

## 第四章



# 目标检测

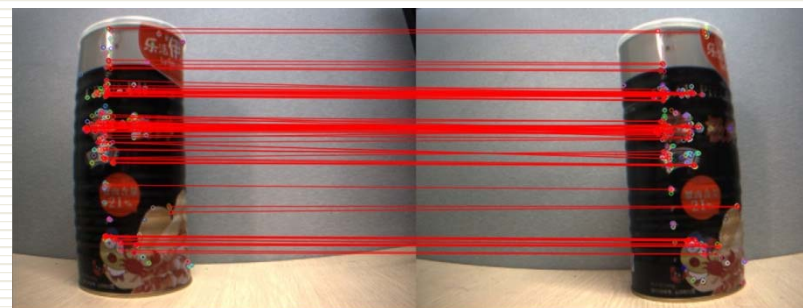
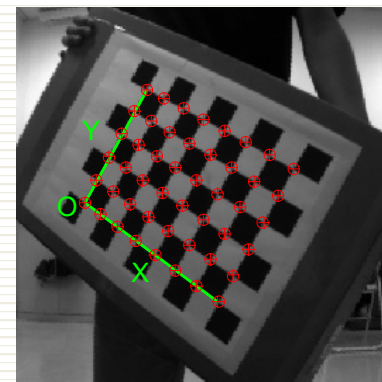
李静

# 目标检测方法

- **静态目标检测**：目标几何尺寸测量、瑕疵检测等，需要提高单帧图像的处理速度；
- **动态目标检测**：提取运动目标特征进行检测。

- 常用 **目标特征** 及表示方式：

- ◆ 颜色：RGB、HSV空间，提取颜色直方图；
- ◆ 纹理：灰度共生矩阵；
- ◆ 轮廓：边缘检测、模板等；
- ◆ 角点：图像中局部曲率突变的点，在角点处沿各个方向都存在较大的灰度值变化。
- ◆ 特征点：SIFT、Surf、ORB等。



# 主要内容



- 一、简单背景目标检测
- 二、特征明显的目标检测
- 三、目标存在旋转或尺度变化
- 四、同类多目标检测或多类多目标分割



# 一、简单背景目标检测

对于摄像机静止，简单背景的目标检测：

- 基于阈值分割的目标检测
- 基于运动信息的目标检测





# 一、简单背景目标检测

## (一) 基于阈值分割的目标检测

### (1) 直方图阈值法

基于直方图有两个波峰，可通过阈值T把前景从背景里提取出来。

设图像以灰度形式表示为  $f(x, y)$ ,  $x = 1, \dots, m$ ,  $y = 1, \dots, n$   $m$ 和 $n$ 分别为CCD所采图像的行像素数和列像素数。阈值处理过程为：

$$F(x, y) = \begin{cases} f(x, y), & f(x, y) \geq T \\ 0, & f(x, y) < T \end{cases}$$



阈值100



# 一、简单背景目标检测

## (2) 自适应阈值法——最大类间方差法

1979年由日本学者大津于提出的，是一种自适应的阈值确定的方法，又叫大津法，简称OTSU。

**原理：**按图像的灰度特性，将图像分成背景和目标两部分。背景和目标之间的**类间方差**越大，说明构成图像的两部分的差别越大，当部分目标错分为背景或部分背景错分为目标都会导致两部分差别变小。因此，使类间方差最大的分割意味着错分概率最小。

**类间方差公式：**

$$g = (\bar{x}_1 - \bar{x})^2 \cdot \frac{n_1}{n} + (\bar{x}_2 - \bar{x})^2 \cdot \frac{n_2}{n}$$

$\bar{x}$  n个数据平均值

$\bar{x}_1$  第一组 $n_1$ 个数据平均值，  $\bar{x}_2$  第二组 $n_2$ 个数据平均值。



# 一、简单背景目标检测

## 算法:

对于图像  $I(x, y)$ , 图像的大小为  $M \times N$ ,

- 前景(即目标)和背景的分割阈值记作  $T$ ,
- 属于前景的像素点数占整幅图像的比例记为  $\omega_0$ , 其平均灰度  $\mu_0$ ;
- 背景像素点数占整幅图像的比例为  $\omega_1$ , 其平均灰度为  $\mu_1$ 。
- 图像的总平均灰度记为  $\mu$ , 类间方差记为  $g$ 。
- 图像中像素的灰度值小于阈值  $T$  的像素个数记作  $N_0$ , 像素灰度大于阈值  $T$  的像素个数记作  $N_1$ ,

