Journal of System Simulation

Volume 28 | Issue 10 Article 19

8-13-2020

ARM-Based Real-Time Video Stabilization Enhancement Algorithm

Xiaoquan Chen
School of Computer Science, Beijing Institute of Technology, Beijing 100081, China;

Zhang Lei
School of Computer Science, Beijing Institute of Technology, Beijing 100081, China;

Huang Hua
School of Computer Science, Beijing Institute of Technology, Beijing 100081, China;

ARM-Based Real-Time Video Stabilization Enhancement Algorithm

Abstract

Abstract: Considering the performance characteristic of ARM processor, an ARM-based real-time video stabilization enhancement algorithm was proposed that identified shakiness of video and implements motion smoothing in the space of geometric transformations. The algorithm *identified the shakiness of video with the amplitudes' change of translation and rotation*. Then the optimal path could be obtained by *linear interpolation* on translational vector and rotational angle. Stable frames were generated by implementing motion compensation of translation and rotation transformations respectively. Because the algorithm used linear interpolation in the space of rigid transformations, it was much faster than the previous algorithms. Moreover, it could control cropping ratio in motion compensation and guarantee no extra distortion drawn into the warped frames. Experiments show that the algorithm can stabilize video based on ARM platform in real time and obtain good video stabilization as well.

Keywords

video stabilization, real-time, rigid transformation, shakiness detection

Recommended Citation

Chen Xiaoquan, Zhang Lei, Huang Hua. ARM-Based Real-Time Video Stabilization Enhancement Algorithm[J]. Journal of System Simulation, 2016, 28(10): 2423-2430.

第 28 卷第 10 期 系统仿真学报© Vol. 28 No. 10
2016 年 10 月 Journal of System Simulation Oct., 2016

基于 ARM 的视频流实时稳像增强算法

陈晓权，张磊，黄华
（北京理工大学计算机学院，北京 100081）

摘要：针对 ARM 处理器的性能特点，提出一种在几何变换空间内进行视频抖动检测和运动平滑的高效视频稳像增强算法。首先*通过平移和旋转分量的振幅变化检测视频流中出现的抖动*；然后对平移向量和旋转角度进行*线性插值*，获得平滑路径的最优变换；最后分别对平移和旋转变换进行运动补偿，生成稳定的视频帧。由于采用刚性变换空间中的线性插值，该算法的计算速度要远高于以往算法，而且可以有效控制补偿后视频帧的裁剪率并保证不会出现画面扭曲现象。实验结果表明，该算法能够在基于 ARM 处理器的平台上实现视频流实时稳像处理，并输出令人满意的稳定视频。

关键词：视频稳像；实时；刚性变换；抖动检测
中图分类号：TP391 文献标识码：A 文章编号：1004-731X (2016) 10-2423-08

ARM-Based Real-Time Video Stabilization Enhancement Algorithm

Chen Xiaoquan, Zhang Lei, Huang Hua

(School of Computer Science, Beijing Institute of Technology, Beijing 100081, China)

Abstract: Considering the performance characteristic of ARM processor, an ARM-based real-time video stabilization enhancement algorithm was proposed that identified shakiness of video and implements motion smoothing in the space of geometric transformations. The algorithm *identified the shakiness of video with the amplitudes' change of translation and rotation*. Then the optimal path could be obtained by *linear interpolation* on translational vector and rotational angle. Stable frames were generated by implementing motion compensation of translation and rotation transformations respectively. Because the algorithm used linear interpolation in the space of rigid transformations, it was much faster than the previous algorithms. Moreover, it could control cropping ratio in motion compensation and guarantee no extra distortion drawn into the warped frames. Experiments show that the algorithm can stabilize video based on ARM platform in real time and obtain good video stabilization as well.

Keywords: video stabilization; real-time; rigid transformation; shakiness detection

引言

随着视频摄像设备的普及化，视频稳像作为一种能够消除视频抖动以增强视觉质量的技术，已经被广泛而深入的研究和应用[1-2]。但是，已有的方法大多数是针对抖动视频进行后期处理。因此，比起时间消耗它们往往更加侧重于稳像的效果，而鲜少关注算法的实时性以及自动的视频抖动检测。然而，针对视频流的实时稳像处理也是普遍存在。比如，使用无法搭载云台等机械稳像器的小型无人机进行视频拍摄时，往往需要同步获取稳定的视频。然而，小型平台上装载的处理器，如 ARM 处理器，由于采用精简指令集(RISC)

收稿日期：2016-05-28 修回日期：2016-07-12；基金项目：国家自然科学基金面上项目(61472035)；
作者简介：陈晓权(1992-)，男，浙江东阳，硕士生，研究方向为图像与视频处理；张磊(1981-)，男，山东宁阳，副教授，研究方向为计算机图形学、图像与视频处理；黄华(1975-)，男，湖北当阳，教授，研究方向为模式识别、图像与视频处理。

