The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Robust Video Stabilization to Outlier Motion using Adaptive RANSAC

Sunglok Choi, Taemin Kim, and Wonpil Yu

Abstract— **The core step of video stabilization is to estimate global motion from locally extracted motion clues. Outlier motion clues are generated from moving objects in image sequence, which cause incorrect global motion estimates. Random Sample Consensus (RANSAC) is popularly used to solve such outlier problem. RANSAC needs to tune parameters with respect to the given motion clues, so it sometimes fail when outlier clues are increased than before. Adaptive RANSAC is proposed to solve this problem, which is based on Maximum Likelihood Sample Consensus (MLESAC). It estimates the ratio of outliers through expectation maximization (EM), which entails the necessary number of iteration for each frame. The adaptation sustains high accuracy in varying ratio of outliers and faster than RANSAC when fewer iteration is enough. Performance of adaptive RANSAC is verified in experiments using four images sequences.**

I. Introduction

Video stabilization is the process to generate a compensated video without undesired motion. It gains more attention due to popularization of video devices. It is applied to enhance quality of video captured by shaking hand-held camera. *Apple iMovie '09* already included this function. Cameras on robots suffer similar vibration due to their mobility. The stabilization is also useful for outdoor vehicles on rough terrain and visual surveillance systems.

Video stabilization generally follows four steps: motion estimation, motion filtering, image warping, and image enhancement. **Motion estimation** is to extract camera motion from images. Three viewpoints are useful to pursue previous works: motion model, (local) motion clues, and (global) motion calculation. Table I shows motion estimation of recent stabilization researches. 2D motion model, especially affine, is favored because adjacent images have small amount of motion. 2D motion in image coordinate approximates camera motion in 3D world coordinate. However, it is difficult to approximate 3D motion in complex situation, where images have large depth variation and motion of multiple planes. Lee *et al.* [2] utilized 3D model incorporated with stereo vision. Pan *et al.* [8] and Zhu *et al.* [9] tackled 2.5D motion model, which uses fore-/background motion segmentation and multiple 2D motion models, respectively. Motion clues are used as data to estimate paramters of the assigned motion model. Two different approaches have been utilized, which are pixel-based and feature-based approaches [10]. Pixel-based approaches use intensity of each pixel directly. Motion between two images is determined to minimize difference measure in spatial domain, which needs complex optimization techniques [11]. Phase correlation, performed in frequency domain, also used to estimate translation of two images [12]. Pixel-based motion estimation needs long computation time and falls in local optima, so it is not popular recently. The other approaches use local features such as KLT tracker and SIFT (Table I). Global motion between two images is calculated using corresponding features, which are decided by tracking and matching. Tracking such as optical flow has small computation time, but it is difficult to track big amount of motion. In contrast, matching is good at big motion, but it needs more computation time. Outlier motion is generated by moving objects or wrongly corresponded features, which degrades accuracy of global motion estimates. Therefore, almost all feature-based techniques incorporated with a robust estimator such as Hough transform, least median squares (LMedS), and RANSAC.

Motion filtering is to separates intended motion and undesired motion. Undesired motion is generally regarded as high frequency motion such as vibration, so motion filtering is called motion smoothing. Low-pass filtering is achieved by FIR, IIR, Kalman [11], [8], [2], and particle filter [7]. Motion vector integration (MVI) is the first order IIR filter [4]. Gaussian kernel smoothing is weighted average of past k and future k motions [5], [3], so it is not appropriate for on-line application. Pilu [13] proposed Viterbi method, which is the theoretically most stable, but it is not on-line method. **Image warping** removes unwanted motion, which is inverse transform of unwanted motion. **Image enhancement** includes video completion and image deblurring. Video completion fills missing parts of the compensated image [11], [5], [3], and image deblurring restores blur by camera motion [5].