$$\omega_0 = N_0 / M \times N \quad \omega_1 = N_1 / M \times N \quad N_0 + N_1 = M \times N \quad \omega_0 + \omega_1 = 1$$

$$\mu = \omega_0 * \mu_0 + \omega_1 * \mu_1 \quad g = \omega_0(\mu_0 - \mu)^2 + \omega_1(\mu_1 - \mu)^2$$

$$g = \omega_0 \omega_1 (\mu_0 - \mu_1)^2$$

采用遍历的方法得到使类间方差最大的阈值  $T$ , 即为所求。



# 一、简单背景目标检测

**OTSU** float u0, u1, w0, w1; int count0, t, maxT; float devi, maxDevi = 0; //方差及最大方差  
**程序:** int i, sum = 0;  
for (i = 0; i < 256; i++){  
    sum = sum + hist[i];  
}  
for (t = 0; t < 255; t++){  
    u0 = 0; count0 = 0;  
    for (i = 0; i <= t; i++) //阈值为t时, c0组的均值及产生的概率;  
    {  
        u0 += i \* hist[i]; count0 += hist[i];  
    }  
    u0 = u0 / count0; w0 = (float)count0/sum;  
    for (i = t + 1; i < 256; i++) //阈值为t时, c1组的均值及产生的概率  
    {  
        u1 += i \* hist[i];  
    }  
    u1 = u1 / (sum - count0); w1 = 1 - w0;  
    devi = w0 \* w1 \* (u1 - u0) \* (u1 - u0); //两类间方差  
    if (devi > maxDevi) //记录最大的方差及最佳位置  
    {  
        maxDevi = devi;  
        maxT = t;  
    }  
}





# 一、简单背景目标检测

## (3) 自适应阈值方法——Kittle法

一种快速的全局阈值法。效果与OTSU差不多，但速度快好多倍。

**思想：**计算整幅图像的梯度灰度的平均值，以此平均值做为**阈值**。

$$f_i(i, j) = \frac{\partial f(i, j)}{\partial i} = f(i+1, j) - f(i-1, j)$$

$$f_j(i, j) = \frac{\partial f(i, j)}{\partial j} = f(i, j+1) - f(i, j-1)$$

$$grad(i, j) = \max(|f_i|, |f_j|)$$

$$u(i, j) = grad(i, j) * f(i, j)$$

$$KT = sum(grad(i, j) * f(i, j)) / sum(grad(i, j))$$

给定一幅灰度图像，编写Kittle算法，求取阈值。

定义：Mat image; float Grads, sumGrayGrads, sumGrads;



# 一、简单背景目标检测

## 阈值效果



原始灰度图



OTSU 阈值117



Kittle 阈值115



# 一、简单背景目标检测

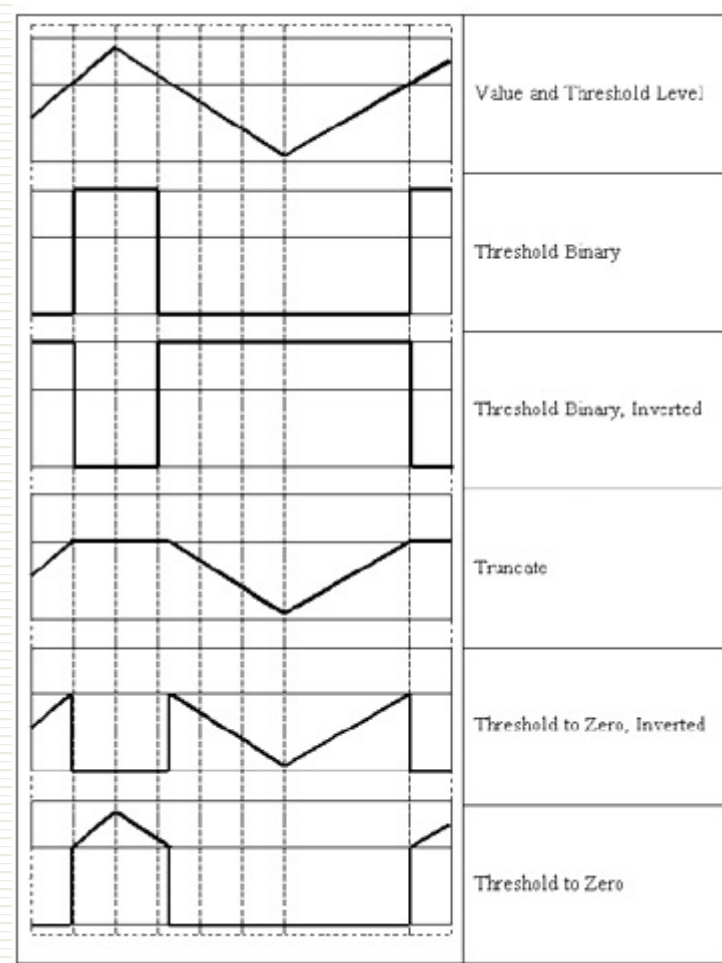
## (4) OpenCV程序实现

Double **threshold**(CvArr\* **src**, CvArr\* **dst**, double **threshold**,  
double **max\_value**, int **threshold\_type**)

