \\&= \frac{\tilde{w}_k^{(i)}}{N \cdot \int p\left(y_{1:k} \middle| x_{0:k}^{(i)}\right) \cdot p\left(x_{0:k}^{(i)}\right) dx_{0:k}^{(i)}} \\&= \frac{\tilde{w}_k^{(i)}}{N \cdot \int \frac{p\left(y_{1:k} \middle| x_{0:k}^{(i)}\right) \cdot p\left(x_{0:k}^{(i)}\right)}{q\left(x_{0:k}^{(i)} \middle| y_{1:k}\right)} \cdot q\left(x_{0:k}^{(i)} \middle| y_{1:k}\right) dx_{0:k}^{(i)}} = \frac{\tilde{w}_k^{(i)}}{\sum_{i=1}^N \tilde{w}_k^{(i)}}\end{aligned}$$

粒子滤波跟踪算法

□ 序贯重要性采样

$$\begin{aligned}\tilde{w}_k^i &= \frac{p(y_{1:k} | x_{0:k}^{(i)}) \cdot p(x_{0:k}^{(i)})}{q(x_{0:k}^{(i)} | y_{1:k})} = \frac{p(y_{1:k-1}, y_k | x_{0:k}^{(i)}) \cdot p(x_{0:k-1}^{(i)}, x_k^{(i)})}{q(x_{0:k-1}^{(i)}, x_k^{(i)} | y_{1:k})} \\&= \frac{p(y_{1:k-1} | x_{0:k}^{(i)}) \cdot p(y_k | y_{1:k-1}, x_{0:k}^{(i)}) \cdot p(x_{0:k-1}^{(i)}) \cdot p(x_k^{(i)} | x_{0:k-1}^{(i)})}{q(x_{0:k-1}^{(i)} | y_{1:k}) \cdot q(x_k^{(i)} | x_{0:k-1}^{(i)}, y_{1:k})} \\&= \frac{p(y_{1:k-1} | x_{0:k-1}^{(i)}) \cdot p(y_k | x_k^{(i)}) \cdot p(x_{0:k-1}^{(i)}) \cdot p(x_k^{(i)} | x_{k-1}^{(i)})}{q(x_{0:k-1}^{(i)} | y_{1:k-1}) \cdot q(x_k^{(i)} | x_{0:k-1}^{(i)}, y_{1:k})} \\&= \tilde{w}_{k-1}^i \cdot \frac{p(y_k | x_k^{(i)}) \cdot p(x_k^{(i)} | x_{k-1}^{(i)})}{q(x_k^{(i)} | x_{0:k-1}^{(i)}, y_{1:k})}\end{aligned}$$

粒子滤波跟踪算法

□ 粒子滤波基本思想

假设系统符合马尔可夫过程，观测变量相互独立，权值递推公式如下：

$$\tilde{w}_k^{(i)} = \tilde{w}_{k-1}^{(i)} \frac{p(y_k | x_k^{(i)}) p(x_k^{(i)} | x_{k-1}^{(i)})}{q(x_k^{(i)} | x_{0:k-1}^{(i)}, y_{1:k})}$$

观测似然度

状态转移概率

随着时间增加，可能发生粒子权值的退化现象，即某一权值趋于1而其余的权值趋于0，权值的退化减少了可用有效样本数度量，在适当情况下重采样粒子。

$$N_{\text{eff}} = \left(\sum_{i=1}^N (w_k^{(i)})^2 \right)^{-1}$$



粒子滤波跟踪算法

- 粒子滤波的实质是大数定理，取足够多的样本就可以使样本均值以概率1趋于数学期望。
 - 在实际应用中，为了获得对后验分布更高的逼近，需要适当增加粒子的个数以防止粒子退化
- 重采样算法是降低粒子匮乏的一种方法。主要思想是将权值低的粒子用权值高的粒子取代，使得计算量不被浪费在无用粒子上
- 粒子滤波重采样主要有
 - 多项式重采样法
 - 分层重采样算法
 - 残差重采样算法
 - 重要性重采样

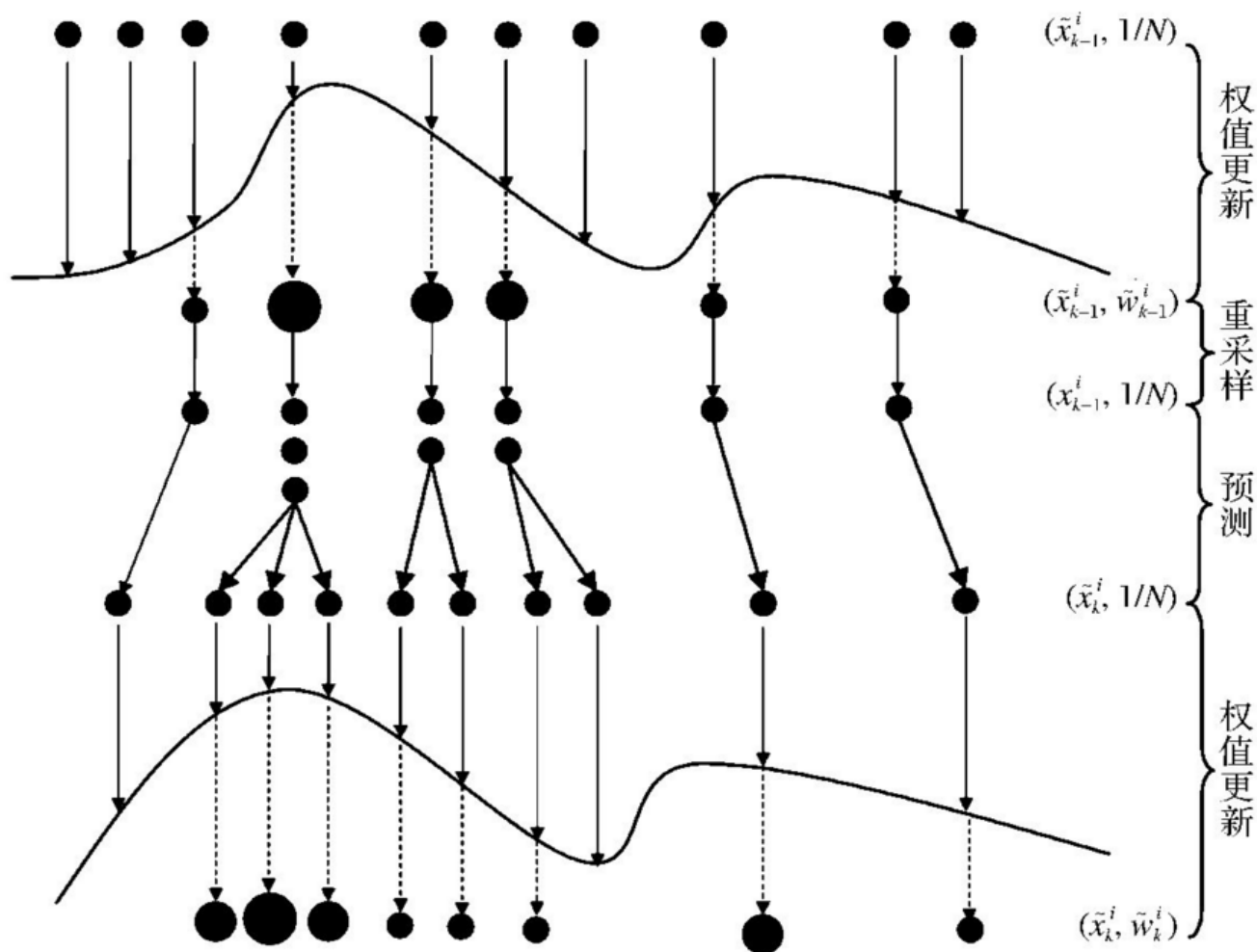


粒子滤波跟踪算法

□ 粒子滤波跟踪时的特征选择

- 对目标状态的传播进行“假设”之后，需要利用得到的 t 时刻的 **观测量** 对其进行验证。因此“系统状态转移-系统观测”可以理解成“**假设-验证**”。
- 所谓 **观测量**，最直观的是指视频图像，可以使灰度图像也可以是处理后的特征量，如颜色、轮廓特征等。
- 使用观测量对系统状态转移的结果进行 **验证**，实际上是一个 **相似性度量** 的过程。
- 由于每个粒子代表目标状态的一个可能性，则系统观测的目的就是使实际情况相近的粒子获得的权值大一些，与实际情况相差较大的粒子获得的权重小一些。

粒子滤波跟踪算法



粒子滤波跟踪算法

□ 粒子滤波流程

(1) 初始化:

在目标跟踪中，每个粒子代表一个图像块，它们的状态 x 代表该图像块的位置、尺寸等信息

确定粒子数目，选择运动模型，确定目标初始位置。

(2) 序贯重要性采样:

实际中，通常认为相邻帧目标运动是平滑的。当前帧中，我们在上一帧目标位置处按照一定模型采样（如高斯模型、均匀采样等）

根据 $x_k^{(i)} \sim q(x_k^{(i)} | x_{0:k-1}^{(i)}, y_{1:k})$ 采样 N 个新粒子 $\{x_k^{(i)}\}_{i=1}^N$ 。

计算粒子权值: $\tilde{w}_k^{(i)} = \tilde{w}_{k-1}^{(i)} \frac{p(y_k | x_k^{(i)}) p(x_k^{(i)} | x_{k-1}^{(i)})}{q(x_k^{(i)} | x_{0:k-1}^{(i)}, y_{1:k})}$

归一化粒子权值: $w_k^{(i)} = \tilde{w}_k^{(i)} / \sum_{j=1}^N \tilde{w}_k^{(j)}$

实际中，粒子权重主要依赖外观模型衡量观测似然度。观测模型估计该粒子和第一帧目标的相似度。观测模型可采用直方图相似性等方案

粒子滤波跟踪算法

□ 粒子滤波流程

(3) 重采样：

计算有效粒子个数: $N_{\text{eff}} = \left(\sum_{i=1}^N \left(w_k^{(i)} \right)^2 \right)^{-1}$

如果 $N_{\text{eff}} < N_{\text{th}}$ ，则增加有效粒子个数，删减无效粒子。

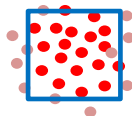
(4) 状态估计: $\hat{x}_k = \sum_{i=1}^N w_k^{(i)} x_k^{(i)}$

