EgoSampling: Wide View Hyperlapse From Egocentric Videos

Tavi Halperin, Yair Poleg, Chetan Arora, and Shmuel Peleg

***Abstract*— The possibility of sharing one's point of view makes the use of wearable cameras compelling. These videos are often long, boring, and coupled with extreme shaking, as the camera is worn on a moving person. Fast-forwarding (i.e., frame sampling) is a natural choice for quick video browsing. However, this accentuates the shake caused by natural head motion in an egocentric video, making the fast-forwarded video useless. We propose EgoSampling, an adaptive frame sampling that gives stable, fast-forwarded, hyperlapse videos. Adaptive frame sampling is formulated as an energy minimization problem, whose optimal solution can be found in polynomial time. We further turn the camera shake from a drawback into a feature, enabling the increase in field of view of the output video. This is obtained when each output frame is mosaiced from several input frames. The proposed technique also enables the generation of a single hyperlapse video from multiple egocentric videos, allowing even faster video consumption.**

***Index Terms*— Egocentric video, fast-forward, hyperlapse, video stabilization.**

I. Introduction

WHILE the use of egocentric cameras is on the rise, watching raw egocentric videos is unpleasant. These videos, captured in an "always-on" mode, tend to be long, boring, and unstable. Video summarization [1]–[3], temporal segmentation [4], [5], and action recognition [6], [7] methods can help browse and consume large amount of egocentric videos. However, these algorithms make strong assumptions in order to work properly (e.g., faces are more important than unidentified blurred images). The information produced by these algorithms helps the user skip most of the input video. Yet, the only way to watch a video from start to end, without making strong assumptions, is to play it in a fast-forward manner. However, the natural camera shake gets amplified in naïve fast-forward (i.e., frame sampling). An exceptional tool for generating stable fast-forward video is the recently proposed "Hyperlapse" [8]. Our work was inspired by [8], but take a different, lighter, approach.

Manuscript received April 24, 2016; revised September 7, 2016 and November 3, 2016; accepted December 27, 2016. Date of publication January 10, 2017; date of current version May 4, 2018. This research was supported in part by the Israel Ministry of Science, in part the by Israel Science Foundation, in part by DFG, in part by the Intel ICRI-CI, and in part by the Google. This paper was recommended by Associate Editor J. Yuan.

T. Halperin, Y. Poleg, and S. Peleg are with the Hebrew University of Jerusalem, Jerusalem, Israel (e-mail: tavi.halperin@mail.huji.ac.il; yair.poleg@mail.huji.ac.il; peleg@mail.huji.ac.il).

C. Arora is with Indraprastha Institute of Information Technology Delhi, New Delhi, India (e-mail: chetan@iiitd.ac.in).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2017.2651051

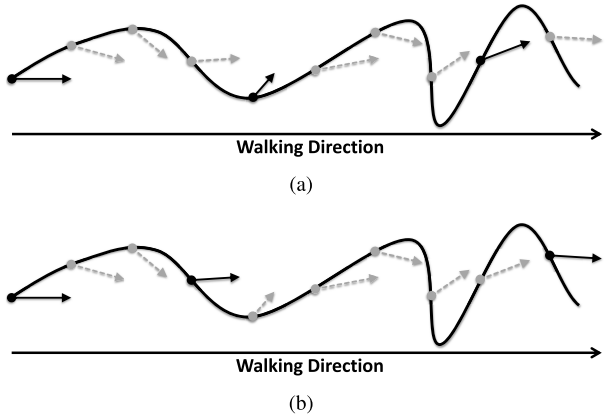

Fig. 1. Frame sampling for Fast-forward. A view from above on the camera path (the line) and the viewing directions of the frames (the arrows) as the camera wearer walks forward for a couple of seconds. (a) Uniform 5× frames sampling, shown with solid arrows, gives output with significant changes in viewing directions. (b) Our frame sampling, represented as solid arrows, prefers forward-looking frames at the cost of somewhat nonuniform sampling.

Fast-forward is a natural choice for faster browsing of videos. While naïve fast-forward uses uniform frame sampling, adaptive fast-forward approaches [9] try to adjust the speed in different segments of the input video. Sparser frame sampling gives higher speed-ups in stationary periods, and denser frame sampling gives lower speed-ups in dynamic periods. In general, content-aware techniques adjust the frame-sampling rate based upon the importance of the content in the video. Typical importance measures include motion in the scene, scene complexity, and saliency. None of the aforementioned methods, however, can handle the challenges of egocentric videos, as we describe next.