的原因，并不善于处理计算复杂、综合性强的任务，尤其在浮点计算能力方面要远逊于使用复杂指令集(CISC)的处理器。因此需要研究更高效的视频流实时稳像算法，以满足基于 ARM 处理器等的平台使用需求。

本文研究基于 ARM 的视频流实时稳像增强算法。主要贡献在于以下两点：

1) 针对 ARM 不适合求解大规模浮点型最小二乘优化的问题，提出一种基于显式几何插值的运动路径平滑和补偿方法。通过对平移向量、旋转角度等内在运动几何量的直接插值，实现平滑路径和运动补偿的快速计算，满足 ARM 实时处理的需求。

2) 在视频流稳像过程中引入抖动检测。只对视频流中出现的抖动进行针对性处理，减少非抖动视频片段的稳像计算，更适合于 ARM 平台上视频流处理。

1 相关工作

根据运动模型路径维数的不同，视频稳像技术可以分为 2D 方法、3D 方法、以及 2.5D 方法。2D 方法通过平移模型，仿射模型[3]等能够近似模拟相机实际运动的数学模型来描述 2D 图像之间的运动转换关系。在此基础上，使用基于像素点[4]或者特征点[5]的方法进行运动估计，而后通过各种低通滤波器，如卡尔曼滤波器[6]，来消除运动路径中的抖动成分。2D 方法实现简单，计算量小，鲁棒性高，但是由于 2D 运动模型无法确切描述相机在三维空间中的运动信息，所以稳像效果往往不佳，不适合于直接在 ARM 平台使用。

与此相对的是，3D 方法试图还原相机在空间中的真实运动信息，包括位置和朝向。然而，这需要用到计算代价较高的 SfM 算法[7-9]对视频图像序列进行三维重建。因此，3D 方法虽然能获得比 2D 方法更好的视频稳像效果，但是时间消耗过大，且鲁棒性低。为了能够在合理的时间复杂度内获得尽可能好的稳像效果，2.5D 方法应运而生。2.5D 方法不显式地重构相机运动的 3D 结构，而是通过增加一些空间关系约束来提高 2D 方法的稳像性能，比如，添加时空约束条件来平滑特征轨迹曲线[10]，使用局部单应变换构造多条相机路径来提高运动估计的鲁棒性[11]，在低维子空间内近似表示图像运动轨迹矩阵[2]，通过极线几何约束重构相机运动[12]等。所以，2.5D 方法不仅可以获得堪比 3D 方法的视频稳像效果，而且计算也较快。但是由于广泛采用浮点型最小二乘优化求解，也不适合 ARM 平台的移植和应用。

2 基于几何插值的运动平滑

针对传统方法在路径平滑时往往依赖计算复杂的最小二乘优化求解的问题，本文提出一种用相对简单的 2D 几何变换模型描述相机运动，但借助内在运动几何量的线性插值实现运动平滑的方法。该方法既降低计算消耗(线性插值的计算复杂度是线性的，这是以往算法所远远不及的)，满足 ARM 处理器简单浮点型快速计算的需求，又能较好地保持稳像效果。

2.1 相机运动模型

Ryuichi Ogino 等人[13]证明相机在三维空间中的平移和旋转运动在平面上的投影可以近似表述为二维平移和旋转的组合。因此，两帧图像上的对应点 $\boldsymbol{p}=(x,y)^{\mathrm{T}}$ 和 $\boldsymbol{p}'=(x',y')^{\mathrm{T}}$ 之间的运动，可以用如下的刚性变换来表示：

$$\boldsymbol{p}'=\boldsymbol{R}\cdot\boldsymbol{p}+\boldsymbol{d} \tag{1}$$

其中：$\boldsymbol{R}=[\cos\theta,-\sin\theta;\sin\theta,\cos\theta]$ 表示旋转角度为 θ 的旋转矩阵；$\boldsymbol{d}=[d_x,d_y]^{\mathrm{T}}$ 表示平移向量。

本文假设相机在空间中的运动路径为 $\boldsymbol{P}_t$，其中 $t\in[t_0,t_1]$ 是相机运动时间间隔，那么，相机运动路径就可以用时间序列二元组表示：$\boldsymbol{P}_t=<\boldsymbol{R}_t,\boldsymbol{d}_t>$。

2.2 平滑路径计算

令输入的视频帧序列为 $\boldsymbol{I}_1,\boldsymbol{I}_2,\cdots,\boldsymbol{I}_n$，对应的相机原始运动路径为 $\{\boldsymbol{P}_1,\boldsymbol{P}_2,\cdots,\boldsymbol{P}_n\}$，其中，

$\boldsymbol{P}_t = \boldsymbol{F}_1 \cdot \boldsymbol{F}_2 \cdots \boldsymbol{F}_t$， $\boldsymbol{F}_t$ 表示 $\boldsymbol{I}_{t-1} \to \boldsymbol{I}_t$ 的帧间刚性变换，且 $\boldsymbol{F}_1$ 为单位矩阵。