Video stabilization includes many research topics, and this paper focuses on robust motion estimation in varying ratio of outlier motion. Motion estimation is the essential step to accomplish high performance in the stabilization. Hough transform needs huge amounts of memory to keep the parameter space, and LMedS fails when outliers are more than half. RANSAC [14] had been regarded simple but powerful method for the outlier problem in the computer vision. Battiato *et al.* [1] also employed their variant of RANSAC for outlier rejection. RANSAC needs to adjust the number of iteration, which is highly related with accuracy of estimation. Adaptive RANSAC, called as *uMLESAC*, is proposed in Section II, which is improvement of authors'

This work was supported partly by the R&D program of the Korea Ministry of Knowledge and Economy (MKE) and the Korea Evaluation Institute of Industrial Technology (KEIT). (2008-S-031-01, Hybrid u-Robot Service System Technology Development for Ubiquitous City)

Sunglok Choi and Wonpil Yu are with Robot Research Department, Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea {sunglok, ywp}@etri.re.kr

Taemin Kim is with Intelligent Robotics Group, NASA Ames Research Center, Moffett Field, CA taemin.kim@nasa.gov

Authors	Motion Model	(Local) Motion Clues	(Global) Motion Calculation
Battiato *et al.* 2008 [1]	Similarity (2D, 4DOF)	Feature Matching (Block)	Least Squares + Pre-/Memory Filter + RANSAC
Lee *et al.* 2008 [2]	Euclidean (3D, 6DOF)	Feature Tracking (KLT Optical Flow)	Least Squares + X84 Outlier Rejection
Hu *et al.* 2007 [3]	Affine (2D, 6DOF)	Feature Matching (SIFT)	Least Squares + Hough Transform
Battiato *et al.* 2007 [4]	Similarity (2D, 4DOF)	Feature Matching (SIFT)	Least Squares + Thresholding
Matsushita *et al.* 2006 [5]	Affine (2D, 6DOF)	Feature Tracking (KLT Optical Flow)	Shum and Szeliski' Image Alignment
Chang *et al.* 2006 [6]	Simplified Affine (2D, 4DOF)	Feature Tracking (Optical Flow)	Trimmed Least Squares
Yang *et al.* 2006 [7]	Affine (2D, 6DOF)	Feature Matching (SIFT)	Least Squares + (Particle Filter)

TABLE I

RECENT THREE-YEAR WORKS ON VIDEO STABILIZATION

previous work [15]. uMLESAC uses an error model, which has two parameters, the ratio of inliers and the magnitude of inlier noise, where inliers are data with small noise compared with outliers. This paper explains difficulty of simultaneous estimation of two parameters of the error model. The improvement in estimating the parameters and its discussion is presented in Section II.A. Robust video stabilization is described in Section III, which employs uMLESAC for motion estimation. Other parts follow commonly used techniques: 2D affine motion model, KLT optical flow, and Kalman filter. Experiments using four image sequences verified performance of uMLESAC compared with other two robust estimators, which is presented in Section IV. Three robust estimators were tuned at three image sequences to have the best performance. Three estimators were applied to the other image sequence, which has worse situation, to observe performance without extra tuning. Experimental results showed that uMLESAC sustain similar performance in the new image sequence. Conclusion includes summary and further works in Section V.

II. uMLESAC, ADAPTIVE RANSAC

uMLESAC is a robust estimator, which strengthens existing estimators (e.g. least squares) to cope with outliers. It is based on MLESAC [16], R-RANSAC [17], and authors' previous work [15]. Formal problem definition is described in [15].

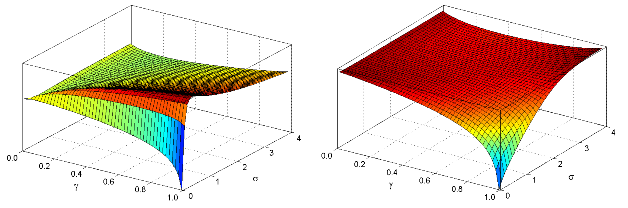
A. Error Model and Its Estimation

uMLESAC uses Torr and Zisserman' error pdf [16]. It models inlier error pdf as *unbiased Gaussian distribution* and outlier error pdf as *uniform distribution* as follows:

$$\mathrm{p}(e|M) = \gamma \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{e^2}{2\sigma^2}\right) + (1-\gamma)\frac{1}{\nu}\,, \quad (1)$$

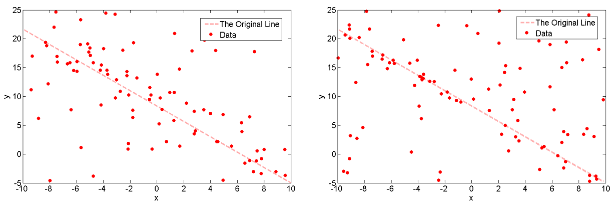
where M is the model to estimate (e.g. affine motion) and ν is the size of error space. The model has two parameters γ and σ^2, where γ is prior probability of being an inlier and σ^2 is variance of Gaussian noise. The parameter γ means the ratio of inliers to whole data and σ^2 means the magnitude of noise which contaminate inliers. The error model assumes that every datum has the same inlier prior probability. It leads posterior probability of being an inlier as follows:

$$\pi_i = \frac{\gamma \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{e^2}{2\sigma^2}\right)}{\gamma \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{e^2}{2\sigma^2}\right) + (1-\gamma)\frac{1}{\nu}}\,. \quad (2)$$