根据每个粒子 (图像块) 的重要程度 (权重)，将他们的状态 (图像块位置，尺寸) 进行加权求和。

粒子滤波跟踪算法

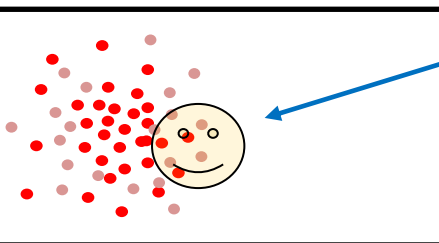
□ 粒子滤波跟踪结果示例

第 $k-1$ 帧



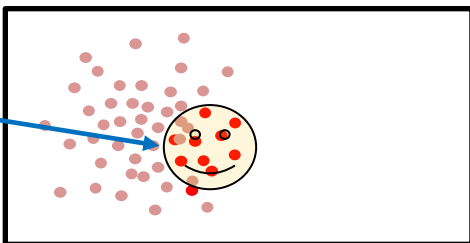
初始帧或上一帧，目标位置及粒子分布情况

第 k 帧

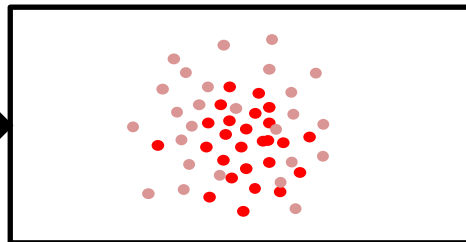


下一帧中，目标发生运动。我们
先在上一帧位置处采样大量粒子。

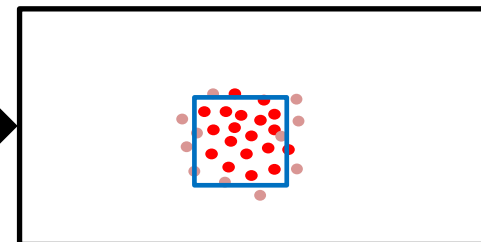
计算粒子权重，
红色粒子权重
更大



粒子重采样



估计目标新的状态





粒子滤波跟踪算法

□ 粒子滤波小结

- 粒子滤波核心思想是利用状态空间的一组带权值的随机样本逼近状态变量的概率密度函数，其显著优点是**多假设性**，不受先验分布及状态转移模型的限制，方便地对目标状态进行表达。
- 该方法需要用大量的样本数量才能很好地近似系统的后验概率密度，算法的复杂度较高。
- 此外，相似背景，遮挡，外观变化等会对粒子滤波的性能产生较大影响。



目标跟踪

- 目标跟踪概述
- 贝叶斯跟踪框架
 - 卡尔曼滤波
 - 粒子滤波
- 均值漂移算法
- 相关滤波器



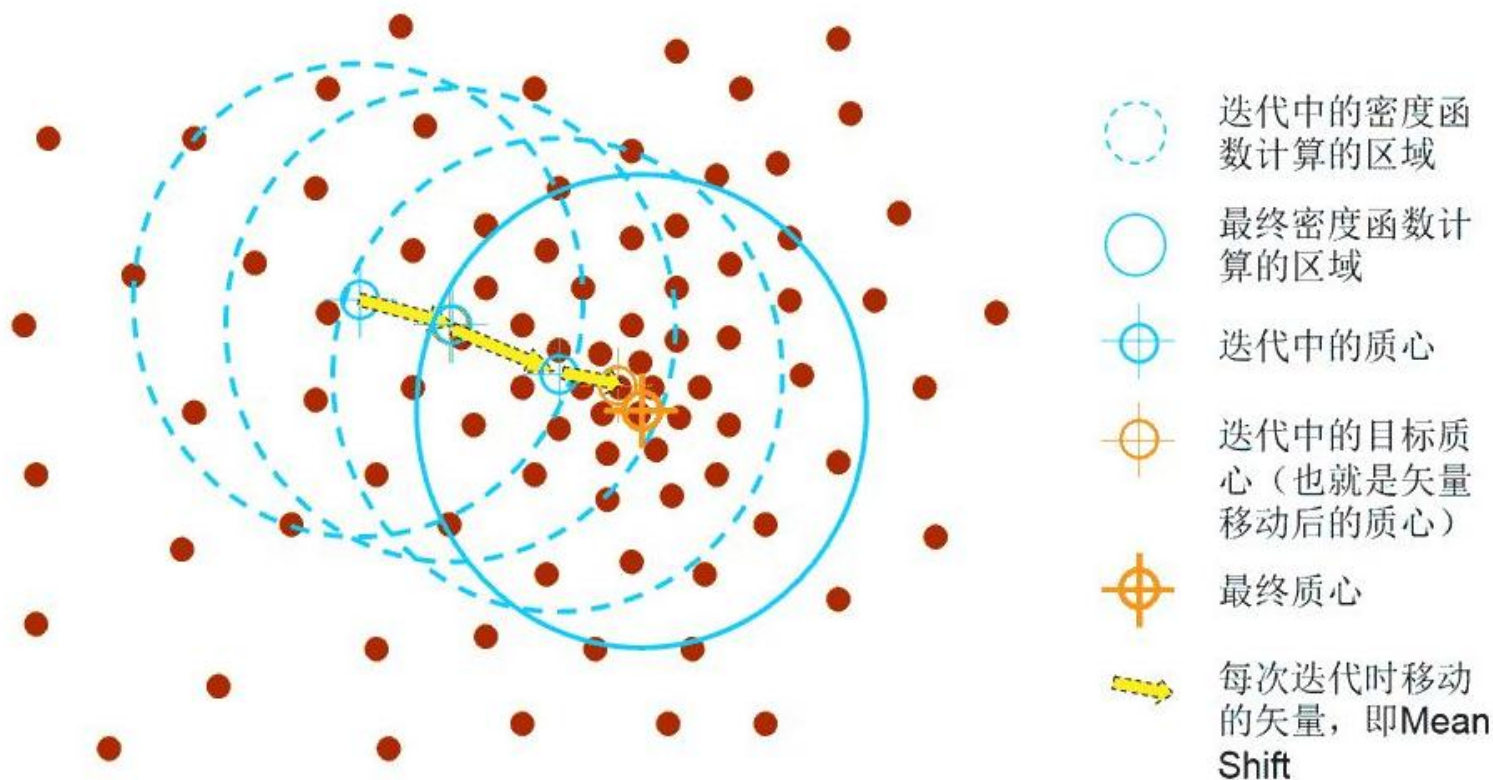
均值漂移算法

□ 均值漂移 (MeanShift) 简介:

- MeanShift 算法是 Fukunaga 于 1975 年提出的。
- 直到 1995 年, Yizong Cheng 针对离 x 越近的采样点对 x 周围的统计特性越有效, 定义了一族核函数。同时他认为所有的样本点重要性不同, 设定了一个权重系数, 扩大了 MeanShift 的使用范围。
- MeanShift 基本思想: 利用概率密度的梯度爬升来寻找局部最优。