我们通过 $\boldsymbol{P}_t$ 对应的平移和旋转运动进行线性插值平滑，生成稳定的路径 $\tilde{\boldsymbol{P}}_t$。其中，首帧 $\boldsymbol{I}_1$ 和尾帧 $\boldsymbol{I}_n$ 是线性插值的两个端点，不需进行运动补偿，即 $\tilde{\boldsymbol{P}}_1 = \boldsymbol{P}_1$， $\tilde{\boldsymbol{P}}_n = \boldsymbol{P}_n$，而且它们之间有如下的运动转换关系：

$$\boldsymbol{P}_n(\boldsymbol{p}) = \begin{bmatrix} \cos\theta_n & -\sin\theta_n \\ \sin\theta_n & \cos\theta_n \end{bmatrix} \cdot \boldsymbol{p} + \boldsymbol{d}_n \tag{2}$$

其中：θ_n 和 $\boldsymbol{d}_n$ 分别是帧 $\boldsymbol{I}_1$ 到帧 $\boldsymbol{I}_n$ 的旋转角度和平移向量。由此，只需分别对平移和旋转进行线性插值就可以得到 $\tilde{\boldsymbol{P}}_t =< \tilde{\boldsymbol{R}}_t, \tilde{\boldsymbol{d}}_t >$，即：

$$\tilde{\boldsymbol{R}}_t = \begin{bmatrix} \cos\theta_t & -\sin\theta_t \\ \sin\theta_t & \cos\theta_t \end{bmatrix},\ \tilde{\boldsymbol{d}}_t = \frac{t-1}{n-1}\boldsymbol{d}_n \tag{3}$$

其中：$\theta_t = \dfrac{t-1}{n-1}\theta_n$ 是旋转角度插值。

将 $\tilde{\boldsymbol{P}}_t$ 表示为矩阵形式，便可得到平滑路径的矩阵形式，即：

$$\tilde{\boldsymbol{P}}_t = \begin{bmatrix} \tilde{\boldsymbol{R}}_t & \tilde{\boldsymbol{d}}_t \\ 0 & 1 \end{bmatrix} \tag{4}$$

那么，任意帧 $\boldsymbol{I}_t$ 对应的运动补偿矩阵为 $\boldsymbol{B}_t = \boldsymbol{P}_t^{-1}\tilde{\boldsymbol{P}}_t$。据此对帧图像进行相应的几何变换即可得到稳定的帧。整个过程不需要任何函数最小二乘优化求解，适合于 ARM 平台的移植。

3 基于ARM平台的视频流实时稳像算法

所谓基于 ARM 处理器的开发平台，即以英国 ARM 公司的内核芯片作为 CPU，同时附加其他外围功能的嵌入式开发板。其具有能耗低、功能强和价格低廉的技术特点，适用于嵌入控制、移动式应用等多种领域。因此，需要对上述视频稳像算法进行专门设计，以符合 ARM 处理器性能特点。

3.1 算法框架

为了能够实时地处理输入的视频流，我们在 ARM 线程设计的基础上采用多线程流水线技术实现视频流稳像。具体来说，整个稳像算法分割成四个线程，以流水线方式实现线程间的同步(如图 1 所示)。这四个线程分别是：1) 视频帧读取线程；2) 特征轨迹构造线程；3) 运动估计和平滑线程；4) 稳定帧显示和保存线程。

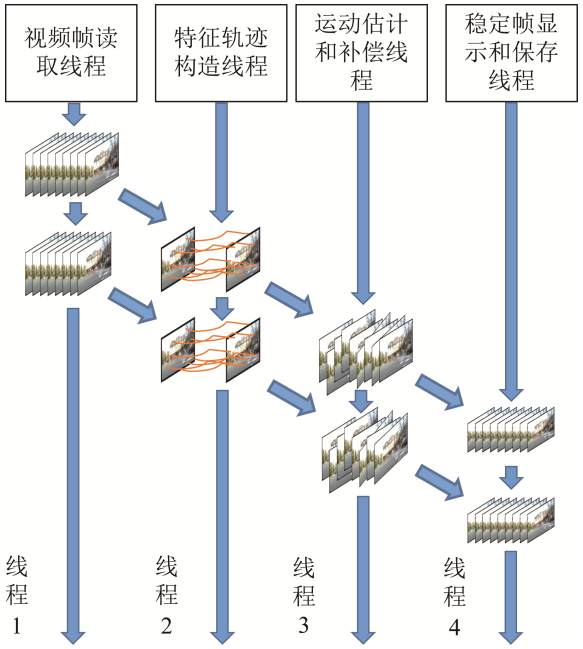


图 1　多线程并行的算法框架

视频帧读取线程负责采集图像，并且在积蓄的数量达到指定帧数(本文设为 30)之后唤醒特征轨迹构造线程；特征轨迹构造线程在采集到的图像序列中执行特征点检测和跟踪算法，完成后唤醒运动估计和补偿线程；运动估计和平滑线程执行抖动检测，如果发生视频抖动，则通过线性插值计算运动平滑和补偿，同时进行裁剪控制，生成稳定帧；稳定帧显示和保存线程来同步展示和存储稳像结果。下面，我们重点介绍运动估计和平滑线程，主要包括运动估计、抖动检测、裁剪可控的视频稳定三个步骤。

