在实际操作中，处理对象是从视频采集卡输入的实时图像序列，为了更好的检测图像中的移动物体，摄像头需要对于背景图进行学习，即获取静止的背景图像信息。

对于视频中运动物体的检测主要措施分为两种:宏观检测法以及微观检测法。顾名思义，宏观检测法就是对于获取到的整副图像进行检测，反之，微观检测法是针对图像的 ROI(感兴趣区域)进行检测。背景差法，帧间差法以及综合法。

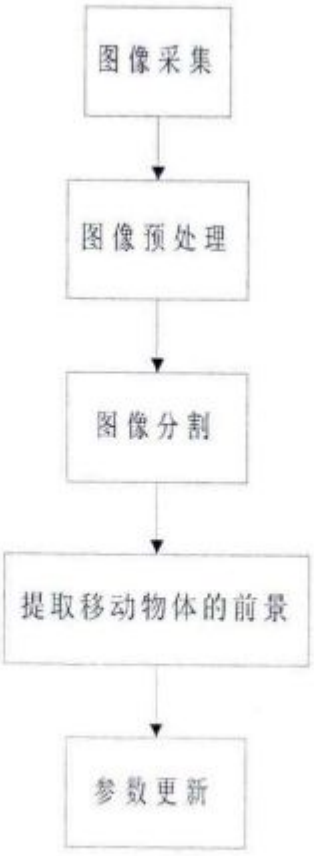


图 2.1　检测运动物体流程图

背景差分法：利用了获取到的背景的图像以及按照一定的算法或者人为的获取一张没有目标物体的背景图像，两者进行相减。然后通过阈值获得二值图像，根据二值图像从而分割出图像中的目标物体，从而可以达到检测出目标物体的目的。弊端：对于所比较的背景图示需要进行实时更新的，才可以满足一定的检测准确性。

帧间差分法：从实时获取的帧图像，进行一帧一帧的图像作差值比较，从而得到的差值图像，由于目标物体不停的运动，这样相连续的帧差值图像就能得到目标物体的运动轨迹，接着，结合图像的分割技术，就能得到移动物体的轮廓。

检测运动目标物体的整个过程体系为:捕捉视频流一转换视频格式一预处理图像一提取目标前景物一减小环境对于图像处理的误差一提取运动物体的特征一精确的跟踪运动物体。
步骤详解：

(I)捕获视频流:利用现场的摄像头获取到实时的视频码流。

(2)图像预处理技术:对于捕获到的视频流图像，为了检测和跟踪的效果更为出色，需要进行预处理，滤除图像中的噪点、平滑处理图像，为之后的分析和处理图像作好准备。
预处理图像数据:图像平滑处理;图像的填充处理。

2.4.1 视频图像的平滑滤波处理

滤波处理图像能够减小图像中的噪声，在提取目标物体之前去除图像的琐碎的细节，简化之后的算法。其产生的效果:平滑曲线，柔化线与线连接的摩擦等。滤波理论上由线性和非线性的两种方式。前者的算法简单，运算速度也比较快，但是对于处理后的图像会造成图像的不清晰；然而，后者相对于前者造成的图像模糊等问题就可以很好的解决，其在去除信号噪声的同时能够很好的保持信号的局部特征，但是，同时，对于其算法的运算速度就会受到一定的影响。

① 邻域平均滤波

这个方法的原理是由一个 NxN 大小的模板 S，在这个范围对于图像进行滑动处理，假设模板的中点象素点的灰度表示为 I（x,y），那么经过邻域平均滤波方法的滤波后，此点的像素值变化成:

$$\hat{I}(x,y)=\frac{1}{N^2}\sum_{x,y\in s} I(x,y)$$

从式可得:邻域滤波的在减小图像噪声的同时，图像将会变得模糊，而且当所需要滑动处理的平面越大，消除噪声的效果也会越显著，但是相对的，图像的质量就会下降。

② 加权平均滤波

此方法是上述方法的优化改进算法。对于同一大小的模板 S，对于其中不同位置的像素值运用不同的数值，规则是离象素点的中心越近的话，其系数就越大，相应的，远离中心像素的位置那么其系数就越小。所得的结果就是，平滑了图像，而边沿和细节都不会有明显的模糊痕迹。