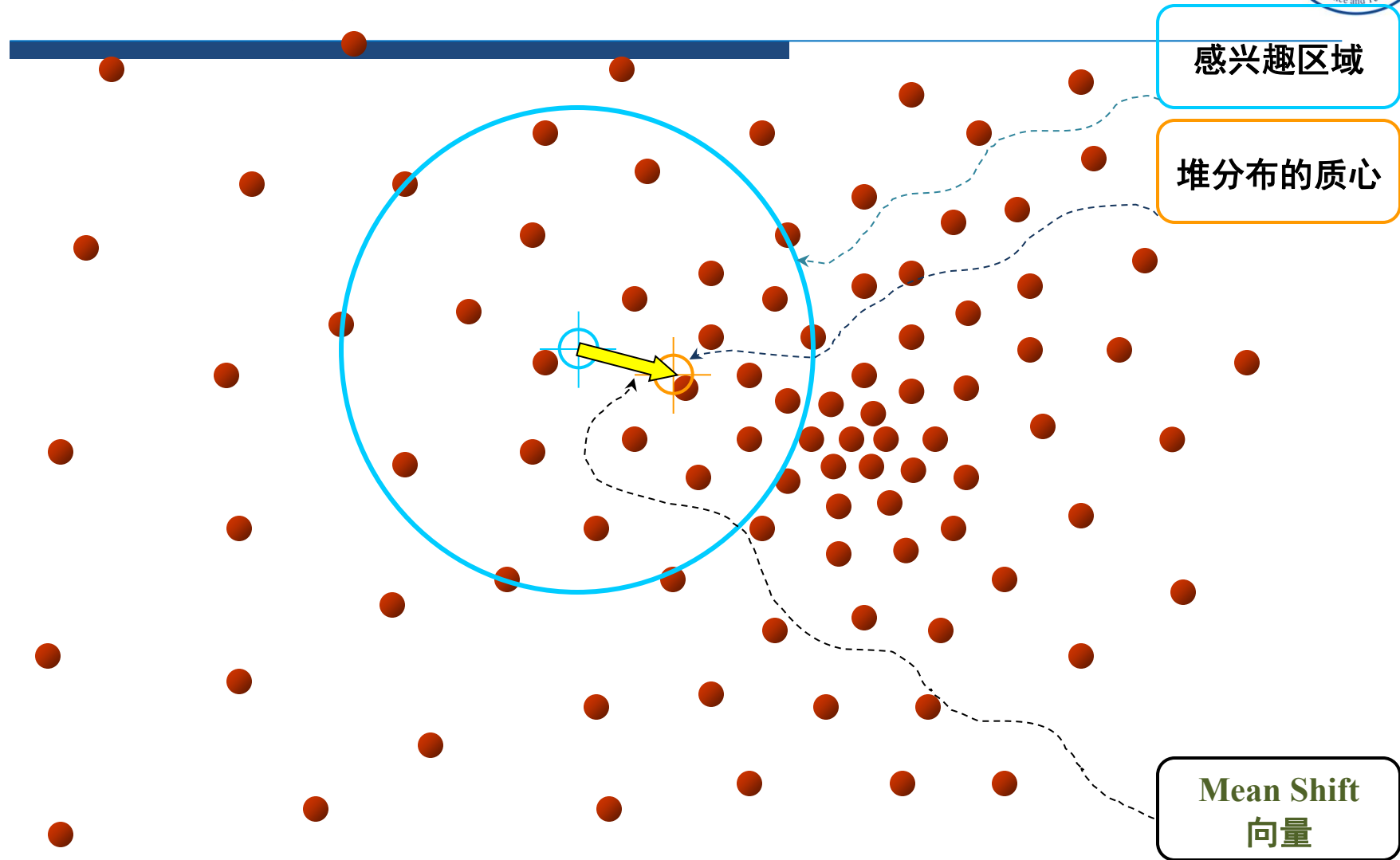
均值漂移算法过程

- MeanShift 向量逐步漂移到局部密度最大点并停止，达到跟踪目的：



目的：找出最密集的区域

直观图示



目标：找到密度最大的区域
桌面上相同的弹球的分布

直观图示

感兴趣区域

堆分布的质心

Mean Shift
向量

目标：找到密度最大的区域
桌面上相同的弹球的分布

直观图示

感兴趣区域

堆分布的质心

Mean Shift
向量

目标：找到密度最大的区域
桌面上相同的弹球的分布

直观图示

感兴趣区域

堆分布的质心

Mean Shift
向量

目标：找到密度最大的区域
桌面上相同的弹球的分布

直观图示

感兴趣区域

堆分布的质心

Mean Shift
向量

目标：找到密度最大的区域
桌面上相同的弹球的分布

直观图示

感兴趣区域

堆分布的质心

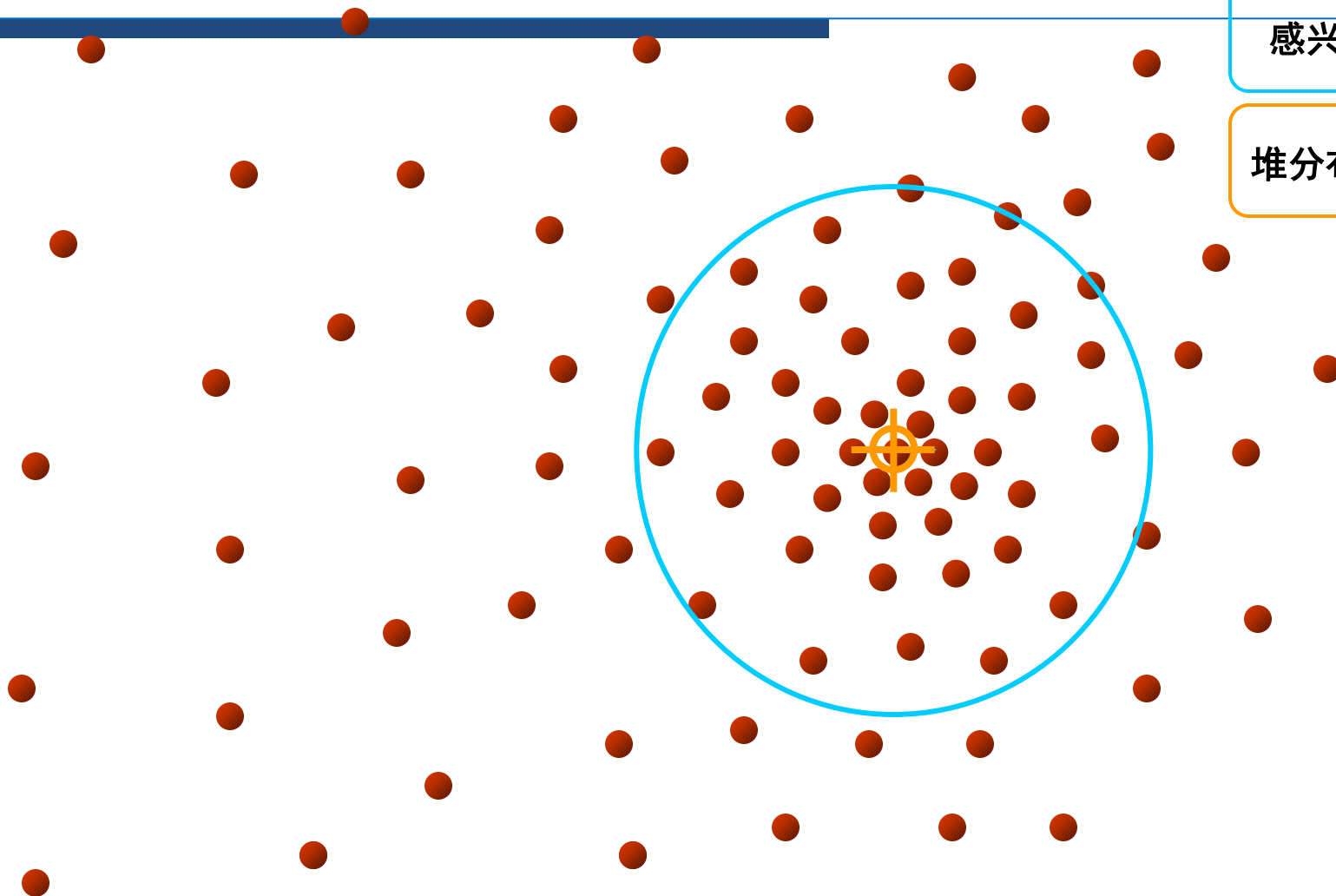
Mean Shift
向量

目标：找到密度最大的区域
桌面上相同的弹球的分布

直观图示

感兴趣区域

堆分布的质心



目标：找到密度最大的区域
桌面上相同的弹球的分布

MeanShift 算法

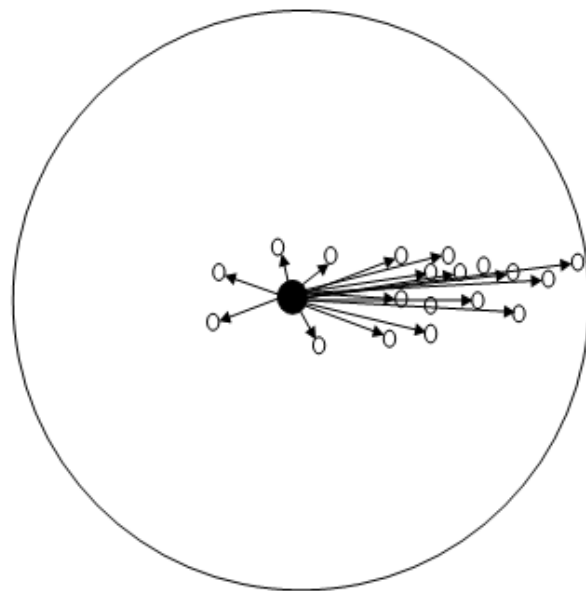
□ 均值漂移 (MeanShift) 原理:

给定 d 维空间中的 n 个样本点 x_i ($i = 1, \dots, n$), 在 x 点的 MeanShift 向量的基本形式定义为:

$$M_h(x) = \left(\frac{1}{k} \sum_{x_i \in S_h} x_i \right) - x = \frac{1}{k} \sum_{x_i \in S_h} (x_i - x)$$

其中, S_h 是一个半径为 h 的高维球区域, k 表示 n 个样本点中有 k 个点落入区域 S_h 中。

直观地, MeanShift 向量表示区域中 k 个样本点相对于点 x 求偏移向量再平均。该向量指向概率密度梯度的方向。



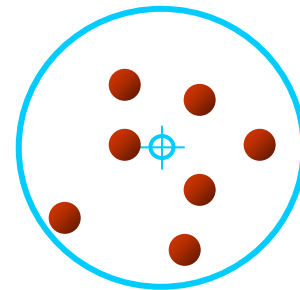
□ 问题: 所有样本点对均值平移向量的贡献相同

扩展的 MeanShift

□ 引入两个参数

- 核函数
- 权重

$$M_h(x) = \frac{1}{k} \sum_{x_i \in S_h} (x_i - x)$$



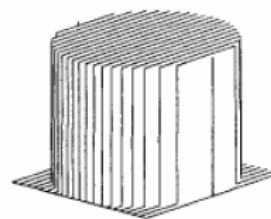
□ 核函数

- 定义: $x \in R^D$, $\|x\|^2 = x^T x$ 。若函数 $K(x)$ 存在一个剖面函数: $k: [0, \infty) \rightarrow R$, 即 $K(x) = k(\|x\|^2)$, 并且 $k(r)$ 满足:

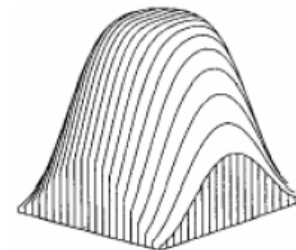
- ✓ 非负的
- ✓ 分段连续的, 且 $\int_0^\infty k(r) dr < \infty$

- 常用核函数

- ✓ 单位均匀核函数
- ✓ 单位高斯核函数



$$K(x) = \begin{cases} 1, & \text{if } \|x\| < 1 \\ 0, & \text{if } \|x\| \geq 1 \end{cases}$$



$$K(x) = e^{-\|x\|^2}$$



扩展的 MeanShift

Mean Shift 扩展形式：

$$m(x) = \frac{\sum_{i=1}^n g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) w(x_i)(x_i - x)}{\sum_{i=1}^n g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) w(x_i)}$$

简化运算：

$$m(x) = \frac{\sum_{i=1}^n x_i g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) w(x_i)}{\sum_{i=1}^n g\left(\left\|\frac{x-x_i}{h}\right\|^2\right) w(x_i)} - x$$

Mean Shift向量的方向和样本点的概率密度函数的梯度方向是一致的



核密度梯度估计

□ 给定 D 维空间 n 个样本点 x_i , $f(x)$ 的核密度估计(也称parzen估计)为

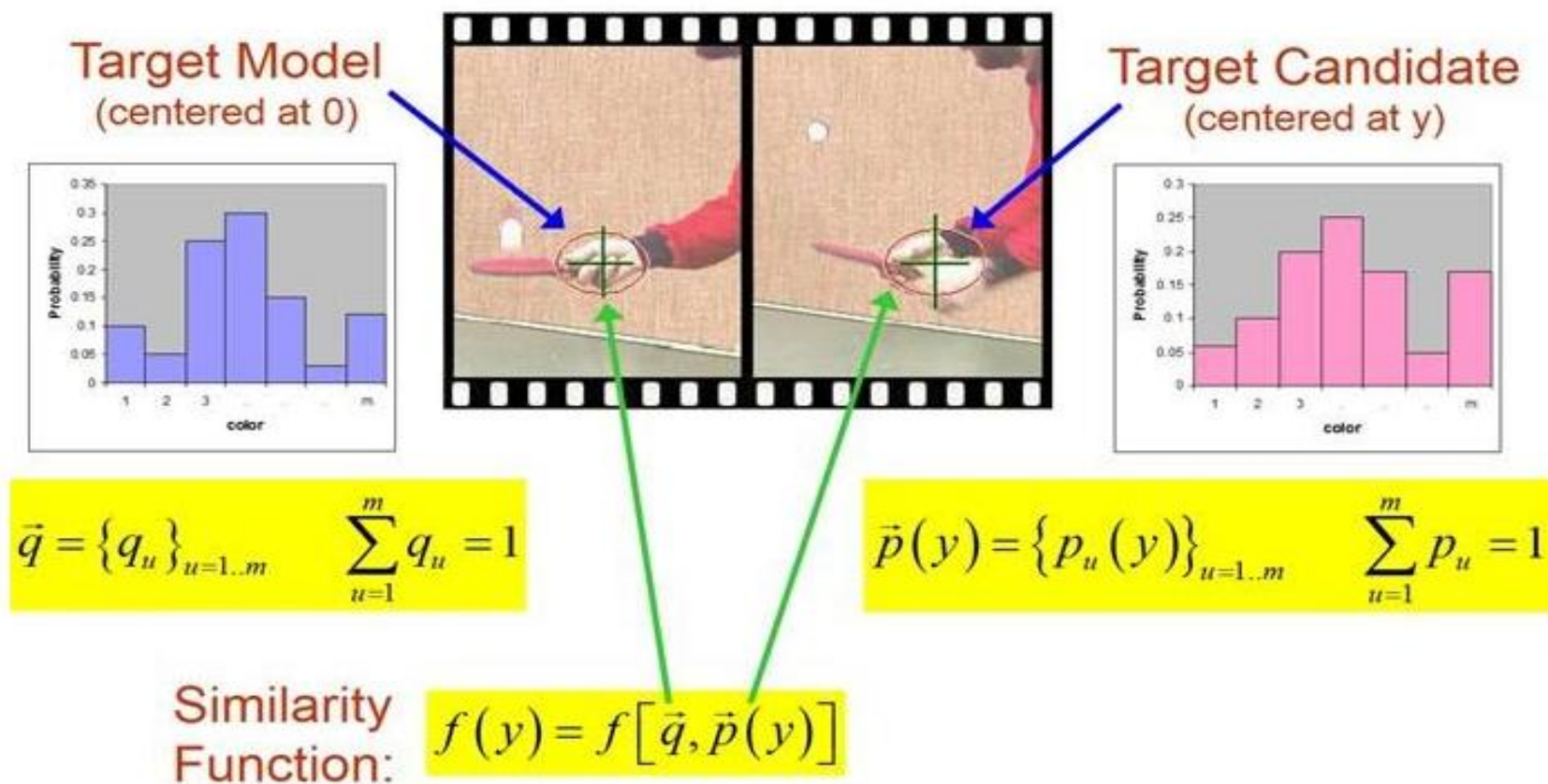
$$\hat{f}(x) = \frac{\sum_{i=1}^n k\left(\left\|\frac{x_i - x}{h}\right\|^2\right) w(x_i)}{h^d \sum_{i=1}^n w(x_i)}$$

