以下是两个代码在不同方面的主要区别：

**设计思路和风格**  
　　第一个代码采用面向对象的设计，将视频稳定化的各个环节封装在一个类中。所有操作（如输入输出初始化、计算帧间变换、逐帧处理、释放资源等）都以独立方法的形式存在，并通过一个execute方法来串联整个流程。相比之下，第二个代码则使用过程式编程，将所有操作集中在一个单独的函数中，所有处理步骤（从视频读取到写入、预览和退出）都在一个函数里完成。

**代码结构**  
　　第一个代码的结构较为模块化，逻辑分层清晰，便于后续扩展和维护。每个功能步骤都有自己独立的方法，且通过类的实例管理内部状态。第二个代码则把所有操作放在一个函数内，代码较为集中，逻辑上没有明确拆分成多个模块，因此在扩展新功能时可能需要对整个函数做较大改动。

**变量命名**  
　　第一个代码中，变量命名更具描述性，例如使用src\_filename、dst\_filename来表示输入输出视频文件；使用vid\_capture、vid\_writer来表示视频捕获和写入对象；处理过程中使用gray\_prev\_frame、gray\_curr\_frame、curr\_color\_frame分别表示前后灰度和彩色帧，累积变换矩阵命名为accumulated\_matrix。第二个代码的变量命名更简洁，如用input\_path、output\_filename表示文件路径，用cap和out表示视频捕获和写入对象；光流跟踪部分则用prev\_pts、curr\_pts以及进一步筛选后的prev\_good、curr\_good来表示角点；累计变换矩阵命名为cumulative\_transform。两者命名虽都能表达含义，但第一个代码在描述性上更强，而第二个则更偏向于常见的缩写和简洁表达。

**错误处理与用户交互**  
　　在错误处理方面，两个代码都在无法打开视频或读取第一帧时给出提示。不过，第一个代码是在初始化IO方法中直接打印错误信息，而第二个代码在读取第一帧失败后不仅打印提示，还及时释放资源后返回。用户交互上，第一个代码仅通过cv2.waitKey(1)刷新预览窗口，没有提供退出按键检测；而第二个代码在显示校正后的视频帧时增加了对“q”键的检测，使得用户可以主动终止预览窗口，这在交互体验上更友好。

**可扩展性与维护性**  
　　面向对象的第一个代码由于模块化程度高，易于后续功能扩展和代码维护；例如若需要添加其他稳定化算法，只需在类中增加方法或继承扩展。第二个代码由于所有功能集中在一个函数中，虽然实现简单直接，但在遇到更复杂的场景或需求变动时，维护和扩展的难度会相对较大。

总的来说，两个代码实现的视频稳定化算法在核心流程上大致相同，但在设计风格、结构组织、变量命名、错误处理和用户交互等方面存在明显区别。第一个代码更适合需要长期维护、扩展性要求高的项目；而第二个代码则适合一次性脚本或简单场景。