(a) $\gamma = 0.9$ *and* $\sigma = 0.3$ (b) $\gamma = 0.3$ *and* $\sigma = 2.0$

Fig. 1. Expected Value of Log Likelihood ($N = 1000$)



(a) $\gamma = 0.7$ *and* $\sigma = 2.5$ (b) $\gamma = 0.4$ *and* $\sigma = 0.5$

Fig. 2. Two Sets of Data from A Line ($N = 100$)

EM gives iterative solution of two parameters γ and σ^2 with respect to the given data as follows:

$$\gamma = \frac{1}{N}\sum_{i=1}^{N}\pi_i \quad \text{and} \quad \sigma^2 = \frac{\sum_{i=1}^{N}\pi_i e_i^2}{\sum_{i=1}^{N}\pi_i}\,, \quad (3)$$

where N is the number of data. The authors' previous work [15] used both equations to estimate γ and σ^2 simultaneously. However, the estimation often failed in case of low inlier ratio or much more complex noise than the error model (1). Figure 1 presents expectation of log likelihood in two different data sets. Data with low inlier ratio has two maxima near ($\gamma < 0.2$, $\sigma < 2$) and ($\gamma > 0.8$, $\sigma > 3$). It means that low inlier ratio distribution can be estimated as higher inlier ratio with bigger noise. Figure 1 also shows ambiguity of two parameters, which has two similar data distribution in spite of significantly different γ and σ. uMLESAC estimates only γ using EM and remains σ^2 as a configuration parameter to resolve ambiguity of γ and σ^2. EM procedure is described in Figure 3 in detail.

B. Model Evaluation

uMLESAC uses maximum likelihood (ML) criterion to select the best model among many estimated models which are generated each iteration. Under Naïve assumption, the likelihood becomes

$$\mathrm{p}(\mathcal{E}|M) = \prod_{i=1}^{n} \mathrm{p}(e_i|M) \,. \tag{4}$$

uMLESAC uses negative log likelihood as follows:

$$\mathrm{NLL}(M) = -\log \prod_{i=1}^{n} \mathrm{p}(e_i|M) = -\sum_{i=1}^{n} \log \mathrm{p}(e_i|M) \,, \tag{5}$$

which makes a small likelihood value numerically possible in the digital computer. The estimation problem is formulated as an optimization problem as follows:

$$\hat{M} = \arg\min_{M} \mathrm{NLL}(M) \,. \tag{6}$$

uMLESAC adopts preliminary evaluation to reject incorrect models before EM iteration and model evaluation. It can reduce computation time because two expensive steps are skipped. $T_{d,d}$ test [17] is utilized, which is passed when an estimated model is consistent with randomly selected d data. The computation time is formulated as

$$J = T(t_M + \widetilde{N} t_E) \,, \tag{7}$$

where T is the number of iteration, t_M is time to estimate the model, t_E is time for EM iteration and evaluation procedures per one datum, and $\widetilde{N}$ is the average number of data used in EM and evaluation. The number of test data, d, is selected to minimize the computation time as follows [17]:

$$d^* = \frac{\log\left(\frac{(t_R+1)\log\gamma}{N(log\delta - log\gamma)}\right)}{\log\delta} \,, \tag{8}$$

where t_R is the ratio of time, t_M/t_E, and δ is probability that an outlier is consistent with the model, $1.96\sigma/\nu$. Preliminary evaluation is described in Figure 3 in detail.

