Postprint: Real-Time Image Stabilization Technology Using Weighted Least Squares and Kalman Filtering

Authors: Gu Le, Zhiyun Chen

Date: 2018-09-12T00:00:00+00:00

Abstract

Videos captured during drone flight and vehicle movement are subject to external influences that cause video shake. By comparing existing electronic image stabilization techniques, this paper proposes an improved algorithm that employs FAST to acquire feature point position information, then utilizes optical flow combined with NCC matching to obtain the position information of reference frame feature points in the current frame. Building upon this foundation, the RANSAC algorithm is integrated to eliminate erroneously matched feature point pairs. To enhance the precision of motion vector estimation, weighted least squares is applied to derive the rigid transformation matrix between adjacent frames, followed by Kalman filtering for motion smoothing to obtain the scanning motion vector and compensation, ultimately yielding real-time stabilized video. Experimental results indicate that the inter-frame transformation fidelity of the stabilized video sequence is improved, and real-time processing speeds are achieved.

Full Text

Improved Weighted Least Squares and Kalman Filtering for Real-Time Image Stabilization

Gu Le, Chen Zhiyun†
(Computing Center, East China Normal University, Shanghai 200062, China)

Abstract

Videos captured from drones and moving vehicles often suffer from jitter due to external disturbances. By comparing existing electronic image stabilization techniques, this paper proposes an improved algorithm that employs FAST to

obtain feature point positions, then uses optical flow combined with NCC matching to determine the positions of reference frame feature points in the current frame. Building upon this foundation, the RANSAC algorithm is incorporated to eliminate incorrectly matched feature point pairs. To improve the accuracy of motion vector estimation, a weighted least squares method is applied to obtain the rigid transformation matrix between adjacent frames. Kalman filtering is then used for motion smoothing to derive the scanning motion vector, which is compensated to produce a stable video in real time. Experimental results demonstrate that the proposed method improves inter-frame transformation fidelity in stabilized video sequences while achieving real-time processing speeds.

Keywords: electronic image stabilization; feature point matching; least squares method; Kalman filtering; motion compensation

0 Introduction

Videos captured by vehicle-mounted cameras or small drones often exhibit jitter because the platform cannot remain perfectly stable. Eliminating video jitter to improve video quality and viewing experience is the primary focus of video stabilization research. Current stabilization techniques include mechanical, optical, electromechanical, and electronic methods [1]. Compared with other approaches, electronic image stabilization offers advantages of low cost, modest hardware requirements, and excellent stabilization performance, making it a widely studied technique.

Electronic image stabilization first estimates motion compensation parameters through motion vector estimation, then smooths these parameters, and finally compensates the original jittery sequence using the smoothed parameters to obtain stable video. Among these steps, the accuracy of motion vector calculation has the greatest impact on overall performance. Therefore, precisely acquiring motion vector variation information represents a core challenge in electronic image stabilization methods [2]. Motion vector estimation algorithms include gray projection methods [3], bit-plane matching [4], representative point matching [5], and feature point matching [6].

This paper compares the strengths and weaknesses of different algorithms. The SIFT (Scale Invariant Feature Transform) method is widely recognized as one of the best corner detectors, producing highly distinctive feature descriptors. Subsequent algorithms have optimized computational time while preserving SIFT's effectiveness, with SURF (Speeded Up Robust Features) considered the most successful improvement [2]. However, SURF has high computational complexity, making real-time applications difficult. FAST (Features from Accelerated Segment Test), proposed by Rosten and Drummond in 2006, is a simple and fast corner detection algorithm that is many times faster than both SIFT and SURF [7]. This paper adopts FAST feature detection for inter-frame registration to obtain local motion vectors between frames, then uses optical flow combined

with NCC (Normalized Cross Correlation) matching to acquire the position information of reference frame feature points in the current frame. The RANSAC (Random Sample Consensus) algorithm is employed to eliminate "bad" feature point pairs, followed by an improved least squares method to obtain the rigid transformation matrix between adjacent frames. Finally, Kalman filtering and bilinear interpolation are applied for compensation to achieve real-time stable video.

shows a comparison of commonly used feature point detection algorithms. As evident from the table, SIFT and SURF feature extraction algorithms have much greater computational complexity and longer processing times than FAST. From an algorithmic efficiency perspective, FAST offers clear advantages. [Figure 3: see original paper] illustrates the feature points extracted by the FAST operator.