□ 概率密度核函数 $f(x)$ 的梯度估计 $\nabla \hat{f}(x)$ 为

$$\begin{aligned}\nabla f(x) &\approx \nabla \hat{f}(x) = \frac{2 \sum_{i=1}^n (x - x_i) k'\left(\left\|\frac{x_i - x}{h}\right\|^2\right) w(x_i)}{h^{2+d} \sum_{i=1}^n w(x_i)} \\ &= \frac{2}{h^2} \left[\frac{\sum_{i=1}^n g\left(\left\|\frac{x_i - x}{h}\right\|^2\right) w(x_i)}{h^d \sum_{i=1}^n w(x_i)} \right] \left[\frac{\sum_{i=1}^n g\left(\left\|\frac{x_i - x}{h}\right\|^2\right) w(x_i) x_i}{\sum_{i=1}^n g\left(\left\|\frac{x_i - x}{h}\right\|^2\right) w(x_i)} - x \right] \\ &\quad \uparrow \\ &\quad g(x) = -k'(x)\end{aligned}$$

均值漂移目标跟踪算法 (TPAMI-03)

□ 均值漂移 (Mean Shift) 跟踪 (生成模型) ^[1]:



- Comaniciu D, Ramesh V, Meer P. Kernel-based object tracking[J]. IEEE Transactions on pattern analysis and machine intelligence, 2003, 25(5): 564-577.

均值漂移目标跟踪算法

核函数：图像
块的中心像素
权重更大

□ 目标核函数直方图

$$\hat{q}_u = C \sum_{i=1}^n k(\|x_i\|^2) \delta[b(x_i) - u]$$

统计直方图

其中 $C = 1 / \sum_{i=1}^n k(\|x_i\|^2)$ 为归一化常数, k 表示核函数(在核估计中通常是平滑作用), 目标区域共 n 个像素点 $\{x_i\}_{i=1, \dots, n}$, 该区域颜色分布离散成 m 级, $b(x_i)$ 表示像素点 x_i 的量化值。

□ 候选（待跟踪目标）核函数直方图

$$\hat{p}_u(y) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{y - x_i}{h}\right\|^2\right) \delta[b(x_i) - u]$$

其中 $C_h = 1 / \sum_{i=1}^{n_h} k\left(\left\|\frac{y - x_i}{h}\right\|^2\right)$, 候选目标区域 $\{x_i\}_{i=1, \dots, n_h}$, 该区域的中心位置为 y , h 表示核函数 k 的窗宽。其余变量物理意义同上。

均值漂移目标跟踪算法

- 通过Bhattacharyya系数度量候选样本 (中心为 y 的图像块) 和目图像标核函数直方图的相似度:

$$\hat{\rho}(y) = \sum_{u=1}^m \sqrt{\hat{p}_u(y) \hat{q}_u}$$

- Taylor 展开, 将 $\hat{p}_u(y)$ 代入并化简得到:

$$\hat{\rho}(y) \approx \frac{1}{2} \sum_{u=1}^m \sqrt{\hat{p}_u(y_0) \hat{q}_u} + \frac{C_h}{2} \sum_{i=1}^{n_h} w_i k \left(\left\| \frac{y - x_i}{h} \right\|^2 \right)$$

$$\text{其中 } w_i = \sum_{u=1}^m \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{y}_0)}} \delta[b(x_i) - u]$$

y_0 是MeanShift迭代的起点位置,
在跟踪中, 通常是目标在上一帧的位置

由于第一项为常量, 因此我们最大化 $\hat{\rho}(y)$, 本质上寻找新的质心 y 使得候选区域和模板的相似程度最大化

均值漂移目标跟踪算法

- MeanShift 方法求解目标的新位置 y_1 :

$$y_1 = \frac{\sum_{i=1}^{n_h} x_i w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)} \quad \text{其中 } g(x) = -k'(x)$$

- 目标跟踪中通常使用 Epanechnikov 核函数
- 在跟踪问题中，核函数起到了位置加权的作用，即更加相信中心位置像素点的信息。在单位均匀核的情况下， $g(x)$ 是常数，上式可以简化为：

$$y_1 = \frac{\sum_{i=1}^{n_h} x_i w_i}{\sum_{i=1}^{n_h} w_i}$$



均值漂移目标跟踪算法

□ MeanShift 实现目标跟踪流程：

1. 在当前帧，计算候选目标的特征 (候选图像块的颜色直方图)

2. 计算候选目标与初始目标的相似度： $\hat{\rho}(y_0) = \sum_{u=1}^m \sqrt{\hat{p}_u(y_0) \hat{q}_u}$

3. 计算权值 $\{w\}_{i=1,2,\dots,m}$

4. 利用 MeanShift 算法，计算目标新位置：

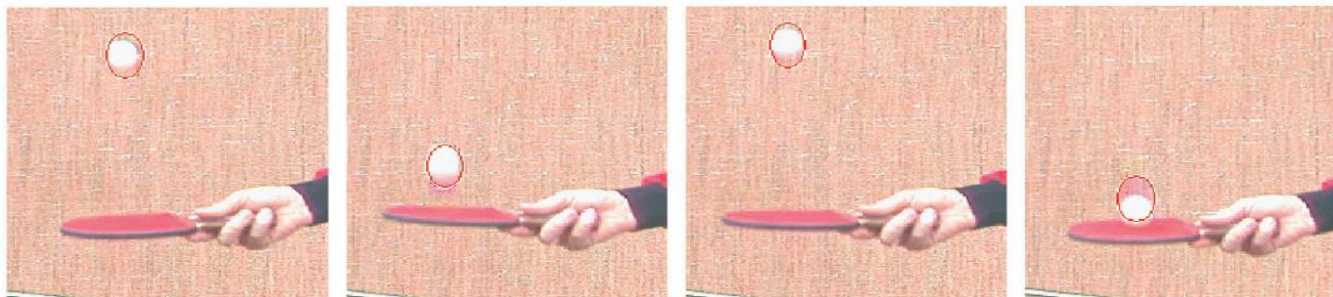
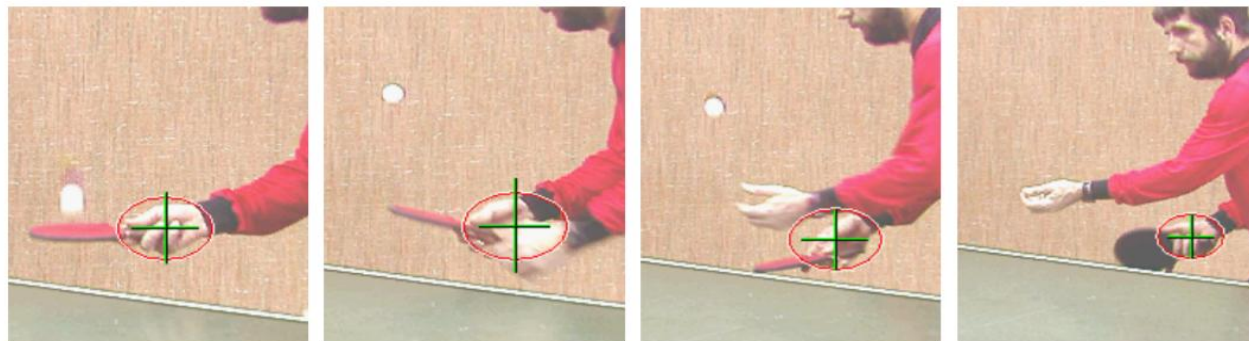
$$y_1 = \frac{\sum_{i=1}^{n_h} x_i w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{y_0 - x_i}{h}\right\|^2\right)}$$

5. 若 $\|y_1 - y_0\| < \varepsilon$ ，则停止，否则 $y_0 \leftarrow y_1$ 转步骤 3

限制条件：新目标中心需位于原目标中心附近

均值漂移目标跟踪算法

□ 实验结果示例：





均值漂移目标跟踪算法

□ MeanShift 跟踪算法小结：

□ 优点：

- 无参密度估计的方法，计算量小，能满足实时性的要求
- 基于核函数直方图的建模对目标的旋转形变鲁棒性好

□ 缺点：

- 仅采用颜色直方图，易受具有相似颜色的背景物体干扰
- 缺乏必要的模板更新过程
- 目标运动速度快或严重遮挡时，跟踪易发生漂移



目标跟踪

- 目标跟踪概述
- 贝叶斯跟踪框架
 - 卡尔曼滤波
 - 粒子滤波
- 均值漂移算法
- 相关滤波器

相关滤波器跟踪算法

- 相关滤波跟踪原理:相关滤波跟踪的基本思想就是, 设计一个滤波模板, 利用该模板与目标候选区域做相关运算, 最大输出响应的位置即为当前帧的目标位置

$$y = x \otimes w$$

$$\hat{y} = \hat{x} \cdot \hat{w}^*$$

- 使用m个训练样本, 基于最小二乘训练滤波器

$$\min \left(\sum_{i=1}^m (\hat{x}_i \cdot \hat{w}^* - y_i)^2 \right)$$

$$\hat{w} = \frac{\sum_i \hat{x}_i \cdot \hat{y}_i^*}{\sum_i \hat{x}_i \cdot \hat{x}_i^*}$$



