



“华为杯”第十四届中国研究生 数学建模竞赛

学 校 中国科学技术大学

参赛队号 K0483

队员姓名	1. 武文韬
	2. 谭婧
	3. 翟梦菲

参赛密码 _____
(由组委会填写)



“华为杯”第十四届中国研究生 数学建模竞赛

题 目 基于监控视频的前景目标提取

摘 要：

本文采用像素级背景减除模型(Background Subtraction)对监控视频中的前景目标进行了提取。

对于**问题一**，我们对静态背景下前景目标的提取问题依次构建了四个模型，分别是中位数模型，均值模型，高斯混合模型（GMM）和独立多模态背景减除模型（IMBS），并在获取前景掩码的过程中使用了**大津算法（Otsu method）**来获取合理的转换阈值。此外，我们观察到像素级的前景抽取算法会产生由离散像素点构成的前景掩码，不利于整块的前景提取。为了克服这个困难，我们为前景掩码建立了**形态学模型**，通过膨胀和腐蚀操作获取连续前景掩码。最后我们横向对比了4种模型在不同静态背景监控视频中的表现。在这一步中，**IMBS模型表现最好，中位数模型和高斯混合模型不相上下，均值模型最差。**

对于**问题二**，我们在静态背景模型的基础上为动态背景模型设计了前景提取模型。动态背景由于背景动态变化而对背景的抽取带来困难，用一般的静态模型抽取背景通常会带有大量噪点。针对这样的难题，我们**原创性独立提出了针对动态背景设计的矩成像滤波模型（Moment Imaging Filter, MIF）**。将像素视为随机变量，对其矩函数赋予颜色并进行成像，再对矩成像的颜色进行过滤，成功实现了对动态背景区域的分割。之后便能够在动态背景区域内进行专门的滤波，消减了动态背景的干扰，获得干净的图像。在这一步中，我们依然对比了不同模型的结果，在给**中位数模型和高斯混合模型加入MIF之后，效果都好于没有融合MIF的IMBS**。但是高斯混合耗时大约是中位数模型的3倍以上，所以在相对长视频处理中中位数模型配合MIF是最优选择。

对于**问题三**，摄影的过程难免会发生抖动，为了给发生抖动的视频进行前景

提取，我们通过两步操作来实现目标。首先将所有抖动的帧以第一帧为基础记性对其。我们采用了 **FAST 算法**来标记图片中的 Haar 小波特征，即图片中的拐角，再用 **MSAC 算法**进行仿射变换和扭曲变换，将后面的帧和第一帧对齐，裁剪掉不重合的部分，得到了稳定无抖动的图像。最后再调用上面的模型，对前景目标进行提取，最终成功获取了稳定的前景掩码矩阵。

对于**问题四**，需要分离出各种不同情境监控视频中的具有显著前景的帧号。我们首先利用前景掩码矩阵的范数定义了前景强度的概念。与此同时，为了更好的得到前景掩码，我们针对每种场景进行了特异性分析，得到了每种场景的难点和响应的处理办法并汇总成表。之后对前景强度的来源和性质进行了分析，归纳总结了前景在一帧中可能出现的状态，得到了前景强度的分布函数具有多峰分布的结论。我们进一步据此提出了显著的定义，即前景强度的临界值。为了获得前景强度的分布，我们原创采用了核密度估计（KDE）的方法，并且利用最大回撤的概念定义和计算了显著前景帧的前景强度最小值，并凭此选择除了前景显著帧。

对于**问题五**，对于多视角的图像处理，我们采用了对极几何的数学工具。首先采用 **SFIT 算法**对多视角中的特种对象进行识别，找到不同视角中相同物体的位置和方向，再选取纵坐标最小点为人脚与地位的焦点，得到人和标志物体的相对位置向量，在通过对极几何的方法，将一个视角中的结果和另外视角相匹配，以所有匹配特征点距离之和最短为目标确定人物位置，从而识别相同人物，并进行框选。

对于**问题六**，我们选了行军这种群体行为进行了建模，我们分析了行军在时间和空间上的周期特征，提出了两条判断视频行为行军的标准，即行军视频的前景掩码应当在固定行数或者列数的时间-空间图上呈现线性排布，并且，其时间-空间图上的频谱应当带宽小于一个阈值。我们采用快速傅里叶变换（FFT）算法来分析行军行为的前景掩码的时间-空间频谱性质。最后，我们从央视网站上下载了中华人民共和国 20 周年阅兵仪式的视频，将两条准则引用于视频类型判断，验证了我们的想法。

关键词： 背景减除模型、GMM、IMBS、MIF、核密度估计（KDE）

目录

1、问题重述.....	2
1.1 问题背景.....	2
1.2 问题的提出.....	2
2、模型假设与符号说明.....	3
2.1 模型假设.....	3
2.2 符号说明.....	3
3、问题一模型建立与求解.....	4
3.1 问题描述及分析.....	4
3.2 模型建立.....	5
3.3 问题一求解与分析.....	11
3.4 问题一的改进.....	15
4、问题二模型建立与求解.....	16
4.1 问题描述及分析.....	16
4.2 模型建立.....	18
4.3 问题二求解与分析.....	21
5、问题三模型建立与求解.....	24
5.1 问题描述及分析.....	24
5.2 模型建立.....	24
5.3 问题三求解与分析.....	31
6、问题四模型建立与求解.....	32
6.1 问题描述及分析.....	32
6.2 模型建立.....	32
6.3 问题四求解与分析.....	37
7、问题五模型建立与求解.....	43
7.1 问题描述及分析.....	43
7.2 模型建立.....	44
7.3 问题五求解与分析.....	46
8、问题六模型建立与求解.....	47
8.1 问题描述及分析.....	47
8.2 模型建立.....	47
8.3 问题六求解与分析.....	49
9、模型的评价与改进建议.....	51
9.1 模型评价.....	51
9.2 模型与方法的改进建议.....	52
参考文献.....	53
附录.....	54

1、问题重述

1.1 问题背景

随着平安城市的大力建设，各地的监控摄像头数量成指数式增长，提供了海量的监控视频数据。面对爆发增长的监控数据，有效快速地从中提取目标信息尤为重要，而传统的人工监控已经不能满足这一需求，亟需建立基于人工智能的可对视频进行自动分析和识别的智能监控系统。随着计算机技术的不断进步，这一智能监控系统的建立已经成为可能，其中如何在复杂、多变、动态的背景下提取前景目标则是一个非常重要而基础的问题。

前景分离技术往往能够对一般的视频处理任务提供有效的辅助。以筛选与跟踪夜晚时罪犯这一应用为例：若能够预先提取视频前景目标，判断出哪些视频并未包含移动前景目标，并事先从公安人员的辨识范围中排除；而对于剩下包含了移动目标的视频，只需辨识排除了背景干扰的纯粹前景，对比度显著，肉眼更易辨识。因此，这一技术已被广泛应用于视频目标追踪，城市交通检测，长时场景监测，视频动作捕捉，视频压缩等应用中。

近年来，研究人员大都采用以下三种方法进行前景目标提取：一是帧间差分法，将相邻两帧或多帧进行差分运算，去除相同背景点得到目标信息；二是背景减除法，拟得有效背景模型，将帧图像与其做差分运算得到目标信息；三是光流法，通过计算相邻两帧像素点的运动信息提取前景目标。

表 1-1 各种检测算法的比较

算法名称	算法有点	算法缺点
帧间差分法	能够适应环境变化、计算相对容易、稳定性好	容易产生空洞现象、静止目标没法检测
背景减除法	速度快、能检测静止目标、准确性高	对光照比较敏感
光流法	适用于运动环境中的检测、能够直接计算运动信息	由于计算量较大所以实时性比较弱

1.2 问题的提出

本题要解决的问题是提取视频前景目标。监控视频主要由固定位置监控摄像头拍摄，此类视频的特点是相对于前景目标背景结构较稳定，变化幅度较小，可利用这一特点进行模型与算法设计。

在给定以上信息的前提下，本题要求解决下列问题：

问题一：对于不包含动态背景、摄像头稳定拍摄时间大约 5 秒的监控视频，构造提取前景目标（如人、车、动物等）的数学模型，并对该模型设计有效的求解方法，从而实现从原视频帧中分离前景目标的效果，其中前景显示为白色，背景显示为黑色。

问题二：对于对包含动态背景信息（如树叶摇动，水波动，喷泉变化，窗帘晃动）的监控视频，设计有效的前景目标提取方案，使得动态背景与真实前景分离。

问题三：对于不稳定拍摄的监控视频，当监控摄像头发生晃动或偏移，视频

也会发生短暂的抖动现象（该类视频变换在短时间内可近似视为一种线性仿射变换，如旋转、平移、尺度变化等）。设计有效的目标提取方案，从而将典型的发生晃动的监控视频前景与晃动的静态背景分离。

问题四：附件 3 提供了 8 组视频，利用上文所构造的建模方法，从每组视频中选出包含显著前景目标的视频帧标号，并将其在建模论文正文中独立成段表示，同时注明前景目标是出现于哪一个视频（如 Campus 视频）的哪些帧（如 241-250，421-432 帧）。

问题五：对于在室内同一时间从不同角度拍摄的近似同一地点的多个监控视频，在充分考虑并利用多个角度视频的前景之间（或背景之间）相关性信息后，设计有效的目标提取方案，使同时拍摄的多个监控视频中前景与背景分离。

问题六：对于有无人群短时聚集、人群惊慌逃散、群体规律性变化（如跳舞、列队排练等）、物体爆炸、建筑物倒塌等异常事件，利用所获取前景目标信息自动判断监控视频中异常事件，要求尝试多种异常事件类型并设计相应的事件检测方案（可通过网络下载包含各种事件的监控视频进行算法验证）。

2、模型假设与符号说明

2.1 模型假设

假设一：视频可以被分解为多帧的图片，每个视频等价于一个 3 维数据 $X \in R^{(w \times h \times t)}$ ，其中 w, h 代表视频帧画面的长，宽， t 代表视频帧的帧数。矩阵中的每个元素为 0 到 255 之间的某一个值，越接近 0，像素越黑暗；越接近 255，像素越明亮，本题仅考虑黑白图片构成的视频。

假设二：相对于前景目标，背景结构较稳定，变化幅度较小，像素点波动较为稳定。

假设三：前景是常见的具有固定形态且可以被监控设备拍摄的物质，其中包括部分虽然不具有固定形态，但分布相对稳定的液体或者气体。

假设四：前景的出现具有局域特征，也就是说，前景在视频中是有界的。

2.2 符号说明

符号	说明
w	视频帧画面的宽度
h	视频帧画面的高度
T	视频的总帧数长度
F_{ijt}	原视频第 t 帧画面中第 i 行第 j 列对应像素点的灰度值
BG_{ijt}	原视频画面提取的第 i 行第 j 列背景对应像素点的灰度值
FG_{ijt}	原视频第 t 帧第 i 行第 j 列对应前景像素点灰度值
B_{ijt}	第 t 帧画面中第 i 行第 j 列背景掩码
G_{ijt}	第 t 帧画面中第 i 行第 j 列前景掩码
FG'	前景矩阵
BG_{ijt}^D	动态背景第 t 帧第 i 行第 j 列对应前景像素点灰度值

BG_{ij}^S	静态背景第 t 帧第 i 行第 j 列对应前景像素点灰度值
L	前景强度

3、问题一模型建立与求解

3.1 问题描述及分析

一般而言，对于运动目标的检测是从视频序列中，提取出每一帧的运动前景，从而将包含运动信息的部分和视频背景部分相互分离。因监控视频背景环境的差异，视频可以分为静态背景和动态背景。对于摄像头位置固定，且拍摄是保持稳定，画面中物体运动仅由监控目标在场景内运动所致，即为静态背景。而当摄像头背景发生规律性变化（如水波、喷泉等）则视为动态背景^[1]。

本题给出了五段大约 5 秒的监控视频，视频特点为不包含动态背景且摄像头稳定不抖动。由于光流法效果不佳，我们对这一问题的处理采用背景减除法。简单地说就是对视频序列进行建模，用一定的模型筛选出各帧画面背景画面，并通过原图像与背景图像做差得到前景掩码，在前景像素点上填充原画面灰度值即可得到前景画面。

在实际处理的过程中，我们采用了均值模型、中位数模型、GMM 模型、IMBS 模型对五段视频进行建模，分别输出每一种方法下对应的前景画面并进行横向对比分析。通过分析我们发现，在背景减除法下，前景图中的噪点较多，在前景物体上存在孔洞，导致实际还原的前景图像不完整，因此我们结合了形态学的处理方法，通过膨胀和腐蚀完成平滑图像，去除噪声点和填充孔洞的工作，最后得到了静态场景下较优的前景视频。

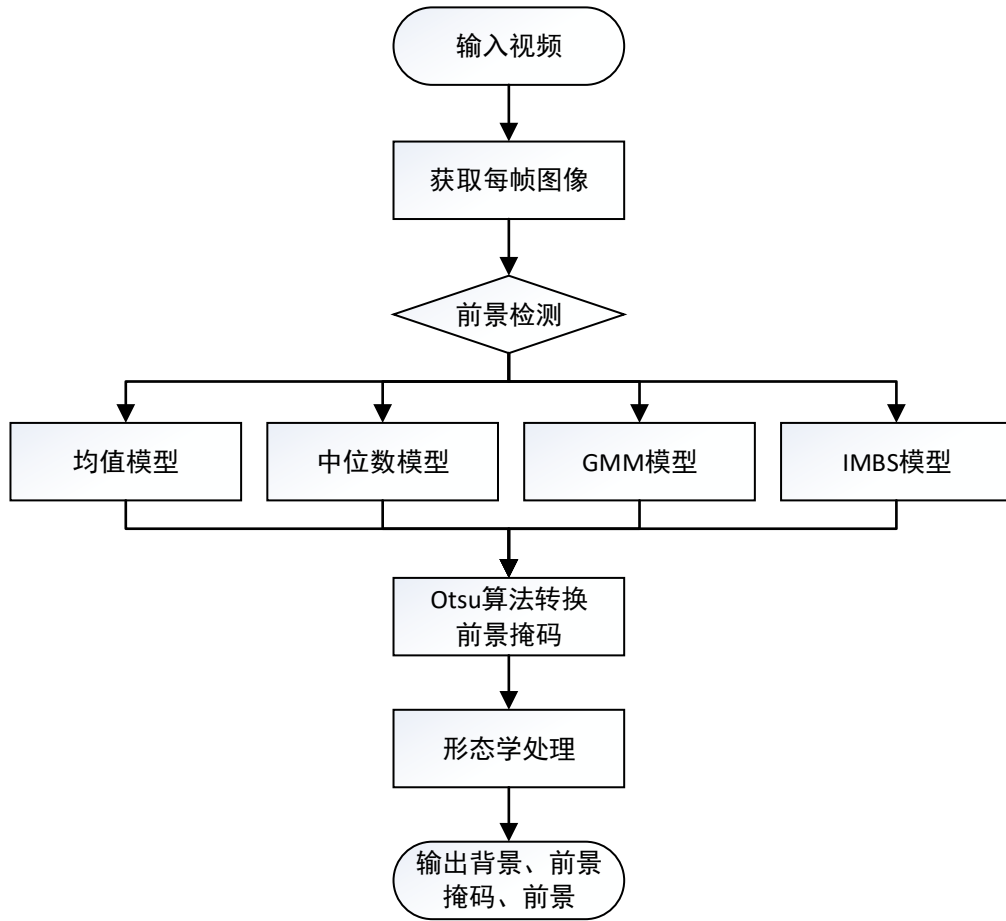


图 3-1：问题一解决思路流程图

3.2 模型建立

假设视频可分离为 $w \times h$ 大小共 T 帧画面，位于第 t 帧的画面第 i 行第 j 列的像素点的灰度值为 F_{ijt} 。当通过一定的算法模型得到了背景画面第 t 帧第 i 行第 j 列像素点的灰度值 BG_{ijt} 后，对原像素点灰度值与背景画面灰度值做差，即使用背景减除法得到一个像素点的灰度值 FG_{ijt} ，可以认为 FG 为一个近似的前景画面。

$$FG_{ijt} = F_{ijt} - BG_{ijt}$$

在获得前景画面后，我们通过最大类间距方差算法（Otsu），将灰度前景图像素点矩阵 FG 计算阈值分割的方法，将其转换为只含有 0-1 数字的前景掩码矩阵 G 。

$$G_{ij} = \begin{cases} 0 & FG_{ijt} < T_H \\ 1 & FG_{ijt} \geq T_H \end{cases}$$

在得到前景掩码矩阵 G 后，将其对应元素 G_{ijt} （像素点）与原画面对应位置像素点灰度值矩阵 F_{ijt} 即可得到扣除背景后的前景帧灰度画面。

$$FG' = G \cdot F$$

3.2.1 模型一：中位数算法

(1) 对于一段稳定拍摄的视频，将视频的每一帧画面作为一个图像存储，假设视频共有 T 帧，每一帧图像对应有 w 行 h 列像素点，由于本题视频均为黑白图片构成的视频，因此每个像素点仅对应一个灰度值，假设 F_{ij}^t ($i=1,2,\dots,w; j=1,2,\dots,h; t=1,2,\dots,T$) 表示视频上第 t 帧上第 i 行第 j 列像素点的灰度值，灰度值为 0 到 255 之间的某一个值，越接近 0，像素越黑暗；越接近 255，像素越明亮。

我们记对于固定第 i 行第 j 列的像素点，在 $F_{ij1}, F_{ij2}, F_{ij3}, \dots, F_{ijT}$ 的取值大部分取值是近似相等的，由中位数的稳定性，记序列第 i 行第 j 列的中位数的灰度值为 f_{ij}^{Med} ，该中位数可近似为背景图像上对应像素点的灰度值。因此由 $(F_{11}^{(Med)}, F_{12}^{(Med)}, \dots, F_{ij}^{(Med)}, \dots, F_{wh}^{(Med)})$ ($i=1,2,\dots,w; j=1,2,\dots,h$) 即可确定视频固定的背景图像 $BG_{ij}^{(Med)}$ 。

(2) 在确定背景图像后，将原视频第 t 帧图像上每一点灰度值与背景图像灰度值做差，可以得到第 t 帧一个差值图像 FG_{ijt} ，即对于地 t 帧第 i 行第 j 列像素点有：

$$FG_{ijt} = |F_{ijt} - BG_{ij}^{(Med)}|$$

其中 $FG_{ij}^{(t)}$ 为所求的前景图像，由于在 t 帧没有运动目标（前景）的背景图，这些像素点灰度值大部分都是接近于 0 的，只有运动目标对应的像素点对应前景图像的灰度值大于 0，即非黑色，这样就可以把含有运动目标的前景图像提取出来。

(3) 对于提取出的共 T 帧前景图像，组合成视频即可得到完整的前景画面 FG 。

3.2.2 模型二：均值算法

(1) 与中位数相同，均值算法中将第 i 行第 j 列的像素点在 $t=1,2,\dots,T$ 的时间的灰度值平均值记为 f_{ij}^{Mid} ，该均值可近似为背景图像上对应像素点的灰度值。因此由 $(F_{11}^{(Mid)}, F_{12}^{(Mid)}, \dots, F_{ij}^{(Mid)}, \dots, F_{wh}^{(Mid)})$ ($i=1,2,\dots,w; j=1,2,\dots,h$) 即可确定视频固定的背景图像 $BG_{ij}^{(Mid)}$ 。

(2) 同样，在确定背景图像后，将原视频第 t 帧图像上每一点灰度值与背景图像灰度值做差，可以得到第 t 帧一个差值图像 FG_{ijt} ，即对于地 t 帧第 i 行第 j 列像素点有：

$$FG_{ijt} = |F_{ijt} - BG_{ij}^{(Mid)}|$$

其中 $FG_{ij}^{(t)}$ 为所求的前景图像，由于在 t 帧没有运动目标（前景）的背景图，这些像素点灰度值大部分都是接近于 0 的，只有运动目标对应的像素点对应前景图像的灰度值大于 0，即非黑色，这样就可以把含有运动目标的前景图像提取出来。

(3) 对提取出的共 T 帧前景图像，组合成视频得到完整的前景画面 FG 。

3.2.3 模型三：混合高斯模型

混合高斯模型的基本原理是，构建 K 个高斯分布对视频序列图像中的每个像素点进行加权描述。若每个像素点的值用变量 X_t 代表，其概率密度函数

$$P(X_t) = \sum_{i=1}^K \frac{\omega_{i,t}}{(2\pi)^{\frac{1}{2}} \left| \Sigma_{i,j} \right|^{\frac{1}{2}}} e^{\frac{-1}{2} (X_t - u_{i,t})^T \Sigma_{ij}^{-1} (X_t - u_{i,t})}$$

在上式中：

$\omega_{i,t}$ ：t 时间点第 i 个权重，满足条件 $\sum_{i=1}^K \omega_{i,t} = 1$

X_t ：t 时间点这个像素的灰度值；

$u_{i,t}$ ：t 时间点第 i 个高斯分布的均值；

$\sigma_{i,t}^2$ ：t 时间点第 i 个高斯分布的方差；

(1) 模型初始化

取视频序列第一帧的每个像素值作为该像素 k 个高斯分布的平均值，并初设较大的方差（一般取值 30）和相似权重。

(2) 模型更新

模型更新首先进行匹配，每获得一帧图像，将该帧每个像素点与该像素点已存在的 k 个高斯分布一一进行匹配，对于匹配的像素点进行更新，否则不予更新。匹配的范围是 $|X_t - u_{i,t}| \leq 2.5\sigma_{i,t}^2$ 。

对匹配的第 i 个高斯分布的参数更新公式如下：

$$\begin{aligned} u_{i,t} &= (1 - \rho_{i,t})u_{i,t-1} + \rho_{i,t}X_t \\ \sigma_{i,t}^2 &= (1 - \rho_{i,t})\sigma_{i,t-1}^2 + \rho_{i,t}(X_t - u_{i,t})^T(X_t - u_{i,t}) \end{aligned}$$