Borrowing the terminology of [4], we note that when the camera wearer is "stationary" (e.g., sitting or standing in place), head motions are less frequent and pose no challenge to traditional fast-forward and stabilization techniques. Therefore, in this paper, we focus only on cases when the camera wearer is "in transit" (e.g., walking, cycling, and driving), and often with substantial camera shake.

Kopf *et al.* [8] recently proposed to generate hyperlapse egocentric videos by 3D reconstruction of the input camera path. A smoother camera path is calculated, and new frames are rendered for this new path using the frames of the original video. Generated video is very impressive, but it may take hours to generate minutes of hyperlapse video. Joshi *et al.* [10] proposed to replace 3D reconstruction by smart sampling of

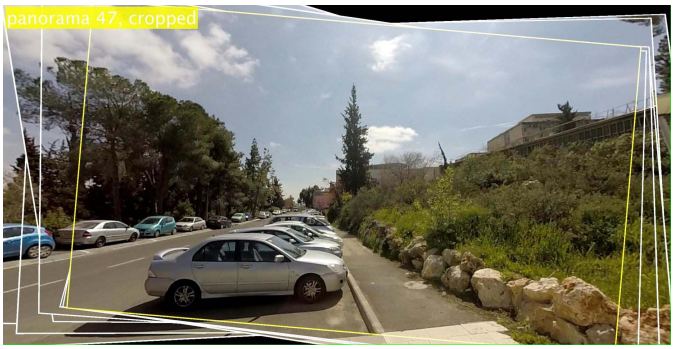

Fig. 2. Output frame produced by the proposed Panoramic Hyperlapse. We collect frames looking into different directions from the video and create mosaics around each frame in the video. These mosaics are then sampled to meet playback speed and video stabilization requirements. Apart from being fast-forwarded and stabilized, the resulting video now also has a wide FOV. The white lines mark the different original frames. The proposed scheme turns the problem of camera shake present in egocentric videos into a feature, as the shake helps to increase the FOV.

the input frames. They bias the frame selection in favor of the forward-looking frames, and drop frames that might introduce shake.

We model frame sampling as an energy minimization problem. A video is represented as a directed acyclic graph whose nodes correspond to input video frames. The weight of an edge between nodes corresponding to frames t and $t+k$ indicates how "stable" the output video will be if frame $t+k$ will immediately follow frame t. The weights also indicate if the sampled frames give the desired playback speed. Generating a stable fast-forwarded video becomes equivalent to finding a shortest path in this graph. We keep all edge weights nonnegative, and note that there are numerous polynomial time algorithms for finding a shortest path in such graphs. The proposed frame sampling approach, which we call EgoSampling, was initially introduced in [11]. We show that sequences produced with EgoSampling are more stable and easier to watch than traditional fast-forward methods.

Frame sampling approaches such as EgoSampling described above, as well as [8], [10], drop frames to give a stabilized video, with a potential loss of important information. In addition, a stabilization postprocessing is commonly applied to the remaining frames, a process which reduces the field of view (FOV). We propose an extension of EgoSampling, in which instead of dropping unselected frames, these frames are used to increase the FOV of the output video. We call the proposed approach Panoramic Hyperlapse. Fig. 2 shows a frame from an output Panoramic Hyperlapse generated with our method. Panoramic Hyperlapse video is easier to comprehend than [10] because of its increased FOV. Panoramic Hyperlapse can also be extended to handle multiple egocentric videos recorded by a group of people walking together. Given a set of egocentric videos captured at the same scene, Panoramic Hyperlapse can generate a stabilized panoramic video using frames from the entire set. The combination of multiple videos into a Panoramic Hyperlapse enables the consumption of videos even faster.

The contributions of this paper are as follows: 1) the generated wide FOV, stabilized, fast-forward videos are easier to comprehend than only stabilized or only fast-forward videos and 2) the technique is extended to combine multiple egocentric video taken at the same scene.

The rest of this paper is organized as follows. Relevant related work is described in Section II. The EgoSampling framework is briefly described in Section III. In Sections IV and V, we introduce the generalized for single and multiple videos, respectively. We report our experiments in Section VI, and draw the conclusion in Section VII.