---

1 Feature Point Detection and Representation Algorithms

Image matching is critical in feature-based electronic image stabilization. This paper presents a real-time digital video stabilization scheme, shown in [Figure 1: see original paper].

1.1 FAST Feature Point Detection

FAST (Features from Accelerated Segment Test) works by selecting a detection point from the image and using 16 surrounding pixels to determine if the point is a corner. As shown in [Figure 2: see original paper], the algorithm first checks whether the gray values of pixels 1, 5, 9, and 13 satisfy the condition; if not, the point is eliminated. On average, this algorithm requires checking only 3.8 points around a candidate to determine whether it is a corner. compares the three detection algorithms: FAST, SIFT, and SURF.

1.2 Feature Point Matching Based on Pyramid Lucas-Kanade and NCC

After obtaining feature point positions through detection, the positions of feature points in the current frame must be calculated. This paper employs a pyramid Lucas-Kanade optical flow algorithm combined with NCC matching.

The Lucas-Kanade (L-K) algorithm [8] operates under three assumptions: (a) brightness constancy, (b) slow temporal motion variation or temporal continuity between adjacent frames, and (c) spatial coherence. The L-K algorithm is efficient with good real-time performance, effectively tracking salient feature points. However, it does not adapt well to illumination changes and is prone to errors when tracking less distinctive feature points.

The Normalized Cross Correlation (NCC) algorithm [9] can compensate for L-K's limitations. NCC effectively reduces the impact of illumination changes on

image comparison. In industrial inspection, surveillance, and similar applications requiring image similarity assessment, NCC is commonly used.

This paper combines L-K and NCC algorithms for feature tracking. Selected feature points are categorized by their response values: points above a threshold are tracked using NCC, while points below the threshold use L-K. NCC computational cost can be optimized using integral images for normalization and frequency-domain products for correlation (convolution) calculations [10]. To further reduce computational cost, a pyramid strategy is also employed. [Figure 4: see original paper] shows the feature point tracking results for frame 306.

1.3 RANSAC Algorithm

To obtain more accurate global motion vectors and prevent feature points on moving objects from affecting global relative motion vector estimation, this paper adopts the RANSAC algorithm [11] to eliminate interference from moving targets.

The classic RANSAC algorithm, first proposed by Fischler and Bolles [12] in 1981, iteratively tries different target spatial parameters to maximize an objective function. The process involves randomly selecting subsets of the dataset to generate model estimates, testing these estimates against remaining points, and selecting the model with the highest score as the final dataset model. Determining whether a point is an inlier requires setting a threshold.

Each subset should be a minimal sample of size m during iteration. The minimal sample size m must be sufficient to compute model parameters. For confidence level ε, at least one sampling iteration must select m inliers to ensure the objective function reaches its maximum. Therefore, the number of sampling iterations k should satisfy:

$$k \geq \frac{\log(1-\varepsilon)}{\log(1-p^m)}$$

where p represents the proportion of inliers, and confidence is typically set to $[0.95, 0.99]$. The loop termination condition follows a binomial distribution for two outcomes: "all inliers" or "not all inliers." The probability of "all inliers" is p^m. If p^m is sufficiently small, it can be approximated as a Poisson distribution. Thus, the probability of n "all inlier" selections is:

$$P(n) = \frac{\lambda^n e^{-\lambda}}{n!}$$

where λ represents the expected number of "all inlier" selections in one frame.