其中： α 为 0.005，是自定义的模型学习率； $\rho_{i,t}$ 为自定义的参数学习率，且

$$\rho_{i,t} \approx \frac{\alpha}{\omega_{i,t}}。$$

若未发现与 X_t 互相搭配的高斯分布，则此时新建以 X_t 作为均值的高斯分布替代之前最小权重的高斯分布，配以较大的方差和较小的权重。按下式更新权重：

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

若第 i 个高斯分布与 X_t 匹配， $M_{i,t}$ 为 1，否则 $M_{i,t}$ 为 0。

(3) 背景描述及前景提取

将每个像素的所有高斯分布按 $\omega_{i,t} / \sigma_{i,t}$ 的比值由大到小排序，取数值 B，

$$B = \arg \min_b \left(\sum_{k=1}^b \omega_k > T \right) (0.5 < T < 1)$$

根据排序结果取 B 个高斯分布作为背景像素描述。然后重新做上述匹配检验，若前 B 个高斯分布中有一个能与 X_t 匹配，则该点是背景点，否则是前景点 [2]。

3.2.4 模型四：独立多模态背景建模方法（IMBS）

独立多模态背景建模方法（Independent Multimodal Background Subtraction, IMBS），是根据递推方法进行背景更新建模的方法，没有事先设定的概率模型，也没有设定估计参数，对于不确定分布的数据具有更好的适应性，因此 IMBS 属于非参数背景建模。IMBS 的优点主要有二点：一是使用了实时聚类进行建模，可以避免存储历史像素值，可以实时计算完成背景检测工作；二是采用了更新机制，从而可以对于背景场景变化具有较好的检测效果，即可以用于动态背景的识别。但独立多模态背景建模同样存在一定的问题，首先是在每个背景场景需要一定时间进行背景建模的初始化工作，如果背景场景存在快速变化，背景检测的结果可能产生一定数量的误判^[3]。

IMBS 的建模原理是通过对每个像素点建立颜色或灰度值分布模型，并使用在线算法进行更新。下面讲从背景建模结构、背景建模工作流程和背景模型更新策略三方面对其进行说明。

（1）IMBS 背景模型结构

随时间 t 的推移，可以得到视频序列某一像素点关于灰度值的时间序列，而背景建模方法主要是从背景灰度值时间序列中提取出现概率高，稳定时间长的部分作为背景像素值，而混合高斯模型（GMM）是通过几个加权的高斯分布函数来反应历史颜色序列中颜色的分布情况，出现概率越高，其权重更大且方差更小，则就具有越大的优先级。但对于较复杂的场景，例如树叶晃动，水面波动等情况，背景像素点分布可能难以用多高斯模型描述。使用一些预先定义好的灰度分布可能不能很好的适应一些复杂的背景场景。

为了处理复杂的背景场景，IMBS 采用在线更新的聚类方法，每一个像素点使用一组基本元组（tuple）的集合来进行表述，其中的每个元组也带有相应的权重。对于每个基本元组，起定义为 $T = \langle r, g, b, d \rangle$ ，其中 r 、 g 、 b 代表 RGB 颜色值， d 是与该元组相关联的样本个数，通过把每个颜色值保存为一个元组而不是分别处理每个颜色，我们可以利用不同颜色值之间的相关性。

令 $B(i, j) = \{T_i\}$ 为像素点 (i, j) 对应的元组集合，其中 T_i 代表每一个元组，对每一个新的样本点可以按照以下流程更新当前像素点的背景模型：

Step1. 对于每一个 $T = \langle r, g, b, d \rangle \in B(i, j)$ $T = \langle r, g, b, d \rangle$ ，根据以下判断新样本和该元组的距离 r ：

$$r = \max\{|I_r - r|, |I_g - g|, |I_b - b|\}$$

Step2. 对于有匹配元组 T ，使得距离 $r < A$ ，其中 A 为判定阈值，则使用下面公式更新匹配元组 T 元组：

$$\begin{aligned} r' &= \frac{r \cdot d + I_r}{d + 1} \\ g' &= \frac{g \cdot d + I_g}{d + 1} \\ b' &= \frac{b \cdot d + I_b}{d + 1} \\ T_i &= \langle r', g', b', d + 1 \rangle \end{aligned}$$

Step3. 如果没有匹配的元组，则使用新样本点创建新元组 $T_n = \langle I_r, I_g, I_b, 1 \rangle$ 更新到元组集合 $B(i, j)$ 中。

当建立好背景模型之后，对于元组集合中每一个元组 T ，计算元组与新颜色值的最大距离 $r = \max\{|I_r - r|, |I_g - g|, |I_b - b|\}$ ，如果有距离 $r < R$ ，则为背景像素，否则为前景像素， R 为事先设定的判定阈值。

(2) IMBS 背景模型更新策略

针对背景模型的选择问题，IMBS 提出了如下方法：首先令 $F(i, j)$ 为对新像素 $I(i, j)$ 计算出背景掩码，有 $F(i, j) = 0$ 时像素点为背景， $F(i, j) = 1$ 时像素点为前景。在更新背景像素点模型时，如果有 $F(i, j) = 1$ 且 $I(i, j)$ 所更新的元组为 T ，则标记该元组为前景元组。在计算前景掩码时，如果新像素 $I(i, j)$ 与一个前景元组相关联，则把当前像素点标记成一个潜在前景点。

通过这种解决方案，可以辨别场景中没有移动的前景区域，如果一个像素被连续判定为潜在前景超过一定时间，则这个像素就被归类为背景像素，相应元组也归类为背景元组。

(3) IMBS 背景模型工作流程

IMBS 算法工作时，需要设定以下参数：采样周期 P ，样本序列长度 N ，元组最小样本数 D 。在 IMBS 算法工作时，每过采样周期 P ，采集当前帧 $I(i, j)$ 就被作为一个场景样本 $S_k, 1 \leq k \leq N$ ；当采集 N 个样本后，用这 N 个样本建立当前一个时间片段的背景模型，可知 IMBS 算法需要经过 $R = P \times N$ 的时间才能建立一个背景模型。

为了同时完成更新背景模型和前景检测的工作，在 IMBS 算法内部采用了两个背景模型 B_{old} 和 B_{new} 。其中 B_{old} 为保存的上一个周期生成的背景模型，用来完成对新输入图像帧的前景检测工作；而 B_{new} 代表当前正在生成的背景模型，使用新输入图像帧进行训练。当 B_{new} 完成建立后，用它替换 B_{old} 并新建一个 B_{new} 。

IMBS 背景模型具体工作流程如下：

Step1. 在背景模型进行初始化时，需要设置背景模型 B_{old} 和 B_{new} 为空。

Step2. 通过计算当前帧与样本帧的时间差判断当前帧是否为样本帧，当时间差大于 P 时为样本帧。

Step3. 当 B_{new} 构建完成时替换背景模型，把 B_{new} 保存到 B_{old} 中。具体过程为：首先清空 B_{old} ，然后把 B_{new} 中满足 $d > D$ 的元组添加到 B_{old} 中，最后清空 B_{new} 。

Step4. 使用背景模型 B_{old} 计算前景掩码 $F(i, j)$ ，使用上文方法判断 $I(i, j)$ 是否属于背景模型，如果属于背景模型，则前景掩码 $F(i, j) = 0$ ，否则 $F(i, j) = 1$ 。

Step5. 设置前景标记是针对训练模型 B_{new} 进行设置。

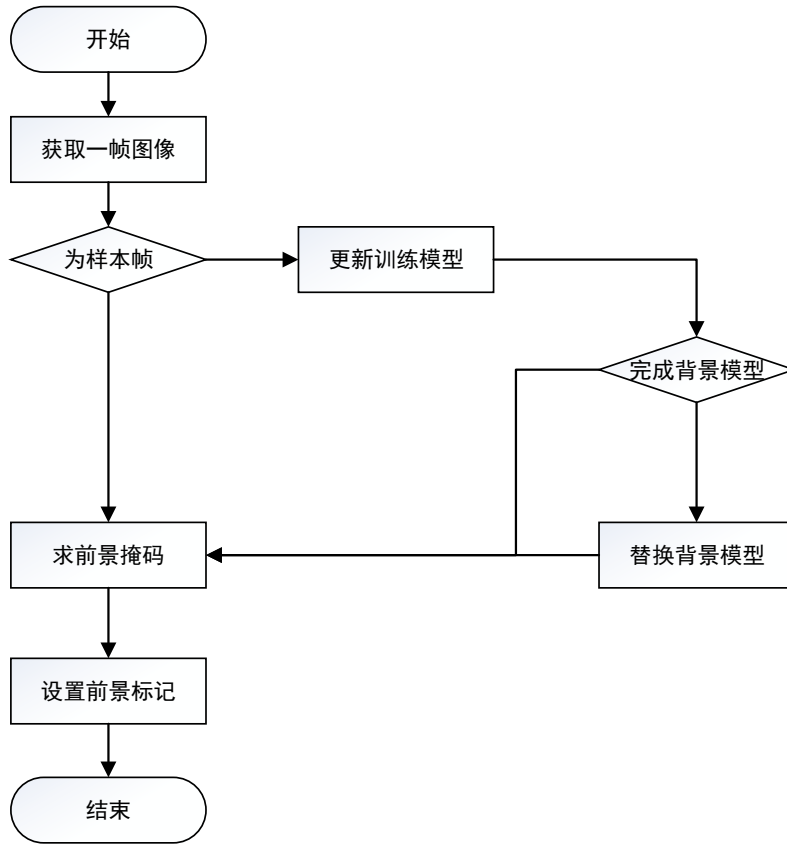


图 3-2: IMBS 模型流程图

3.2.5 最大类间方差法 (Otsu)

大类间方差法是由津展之在 1979 年提出的基于判别分析理论推导出的自动的无参数无监督的阈值分割方法^[9]。具体原理如下：

设数字图像灰度级 ($G=1,2, \dots, L$) 处在灰度级 i 所有象素个数用 f_i 表示，总的像素用 N 表示，则有 $N = f_1 + f_2 + \dots + f_L = \sum_{i=1}^L f_i$ ，而 P_i 表示图像中灰度级为 i 出现的概率，定义 $P_i = \frac{f_i}{N}$ ，则 $P_i \geq 0$ ， $\sum_{i=1}^L P_i = 1$ 。用阈值 t 将像素点按照灰度级的大小分为 C_0 和 C_1 两类， $C_0 = \{1, 2, \dots, T\}$ ， $C_1 = \{t+1, t+2, \dots, L\}$ ，则两类出现的概率分布为：

$$\omega_0 = P_r(C_0) = \sum_{i=1}^L p_i = \omega(t)$$

$$\omega_1 = P_r(C_1) = \sum_{i=1}^L p_i = 1 - \omega(t)$$

各类灰度均值为：

$$u_0 = \sum_{i=1}^L i \times P_r(i | C_0) = \sum_{i=1}^L \frac{i p_i}{\omega_0}$$

$$u_0 = \sum_{i=1}^L \frac{i p_i}{\omega_1} = \frac{u_r - u(t)}{1 - \omega_0}$$

其中 $u_r = u(L) = \sum_{i=1}^L iP_i, u(t) = \sum_{i=1}^t iP_i, \omega(t) = \sum_{i=1}^t P_i$, 未评价阈值的优劣引入评判函数:

$$\lambda = \frac{\sigma_B^2}{\sigma_w^2}, \quad \kappa = \frac{\sigma_r^2}{\sigma_w^2}, \quad \eta = \frac{\sigma_B^2}{\sigma_r^2}$$

其中 σ_w^2 、 σ_r^2 、 σ_B^2 分别为组内方差、总体方差和组间方差。我们的目标表示为选择最优的阈值 t 使 λ , κ , η 三个判决函数达到最大, 由关系式可推导出组间方差可作为分类性能高低的判决函数:

$$\sigma_B^2(t) = \frac{[u_r \omega(t) - u(t)]^2}{\omega(t)[1 - \omega(t)]}$$

最佳阈值 t^* 为:

$$\sigma_B^2(t^*) = \max_{1 \leq t \leq L} \{\sigma_B^2(t)\}$$

3.3 问题一求解与分析

3.3.1 问题一图像输出结果

下面, 我们通过四种模型, 分别对五组视频中的关键帧进行背景的提取, 进一步, 通过 Otsu 阈值得到背景掩码, 同时填充原图像中对应灰度值得到前景图像, 各组视频在四类模型下的输出结果如下:

表 3-1: airport 静态视频前景及背景提取结果








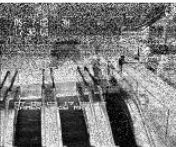




画面 模型	原图	背景	前景掩码	前景
均值模型				
中位数模型				
GMM模型				
IMBS模型				

表 3-2: hall 静态视频前景及背景提取结果













画面 模型	原图	背景	前景掩码	前景
均值模型				
中位数模型				
GMM模型				
IMBS模型				

表 3-3: office 静态视频前景及背景提取结果

















画面 模型	原图	背景	前景掩码	前景
均值模型				
中位数模型				
GMM模型				
IMBS模型				

表 3-4: pedestrian 静态视频前景及背景提取结果



















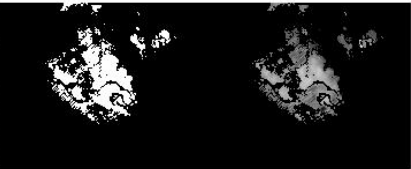






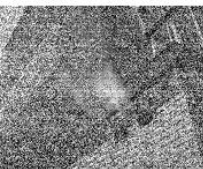
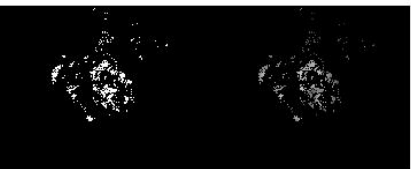





画面 模型	原图	背景	前景掩码	前景
均值模型				
中位数模型				
GMM模型				
IMBS模型				

表 3-5: smoke 静态视频前景及背景提取结果

画面 模型	原图	背景	前景掩码	前景
均值模型				
中位数模型				
GMM模型				
IMBS模型				

3.3.2 问题一结果的讨论

对于静态视频的结果，我们有如下分析：

(1) 在 **airport** 视频中，GMM 模型前景结果噪点较多，均值模型、中位数模型、IMBS 模型效果均好于 GMM 模型，IMBS 背景提取不纯净，均值与中位数模型效果最好。

(2) 在 **hall** 视频中，GMM 模型背景中噪点较多，但前景提取任务轮廓较为清晰；IMBS 模型背景不纯净，同时前景中任务轮廓不准确；而均值模型和中位数模型在背景识别和前景的图像上表现较好，综合来看，中位数模型效果最优。

(3) 在 **office** 视频中，同样存在上述问题，GMM 背景存在噪点、IMBS 背景不纯净，从前景效果来看，四种方法在前景部分均准确提取到了人物轮廓，总体来看中位数模型和 IMBS 模型提取较好。

(4) 在 **pedestrian** 视频中，此场景任务较多，识别较为困难，中位数模型的背景提取相对纯净，而在识别效果上，中位数与 IMBS 模型中均提取到了画面左侧身着深色衣服的行人，提取效果较好。

(5) 在 **smoke** 视频中，烟雾是持续存在的，因此除 GMM 噪点较多外，背景无明显差异，从前景的提取来看，均值模型和中位数模型提取烟雾较为完整。

总体来看，中位数视频效果最优，背景提取纯净且前景较为完整；IMBS 模型在背景图像上效果不佳，但前景图像较好；均值模型与中位数模型效果类似，但在前景提取中效果略差于中位数模型；最后 GMM 模型在背景中噪点过多，但

同样顺利提取到了关键人物的轮廓，相比于其他三种方法效果稍差。

3.4 问题一的改进

3.4.1 静态背景存在的问题

在问题一的结果当中，可以发现不论是 IMBS、GMM、中位数、均值模型中，在提取前景后，前景成像方面均有一些噪点和黑洞，这是由于移动目标部分的像素点与背景颜色较为接近，难以通过以上的算法提取出，为得到更为精准的前景图像，同时不遗漏关键信息，下面我们进行形态学处理的改进，以填补移动前景轮廓内所有的像素点。

3.4.2 形态学分析

在形态学分析中，所有像素状态都是通过对输入图像的相应像素及邻域使用一定的规则进行确定。首先定义两种填充图像的规则——膨胀与腐蚀^[4]。

首先，膨胀（Dilation）是指，输出像素值是输入图像相应像素邻域内所有像素的最大值。在二进制图像中，如果像素点相邻像素的任何像素值为 1，那么对应的输出像素值为 1；而在腐蚀（erosion）操作中，输出像素值是输入图像相应像素邻域内所有像素的最小值。在二进制图像中，如果任何一个像素点相邻像素的任何像素值为 0，那么对应的输出像素值为 0。

在膨胀和腐蚀执行之前，我们选取一个结构元素（如圆形、方形）。结构元素通常比待处理的图像小，二维平面上由 0-1 矩阵组成，结构元素的原点指定了图像中需要处理的像素范围，结构元素中数值为 1 的点决定结构元素的邻域像素在进行膨胀或腐蚀操作时是否需要参与计算。对于图像边缘的元素，由结构元素定义的邻域将会有一部分位于图像边界之外。

为有效处理边界像素，进行形态学运算的函数通常都会给出超出图像、未指定数值的像素指定一个数值，这样就类似于函数给图像填充了额外的行和列。膨胀即使对于结构元素中超出图像部分的像素值定义为 1，即显示为白色，而对于腐蚀则恰恰相反，像素值定义为 0，即为黑色。

为得到有效边界，进行如下的图像处理过程：

Step 1.膨胀：超出图像边界的像素值定义为该数据类型允许的最大值，对于二进制图像，这些像素值设置为 1；对于灰度图像，unit8 类型的最小值也为 255。

Step 2. 腐蚀：超出图像边界的像素值定义为该数据类型允许的最小值，对于二进制图像，这些像素值设置为 0；对于灰度图像，unit8 类型的最小值也为 0。

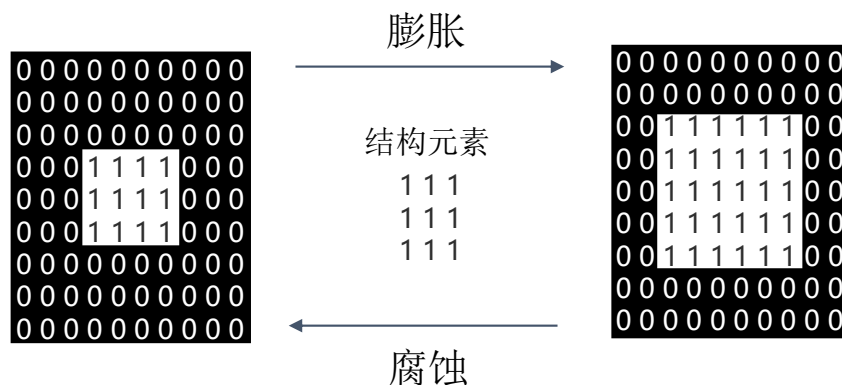


图 3-3：形态学膨胀与腐蚀示意图

3.4.3 形态学处理结果

我们通过对前景掩码进行先膨胀后腐蚀两个过程，在填充图像后，消减成块图像边缘，从而得到相比之前更加完整的前景画面，以下面三个视频为例，hall 与 office 的画面当中人物的轮廓内的图像基本都填充在前景图像当中，相比于处理前的效果，人物更加完整、清晰。而在 smoke 视频中，画面中间烟雾以及右上角的一小团烟雾均完整体现在前景画面当中，因此，形态学处理可有效填充前景中移动目标主体，解决主体显示不完整的问题。

表 3-6：静态视频前景帧形态学处理结果

画面 视频	原图	前景掩码	前景掩码 (形态学处理)	前景
hall				
office				
smoke				

4、问题二模型建立与求解

4.1 问题描述及分析

上一问我们对于静态背景构造了目标前景的提取的数学模型，而现实场景中，监控视频容易受到现实场景中诸多因素的影响，例如光线强度的变化、背景中景物的变化、运动前景目标阴影和前景目标间的相互遮盖、前景目标的颜色形状、运动速度都会导致背景运动，例如树叶摇动，水波动，喷泉变化，窗帘晃动等情景在问题一的算法中无法区分动态部分是否属于前景。

本问的核心问题是运用特定的算法有效对运动目标检测以及对动态背景的识别，我们首先利用第一问的四种算法对动态场景进行前景提取，鉴于结果不理想，在动态背景下识别存在误差，我们原创独立提出了一种基于样本矩判定动态背景的方法 MIF，可以显著提升动态场景的前景识别问题。

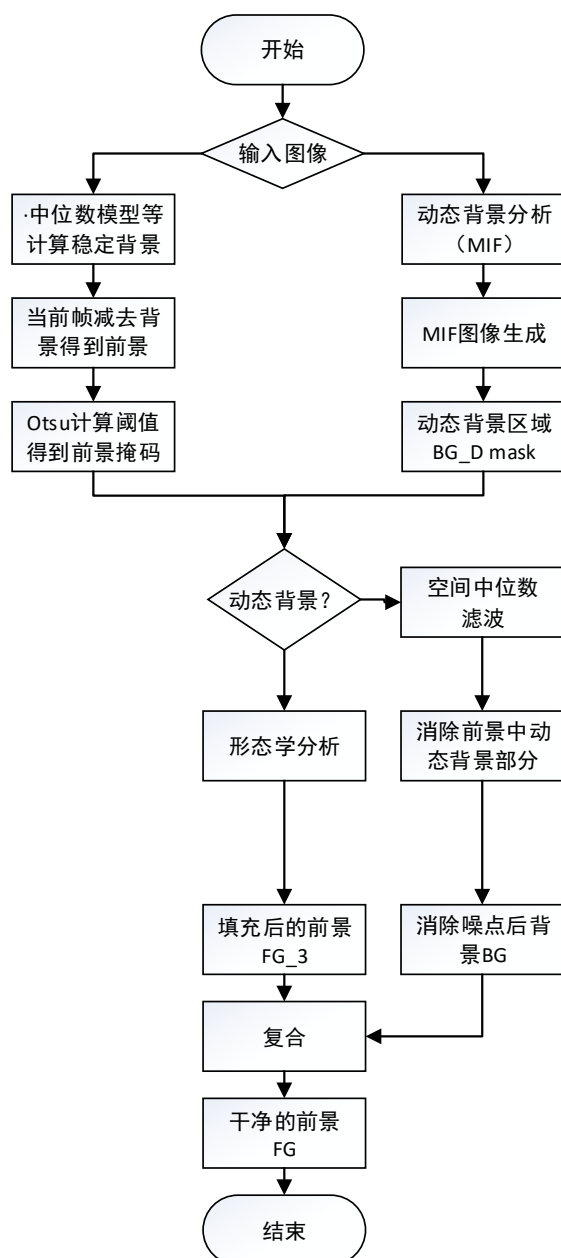


图 4-1：问题二解决方案流程图

4.2 模型建立

4.2.1 中位数模型、GMM 模型与 IMBS 模型

在对动态场景进行前景识别的过程中，均值模型表现不如另外三类模型，且均值模型和中位数模型结构类似，因此在本问中，初始前景的筛选主要通过三类模型：中位数模型、GMM 模型和 IMBS 模型。

