BMVC 2017 Sub#153: SUPPLEMENTARY MATERIAL

A.1 *Analytical Solution (solution given in Equation (7) for the stabilization function in Equation (6))*

An optimization function has been proposed for stabilization for each block. The estimated Frechet mean and the stabilized Frechet mean are represented as F_m and F_m^s respectively. For row r and column c (c=1,2), let $F_m(r,c)$ and $F_m^s(r,c)$ represent the elements of F_m and F_m^s matrices respectively. The stabilization is done on x and y components of the Frechet mean individually (separately).

For the first block (from Equation (6)),

$$\hat{F}_m^s = argmin_{F_m^s}(||F_m^s - F_m||^2 + \lambda \sum_{i=1}^{N} ||F_m^s(i,:) - \tilde{F}_m^s)^2||^2) \tag{A.1.1}$$

where, $\tilde{F}_m = \frac{1}{N}\sum_{j=1}^{N} F_m^s(j,:)$

Solving using the Jacobi iteration method [1] for each column vector of the Frechet mean ($F_m(:,c)$) of the background trajectories, as:

$\frac{\partial(\mathcal{F}\{F_m^s\})}{\partial(F_m(r,c))} = 0$

We get:

$2(F_m^s(r,c) - F_m(r,c)) + 2\lambda \sum_{i=1,i\neq r}^{N} (F_m^s(i,c) - \tilde{F}_m(c))(-\frac{1}{N}) + 2\lambda(F_m^s(r,c) - \tilde{F}_m(c))(1-\frac{1}{N}) = 0$

$F_m^s(r,c) - F_m(r,c) - \frac{\lambda}{N}\sum_{i=1,i\neq r}^{N} F_m^s(i,c) + (\lambda - \frac{\lambda}{N})F_m^s(r,c) + \frac{\lambda}{N}\sum_{i=1,i\neq r}^{N} \tilde{F}_m(c) - (\lambda - \frac{\lambda}{N})\tilde{F}_m(c) = 0$

$(F_m^s(r,c) - F_m(r,c)) - \frac{\lambda}{N}\sum_{i=1,i\neq r}^{N} F_m^s(i,c) + \lambda F_m^s(r,c) - \frac{\lambda}{N}F_m^s(r,c) = 0$

$F_m^s(r,c)(1+\lambda-\frac{\lambda}{N}) = F_m(r,c) + \frac{\lambda}{N}\sum_{i=1,i\neq r}^{N} F_m^s(i,c)$

$F_m^s(r,c) = \alpha F_m(r,c) + \beta \sum_{i=1,i\neq r}^{N} F_m^s(i,c)$

Thus, the solution for eqn (A.1.1) as an iterative solution is,

$$(\hat{F}_m^s(r,c))^{t+1} = \alpha F_m(r,c) + \beta \sum_{i=1,i\neq r}^{N} (\hat{F}_m^s(i,c))^t \tag{A.1.2}$$

Obtaining the solution for both x and y components, Equation A.1.2 can be written as:

$$(\hat{F}_m^s(r,:))^{t+1} = \alpha F_m(r,:) + \beta \sum_{i=1,i\neq r}^{N} (\hat{F}_m^s(i,:))^t \tag{A.1.3}$$

where, $\alpha = \frac{N}{N+\lambda N-\lambda}$, $\beta = \frac{\lambda}{N} * \alpha$ and t is the iteration index.

Thus the equation in A.1.3 provides the solution in Equation 7, in main document.