相关滤波器跟踪算法

- 岭回归（正则化最小二乘）的目标在于学习映射函数 $f(z) = w^T z$ 来最小化如下的均方误差

$$\min_w \sum_i (f(x_i) - y_i)^2 + \lambda \|w\|^2$$

- 岭回归具有如下的闭式解：

$$w = (X^T X + \lambda I)^{-1} X^T y$$

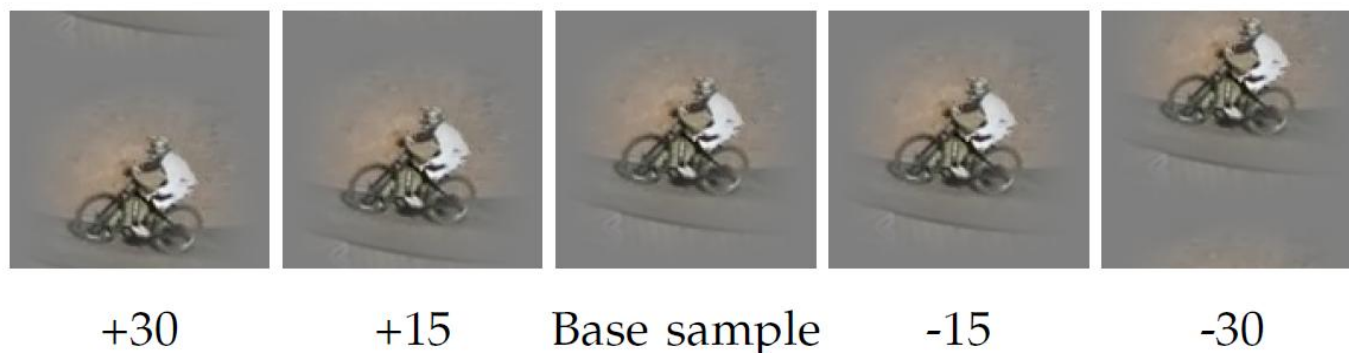
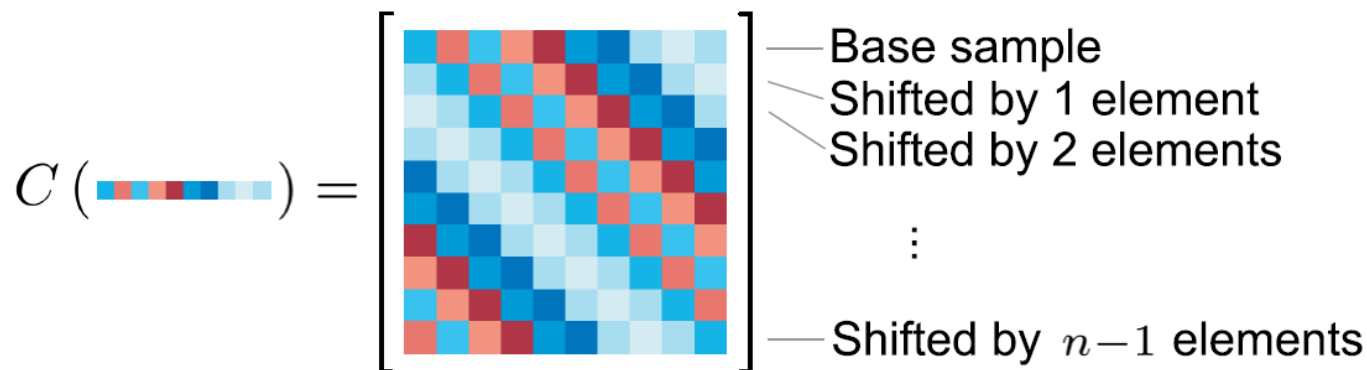
- 其中 X 为样本矩阵，每一行为一个样本 x_i
- 上述闭合解的复杂度在于矩阵求逆。
- 相关滤波器 (Correlation Filter)，源于岭回归，通过频域求解以及循环矩阵的性质巧妙避免上述矩阵求逆过程。

相关滤波器跟踪算法 (TPAMI-15)

□ 相关滤波器 (Correlation Filter, CF) : 判别式跟踪算法

- 通过对感兴趣区域的水平和垂直方向圆周移位, 可以获取大量的训练样本。通过训练这些样本获取滤波器用于跟踪。

□ 训练样本示例:



相关滤波器跟踪算法

□ 通过将单张图片特征进行循环移位，数据矩阵 X 有如下形式：

$$X = C(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix}$$

通过构造上述的数据矩阵，目的在于极大加速岭回归的求解。

$$\mathbf{w} = (X^T X + \lambda I)^{-1} X^T \mathbf{y}.$$



相关滤波器跟踪算法

$$w = (X^T X + \lambda I)^{-1} X^T y$$

我们将上述的闭合解转化成复数域, 如下:

$$w = (X^H X + \lambda I)^{-1} X^H y$$

其中 H 代表共轭转置, $*$ 代表复共轭, 即 $X^H = (X^*)^T$

此外, 由于矩阵 X 是循环的, 可以通过离散傅里叶变换(DFT)对角化:

$$X = F \text{diag}(\hat{x}) F^H$$

其中 $\hat{\cdot}$ 表示傅里叶变换, 即 $\hat{x} = \mathcal{F}(x)$, X 是向量 x 产生的循环矩阵

F 为DFT矩阵



相关滤波器跟踪算法

基于 $w = (X^H X + \lambda I)^{-1} X^H y$ 和 $X = F \text{diag}(\hat{x}) F^H$

有了如上的准备工作，我们可以逐步得到岭回归的频域闭合解：

$$X^H X = F \text{diag}(\hat{x}^*) F^H F \text{diag}(\hat{x}) F^H$$

$$X^H X = F \text{diag}(\hat{x}^*) \text{diag}(\hat{x}) F^H$$

$$X^H X = F \text{diag}(\hat{x}^* \odot \hat{x}) F^H$$

$$w = (F \text{diag}(\hat{x}^* \odot \hat{x}) F^H + \lambda F I F^H)^{-1} X^H y$$

$$= (F \text{diag}(\hat{x}^* \odot \hat{x} + \lambda) F^H)^{-1} X^H y$$

$$= F \text{diag}(\hat{x}^* \odot \hat{x} + \lambda)^{-1} F^H F \text{diag}(\hat{x}^*) F^H y$$

$$= F \text{diag} \left(\frac{\hat{x}^*}{\hat{x}^* \odot \hat{x} + \lambda} \right) F^H y$$

相关滤波器跟踪算法

$$F^H w = \text{diag} \left(\frac{\hat{x}^*}{\hat{x}^* \odot \hat{x} + \lambda} \right) F^H y$$

$$\hat{w} = \text{diag} \left(\frac{\hat{x}^*}{\hat{x}^* \odot \hat{x} + \lambda} \right) \hat{y}$$

由于 $\text{diag}(\cdot)$ 矩阵和向量间的矩阵乘法可替代成元素间乘法 (element-wise product), 上式化简成如下表达:

$$\hat{w} = \frac{\hat{x}^* \odot \hat{y}}{\hat{x}^* \odot \hat{x} + \lambda}$$



完全避免了矩阵求逆,
元素间的乘积极其高效



相关滤波器跟踪算法

□ 检测过程：

- 当相关滤波器学习完成后。给出一个新的数据矩阵 Z ，我们可以通过如下预测其回归值 (response map)：

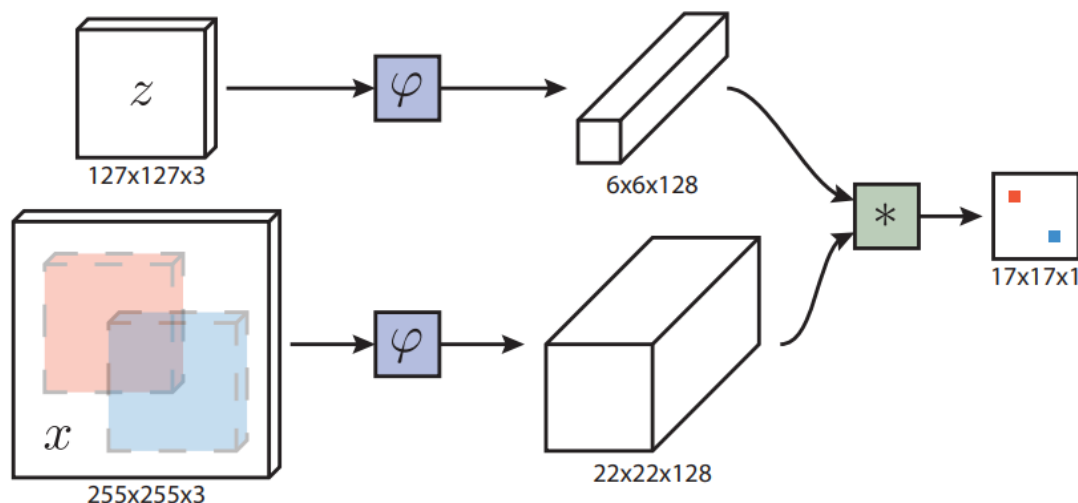
时域 $r(w, Z) = Zw$ 或频域结果 $\hat{r}(w, Z) = \hat{z} \odot \hat{w}$

- **左式：**我们依旧可以在时域进行求解，其中 Z (大写) 是基样本 z 的循环移位矩阵。该矩阵和滤波器 w 进行相乘即可得到搜索样本的预测值。
- **右式：**我们同样可以在频域进行快速求解，从而避免手工地对搜索样本 z 进行循环移位以得到 Z 。注意，右式求得的响应值是频域的，需要进行反变换回到时域。

孪生网络跟踪算法

□ 算法原理：

- 将跟踪任务转化为相似性学习来解决。给定一个模板帧 z 和一个搜索帧 x ，算法通过一个孪生网络学习一个非线性嵌入函数 φ ，然后在嵌入空间中计算相似性。
- 采用全卷积网络实现稠密高效的滑动窗口计算




孪生网络跟踪算法

□ 训练过程：

■ 特征提取 $f_z = \phi(z), \quad f_x = \phi(x)$

■ 相关性计算 $S(z, x) = f_z \star f_x$

■ Loss函数
$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \ell(S(z_i, x_i), y_i)$$

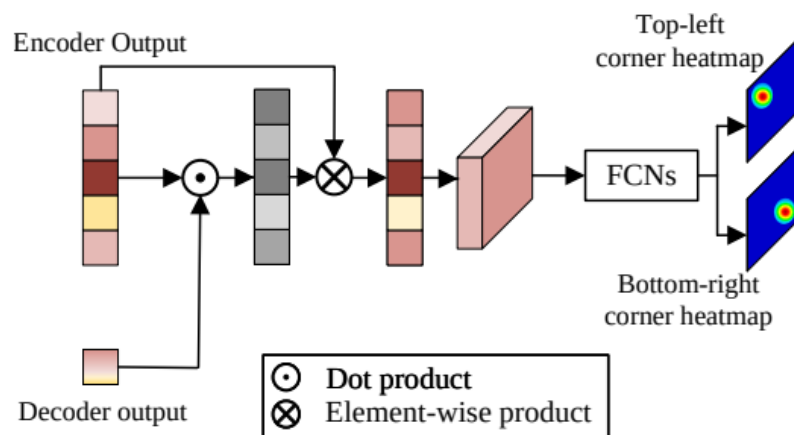
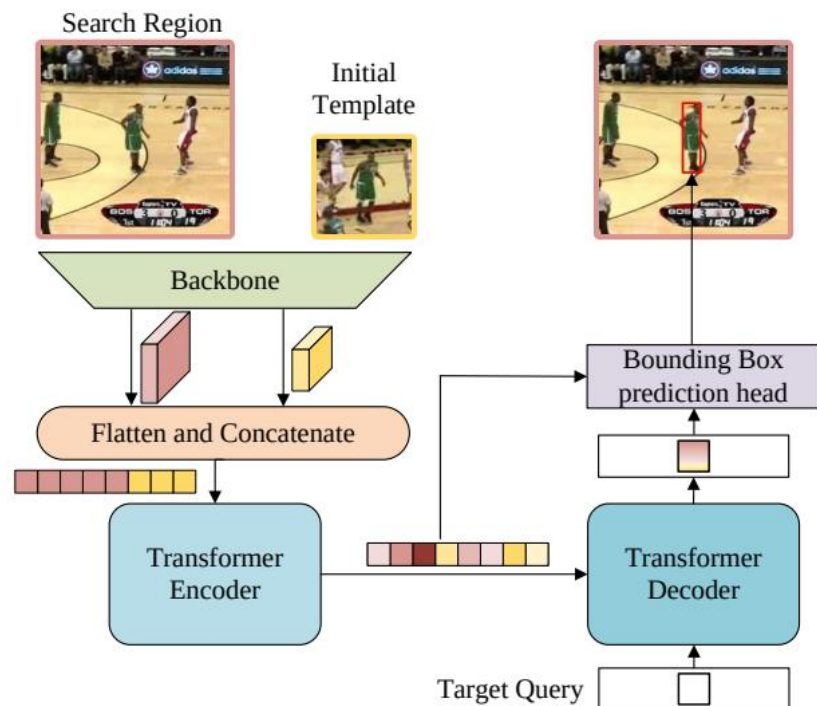