3.2 运动估计

为满足系统的实时性要求，我们在特征轨迹构造线程使用 KLT 算法[1]进行特征点的检测和跟踪。在获得特征点匹配对 $\boldsymbol{p}_l \in \mathbf{P} \mapsto \boldsymbol{p}'_l \in \mathbf{P}'$ $(l = 1, \cdots, L)$ 之后，公式(1)中的平移变换便可由 $\boldsymbol{d} = \boldsymbol{p}'_o - \boldsymbol{p}_o$ 计算得到，其中 $\boldsymbol{p}'_o$ 和 $\boldsymbol{p}_o$ 分别是两个特征点集的几何中心。然后，对特征点集合中心

化并构建相应矩阵，即 $\bar{\mathbf{P}}=[\boldsymbol{p}_l-\boldsymbol{p}_o]_{2\times L}$ 和 $\bar{\mathbf{P}}'=[\boldsymbol{p}'_l-\boldsymbol{p}'_o]_{2\times L}$，据此，可计算出最优旋转变换矩阵 $\boldsymbol{R}=\boldsymbol{V}\boldsymbol{U}^{\mathrm{T}}$，$\boldsymbol{V}$ 和 $\boldsymbol{U}$ 通过 SVD 分解得到，即 $\bar{\mathbf{P}}\bar{\mathbf{P}}'^{\mathrm{T}}=\boldsymbol{U}\mathrm{diag}(\sigma_1,\sigma_2)\boldsymbol{V}^{\mathrm{T}}$。根据平移变换和旋转变换，我们可以估计基于几何变换的帧间运动 $\boldsymbol{P}_t$。

3.3 抖动检测

便携设备拍摄的视频中除了抖动的部分之外通常还存在着一些稳定的帧，而对稳定帧进行去抖处理是不必要且浪费计算资源的。因此，我们在此提出一种简单而有效的抖动检测算法来衡量每一个视频分段的抖动程度，并据此决定是否对其进行去抖处理，从而能够进一步地降低 ARM 处理器的计算负担。

显而易见的是，一次抖动的产生至少需要 3 帧图像，换而言之，连续的 3 帧图像是构成视频抖动的基本单位。基于这一点观察，我们可以将本来难以量化的抖动现象进行分解，从而能够在更小的尺度上对抖动程度进行度量。具体来说，对于帧序列中除首尾之外的任意帧 $\boldsymbol{I}_k$，我们使用与其相邻的前后两帧来计算一次平移振幅和旋转振幅，即：

$$\left|\ddot{\boldsymbol{d}}_k\right|=\delta\left(\frac{(\boldsymbol{d}_{k+1}-\boldsymbol{d}_k)(\boldsymbol{d}_k-\boldsymbol{d}_{k-1})}{\left\|\boldsymbol{d}_{k+1}-\boldsymbol{d}_k\right\|_2\left\|\boldsymbol{d}_k-\boldsymbol{d}_{k-1}\right\|_2}-1\right)\left\|\ddot{\boldsymbol{d}}_k\right\|_2 \tag{5}$$

$$\left|\dot{\theta}_k\right|=\delta\left(\theta_{k-1}\theta_k\right)\left\|\dot{\theta}_k\right\|_2 \tag{6}$$

其中：$\ddot{\boldsymbol{d}}_k=(\boldsymbol{d}_{k+1}+\boldsymbol{d}_{k-1}-2\boldsymbol{d}_k)/2$，$\boldsymbol{d}_{k-1},\boldsymbol{d}_k,\boldsymbol{d}_{k+1}$ 各自表示首帧到 $\boldsymbol{I}_{k-1},\boldsymbol{I}_k,\boldsymbol{I}_{k+1}$ 的平移向量，且 $\dot{\theta}_k=\theta_k-\theta_{k-1}$，$\theta_{k-1},\theta_k$ 各自表示首帧到 $\boldsymbol{I}_{k-1},\boldsymbol{I}_k$ 的旋转角度，$\delta(\cdot)$ 满足 $\delta(x<0)=1$ 且 $\delta(x\geqslant 0)=0$。然后，每一个视频分段的抖动程度就可以用振幅的平均值来描述，即

$$\bar{d}=\frac{1}{n}\sum_k\left|\ddot{\boldsymbol{d}}_k\right|,\ \bar{\theta}=\frac{1}{n}\sum_k\left|\dot{\theta}_k\right| \tag{7}$$

其中：n 表示帧数。

本文中通过设置阈值 α 和 β 来判断是否发生了抖动。对于每个视频分段，我们比较其平均平移振幅 $\bar{d}$ 与 α 以及平均旋转振幅 $\bar{\theta}$ 与 β 之间的大小关系，若 $\bar{d}>\alpha$ 或者 $\bar{\theta}>\beta$，则判定发生抖动，下一步对之进行运动补偿。本文中关于 α 和 β 的值，我们分别设为 0.4 和 0.02。

