**大华软件大赛复赛赛题**

——大华前端相机运动检测算法

# 题目背景

运动检测是视频处理过程中最基本的一个环节，精准地找到运动区域，有助于在视频编码上调整各区域的压缩比，减小编码后码流大小；在视频分析上通过对运动区域进行检测，将运动目标快速识别提取，以便对运动区域进一步识别。

运动检测可在相机端作为一个基本的功能需求，俗称“动检”：主要用于检测连续视频数据中的运动区域，也可以作为一个独立功能剥离出来，独立呈现给客户。

# 题目概述

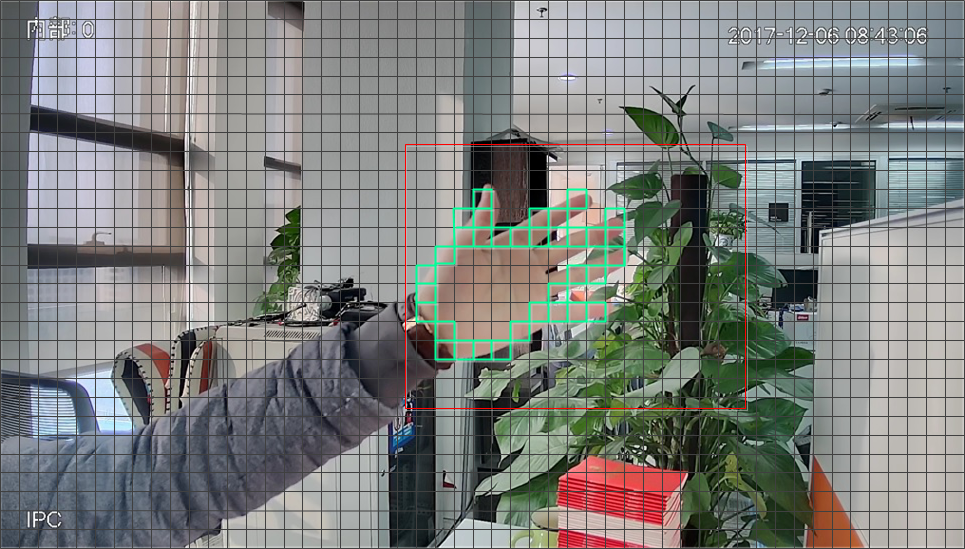
运动检测，英文翻译为”Motion detection technology”，常用于无人职守监控录像和自动报警。通过摄像头按照不同分辨率采集得到的图像会被CPU按照一定算法进行计算和比较，当画面有变化时，如有人走动，镜头被移动，计算比较结果得出的数字会超过阈值并指示系统能自动做出相应的处理。

运动检测允许在指定区域能识别图像的变化，检测运动物体的存在并避免由光线变化带来的干扰。由于背景图像的动态变化，如天气、光照、影子及混乱干扰等的影响，运动检测通常是一项相当困难的工作。

在视频编码领域方面，有I帧、P帧的概念，通过视频帧的比较来检测图像变化。其中I帧为关键帧，作为固定背景，后续的帧与I帧进行比较，对比较的差异部分进行编码形成P帧。

运动检测可以降低公共机构和企业场所的人工监控成本，并且避免人员长期值守疲劳导致的监察失误，可以极大地提高监控效率和监控精度。

为了将运动检测更直观地呈现，请参照下图：红色为检测区域，绿色为检测出存在动检的区块。



# 基础知识

在日常生活中，我们知道电影是一张张单独的胶片构造而成，经过快速播放，使得人眼感觉出画面的连续性。将电源胶片一张张对比，会发现相邻两张胶片的画面内容非常接近，只有局部发生了细微了变化。

在理想条件下，前后两幅画面除了具有运动物体的那部分区域像素发生了变化，其它相同位置上的像素几乎差异不大，近似忽略。

对于找出运动区域，最快捷的方法就是前后两副画面中相同坐标的像素进行相减，得到差值，存在差值大于0的像素位置，就表明画面发生变化，差值越大表明图像发生的变化越剧烈。

表示一个像素的颜色，可以采用RGB的方式，也可以用YUV的方式。

RGB为红绿蓝三基色，任何一种颜色都可以用这三种基础颜色通过不同的比例进行组合；YUV格式另外一种图像数据表示方法，其中Y表示亮度信号，UV表示色彩信号，一张只有Y数据的图像为一张灰度图，Y的取值范围为[0, 255]。