II. Related Work

The related work to this paper can be broadly categorized into four categories.

A. Video Summarization

Video summarization methods scan the input video for salient events, and create from these events a concise output that captures the essence of the input video. While video summarization of third-person videos has been an active research area, only a handful of these works address the specific challenges of summarizing egocentric videos. In [2], [13], important keyframes are sampled from the input video to create a story-board summarization. In [1], subshots that are related to the same "story" are sampled to produce a "story-driven" summary. Such video summarization can be seen as an extreme adaptive fast-forward, where some parts are completely removed while other parts are played at original speed. These techniques require a strategy for determining the importance or relevance of each video segment, as segments removed from summary are not available for browsing.

B. Video Stabilization

There are two main approaches for video stabilization. While 3D methods reconstruct a smooth camera path [14], [15], 2D methods, as the name suggests, use 2D motion models followed by nonrigid warps [16]–[20]. As noted by [8], stabilizing egocentric video after regular fast-forward by uniform frame sampling, fails. Such stabilization cannot handle outlier frames often found in egocentric videos, e.g., frames when the camera wearer looks at his shoe for a second, resulting in significant residual shake present in the output videos.

The proposed EgoSampling approach differs from both traditional fast-forward as well as video stabilization. Rather than stabilizing outlier frames, we prefer to skip them. However, traditional video stabilization algorithms [16]–[20] can be applied as postprocessing to our method, to further stabilize the results.

Traditional video stabilization crop the input frames to create stable looking output with no empty region at the boundaries. In attempt to reduce the cropping, Matsushita *et al.* [21] suggest to perform inpainting of the video boundary based on information from other frames.

C. Hyperlapse

Kopf *et al.* [8] have suggested a pioneering hyperlapse technique to generate stabilized egocentric videos using a

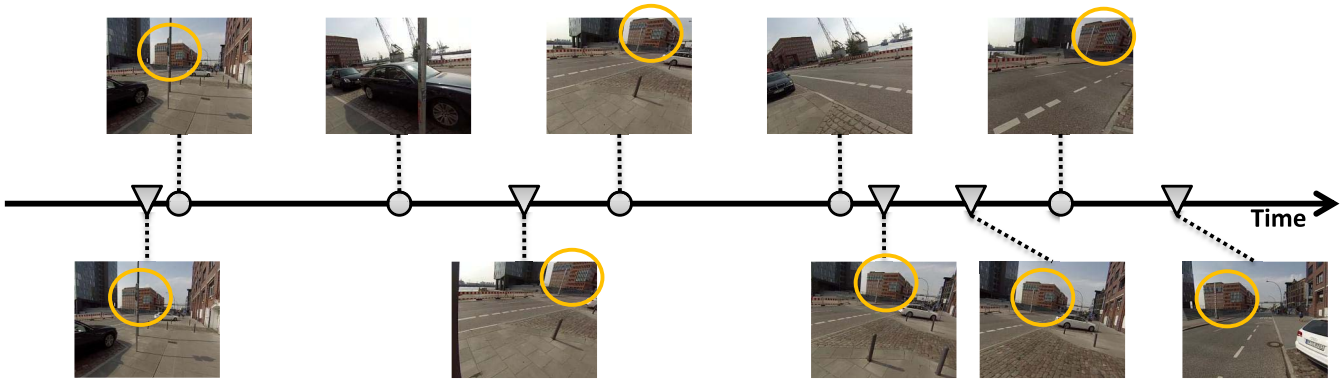

Fig. 3. Representative frames from the fast-forward results on "Bike2" sequence [12]. The camera wearer rides a bike and prepares to cross the road. Top: uniform sampling of the input sequence leads to a very shaky output as the camera wearer turns his head sharply to the left and right before crossing the road. Bottom: EgoSampling prefers forward-looking frames and therefore samples the frames nonuniformly so as to remove the sharp head motions. The stabilization can be visually compared by focusing on the change in position of the building (yellow circle) appearing in the scene. The building does not even show up in two frames of the uniform sampling approach, indicating the extreme shake. Note that the fast-forward sequence produced by EgoSampling can be postprocessed by traditional video stabilization techniques to further improve the stabilization.

combination of 3D scene reconstruction and image-based rendering techniques. A new and smooth camera path is computed for the output video, while remaining close to the input trajectory. The results produced are impressive, but may be less practical because of the large computational requirements. In addition, 3D recovery from egocentric video may often fail. A similar paper to our EgoSampling approach, [10], avoids 3D reconstruction by posing hyperlapse as frame sampling, and can even be performed in real time.

