1. **调用方法：**
2. VideoDenoisingME :: processing

**输入参数：**

vector<cv::Mat>srcFrames：输入视频（uchar）

vector<cv::Mat>&dstFrames,：输出视频

int K,：算法参数K，取11

int temporalWindowSize,：时间窗口大小，推荐11

int patchWindowSize：比较模块窗口大小，推荐7

**输出：**

无

1. VideoDenoisingME :: videoDenoising

**输入参数**

vector<cv::Mat>framesSrc: 输入视频片段，大小为temporalWindowSize

cv::Mat&framesOut：将视频片断中间一帧的图像处理后的结果

int \_K, 算法参数K，取11

int temporalWindowSize, 时间窗口大小，推荐11

int patchWindowSize，比较模块窗口大小，推荐7

调用方法二每一帧都需要20min，方法一效果见性能。

**二、 性能:**

环境：i7-2365QM，2.0GHz， cpu运行

对象：480\*640，3通道的视频

时间：处理第一帧需要20min左右，后面每帧需要12min，实测处理137帧视频用时：30 hours。

分析：第一帧需要时间长原因在于需要初始化2H+1帧之间的光流，而后面处理的时候只需要计算最新一帧与前一帧的光流。：

比较：Opencv的fastNLMColorMuti算法所需时间为：70min

**三、效果：**

原图



resource.avi

处理后



视频；result.avi

**四、 注意事项**：

1. 多线程操作基于OpenMP，需要在vs项目属性里将openMP选项勾上。不勾不会影响编译，但会影响速度。
2. 项目里不能直接使用using namespace cv，不然opencv中的Vector类会跟光流算法中定义的Vector类冲突。
3. DImage是光流算法里定义的数据类型，使用方法与IplImage有些类似。

**五、 引用：**

光流算法直接引用自http://people.csail.mit.edu/celiu/OpticalFlow/，调用了OptialFlow:: Coarse2FineFlow函数。