C. Adaptive Termination

uMLESAC calculates the necessary number of iteration, T, with respect to the given data distribution. The number of iteration should be enough to satisfy two conditions: 1) *m sampled data to estimate a model are all inliers* and 2) *d sampled data for preliminary evaluation are passed in $T_{d,d}$ test*. Two conditions entail the number of iteration as

$$T = \frac{\log(1 - \mathrm{P}_s)}{\log(1 - \gamma^{m+d})} \,, \tag{9}$$

where m is the necessary number of data to estimate a model and P_s is probability to satisfy the two condition. uMLESAC can control trade-off between accuracy and computation time using P_s. Overall procedure of uMLESAC is described in Figure 3, where $\mathcal{D}$ is the given N data.

```
CONFIGURATION VARIABLES of uMLESAC
$\mathrm{P}_s$ : Success probability (0.99 is used in this paper)
$\sigma$ : The standard deviation of inlier noise
$T_{max}$ : The maximum number of iteration (600 is used in this paper)
$\delta_{em}$ : Tolerance of $\gamma$ in EM iteration (0.05 is used in this paper)
PROCEDURE of uMLESAC
$d \leftarrow 0$
$T \leftarrow T_{max}$
$NLL_{min} \leftarrow \infty$
$iteration \leftarrow 0$
WHILE UNTIL $iteration < T$
    $iteration \leftarrow iteration + 1$
    **1. Sample data randomly.**
    $\mathcal{S} \leftarrow$ random samples of $\mathcal{D}$   $(\mathrm{N}(\mathcal{S}) = m)$
    **2. Estimate $M$ from sampled data.**
    $M \leftarrow \mathrm{EstimateModel}(\mathcal{S})$
    **3. Evaluate $M$ using $T_{d,d}$ Test**
    $\mathcal{S}' \leftarrow$ random samples of $\mathcal{D}$   $(\mathrm{N}(\mathcal{S}') = d)$
    FOR EACH $dd$ OF $\mathcal{S}'$
        IF $|\mathrm{CalculateError}(dd; M)| > 1.96\sigma$ GOTO STEP 1
    ENDFOR
    **4. Calculate $\mathcal{E}$ with respect to $M$**
    $\mathcal{E} \leftarrow \mathrm{CalculateError}(\mathcal{D}; M)$
    **5. Estimate $\gamma$ using EM.**
    $\gamma \leftarrow 0.5$
    DO
        $\gamma_{prev} = \gamma$
        $\gamma \leftarrow \frac{1}{N}\sum_{i=1}^{N} \pi_i$
    WHILE UNTIL $|\gamma - \gamma_{prev}| < \delta_{em}$
    **6. Evaluate $M$ using ML.**
    IF $\mathrm{NLL}(M) < NLL_{min}$ THEN
        $NLL_{min} \leftarrow \mathrm{NLL}(M)$
        $M_{best} \leftarrow M$
        $d \leftarrow \max\left(0, \arg\min_{\lceil d^* \rceil, \lfloor d^* \rfloor} J(d)\right)$
        $T \leftarrow \min\left(\frac{\log(1-\mathrm{P}_s)}{\log(1-\gamma^{m+d})}, T_{max}\right)$
    ENDIF
ENDWHILE
RETURN $M_{best}$
```

Fig. 3. Pseudo Code of uMLESAC

III. Robust Video Stabilization

uMLESAC is utilized to estimate 2D affine motion from KLT tracker. Kalman filter separates low frequency motion from the estimated motion. Stabilization is performed by removing high frequency motion. Figure 5 presents the stabilization procedure.

A. Motion Estimation

KLT tracker [18] is utilized to extract local motion, which generates corresponding points of adjacent images. KLT tracker finds motion of two images to minimized the dissimilarity as follows:

$$\epsilon = \int\int_W \left[I_{t+1}(A\boldsymbol{x}) - I_t(\boldsymbol{x})\right]^2 w(\boldsymbol{x}) d\boldsymbol{x} \,, \tag{10}$$

where W is the given window, $w(\boldsymbol{x})$ is a weight function, and A is 3×3 affine matrix. The proposed stabilization uses OpenCV implementation of KLT, whose windows size is adjusted 20 and the weight function is simply $w(\boldsymbol{x}) = 1$.

uMLESAC is used to estimate global motion from locally extracted motion by KLT tracker. The model M is 2D affine motion model as follows:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{bmatrix} \,. \tag{11}$$

It is estimated through commonly used least squares using three corresponding points ($m = 3$), whose implementation

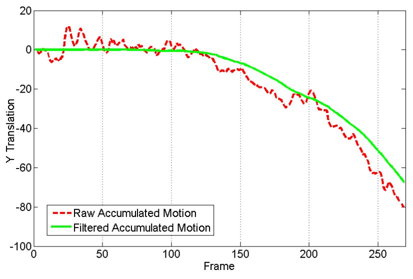


Fig. 4. Accumulated Motion (The *LightChange* Sequence)