阈值类型	操作
CV_THRESH_BINARY	$dst(x,y) = (src(x,y) > T) ? M : 0$
CV_THRESH_BINARY_INV	$dst(x,y) = (src(x,y) > T) ? 0 : M$
CV_THRESH_TRUNC	$dst(x,y) = (src(x,y) > T) ? M : src$
CV_THRESH_TOZERO_INV	$dst(x,y) = (src(x,y) > T) ? 0 : src$
CV_THRESH_TOZERO	$dst(x,y) = (src(x,y) > T) ? src : 0$

## CV\_THRESH\_OTSU

使用大律法OTSU得到的全局自适应阈值来进行二值化图片，而参数中的threshold不再起作用



# 实验课 —— 编写程序



1、编写固定阈值分割算法；

$$F(x, y) = \begin{cases} f(x, y), & f(x, y) \geq T \\ 0, & f(x, y) < T \end{cases}$$

2、编写自适应阈值分割算法（Ostu, Kittler）；



# 一、简单背景目标检测

## (二) 基于运动信息的目标检测

### ❖ 时间差分法

通过比较相邻**2或3帧** **图像差异**实现场景变化检测，对动态环境有较强适应性，但检测精度不高，难获得目标精确描述。

### ❖ 背景减除法

适用于摄像机静止情形，其关键是**背景建模**，性能与监控场景复杂情况和系统要求有关，典型算法有中值、自适应模型、高斯模型、多模态均值等。

# 一、简单背景目标检测

## 1、时间差分法

通过比较相邻2或3帧**图像差异**实现场景变化检测，对动态环境有较强适应性，但检测精度不高，难获得目标精确描述。



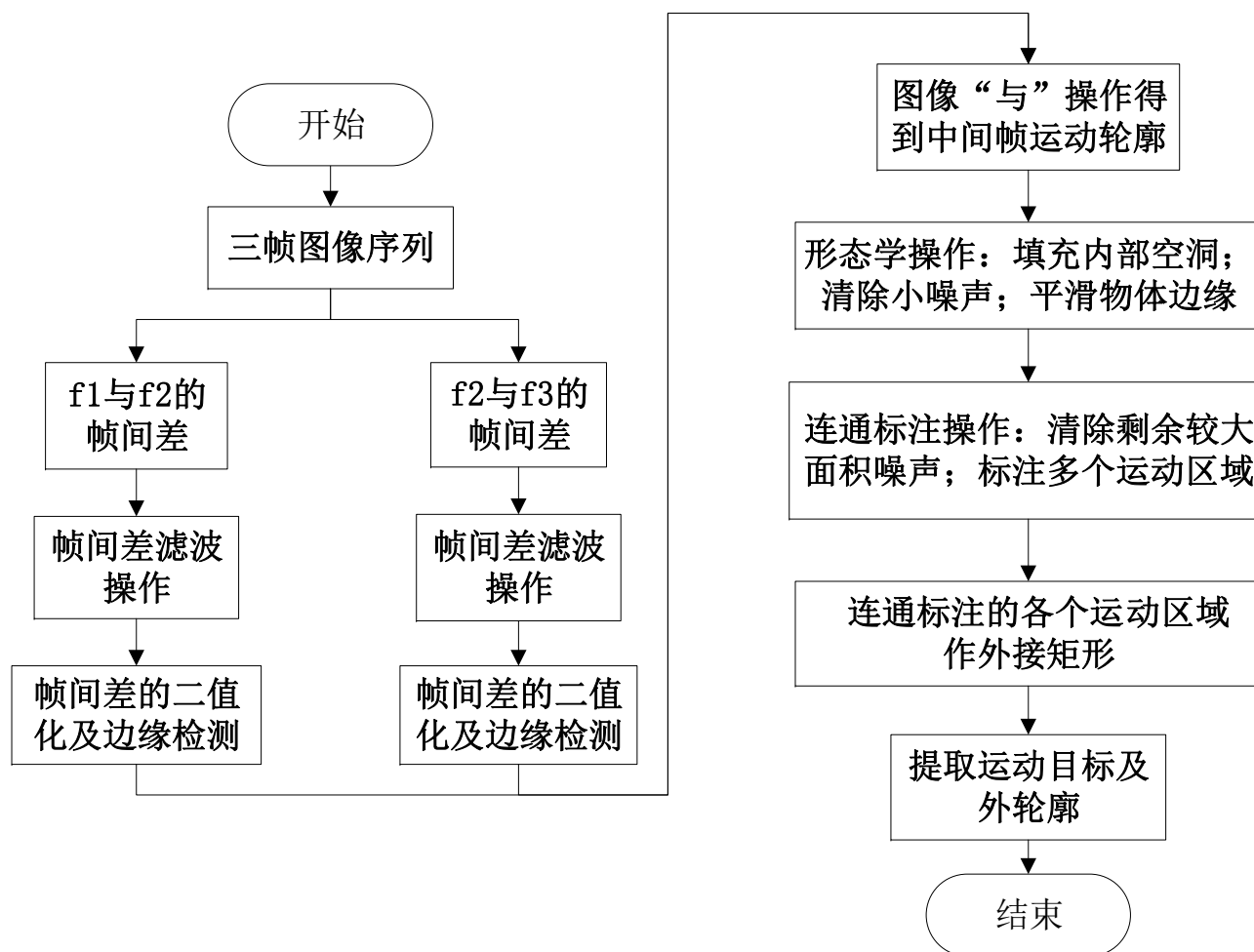
$$D_t(x, y) = \begin{cases} 1, & |I_t(x, y) - I_{t-1}(x, y)| > T \\ 0, & \text{otherwise} \end{cases}$$