三类具体模型构成与第一问相同，不再赘述。

4.2.2 矩成像滤波（MIF）分离动态背景模型

在筛选初始前景的过程中，动态背景对于前景的提取存在较大的干扰。为了有效提取含动态背景的视频监控前景信息，我们建立了 MIF 模型进行动态场景的分离。

首先明确动态背景的特征。本题中区分的动态背景主要包括树叶摇动，水波动，喷泉变化，窗帘晃动等，此类视频的特征是动态背景在时间维度变化较为规律，而前景（人物、车辆）在时间上变化不规律，基于这一假设，我们进行了如下运算过程。

（1）MIF 成像过程

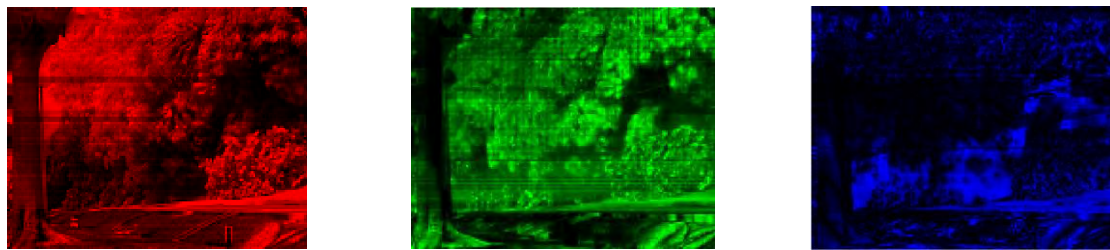
对于一个含有 T 帧，画面像素点为 $w \times h$ ，且像素点色彩为灰度的视频，我们为了尽量提取有效信息，反映出所有 T 帧像素点的变化情况，选择三个指标作为判断是否为动态背景的依据：中位数 $MIFR_{ij}$ 、方差 $MIFG_{ij}$ 、偏度 $MIFB_{ij}$ ，并将每个像素点的三个指标作为一组 RGB 数组 $[MIFR_{ij}, MIFG_{ij}, MIFB_{ij}]$ 显示在 MIF 图像上，其中：

$$\begin{aligned} MIFR_{ij} &= F_{jit}^{(Med)} \\ MIFG_{ij} &= \frac{1}{T-1} \sum_{t=1}^T (F_{ijt} - \bar{F}_{ij})^2 \\ MIFB_{ij} &= \frac{1}{T-1} \sum_{t=1}^T [(F_{ijt} - \bar{F}_{ij})^3 / MIFG_{ij}^{\frac{3}{2}}] \end{aligned}$$

（2）MIF 图像色彩分析

\bar{F}_{ij} 为第 i 行第 j 列像素点在 $[1, T]$ 时刻上灰度值的均值，其反映出的是该像素点稳定的灰度信息，我们将均值 \bar{F}_{ij} 作为 MIF 图像中 $MIFR_{ij}$ 红色通道数值，红色通道的图像越红的部分反映像素点变化。而标准差 $\left[\sum_{t=1}^T (F_{ijt} - \bar{F}_{ij})^2 / (T-1) \right]^{1/2}$ 反映的是该像素点在均值附近的波动情况，因此对于监控视频图像而言，仅有无前景经过且光线、环境无变化的静态图像像素点波动值较小，可以通过方差排除纯静态背景，因此将方差作为 MIF 图像 $MIFG_{ij}$ 绿色通道数值。最后，偏度作为分布是否对称的判据，也构成了动态背景判断的重要判据之一。

我们将 RGB 三色通道进行合成，于是便得到了 MIF 的真彩图像。这样的图像能够使我们容易分辨动态背景区域和其他区域，也方便我们在色彩空间中进行约束和滤波的操作。



M IF-R

M IF-G

M IF-B

图 4-2 像素点序列一阶矩均值，二阶矩方差，三阶矩偏度构成的 RGB 三色通道

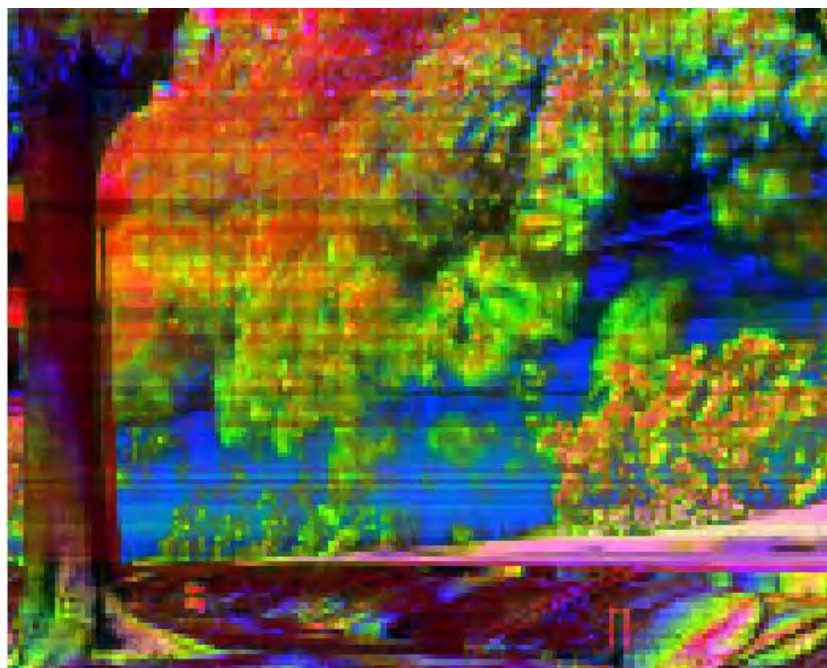
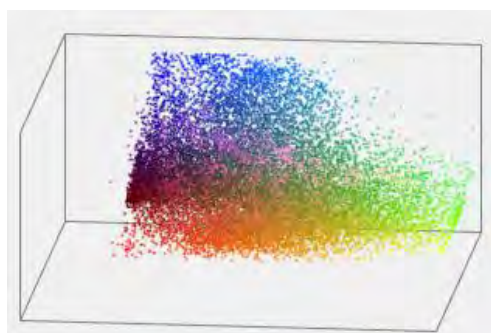
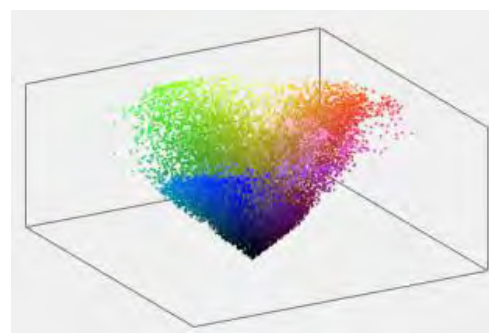


图 4-3 MIF 成像所得到的矩函数的 RGB 真彩图像



(a) GRB Space



(b) HSV Space

图 4-4 不同颜色空间下 MIF 成像的色彩分布

其中左图(a)为 RGB 空间，右图(b)为 HSV 空间

仔细观察 MIF 的成像真彩图像，其中树叶的部分沉陷红色和绿色的结合，呈

现棕红色和黄色，但是没有蓝色叠加，这就表示，这些点均值大，波动率大，但是偏度较小。而相比之下，道路背景发蓝，说明均值不高，波动率也不大，但是有偏，说明这里原来是暗色静态背景，并且可能有前景路过。

(3) 动态背景像素点的判别

仔细思考可以发现，对于动态背景像素点以及有前景经过的像素点的灰度值随时间变化的曲线大致如下图所示，其中动态背景像素点例如水波、电梯等是规律变化，因此在概率分布上表现为对称分布，而有前景经过的像素点，由于前景经过的时间是随机的，且每个经过的前景物体不尽相同，因此在灰度值在时间上的分布可能是有偏的，用偏度衡量像素点在时间轴上变化的第三个维度，作为我们区分波动像素点是否属于动态背景的依据。

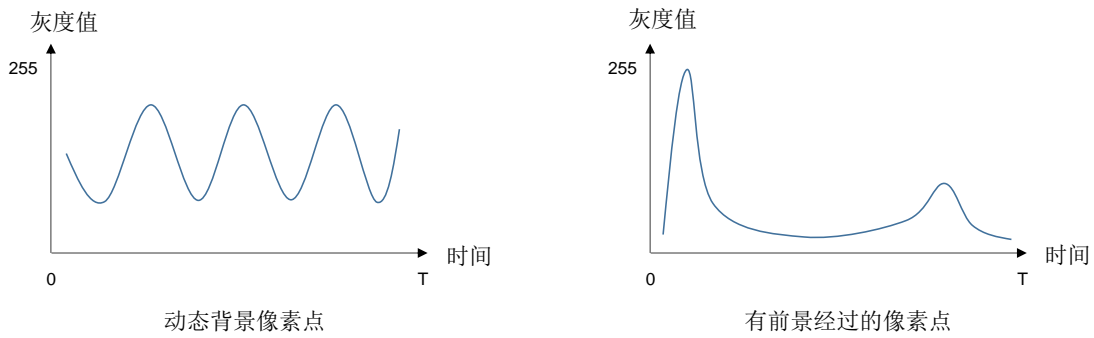


图 4-5: 不同像素点灰度值变化图

由于动态背景点波动大，方差 $MIFG_{ij}$ 较大，同时偏度 $MIFB_{ij}$ 要在一定范围之内，我们给定阈值 T_{GH} 、 T_{BH} ，对于方差 $MIFG_{ij}$ 大于 T_{GH} 及偏度的绝对值 $|MIFB_{ij}|$ 小于阈值 T_{BH} 的像素点判定为动态背景像素点，由 h 行 w 列 B_{ij}^D 组成的 (0-1) 矩阵即为动态背景图像 MIF-mask。

$$B_{ij}^D = \begin{cases} 1 & MIFG_{ij} > T_{GH} \ \&\& \ |MIFB_{ij}| < T_{BH} \\ 0 & \text{其他} \end{cases}$$

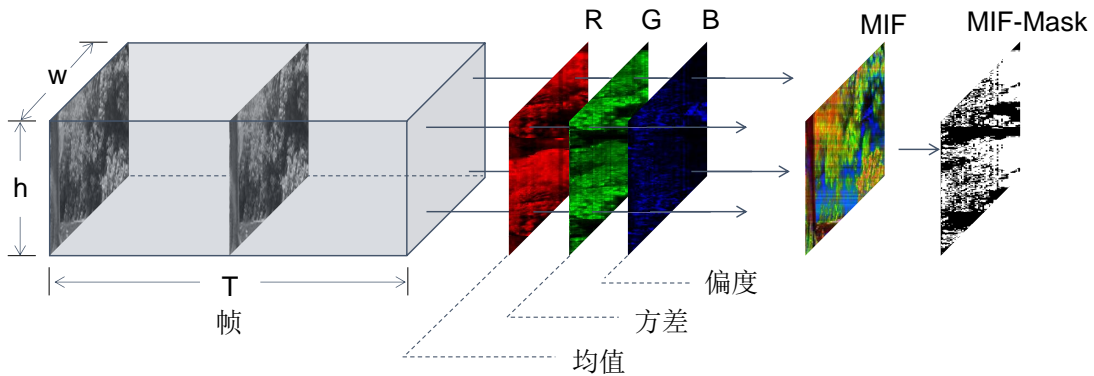


图 4-6: MIF 模型示意图



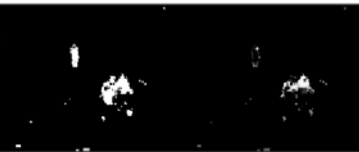





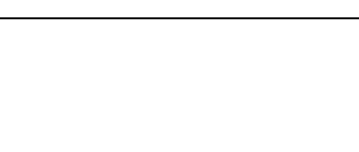
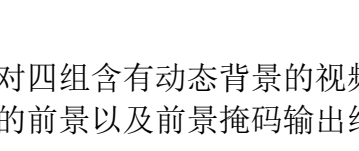
用颜色空间的角度来考虑，我们这里索要挑选的动态背景是 RGB 空间中的

一个矩形区域，在 HUV 空间中的一个锥形区域。

作为例子下面我们通过两个含有动态背景的视频（airport、campus）进行前景提取与 MIF 前景提取。可以看到在 MIF 处理前，airport 视频通过中位数模型得到的前景掩码中有明显的电梯轮廓，而在 campus 视频中，背景有较多树叶波动造成的噪点。

通过 MIF 模型识别动态背景范围，可以看到 airport 视频的电梯部分以及 campus 视频中的树叶部分的提取范围较为准确，进一步，通过空间中位数滤波，可以将动态背景部分进一步降噪，与经形态学提取的前景图像相结合可以得到经过动态背景滤波后的前景掩码，可以看到与 MIF 处理之前相比，其动态部分的噪点有显著改善，动态背景对于前景的影响有明显减弱。

表 4-1：MIF 模型处理前后前景图像一览

视频	画面	原图	前景掩码 (无MIF处理)	动态背景范围	动态背景滤波的 前景掩码	前景 (MIF处理)
airport						
campus						

4.3 问题二求解与分析

4.3.1 问题二图像输出结果

下面，我们通过三种模型与 MIF 模型结合，分别对四组含有动态背景的视频中的关键帧进行背景的提取。经过 MIF 模型处理后的前景以及前景掩码输出结果如下：

表 4-2: campus 动态视频前景及背景提取结果 (MIF 处理)






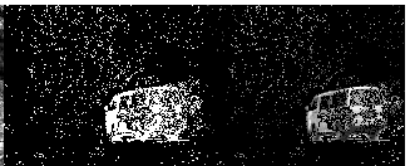
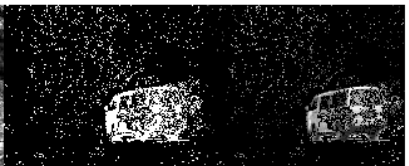
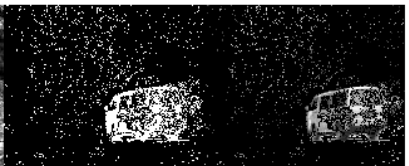




画面 模型	原图	背景	前景掩码	前景
MIF-中位数模型				
MIF-GMM模型				
MIF-IMBS模型				

表 4-3: curtain 动态视频前景及背景提取结果 (MIF 处理)













画面 模型	原图	背景	前景掩码	前景
MIF-中位数模型				
MIF-GMM模型				
MIF-IMBS模型				

表 4-4: fountain 动态视频前景及背景提取结果 (MIF 处理)

















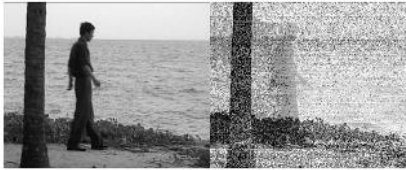







画面 模型	原图	背景	前景掩码	前景
MIF-中位数模型				
MIF-GMM模型				
MIF-IMBS模型				

表 4-5: watersurface 动态视频前景及背景提取结果 (MIF 处理)

画面 模型	原图	背景	前景掩码	前景
MIF-中位数模型				
MIF-GMM模型				
MIF-IMBS模型				

4.3.1 问题二结果的讨论

对于含有动态背景视频的结果，我们有如下分析：

(1) 在 campus 视频中，三个模型下的背景均较为清晰，从前景提取效果来

看，MIF 与中位数模型结合的提取效果最好。

(2) 在 curtain 视频中，从背景提取来看，中位数模型最优，在前景的提取上，三个模型下噪均比较少，且当前帧人物的轮廓较清晰，综合来看中位数与 IMBS 效果较优。

(3) 在 fountain 视频中，背景的提取较为纯净，与前两段视频不同的是，IMBS 的前景效果最优，人物轮廓清晰，而中位数模型以及 MGG 模型中出现了少量噪点。

(4) 在 watersurface 视频中，场景较为简单，由于画面中任务腿部与部分景色重合，因此三个画面中都有部分无法提取，从最终效果来看，IMBS 与中位数模型均无噪点，前景提取比较干净。

总的来说，经过 MIF 模型去动态背景后，结合第一问较优的算法以及形态学处理，MIF-中位数模型以及 MIF-IMBS 模型已经可以比较清晰准确地提取出前景形态信息，同时过滤掉动态背景带来的影响。

5、问题三模型建立与求解

5.1 问题描述及分析

问题三主要解决的是监控摄像头发生晃动或偏移时，视频画面发生短暂的抖动现象。首先，我们在明确摄像机模型与图像运动在不同场景下坐标系对应的转换关系，接下来，我们利用 Video Stabilization 算法寻找视频各帧中的角点，角点是视频各帧画面中最能反映出图像信息的关键点，通过点之间的变换矩阵，调整各帧在平面上的位置，对抖动视频中每一帧图像进行旋转、缩放以及位移变换，利用变换矩阵即可得到平稳不抖动背景的图像，拼接即可得到视频。

在对图像的处理之后，得到稳定的画面，即可通过前两问中的模型进行前景目标的提取，在此分析基础上做如下假设：

(1) 摄像机抖动，画面发生偏移、变形等。

(2) 图像中存在动态背景，需考虑并消除动态背景的影响。

5.2 模型建立

在监控视频中，存在着各种形式的运动，其中一些是无法避免的，如第二问中动态背景的变化，而另外一些例如由于摄像头发生晃动或偏移，视频画面发生抖动现象，可以通过一定的方法消除摄像头晃动对画面的影响。因此在研究抖动问题之前，首先要明确摄像机模型与图像运动模型之间的关系，在此基础上才能进行后续的运动参数估计以及抖动补偿。

5.2.1 不同情况下视频抖动的转换关系

假定任意一种摄像机的运动形式都可以用旋转、平移和变焦表示。设 $(X, Y, Z)^T$ 是摄像机坐标系下某物体的一点， $(X, Y, Z)^T$ 与 $(X', Y', Z')^T$ 分别为同一点在不同时刻的三维坐标，相应的图像平面上的点的坐标为 (x, y) 和 (x', y') ，如果三维场景中的物体的运动为位移、旋转和线性变化，则存在如下关系：

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

而摄像机的成像系统将三维场景变换成二维灰度图像，这种变换是从一个三维空间到二维空间的一个映射：

$$(X, Y, Z)^T \rightarrow (x, y)^T$$

其中 $(X, Y, Z)^T$ 是摄像机坐标系下物品上的一点，利用透视投影法或者正交投影法将三维空间映射到二维空间，我们就可以获得摄像机坐标系与图像坐标系的对应关系。

在定义了摄像机坐标系与图像坐标系的关系后，首先选取合适的图像运动模型来描述摄像机的运动信息，常见的模型有三类：一是平移模型，二是 Similarity 模型^[13]，三是 Affine 变换模型。

(1) 在平移模型中，摄像机的运动是指图像在二维空间上发生了 x 方向和 y 方向的位移，其模型可表示为：

$$p' = p + t = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} tx \\ ty \end{pmatrix}$$

其中 p 表示坐标，tx 和 ty 分别表示 p' 相对于 p 在 x 轴与 y 轴的偏移量。

(2) Similarity 模型中，摄像机本身除平移运动外还可能发生旋转运动，例如车辆行驶过程中因路面不平导致摄像头左右摇晃，从而使得图像旋转，利用缩放因子 s 对对变焦运动进行描述，因此图像发生平移、旋转、变焦运动时，可采用 Similarity 运动模型。

$$p' = s \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} p + \begin{pmatrix} tx \\ ty \end{pmatrix}$$

其中 θ 为旋转角度，s 为缩放因子。


(3) Affine 变换模型是一种六参数线性变换模型，即具有平行线变换成平行线，有限点映射到有限点的一般特性。具体表现可以是各个方向尺度变换系数一致的均匀尺度变换或变换系数不一致的非均匀尺度变换及剪切变换等，可描述平移运动，旋转运动及小范围的缩放和形变。


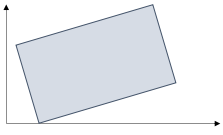

$$p' = s \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} p + \begin{pmatrix} tx \\ ty \end{pmatrix}$$

其中 a1 和 a4 是缩放参数，a2 和 a3 是旋转参数。

总体而言，参数个数越多，对摄像机运动的描述就越精确，相应计算复杂度也就越高。例如平移模型计算复杂度较低，但对连续帧的摄像机运动参数进行叠加累计误差较为严重，而更多参数会使复杂度更大，不同运动模型的示例如下：

:5-1: 视频抖动坐标变化示意图

运动模型	坐标变换关系	变换参数	图示
原始图像	-	-	

平移	$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} tx \\ ty \end{pmatrix}$	位移： tx 、 ty	
Similarity	$\begin{pmatrix} x' \\ y' \end{pmatrix} = s \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} p + \begin{pmatrix} tx \\ ty \end{pmatrix}$	缩放： s 旋转： θ 位移： tx 、 ty	
Affine 变换	$\begin{pmatrix} x' \\ y' \end{pmatrix} = s \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} p + \begin{pmatrix} tx \\ ty \end{pmatrix}$	缩放： a_1 、 a_4 旋转： a_2 、 a_3 位移： tx 、 ty	

5.2.2 Video Stabilization 算法

对于摄像机视频抖动问题，解决思路通常是提取出具有显著特征的角点，跟踪图像角点并将其作为锚点消除视频带来的抖动现象。下面所使用的 VS (Video Stabilization)^[14] 算法是一种在没有任何先验信息的情况下，自动搜索视频序列中的背景图像，并从第一帧视频中提取角点并在其他帧角点发生偏移的情况下矫正摄像机画面。