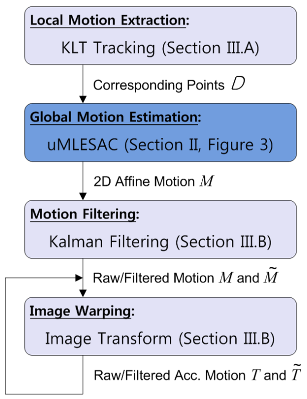

Local Motion Extraction:
KLT Tracking (Section III.A)
Corresponding Points D
Global Motion Estimation:
uMLESAC (Section II, Figure 3)
2D Affine Motion M
Motion Filtering:
Kalman Filtering (Section III.B)
Raw/Filtered Motion M and $\widetilde{M}$
Image Warping:
Image Transform (Section III.B)
Raw/Filtered Acc. Motion T and $\widetilde{T}$

Fig. 5. Flow Chart of Video Stabilization

is EstimateModel($\mathcal{S}$) function. Error between the model M and the corresponding points, $d = (\boldsymbol{x}_t,\, \boldsymbol{x}_{t+1})$, is calculated as

$$e = \left| M\boldsymbol{x}_t - \boldsymbol{x}_{t+1} \right| , \tag{12}$$

whose implementation is CalculateError($d; M$) function.

B. Motion Filtering and Image Warping

Kalman filter is employed for motion filtering, which relieves high frequency motion. Constant velocity model is used under assumption that each parameter is independent [11]. State-space and observation equations becomes

$$\begin{bmatrix} m_{ij} \\ \dot{m}_{ij} \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} m_{ij} \\ \dot{m}_{ij} \end{bmatrix}_{t} + w_{t+1} \tag{13}$$

$$\begin{bmatrix} m_{ij} \end{bmatrix}_{t+1} = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} m_{ij} \\ \dot{m}_{ij} \end{bmatrix}_{t} + v_{t+1} , \tag{14}$$

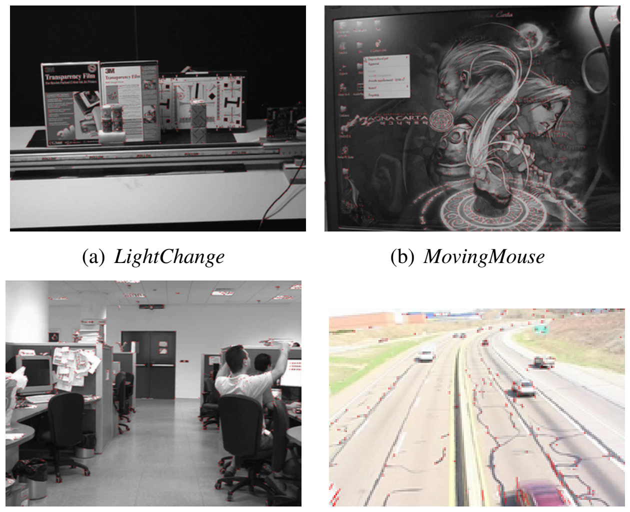
where process and measurement noise, w_{t+1} and v_{t+1}, are $N(0, Q_{ij})$ and $N(0, R_{ij})$, respectively.

Motion needs to be accumulated to compensate motion from the initial as follows:

$$T_0^{t+1} = T_0^t\, M_{t+1} \text{ and } \widetilde{T}_0^{t+1} = \widetilde{T}_0^t\, \widetilde{M}_{t+1} , \tag{15}$$

where T_0^t and $\widetilde{T}_0^t$ are raw and filtered accumulated motion and $\widetilde{M}_t$ is filtered motion of M_t via Kalman filter. Figure 4 presents an examples of T_0^t and $\widetilde{T}_0^t$. The stabilized image $\widetilde{I}_t$ is generated from the original image I_t through