- ❖ **优点**：鲁棒性好，运算量小，易于软件实现
- ❖ **缺点**：对噪声有一定的敏感性，运动实体内部也容易产生**空洞现象**，阈值T缺乏自适应性，当光照变化时，检测算法难以适应环境变化



# 一、简单背景目标检测

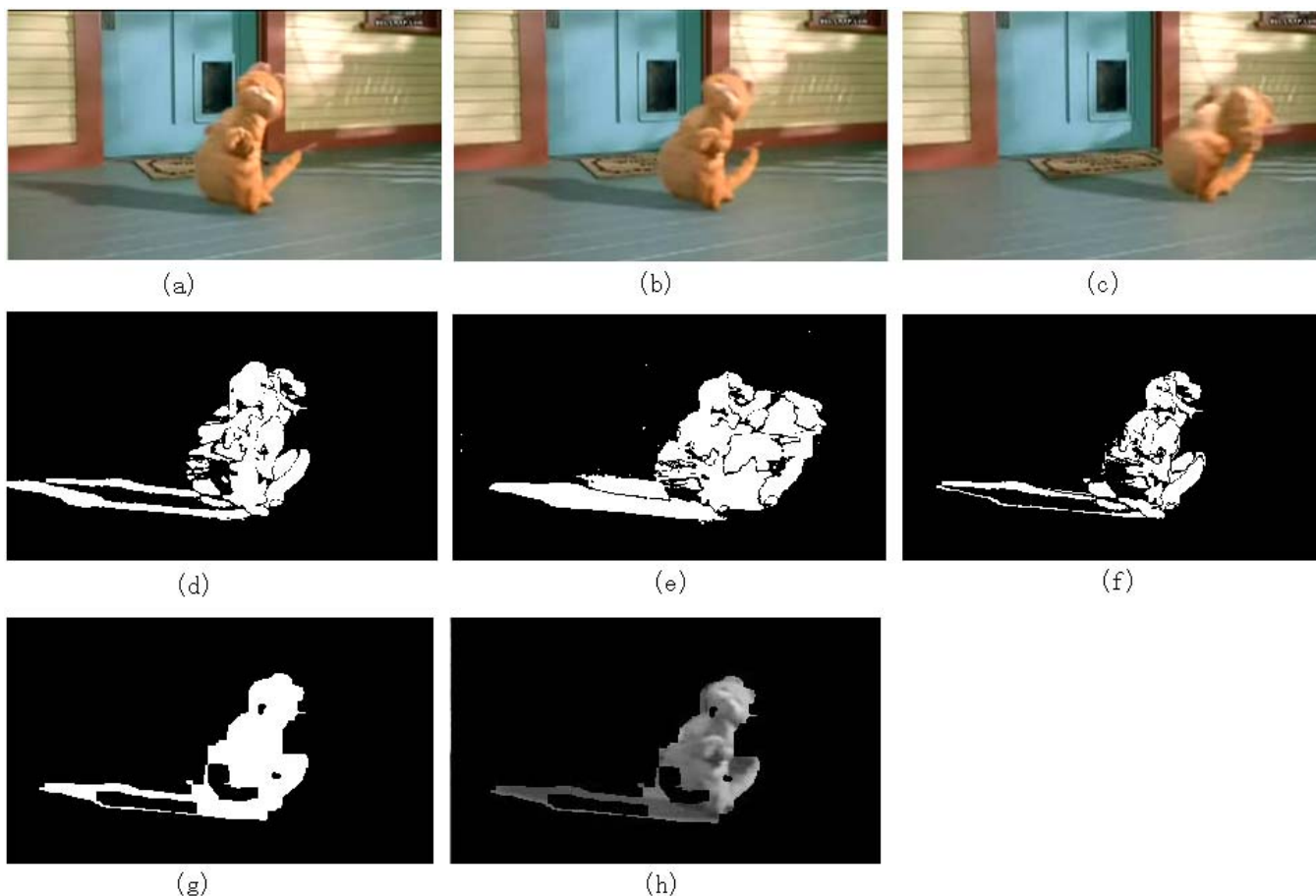
## 视频举例——对称差分法





# 一、简单背景目标检测

## 对称差分法实验效果



(a) 第一帧； (b) 第二帧； (c) 