VS 算法分为两个步骤：首先，确定视频序列中所有相邻帧的放射图像变换，确定相邻帧图像之间的对应关系。第二，我们使用变换方法扭曲视频画面，再将所有帧合成为一个新的视频即可消除画面的抖动。

下面，我们尝试读取视频的相邻两帧，并生成一个红-蓝叠加的图像，可以看到两帧之间像素点有较大的垂直和水平方向的平移，因此可以对此视频使用 VS 算法。



图 5-1：car6 视频相邻两帧画面图



图 5-2: car6 视频相邻两帧合成图

(1) 角点的提取。

由于我们的目标是对相邻两帧通过仿射变换做转换，因此需要确定相邻两帧之间的一种一一对应的关系，这种对应关系需要通过提取视频全局角点，而后确定两帧间的变换关系。

关于角点的提取，我们需提取尽可能在所有帧上都可找到的对应点，同时可以突出图像的突出特征，我们使用 FAST 算法提取角点，两帧的截图如下：



图 5-3: car6 视频相邻两帧角点识别图

下面简单介绍 FAST 算法。FAST 是由 Edward Rosten 和 Tom Drummond 在 2006 年^[10]最早提出的用于识别图像中的角点的算法，全称是 Features From Accelerated Segment Test，其中 FAST 角点定义为若某像素点与其周围领域内足够多的像素点处于不同的区域，则该像素点可能为角点，也就是某些属性与众不同，考虑灰度图像，即若该点的灰度值比其周围领域内足够多的像素点的灰度值大或者小，则该点可能为角点。角点是图像所承载的信息的主要特征，理想情况下应在不同帧对应的图像上重复出现^[11]。

FAST 的算法步骤如下：

Step 1. 从图片中选取一个像素 P ，下面我们将判断它是否是一个角点。我们首先把它的亮度值设为 I_p 。

Step 2. 设定一个合适的阈值 t 。

Step 3. 考虑以该像素点为中心的一个半径等于 3 像素的离散化的 Bresenham 圆，这个圆的边界上有 16 个像素。

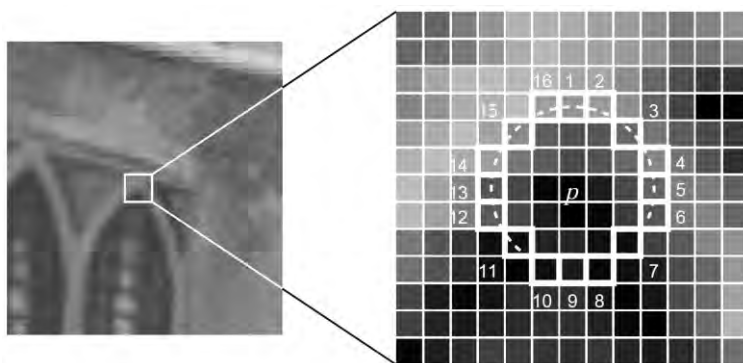


图 5-4: FAST 角点示意图

Step 4. 如果在这个大小为 16 个像素的圆上有 n 个连续的像素点，它们的像素值要么都比 I_p+t 大，要么都比 I_p-t 小，那么它就是一个角点，如上图白色虚线所示。 n 的值可以设置为 12 或者 9。

对于图像中的每一个点，我们都要去遍历其邻域圆上的 16 个点的像素，效率较低。为解决这一问题，FAST 算法中使用一种高效的测试（high-speed test）来快速排除一大部分非角点的像素，例如在上图中仅检查 1、9、5、13 四个位置的像素，首先检查位置 1 和 9 的像素，若它们都比阈值暗或比阈值亮再比较位置 5 和 13，若 p 是一个角点，超过四分之三圆的部分应该满足判断条件：上述像素点中至少 3 个角点必须大于 I_p+t 或者小于 I_p-t 。如果不满足，那么 p 不可能是一个角点。在进行所有点的初步筛选后，在候选角点中再做角点完整检验，即检验圆上所有点。

上述算法的缺点在于，在设置 $n < 12$ 时不能使用快速筛选的方法过滤非角点的点，同时，检测出的角点并不是最优的，因为它的效率取决于问题的排序与角点的分布，我们可以通过机器学习焦点分类器做改进。

改进的 FAST^[5]算法如下：

Step 1. 运用 FAST 角点检测算法来获取测试图片集上的所有角点。

Step 2. 对于每个角点，我们把它邻域圆上的 16 个点存储下来保存在一个 vector 内，处理所有步骤 2 中得到的角点，并把它们存储在 P 中。

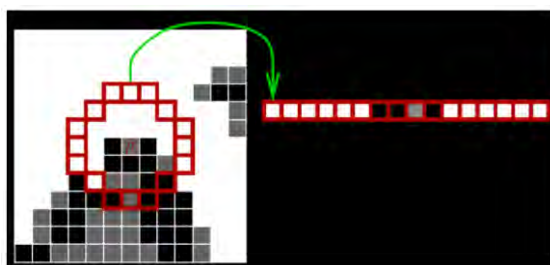


图 5-5: FASK 算法存储示意图

Step 3. 对于图像上的点 p ，其邻域圆上位置为 $x, x \in \{1, 2, \dots, 16\}$ 的点表示为 $p \rightarrow x$ ，可用下面的公式将点分为三类：

$$S_{p \rightarrow x} = \begin{cases} d & I_{p \rightarrow x} < I_p - t & (\text{darker}) \\ s & I_p + t \leq I_{p \rightarrow x} < I_p - t & (\text{similar}) \\ b & I_p + t \leq I_{p \rightarrow x} & (\text{brighter}) \end{cases}$$

Step 4. 设 P 为训练图像集中所有像素点的集合，我们任意 16 个位置中的一个位置 x ，可以把集合 P 分为三个部分 P_d 、 P_s 和 P_b ，其定义如下：

$$P_d = \{p \in P : S_{p \rightarrow x} = d\}$$

$$P_s = \{p \in P : S_{p \rightarrow x} = s\}$$

$$P_b = \{p \in P : S_{p \rightarrow x} = b\}$$

换句话说，对于任意给定的位置 x ，它都可以把所有图像中的点分为三类，第一类 P_d 包括了所有位置 x 处的像素在阈值 t 下暗于中心像素，第二类 P_s 包括了所有位置 x 处的像素在阈值 t 下近似于中心像素，第三类 P_b 包括了所有位置 x 处的像素在阈值 t 下亮于中心像素。

Step 5. 定义一个变量 K_p ，若 p 是一个角点 K_p 为真，否则为假，使用决策树分类器检查并递归所有子集直到 K_p 的熵为 0，同时被创建的决策树用于其他图片的 FAST 检测。

(2) 确定点与点之间的对应关系。

对于每个点，我们以该点为中心的快速视网膜关键点（FREAK），在点与点之间测量汉明距离，将帧 A 与帧 B 中的点相匹配。由于没有唯一性约束，因此来自 B 帧的点可以对应 A 帧上的多个点。

接下来匹配当前和之前帧中找到的特征，由于 FREAK 描述符是二进制的，因此可使用汉明距离来查找响应点，下图是上面两幅图角点的叠加，A 帧中的角点为红色圆圈，B 帧角点为绿色十字星，两帧之间对应的点用黄色实线相连反映对应关系，这些判断大多是正确的，但也存在一些异常值。



图 5-6: car6 视频相邻两帧角点对应关系图

(3) 从带有噪声的对应点中估计转换关系。

Step2 中获取的对应点很多是不正确的，但我们仍然可以使用 MSAC 方法到处两个图像之间的鲁邦估计，该算法是 RANSAC 算法的一个变体，在给定一组点对应关系时，可以搜索有效内在对应关系。从上述关系中导出变换，使得来自第一组点与来自第二组的内在关系匹配最紧密，该仿射与 Affine 变换较为类似，采用以下形式：

$$H_{sRt} = \begin{pmatrix} a1 & a3 & 0 \\ a2 & a4 & 0 \\ tx & ty & 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = s \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} p + \begin{pmatrix} tx \\ ty \end{pmatrix}$$

其中，参数 a 定义变换的缩放-旋转-平移，参数 t 是平移参数，用于图像的扭曲，使两帧的相同角点变换到相同位置。以上面两帧为例，将 A 帧与 B 帧的通过转换后的画面通过红蓝图相叠加，得到如下的结果：

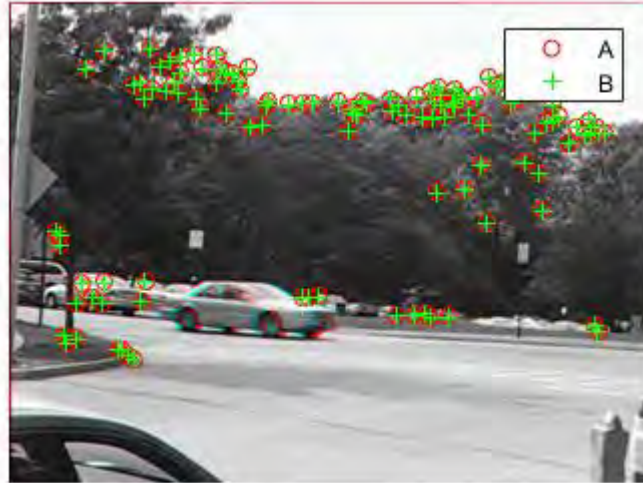


图 5-7: car6 视频相邻两帧角点叠加红蓝图

从结果来看，两帧的大部分内容叠加效果较为吻合，红色的圈与绿色的十字星点相重合，且背景中红蓝区域较少，绝大多数相匹配的红蓝图显示为黑白色。事实上，摄像的帧率越高，两帧之间运动物体的形态变化差异越小，假设背景物体没有发生变化，实际上变换捕捉到的是相机的运动，相邻两帧之间摄像机的运动足够小或者视频帧的频率越高，视频去抖动效果越好。

(4) 转换的近似和平滑。

给定一组视频 F_t ， $t = 0, 1, 2, \dots$ ，我们是用上述过程估计所有帧 F_t 与其下一帧 F_{t+1} 之间的仿射变换 H_t ，因此 t 时刻相对于第一帧的累计失真变换将是前面所有相邻两帧变换的乘积：

$$H_{cumulative,t} = \prod_{i=0}^{t-1} H_i$$

我们使用上述变换的留个参数，为了保证数值的简单性和稳定性，将矩阵重

新拟合为更加简单的缩放-旋转-平移变换，与上述变换矩阵相比只有四个参数，新的变换矩阵的形式为：

$$H = \begin{pmatrix} s \times \cos \theta & s \times (-\sin \theta) & 0 \\ s \times \sin \theta & s \times \cos \theta & 0 \\ tx & ty & 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = s \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} p + \begin{pmatrix} tx \\ ty \end{pmatrix}$$

我们将上述变换的矩阵 H 与缩放-旋转-平移等效矩阵 H_{sRt} 拟合，从而显示出转换过程，为显示出转换错误最小，我们用两个变换重新投影 B 帧，同时显示红蓝组合图像，当图像看起来是黑白的时候，像素之间的差异可以近似忽略不计。

与此同时，我们取相邻帧图像均值，可以看到去抖动前后的图像变化，右图是使用 Video Stabilization 算法后的图像，可以看到图像背景较为稳定。



图 5-8: car6 视频算法前后效果示意图









(5) 在所有变换运行完毕后，将变换后的帧画面组合成视频，接下来，与上一问相同，对去抖动的视频按照第二问的方法做前景的提取。

5.3 问题三求解与分析

下面我们对两组视频经变换后的去抖动画面做前景提取分析，与第二问结果类似，在 MIF 处理前，背景图像噪点较多，而经过动态范围识别以及空间中位数滤波处理后，得到的前景图像可以比较好的反映当前画面前景物体。

事实上，在 people2 视频中，当前帧的识别效果没有很理想，主要是由于该帧上人物变化动作速度较慢，前景不易识别，而对于前景变化速度较快的 car6 视频中，则很好地提取出了汽车的前景画面。

表 5-2: 摄像头发生晃动时前景及背景的提取结果 (MIF 处理)

模型 \ 画面	转换后原图 (Video Stabilization)	前景掩码 (无MIF处理)	动态范围背景	前景 (MIF处理)
car6				
people2				

6、问题四模型建立与求解

6.1 问题描述及分析

前景帧问题是监控视频分析实际使用中常用的手段。前景帧提取是指从一个监控视频中提取出前景在整体画面中展现显著的特定的前景帧。在这个问题中, 我们首先建立了前景帧强度的衡量指标, 该指标在时间上形成一个时间序列。由于视频中景物出现的行为状态, 前景帧的强度在不同时间上的分别表现出多个峰值的特征, 并且这些峰会偏离正态分布。这种多个峰值的分布估计采用了核密度估计的方法来完成。再根据得到的强度的分布来确定显著强度的阈值。利用得到的阈值, 筛选 8 个视频中出表现显著的关键帧, 并将编号保存。

6.2 模型建立

前景帧提取的主要做法可以分为三步。首先, 在监控视频中识别出前景, 得到前景掩码。然后, 根据前景掩码计算前景在画面中的强度。并通过阈值衡量判断该帧中的前景是否显著。最后, 将显著的前景帧进行提取并进行更深层次的分析。

前景帧提取技术的技术难点在于:

- 当前帧中前景物的识别;
- 前景物显著的判别依据。

这些问题中, 难点 1) 在前面两问的分析中已经给出了解决。在本问题中, 我们采用了中值方法配合 MIF 滤波方法对每一帧的前景掩码进行提取。但是在本例中, 既包含了动态背景, 也包含了相对静止的背景, 还包含了模式在期间变化的动态背景。由于背景状态的不同, 为了使模型能够更加适应特定的背景环境, 我们需要对每种情况提出具有针对性的调整措施。

而对于难点 2) 前景显著的判据是我们本问题的难点。前景掩码标志了一帧的前景的分布。因此我们解决这个问题的思路是, 首先根据前景掩码建立前景强度的评估指标, 之后对前景强度指标的分布进行建模, 我们将采用一种非参数统计的方法, 根据前景强度的样本对其分布进行重构, 在针对分布函数多峰的特点, 给出一种非参数的阈值确定方法。

6.2.1 前景强度模型

前景掩码的定义为：

$$G_{ij} = \begin{cases} 1 & \text{pixle}(i, j) \in foreground \\ 0 & otherwise \end{cases}$$

由于在前景掩码中，1 表示该像素点为前景，而 0 表示改像素点为背景。前景掩码包含了前景像素位置和数量的信息，这些信息可以用来对前景强度进行建模。

（1）整体强度模型：我们利用前景掩码中包含的情景像素数量信息，在一帧中，我们将前景像素所占总像素的比例定义为 t 时刻前景整体强度 L ：

$$L_t = \frac{1}{h \cdot w} \sum_{i=1}^h \sum_{j=1}^w G_{ijt}$$

前景掩码能够反映前景作为一个整体是否显著区别于背景。

（2）计数强度模型：我们对前景掩码中的位置信息进行提取，来纯化为一个能衡量当前帧中所含有前景个数的统计模型。通过 **k-means** 等聚类分析可以得到聚类的个数，因而可以利用聚类的个数来作为判据。

在我们的视频中，大部分视频出现的都是一个主题前景的形式，此外考虑到计算效率的约束，我们采用第一个整体强度模型作为单个帧前景强度的衡量标准。

对于一个视频来说，可以逐帧的计算前景强度 L_t ，得到一个前景强度的时间序列。假设该前景强度序列是平稳的。则对一个前景强度时间来说，有可能存在两个状态：在没有前景进入的当前帧的状态和当前景出现在一帧中的状态。

当前景没有在当前帧中出现时，当前帧的整体前景强度就是由于本底噪音引起的前景强度 L_0 ，而在前景以某种方式进入当前帧时，整体强度在本底噪音的基础上叠加了真正前景的强度 $L_t = L_0 + L_t^{fs}$ 。

此外，我们观察到分析前景物体在场景中运动的方式，可以分为如下几种

- 移动进入，移动推出
- 闪现，也被称为鬼影
- 始终在场景中，但姿态和位置发生变化

这三种前景物的运动方式影响了前景强度的分布。当前景移动进入和退出时，会在本底前景强度外多一个峰值，因为前景物在进入画面并滞留一段时间的过程中，在分布上累积了一个距离本底背景前景大小的强度，并且具有停留时间哥样本，这造成了最终分布的峰值。同理闪现的鬼影也可以产生远离本底背景的峰值，只是由于停留时间很短，所以只能累积一个样本。但是始终停留在场景中变化的前景则不会使分布产生两个峰，相反，只会改变本底的位置。因此可以推测，前景强度时间序列分布应该呈现多峰值分布的状态。

特别的，前景的运动通常是时间的函数，而且有规律可寻，并不是完全的随机变化，这将导致前景强度 L_t^{fs} 并不来自高斯分布，因此混合高斯模型在这里表现并不是最佳的。

我们可以采用一种非参数统计的方法，核密度估计方法来对整体强度。

6.2.2 核密度估计(Kernel Density Estimation)

(1) 核密度估计定义

核密度估计是一种非参数估计方法，在机器学习领域，是一种非监督性学习方法。用于从给定分布的样本重建总体的分布函数^[12]。

已知一个分布函数的采样值，可以通过下面模型来估计该分布的密度函数

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^n k\left(\frac{X_i - x}{h}\right)$$

这种估计密度函数的方法称为核密度估计。要 \hat{f} 是 f 的一致估计量，只要核函数 $k(\cdot)$ 满足：

1. 归一化, $\int k(v)dv = 1$
2. 对称性, $k(v) = k(-v)$
3. 二阶矩有限, $\int v^2 k(v)dv < \infty$

密度估计 \hat{f} 有一个渐进正态分布，也就是说 $\hat{f}(x)$ 统计量服从中心极限定理。

(2) 交错鉴定法确定核函数宽度

交错鉴定法是一种完全由数据驱动的方法，其核心在于用一部分样本拟合模型来检验另一部分样本的拟合程度。通过不断改变训练集测试集，来评价模型的好坏。当每次都只留一个样本作为检验对象，其他样本均做训练集时，所得到的估计量称为去一核估计量(leave-one-out estimator)。

通过这种方法，我们可以来估计 \hat{f} 和 f 的希尔伯特距离，并以距离作为判据来选择窗宽，这种方法称为最小二乘交叉检验。

$$\begin{aligned} L(\hat{f}, f) &= \int [\hat{f}(x) - f(x)]^2 dx \\ &= \int \hat{f}(x)^2 dx - 2 \int \hat{f}(x)f(x)dx + \int f(x)^2 dx \end{aligned}$$

其中第三项和 \hat{f} 无关，视为常数

$$\int f(x)^2 dx = C$$

第二项采用去一核估计量估计，即

$$\int \hat{f}(x)f(x)dx = E_x [\hat{f}(X)] = \frac{1}{n} \sum_{i=1}^n \hat{f}_{-i}(X_i) + O(n^{-1/2})$$

其中 $E_x[\cdot]$ 是对 x 求期望，用来区别对观测量 X_i 求期望，本质上是已知 X_i 的条件期望。在 X_i 处的去一核估计量 $\hat{f}_{-i}(X_i)$ 定义为：

$$\hat{f}_{-i}(X_i) = \frac{1}{(n-1)h} \sum_{j \neq i}^n k\left(\frac{X_i - X_j}{h}\right)$$

表示用除了 X_i 这个观测量外的其他观测量来估计 X_i 处的密度函数。如果不采用去一核估计量进行估计而是采用全部的样本，最终这一项会等于一个和样本无关常数 $1/h \cdot k(1/h)$ ，失去了对样本评价的作用。

第一项直接代入 $\hat{f}(x)$ 的估计式，可以得到

$$\begin{aligned}
\int \hat{f}(x)^2 dx &= \int \left[\frac{1}{nh} \sum_{i=1}^n k\left(\frac{X_i - x}{h}\right) \right]^2 dx \\
&= \frac{1}{n^2 h^2} \sum_{i=1}^n \sum_{j=1}^n \int k\left(\frac{X_i - x}{h}\right) k\left(\frac{X_j - x}{h}\right) dx \\
&= \frac{1}{n^2 h} \sum_{i=1}^n \sum_{j=1}^n \int k\left(\frac{x - X_i}{h}\right) k\left(\frac{X_i - X_j}{h} - \frac{x - X_i}{h}\right) d\left(\frac{x - X_i}{h}\right) \\
&= \frac{1}{n^2 h} \sum_{i=1}^n \sum_{j=1}^n \bar{k}\left(\frac{X_i - X_j}{h}\right)
\end{aligned}$$

其中 $\bar{k}(t) = \int k(x)k(t-x)dx$ 是 $k(\cdot)$ 的重卷积核(two-fold convolution)，一般是两个独立同分布的随机变量之和的分布。可证明， $\bar{k}(\cdot)$ 也是偶函数。

总体分布函数为 $f(x)$ ，通过去一核估计交叉检验得到的估计量 \hat{f} 的积分平方误差 CV 为：

$$CV_f(h) = \frac{1}{n^2 h} \sum_{i=1}^n \sum_{j=1}^n \bar{k}\left(\frac{X_i - X_j}{h}\right) - \frac{2}{n(n-1)h} \sum_{i=1}^n \sum_{j \neq i}^n k\left(\frac{X_i - X_j}{h}\right) + C$$

其中 $\bar{k}(t) = \int k(x)k(t-x)dx$ 是 $k(\cdot)$ 的重卷积核。

可以通过成熟的数值算法对 $CV_f(h)$ 进行优化求解得到使交叉检验 CV_f 最小的核宽度 h 。

将 $CV_f(h)$ 的首项提出，并使首项最小，会发现得到的最优解退化为 IMSE 最优解的情形。

6.2.3 基于 KDE 分布前景显著判据

根据前面对前景强度 L 的分布的分析可以知道，前景强度 L 的时间序列在统计上表现出多峰分布的特征。并且这些峰不一定是正态分布的。因此，我们直接采用核密度估计的方法来估计整体前景强度 L 的分布函数。为了能够进行核密度估计方法，我们需要知道核函数密度的带宽。这里将采用最小化交叉检验函数 CV_f 的方法来获得。