$$\widetilde{I}_t = \widetilde{T}_0^t (T_0^t)^{-1}\, I_t . \tag{16}$$



(a) *LightChange* (b) *MovingMouse*

(c) *IndoorNavigation* (d) *Traffic*

Fig. 6. Four Image Sequences with KLT Trackers

IV. Experiments

A. Configuration

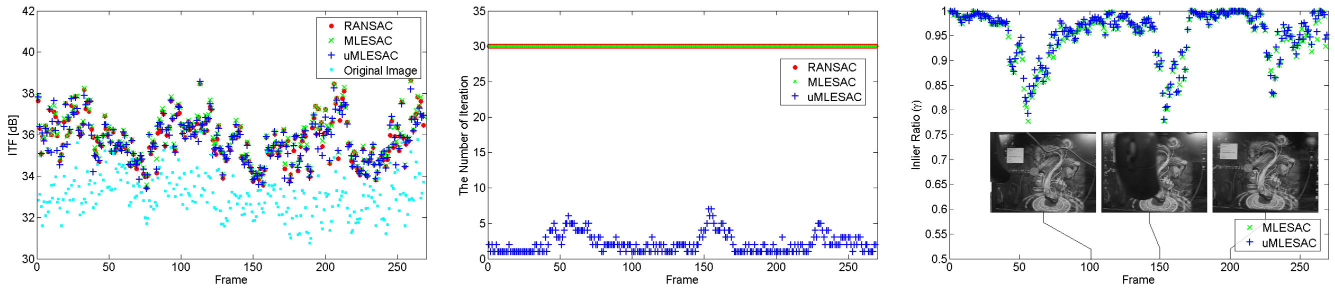
Four data sets were used to evaluate robustness of RANSAC, MLESAC, and uMLESAC. Each first image with KLT trackers is presented in Figure 6. *LightChange*, *MovingMouse*, and *IndoorNavigation* are from Battiato *et al.* [1]. *LightChange* has small zoom motion in varying illumination, *MovingMouse* contains varying outlier motion by moving mouse, and *IndoorNavigation* has shaking forward motion in an office environment. *Traffic* is from CMU PIA Consortium homepage [19], which has highly varying ratio of outlier motion by moving cars.

Performance of video stabilization is quantified by temporal smoothness and computing time. Interframe transform fidelity (ITF) is widely used to measure temporal smoothness. It is peak signal-to-noise ratio (PSNR) between adjacent images, which defined as follows:

$$\mathrm{ITF}(I_t, I_{t+1}) = 10 \log \frac{255^2}{\mathrm{MSE}(I_t, I_{t+1})} , \tag{17}$$

where MSE is mean squared error of two images. Computing time is measured using *QueryPerformanceCounter* function in Win32 API, whose resolution is upto nano second.

Experiments were performed to observe robustness of RANSAC, MLESAC, and uMLESAC in varying situation as follows. At first, parameters of three methods are adjusted to sustain high ITF with minimum computing time in three image sequences, *LightChange*, *MovingMouse*, and *IndoorNavigation*. Second, three methods are applied to the *Traffic* sequence without tuning again. ITF and computing time were recorded in each frame of image sequence. The experiments focussed on observing performance variation in the *Traffic* sequence. The number of iteration of RANSAC and MLESAC were adjusted to 30 after the first step. The threshold of RANSAC and standard deviation of MLESAC/uMLESAC were tuned as 1.96×0.3 and 0.3 respectively.



(a) ITF (b) The Number of Iteration (c) Estimated Inlier Ratio (γ)

Fig. 7. Experiment Results in the *MovingMouse* Sequence

B. Results and Discussion

1) Three Image Sequences: RANSAC, MLESAC, and uMLESAC sustained ITF more than 33 in the *LightChange* and *MovingMouse* sequence. Figure 7(a) shows ITF of the *MovingMouse* sequence. Three methods had similar ITF, but higher than the original sequence. They stabilized the *IndoorNavigation* sequence in the beginning, but they failed after 100 frames. Stabilization using 2D projective model (plannar homography) also failed. The sequence has three 2D planes (a floor, ceil, and front) and 3D complex sides (Figure 6(c)). 2D motion model can not describe them together, small amount of error was generated at each frame. Its accumulation (15) became huge after 100 frames.

uMLESAC was 3 times faster than RANSAC and 30 times faster than MLESAC on average in *LightChange* and *MovingMouse* sequence. uMLESAC had similar computing time with RANSAC and 10 times faster than MLESAC on average in *IndoorNavigation* sequence. RANSAC and MLESAC use the constant number of iteration all the time, so they should always consider the worst situation. However, adaptive termination of uMLESAC controled the number of iteration with respect to the estimated inlier ratio, γ. uMLESAC iterated few times in most of time and it iterated more in worse situation. Figure 7(b) and 7(c) present the number of iteration and estimated inlier ratio in *MovingMouse*.

uMLESAC with $T_{d,d}$ test had simliar computing time with uMLESAC without it. In other words, preliminary evaluation using $T_{d,d}$ test did not accelerate the algorithm. $T_{d,d}$ test can reduce computing time if $J(0) > J(d)$ is valid [17], which entails

$$N > (t_r + 1)\frac{1-\gamma^d}{\gamma^d - \delta^d} \, . \tag{18}$$

All three sequences satisfied the condition, but acceleration was not significant because most of each frame needs the small number of iteration.