Sampling-based hyperlapse such as EgoSampling proposed by us or [10] bias the frame selection toward forward-looking views. This selection has two effects: 1) the information available in the skipped frames, likely looking sideways, is lost and 2) the cropping, which is part of the subsequent stabilization step, reduces the FOV. We propose to extend the frame sampling strategy by Panoramic Hyperlapse, which uses the information in the side looking frames that are discarded by frame sampling.

D. Multiple Input Videos

The state-of-the-art hyperlapse techniques address only a single egocentric video. For curating multiple nonegocentric video streams, Jiang and Gu [22] suggested spatial–temporal content-preserving warping for stitching multiple synchronized video streams into a single panoramic video. Hoshen *et al.* [23] and Arev *et al.* [24] produce a single output stream from multiple egocentric videos viewing the same scene. This is done by selecting only a single input video, best representing each time period. In both the techniques, the criterion for selecting the video to display requires strong assumptions of what is interesting and what is not.

We propose Panoramic Hyperlapse in this paper, which supports multiple input videos, by fusing input frames from multiple videos into a single output frame having a wide FOV.

III. EgoSampling

The key idea in this paper is to generate a stable fast-forwarded output video by selecting frames from the input video having similar forward-viewing direction, which is also the direction of the wearer's motion. Figs. 1 & 3 intuitively describe this approach. This approach works well for forward-moving cameras. Other motion directions, e.g., cameras moving sideways, can be accelerated only slightly before becoming hard to watch.

As a measure for forward-looking direction, we find the Epipolar point between all pairs of frames, I_t and I_{t+k}, where $k \in [1, \tau]$, and τ is the maximum allowed frame skip. Under the assumption that the camera is always translating (recall that we focus only on wearer's in "transit" state), the displacement direction between I_t and I_{t+k} can be estimated from the fundamental matrix $F_{t,t+k}$ [25]. We prefer using frames whose epipole is closest to the center of the image.

Recent V-SLAM approaches such as [26], [27] provide camera ego-motion estimation and localization in real time. However, we found that the fundamental matrix computation can fail frequently when k (temporal separation between the frame pair) grows larger. As a fallback measure, whenever the fundamental matrix computation breaks, we estimate the direction of motion from the FOE of the optical flow. We do not compute the FOE from the instantaneous flow, but from integrated optical flow as suggested in [4] and computed as follows. We first compute the sparse optical flow between all consecutive frames from frame i to frame j. Let the optical flow between frames t and $t+1$ be denoted by $g_t(x, y)$ and $G_{i,j}(x, y) = (1/k) \sum_{t=i}^{j-1} g_t(x, y)$. The FOE is computed from $G_{i,j}$ as suggested in [28], and is used as an estimate of the direction of motion.

A. Graph Representation

We model the joint fast-forward and stabilization of egocentric video as graph energy minimization. The input video is represented as a graph, with a node corresponding to each frame in the video. There are weighted edges between every pair of graph nodes, i and j, with weight proportional to our preference for including frame j right after i in the output video. There are three components in this weight.

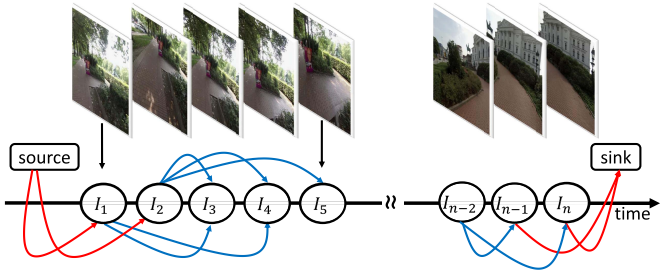

Fig. 4. We formulate the joint fast-forward and video stabilization problem as finding a shortest path in a graph constructed as shown. There is a node corresponding to each frame. The edges between a pair of frames (i, j) indicate the penalty for including a frame j immediately after frame i in the output (please refer to the text for details on the edge weights). The edges between the source/sink and the graph nodes allow skipping frames from start and end. The frames corresponding to nodes along the shortest path from the source to the sink are included in the output video.