假设一张图片宽高用w、h表示，char data[w][h]表示一张黑白图像，data[x][y]表示第y行第x列上的亮度值，假设data[x][y]=90，则该像素的颜色对应RGB值为(90,90,90)。

# 目前常用的方法

由于现实环境中，受到自然环境外接的各种干扰（比如光的折射、反色，太阳在持续运动，大气层空气流动等），图像采集模块作为一个电子元器件，感受到的外部光子电荷是存在差异的，所以人眼看上去没有发生变化的两幅图像，在数值上也是存在略微差异。

在安防监控领域中，检测到运动目标后，可以联动报警（鸣笛），而找出“有意义的运动区域”需要过滤掉一些不必要的干扰，比如秋天被风吹动的落叶，因此报警触发条件需要根据一定的策略方法进行过滤。

# 背景减除

背景减除方法是目前运动检测中最常用的一种方法，它是利用当前图像与背景图像的差分来检测出运动区域的一种技术。它一般能够提供最完全的特征数据，却对于动态场景的变化，如光照和外来无关事件等干扰特别敏感。

# 时间差分

时间差分（又称相邻帧差）方法，是在连续的图像序列中两个或三个相邻帧间，采用基于像素的时间差分并且阈值化来提取图像中的运动区域。时间差分运动检测方法对于动态环境有较强的自适应性，但一般不能完整的提取出所有相关的特征像素点，在运动实体内部容易产生空洞现象。

# 光流

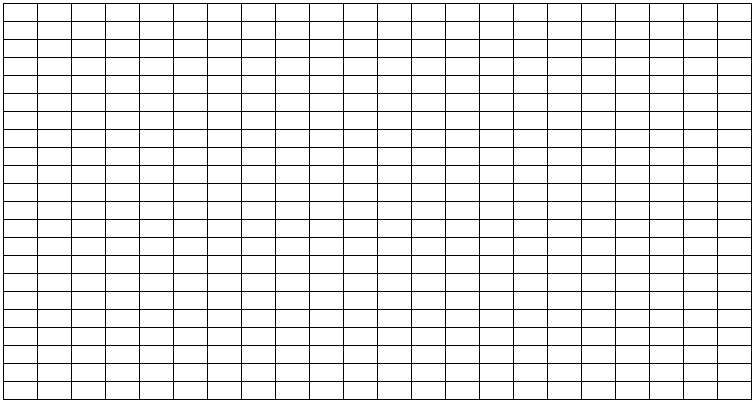
基于光流方法的运动检测，采用了运动目标随时间变化的光流特性。比如通过计算位移向量来初始化基于轮廓的跟踪算法，从而有效地提取和跟踪运动目标。该方法的要点在于摄像机运动（不明意义）前提下也能检测出独立的运动目标。然而，大部分的光流技术方法相当复杂，且抗噪性能差，如果没有特别的硬件装置则不能被应用于全帧视频流的实时处理。