都不会有明显的模糊痕迹。在处理图像的实际应用中，常用的模板有 $\frac{1}{10}\begin{bmatrix}1&1&1\\1&20&1\\1&1&1\end{bmatrix}$，$\frac{1}{5}\begin{bmatrix}0&1&0\\1&10&1\\0&1&0\end{bmatrix}$，$\frac{1}{16}\begin{bmatrix}1&2&1\\2&40&2\\1&2&1\end{bmatrix}$ 从前两个矩阵模板的数值上就可以看出，此模板是根据系数和模板的中心距离的反比来确定其内部的权值的；而后一种常用的模板，则是利用了高斯模板，从而来确定内部的系数权值的。

中值滤波

此方法属于非线性的滤波，其是基于邻域运算的，其是利用了模板中的像素灰度的由降序排列后，便于查找出其序列的中间值，并且输出其中间值。假设，模板 S 的大小是 MxN，那么图像在某点的灰度值是 I（x,y），经过此方法滤波后的结果是:

$$I'(x,y)=med\{I(x-k,y-l),k,l \text{ 属于 } W\}$$

Gauss 低通滤波

此方法属于非线性的滤波。其特点:具有使低频信一号较易通过并且抑制较高频率信号的作用。高斯滤波的方程表示:

$$H(u,v)=Ae^{-D^2(x,y)/2\sigma^2} \qquad (2\text{-}5)$$

式中，H (u)表示频率域; σ 表示高斯曲线标准差: $D(u,v)$ 表示经过傅立叶变换后的某点距离远点的距离。当取。 σ =D_0时，即D_0取到截止频率，当滤波器的频率域下降到其最大值 0.607 时，利用这种方法滤波后得到的图像，能够增强图像的细节部分，在保证全部图像清晰的情况下，在局部去除不需要的噪声。

2.4.2 图像的腐蚀、填充

本论文中对于目标物体的检测和跟踪，为了确保其精确性，在对于帧图像处理时期运用了图像形态学中的腐蚀膨胀以及目标物体的检测边缘后的填充。

1)图像的腐蚀膨胀。由于各个摄像机的性能问题以及其使用的不同的环境因素，使得帧图像中会存在许多杂乱的小点，而这些点其实大部分是噪音和干扰。那么利用了形态学中的腐蚀算法是为了将这些不需要的小噪点去除。而膨胀算法目的则是将属于某个球的像素点尽可能的找到，通过图像的处理得到较为完整的球点。借此，通过坐标值求均值的方法能较为精准求取小球的球心。综上，腐蚀膨胀的算法的目的是填充遗漏的小球内部的空隙，寻求更为完整的小球;去掉多余不需要的杂乱的噪点

2)图象的填充。检测出目标物体后，利用边缘检测只能检测出边缘，为了能更好的辨识出物体，利用形态学的漫水填充算法。

(3)初始化以及更新背景图像:对于图像检测的时候，由于需要分割前景物体和背景，所以对于背景都需要先进行初始化，或者对于背景图实时进行更新。

在做图像差分之前，首先，需要确定一幅背景图，将其初始化，才能在之后的检测中和当前实时的背景图进行差分计算，这样才能得到良好效果的前景图像。通过指定法确定第一帧背景图像，即认为的指定第一张图片作为背景图像，整个检测过程，通过算法实时更新背景图像。

整个图像的初始化流程整体简述:

判断读取的是否为第一帧图像，若是则需要初始化;(2)对于 OpenCV 处理图像的格式需

要先转换为单通道灰度值;(3)将实时采集到的图像进行高斯的平滑滤波处理，去除噪点后可以得到的图像象素点为 I(x, y}: cvSmooth(pFrameMat, pFrameMat, CV GAUSSIAN,3, 0, 0);(高斯平滑处理图像函数格式)。（4）对于图像进一步的去除噪点处理，可以使用形态学滤波：图像腐蚀 cvErode(pFrImg, pFrImg, 0, 1);图像膨胀 cvDilate(pFrImg, pFrImg, fl, 1)

第一阶段:背景初始化完成后，接着就是实时更新背景图像。OpenCV 视觉库中的数学函数:cvRunningAvg，用于实时更新背景图像。其函数的原型为:void cvRunningAvg(const CvArr* image, CvArr* acc, double alpha, const CvArr*mas1}NULL)。其中函数参数的表示意义:image:表示输入的图像;acc 表示输入图像的累积;alpha:表示帧图像的权重;mask 表示可选的运算。(实际应用，mask=null) 函数中出现了图像的累积。对于背景图像利用累积差分的功态形成，