其中 ℓ 为损失函数，如逻辑损失（Logistic Loss）。其中 y_i 是目标的真实标签（目标中心为1，其他位置为-1的得分图）。

时空Transformer跟踪算法

□ 算法原理：

- 编码器对目标物体和搜索区域之间的全局时空特征依赖性进行建模，解码器通过学习查询嵌入预测目标物体空间位置。
- 将目标跟踪作为一个直接的边界框预测问题



Bounding Box prediction head

时空Transformer跟踪算法

□ 训练过程:

- 模型输出为bounding box的左上角和右下角的概率热度图:

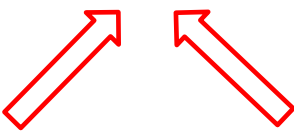
$$P_{tl}(x, y) \quad P_{br}(x, y)$$

- Bounding box获取:

$$\begin{aligned} (\widehat{x}_{tl}, \widehat{y}_{tl}) &= \left(\sum_{y=0}^H \sum_{x=0}^W x \cdot P_{tl}(x, y), \sum_{y=0}^H \sum_{x=0}^W y \cdot P_{tl}(x, y) \right), \\ (\widehat{x}_{br}, \widehat{y}_{br}) &= \left(\sum_{y=0}^H \sum_{x=0}^W x \cdot P_{br}(x, y), \sum_{y=0}^H \sum_{x=0}^W y \cdot P_{br}(x, y) \right), \end{aligned}$$

- 损失函数:

$$L = \lambda_{iou} L_{iou}(b_i, \hat{b}_i) + \lambda_{L_1} L_1(b_i, \hat{b}_i).$$


Groundtruth Prediction



目标跟踪

- ☐ 单目标跟踪
- ☐ 多目标跟踪
- ☐ 主动目标跟踪



多目标跟踪

- 问题定义
- 经典算法
 - 匈牙利算法简介
 - SORT算法
 - DeepSORT算法

多目标跟踪

□ 单目标跟踪 vs 多目标跟踪

- 单目标跟踪的初始目标在**第一帧给定**（通常矩形框表示）
- 单目标跟踪要求可以处理**任意类别物体**
- 多目标跟踪**没有初始标注**，由目标检测算法生成目标框。多目标跟踪算法主要负责每个目标的数字标识（ID）的帧间关联。跟踪过程中，原目标可能消失，新目标可能加入
- 由于依赖目标检测算法，多目标跟踪通常用于**特定物体场景**：如车辆、行人的多目标跟踪





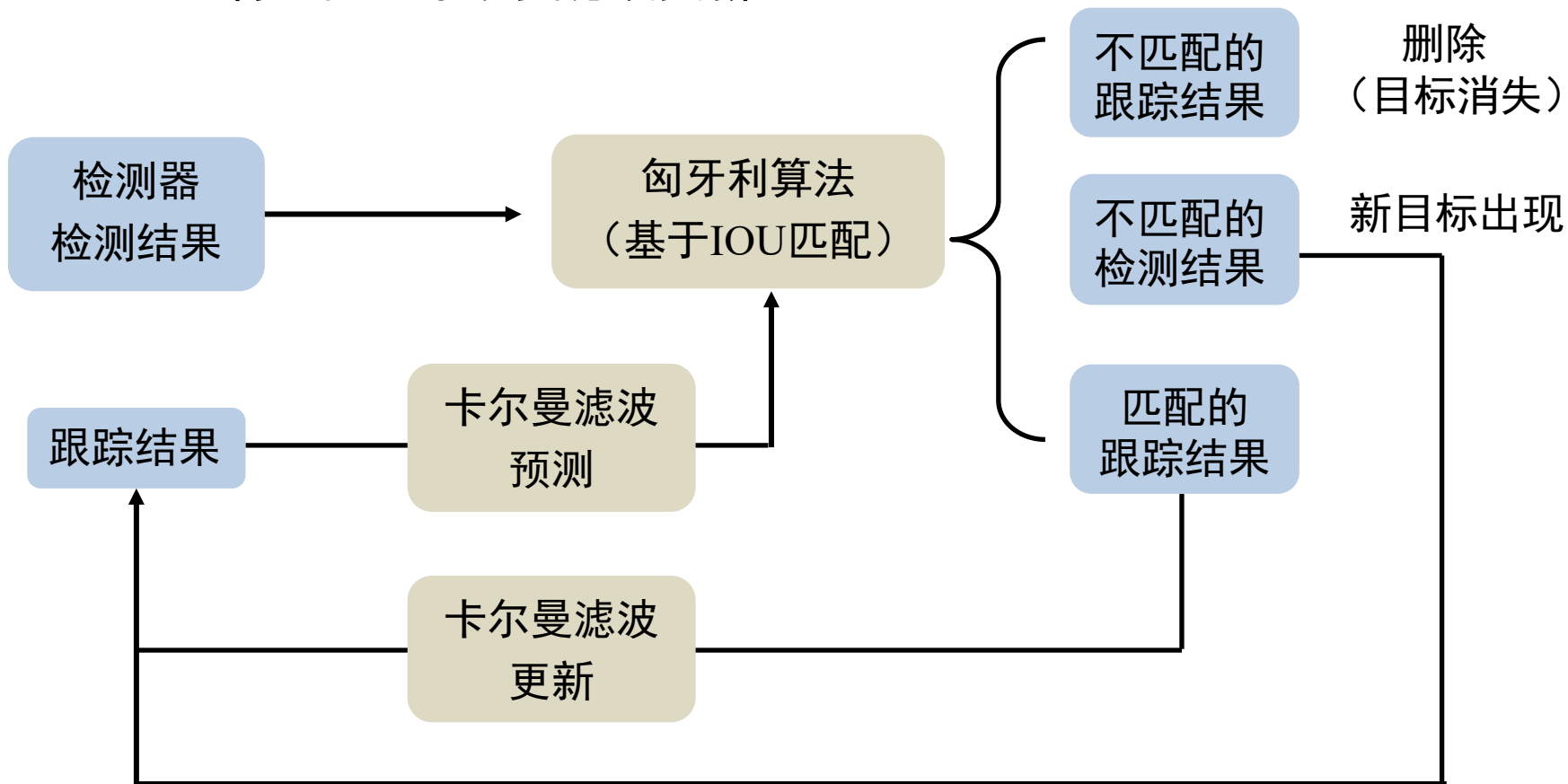
多目标跟踪

- SORT (Simple Online and Real-time Tracking) 算法
 - 首先利用训练好的目标检测器 (Faster RCNN) 得到每一帧的检测结果。SORT进一步使用卡尔曼滤波和匈牙利算法进行前后帧的数据关联
 - 匈牙利算法：将前一帧中的跟踪框与当前帧中的检测框进行关联，通过外观信息、马氏距离、或者矩形框的重合度 (IOU) 计算代价矩阵
 - 卡尔曼滤波：基于目标检测器与跟踪器的预测值，结合预测（先验分布）和测量更新（似然）的状态估计算法，实现更精确的估计。具体内容在单目标跟踪中已经介绍

多目标跟踪

□ SORT (Simple Online and Real-time Tracking) 算法

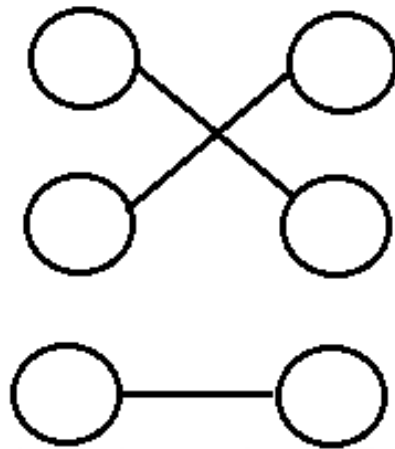
- 卡尔曼预测轨迹 (跟踪结果) → 匈牙利算法将检测和跟踪结果进行匹配 → 卡尔曼滤波更新



多目标跟踪

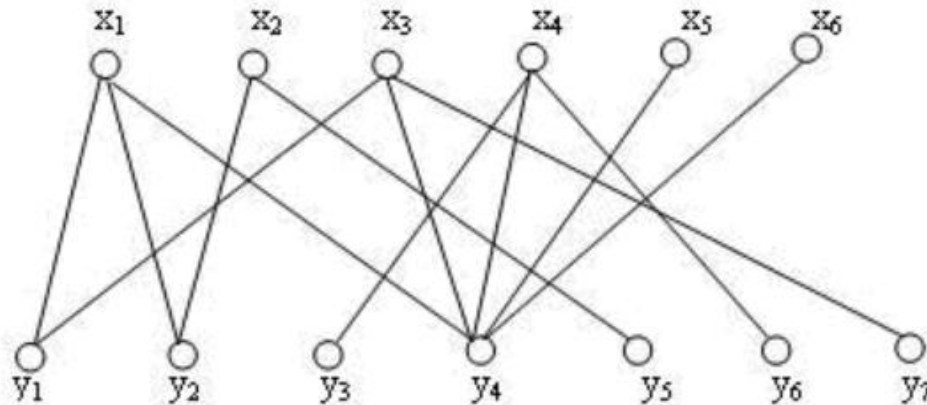
□ 匈牙利算法

- 匈牙利算法是一种在多项式时间内求解任务分配问题的组合优化算法
- 图G的一个匹配是由一组没有公共端点的不是圈的边构成的集合
 - ✓ 匹配是边的集合
 - ✓ 在该集合中，任意两条边不能有共同的顶点