第三帧； (d)  $|f_1 - f_2|$  (e)  $|f_3 - f_2|$  (f) 二值化结果； (g) 形态学处理结果； (h) 目标提取结果

对称差分法测试结果





# 一、简单背景目标检测

## 2、背景减除法

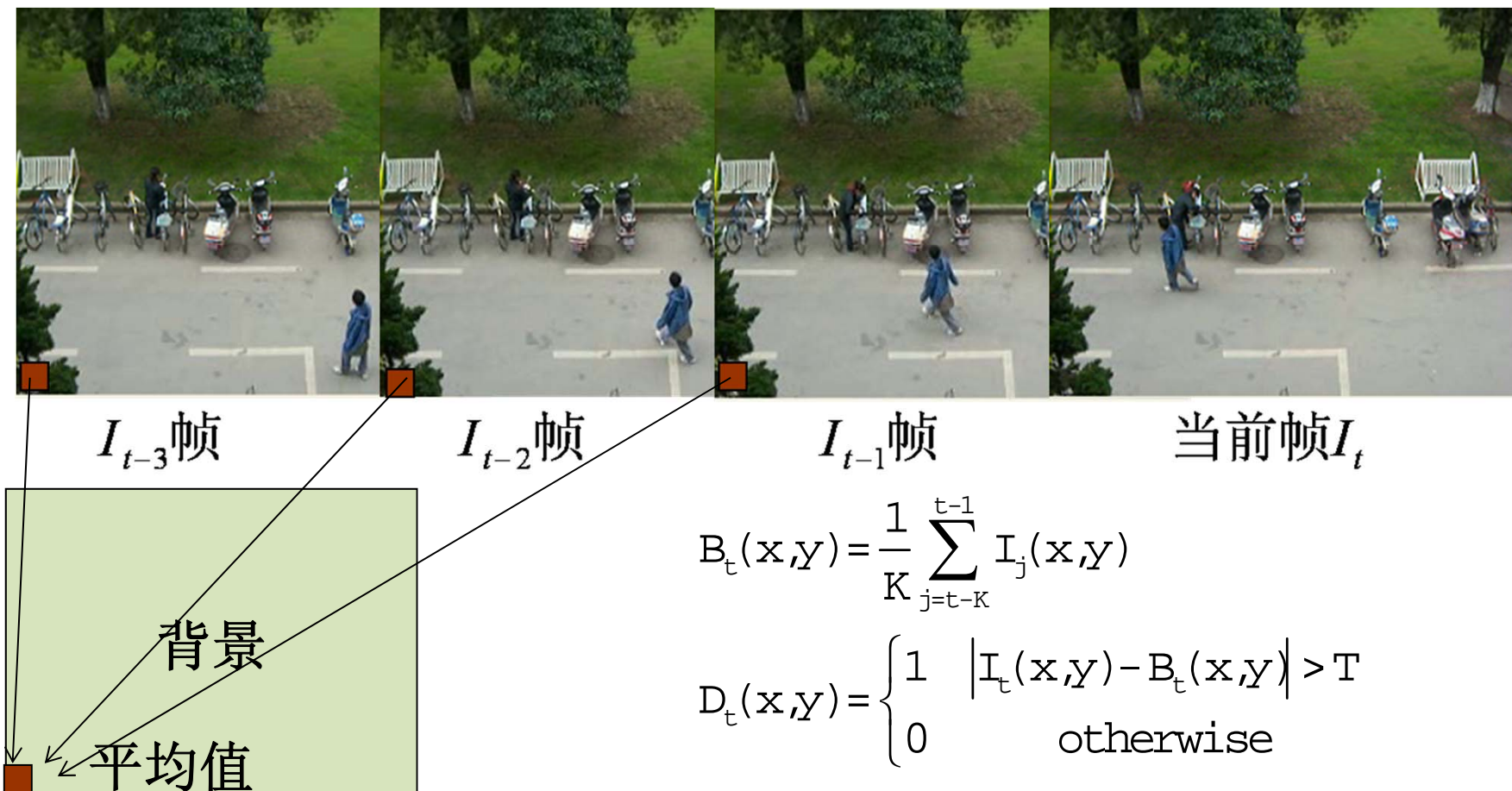
适用于摄像机静止情形，其关键是**背景建模**，性能与监控场景复杂情况和系统要求有关，典型算法有中值、自适应模型、高斯模型、多模态均值等。