A.2 *Quantitative results of 20 real-world natural jittery videos*

The quantitative results of foreground (moving) object segmentation for 20 real-world natural jittery videos are given in table A.2.1 below:

	IoU scores of competing methods					
Video samples	[12]	[9]	[8]	[4]	[11]	Proposed
Walk1	0.401	0.135	0.02	0.715	0.139	**0.720**
Walk2	0.009	0.123	0	0.480	0.151	**0.841**
Cheery_Girl	0.745	0.201	0.09	0.587	0.573	**0.756**
Doll	0.139	0.926	0.819	0.350	0.078	**0.933**
Baby	0.116	0.671	0.007	0.360	0.222	**0.847**
Skating1	0.033	0.248	0.318	0.627	0.523	**0.713**
Skating2	-	0.106	0.327	0.531	0.536	**0.596**
Car	0.029	0.06	0.058	0	0	**0.233**
Climb1	0.54	0.764	0.03	**0.844**	0.476	0.810
Climb2	**0.591**	0.024	0.418	0.443	0.416	0.505
Drone1	0.715	0.755	0.658	0.703	0.689	**0.770**
Drone2	0.487	0.436	0.549	0.325	0.348	**0.588**
Drone3	0.41	0.531	0.561	0.601	0.630	**0.661**
Dog	0.736	0.733	0.559	0.758	0.775	**0.785**
Train	0.211	0.37	0.535	0.837	0.831	**0.850**
Cycling1	0.558	0.359	0.610	0.462	0.342	**0.613**
Cycling2	0.654	0.649	0.462	0.831	0.689	**0.833**
Cycling3	0.701	0.342	0.305	0.490	0.401	**0.723**
Staircase1	0.726	0.296	0.713	0.651	0.488	**0.782**
Staircase2	0.875	0.889	0.801	0.001	0.103	**0.901**
Average	0.456	0.431	0.392	0.529	0.421	**0.723**

Table A.2.1: Performance analysis of segmentation on 20 real-world jittery videos in terms of IoU score (higher, the better).

Our method performs the best for all the methods, except in two cases, where ours is a close second in both. On an average over all the 20 videos (see bottom row in table A.2.1), our method outperforms the other methods by a large margin. The methods [8, 9, 11, 12] depend on the local motion of the pixels in the frames for distinguishing between the foreground and the background, which fail in case of jittery videos. The method [4] (second ranked, based on average IoU score) performs a nearest neighbor search with visual cues along with the motion cues. Hence, this method performs the best after ours, although the gap is appreciable.

The qualitative results of the videos are uploaded along with this document.

A.3 *Artificial Jitter Simulation*

For performance analysis of the proposed method on jittery videos, jitter is simulated and introduced into stable videos of the standard segmentation dataset, SegTrackv2 [2]. For this purpose, a jitter pattern is captured from a real world jittery video, using homography [6] estimation. This entire process is described below:

The stabilization technique in [7] is first performed on an unstable video to get the stabilized video. For both unstable and stabilized videos, average homography is estimated using RANSAC (RANdom SAmple Consensus) [5] for each pair of successive frames. For estimating the homographies, we used the point trajectories described in [10]. Let $\{H_t\}$ be the set of homographies of the jittery video and $\{H_t^s\}$ be the set of homographies of the corresponding stable video, where t is the frame (or time) index. Let, the homography between the unstable frame and corresponding stable frame be J_t. This homography, J_t (termed jitter matrix), follows a model:

$$J_t = \prod_{i=t-1}^{1} H_i \Big(\prod_{i=t-1}^{1} H_i^s\Big)^{-1}, \forall t \tag{A.3.1}$$

Any set of stable video frames can now be warped with the jitter matrices ($J_t, \forall t$), using eqn (A.3.1) to synthetically produce the jittery (unstable) frames. Let S_t be a stable frame and S_t^J be the synthetically produced jittery frame of S_t. The frame S_t^J is obtained by warping the stable frame with the jitter matrix (J_t), i.e $S_t^J = J_t S_t$.