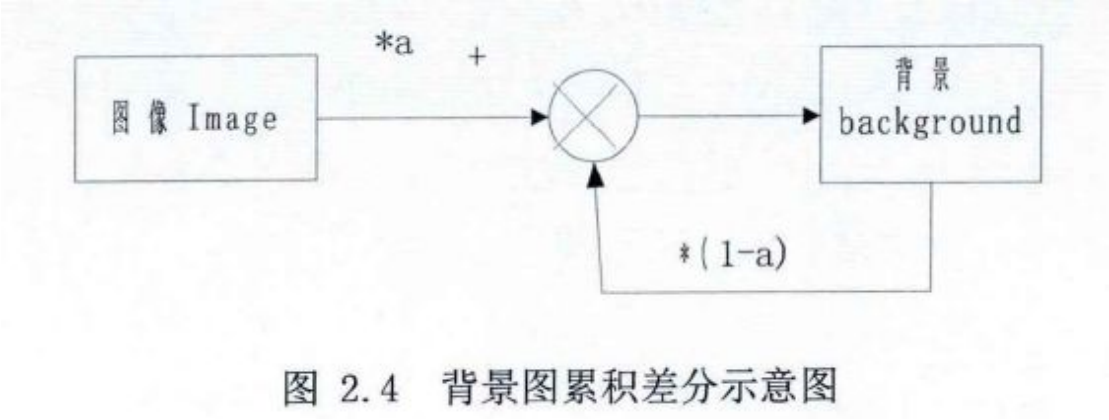


图 2.4 背景图累积差分示意图

从上图中，可以得到公式：

$$background\ (x,y) = (1-a)\times background\ (x,y) + a\times image\ (x,y) \tag{2-6}$$

式中，image表示获得的最新的图像；background表示生成的背景图像；a表示每帧图像的权值（a<1)。所以在更新图像的函数中，其图像累积计算式可表示为：

$$acc(x,y) = a\times image(x,y) + (1-a)\times acc(x,y) \tag{2-7}$$

累积差分法的使用条件为：mask(x,y)!=0。

(4)从图像中提取目标物体:首先对于采集到的图像进行分割，接着将前景物体和背景分离出来，最后阈值化得到运动物体的二值化图像。

第一阶段:二值化图像，然后分割。第二阶段:对于图像的分析处理前，进行图像的填充，保证切割前景图的完整。整个提取前景物的过程示意图如下图 2.5。

图 2.5 提取前景物的流程示意图

分析处理完图像之后，就是区分前景图像和背景图像，从复杂的背景图像中提取出目标的移动物体，在图像处理中，将这种技术称之为，图像分割技术。实际应用中较为广泛的是:阈值分割、边缘检测、区域生长

2.5.1 边缘检测

边缘检测中的边缘是指目标物体，即前景物和背景图的交界处，这些部分往往是整幅图像中变化差异最大的地方:图像的灰度值和亮度都会产生跳变，数学算法模型中就会表现出一阶导的不连续行，由此原因，其实利用图像的梯度函数就可以求得图像的边缘，在
实际应用中，被广泛应用的有 **Sobel** 算子;**Prewitt** 算子;**Roberts** 算子;**Canny** 算子。

（1）Sobel 算子

此算子的模板：3×3，如式所示：

$$\begin{bmatrix} m_0 & m_1 & m_2 \\ m_7 & (i,j) & m_3 \\ m_6 & m_5 & m_4 \end{bmatrix}$$

利用公式 $M=\sqrt{S_x^2+S_y^2}$ 计算梯度的幅值，式中，

$$s_x=(m_2+am_3+m_4)-(m_0+am_7+m_6) \tag{2-8}$$

$$s_y=(m_2+am_3+m_4)-(m_0+am_7+m_6) \tag{2-9}$$

当式子中的 a=2，可以得到 Sobel 的垂直和水平的模板分别是：

$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$，$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$。

（2）Prewitt 算子

Prewitt 算子的原理实质和 Sobel 算子相同，其差别就是在于系数取 a=1，所以水平和垂直的模板分别是：

$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$，$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$。