当得到了一个躲峰的密度估计函数之后，根据之前的分析，不同整体强度上的第一个峰一定是本底噪音引起的峰值，而且这也是最强的峰。之后远离本底噪音的峰值是由于前景的进入而导致的。所以我们的策略是，首先在分布函数上找到最大峰值点，之后我们希望得到一个能区分由背景噪音引起的峰和前景造成的峰的阈值点，而他们之间的最小值就是这样一个点，即找使得最大回撤最大的对应的强度值作为阈值。其中最大回撤定义为：

$$MDD(T) = \max_{\tau \in (0, T)} \left[\max_{t \in (0, \tau)} X(t) - X(\tau) \right]$$

计算阈值算法的流程为：

算法 6-1

Step 0. Start 打开视频文件，初始化前景强度向量 L_t ；
Step 1. 利用中位数法(Med)，混合高斯模型方法(GMM)，或者独立模式背景提取(IMBS)方法获取视频的离线本底背景；
Step 3. 读取下一帧，从本帧中减去背景得到前景矩阵
Step 4. 利用 Otsu 方法计算前景掩码的阈值，并计算前景掩码
Step 5. 针对视频特性，利用本文提出的 MIF 方法等方法对前景掩码进行滤波
Step 6. 计算当前帧的前景前度 L ，并存入前景强度向量 L_t
Step 7. 重复 Step 3~Step 6 直到处理完视频中的所有帧
Step 8. 利用 CV 函数的优化计算用来估计 L_t 分布的最佳核密度估计带宽 bw
Step 9. 利用 $0.8*bw$ 计算核密度估计，得到 L_t 的密度分布函数
Step 10. 计算最大回撤发生的区间，以区间的末端点 L 值作为阈值
Step 11. 大于阈值的帧被视为前景显著的关键帧。
Step 12. End

6.2.4 针对情景设置不同的过滤方法

由于视频的动态范围不同，视频的背景和前景的特点不同，需要针对不同的视频设计不同的降噪滤波方法，以保证对于每个视频都能达到较高的信噪比，实现视频过滤。而在动态背景中，采用 MIF 方法确定动态背景的范围，成为了我们得到稳定结果的关键一步。

表 6-1：不同视频的特点及所进行的优化方案

视频	特点	针对视频的优化
Campus.avi	背景中数为动态背景，产生大量噪音	利用本文的 MIF 方法确定动态背景范围，并进行降噪
Curtain.avi	窗帘摆动频率较低，动态范围较广，条形	
Escalator.avi	扶梯周期性变化，但不是前景	利用 MIF 确定电梯的作为动态背景，进行中值降噪
Fountain.avi	水流快速变化，喷泉有两种模式	利用 MIF 确定动态背景范围，并且减少计算背景的时间窗口长度
Hall.avi	静态背景，目标多	直接用中位数模型计算背景，即可以得到比较的结果
Lobby.avi	背景有灯光的明暗交替变化	减小计算背景的时间窗口，并在最终结果中扣除灯光切换的影响帧
Office.avi	目标站立时间久	利用前几帧中卫数方法求背景
Overpass.avi	有水流和树木的动态背景	利用 MIF 扣除水纹和树影

根据表，我们对 8 组视频中的每个视频的特点进行了分析，并设计了响应的对应方案，均采用稳健的中位数模型和 MIF 进行动态背景分析，并对动态背景区域进行了空间中位数滤波。为了更加精确的得到像素强度，我们没有对掩码图像进行形态学处理。

6.3 问题四求解与分析

6.3.1 基于 KDE 的阈值选取

(1) Office 视频

我们在这里用 office 视频处理过程来说明利用 KDE 进行阈值选取的过程。我们首先利用 MIF-中位数模型对每一帧前景图像进行提取，通过上述算法得到视频前景强度的核密度估计以及时间序列图。其中，核密度估计图为了确定合理阈值（途中橙色虚线），而时间序列图是为了提取出前景所在帧的位置。

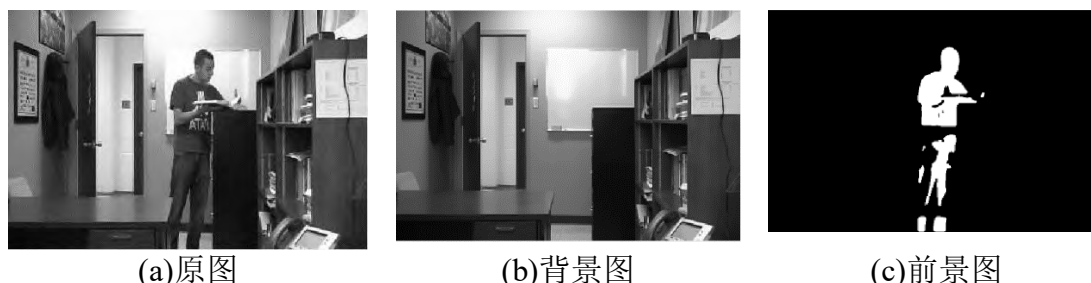


图 6-1 基于 KDE 的 office 视频前景帧识别

从前景强度核密度估计图来看，阈值选取自动选取为 $6.81E+05$ ，同时，在核密度估计的时间序列图上通过阈值可以自动筛选出前景帧。值得注意的是，在时序图中，存在某一帧上的瞬间变化的情况，此帧在图像上表现为前景图像一闪而过，我们将此类帧定义为“鬼影帧”，并于连续的前景目标运动的正常帧加以区分。

从结果上来看，“鬼影帧”为第 197/372/501/2080 帧，而前景目标帧为 583-2039 帧。

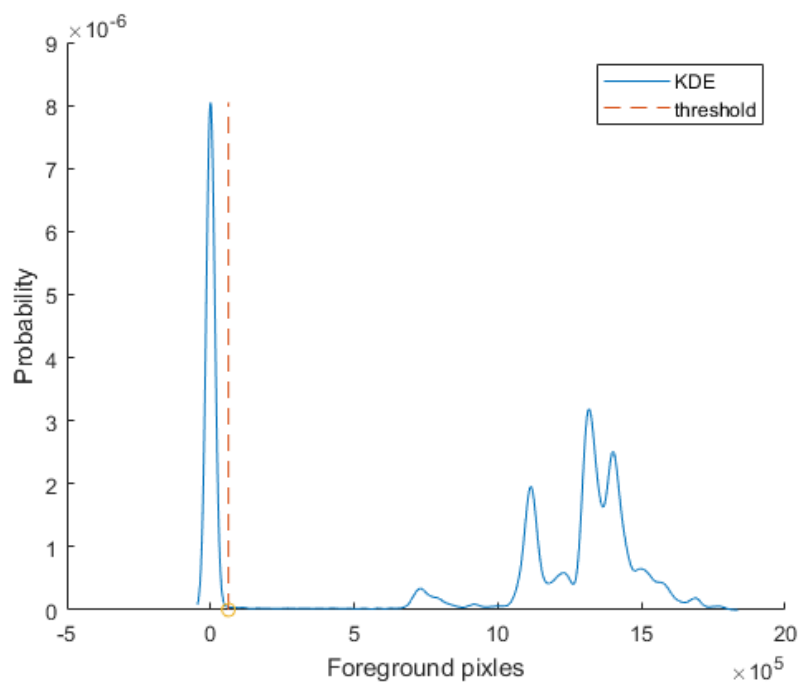


图 6-2 office 视频前景强度核密度估计

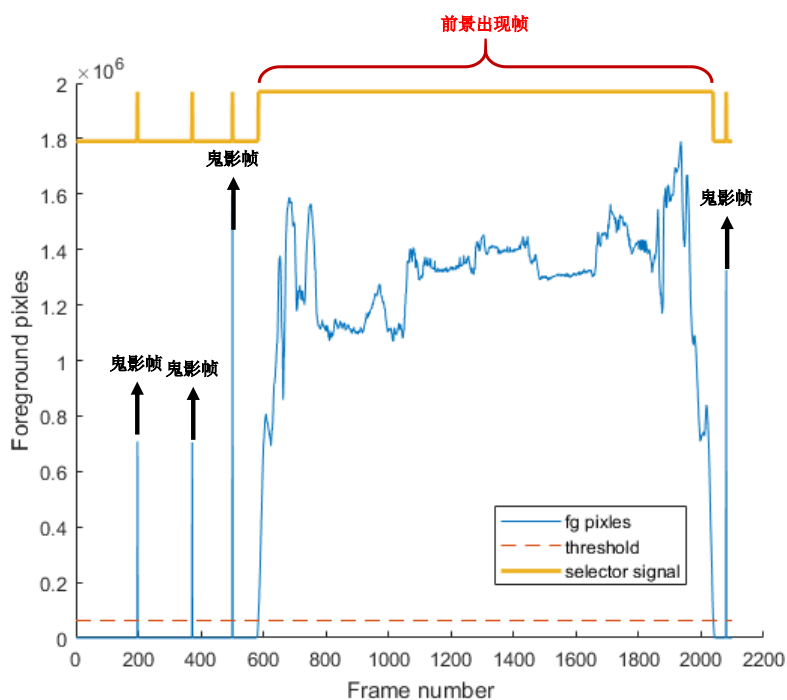


图 6-3 office 视频前景强度时间序列

(2) campus 视频

campus.avi 视频的输出结果如下，在第 205 帧画面中，前景经过 MID-中位数模型已很好的提取出，同样，基于前景强度核密度估计确定阈值为 $2.47\text{E}+05$ ，从结果上来看，“鬼影帧”为第 85/600/1192/1264 帧，而前景目标帧为 199-228；308-543；640-683；689-905；1005-1038；1322-1404 帧。



图 6-4 基于 KDE 的 campus 视频前景帧识别

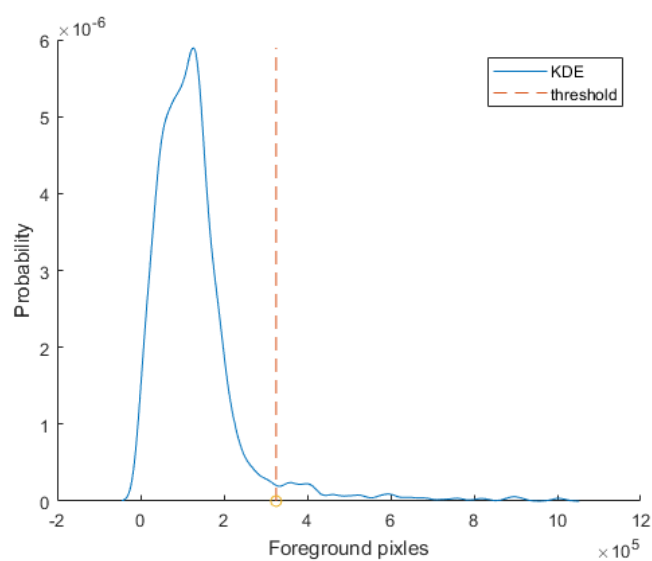


图 6-5 campus 视频前景强度核密度估计

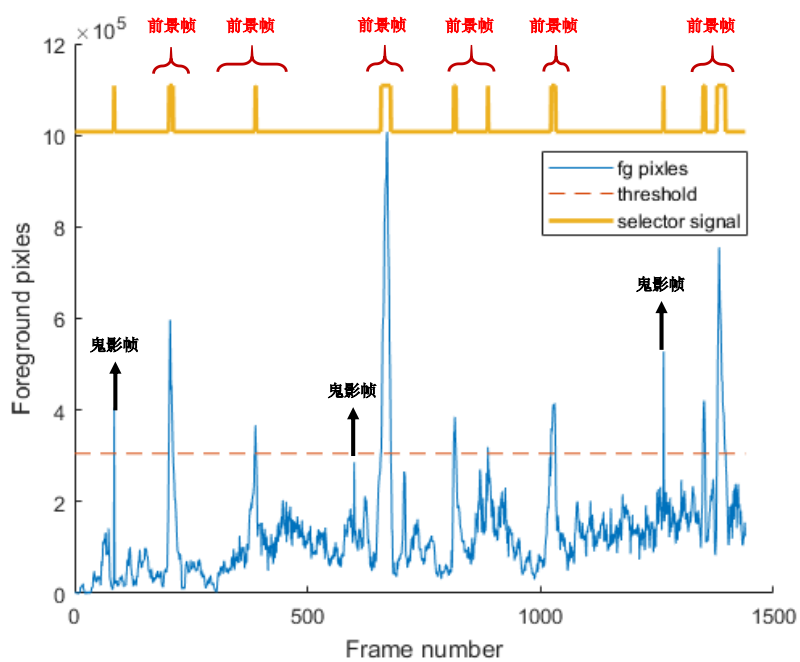


图 6-6 campus 视频前景强度时间序列

(3) Escalator 视频

同样，Escalator.avi 视频的输出结果如下，前景经过 MID-中位数模型已很好的提取出，同样，基于前景强度核密度估计确定阈值为 $3.29E+04$ ，从结果上来看，“鬼影帧”为第 2415/2539/2754 帧，而前景目标帧为 0-2391；2774-3414 帧。



图 6-7 基于 KDE 的 Escalator 视频前景帧识别

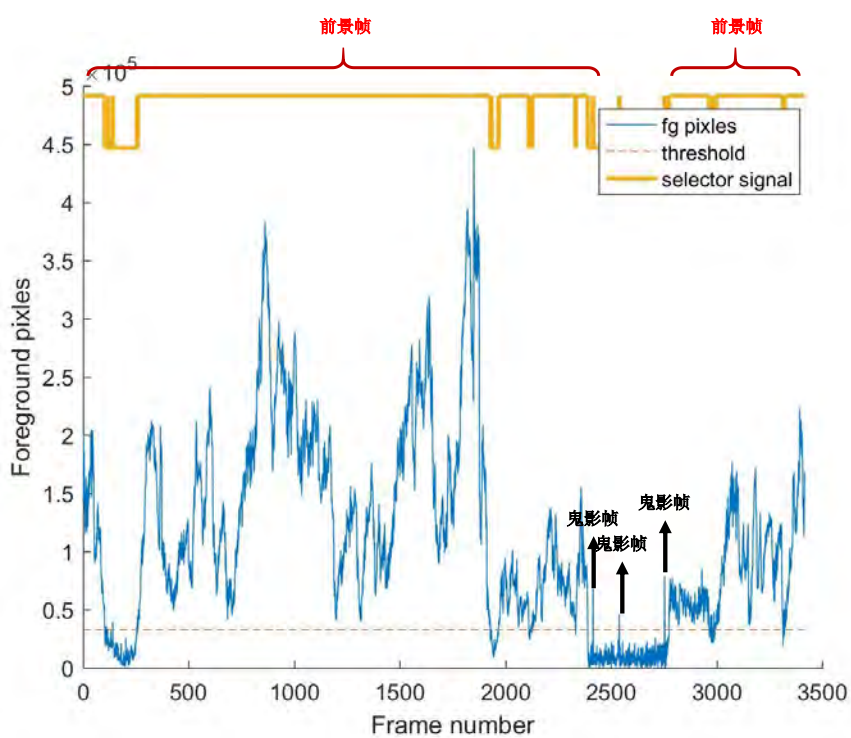


图 6-8 Escalator 视频前景强度时间序列

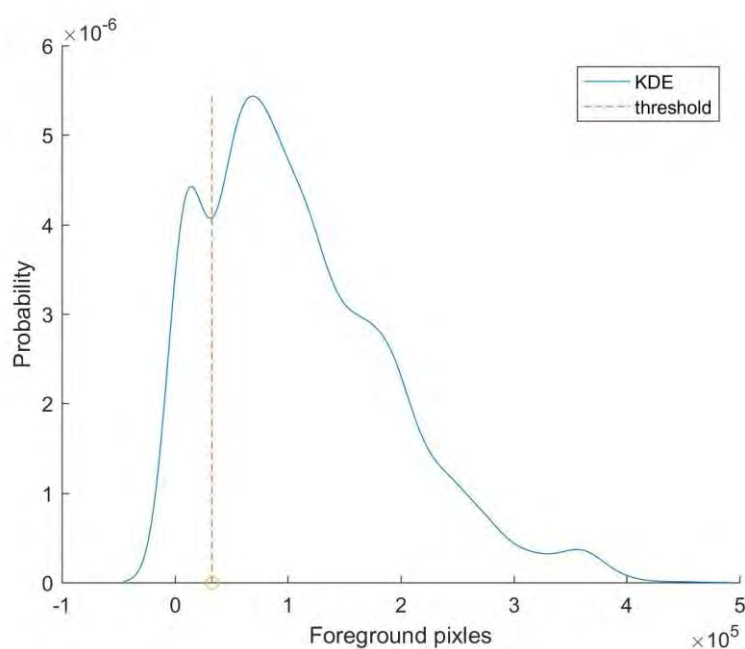


图 6-9 Escalator 视频前景强度核密度估计

6.3.2 所有前景显著帧的结果

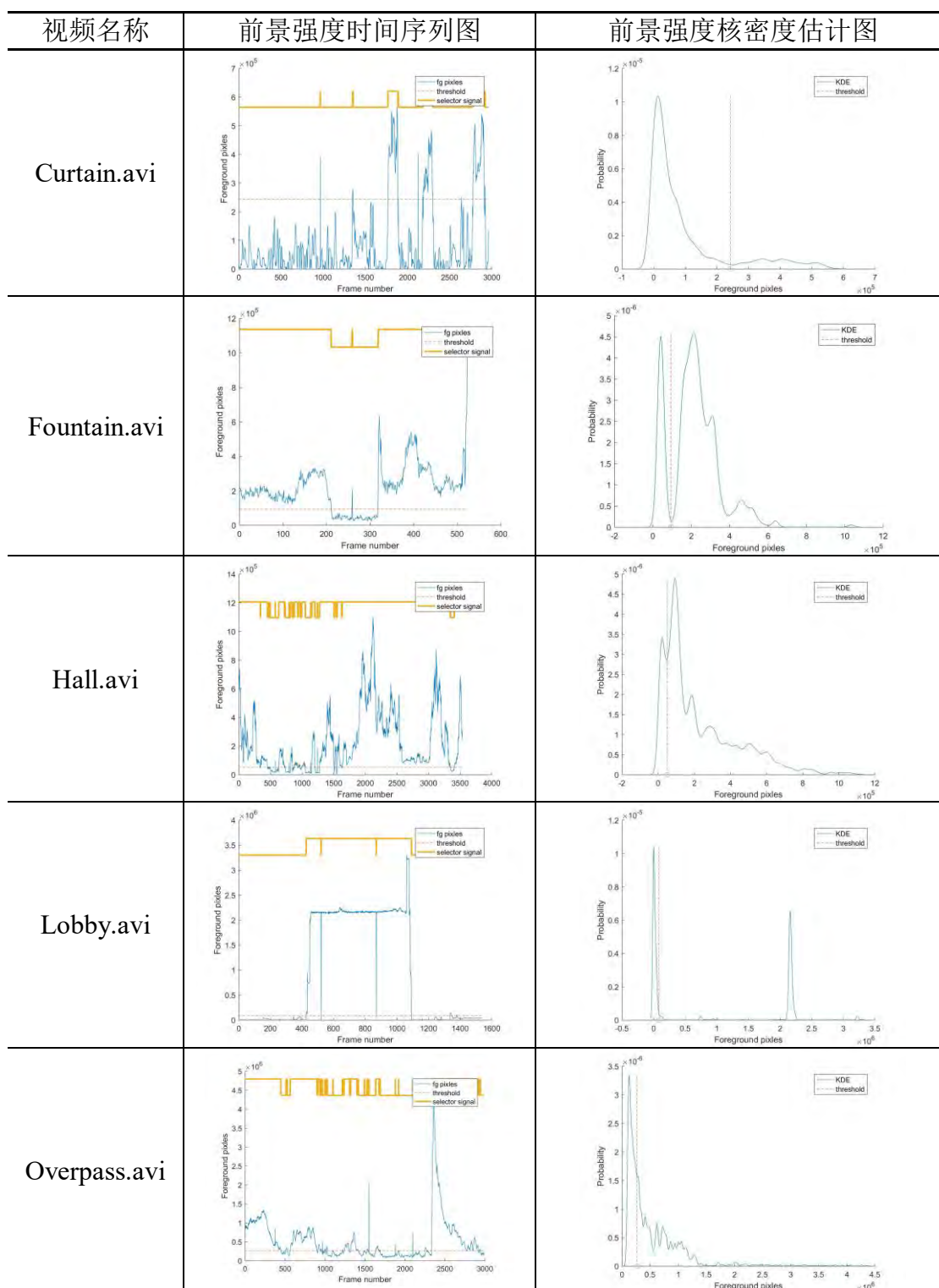
由于篇幅的限制，我们不对每一个视频进行展开讨论，所有视频的阈值、鬼影帧、以及前景出现帧的结果如下表所示：

表 6-2：基于 KDE 的核密度估计鬼影帧与前景帧识别结果

视频名称	阈值	鬼影帧	前景出现帧
Campus	2.47E+05	85/600/1192/1264	199-228; 308-543; 640-683; 689-905; 1005-1038; 1322-1404
Curtain	2.43E+05	411/967/1560/2126/2642	1756-1905; 2173-2316; 2767-2931
Escalator	3.29E+04	2415/2539/2754	0-2391; 2774-3414
Fountain	9.35E+04	141/259/335	154-212; 318-523
Hall	5.08E+04	578/795/1138/1246	0-452; 634-1052; 1153-3340; 3451-3534
Lobby	8.39E+04	521/870/1161	153-198; 344-394; 621-673; 967-1027; 1237-1284; 1333-1538
Office	6.81E+05	197/372/501/2080	583-2039
Overpass	6.00E+05	374/968/1551/1881/2098	600-709; 737-873; 2336-2869

对于其他视频的前景强度核密度估计图与前景强度时间序列不展开分析，可通过前景强度核密度估计的橙色虚线的阈值提取出时间序列图中的前景所在帧，同样，在时序图中单独一帧显著高于其他帧的所在帧为“鬼影帧”，其他视频输出结果如下：

表 6-3：其他视频基于 KDE 的核密度估计识别结果



7、问题五模型建立与求解

7.1 问题描述及分析

问题四主要解决的是从不同角度同时拍摄的近似同一地点的多个监控视频中有效检测和提取视频前景目标。本题本质上是考察在不同的监控图像序列中找到运

动目标之间的对应关系并进行同一性标记^[6]。我们在基于对极几何关系理论的基础上，用 SIFT 算法找出两幅图像中的对应点 x_i 和 x'_i ，再利用四点共面形成的单应矩阵求解基础矩阵 F ，即得到图像两两间特征点对应关系，从而实现在不同监控图像中标定同一对象。