背景模型建立方法：

- ❧ 均值模型
- ❧ 自适应背景模型
- ❧ 单高斯模型
- ❧ 混合高斯模型

# 一、简单背景目标检测

## (1) 均值模型



$I_{t-3}$  帧       $I_{t-2}$  帧       $I_{t-1}$  帧      当前帧  $I_t$

背景

平均值

$$B_t(x,y) = \frac{1}{K} \sum_{j=t-K}^{t-1} I_j(x,y)$$
$$D_t(x,y) = \begin{cases} 1 & |I_t(x,y) - B_t(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$

Default:  $T=60$ ,  $K=3$

**前提：**在前K帧图像中，某像素点在超过一半的时间里呈现场景背景像素值。



# 一、简单背景目标检测

## (2) 自适应背景模型

$$\alpha \times \text{当前帧k} + (1 - \alpha) \times \text{前一背景k-1} = \text{当前背景}$$

当前帧k

前一背景k-1

$$B_1(x,y) = I_1(x,y)$$

$$B_t(x,y) = \alpha I_t(x,y) + (1 - \alpha) B_{t-1}(x,y)$$

$\alpha$  为自适应参数，其取值

$$D_t(x,y) = \begin{cases} 1 & |I_t(x,y) - B_t(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$

直接影响背景的更新质量

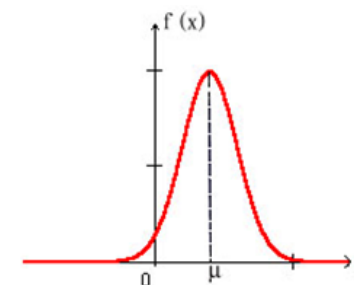
Default:  $\alpha=0.03$ ,  $T=60$



# 一、简单背景目标检测

## (3) 单高斯模型

高斯分布与背景建模的关系：图像中每一个像素点的颜色值作为一个随机过程，并假设该点的像素值出现的**概率服从高斯分布**。令  $I(x, y, t)$  表示像素点  $(x, y, t)$  在  $t$  时刻的像素值，则有：



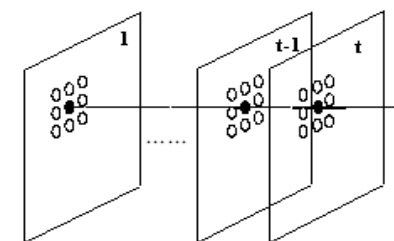
$$P(I(x, y, t)) = G(x, ut, \delta_t) = \frac{1}{\delta\sqrt{2\pi}} \exp\left(-\frac{(x-u)^2}{2\delta^2}\right)$$

其中  $u_t$  和  $\delta_t$  分别为  $t$  时刻像素高斯分布的期望和标准差。

假设每个像素的灰度值在时间域上满足高斯分布：

$$D(x, y) = \begin{cases} 1 & |I_t(x, y) - \mu_t(x, y)| > \lambda \delta_t(x, y) \\ 0 & \text{otherwise} \end{cases}$$