（3）Roberts 算子

其原理公式如下：

$$R(i,j)=|f(i,j)-f(i+1,j+1)|+|f(i+1,j)-f(i,j+1)| \tag{2-10}$$

利用卷积模板，则：$R(i,j)=|R_x|+|R_y|$

式中，$R_x=\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, R_y=\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$

（4）Canny 算子

不同于上述的几个算子，基于 Canny 算子的边缘检测需要符合 3 个判决的准则[48]：（1）信噪比。即,从图像中提取出的边缘质量随着信噪比的提高，也会随之越高。（2）单位边缘响应。我们利用检测算子的零交叉点的平均距离满足下式的要求，这样单位边缘就只存在一个响应,其理论公式为：

$$D(f)=\pi\left\{\frac{\int_{-\infty}^{+\infty}h'^2(x)dx}{\int_{-w}^{+w}h''(x)dx}\right\}^{\frac{1}{2}} \tag{2-11}$$

式子中，$h'(x)$是 $h(x)$的一阶导数，$h''(x)$是 $h(x)$的二阶导数。根据推导公式，可以明显地看出，Canny 算子由于同时运用到了一阶、二阶导数，其算法的结果更为的精准，检测图像边缘的能力和效果肯定会更好，论文中，对于运动物体的边缘检测就是利用了这个方法，其具体过程是：（1）对于获取到的图像，利用高斯滤波进行平滑处理；（2）梯度方向和幅值的计算；（3）非极大值进行抑制对于图像中所计算出的梯度的幅值；（4）利用双阈值算法进行检测边缘以及联结边缘。

2.5.2 阈值二值化的分割

对于检测物体从而获得的前景和背景的差图，利用阈值分割技术，从而将目标物体分离出图像，阈值分割[44-45]先确定图像中每个像素点的处于灰度范围的某个灰度值，将所得到的图像中各个像素的灰度值和之前确定阈值进行比较，比较之后的结果就是将所有的像素点分成了两类，一类为像素的灰度小于阈值，反之,就是另一类。其分割的过程：首先，确定某个分割的阈值 T。接着，分割图像,其值取决于之前确定的阈值，并且进行二值化图像像素。选取合适的阈值可以减少由于环境造成的光照影响，现在常用的动态阈值法有直方图法以及最大类间方差法（OTSU）[46]。

直方图法的实质是利用图像的像素邻域局部的特征。将原来的直方图变换成一个新的直方图，从而来确定一个新的阈值，对于低梯度直方图，那么两峰之间的谷就会比原来直方图上的更深，此时，图像的分割阈值 T 选取谷底的灰度值；相对的，高梯度直方图的峰是从原图的谷转换而来的，所以就用峰的灰度值取做分割的阈值 T。OTSU 阈值算法的原理为：首先，设图像的灰度的范围是 0，1，2.....L-1，设 m_i 为灰度为 i 的像素点的个数，那么像素点的总数 M 表示为：

$$\mathrm{M}=\sum_{0}^{L-1}m_i \tag{2-12}$$

假设，$p_i=\frac{m_i}{M}$ 是像素点 i 出现的概率，并且有概率的知识可得 $\sum_{i=0}^{L-1}p_i=1$。此算法通过确定某个阈值 T，将图像分成前景点 C_0={0，1，2.....t}以及背景点 C_1={t+1,t+2,.....L-1}。则 C_0 以及 C_1 的均值以及出现的概率分别表示为：

$$w_0 = \sum_{i=0}^{t} p_i = w(t) \tag{2-13}$$

$$w_1 = \sum_{i=t+1}^{L-1} p_i = 1 - w(t) \tag{2-14}$$

$$\mu_0 = \sum_{i=0}^{t} \frac{ip_i}{w_0} = \frac{\mu(t)}{w(t)} \tag{2-15}$$

$$\mu_1 = \sum_{i=0}^{t} \frac{ip_i}{w_1} = \frac{\mu_T(t) - \mu(t)}{1 - w(t)} \tag{2-16}$$

式中，$\mu_0 = \sum_{i=0}^{t} ip_i, \mu_T = \sum_{i=0}^{L-1} ip_i$。

这样，就可以得到C_0以及C_1的类方差：

$$\sigma_0^2(t) = \sum_{i=0}^{t} \frac{(i-\mu_0)^2 p_i}{w_0} \tag{2-17}$$

$$\sigma_1^2(t) = \sum_{i=t+1}^{L-1} \frac{(i-\mu_0)^2}{w_1} \tag{2-18}$$

由上式计算可以得到，

类内方差：$\sigma_w^2(t) = w_0\sigma_0^2 + w_1\sigma_1^2$ (2-20)