2) The Traffic Image Sequence: uMLESAC sustained ITF more than 34 dB, but RANSAC and MLESAC had lower ITF near 120 frame (Figure 8(a)). The *traffic* sequence had many moving cars near 120 frame, which generated outlier motion much more than before. It caused inaccurate global motion estimate and big drift error as shown Figure 9(a). The estimated inlier ratio by MLESAC and uMLESAC represented the degree of difficulty in this situation, whose value was near 0.2 (Figure 8(c)). RANSAC and MLESAC had similar ITF with uMLESAC after tuning their iteration to 80, but it increased computing time 2.5 times more.

uMLESAC was 2 times faster than RANSAC and 20 times faster than MLESAC on average. It was much slower than both near 120 frame, and 6 times slower than RANSAC near 550 and 900 frames.

V. Conclusion

uMLESAC is adaptive extension of MLESAC [16]. It uses the mixture of Gaussian and uniform distribution to model probability density of inlier and outlier. EM gives iterative solution to two parameters of the distribution, γ and σ, with respect to the given data. This paper showed ambiguity between two parameters and claimed difficulty of finding their true values simultaneously. uMLESAC estimates only γ using EM and calculates the necessary number of iteration for the given distribution.

Experiments verified performance of uMLESAC for video stabilization on varying ratio of outlier motion. uMLESAC sustained high ITF in spite of the new image sequence. The adaptive termination accelerated the alogrithm when few iteration was enough to estimation motion near the truth.

Many meaningful works have been performed on video stabilization, but the video stabilization needs to be investigated more. KLT tracker and Kalman filter are required to be self-adjusted for advanced adaptation. A motion model needs to be examined more. The experiment using *IndoorNavigation* sequences presented limitation of 2D motion model. Accumulated error through (15) should be solved for long-term surveillance system.

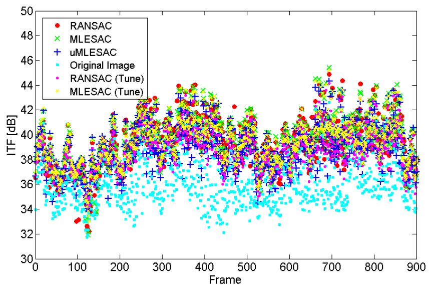
Acknowledgment

The authors would like to thank Yeon Geul Ryu for his research materials, Dong-Hyun Lee for his sincere comment, and Battiato *et al.* [1] for their image sequences.

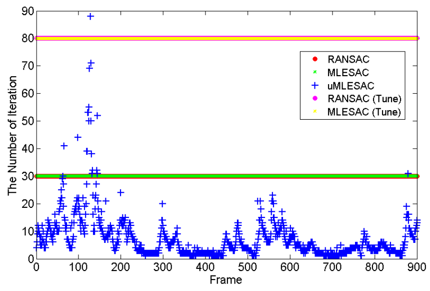
References

[1] S. Battiato, G. Puglisi, and A. R. Bruna, “A robust video stabilization system by adaptive motion vectors filtering,” in *Proceedings of the IEEE International Conference on Multimedia & Expo (ICME)*, 2008.

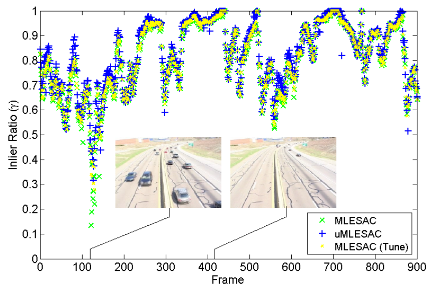
[2] K. K. Lee, K. H. Wong, M. M. Y. Chang, Y. K. Yu, and M. K. Leung, “Extended kalman filtering approach to stereo video stabilization,” in *Proceedings of the IEEE International Conference on Pattern Recognition (ICPR)*, 2008.