图 2 展示了一个视频的平移振幅和旋转振幅随时间的变化。其中，明显可以看出在视频的前 180 帧发生抖动时平移振幅变化剧烈，而在视频稳定的时候平移振幅和旋转振幅的变化都趋向平缓。这充分说明了使用平移振幅和旋转振幅来检测视频抖动的有效性。

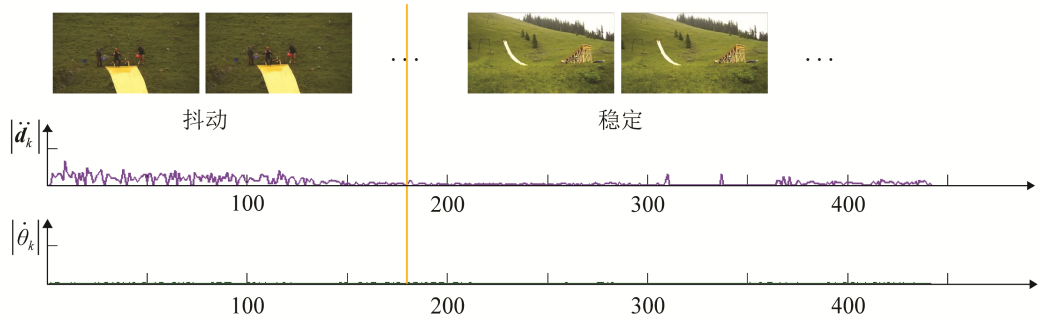


图 2 平移振幅和旋转振幅变化图

3.4 裁剪可控的视频稳定

根据 3.1 节算法框架可知，算法以指定长度为 m 的视频分段作为稳像处理的基本单元。同时，为确保段与段之间的稳定性，除最后一段外，其余段的最后一帧同时也是它后续相邻段的起始帧。每一段的平滑路径通过计算简单的线性插值获得，而不必如方法[1,10]那样进行复杂的最小二乘能量函数求解，所以能够满足在 ARM 处理器上的实时需求。

设某个视频分段的帧序列为 $\boldsymbol{I}_1,\boldsymbol{I}_2,\cdots,\boldsymbol{I}_m$，且

对于任意帧 $\boldsymbol{I}_k$， $\boldsymbol{I}_k \to \boldsymbol{I}_1$ 的平移向量 $\boldsymbol{d}_k$ 和旋转角度 θ_k 都已在运动估计中计算得到。另外，运动平滑后 $\boldsymbol{I}_k$ 相对于 $\boldsymbol{I}_1$ 的平移向量 $\tilde{\boldsymbol{d}}_k$ 和旋转角度 $\tilde{\theta}_k$ 也能通过线性插值获得，即：

$$\tilde{\boldsymbol{d}}_k = \frac{k-1}{m-1}\boldsymbol{d}_m,\ \tilde{\theta}_k = \frac{k-1}{m-1}\theta_m \tag{8}$$

所以，帧 $\boldsymbol{I}_k$ 的平移补偿和旋转补偿分别为：$\hat{\boldsymbol{d}}_k = \boldsymbol{d}_k + \tilde{\boldsymbol{d}}_k$ 以及 $\hat{\theta}_k = \theta_k + \tilde{\theta}_k$。也就是说，让帧 $\boldsymbol{I}_k$ 先绕其特征点几何中心旋转 $\hat{\theta}_k$，再平移 $\hat{\boldsymbol{d}}_k$，即可生成稳定帧。

相应变换可表示为如下矩阵形式：

$$\boldsymbol{B}_k = \begin{bmatrix} \hat{\boldsymbol{R}}_k & \hat{\boldsymbol{d}}_k \\ 0 & 1 \end{bmatrix},\ \hat{\boldsymbol{R}}_k = \begin{bmatrix} \cos\hat{\theta}_k & -\sin\hat{\theta}_k \\ \sin\hat{\theta}_k & \cos\hat{\theta}_k \end{bmatrix} \tag{9}$$

另外，原始帧经过运动补偿后，必然会有部分信息丢失，导致裁剪后的视频分辨率变小。然而在视频观看体验中，视频分辨率又是影响人眼感受的一个很重要的因素。对于用户来说，也更倾向于能够自主控制稳定视频的分辨率大小。这是在以往的视频稳像算法中鲜有考虑的步骤。

幸运的是，在本文提出的算法中很容易利用 ARM 处理器的计算能力对裁剪进行控制(如图 3 所示)。假设用户指定裁剪窗口(黄框)的高度和宽度与原始视频(黑框)高度和宽度的比例为 r，然后我们判断经过补偿变换后的视频帧(红框)是否依然包括裁剪窗口，若不包括，则在保持旋转补偿变换方向与平移补偿变换方向不变的前提下，计算出最大的旋转量和平移量，并以此对图像进行运动补偿(绿框)。通过这种操作，即可得到可控分辨率的稳定视频。