# 其它

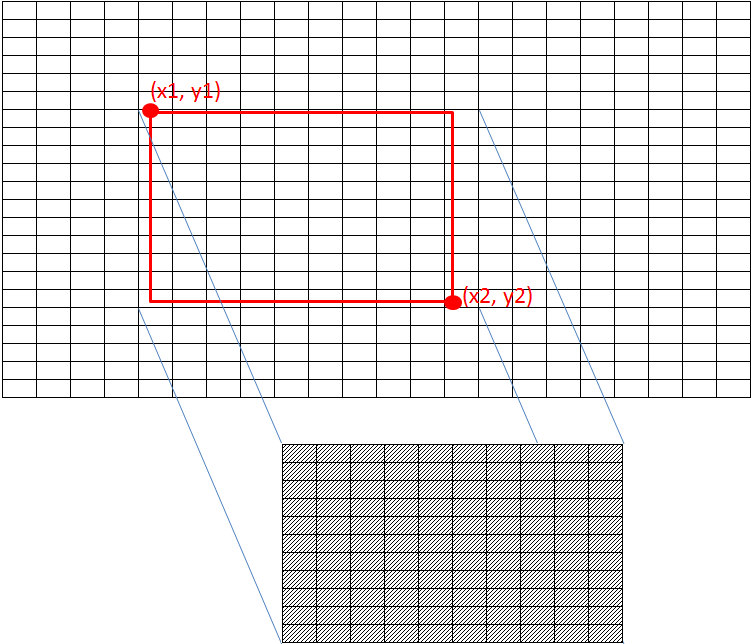
在运动检测中还有一些其它的方法，如运动向量检测法等。

# 题目说明

对于大华前端摄像机中的动检功能，其特征为将一副画面分割成MxN的矩形单元格。



用户可以选定一个矩形区域，对矩形区域所包含的单元格进行检测。该矩形区域称为检测区域

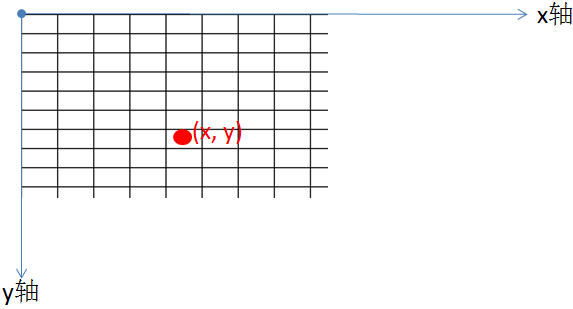


是否判别为触发动检，会有两个参数进行设定，一个是阈值，一个是灵敏度。

* 灵敏度——指单个单元格内图像发生变化的等级过滤值，灵敏度范围为[1,10]，算法自行决定设定各等级对应的图像变化剧烈程度，当该单元格的变化程度达到用户设定的灵敏度等级，则认为该单元格发生了动检。
* 阈值——指的是动检检测区域面积阈值，如上图红色矩形框中选定的单元格为10x11，总共110个单元格，阈值采用百分比表示，10%表示只要110个单元格中有11个单元格在指定的灵敏度下被认为发生了动检，则最终认定为该区域发生了动检。

# 参数设定

画面左上角为坐标原点，水平方向为x轴，纵向为y轴。



参赛者提交的代码会编译成静态库（需要同步提供源码），供测试程序调用，其流程为：



其中分割行列数MxN，检测区为MxN的子区域，该子区域最大选定单元格矩阵为128x128。同一段视频会使用不同的参数进行测试验证。

测试程序代码简单示例：测试视频分辨率1920x1080，分割成108行192列的10x10的矩形单元格，检测的子区域左上角坐标（0,0），右下角坐标（50,50）,灵敏度10，阈值50%。

……

// FRAME\_SIZE表示一帧图象的大小,以1080P为例, FRAME\_SIZE =1920\*1080\*3/2

char yuvData[FRAME\_SIZE];

char resultMatrix[128][128]; //检测区域最大128x128

bool alarmResult;

FILE \*InputFile;

set\_resolution(1920, 1080);

set\_cell\_split(108, 192);

set\_detection\_region(0, 0, 192, 108 );

set\_threshold\_sensity(10,50);

InputFile = fopen(“D:\\test.yuv”, “rb+”); //确保测试时文件打开成功

……

for (int i=1; i<=100; i++) //确保输入文件为100帧

{

fread(yuvData,sizeof(char),sizeof(yuvData), InputFile);

yuv\_process(yuvData, resultMatrix, &alarmResult);

……

//将每一帧运动检测结果矩阵和报警结果保存到文件用于后续评分

……

}

……

以上为测试程序的示例，测试程序用于验证选手程序运行结果，若选手提交代码的关键接口设计未按照题目要求，导致测试程序无法运行则判定为0分。

# 关键接口设计

* set\_resolution(int width,int height)

|  |  |  |
| --- | --- | --- |
| 参数 | IN/OUT | 描述 |
| width | IN | 检测视频画面的帧宽度 |
| height | IN | 检测视频画面的帧高度 |

* set\_cell\_split(int row,int col)

|  |  |  |
| --- | --- | --- |
| 参数 | IN/OUT | 描述 |
| row | IN | 一帧画面划分的行数 |
| col | IN | 一帧画面划分的列数 |

* set\_detection\_region(int x1,int y1, int x2,int y2)

|  |  |  |
| --- | --- | --- |
| 参数 | IN/OUT | 描述 |
| x1 | IN | 检测区域左上角横坐标 |
| y1 | IN | 检测区域左上角纵坐标 |
| x2 | IN | 检测区域右下角横坐标 |
| y2 | IN | 检测区域右下角纵坐标 |

* set\_threshold\_sensity(int t,int s)

|  |  |  |
| --- | --- | --- |
| 参数 | IN/OUT | 描述 |
| t | IN | 检测灵敏度 |
| s | IN | 检测阈值,采用百分数表示，例如s=50，代表50% |