**Default:**  $\lambda = 4.5$





# 一、简单背景目标检测

## 算法流程:

- 1) 用第一帧图像初始化背景模型, 其中std\_init通常设置为20。

$$u_0(x, y) = I(x, y, 0)$$

$$\delta_0(x, y) = \text{std\_init}$$

$$\delta_0^2(x, y) = \text{std\_init} * \text{std\_init}$$

- 2) 检测前景与背景像素。

背景像素检测公式:  $|I(x, y, t) - u_{t-1}(x, y)| < \lambda \delta_{t-1}$

前景像素检测公式:  $|I(x, y, t) - u_{t-1}(x, y)| \geq \lambda \delta_{t-1}$

- 3) 模型更新

$$\mu_t = \alpha \mu_{t-1} + (1 - \alpha) I_t$$

$$\delta_t^2 = \alpha \delta_{t-1}^2 + (1 - \alpha) (I_t - \mu_t)^2$$



**优点:** 在室内或不是很复杂的室外环境中, 单高斯模型亦能达到很好的检测效果, 处理速度快, 分割对象比较完整。

**缺点:** 对于复杂变化的背景, 噪声增多, 背景模型将变得不太稳定, 有可能将外界干扰 (典型的如树枝摇晃等) 判断为运动目标, 造成系统的误判。



# 一、简单背景目标检测

## (4) 混合高斯模型

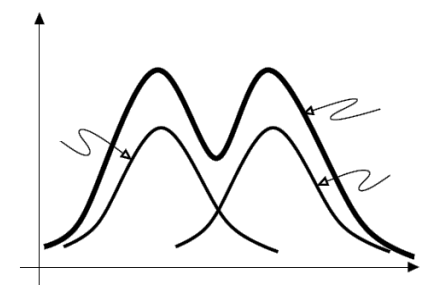
针对复杂背景（特别是有微小重复运动的场合，如摇动的树叶、灌木丛、旋转的风扇、海面波涛、雨雪天气、光线反射等），可采用**多个单高斯函数**来描述场景背景，并且利用在线估计来更新模型，可靠地处理了**光照缓慢变化、背景混乱运动**（树叶晃动）等影响。

设用来描述每个像素点背景的高斯分布共有K个

$$P(X_i) = \sum_{i=1}^K w_{i,t} G_i(X_t, \mu_{i,t}, \delta_{i,t})$$

$$G_i(X_t, \mu_{i,t}, \delta_{i,t}) = \frac{1}{\delta \sqrt{2\pi}} e^{-\frac{(x - \mu_{i,t})^2}{2\delta_{i,t}^2}}$$

$w_{i,t}$  为t时刻第i个高斯分布的权值





# 一、简单背景目标检测

## 算法流程：

- 模型初始化
- 模型匹配与参数更新
- 生成背景分布
- 检测前景

### 1) 模型初始化

- 取第一帧图像中的像素值对高斯分布的均值进行初始化
- 对第一个高斯分布的权值赋予较大值（0.5），其他高斯分布赋较小权值（ $0.5/(K-1)$ ）
- 所有高斯函数的方差取较大初始值（30）





# 一、简单背景目标检测

## 2) 模型匹配与参数更新

K个分布按  $\omega_{i,t} / \delta_{i,t}$  排序，将新像素与模型中的高斯分布依序匹配：

**匹配条件：**  $|I_t - \mu_{i,t-1}| \leq D_1 \delta_{i,t-1}$   $D_1$  为自定义参数

□ 若该高斯分布匹配，其参数按下式**更新**。

$$\omega_{i,t} = (1 - \alpha)\omega_{i,t-1} + \alpha$$

$$\mu_{i,t} = (1 - \rho)\mu_{i,t-1} + \rho I_t$$

$$\delta_{i,t}^2 = (1 - \rho)\delta_{i,t-1}^2 + \rho(I_t - \mu_{i,t})^2$$

式中  $0 \leq \alpha \leq 1$  是自定义的学习率， $\rho \approx \frac{\alpha}{\omega_{i,t}}$  是参数学习率。不匹配的分布仅权值按  $\omega_{i,t} = (1 - \alpha)\omega_{i,t-1}$  衰减。

□ 若无分布和  $I_t$  匹配，则**最小权值分布**被替换成均值为  $I_t$ ，标准差为  $\delta_0$ ，权值为  $\omega_{K,t} = (1 - \alpha)\omega_{K,t-1} + \alpha$  的高斯分布。其余分布**仅权值**按  $\omega_{i,t} = (1 - \alpha)\omega_{i,t-1}$  更新。





# 一、简单背景目标检测

## 3) 生成背景分布

分别按优先级  $\omega_{i,t} / \delta_{i,t}$  从大到小排列，T为背景权值部分和阈值，如果前  $N_B$  个分布的权值和大于T，则这些分布是背景分布，其它为前景分布。