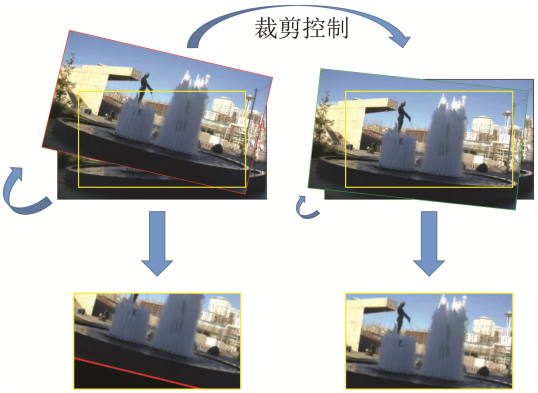


图 3　视频帧变换过程中的裁剪控制

4　实验

在这一章中，我们采用 ODROID 公司出产的 XU4 开发板测试本文的视频稳像算法的性能。此外，我们还提供了抖动检测和视频稳像算法的实验结果，同时也对本文算法与当前公认为顶级的几种视频稳像算法进行了比较。此部分实验在一台配置为 Intel 四核处理器，主频 3.1 GHz，内存 8 G 的计算机上完成。实验中使用的视频数据将会在未来陆续公开。下面分别介绍相应的实验内容。

4.1 ARM 算法测试

我们搭建了如图 4 所示的 ARM 系统平台，三个主要部件及其参数分别为：

1 为 ARM 开发板：2.0 GHz 的 Samsung 八核处理器，2 G 内存；

2 为 RMONCAM 摄像头：分辨率 720×480，帧率 30 fps；

3 为 Dell 显示屏：分辨率 800×480，触摸屏。

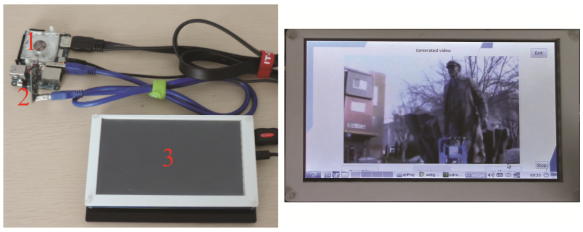


图 4　ARM 系统平台及界面

在实验的这一部分，我们测试了稳像算法的如下几个性能指标：

(1) CPU、内存的资源占用率；

(2) 系统响应时间，即从开始拍摄到开始显示稳像结果所用时间；

(3) 在包括线程等待时间的前提下，特征轨迹构造线程以及运动估计和补偿线程各自处理一次视频分段的平均消耗时间和最大消耗时间。

在测试时，我们将整个系统平台用盒子封装，然后携带至不同场景，一边拍摄一边同步稳像，先后 5 次，每次持续 10 min 左右。本文的视频补充材料展示了一些稳像效果。

在实验中，稳像算法的最大 CPU 利用率为 83%，平均 CPU 利用率为 49%，最大内存消耗为 548 MB，平均内存消耗为 526 MB。系统的平均响应时间为 2.49 s，最大响应时间为 2.6 s。

在包括线程等待时间的前提下，特征轨迹构造线程以及运动估计和补偿线程各自处理一次视频分段的平均消耗时间 t_1 和最大消耗时间 t_2 如表 1 所示，时间单位为毫秒。

由于摄像头帧率为 30 fps，且此次实验中我们令 m 为 30，所以由多线程流水线处理的特点可知，稳像系统达到实时的前提条件是 t_1 和 t_2 都必须小于 1 s，即意味着每次在显示和保存稳像结果之前相应帧的稳像处理便已完成，无需进行等待。而从表 1 可以看出，两个线程的平均耗时和最大耗时都小于 1 s，实验结果与实际主观观测结果相符。

表 1 线程消耗时间/ms

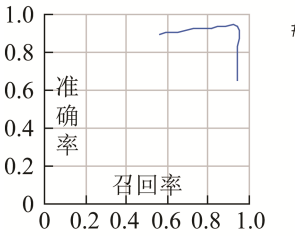
线程	t_1	T_2
特征轨迹构造	745	899
运动估计和补偿	764	828

4.2 抖动检测效果

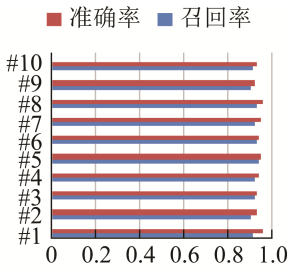
为了评估本文提出的视频抖动检测效果，我们收集了 10 个用便携设备拍摄的长视频，每个视频的长度均不少于 5 min，分段之后共得到 3 460 个视频分段。随后，对每一段执行一次 3.3 节提出的抖动检测算法，并记录判断结果。