7.2 模型建立

7.2.1 对极几何与基础矩阵

两幅图像之间存在对极几何关系，如图

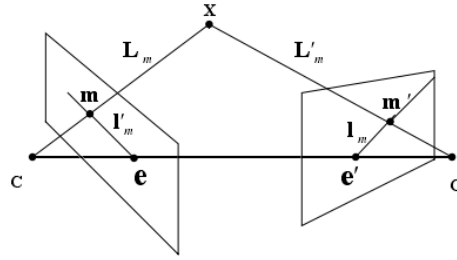


图 7-1 对极几何关系示意图

两个相机的光心分别记为 C 和 C' 。连接两个相机光心的直线称为基线，通过两个光心的平面称为极平面，任意两张极平面均相交于基线。设 m 和 m' 分别为空间点 X 在两个相机上的投影，记为一组对应点 $m \leftrightarrow m'$ 。 m 的反投影线 L_m 与 m' 的反投影线 $L_{m'}$ 相交于空间点 X 。基线与图像平面相交于两个点，记为极点 e 和 e' 。 L_m 在第二个图像平面的投影为 l_m ， $L_{m'}$ 在第一个相机图像平面的投影为 $l_{m'}$ ，称为对极线。两幅视图之间的对极几何关系可以用基础矩阵来刻画。假设两个相机的投影矩阵分别为 P 和 P' ，记它们的像平面分别为 I 和 I' ，对于 $\forall m \in I$ 的反投影线 L_m 的参数方程为：

$$X(s) = P^+m + sC, \quad s \in (-\infty, +\infty)$$

其中， P^+ 是 P 的广义逆矩阵，即 $PP^+ = I$ 。 C 是第一个相机的光心，那么 $PC = 0$ 可推知， $l_m = [e']_x P' P^+ m$ 。记 $F = [e']_x P' P^+$ ，则 $[e']_x$ 为对极点 e' 的反对称阵， F 称为两个相机间或者两幅图像 I 和 I' 间的基础矩阵。基础矩阵描述了图像点 m 与其对应直线 L_m 的对应关系

$$l_m = Fm$$

由于图像点 m 在第二幅图像上的对应点 m' 在直线 L_m 上，故有

$$m'^T Fm = 0$$

且 $Fe = 0$ ， $F^T e' = 0$ ，此时， F 矩阵刻画了两幅图像间的对极几何关系。

7.2.2 基于对极线的四点共面基础矩阵算法

当取存在共面约束关系的空间点的投影计算基本矩阵时，会对基础矩阵的计算产生新的约束。如图 2 所示，已知在图像 I 和图像 I' 上有 6 组对应点 $x_i \leftrightarrow x'_i$ ，它们是三维场景中点 $X_i (i=1, \dots, 6)$ 在两图像平面的投影。其中，前 4 点位于空间平面 π 上。相机光心 C 点和 X_5, X_6 的连线与空间平面 π 相交于点 X_5 和 X_6 。

X_5 和 X_6 在图像 I 上的投影重合为 x_5 ；在图像 I' 上的投影分别为 x'_5 和 x'_6 ； x_6 、 x'_6 和 x'_5 关系亦然。由空间 4 点共面可知前四组对应点满足单应关系 $x'_i = Tx_i (i=1, \dots, 6)$ ，可线性求解单应性矩阵 $T_{3 \times 3}$ ，从而求出 $x'_5 = Tx_5$ ， $x'_6 = Tx_6$ 。根据对极几何理论，对极线 $l_1 = x'_5 \times x_5$ ，用基础矩阵表示为 $l_1 = Fx_5$ 。对极线 $l_2 = x'_6 \times x_6$ 。两对极线的交点记为对极点 e' ，那么 $l_1 = e' \times x_5 = e' \times Tx_5$ 。合并上述两式，基础矩阵可用对极点和单应矩阵表示为

$$F = e' \times T = [e']_x T$$

在上述运算中，巧妙地虚拟了空间点 X_5 和 X_6 ，先计算对极点 e' ，再用对极点和四点共面形成的单应矩阵求解基础矩阵。

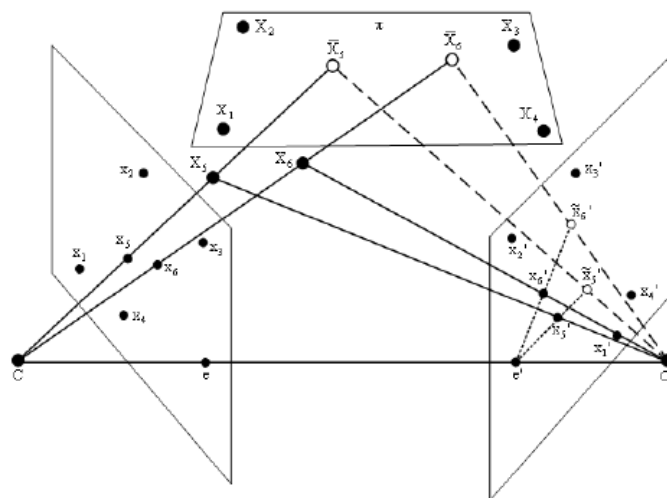


图 7-2 基于对极线的四点共面示意图

7.2.3 SIFT 算法

SIFT 算法的主要思想是在不同的尺度上构建二维高斯模糊的图像金字塔，并用 Hessian 矩阵计算金字塔层的图像特征^[7]。SIFT 特征点检测算法的处理简要介绍如下：

(1) 生成高斯差分尺度空间：首先对不同大小的图像，用不同尺度的高斯核对图像进行卷积，之后再对生成的高斯尺度空间相邻的尺度相减，这样就得到了不同大小不同尺度的高斯差分尺度金字塔。

(2) 搜索差分尺度空间的局部极值点：在金字塔每一层中，寻找极值点，使得该点在本层及相邻的尺度空间中的邻域 26 ($3 \times 3 \times 3$) 个点中都是极值点。

(3) 精确定位局部极值点：用泰勒展开式结合在 x 轴方向、 y 轴方向、 s 尺度方向上的差分公式，求取极值点的精确位置。此步骤可以将精确度提高到亚像素级别。

(4) 过滤较强的边缘响应：对求取的精确极值点，利用海森矩阵的秩和迹，判断特征值范围，从而过滤在边缘响应强的伪极值点。

(5) 寻找特征点的主方向：利用梯度差分公式，统计该极值点像素邻域的区域梯度直方图。将梯度值通过高斯函数进行加权，再将 360 分成 36 个区间，统计区间的梯度值之和，将最大的或是较大的方向定为主方向。

(6) 生成特征点描述子：首先特征点邻域用前一步骤生成的主方向进行旋

转，然后同样是统计邻域的梯度直方图，并用高斯函数进行加权。最后将梯度直方图的值以 45 为一个区间进行拼合。

经过上述步骤，可以生成具有平移、旋转、尺度不变性的 SIFT 特征点描述子。

7.2.4 RANSAC 算法

RANSAC 方法即随机采样一致性算法，其主要思想是：对图像匹配点对进行随机采样，寻找一个参数集合，使得参数集合与最大的数据子集相一致，而这种一致性检查需要由用户设定一个误差的阈值^[8]。采用 RANSAC 方法估计基础矩阵步骤如下：

Step1: 确定随机抽样次数 K。

Step2: 对于每次采样得到的子样本，子样本点个数 $m(m=7, 8)$ ，求解基础矩阵为 F_i 。并计算所有图像匹配点对的残差平方 d_i ，其中值记为 err_i 。取阈值 $T=1.96\sigma$ ， $\sigma=1.4826(1+5/(n-m))\sqrt{err_i}$ 。计算 n 对匹配点中残差平方 d_i 小于阈值 T 的匹配点对数，记为 num_i 。若 d_i 小于阈值 T ，则权值为 1；否则权值为 0，记录此时的权值矩阵 W_i 。

Step3: 取 $num_i(K(i=1, 2, \dots, K))$ 中的最大值，此时对应的权值矩阵记为 W_{best} 。剔除 W_{best} 中权值为零对应的匹配点对，得到新的图像匹配点对集合，即内点集合。由新获取的内点集合，重新估计基础矩阵。

7.3 问题五求解与分析

算法步骤如下，

- 1) 利用 sift 算法找出两幅图像中的对应点 x_i 和 x_i' 。
- 2) 利用平面间的单应性算法找出图像中的共面点。
- 3) 运行 RANSAC 算法，进行如下 N 次最小点集采样计算：
 - (a) 建立 6 点对应点集合，其中 4 点空间共面；
 - (b) 求解投影矩阵 P 和 P' ，计算基础矩阵 F ；
 - (c) 计算对应点到对极线距离的平方

$$d_j = (x_i'^T F x_i)^2 \left(\frac{1}{(F x_i)_1^2 + (F x_i)_2^2} + \frac{1}{(F^T x_i')_1^2 + (F^T x_i')_2^2} \right)$$

当 d_j 小于指定的阈值时，认为是正确匹配点，否则认为是误匹配。统计匹配点的数目；

(d) 在进行 N 次采样后，找出具有最多匹配点的 F ；利用这些匹配点计算 F 的最小二乘解。

- 4) 利用求得基础矩阵 F 标定不同图像中对应相同目标的特征点。

我们利用上述模型，编写程序得到如下结论，验证了三个影像中的相对位置，由相对位置定位了人的绝对位置。

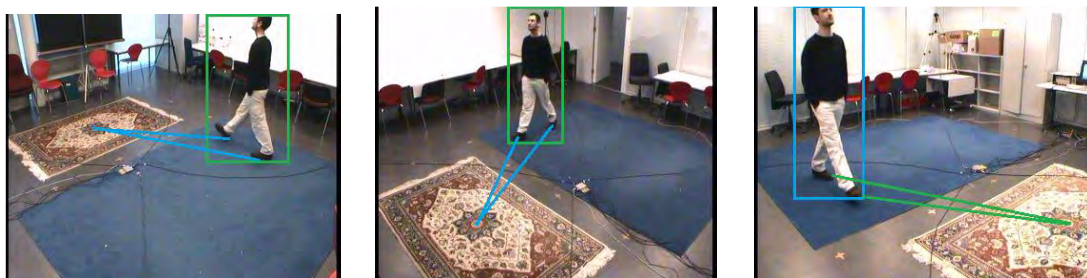


图 7-3 多视角中人对标志物的相对位置与绝对位置

8、问题六模型建立与求解

8.1 问题描述及分析

群体行为分析是一种公众行为，是指公众大范围的具有集体性行为 (Collective behavior)^[15] 的行为。由于群体行为有很大的随意性，而且经常会出现遮挡和重叠的现象，因此，对群体行为的分析很多诉诸于统计方法。最近还有些工作将物理学中混沌理论用于公众行为的分析。这些理论多着眼于行为时间和空间的特征。

在时间和空间上最具有明显特征的行为就是部队行军了。部队具有严格的纪律性，令必行，禁必止。特别是我国 20 周年的阅兵式，浩然正气，扬我国威。而三军仪仗队表现出来的美感实际上就是时间上和空间上严格周期性的体现。我们就来分析一下我国阅兵式的视频，来探究其中的周期性，并提出一个识别部队阅兵式的判据。

20 世纪最伟大的成就，傅里叶变换联系了信号的时域空间和频域空间，奠定了 20 世纪以来物理学的基石。对周期和准周期信号进行傅里叶变换可以用来获取信号的频率，而对非周期信号处理可以得到相应信号的频谱。

本文着重描述了一种部队行进过程中的频谱特征获取方法，用来发现信号中的频域特性。

8.2 模型建立

我们可以对视频数据构建一个频谱分析的特征。即对响应的具有周期性的视频中提取周期信号作为原有视频的特征。

8.2.1 傅里叶变换

傅里叶变换由法国学者 Fourier 在研究热传导方程时提出。认为一个定义在 \mathbb{R} 上并且周期为 $2l$ 的函数：

$$f(x+2l) = f(x)$$

都可以用无穷多正弦函数和余弦函数的和来逼近。并且注意到，不同周期的正弦函数之间和余弦函数之间以及他们彼此交叉在 \mathbb{R} 上的积分都是 0，这被称为正弦函数和余弦函数正交性。也就是说：

$$f(x) = a_0 + \sum_{k=1}^{\infty} \left(a_k \cos \frac{k\pi x}{l} + b_k \sin \frac{k\pi x}{l} \right)$$

并且三角函数族满足

$$\begin{aligned}\int_{-l}^l 1 \cdot \cos \frac{k\pi x}{l} dx &= 0 \quad k \neq 0 \\ \int_{-l}^l 1 \cdot \sin \frac{k\pi x}{l} dx &= 0 \\ \int_{-l}^l \cos \frac{k\pi x}{l} \cdot \cos \frac{n\pi x}{l} dx &= 0 \quad k \neq n \\ \int_{-l}^l \sin \frac{k\pi x}{l} \cdot \sin \frac{n\pi x}{l} dx &= 0 \quad k \neq n \\ \int_{-l}^l \cos \frac{k\pi x}{l} \cdot \sin \frac{n\pi x}{l} dx &= 0 \\ \int_{-l}^l \cos \frac{k\pi x}{l} \cdot \cos \frac{k\pi x}{l} dx &= l\end{aligned}$$

利用三角函数的正交性，可以求得展开系数

$$\begin{aligned}a_k &= \frac{1}{\delta_k l} \int_{-l}^l f(\xi) \cos \frac{k\pi \xi}{l} d\xi \\ b_k &= \frac{1}{\delta_k l} \int_{-l}^l f(\xi) \sin \frac{k\pi \xi}{l} d\xi\end{aligned}$$

其中 $\delta_k = 2$ 当 $k=0$ 且 $\delta_k = 1$ 当 $k=1$ 。

这里以理论一开始并没有收到数学家的接收，他们认为一个不可导甚至不连续的函数能被一系列无穷可导的函数求和来逼近是不能理解的。然而直到广义函数的提出，傅里叶级数才被数学界所接受。

设 $f(x)$ 是定义在区间 $-\infty < x < \infty$ 上的函数，一般来讲，这个函数是非周期的，因此不能展开为 Fourier 级数。为了研究这样的函数的 Fourier 展开，我们采用周期扩展的办法，将视为某个周期函数 $g(x)$ 在周期 $2l \rightarrow \infty$ 的极限情形。这样 $g(x)$ 就可以被展开为傅里叶级数。

为此，引入不连续参量 $\omega_k = k\pi/l$, $\Delta\omega_k = \omega_k - \omega_{k-1} = \pi/l$ ，那么

$$g(x) = a_0 + \sum_{k=1}^{\infty} (a_k \cos \omega_k x + b_k \sin \omega_k x)$$

对应的 Fourier 系数为

$$\begin{aligned}a_k &= \frac{1}{\delta_k l} \int_{-l}^l f(\xi) \cos \omega_k \xi d\xi \\ b_k &= \frac{1}{\delta_k l} \int_{-l}^l f(\xi) \sin \omega_k \xi d\xi\end{aligned}$$

当长度 $l \rightarrow \infty$ 时，若 $\lim_{l \rightarrow \infty} \int_{-l}^l f(\xi) d\xi$ 有限，则可证明，极限形式为

$$f(x) = \int_0^{\infty} A(\omega) \cos \omega d\omega + \int_0^{\infty} B(\omega) \sin \omega d\omega$$

其中

$$\begin{aligned}A(\omega) &= \frac{1}{\pi} \int_{-\infty}^{\infty} f(\xi) \cos(\xi) d\xi \\ B(\omega) &= \frac{1}{\pi} \int_{-\infty}^{\infty} f(\xi) \sin(\xi) d\xi\end{aligned}$$

称为 Fourier 系数而对应的

$$C(\omega) = \sqrt{A(\omega)^2 + B(\omega)^2}$$
$$\varphi(\omega) = \tan^{-1}(B(\omega) / A(\omega))$$

分别称为振幅谱和相位谱。

响应的复频域 Fourier 变换为

$$F(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx$$

具有了傅里叶变换以后，我们可以将一维傅里叶变换扩展到二维。也就是说，我们因此可以计划对一个二维矩阵进行傅里叶变换，那就就是说，我们可以对一张图片进行二维 Fourier 变换。

高维度 Fourier 变换的通用表达式为

$$f(\mathbf{r}) = \iiint_{\mathbf{R}} F(\mathbf{k}) e^{i\mathbf{k} \cdot \mathbf{r}} d\mathbf{k}$$
$$F(\mathbf{k}) = \frac{1}{(2\pi)^n} \iiint_{\mathbf{R}} f(\mathbf{r}) e^{-i\mathbf{k} \cdot \mathbf{r}} d\mathbf{r}$$

8.2.2 队列行进建模

队列行进是一项具有高度组织和纪律化的行动，我们认为队列行进具有很强的纪律性，每一个步伐都要求整齐一致。假设士兵行进的步伐一致，频率固定。那么在时间空间上，一个士兵的特征会随着时间在空间上线性传播，也就是说，队列行进具有下面连个特征

- 1) 前景掩码在给定一个空间固定的时间-空间图上的应当为近似的直线段。
- 2) 一个像素点的时间序列做傅里叶变换，赢得到比较窄的带宽。

8.3 问题六求解与分析

首先，我们从央视的视频中读取行军的灰度图像数据，并找到队列行进过程中的相关帧，进行裁剪，下面的队列行进在稳定的镜头环境下进行，镜头的稳定大于持续了 300 帧的长度。下图为其中的一帧的图像。

通过如下操作来直接获取队列行进中的频率信息

算法 8-1

- Step 1. 读取一帧视频，并转变为灰度图像
 - Step 2. 对灰度图像进行去除抖动和形变处理
 - Step 3. 对灰度图像进行前景拾取，得到前景的掩码
 - Step 4. 对于其中一行，选取行号固定，根据列号和时间抽出一张时间-空间图
 - Step 5. 对该时空图进行 Fourier Analysis
 - Step 6. 分析频率特征
 - Step 7. 进行时空图的线性验证和频谱的带宽验证
 - Step 7. End
-

从视频中拾取一帧，并进行灰度化处理，得到行军过程的灰度图像。



图 8-1 20 年国庆阅兵的灰度图像

之后，从视频的灰度图像中提取前景信息，得到前景的掩码，可以看到前景掩码中选择到的情景是人的服装。



图 8-2 经过中位数模型和 MIF 的前景矩阵

因而可已得到一个前景掩码的集合，在界面上可以很明显的表现出集体行为，确定行号为 90，并且所有响应的列和对应的时间组成一个时间-空间图，纵轴表示时间流逝，横轴表示空间前进，可以观察看到部队在意相同的运动方式行进，表现出集体行为。

所以，条件 1) 得到验证。

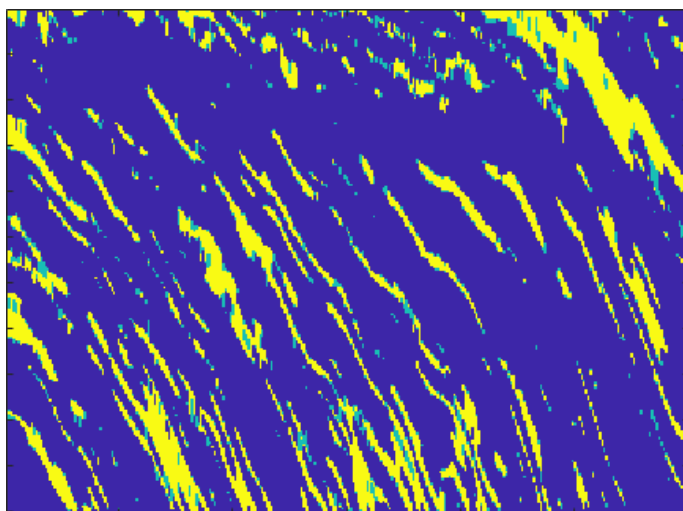


图 8-3： 前景掩码组成的面板数据在固定了行号以后所构成的时间-空间图，其中纵轴表示时间流逝，横轴表示空间前进，可以观察看到部队在意相同的运动方式行进，表现出集体行为。这里也验证了线性性的出现。

将二维的时空图对该界面进行傅里叶分解可以得到相对应的二维频谱。可以看到除了主峰之外还有两个小峰，说明部队行进表现出周期性。

因此条件 2) 得到验证。

因此通过条件 1) 2) 我么正确的判断出了这是一幅整体运动的视频，验证了自己提出的两条判据。

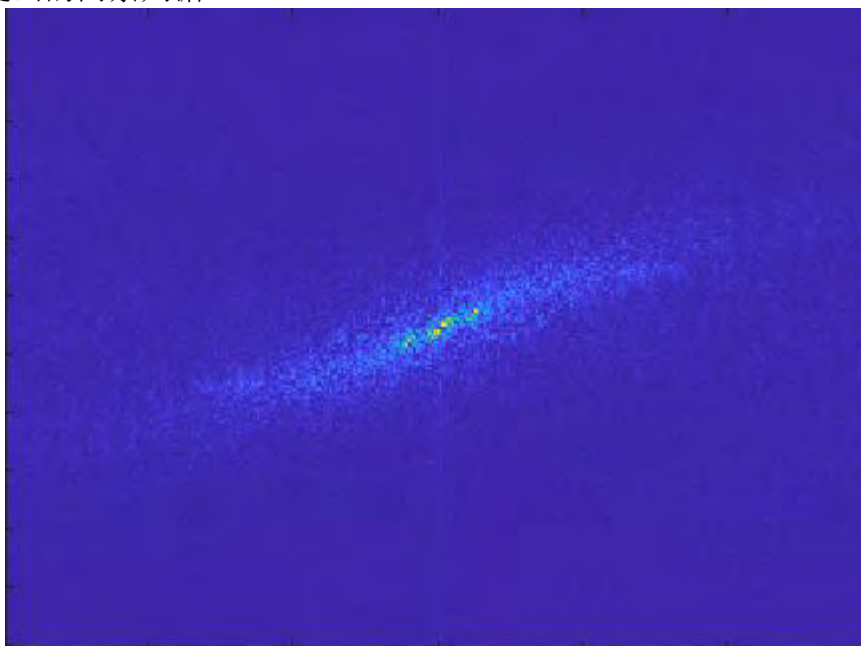


图 8-4： 可以看到二维频谱中的两个峰，对应了时间和空间上的频率，而其斜率代表了速度。

9、模型的评价与改进建议

9.1 模型评价

我们在监控视频的前倾提取问题上尝试了多种模型，包含中位数模型，平均

值模型，混合高斯模型和 IMBS。他们都是像素级的背景减去模型。共同的有点事分辨率非常高，可以精确到像素，缺点是得到对的最终结果也会发生由散点构成的情况。

为了克服这种散点前景掩码的情况，我们提出了形态学的膨胀和腐蚀操作。形态学操作确实可以将离散的点连接到一起，但是，同样的操作也会对背景噪音进行，作为副作用，也会增加噪音，降低信噪比。而且会在一定程度上降低像素级前景筛选器的分辨率。