类间方差：$\sigma_B^2(t) = w_0(\mu_0 - \mu_T)^2 + w_1(\mu_1 - \mu_T)^2$ (2-21)

总体方差：$\sigma_T^2(t) = \sigma_w^2(t) + \sigma_B^2(t)$ (2-22)

阈值的选取对于二值化图像，分割出目标物体有很大的影响，为了选取最优阈值 t，可以通过等价判决准则：$\eta(t) = \sigma_B^2 / \sigma_w^2$，则最优阈值 t 为：$t = \arg\{\max \eta(t)\}$，借由这些准则，所得出的 t，即为前景和背景分割的最优阈值。

2.6 本文设计的检测、分析算法

2.6.1 Gauss 滤波平滑算法

在本论文的检测实验原理需要进行图像的预处理工作。对于摄像头采集到的实时的视频序列图像，为了提升检测的准确性和精度，降低后续处理分析工作的负担，就需要事先进行局部的滤波处理。OpenCV 库中平滑处理函数 cvSmooth，函数的原型为：void cvSmooth(const CvArr* src,CvArr* dst.int smoothtype.param1=3,double param2, double param3,double param4);函数中，参数 const CvArr* scr 表示输入的元图像序列或者图像象素点的矩阵；int smoothtype:表示滤波的方式，对于 OpenCV 提供了较为多种的滤波方式，共有 5 种，比如 CV_GAUSSIAN:表示高斯滤波；CV_MEDIAN：表示中值滤波方式。对于帧图像中的噪点问题，高斯滤波能有效地解决，较好的呈现出图像的清晰度。本论文对滤波问题的设计关键代码为：

```
cvSmooth( image, out, CV_GAUSSIAN, 3, 3 );
```

其中，参数CV_GAUSSIAN表示为：对图像进行核大小为3×3的高斯卷积。

2.6.2 边缘检测算法

在识别进行运动物体的前提是对图像中的目标物体进行边缘检测。可以从其算法 2-9 方程式中，看出 Canny 算法结合了一阶以及二阶的求导，相比较于其他的几种边缘检测的方式，算法上尽管较复杂些，但是其效果也是更令人满意的。所以，在本论文中，对于图像的边缘检测运用了图像的 Canny 算子，专业的图像处理 OpenCV 封装了 Canny 算子函数，简单调用就能够实现。其关键代码如下：

```
IplImage* a;
    a=m_image.GetImage();
    cvCanny(a,a,50,150,3);
    UpdateAllViews(NULL);
```

对于图像处理后，由图可知，整个处理过程，必须先将原图进行灰度的处理，接着，通过Canny算子就可以较为精准的将图片中的物体轮廓找出。

2.6.3 阈值二值化图像处理（OTSU）

本论文运用了 OTSU 阈值方法对图像进行了二值化的分割，并且为了解决画面实时变化的客观存在的原因，图像二值化的过程中还伴随着实时更新背景图像，结合两者，使得检测目标物体的运动和跟踪都有良好的效果。

OTSU 关键步骤代码如下所述:

(1)计算图片的像素值的灰度

```
    int pixelCount[GrayScale]={0};
```

```
float pixelPro[GrayScale]={0};
int i, j, pixelSum = width * height, threshold = 0;
uchar* data = (uchar*)frame->imageData;
float w0, w1, u0tmp, u1tmp, u0, u1, deltaTmp, deltaMax = 0;
```

(2)计算每个灰度级中像素的个数

```
for(i = 0; i < height; i++)
{   for(j = 0;j < width;j++)
    {    pixelCount[(int)data[i * width + j]]++;
    }
}
```