为验证抖动检测算法的有效性，我们邀请了 60 个人来观看这些视频分段，并让他们根据自身感受判断其是否发生了抖动。在记录下所有人的判断结果后，我们针对每一段分别统计了认为“稳定”和认为“抖动”的人数，取所占人数更多的一方作为这一段的真实结果。由此，我们便能计算出抖动检测算法对应于每一个视频的准确率和召回率。

另外，我们还通过改变 α 和 β 得到了如图 5(a)所示的准确率-召回率曲线，其中 $\alpha \in [0.3, 0.7]$，改变间隔为 0.05，$\beta \in [0, 0.1]$，改变间隔为 0.02。我们发现当 α 的值为 0.4，β 为 0.02 时，该算法能够较好地识别出视频抖动，如图 5(b)所示，召回率在 90%以上，准确率也不低于 92%。



(a) 在不同阈值下的准确率-召回率曲线。



(b) 在本文默认阈值下 10 个视频的准确率和召回率

图 5 抖动检测实验结果

4.3 视频稳像结果对比

本文提出的视频稳像算法能够以极小的计算消耗获得可与其他方法相比甚至更好的结果。在这里，我们选择了两种近年来著名的视频稳像方法和两款商业应用软件来进行比较。两种方法分别是 Wang 等人[10]提出的 STO 方法与 Liu 等人[11]提出的 BCP 方法，两款软件分别是 Adobe After Affect (AE)和 YouTube 网页稳像器。图 5 展示了部分实验结果对比。

对比实验测试视频主要来自于文献[11]提供的公共数据集，数据集中提供了 174 个短视频，包括 simple、quick rotation、zooming、large parallax、driving、crowd、running 共 7 个类别。另外，我们还添加了 12 个类别为 riding 的新视频，这一类视频是在骑车时通过固定在头部的摄像头拍摄得到的。

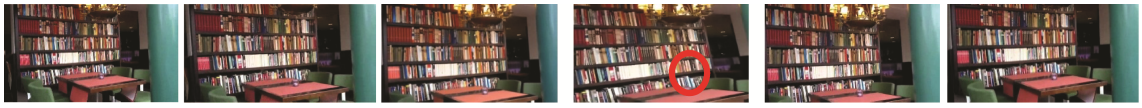
在衡量一种视频稳像方法时，通常可以从稳像效果和时间消耗两方面入手。关于稳像效果，本文通过扭曲和裁剪分析几种方法的优劣，同时也让用户从主观上比较稳像结果的好坏。至于时间消耗，则通过统计几种方法的运行时间来进行比较。

在扭曲方面，因为本文提出的方法采用全局刚性变换作为相机运动的数学模型，所以最后经过补偿变换生成的稳定图像中不会出现扭曲现象。相反，其他 4 种方法却都是通过将图像划分成网格，然后使用一系列局部变换来生成稳定图像，因此便可能出现局部扭曲，如图 6(b)~(d)。

在裁剪方面，本文提出的算法允许用户自主设定裁剪的比例。此外，BCP 和 YouTube 中也将裁剪比例作为路径优化时的一个约束条件，而 STO 和 AE 则没有进行裁剪控制，所以常常会导致生成的稳定视频裁剪过度，如图 6(a)~(b)。

(a) zooming 类型视频结果

(b) riding 类型视频结果



(c) large parallax 类型视频结果

(d) driving 类型视频结果

注：第 1 列为原始输入帧，后续依次为 STO，BCP，AE，YouTube 以及本文算法的结果。红色椭圆表示图像产生扭曲，红框表示稳定之后图像被过度裁剪。

图 6　对比实验结果

进一步，我们还通过用户调查进行主观对比。我们在每种视频类别里随机选取 3 个视频作为样本，然后邀请 40 名年龄在 20 岁与 60 岁之间，职业背景不同的参与者来对各个方法进行打分。我们采用二选一机制，即每次只要求参与者在本文算法结果与其他方法结果之间选择效果更优者，并且视频顺序是打乱的，参与者不清楚他选择的视频由哪种方法生成。主观比较的统计结果如图 7 所示，其中的 8 帧图像分别取自 8 种类别的视频，可以看出本文算法并不输于其他四种方法。