2 Motion Vector Estimation

2.1 Traditional Motion Vector Estimation

After RANSAC filtering, a 4-parameter affine transformation model [12] is used to calculate motion vectors:

$$\begin{pmatrix} X_n \\ Y_n \end{pmatrix} = \begin{pmatrix} a & -b \\ b & a \end{pmatrix} \begin{pmatrix} X_{n-1} \\ Y_{n-1} \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix}$$

where a represents the image scaling component, b represents the rotation component, and t_x, t_y represent translation components between images. (X_{n-1}, Y_{n-1}) denotes reference frame feature point coordinates, and (X_n, Y_n) denotes current frame feature point coordinates. The rigid transformation matrix between adjacent frames can be obtained through least squares method.

Rigid transformation can be expressed as:

$$\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_n \\ y_n \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$$

2.2 Improved Motion Vector Estimation

Traditional least squares methods can only roughly calculate motion vectors because feature points at different depths significantly affect motion vector accuracy [13]. Therefore, this paper improves the traditional least squares method by using particle filtering to obtain feature point weights, then applying weighted least squares for more accurate motion vector estimation.

The basic idea of particle filtering [14] is to approximate the system's posterior probability density using weighted particles and estimate the system state, where X_t and Z_t represent the target state and observation at time t, respectively.

First, a set of feature point pairs is randomly selected to form particles, then motion vectors are obtained using equation (3). Note that more than three feature point pairs are required. A feature point may belong to multiple particles. After obtaining N particles, their weights are determined by pointwise mean square criterion, as shown in equation (5):

$$\omega_{i,t} = \frac{\exp\left(-\frac{(I_t - I_{i,t})^2}{2\sigma^2}\right)}{\sum_{i=1}^{N} \exp\left(-\frac{(I_t - I_{i,t})^2}{2\sigma^2}\right)}$$

where $\omega_{i,t}$ represents the weight of the i-th particle at time t; I_t is the image at time t; $I_{i,t}$ is the compensated image calculated based on the i-th particle using equation (5). Since compensated images contain undefined regions, the center

region of the reference frame is used during calculation, where w represents the width and h the height of the reference frame.

Feature point weights can be obtained from particle weights:

$$\omega_{j,t} = \frac{\sum_{k=1}^{N_{j,t}} W_{k,t}}{N_{j,t}}$$

where $\omega_{j,t}$ is the weight of the j-th feature point at time t; $N_{j,t}$ is the number of particles corresponding to the j-th feature point; $W_{k,t}$ is the weight of the corresponding particle. Weighted least squares can then accurately estimate the current frame's motion vector.

When selecting the number of feature points, consideration must be given to the fact that different particles have varying numbers of feature points, which affects global motion vector accuracy.

---

3 Kalman Motion Compensation and Video Output

Kalman filtering extracts intentional motion vectors from the motion vectors. The entire process predicts the current motion vector based on the previous estimate and covariance. The Kalman filter prediction equations are:

$$\begin{cases} \hat{s}(k|k-1) = F\hat{s}(k-1|k-1) \\ P(k|k-1) = FP(k-1|k-1)F^T + Q \end{cases}$$

The Kalman filter update equations are:

$$\begin{cases} \hat{s}_g(k|k) = \hat{s}(k|k-1) + K_g(k)[z(k) - H\hat{s}(k|k-1)] \\ K_g(k) = P(k|k-1)H^T[HP(k|k-1)H^T + R]^{-1} \\ P_g(k|k) = [I - K_g(k)H]P(k|k-1) \end{cases}$$

In equation (8), s is the system state vector, F represents the state transition matrix, F^T is its transpose, and Q is the measurement noise variance. In equation (9), $\hat{s}_g(k|k)$ is the Kalman filter result, $K_g(k)$ is the Kalman gain, H represents the observation matrix, and $P_g(k|k)$ is the filtered covariance. R is typically taken as 1 mm.

Note: Figure translations are in progress. See original paper for figures.