多目标跟踪

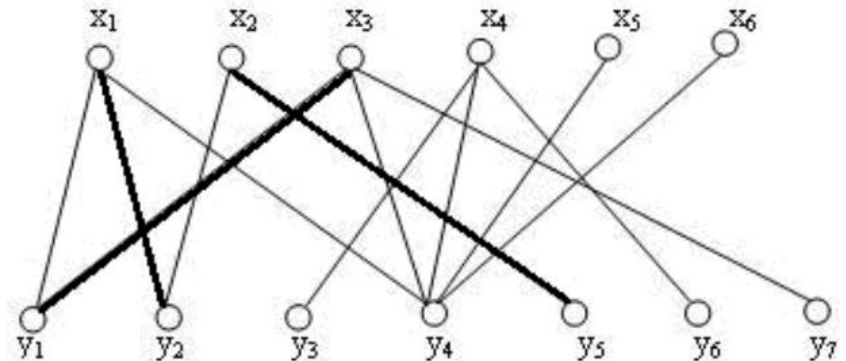
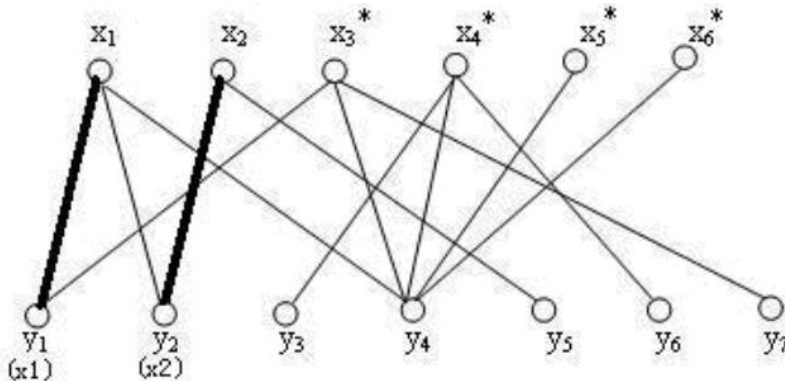
- 如果匹配中, $|V_1| \leq |V_2|$ 且匹配数 $|M| = |V_1|$, 此匹配称为**完全 (完备) 匹配**, 特别地, 当 $|V_1| = |V_2|$, 称为**完美匹配**。实现最可能多的匹配对数, 称为**最大匹配**
- 完美匹配一定是最大匹配, 最大匹配不一定是完美匹配
- 基本概念: 设 $G = (V, E)$ 是一个无向图。如顶点集 V 可分割为两个互不相交的子集, 选择这样的子集中边数最大的子集称为图的最大匹配问题, 如下图所示。



多目标跟踪

□ 匈牙利算法求解

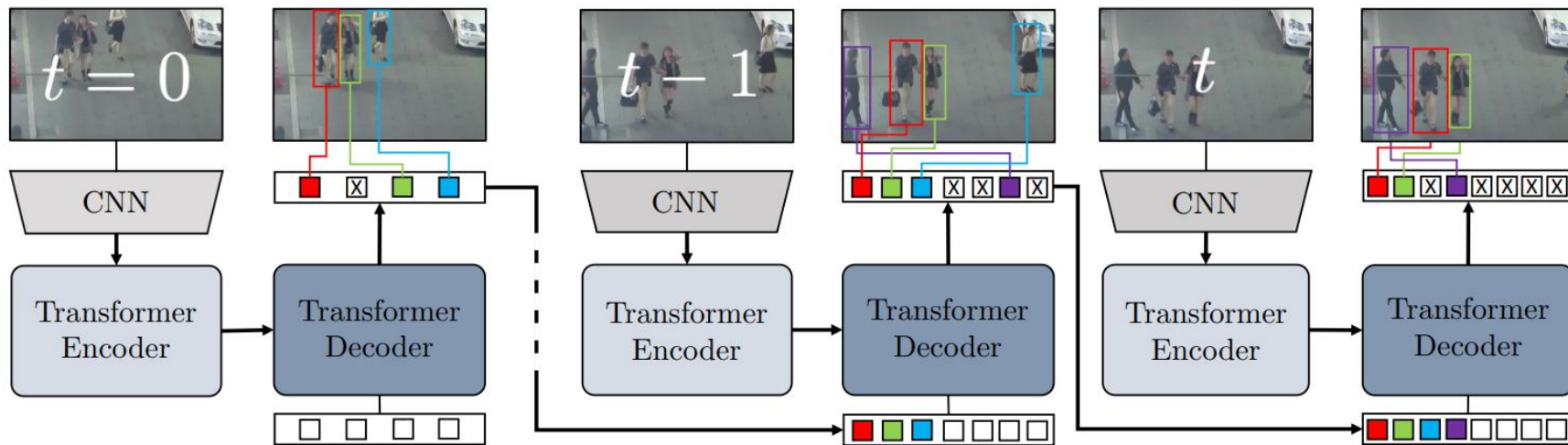
- 增广路径定义：M是图G的一个匹配。若P是图G中一条连通两个未匹配顶点的路径，并且不属于M的边和属于M的边（即待匹配和已匹配的边）在P上交替出现，且两个端点都不与M中的边关联，则称P为相对于M的一条增广路径。
- 左图：现有匹配 M 为 (x_1, y_1, x_2, y_2)
- 右图：路径 P $(x_3, y_1, x_1, y_2, x_2, y_5)$ 是 M 的增广路径
- 匈牙利算法寻找最大匹配，就是通过不断寻找原本匹配M的增广路径，得到更大的匹配。最大匹配不是唯一的，但是最大匹配的大小是唯一的



多目标跟踪

□ TrackFormer: Multi-Object Tracking with Transformers

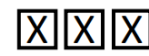
- 将多目标跟踪重定义为帧到帧集预测（set prediction）问题，并采用Transformer的Encoder-Decoder架构实现端到端跟踪
- Transformer Decoder从静态对象query中初始化新轨迹，并在空间和时间上自动跟踪现有轨迹。



目标跟踪query



静态对象query



退出目标

多目标跟踪

□ TrackFormer: Multi-Object Tracking with Transformers

■ 训练过程:

- ✓ 步骤一：将目标跟踪结果及静态检测结果与真实标注进行匹配

$K_t \cap K_{t-1}$: 训练时根据真实标注进行匹配（目标跟踪）

$K_{t-1} \setminus K_t$: 训练时与背景类别匹配（目标退出）

$K_t \setminus K_{t-1}$: 寻找最小代价映射 σ 进行匹配（新目标）

$$\hat{\sigma} = \arg \min_{\sigma} \sum_{k_i \in K_{\text{object}}} \mathcal{C}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}),$$

其中: $\mathcal{C}_{\text{match}} = -\lambda_{\text{cls}} \hat{p}_{\sigma(i)}(c_i) + \mathcal{C}_{\text{box}}(b_i, \hat{b}_{\sigma(i)}).$

$$\mathcal{C}_{\text{box}} = \lambda_{\ell_1} \|b_i - \hat{b}_{\sigma(i)}\|_1 + \lambda_{\text{iou}} \mathcal{C}_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}),$$

确定最优
匹配 $\pi(i)$

多目标跟踪

□ TrackFormer: Multi-Object Tracking with Transformers

■ 训练过程:

✓ 步骤二：根据匹配结果进行损失函数计算

$$\mathcal{L}_{\text{MOT}}(y, \hat{y}, \pi) = \sum_{i=1}^N \mathcal{L}_{\text{query}}(y, \hat{y}_i, \pi).$$

其中： $N = N_{\text{object}} + N_{\text{track}}$

$$\mathcal{L}_{\text{query}} = \begin{cases} -\lambda_{\text{cls}} \log \hat{p}_i(c_{\pi=i}) + \mathcal{L}_{\text{box}}(b_{\pi=i}, \hat{b}_i), & \text{if } i \in \pi \\ -\lambda_{\text{cls}} \log \hat{p}_i(0), & \text{if } i \notin \pi. \end{cases}$$



对匹配上的结果采用类别和bounding box的loss约束；对未匹配上的结果（与背景匹配）采用背景类别loss约束。



目标跟踪

- ☐ 单目标跟踪
- ☐ 多目标跟踪
- ☐ 主动目标跟踪



主动目标跟踪

- ☐ 概述
- ☐ 问题定义
- ☐ 实验场景
- ☐ 最新进展

主动目标跟踪

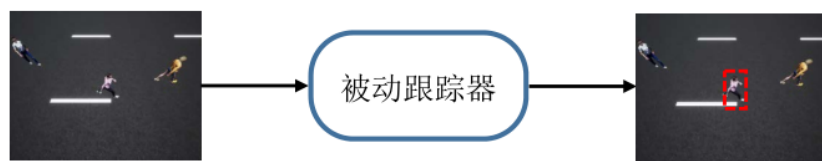
□ 概述

■ 什么是主动目标跟踪？

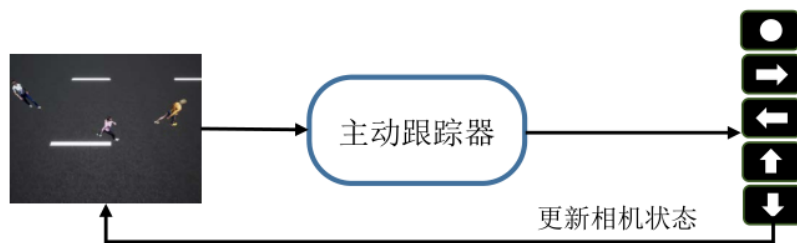
- ✓ 主动跟踪器根据当前和过去的观测序列自适应的输出相机的动作，可以使跟踪器始终保持跟踪目标。

■ 对比传统目标跟踪算法

- ✓ 传统的目标跟踪(也称被动跟踪)输出目标在视频序列图片中每一帧的矩形框，但是并不能改变摄像头的姿态以及方向。



(a) 被动目标跟踪框架



(b) 主动目标跟踪框架

图 1.1 被动与主动跟踪框架对比。

主动目标跟踪

■ 被动与主动目标跟踪的可视化结果



被动跟踪

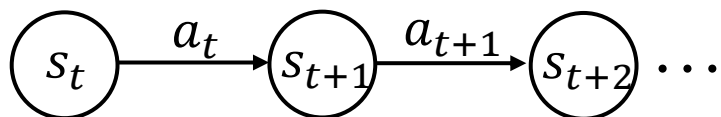


主动跟踪

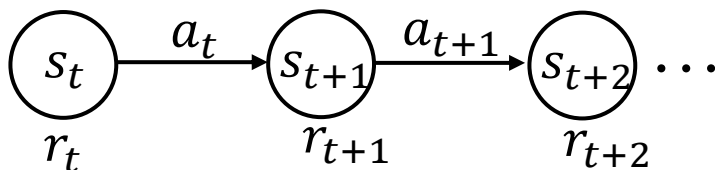
主动目标跟踪

□ 问题定义

- 主动跟踪器的决策过程满足马尔科夫性质， s_t 代表 t 时刻观测视野， a_t 为跟踪器根据当前观测 s_t 从策略分布 $\pi(\cdot | s_t)$ 采取的动作，则主动跟踪器的状态转移图如下：



- 为了进一步评估每一个状态跟踪效果的好坏可以定义奖励函数，这样在每一个时刻都会有一个奖励值。因此可以得到累积奖励值为 $R_{t:\infty} = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ ，其中 γ 为折扣因子。



- 最后定义跟踪器参数为 θ ，最终目标为最大化期望累积奖励 $\mathbb{E}(R_{1:\infty} | \pi(a|s; \theta))$ ，此问题很适合强化学习算法求解。