在时间消耗方面，我们统计了不同方法在同一台计算机上的运行时间，见表 2。

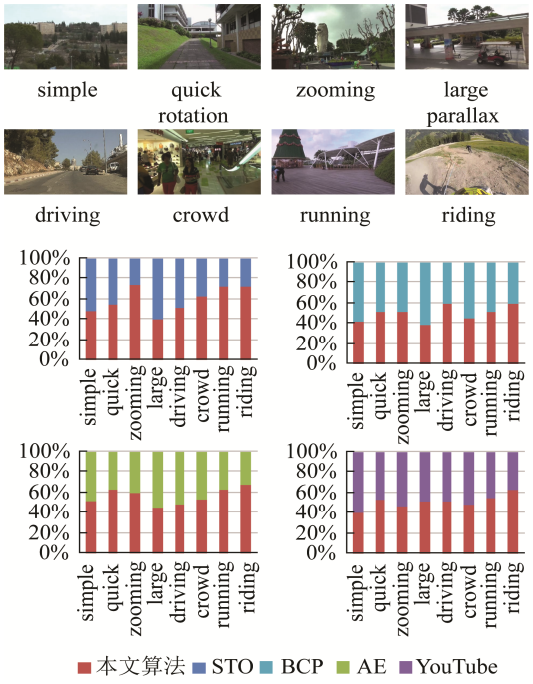


图 7　不同视频稳像结果的用户调查对比

表 2 不同方法的帧平均消耗时间/ms

视频稳像方法	分辨率	运行时间
STO	1280×720	48
BCP	1280×720	400
AE	640×360	250
YouTube	720×480	50
本文算法	1280×720	27

从表 2 可以看出本文算法拥有更为优越的性能，其中时间单位是 ms。对于分辨率为 1280×720 的视频，本文算法能够达到 37 帧/秒的处理速度，而 STO 和 BCP 各自只能达到 20.8 与 2.5 的帧率。此外，AE 和 YouTube 的算法效率也不高，前者对分辨率为 640×360 的视频的处理速度仅为 4 帧/秒，后者对分辨率为 720×480 的视频的处理速度也只有 20 帧/s。

5 结论

本文提出了一种在几何变换空间内进行视频抖动检测和运动平滑的高效视频稳像增强算法，能够在基于 ARM 处理器的平台上实现视频流的实时稳像处理。该算法通过平移和旋转分量的振幅变化检测视频流中出现的抖动。对于发生抖动的视频序列，则通过对平移向量和旋转角度的线性插值来获得平滑路径的最优几何变换，极大地降低了算法的时间复杂度。最后根据最优几何变换进行运动补偿，即可生成稳定的视频帧。

不足的是，由于该算法采用特征点匹配的方式进行运动估计，所以对于特征点稀少的视频便可能无法得到较好的稳像结果。同时，刚性变换模型无法描述非刚性对象的运动，不能很好地处理包含非刚性对象的视频。此外，二维刚性变换本质上也并不能非常准确地描述相机在三维空间中的运动，因此对于运动复杂的视频便难以获得理想的结果。一种具有可行性的算法优化方案是采用多条局部路径共同描述相机运动的方法，以此增加运动估计精度。

参考文献：

[1] Grundmann M, Kwatra V, Essa I. Auto-Directed Video Stabilization with Robust L1 Optimal Camera Paths [C]// IEEE Conference on Computer Vision & Pattern Recognition. USA: IEEE, 2011: 225-232.

[2] Liu F, Gleicher M, Wang J, et al. Subspace Video Stabilization [J]. ACM Transactions on Graphics (S0730-0301), 2011, 30(1): 623-636.

[3] 姜文涛, 陈卫东, 李良福. 一种基于特征点跟踪的电子稳像算法 [J]. 应用光学, 2010, 31(1): 73-77.

[4] 张国栋, 王明泉, 郭栋. 基于灰度投影算法的实时电子稳像研究 [J]. 微电子学与计算机, 2010, 27(10): 53-56.

[5] 张坤, 许廷发, 王平, 等. 高精度实时全帧频 SURF 电子稳像方法 [J]. 光学精密工程, 2011, 19(8): 1964-1972.

[6] 邢慧, 颜景龙, 张树江. 基于 Kalman 滤波的稳像技术研究 [J]. 兵工学报, 2007, 28(2): 175-177.

[7] Liu F, Gleicher M, Jin H, et al. A. Content-Preserving Warps for 3D Video Stabilization [J]. Acm Transactions on Graphics (S0730-0301), 2009, 28(3): 341-352.

[8] Zhang G, Hua W, Qin X, et al. Video Stabilization Based on a 3D Perspective Camera Model [J]. Visual Computer (S0178-2789), 2009, 25(11): 997-1008.

[9] Zhou Z, Jin H, Ma Y. Plane-Based Content Preserving Warps for Video Stabilization[C]// IEEE Conference on Computer Vision & Pattern Recognition. USA: IEEE 2013:2299-2306.

[10] Wang Y S, Liu F, Hsu P S, et al. Spatially and Temporally Optimized Video Stabilization [J]. IEEE Transactions on Visualization & Computer Graphics (S1077-2626), 2013, 19(8): 1354-61.

[11] Liu S, Yuan L, Tan P, et al. Bundled Camera Paths for Video Stabilization [J]. ACM Transactions on Graphics (S0730-0301), 2013, 32(4): 96-96.

[12] Goldstein A, Fattal R. Video Stabilization Using Epipolar Geometry [J]. ACM Transactions on Graphics (S0730-0301), 2012, 32(5): 573-587.

[13] Ogino R, Suzuki T, Nishi K. Camera-Shake Detection and Evaluation of Image Stabilizers [C]// SICE Annual Conference, 2008. USA: IEEE, 2008: 528-531.