为了处理动态背景问题，我们原创独立提出了 MIF 模型。MIF 模型具有很好的颜色直观，也能很好的识别动态背景区域，并且参数对于不同模型相对稳定。但是只用三个矩来的线性约束来构造动态背景区的的确约束能力有限，有很多区域还是会被误选。

视角稳定模型能很好通过仿射和扭曲变换来稳定视野。但是在镜头剧烈晃动时无法对应捕捉两帧的对应关系。

基于 KDE 的显著帧检测技术是一种非参数的统计技术，能够不设置阈值的情况下自动检测响应阈值，并能起到很好的二进制化效果。然而这种方法对前景帧的质量很敏感，需要前景帧具有较高的信噪比才能工作，否则，强度的估计只能在噪音之中无法分辨。

从计算精度上看，配合 MIF 的二次滤波模型效果都有提升，而在基础模型中，IMBS 最好，混合高斯和中位数模型持平，而均值模型不太好。

从计算消耗上看，中位数和平均值计算资源消耗最小。

结论：推荐使用 MIF+中位数进行前景选取。

9.2 模型与方法的改进建议

由于我们这次研究的模型都是像素级的背景减去模型，以后可以使用基于超像素的模型来避免前景目标松散像素化的问题。

文章的 KDE 模型方法也可以使用高斯混合模型来替代，通过判断掩码强度是否显著属于第一个峰来确定该图片中的前景是否是显著突出的。

最后，文中均采用的传统统计学习模型，在具有 GPU 显卡加速的情况下，深度学习在图像识别和处理中具有非常大的发展前景。

参考文献

- [1]宋纯贺. 视频监控系统中视频处理相关问题研究[D].东北大学,2011.
- [2]贾阳,林高华,王进军,方俊,张永明. 基于显著性检测和高斯混合模型的早期视频烟雾分割算法[J]. 计算机工程,2016,42(02):206-209+217. [2017-09-20].
- [3]赵海岳. 红外和可见光视频运动目标检测方法研究[D].深圳大学,2016.
- [4]田鹏辉. 视频图像中运动目标检测与跟踪方法研究[D].长安大学,2013.
- [5]易盟. 基于特征点的图像配准及其在稳像中的应用[D].西安电子科技大学,2013.
- [6]梁华. 多摄像机视频监控中运动目标检测与跟踪[D].国防科学技术大学,2009.
- [7]徐剑,丁晓青,王生进,吴佑寿. 多视角多行人目标检测、定位与对应算法[J]. 清华大学学报(自然科学版),2009,49(08):1139-1143. [2017-09-20]. DOI : 10.16511/j.cnki.qhdxxb.2009.08.025
- [8]盛安宇. 多视角摄像机视频拼接算法研究[D].华中科技大学,2015. [8] Bloisi D, Iocchi L. Independent multimodal background subtraction, CompIMAGE. 2012: 39-44.
- [9] Otsu NA.Threshold Selection Method from Gray-level Histogram. IEEE Trans. 1979.SMC-9:62-66
- [10] Edward Rosten and Tom Drummond, “Machine learning for high speed corner detection” in 9th European Conference on Computer Vision, vol. 1, 2006, pp. 430–443.
- [11] Edward Rosten, Reid Porter, and Tom Drummond, “Faster and better: a machine learning approach to corner detection” in IEEE Trans. Pattern Analysis and Machine Intelligence, 2010, vol 32, pp. 105-119.
- [12] Bloisi D, Iocchi L. Independent multimodal background subtraction, CompIMAGE. 2012: 39-44.
- [13] Laugraud B, Piérard S, Braham M, et al. Simple median-based method for stationary background generation using background subtraction algorithms[C]//International Conference on Image Analysis and Processing. Springer, Cham, 2015: 477-484.
- [14] Stauffer C, Grimson W E L. Adaptive background mixture models for real-time tracking[C]//Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on. IEEE, 1999, 2: 246-252.
- [15] Background Modeling and Foreground Detection for Video Surveillance[M]. CRC press, 2014.

附录

一、程序源代码清单

```
function [mov, dpanel] = aviread(filename)
% Read Audio/Video(AVI) file
%% read the video
vidObj = VideoReader(filename);

vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

mov = struct('cdata', zeros(vidHeight, vidWidth, 1,
    'uint8'), ...
    'colormap', []);

vlength = 0;
while hasFrame(vidObj)
    vlength = vlength+1;
    mov(vlength).cdata = readFrame(vidObj);
end

% convert uint data to grayscale and into double
k = 1;
dpanel = zeros(vidHeight, vidWidth, vlength);
for k = 1:vlength
    dpanel(:, :, k) = im2double(rgb2gray(mov(k).cdata));
end

Extract_time_frame.m
function [frame_data, vlength] = extract_time_frame(filename,
frame_num)
% Author: Wentao Wu

vidObj = VideoReader(filename);

vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

s = struct('cdata', zeros(vidHeight, vidWidth, 1,
    'uint8'), ...
    'colormap', []);
```

```

vlength = 0;
while hasFrame(vidObj)
    vlength = vlength+1;
    s(vlength).cdata = readFrame(vidObj);
end

frame_data = s(frame_num).cdata;

wentaoGMM.m
clear;
close all;

% This program used to detect moving objects in a video %
% Author: Ravdeep Johar
% Please input any avi file below %

% Video = input('Enter the name of the video file:', 's');
Video = 'watersurface/input.avi';
if isempty(Video)
    error('myApp:argChk', 'You did not enter a video file!')
end

inputVideo = aviread(Video);

% frame size variables

fr = inputVideo(1).cdata;           % read in 1st frame as
background frame
fr_bw = rgb2gray(fr);               % convert background to
greyscale
fr_size = size(fr);                 % get the size of the frame
width = fr_size(2);                 % get the width of the frame
height = fr_size(1);                % get the height of the frame
foreground = zeros(height, width);   % initialize
variable to store foreground
background = zeros(height, width);   % initialize
variable to store background

%
frame_num = 30;                     % number of frame
K = 3;                               % number of
gaussian components (can be upto 3-5)

```

```

M = 3; % number of
background components
D = 2.5; % positive
deviation threshold
alpha = 0.01; % learning rate
(between 0 and 1) (from paper 0.01)
foregroundThreshold = 0.20; % foreground
threshold (0.25 or 0.75 in paper)
sd_initial = 6; % initial
standard deviation (for new components) var = 36 in paper
th_mul = 1.0; % threshold
multiplier
morph_enable = 1; % whether to perform
morphological operation
SE_shape = 'disk'; % shape of
morphological elements:
% diamond, disk,
line,
% octagon,
rectangle,
% square, cube,
sphere
SE_R = 2; % Radius of
Element Structure in Morphological filter

weight = zeros(height,width,K); % initialize
weights array
mean = zeros(height,width,K); % pixel means
standardDeviation = zeros(height,width,K); % pixel
standard deviations
diffFromMean = zeros(height,width,K); % difference
of each pixel from mean
learningRate = alpha/(1/K); % initial p
variable (used to update mean and sd)
rankComponent = zeros(1,K); % rank of
components (w/sd)

% initialize components for the means and weights

pixel_depth = 8; % 8-bit resolution
pixel_range = 2^pixel_depth -1; % pixel range (# of
possible values)

```

```

for i=1:height
    for j=1:width
        for k=1:K

            mean(i,j,k) = rand*pixel_range;           % means
random (0-255), it initializes the mean to some random value.
            weight(i,j,k) = 1/K;                     % weights
uniformly dist
            standardDeviation(i,j,k) = sd_initial;    %
initialize to sd_init

        end
    end
end

% Applying the proposed algorithm to the video
fgMovie = struct('cdata',zeros(height, width, 1,
'uint8'),...
'colormap',[]);
bgMovie = struct('cdata',zeros(height, width, 1,
'uint8'),...
'colormap',[]);
closedMaskMovie = struct('cdata',zeros(height, width, 1,
'uint8'),...
'colormap',[]);
fgcloseMovie = struct('cdata',zeros(height, width, 1,
'uint8'),...
'colormap',[]);
for n = 1:length(inputVideo)
    % reading the frames.
    fr = inputVideo(n).cdata;
    % converting the frames to grayscale.
    fr_bw = rgb2gray(fr);

    % calculating the difference of each pixel values from
mean.
    for m=1:K
        diffFromMean(:, :, m) = abs(double(fr_bw) -
double(mean(:, :, m)));
    end

    % update gaussian components for each pixel values.
    for i=1:height
        for j=1:width

```

```

        match = 0; % its changed to 1 if the component is
matched
        for k=1:K
            % pixel matches component
            if (abs(diffFromMean(i,j,k)) <=
D*standardDeviation(i,j,k))
                % variable to signal component match
                match = 1;

                % update weights, mean, standard deviation
and
                % learning factor
                weight(i,j,k) = (1-alpha)*weight(i,j,k) +
alpha;
                learningRate = alpha/weight(i,j,k);
                mean(i,j,k) = (1-learningRate)*mean(i,j,k)
+ learningRate*double(fr_bw(i,j));
                standardDeviation(i,j,k) =
sqrt((1-learningRate)*(standardDeviation(i,j,k)^2) +
learningRate*((double(fr_bw(i,j)) - mean(i,j,k))^2);
            else % if pixel
doesn't match component
                weight(i,j,k) =
(1-alpha)*weight(i,j,k); % weight slightly decreases
            end
        end

        weight(i,j,:) =
weight(i,j,:)./sum(weight(i,j,:));

        %Save the background using all the components of
gaussian.
        background(i,j)=0;
        for k=1:K
            background(i,j) = background(i,j)+
mean(i,j,k)*weight(i,j,k);
        end

        % if no components match, create new component and
decrease the
        % parameters values
        if (match == 0)

```

```

        [min_w, min_w_index] = min(weight(i,j,:));
        mean(i,j,min_w_index) = double(fr_bw(i,j));
        standardDeviation(i,j,min_w_index) =
sd_initial;
    end

    rankComponent =
weight(i,j,:)./standardDeviation(i,j,:); %
calculate component's rank
    rankIndex = [1:1:K];

    % sort rank values
    for k=2:K
        for m=1:(k-1)

            if (rankComponent(:, :, k) >
rankComponent(:, :, m))
                % swap max values
                rank_temp = rankComponent(:, :, m);
                rankComponent(:, :, m) =
rankComponent(:, :, k);
                rankComponent(:, :, k) = rank_temp;

                % swap max index values
                rank_ind_temp = rankIndex(m);
                rankIndex(m) = rankIndex(k);
                rankIndex(k) = rank_ind_temp;

            end
        end
    end

    % calculate foreground and save it.
    match = 0;
    k=1;

    foreground(i,j) = 0;
    while ((match == 0) && (k<=M))

        if (weight(i,j,rankIndex(k)) >=
foregroundThreshold)
            if (abs(diffFromMean(i,j,rankIndex(k))) <=
D*standardDeviation(i,j,rankIndex(k)))
                foreground(i,j) = 0;
            end
        end
        k=k+1;
    end
end

```

```

        match = 1;
    else
        foreground(i,j) = fr_bw(i,j);
    end
end
k = k+1;
end
end
end

% Filter
foreground = medfilt2(foreground, [3, 3]);

%Structure element for performing morphological
operations
%SE=[0 0 0 0 0
%   0 1 1 1 0
%   0 1 1 1 0
%   0 1 1 1 0
%   0 0 0 0 0];

SE = strel(SE_shape, SE_R);

%Performing closing on the foreground frames using SE
if(morph_enable)
    closedFrame=imclose(foreground, SE);
else
    closedFrame=foreground;
end

% Plotting the foreground , background , original video
and the morphed
% video on the screen.
%   figure(1),
%   subplot(4,1,1),imshow(fr), title('Original Video');
%   subplot(4,1,2),imshow(uint8(background)),
title('Background Model');
%   subplot(4,1,3),imshow(uint8(foreground)) ,
title('Foreground( Moving Objects )');
%   subplot(4,1,4),imshow(uint8(closedFrame)) ,
title('After Morphological operation');

% put foreground frames into movie.
[X1, cm1] = gray2ind(foreground, 256);

```

```

fgMovie(n).cdata = X1;
fgMovie(n).colormap = cm1;
% put background frames into movie.
[X2, cm2] = gray2ind(mapminmax(background,0,1), 256);
bgMovie(n).cdata = X2;
bgMovie(n).colormap = cm2;
% put closed frames mask into movie.
closedMask_th = graythresh(closedFrame);
closedMask = closedFrame > th_mul*closedMask_th;
[X3, cm3] = gray2ind(closedMask, 256);
closedMaskMovie(n).cdata = X3;
closedMaskMovie(n).colormap = cm3;
% put the closed foreground movie
fgclosedFrame = fr_bw;
fgclosedFrame(~closedMask) = 0;
[X4, cm4] = gray2ind(fgclosedFrame, 256);
fgcloseMovie(n).cdata = X4;
fgcloseMovie(n).colormap = cm4;

end

disp(['show movie'])
figure()
set(gcf,'position',[300 300 width height]);
set(gca,'units','pixels');
set(gca,'position',[0 0 width height]);
movie(fgMovie, 1, 30);

figure()
% image(s(5).cdata)
set(gcf,'position',[600 300 width height]);
set(gca,'units','pixels');
set(gca,'position',[0 0 width height]);
movie(closedMaskMovie, 1, 30);

figure()
% image(s(5).cdata)
set(gcf,'position',[900 300 width height]);
set(gca,'units','pixels');
set(gca,'position',[0 0 width height]);
movie(fgcloseMovie, 1, 30);

% output a specific frame
if morph_enable

```



```

        frame_res = uint8(zeros(height,4*width));
        frame_res(:,0*width+1:1*width) =
rgb2gray(inputVideo(frame_num).cdata);
        frame_res(:,1*width+1:2*width) =
bgMovie(frame_num).cdata;
        %frame_res(:,2*width+1:3*width) =
fgMovie(frame_num).cdata;
        frame_res(:,2*width+1:3*width) =
closedMaskMovie(frame_num).cdata;
        frame_res(:,3*width+1:4*width) =
fgcloseMovie(frame_num).cdata;
    else
        frame_res = uint8(zeros(height,4*width));
        frame_res(:,0*width+1:1*width) =
rgb2gray(inputVideo(frame_num).cdata);
        frame_res(:,1*width+1:2*width) =
bgMovie(frame_num).cdata;
        frame_res(:,2*width+1:3*width) =
fgMovie(frame_num).cdata;
        frame_res(:,3*width+1:4*width) =
fgcloseMovie(frame_num).cdata;
    end
    figure()
    imshow(frame_res)

```

MIF_fast.m

```

%% Author: Wentao Wu
%% execute simple median and MIF

%% parameter
window = 0;    % windows=0 for use all frame to calculate
background
                % windows>0 for use last window number of frames
to calculate
th_mul = 1.0; % adjust the threshold for foreground
block_size = 4;

%% read the video
vidObj = VideoReader('yuebing.mp4');    % 粤北; 粤北; 粤北
vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

s = struct('cdata',zeros(vidHeight, vidWidth, 1,
'uint8'),...

```

```

        'colormap', []);

vlength = 0;
disp(['reading original video'])
while hasFrame(vidObj)
    vlength = vlength+1;
    s(vlength).cdata = readFrame(vidObj);
    disp(['vlength = ', num2str(vlength)])
end
disp(['reading complete.'])
whos s

% convert uint data to grayscale and into double
k = 1;
spanel = zeros(vidHeight, vidWidth, vlength);
for k = 1:vlength
    spanel(:, :, k) = im2double(rgb2gray(s(k).cdata));
end
whos spanel

%% Background video through Simple Median method
disp(['calculating bg'])
bgpanel = zeros(vidObj.Height, vidObj.Width, vlength);
if window > 0
    for i = 1:vlength
        bgpanel(:, :, i) =
median(spanel(:, :, max(1, i-window):i), 3);
    end
else
    %% WARNING: note the background source!!!
    bgpanel = repmat(median(spanel(:, :, 1:end),
3), 1, 1, vlength);
end
disp(['calculating bg complete'])

%% Construct the background movie
bgMovie = struct('cdata', zeros(vidObj.Height, vidObj.Width,
1, 'uint8'), ...
    'colormap', []);
for i = 1:vlength
    [X1, cm1] = gray2ind(bgpanel(:, :, i), 256);
    bgMovie(i).cdata = X1;
    bgMovie(i).colormap = cm1;
end

```

```

%% Get the foreground target
fgpanel = abs(spanel - bgpanel);

% determine the threshold
% figure
% hold on
% plot(squeeze(max(max(fgpanel,[], 1), [], 2)))
% plot(squeeze(min(min(fgpanel,[], 1), [], 2)))
% hold off

fgpanel_th = zeros(vidObj.Height, vidObj.Width);
g_fg_th = graythresh(fgpanel);
for i = 1:vlength
    fg_th = graythresh(fgpanel(:,:,i));
    % TODO limit fg_th to global
    fg_th_limited = min(max(fg_th, 0.9*g_fg_th),
1.1*g_fg_th);
    fgpanel_th (:,:,i) = fgpanel(:,:,i) > fg_th_limited;
end

%% Construct Mask
fgmask = struct('cdata', zeros(vidObj.Height, vidObj.Width,
1, 'uint8'), ...
    'colormap', []);
for i = 1:vlength
    [X1, cm1] = gray2ind(fgpanel_th(:,:,i), 256);
    fgmask(i).cdata = X1;
    fgmask(i).colormap = cm1;
end

%% Construct the foreground movie
disp(['computing fg video'])
Movie = struct('cdata', zeros(vidObj.Height, vidObj.Width, 1,
'uint8'), ...
    'colormap', []);
for i = 1:vlength
    [X1, cm1] = gray2ind(fgpanel_th(:,:,i).*spanel(:,:,i),
256);
    Movie(i).cdata = X1;
    Movie(i).colormap = cm1;
end
disp('compute complete')

```

```

%MIF_method; Essentially median + morphological
disp('med filter and morph')
clear_fgmask = struct('cdata',zeros(vidObj.Height,
vidObj.Width, 1, 'uint8'),...
    'colormap',[]);
for i = 1: vlength
    % median filter
    fgpanel_th_filtered = medfilt2(fgpanel_th(:,:,i),
[block_size, block_size]);
    se = strel('disk',3);
    fgpanel_th_fil_morph = imclose(fgpanel_th_filtered, se);
    %fgpanel_th_fil_morph = fgpanel_th_filtered;
    [X1, cm1] = gray2ind(fgpanel_th_fil_morph, 256);
    clear_fgmask(i).cdata = X1;
    clear_fgmask(i).colormap = cm1;
end
disp('complete med filter and morph')

%% Display the movie
figure()
% image(s(5).cdata)
set(gcf,'position',[300 300 vidObj.Width vidObj.Height]);
set(gca,'units','pixels');
set(gca,'position',[0 0 vidObj.Width vidObj.Height]);
movie(clear_fgmask,1,vidObj.FrameRate);

% calculate the sum of the pixels
frame_bright = zeros(vlength,1);
for i = 1:vlength
    frame_bright(i) = sum(sum(clear_fgmask(i).cdata));
end

th = 2.993e+04;
ssignal = frame_bright > th;

figure();
hist(frame_bright, 50)

figure()
hold on
[f, xi] = ksdensity(frame_bright, 'Kernel','normal',

```

```

'BandWidth', 15000, 'npoints',1000);
plot(xi, f)
plot([th, th],[0, max(f)], '--')
plot(th, 0, 'o')
xlabel('Foreground pixles')
ylabel('Probability')
legend('KDE','threshold')

figure()
hold on
plot(frame_bright)
plot([0, vlength],[th, th], '--')
plot(ssignal*max(frame_bright)*0.1+1.*max(frame_bright),
'LineWidth',2)
xlabel('Frame number')
ylabel('Foreground pixles')
legend('fg pixles','threshold','selector signal')

```

MIF_median.m

```

%% execute simple median and MIF
%simple_median_method;
%MIF_method;

%% result summary
figure()
subplot()
disp(['Height = ', num2str(vidHeight), ...
      '; Width = ', num2str(vidWidth), ...
      '; total frames = ', num2str(vlength)])
imshow(closed_mif_fig_th)
figure()
set(gcf,'position',[900 300 vidObj.Width vidObj.Height]);
set(gca,'units','pixels');
set(gca,'position',[0 0 vidObj.Width vidObj.Height]);
movie(fgmask, 3, vidObj.FrameRate);

%% Dynamic bg filter && Statical bg filter
MIFMovie = struct('cdata',zeros(vidObj.Height, vidObj.Width,
1, 'uint8'),...
    'colormap',[]);
fgmask_dbg = zeros(vidHeight, vidWidth);
fgmask_sbg = zeros(vidHeight, vidWidth);
for i = 1:vlength

```

```

fgmask_frame = fgmask(i).cdata;
dbgmask_frame = closed_mif_fig_th;
% In dynamics background, perform mid-value filter to
renoise salt & pepper
% noise.
% find out the dynamic region in fgmask
fgmask_dbg(closed_mif_fig_th) =
fgmask_frame(closed_mif_fig_th);
fgmask_dbg_denoised = medfilt2(fgmask_dbg, [6,6]);
fgmask_frame(closed_mif_fig_th) =
fgmask_dbg_denoised(closed_mif_fig_th);
% find out the static region in fgmask
fgmask_sbg(~closed_mif_fig_th) =
fgmask_frame(~closed_mif_fig_th);
se = strel('disk',4);
fgmask_sbg_morph = imclose(fgmask_sbg, se);
fgmask_frame(~closed_mif_fig_th) =
fgmask_sbg_morph(~closed_mif_fig_th);
[X1, cm1] = gray2ind(fgmask_frame, 256);
MIFMovie(i).cdata = X1;
MIFMovie(i).colormap = cm1;
end