(3)计算每个灰度级的像素数目占整幅图像的比例

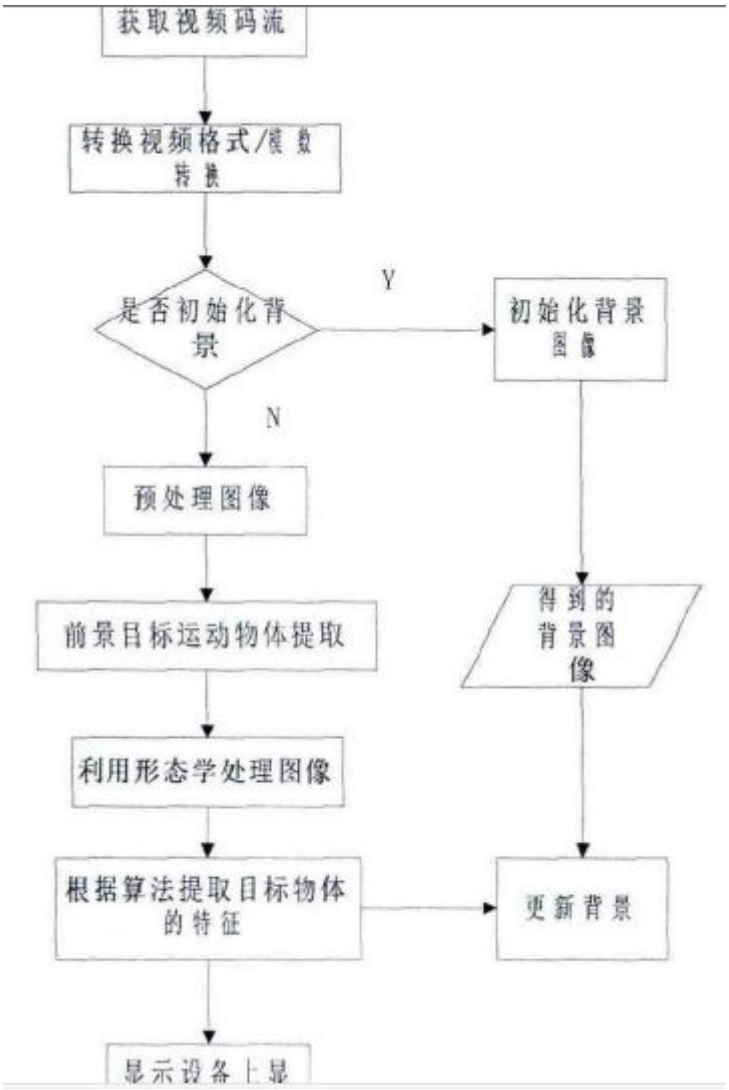
```
for(i = 0; i < GrayScale; i++)
{   pixelPro[i] = (float)pixelCount[i] / pixelSum; }
```

(4)取遍所有灰度级的范围[0,255],寻找到合适的threshold

```
for(i = 0; i < GrayScale; i++)
{   w0 = w1 = u0tmp = u1tmp = u0 = u1 = deltaTmp = 0;
    for(j = 0; j < GrayScale; j++)
    {   if(j <= i)
        {    w0 += pixelPro[j];    u0tmp += j * pixelPro[j];    }
    else
    {    w1 += pixelPro[j];    u1tmp += j * pixelPro[j];    }
    }
    u0 = u0tmp / w0;
    u1 = u1tmp / w1;
    deltaTmp = (float)(w0 *w1* pow((u0 - u1), 2)) ;
    if(deltaTmp > deltaMax)
    {   deltaMax = deltaTmp;    threshold = i;   }

    }
return threshold;
```

(5)调试并且优化程序:对于建立好的项目进行测试，根据所得出的效果，改变阈值，加入更多的算法提高检测目标物体的精准性，最后，根据环境的不同和光照的变化，对于不同的场景，都需要调试阈值等条件，使检测和跟踪的性能效果较好。



目标运动物体的检测方法

1、连续帧间差分法

连续帧间的差分法的原理是基于每个像素的运动检测方法。其借由对视频中的图像序列中相邻的每帧的图像进行差分运算， 这样就可以捕获到运动物体的轮廓。因为帧间的差分方法是利用实时的每帧的图像进行比较从而得到运动物体的轮廓，所以此方法对于环境中的动态物体具有较好的自适应的能力，但是对一于运动的物体的象素点的特征却很难完整的捕获，造成了运动物体的内部存在空洞，分离前景物和背景就会造成一定量的误差。

当监控场景中出现异常物体运动时，帧与帧之间会出现较为明显的差别，两帧相减，得到两帧图像亮度差的绝对值，判断它是否大于阈值来分析视频或图像序列的运动特性，确定图像序列中有无物体运动。图像序列逐帧的差分，相当于对图像序列进行了时域下的高通滤波。