1) Shakiness Cost ($S_{i,j}$): This term prefers forward looking frames. The cost is proportional to the distance of the computed motion direction (epipole or FOE) designated by $(x_{i,j}, y_{i,j})$ from the center of the image $(0, 0)$

$$S_{i,j} = \|(x_{i,j}, y_{i,j})\|. \tag{1}$$

2) Velocity Cost ($V_{i,j}$): This term controls the playback speed of the output video. The desired speed is given by the desired magnitude of the optical flow, K_{flow}, between two consecutive output frames

$$V_{i,j} = \left(\sum_{x,y} G_{i,j}(x, y) - K_{\text{flow}}\right)^2. \tag{2}$$

3) Appearance Cost ($C_{i,j}$): This is the earth mover's distance (EMD) [29] between the color histograms of frames i and j. The role of this term is to prevent large visual changes between frames. A quick rotation of the head or dominant moving objects in the scene can confuse the FOE or epipole computation. This term acts as an anchor in such cases, preventing the algorithm from skipping a large number of frames.

The overall weight of the edge between nodes (frames) i and j is given by

$$\mathcal{W}_{i,j} = \alpha \cdot \mathcal{S}_{i,j} + \beta \cdot V_{i,j} + \gamma \cdot C_{i,j} \tag{3}$$

where α, β, and γ represent the relative importance of various costs in the overall edge weight.

With the problem formulated as above, sampling frames for stable fast-forward is done by finding a shortest path in the graph. We add two auxiliary nodes, a *source* and a *sink* in the graph to allow skipping some frames from start or end. To allow such skip, we add zero weight edges from start node to the first D_{start} frames and from the last D_{end} nodes to sink. We then use Dijkstra's algorithm [30] to compute the shortest path between source and sink. The algorithm does the optimal inference in time polynomial in the number of nodes (frames). Fig. 4 shows a schematic illustration of the proposed formulation.

B. Second-Order Smoothness

The formulation described in the previous section prefers to select forward-looking frames, where the epipole is closest to the center of the image. With the proposed formulation, it may so happen that the epipoles of the selected frames are close to the image center, but on the opposite sides, leading to a jitter in the output video. In this section, we introduce an additional cost element: stability of the location of the epipole. We prefer to sample frames with minimal variation of the epipole location.

To compute this cost, nodes now represent two frames, as can be seen in Fig. 6. The weights on the edges depend on the change in epipole location between one image pair to the successive image pair. Consider three frames I_{t_1}, I_{t_2}, and I_{t_3}. Assume that the epipole between I_{t_i} and I_{t_j} is at pixel (x_{ij}, y_{ij}). The second-order cost of the triplet (graph edge) $(I_{t_1}, I_{t_2}, I_{t_3})$, is proportional to $\|(x_{23} - x_{12}, y_{23} - y_{12})\|$.

This second-order cost is added to the previously computed shakiness cost. The graph with the second-order smoothness term has all edge weights nonnegative and the running-time to find an optimal solution to shortest path is linear in the number of nodes and edges, i.e., $O(n\tau^2)$. In practice, with $\tau = 100$, the optimal path was found in all examples in less than 30 s. Fig. 5 shows results obtained from both first-order and second-order formulations.

IV. Panoramic Hyperlapse of a Single Video

Sampling-based hyperlapse techniques (hereinafter referred to as "sampled hyperlapse"), such as EgoSampling, or as given in [10], drop many frames for output speed and stability requirements. Instead of simply skipping the unselected frames that may contain important events, we suggest "Panoramic Hyperlapse," which uses all the frames in the video for building a panorama around selected frames.

A. Creating Panoramas

For efficiency reasons, we create panoramas only around carefully selected central frames. The panorama-generation process starts with the chosen frame as the reference frame. This is a common approach in mosaicing that reference view for the panorama should be "the one that is geometrically most central" [31, p. 73]. In order to choose the best central frame, we take a window of ω frames around each input frame and track feature points through this temporal window.

Let $f_{i,t}$ be the displacement of feature point $i \in \{1 \ldots n\}$ in frame t relative to its location in the first frame of the temporal window. The displacement of frame t relative to the first frame is defined as

$$\text{pos}_t = \frac{1}{n} \sum_{i=1}^{n} f_{i,t} \tag{4}$$

and the central frame is

$$\bar{t} = \underset{t}{\text{argmin}} \left\{ \left\| \text{pos}_t - \frac{1}{\omega} \sum_{s=1}^{\omega} \text{pos}_s \right\| \right\}. \tag{5}$$

Fig. 5. Comparative results for fast-forward from naïve uniform sampling (first row), EgoSampling using first-order formulation (second row) and using second-order formulation (third row). Note the stability in the sampled frames as seen from the tower visible far away (yellow circle). The first-order formulation leads to a more stable fast-forward output compared to naïve uniform sampling. The second-order formulation produces even better results in terms of visual stability.