```

```

%% Play the result movie
figure()
set(gcf,'position',[1200 300 vidObj.Width vidObj.Height]);
set(gca,'units','pixels');
set(gca,'position',[0 0 vidObj.Width vidObj.Height]);
movie(MIFMovie, 3, vidObj.FrameRate);

```

```

%% show the result figure
frame_num = 30;
frame_res = uint8(zeros(vidObj.Height,5*vidObj.Width));
reselect_fg = rgb2gray(s(frame_num).cdata);
reselect_fg(MIFMovie(frame_num).cdata<128) = 0;
frame_res(:,0*vidObj.Width+1:1*vidObj.Width) =
rgb2gray(s(frame_num).cdata);
frame_res(:,1*vidObj.Width+1:2*vidObj.Width) =
fgmask(frame_num).cdata;
frame_res(:,2*vidObj.Width+1:3*vidObj.Width) =
255*closed_mif_fig_th;
frame_res(:,3*vidObj.Width+1:4*vidObj.Width) =
MIFMovie(frame_num).cdata;

```

```

frame_res(:,4*vidObj.Width+1:5*vidObj.Width) =
reselect_fg;
figure()
imshow(frame_res)

MIF_method.m
%% MIF_method script
% MIF_method.m tests the global median method to background
% subtraction with Moments Imaging Filter method.

%% Parameters
% mif_fig_th = (mif_fig(:, :, 2) > th_var) & (mif_fig(:, :, 3)
< th_skew);
th_var = 0.2; % threshold for variance
th_skew = 0.3; % threshold for skewness
se_r = 2; % radius of disk

%% read the video
vidObj =
VideoReader('airport/input.avi'); % Airport video

vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

s = struct('cdata', zeros(vidHeight, vidWidth, 1,
'uint8'), ...
'colormap', []);

vlength = 0;
while hasFrame(vidObj)
    vlength = vlength+1;
    s(vlength).cdata = readFrame(vidObj);
end
whos s

% convert uint data to grayscale and into double
k = 1;
spanel = zeros(vidHeight, vidWidth, vlength);
for k = 1:vlength
    spanel(:, :, k) = im2double(rgb2gray(s(k).cdata));
end
whos spanel

%% calculate background
figure()

```

```

imshow(spanel(:,:,1))

mif_fig = zeros(vidHeight, vidWidth, 3);
mif_fig(:,:,1) = mapminmax(median(spanel, 3),0,1);
mif_fig(:,:,2) = mapminmax(std(spanel, 0, 3),0,1);
mif_fig(:,:,3) = mapminmax(abs(skewness(spanel,1, 3)),0,1);

figure()
imshow(im2uint8(mif_fig))

%% each layer display
figure()
subplot(3,1,1)
imshow(mif_fig(:,:,1))
subplot(3,1,2)
imshow(mif_fig(:,:,2))
subplot(3,1,3)
imshow(mif_fig(:,:,3))

%% segmentation of RGB figure
mif_fig_th = (mif_fig(:,:,2) > th_var) & (mif_fig(:,:,3) <
th_skew);

SE = strel('disk', se_r);
closed_mif_fig_th = imclose(mif_fig_th, SE);

figure()
subplot(2,2,1)
imshow(mif_fig(:,:,2) > th_var)
subplot(2,2,2)
imshow(mif_fig(:,:,3) < th_skew)
subplot(2,2,3)
imshow(mif_fig_th)
subplot(2,2,4)
imshow(closed_mif_fig_th)

%% Display the movie
figure()
% image(s(5).cdata)
set(gcf,'position',[300 300 vidObj.Width vidObj.Height]);
set(gca,'units','pixels');
set(gca,'position',[0 0 vidObj.Width vidObj.Height]);
movie(s,1,vidObj.FrameRate);

```



```

MIFFT_method.m
%% MIFFT_method script
% MIFFT_method.m tests the global median method to background
% subtraction with Moments Imaging with FFT component method.

%% read the video
vidObj = VideoReader('Campus.avi');

vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

s = struct('cdata',zeros(vidHeight, vidWidth, 1,
'uint8'),...
'colormap',[]);

vlength = 0;
while hasFrame(vidObj)
    vlength = vlength+1;
    s(vlength).cdata = readFrame(vidObj);
end
whos s

% convert uint data to grayscale and into double
k = 1;
spanel = zeros(vidHeight, vidWidth, vlength);
for k = 1:vlength
    spanel(:, :, k) = im2double(rgb2gray(s(k).cdata));
end
whos spanel

%% calculate background
figure()
imshow(spanel(:, :, 1))

trans_fig = zeros(vidHeight, vidWidth, 3);
trans_fig(:, :, 1) = mapminmax(median(spanel, 3), 0, 1);
trans_fig(:, :, 2) = mapminmax(std(spanel, 0, 3), 0, 1);
trans_fig(:, :, 3) = abs(skewness(spanel, 1, 3));

if 0
for i = 1:vidHeight
    for j = 1:vidWidth
        ts = squeeze(spanel(i, j, :));
        fftts = fft(ts-mean(ts));
    end
end

```

```

p = abs(ffttts(1:floor(vlength/2.0)));
if (i == 20) && (j == 15)
    figure()
    subplot(2,1,1)
    plot(ts)
    ylim([0,1])
    subplot(2,1,2)
    plot(p)
    %if max(ts)-min(ts) > 0.2
    %    disp('foreground')
    %end
    disp('foreground')
    % close
end
[~, trans_fig(i,j,3)] = max(p);
% disp(['i = ',num2str(i), ';j = ',num2str(j)])
end
end
end

trans_fig(:, :, 3) = mapminmax(trans_fig(:, :, 3), 0, 1);

figure()
imshow(im2uint8(trans_fig))

%% each layer display
figure()
subplot(3,1,1)
imshow(trans_fig(:, :, 1))
subplot(3,1,2)
imshow(trans_fig(:, :, 2))
subplot(3,1,3)
imshow(trans_fig(:, :, 3))

%% Display the movie

figure()
% image(s(5).cdata)
set(gcf, 'position', [300 300 vidObj.Width vidObj.Height]);
set(gca, 'units', 'pixels');
set(gca, 'position', [0 0 vidObj.Width vidObj.Height]);
movie(s, 1, vidObj.FrameRate);

```

morph_method.m

```
% doing the morphological operation
%frame_num = 10;

% originalBW = fgpanel_th(:,:,frame_num);
originalBW = fgMovie(frame_num).cdata;
figure;
imshow(originalBW);
se = strel('disk',6);
closeBW = imclose(originalBW,se);
figure;
imshow(closeBW)

% figure();
% [h, w] = size(originalBW);
% in_a_row = zeros(h, 4*w);
% in_a_row(:,0*vidObj.Width+1:1*vidObj.Width) =
spanel(:,:,frame_num);
% in_a_row(:,1*vidObj.Width+1:2*vidObj.Width) = originalBW;
% in_a_row(:,2*vidObj.Width+1:3*vidObj.Width) = closeBW;
% in_a_row(:,3*vidObj.Width+1:4*vidObj.Width) =
closeBW.*spanel(:,:,frame_num);
% figure()
% imshow(in_a_row)

figure();
[h, w] = size(originalBW);
in_a_row = zeros(h, 4*w);
original_video =
im2double(rgb2gray(inputVideo(frame_num).cdata));
X1= im2double(closeBW).*original_video;
in_a_row(:,0*w+1:1*w) = original_video;
in_a_row(:,1*w+1:2*w) = im2double(originalBW);
in_a_row(:,2*w+1:3*w) = im2double(closeBW);
in_a_row(:,3*w+1:4*w) = X1;
figure()
imshow(in_a_row)
```

simple_median_method.m

```
%% simple_median_method script
% simple_median_method.m tests the global median method to
background
% subtraction

%% parameter
```

```

window = 0;    % windows=0 for use all frame to calculate
background
                % windows>0 for use last window number of frames
to calculate
frame_num = 30; % display the specific frame in the video
bg_frame = 10;
th_mul = 1.2; % adjust the threshold for foreground

%% read the video
vidObj =
VideoReader('waterSurface\input.avi');    %ÁÈ;ÊÓÆµÎÄ¼p
vidHeight = vidObj.Height;
vidWidth = vidObj.Width;

s = struct('cdata',zeros(vidHeight, vidWidth, 1,
'uint8'),...
    'colormap',[]);

vlength = 0;
disp(['reading original video'])
while hasFrame(vidObj)
    vlength = vlength+1;
    s(vlength).cdata = readFrame(vidObj);
    disp(['vlength = ',num2str(vlength)])
end
disp(['reading complete.'])
whos s

% convert uint data to grayscale and into double
k = 1;
spanel = zeros(vidHeight, vidWidth, vlength);
for k = 1:vlength
    spanel(:, :, k) = im2double(rgb2gray(s(k).cdata));
end
whos spanel

%% Background video through Simple Median method
disp(['calculating bg'])
bgpanel = zeros(vidObj.Height, vidObj.Width, vlength);
if window > 0
    for i = 1:vlength
        bgpanel(:, :, i) =
median(spanel(:, :, max(1, i-window):i), 3);
    end

```

```

else
    bgpanel = repmat(median(spanel(:,:1:bg_frame),
3),1,1,vlength);
end
disp(['calculating bg complete'])

%% Construct the background movie
bgMovie = struct('cdata',zeros(vidObj.Height, vidObj.Width,
1, 'uint8'),...
    'colormap',[]);
for i = 1:vlength
    [X1, cm1] = gray2ind(bgpanel(:,:,i), 256);
    bgMovie(i).cdata = X1;
    bgMovie(i).colormap = cm1;
end

%% Get the foreground target
fgpanel = abs(spanel - bgpanel);

% determine the threshold
% figure
% hold on
% plot(squeeze(max(max(fgpanel,[], 1), [], 2)))
% plot(squeeze(min(min(fgpanel,[], 1), [], 2)))
% hold off

fgpanel_th = zeros(vidObj.Height, vidObj.Width);
global_th = graythresh(fgpanel);
for i = 1:vlength
    fg_th = graythresh(fgpanel(:,:,i));
    fg_th_adj = max(min(fg_th, 1.1*global_th),
0.9*global_th);
    fgpanel_th(:,:,i) = fgpanel(:,:,i) > th_mul*fg_th_adj;
    fgpanel_th(:,:,i) = medfilt2(fgpanel_th(:,:,i), [6,6]);
    se = strel('disk',3);
    fgpanel_th(:,:,i) = imclose(fgpanel_th(:,:,i), se);
end

%for i=1:vlength
%    fgpanel_th(:,:,i) = imbinarize(fgpanel(:,:,i));
%end

%% Construct Mask
fgmask = struct('cdata',zeros(vidObj.Height, vidObj.Width,

```

```

1, 'uint8'),...
    'colormap',[]);
for i = 1:vlength
    [X1, cm1] = gray2ind(fgpanel_th(:,:,i), 256);
    fgmask(i).cdata = X1;
    fgmask(i).colormap = cm1;
end

%% Construct the foreground movie
disp(['computing fg video'])
Movie = struct('cdata', zeros(vidObj.Height, vidObj.Width, 1,
    'uint8'),...
    'colormap',[]);
for i = 1:vlength
    [X1, cm1] = gray2ind(fgpanel_th(:,:,i).*spanel(:,:,i),
256);
    Movie(i).cdata = X1;
    Movie(i).colormap = cm1;
end
disp('compute complete')

%% Display the movie

figure()
set(gcf, 'position', [300 300 vidObj.Width vidObj.Height]);
set(gca, 'units', 'pixels');
set(gca, 'position', [0 0 vidObj.Width vidObj.Height]);
movie(s, 1, vidObj.FrameRate);

figure()
set(gcf, 'position', [600 300 vidObj.Width vidObj.Height]);
set(gca, 'units', 'pixels');
set(gca, 'position', [0 0 vidObj.Width vidObj.Height]);
movie(bgMovie, 1, vidObj.FrameRate);

figure()
set(gcf, 'position', [900 300 vidObj.Width vidObj.Height]);
set(gca, 'units', 'pixels');
set(gca, 'position', [0 0 vidObj.Width vidObj.Height]);
movie(fgmask, 3, vidObj.FrameRate);

figure()
set(gcf, 'position', [1200 300 vidObj.Width vidObj.Height]);
set(gca, 'units', 'pixels');

```

```

set(gca,'position',[0 0 vidObj.Width vidObj.Height]);
movie(Movie, 3, vidObj.FrameRate);

%% output a specific frame
frame_res = uint8(zeros(vidObj.Height,4*vidObj.Width));
frame_res(:,0*vidObj.Width+1:1*vidObj.Width) =
rgb2gray(s(frame_num).cdata);
frame_res(:,1*vidObj.Width+1:2*vidObj.Width) =
bgMovie(frame_num).cdata;
frame_res(:,2*vidObj.Width+1:3*vidObj.Width) =
fgmask(frame_num).cdata;
frame_res(:,3*vidObj.Width+1:4*vidObj.Width) =
Movie(frame_num).cdata;
figure()
imshow(frame_res)

video_stab.m
% output to be of intensity only video.

% Input video file which needs to be stabilized.
filename = 'campus.avi';

hVideoSource = vision.VideoFileReader(filename, ...
    'ImageColorSpace',
    'Intensity',...
    'VideoOutputDataType',
    'double');

%%
% translation between successive video frames.
hTM = vision.TemplateMatcher('ROIInputPort', true, ...
    'BestMatchNeighborhoodOutputPort',
true);

%%
% Create a System object to display the original video and
the stabilized
% video.
hVideoOut = vision.VideoPlayer('Name', 'Video
Stabilization');
hVideoOut.Position(1) = round(0.4*hVideoOut.Position(1));
hVideoOut.Position(2) =
round(1.5*(hVideoOut.Position(2)));
hVideoOut.Position(3:4) = [650 350];

```

```

%%
% Here we initialize some variables used in the processing
loop.
pos.template_orig = [109 100]; % [x y] upper left corner
pos.template_size = [22 18]; % [width height]
pos.search_border = [15 10]; % max horizontal and vertical
displacement
pos.template_center = floor((pos.template_size-1)/2);
pos.template_center_pos = (pos.template_orig +
pos.template_center - 1);
fileInfo = info(hVideoSource);
W = fileInfo.VideoSize(1); % Width in pixels
H = fileInfo.VideoSize(2); % Height in pixels
BorderCols = [1:pos.search_border(1)+4
W-pos.search_border(1)+4:W];
BorderRows = [1:pos.search_border(2)+4
H-pos.search_border(2)+4:H];
sz = fileInfo.VideoSize;
TargetRowIndices = ...

pos.template_orig(2)-1:pos.template_orig(2)+pos.template_
size(2)-2;
TargetColIndices = ...

pos.template_orig(1)-1:pos.template_orig(1)+pos.template_
size(1)-2;
SearchRegion = pos.template_orig - pos.search_border - 1;
Offset = [0 0];
Target = zeros(18,22);
firstTime = true;

%% Stream Processing Loop
% This is the main processing loop which uses the objects we
instantiated
% above to stabilize the input video.
while ~isDone(hVideoSource)
    input = step(hVideoSource);

    % Find location of Target in the input video frame
    if firstTime
        Idx = int32(pos.template_center_pos);
        MotionVector = [0 0];
        firstTime = false;
    end
end

```



```

else
    IdxPrev = Idx;

    ROI = [SearchRegion,
pos.template_size+2*pos.search_border];
    Idx = step(hTM, input, Target, ROI);

    MotionVector = double(Idx-IdxPrev);
end

[Offset, SearchRegion] = updatesearch(sz,
MotionVector, ...
    SearchRegion, Offset, pos);

% Translate video frame to offset the camera motion
Stabilized = imtranslate(input, Offset, 'linear');

Target = Stabilized(TargetRowIndices, TargetColIndices);

% Add black border for display
Stabilized(:, BorderCols) = 0;
Stabilized(BorderRows, :) = 0;

TargetRect = [pos.template_orig-Offset,
pos.template_size];
SearchRegionRect = [SearchRegion, pos.template_size +
2*pos.search_border];

% Draw rectangles on input to show target and search region
input = insertShape(input, 'Rectangle', [TargetRect;
SearchRegionRect],...
    'Color', 'white');

% Display the offset (displacement) values on the input
image
txt = sprintf('(%+05.1f,%+05.1f)', Offset);
input = insertText(input(:,:,1),[191
215],txt,'FontSize',16, ...
    'TextColor', 'white', 'BoxOpacity', 0);

% Display video
step(hVideoOut, [input(:,:,1) Stabilized]);
end

%% Release
% Here you call the release method on the objects to close

```

```

any open files
% and devices.
release(hVideoSource);

```

imbs.hpp

```

#ifndef __IMBS_HPP__
#define __IMBS_HPP__

//OPENCV
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/imgproc/imgproc_c.h>
#include <opencv2/features2d/features2d.hpp>

//C++
#include <iostream>
#include <vector>
#include <fstream>
#include <thread>
#include <functional>

using namespace cv;
using namespace std;

class BackgroundSubtractorIMBS
{
public:
    //! the default constructor
    BackgroundSubtractorIMBS();
    //! the full constructor
    BackgroundSubtractorIMBS(double fps,
        unsigned int fgThreshold=20,
        unsigned int associationThreshold=5,
        double samplingPeriod=500.,
        unsigned int minBinHeight=2,
        unsigned int numSamples=20,
        double alpha=0.65,
        double beta=1.15,
        double tau_s=60.,
        double tau_h=40.,
        double minArea=50.,
        double persistencePeriod=10000.,

```

```

        bool morphologicalFiltering=false
    );
    //! the destructor
    ~BackgroundSubtractorIMBS();
    //! the update operator
    void apply(InputArray image, OutputArray fgmask, double
learningRate=-1.);

    //! computes a background image which shows only the
highest bin for each pixel
    void getBackgroundImage(OutputArray backgroundImage)
const;

    //! re-initiaization method
    void initialize(Size frameSize, int frameType);

    bool loadBg(const char* filename);
    void saveBg(string* filename);

private:
    //method for creating the background model
    void createBg(unsigned int bg_sample_number);
    //method for creating the incremental background model
    void createIncrementalBg(unsigned int bg_sample_number);
    //method for updating the background model
    void updateBg();
    //method for computing the foreground mask
    void getFg();
    //method for computing the incremental foreground mask
    void getIncrementalFg();
    //method for refining foreground mask
    void filterFg();
    //method for filtering out blobs smaller than a given area
    void areaThresholding();
    //method for getting the current time
    double getTimestamp();
    //method for converting from RGB to HSV
    void convertImageRGBtoHSV(const Mat& imageRGB);
    //Utils//
    //abs function
    int abs_(int _value);
    //max between three numbers
    int max_(int _a, int _b, int _c);
    //min between two numbers

```

```

int min_(int _a, int _b);

//current input RGB frame
Mat frame;
vector<Mat> frameBGR;
//frame size
Size frameSize;
//frame type
int frameType;
//total number of pixels in frame
unsigned int numPixels;
//current background sample
Mat bgSample;
vector<Mat> bgSampleBGR;
//current background image which shows only the highest
bin for each pixel
//(just for displaying purposes)
Mat bgImage;
//current foreground mask
Mat fgmask;
Mat fgIncrementalMask;
Mat fgMask_;

Mat fgfiltered;

string* bgFilename;
bool loadedBg;

//number of fps
double fps;
//time stamp in milliseconds (ms)
double timestamp;
//previous time stamp in milliseconds (ms)
double prev_timestamp;
double initial_tick_count;
//initial message to be shown until the first bg model
is ready
Mat initialMsgGray;
Mat initialMsgRGB;
Mat imageHSV;

float FLOAT_TO_BYTE;

```

```

float BYTE_TO_FLOAT;
float fhV1;
float fhV2;

//struct for modeling the background values for a single
pixel
typedef struct {
    vector<Vec3b> binValues;
    vector<uchar> binHeights;
    vector<bool> isFg;
} Bins;

vector<Bins> bgBins;
vector<Bins> incrementalBgBins;
public:
    //struct for modeling the background values for the entire
frame
    typedef struct {
        vector<Vec3b> values;
        vector<bool> isValid;
        vector<bool> isFg;
        vector<uchar> counter;
    } BgModel;

    //bool isBackgroundCreated;
private:
    vector<BgModel> bgModel;
    vector<BgModel> incrementalBgModel;
    //method for suppressing shadows and highlights
    void hsvSuppression(const vector<BgModel> &_bgModel);

    //SHADOW SUPPRESSION PARAMETERS
    float alpha;
    float beta;
    uchar tau_s;
    uchar tau_h;

    unsigned int minBinHeight;
    unsigned int numSamples;
    unsigned int incrementalNumSamples;
    unsigned int samplingPeriod;
    unsigned int incrementalSamplingPeriod;
    unsigned int increamental_reduction_variable;
    unsigned long prev_bg_frame_time;

```

```

unsigned long prev_incremental_bg_frame_time;
unsigned int bg_frame_counter;
unsigned int incremental_bg_frame_counter;
unsigned int associationThreshold;
unsigned int maxBgBins;
unsigned int nframes;

double minArea;
bool bg_reset;
unsigned int persistencePeriod;
bool prev_area;
bool incremental_bg;
unsigned int fgThreshold;
uchar SHADOW_LABEL;
uchar PERSISTENCE_LABEL;
uchar FOREGROUND_LABEL;
//persistence map
vector<unsigned int> persistenceMap;
Mat persistenceImage;

vector<unsigned int> counters;
vector<unsigned int> incrementalCounters;
vector<unsigned int> counterCopy;

bool morphologicalFiltering;

public:
    unsigned int getMaxBgBins() {
        return maxBgBins;
    }
    unsigned int getFgThreshold() {
        return fgThreshold;
    }
    void getBgModel(vector<BgModel> &bgModel_copy);
};

#endif // __IMBS_HPP__

```