* yuv\_process(const char \*data, char result\_matrix[128][128], bool \*alarm)

|  |  |  |
| --- | --- | --- |
| 参数 | IN/OUT | 描述 |
| data | IN | 图像数据，数据格式为一个维护数组data[w][h]，其组织形式为：  data[x][y]表示在(x,y)坐标上的像素亮度。 |
| result\_matrix | OUT | 结果矩阵，result\_matrix[0][0]为子区域左上角第一个单元格的结果。  result\_matrix[x][y]表示子区域内第[x,y]单元格的是否发生变化。  该结果矩阵可根据灵敏度进行筛选，参见灵敏度描述。 |
| alarm | OUT | 最终裁定结果，结合灵敏度、面积阈值判定检测区域内是否符合产生报警条件，返回0表示无报警，返回1表示有报警。对于C++直接使用bool，对于C参考使用#define bool int，最终根据返回值0和1判断。 |

# 输入参数

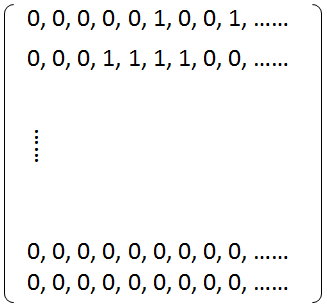
1. 一帧画面分割为矩形单元格的行列数（MxN）；
2. 检测区域（基于原始画面的坐标，原始画面左上角为原始坐标(0, 0)）：左上角坐标(x1, y1)和右下角坐标(x2, y2)，0≤x1<x2<127 ，0≤y1<y2<127；
3. 报警面积阈值threshold，0<threshold<100；
4. 单元格灵敏度sensity，1≤sensity≤10；
5. YUV视频图像分辨率width，height；

# 测试数据

测试数据为10段视频数据，格式为YUV420P格式，每段视频分辨率各异。

# 输出参数

输出检测区域的二维矩阵，其中1表示该单元格有运动，0表示没有运动。



同时输出是否满足面积阈值，满足为1，不满足为0.

# 编程语言

支持C，C++。GCC版本：4.4.7，G++版本：4.4.7，GNU make版本：3.81。

# 评判标准

参赛者提供源代码，对其进行编译运行测试。为了方便参赛者自行调试验证，主办方会提供10段不同的场景（刮风、下雨、灯照、夜晚、晴朗……）录像，最终考核会采用另外10段录像进行验证。

**【场景录像下载地址：https://pan.baidu.com/s/1BenaqORNfSw4G5kXlwKx8A】**

最终考核平台为主频为1GHz的ARM单核处理器，考核时会对50组测试用例进行评分，单组测试用例产生方法为，从10段考核备用录像中选取1段，并从中选取连续100帧视频数据，设置满足3.1.2章节中定义的输入参数，单组测试用例的评分标准如下：

* + 检测结果准确率(B)；一组测试视频共100帧，比较每帧处理后的输出参数和预先测试的标准结果比较；
  + 同一段视频数据(100帧)在相同参数下运行10次，计算平均每帧耗时（精确到毫秒）(cost\_ms)；
  + 不同参数下所耗内存（精确到字节, 基准为分辨率WxH个字节）(cost\_mem)；

单组测试用例成绩=(Bx100) x (cost\_ms + cost\_mem)。满分为200

最终考核成绩=50组测试用例成绩总和/50，满分为200；当参赛选手得分相同时，则查看源码从算法实用性和代码书写风格等方面进行排名。

|  |  |
| --- | --- |
| 考核点  得分 | 耗时（单位：毫秒） |
| 10 | ≤180 |
| 20 | ≤150 |
| 30 | ≤120 |
| 40 | ≤100 |
| 50 | ≤90 |
| 60 | ≤80 |
| 70 | ≤70 |
| 80 | ≤60 |
| 90 | ≤50 |
| 100 | ≤40 |

|  |  |
| --- | --- |
| 考核点  得分 | 内存（单位：字节） |
| 10 | ≤4.0 \*基准 |
| 20 | ≤3.0 \*基准 |
| 30 | ≤2.0 \*基准 |
| 40 | ≤1.5 \*基准 |
| 50 | ≤1.0 \*基准 |
| 60 | ≤0.8 \*基准 |
| 70 | ≤0.6 \*基准 |
| 80 | ≤0.4 \*基准 |
| 90 | ≤0.2 \*基准 |
| 100 | ≤0.1 \*基准 |

示例：检测数据分辨率为704x576，测试结果为检测率达到90%，平均耗时50ms，所需内存1MB。

* + 内存方面占比为(1024\*1024)/(704\*576)=2.586，所以占比20%
  + 耗时为50ms，占比为90%
  + 得分=(90%x (90 + 20) =99分