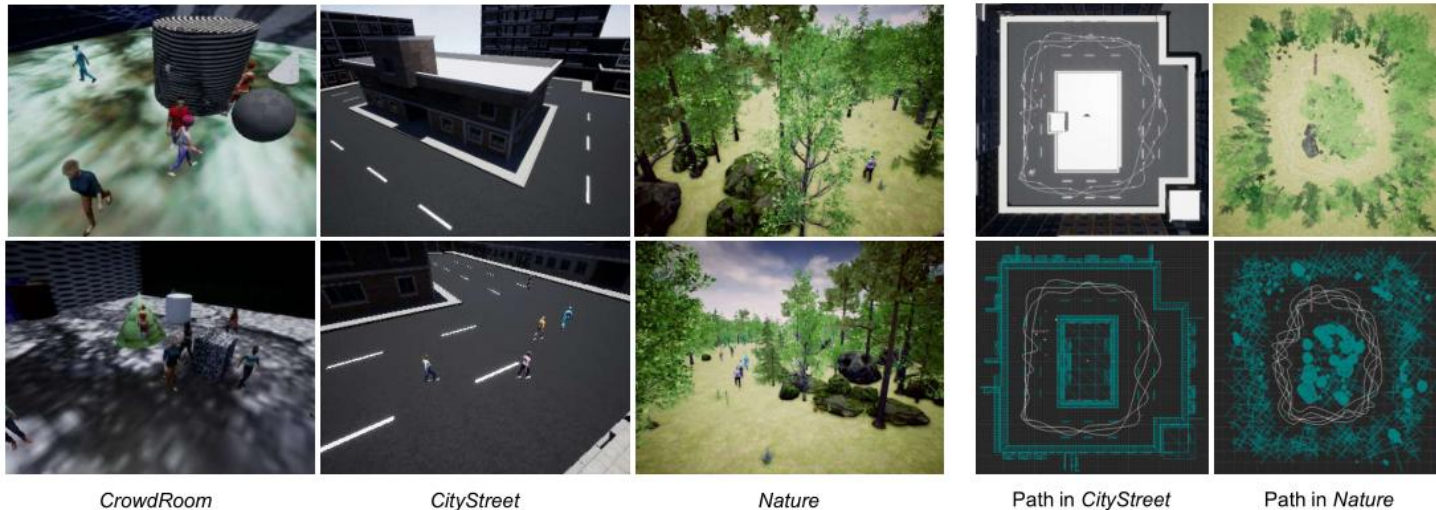
主动目标跟踪

□ 实验场景

主动跟踪需要根据摄像头的视野实时的调整姿态，但是在真实场景下实现较为困难，当前的最新算法都是在虚拟环境中训练与测试算法。

■ 虚幻引擎(Unreal Engine)

- ✓ 虚幻引擎可以帮助构建3D实时并且逼近真实的虚拟环境，通过虚幻引擎制作多样的环境可以快速帮助我们评测算法。



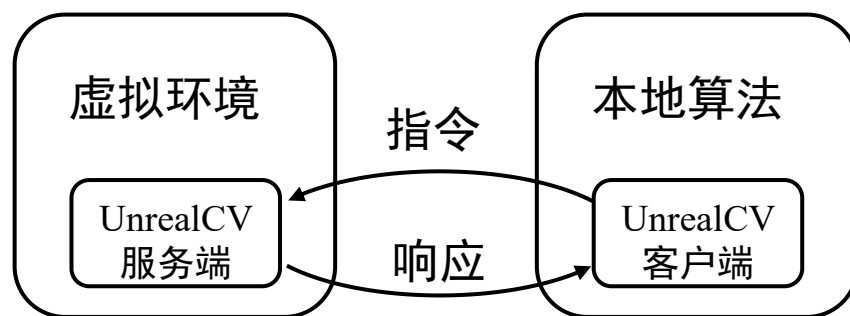
- ✓ 如上图，虚拟环境可以利用不同的素材创建不同的场景，也可以自定义环境中物体的纹理，人行走的方向，运动速度等。

主动目标跟踪

□ 实验场景

■ UnrealCV

- ✓ 有了虚拟环境，还需要让我们的代码与虚拟环境进行互动，UnrealCV就是解决代码与环境通信问题的接口，其原理如下：

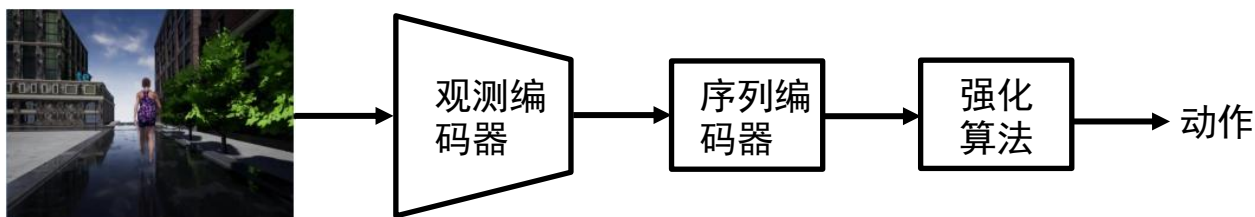


- ✓ 本地算法通过UnrealCV客户端发送指令。UnrealCV服务端收到指令后将虚拟环境执行后的结果返回给客户端。UnrealCV给我们提供了丰富的命令，比如：设置摄像头的位置以及姿态、获取摄像头的视野、设置行人的运动状态等。

主动目标跟踪

□ 单摄像头跟踪单人场景(2019 TPAMI)

- 场景中只包含一个目标和一个可移动摄像头，目标一般不会出现遮挡等问题，场景比较简单。
- 摄像头的动作包括左右旋转，前进后退。



观测编码器：用于提取当前观测视野的特征。

序列编码器：捕获长时间的视觉信息，常用的有LSTM，GRU等。

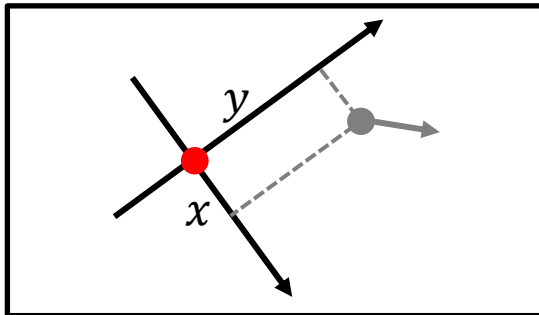
强化算法：可以选择不同的强化算法更新网络参数，比如DQN，A3C，PPO等。

主动目标跟踪

□ 单摄像头跟踪单人场景(2019 TPAMI)

■ 奖励函数

- ✓ 目标：使得摄像头(可视为一个机器人)始终保持跟踪目标，且目标与摄像头的距离合适。
- ✓ 可以以摄像头位置(红色点)和方向建立坐标系，灰色点 (x, y) 为目标位置，其中 y 方向为摄像头朝向， w 表示目标朝向与 y 方向的夹角。



因此奖励函数可以设计如下：

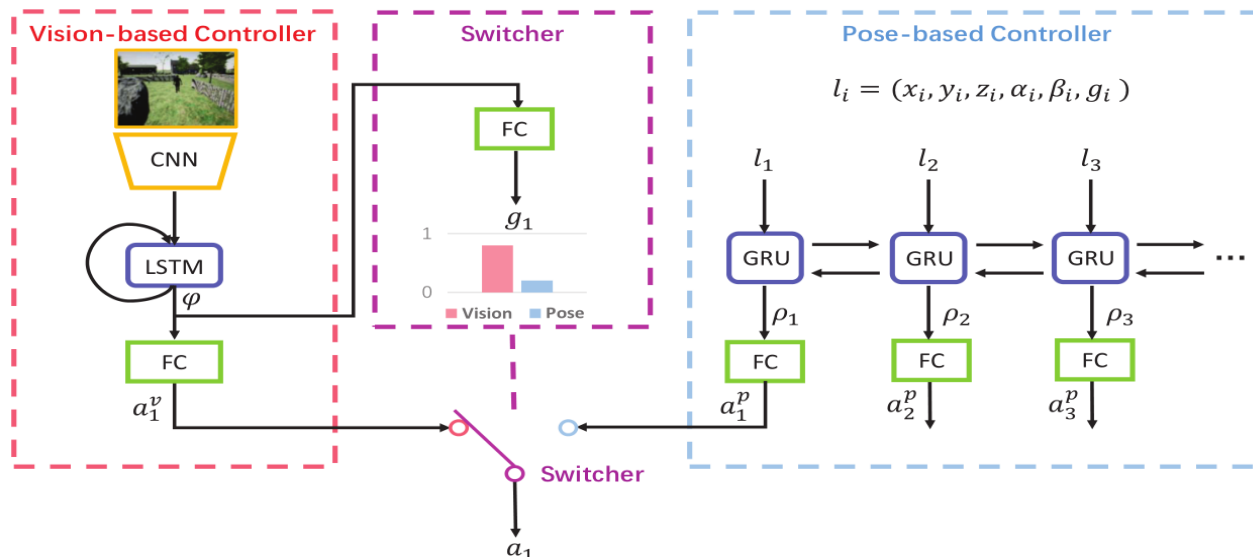
$$r = A - \left(\frac{\sqrt{x^2 + (y - d)^2}}{c} + \lambda |w| \right)$$

其中， A, d, c, λ 为定义的超参数。

主动目标跟踪

□ 多摄像头协同跟踪单人场景(2020 AAAI)

- 场景中包含一个目标和多个位置固定摄像头。
- 摄像头的动作包括左右旋转，垂直旋转，缩放。



Vision-based controller: 每个摄像头都有一个独立的视觉控制器跟踪目标。

Switcher: 判断目标是否在当前摄像头丢失。

Pose-based Controller: 当目标在当前摄像头丢失后，依靠别的摄像头的位置与姿态修正当前摄像头的姿态。

主动目标跟踪

□ 多摄像头协同跟踪单人场景(2020 AAI)

■ 奖励函数

- ✓ 目标：使得摄像头始终保持跟踪目标，目标处于摄像机视野的中心位置且目标大小合适。
- ✓ 定义 $\Delta\alpha_t$ 和 $\Delta\beta_t$ 分别表示目标与摄像机之间的偏转角和俯仰角的偏差。那么可以定义奖励函数为：

$$R_{d,t} = \begin{cases} 1 - \frac{\Delta\alpha_t}{\alpha_{max}} - \frac{\Delta\beta_t}{\beta_{max}}, & (a) \\ 0, & (b) \\ -1, & (c) \end{cases}$$

其中，(a)表示目标在视野中，(b)表示目标被遮挡，(c)表示目标超出视野。 α_{max} 和 β_{max} 表示相机最大能控制的偏转角与俯仰角的旋转角度。此外为了目标在视野中大小合适，对于缩放定义奖励函数为：

$R_{z,t} = 1 - \frac{\Delta\xi_t}{\xi_{max}}$ ，其中， $\Delta\xi_t$ 表示t时刻缩放比例的偏差。则最终的奖励函数为：

$$R_t = R_{d,t} + R_{z,t}$$