(a) ITF

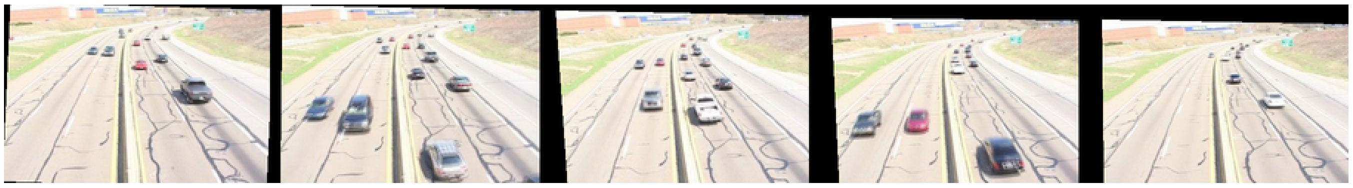


(b) The Number of Iteration

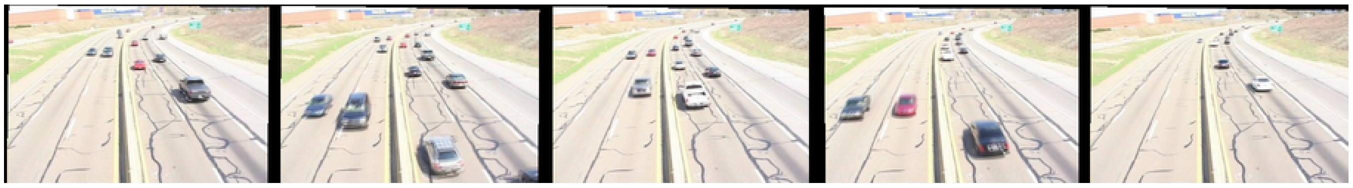


(c) Estimated Inlier Ratio (γ)

Fig. 8. Experiment Results in the *Traffic* Sequence



(a) Stabilization using RANSAC



(b) Stabilization using uMLESAC

Fig. 9. The Stabilized *Traffic* Sequence (80, 120, 160, 200, 240 frame)

[3] R. Hu, R. Shi, I. fan Shen, and W. Chen, "Video stabilization using scale-invariant features," in *Proceedings of the IEEE International Conference on Information Visualization (IV)*, 2007.

[4] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato, "Sift features tracking for video stabilization," in *Proceedings of the IEEE International Conference on Image Analysis and Processing (ICIAP)*, 2007.

[5] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 7, pp. 1150–1163, 2006.

[6] H.-C. Chang, S.-H. Lai, and K.-R. Lub, "A robust real-time video stabilization algorithm," *Journal of Visual Communication and Image Representation*, vol. 17, no. 3, pp. 659–673, 2006.

[7] J. Yang, D. Schonfeld, C. Chen, and M. Mohamed, "Online video stabilization based on particle filters," in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2006.

[8] Z. Pan and C.-W. Ngo, "Selective object stabilization for home video consumers," *IEEE Transactions on Consumer Electronics*, vol. 51, no. 4, pp. 1074–1084, 2005.

[9] Z. Zhu, G. Xu, Y. Yang, and J. S. Jin, "Camera stabilization based on 2.5d motion estimation and inertial motion filtering," in *Proceedings of the IEEE International Conference on Intelligent Vehicles*, 1998.

[10] R. Szeliski, "Image alignment and stitching: A tutorial," Microsoft Research, Tech. Rep. MSR-TR-2004-92, 2004.

[11] A. Litvin, J. Konrad, and W. C. Karl, "Probabilistic video stabilization using kalman filtering and mosaicking," in *Proceedings of IS&T/SPIE Symposium on Electronic Imaging, Image and Video Communications and Processing*, 2003.

[12] C. Guestrin, F. Cozman, and E. Krotkov, "Fast software image stabilization with color registration," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1998.

[13] M. Pilu, "Video stabilization as a variational problem and numerical solution with the viterbi method," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[14] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[15] S. Choi and J.-H. Kim, "Robust regression to varying data distribution and its application to landmark-based localization," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, October 2008.

[16] P. Torr and A. Zisserman, "MLESAC: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, pp. 138–156, 2000.

[17] O. Chum and J. Matas, "Randomized RANSAC with $T_{d,d}$ test," in *Preceedings of the 13th British Machine Vision Conference (BMVC)*, 2002, pp. 448–457.

[18] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994.

[19] "Carnegie Mellon University People Image Analysis (PIA) Consortium." http://www.consortium.ri.cmu.edu/