The method of jitter simulation provides a control of the different levels of jitter introduced. Parameters are extracted from jitter matrices, estimated frame-wise using eqn (A.3.1). The jitter matrix (J_t) is assumed to be an affine matrix and is decomposed into translation, rotation, shear and scale matrices [1]. The parameters of only the translation, rotation and shear matrices are then perturbed to fuse jitter. For each frame, the parameters of the matrices are randomly perturbed by adding small random values (Gaussian distributed iid, with standard deviation σ). Let $\mathcal{P}_t$ be a parameter of J_t, and $\mathcal{P}_t^\sigma$ be the perturbed transformation parameter. Perturbation is done by adding a random value X_t^σ for frame t, to obtain the altered transformation parameters as:

$$\mathcal{P}_t^\sigma = \mathcal{P}_t(1 + c_t * X_t^\sigma), \forall t \tag{A.3.2}$$

where, c_t is a scale factor, which is proportional to the range of parameter values of J_t for the video. Altered parameters $\mathcal{P}_t^\sigma$ are used to form J_t^σ, with parameter σ controlling the degree of randomness added. This alteration is done for all the transformation parameters (translation, rotation and shear) of J_t, using random values. The levels of σ (for Gaussian iid) are chosen as follows:

(i) Low level of jitter with $\sigma = 0.05$,

(ii) Medium level of jitter with $\sigma = 0.15$, and

(iii) High level of jitter with $\sigma = 0.25$

The perturbed jitter matrix (J_t^σ) is used to warp the stable frame, S_t, to get the jittery frame, S_t^J, i.e $S_t^J = J_t^\sigma S_t, \forall t$. This in effect alters the homography to a small degree, introducing frame-wise jitter on smooth transitions of a stabilized video. With increasing levels (σ) of jitter added to stable frames, we obtain three levels of synthetic jitters with varying disturbances, and then evaluate the performance.

As a part of experimentation, for three levels of jitter ($\sigma = 0.05, 0.15, 0.25$), three sets of jitter matrices (from three unstable-stable pair of real-world natural jittery videos) were formed and fused into 8 stable videos of SegTrack2 [2] to generate 72 (8*3*3) synthetic jittery videos. The randomness of perturbation is caused by the parameter σ (see eqn A.3.2).

The average of the segmentation results on each level of jitter (24 videos for each jitter level) is given in the table below:

	IoU scores of competing methods					
Jitter Level (σ)	[12]	[9]	[8]	[4]	[11]	Proposed
Low ($\sigma = 0.05$)	0.586	0.637	0.327	0.692	0.535	**0.695**
Medium ($\sigma = 0.15$)	0.551	0.575	0.525	0.686	0.506	**0.690**
High ($\sigma = 0.25$)	0.543	0.585	0.479	0.654	0.470	**0.688**

Table A.3.1: Comparison of the average (over 72 videos) segmentation performances for low,medium and high levels of jitter, fused into stable videos (8) of SegTrack2 dataset [2] (higher, the better).

As in table A.3.1, on an average for each of the jitter level fused, our method performs the best. The method [4] in this case performs the second best, but with a degraded performance compared to our proposed approach. Qualitative results for the highest jitter level is uploaded along with this document.

References

[1] Linear Algebra-Decomposition of a Nonlinear Affine Matrix. http://math.stackexchange.com/questions/78137/decomposition-of-a-nonsquare-affine-matrix/.

[2] SegTrack v2 Dataset. http://web.engr.oregonstate.edu/~lif/SegTrack2/dataset.html.

[3] Ilía Nikolaevich Bronshtein, Konstantin A Semendyayev, Gerhard Musiol, and Heiner Muehlig. *Handbook of mathematics*. Springer Science & Business Media, 2007.

[4] Alon Faktor and Michal Irani. Video segmentation by non-local consensus voting. In *BMVC*, 2014.

[5] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[6] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.

[7] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM TOG*, 32(4):78, 2013.

[8] Peter Ochs, Jitendra Malik, and Thomas Brox. Segmentation of moving objects by long term video analysis. *IEEE TPAMI*, 36(6):1187–1200, 2014.

[9] Anestis Papazoglou and Vittorio Ferrari. Fast object segmentation in unconstrained video. In *ICCV*, 2013.

[10] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *ECCV*. 2010.

[11] Wenguan Wang, Jianbing Shen, and Fatih Porikli. Saliency-aware geodesic video object segmentation. In *CVPR*, 2015.

[12] Dong Zhang, Omar Javed, and Mubarak Shah. Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*, 2013.