ViBe.h（头文件，一般做申明函数、类使用，不做具体定义）

1. #pragma once
2. #include <iostream>
3. #include "opencv2/opencv.hpp"
5. **using namespace cv;**
6. **using namespace std;**
8. #define NUM\_SAMPLES 20      //每个像素点的样本个数
9. #define MIN\_MATCHES 2       //#min指数
10. #define RADIUS 20       //Sqthere半径
11. #define SUBSAMPLE\_FACTOR 16 //子采样概率，决定背景更新的概率

14. **class ViBe\_BGS**
15. {
16. **public:**
17. ViBe\_BGS(**void);  //构造函数**
18. ~ViBe\_BGS(**void);  //析构函数，对开辟的内存做必要的清理工作**
20. **void init(const Mat \_image);   //初始化**
21. **void processFirstFrame(const Mat \_image); //利用第一帧进行建模**
22. **void testAndUpdate(const Mat \_image);  //判断前景与背景，并进行背景跟新**
23. Mat getMask(**void){return m\_mask;};  //得到前景**
25. **private:**
26. Mat m\_samples[NUM\_SAMPLES];  //每一帧图像的每一个像素的样本集
27. Mat m\_foregroundMatchCount;  //统计像素被判断为前景的次数，便于跟新
28. Mat m\_mask;  //前景提取后的一帧图像
29. };

ViBe.cpp（上面所提到的申明的具体定义）

1. #include <opencv2/opencv.hpp>
2. #include <iostream>
3. #include "ViBe.h"
5. **using namespace std;**
6. **using namespace cv;**
8. **int c\_xoff[9] = {-1,  0,  1, -1, 1, -1, 0, 1, 0};  //x的邻居点，9宫格**
9. **int c\_yoff[9] = {-1,  0,  1, -1, 1, -1, 0, 1, 0};  //y的邻居点**
11. ViBe\_BGS::ViBe\_BGS(**void)**
12. {
14. }
15. ViBe\_BGS::~ViBe\_BGS(**void)**
16. {
18. }
20. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Assign space and init \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
21. **void ViBe\_BGS::init(const Mat \_image)  //成员函数初始化**
22. {
23. **for(int i = 0; i < NUM\_SAMPLES; i++) //可以这样理解，针对一帧图像，建立了20帧的样本集**
24. {
25. m\_samples[i] = Mat::zeros(\_image.size(), CV\_8UC1);  //针对每一帧样本集的每一个像素初始化为8位无符号0，单通道
26. }
27. m\_mask = Mat::zeros(\_image.size(),CV\_8UC1); //初始化
28. m\_foregroundMatchCount = Mat::zeros(\_image.size(),CV\_8UC1);  //每一个像素被判断为前景的次数，初始化
29. }
30. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Init model from first frame \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
31. **void ViBe\_BGS::processFirstFrame(const Mat \_image)**
32. {
33. RNG rng;    //随机数产生器
34. **int row, col;**
35. **for(int i = 0; i < \_image.rows; i++)**
36. {
37. **for(int j = 0; j < \_image.cols; j++)**
38. {
39. **for(int k = 0 ; k < NUM\_SAMPLES; k++)**
40. {
41. // Random pick up NUM\_SAMPLES pixel in neighbourhood to construct the model
42. **int random = rng.uniform(0, 9);  //随机产生0-9的随机数，主要用于定位中心像素的邻域像素**
43. row = i + c\_yoff[random]; //定位中心像素的邻域像素
44. **if (row < 0)   //下面四句主要用于判断是否超出边界**
45. row = 0;
46. **if (row >= \_image.rows)**
47. row = \_image.rows - 1;
48. col = j + c\_xoff[random];
49. **if (col < 0)    //下面四句主要用于判断是否超出边界**
50. col = 0;
51. **if (col >= \_image.cols)**
52. col = \_image.cols - 1;
53. m\_samples[k].at<uchar>(i, j) = \_image.at<uchar>(row, col);  //将相应的像素值复制到样本集中
54. }
55. }
56. }
57. }
59. /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Test a new frame and update model \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/
60. **void ViBe\_BGS::testAndUpdate(const Mat \_image)**
61. {
62. RNG rng;
63. **for(int i = 0; i < \_image.rows; i++)**
64. {
65. **for(int j = 0; j < \_image.cols; j++)**
66. {
67. **int matches(0), count(0);**
68. **float dist;**
69. **while(matches < MIN\_MATCHES && count < NUM\_SAMPLES) //逐个像素判断，当匹配个数大于阀值MIN\_MATCHES，或整个样本集遍历完成跳出**
70. {
71. dist = abs(m\_samples[count].at<uchar>(i, j) - \_image.at<uchar>(i, j)); **//当前帧像素值与样本集中的值做差，取绝对值**
72. **if (dist < RADIUS)  //当绝对值小于阀值是，表示当前帧像素与样本值中的相似**
73. matches++;
75. count++;  //**取样本值的下一个元素作比较**
76. }
77. **if (matches >= MIN\_MATCHES)  //匹配个数大于阀值MIN\_MATCHES个数时，表示作为背景**
78. {
79. // It is a background pixel
80. m\_foregroundMatchCount.at<uchar>(i, j) = 0;  //被检测为前景的个数赋值为0
81. // Set background pixel to 0
82. m\_mask.at<uchar>(i, j) = 0;  //该像素点值也为0
83. // 如果一个像素是背景点，那么它有 1 / defaultSubsamplingFactor 的概率去**更新自己的模型样本值**
84. **int random = rng.uniform(0, SUBSAMPLE\_FACTOR);   //以1/defaultSubsamplingFactor概率跟新背景**
85. **if (random == 0)**
86. {
87. random = rng.uniform(0, NUM\_SAMPLES);
88. m\_samples[random].at<uchar>(i, j) = \_image.at<uchar>(i, j); //**以上述概率把该像素的值作为它的一个背景值。**
89. }
90. **// 同时也有 1 / defaultSubsamplingFactor 的概率去更新它的邻居点的模型样本值**
91. random = rng.uniform(0, SUBSAMPLE\_FACTOR);
92. **if (random == 0)**
93. {
94. **int row, col;**
95. random = rng.uniform(0, 9);
96. row = i + c\_yoff[random];
97. **if (row < 0)   //下面四句主要用于判断是否超出边界**
98. row = 0;
99. **if (row >= \_image.rows)**
100. row = \_image.rows - 1;
101. random = rng.uniform(0, 9);
102. col = j + c\_xoff[random];
103. **if (col < 0)   //下面四句主要用于判断是否超出边界**
104. col = 0;
105. **if (col >= \_image.cols)**
106. col = \_image.cols - 1;
108. random = rng.uniform(0, NUM\_SAMPLES);
109. m\_samples[random].at<uchar>(row, col) = \_image.at<uchar>(i, j);  **//模型样本值更新**
110. }
111. }
112. **else  //匹配个数小于阀值MIN\_MATCHES个数时，表示作为前景**
113. {
114. // It is a foreground pixel
115. m\_foregroundMatchCount.at<uchar>(i, j)++;  **//检测为前景的个数加1**
116. // Set background pixel to 255
117. m\_mask.at<uchar>(i, j) =255;    **//前景点用白色（255）表示**
119. //如果某个像素点连续N次（这里为50次）被检测为前景，则认为一块静止区域被误判为运动，将其更新为**背景点**
120. **if (m\_foregroundMatchCount.at<uchar>(i, j) > 50)**
121. {
122. **int random = rng.uniform(0, SUBSAMPLE\_FACTOR);**
123. **if (random == 0)**
124. {
125. random = rng.uniform(0, NUM\_SAMPLES);
126. m\_samples[random].at<uchar>(i, j) = \_image.at<uchar>(i, j);
127. }
128. }
129. }
130. }
131. }
132. }