(2)背景差分法

背景差分法的原理是实时输入的图像和背景进行比较，才能从复杂背景中分离运动物体。首先，确足一张背景图像，然后对采集到的图像与之进行差分的计算，开辟一个新的存储空间存放差分的结果图。差分图像中，某点的像素值小于等于设定的阈值，认为视频中的图像的这个像素点归为背景区域;相对的，大于此阈值，就此像素点归为运动的目标区域。根据这个原理可以看出:运动物体的像素必须和背景像素的灰度值有一定的差别。

背景差分法检测运动目标速度快，检测准确，易于实现，其关键是背景图像的获取。在实际应用中，静止背景是不易直接获得的，同时，由于背景图像的动态变化，需要通过视频序列的帧间信息来估计和恢复背景，即背景重建，所以要选择性的更新背景。

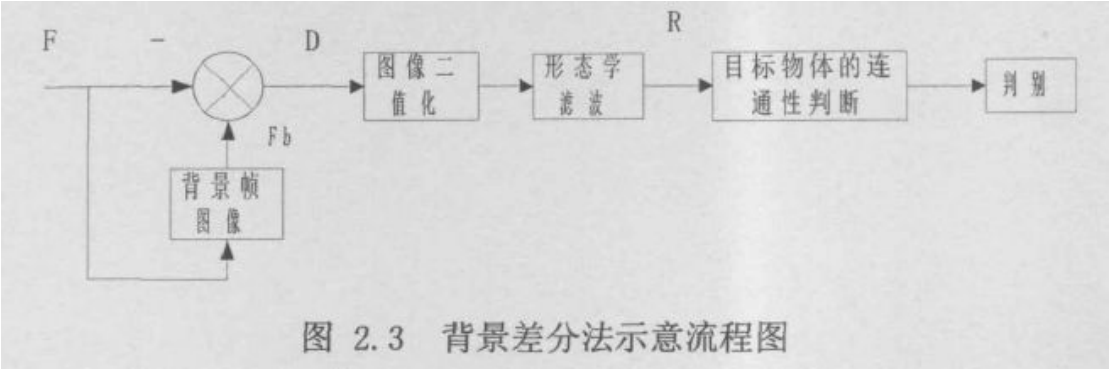


图 2.3 背景差分法示意流程图

由流程图可得,其差分公式为：

$$D(x,y)=|F(x,y)-Fb(x,y)| \qquad (2\text{-}1)$$

式中,F(x,y)是连续帧图像，Fb 是背景的连续帧图像，D（x,y)是两帧图像的帧差图像。

连通区域公式为：

$$R(x,y)=\begin{cases}0, D(x,y)>T\\ 1, D(x,y)<T\end{cases} \qquad (2\text{-}2)$$

式中，T 就是事先确定的二值化阈值。R（x,y）是连通区域的结果值。

(3)光流法

光流的概念是:图像中模型运动的速度，拟定其一种 2D 的瞬时速度场。其实 2D 速度矢量就是可见的 3D 速度矢量在平面上的一个投影，给图像中的每个像素一个速度的矢量，从而形成一个图像的运动场，每时每刻图像上的点都与 3D 立体物体上的点一一对应，即，投影关系就是 2D 转换到 3D 的两者间的关系，根据速度矢量的特征，从而分析动态情况在整个图像中，光流矢量其实是连续变化的，一点发现速度矢量出现了不连续，有断裂的情况，那么就可以得出运动物体的位置。

给图像中的每一个像素点赋予一个速度矢量，这就形成了一个图像运动场，在运动的一个特定时刻，图像上的点与三维物体上的点一一对应，这种对应关系可由投影关系得到，根据各个像素点的速度矢量特征，可以对图像进行动态分析。如果图像中没有运动物体，

则光流矢量在整个图像区域是连续变化的。当图像中有运动物体时，目标和图像背景存在相对运动，运动物体所形成的速度矢量必然和邻域背景速度矢量不同，从而检测出运动物体及位置。采用光流法进行运动物体检测的问题主要在于大多数光流法计算耗时，实时性和实用性都较差。但是光流法的优点在于光流不仅携带了运动物体的运动信息，而且还携带了有关景物三维结构的丰富信息，它能够在不知道场景的任何信息的情况下，检测出运动对象。