## 4) 检测前景

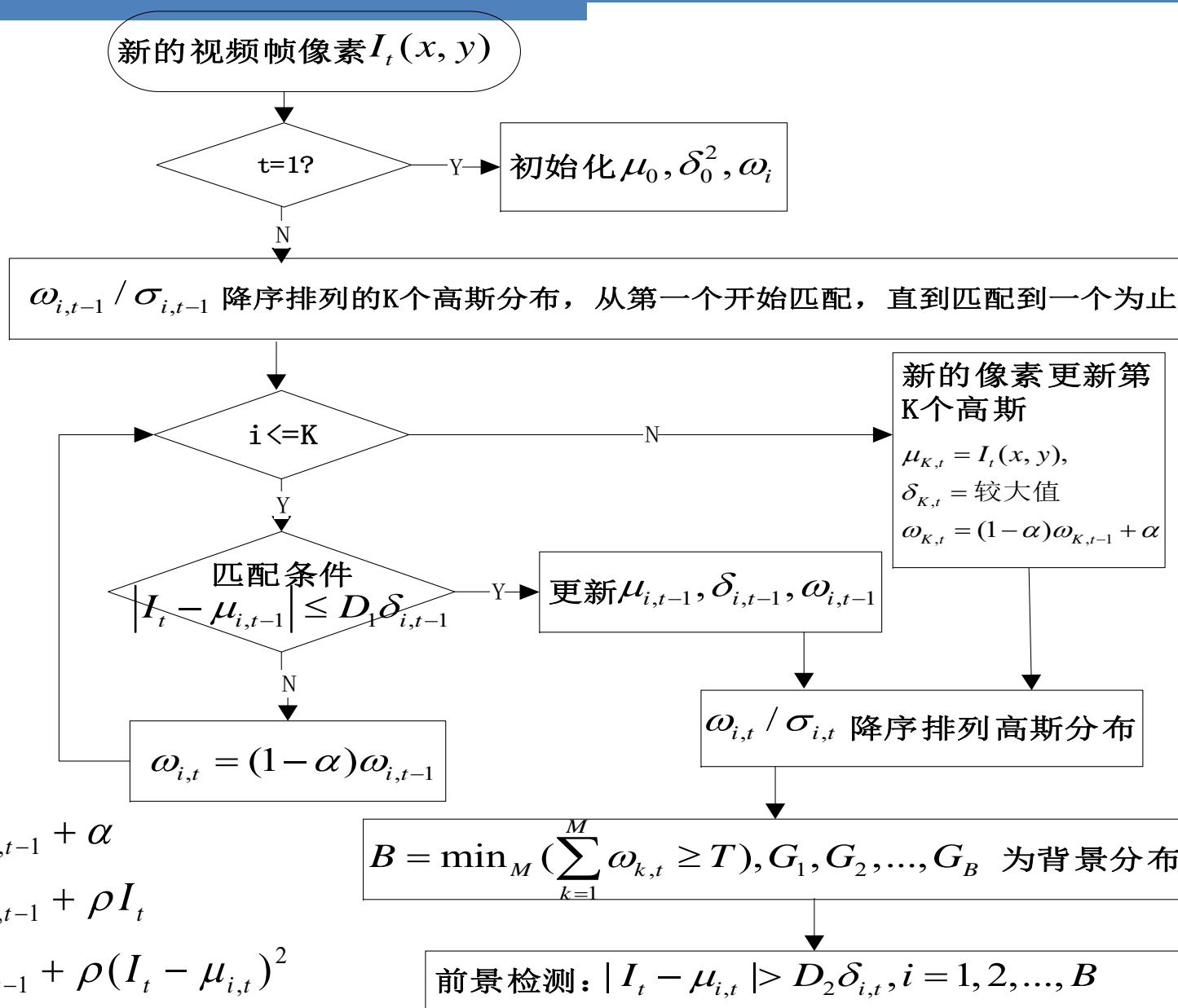
若所有背景分布与  $I_t$  都满足下式，则判定为前景点，否则为背景点。  
( $D_2$ 为自定义参数)

$$|I_t - \mu_{i,t}| > D_2 \delta_{i,t}, i = 1, 2, \dots, N_B$$



# 一、简单背景目标检测

## 混合高斯模型流程图



更新方程:

$$\omega_{i,t} = (1 - \alpha)\omega_{i,t-1} + \alpha$$

$$\mu_{i,t} = (1 - \rho)\mu_{i,t-1} + \rho I_t$$

$$\delta_{i,t}^2 = (1 - \rho)\delta_{i,t-1}^2 + \rho(I_t - \mu_{i,t})^2$$

# 实验课 —— 编写程序



## 1、编写单高斯背景建模程序

[https://blog.csdn.net/qq\\_22562949/article/details/46445891](https://blog.csdn.net/qq_22562949/article/details/46445891)

## 2、编写混合高斯背景建模程序