main.cpp（你懂的……大笑）

1. #include <opencv2/opencv.hpp>
2. #include "ViBe.h"
3. #include <iostream>
4. #include <cstdio>
5. #include<stdlib.h>
6. **using namespace cv;**
7. **using namespace std;**
9. **int main(int argc, char\* argv[])**
10. {
11. Mat frame, gray, mask;
12. VideoCapture capture;
13. capture.open("E:\\overpass\\11.avi");    //输入视频地址
15. **if (!capture.isOpened())**
16. {
17. cout<<"No camera or video input!\n"<<endl;
18. **return -1;**
19. }
21. ViBe\_BGS Vibe\_Bgs; //定义一个背景差分对象
22. **int count = 0; //帧计数器，统计为第几帧**
24. **while (1)**
25. {
26. count++;
27. capture >> frame;
28. **if (frame.empty())**
29. **break;**
30. cvtColor(frame, gray, CV\_RGB2GRAY); //转化为灰度图像
32. **if (count == 1)  //若为第一帧**
33. {
34. Vibe\_Bgs.init(gray);
35. Vibe\_Bgs.processFirstFrame(gray); //背景模型初始化
36. cout<<" Training GMM complete!"<<endl;
37. }
38. **else**
39. {
40. Vibe\_Bgs.testAndUpdate(gray);
41. mask = Vibe\_Bgs.getMask();    //计算前景
42. morphologyEx(mask, mask, MORPH\_OPEN, Mat());   //形态学处理消除前景图像中的小噪声，这里用的开运算
43. imshow("mask", mask);
44. }
46. imshow("input", frame);
48. **if ( cvWaitKey(10) == 'q' )    //键盘键入q,则停止运行，退出程序**
49. **break;**
50. }
51. system("pause");
52. **return 0;**
53. }