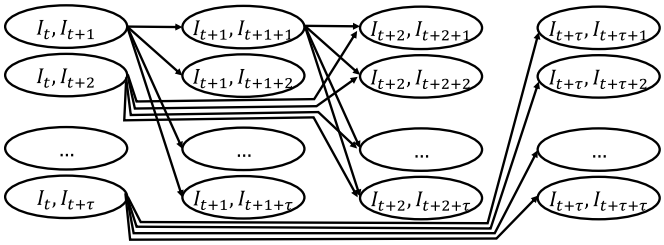

Fig. 6. Graph formulation, as described in Fig. 4, produces an output which has almost forward-looking direction. However, there may still be large changes in the epipole locations between two consecutive frame transitions, causing jitter in the output video. To overcome this, we add a second-order smoothness term based on triplets of output frames. Now the nodes correspond to pairs of frames, instead of single frames in the first-order formulation described earlier. There are edges between frame pairs (i, j) and (k, l), if $j = k$. The edge reflects the penalty for including frame triplet (i, k, l) in the output. Edges from source and sink to graph nodes (not shown in the figure) are added in the same way as in the first-order formulation to allow skipping frames from start and end.

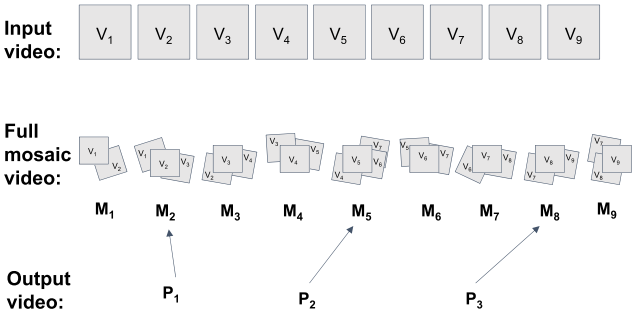

Fig. 7. Panoramic Hyperlapse creation. At the first step, for each input frame v_i, a mosaic M_i is created from frames before and after it. At the second stage, a Panoramic Hyperlapse video P_i is sampled from M_i using sampled hyperlapse methods such as [10] or EgoSampling.

Given the natural head motion alternately to the left and right, the proposed frame selection strategy prefers forward-looking frames as central frames.

After choosing the central frame, we align all the frames in the ω window with the central frame using a homography, and stitch the panorama using the "Joiners" method [32], such that central frames are on the top and peripheral frames are at the bottom. More sophisticated stitching and blending, e.g., min-cut and Poisson blending, can be used to improve the appearance of the panorama, or dealing with moving objects, etc.

B. Sampling Panoramas

After generating panoramas corresponding to different central frames, we sample a subset of panoramas for the hyperlapse video. The sampling strategy (Illustrated in fig. 7) is similar to the process described in Section III, with the nodes now corresponding to panoramas and the edge weight representing the cost of the transition from panorama p to panorama q, defined as

$$W_{p,q} = \alpha \cdot S_{p,q} + \beta \cdot V_{p,q} + \gamma \cdot \text{FOV}_p. \tag{6}$$

Here, the shakiness $S_{p,q}$ and the velocity $V_{p,q}$ are measured between the central frames of the two panoramas. FOV_p denotes the size of the panorama p, and is counted as the number of pixels painted by all frames participating in that panorama. We measure it by warping the four corners of each frame to determine the area that will be covered by the actual warped images. In the end, we run the shortest path algorithm to select the sampled panoramas as described in the previous section.

Fig. 8 shows the participation of input frames in the panoramas for one of the sample sequences. We show in gray the candidate panoramas before sampling, and the finally selected panoramas are shown in red. The span of each row shows the frames participating in each panorama.

C. Stabilization

In our experiments, we performed minimal alignment between panoramas, using only a rigid transformation between the central frames of the panoramas. When feature tracking was lost, we placed the next panorama at the center of the

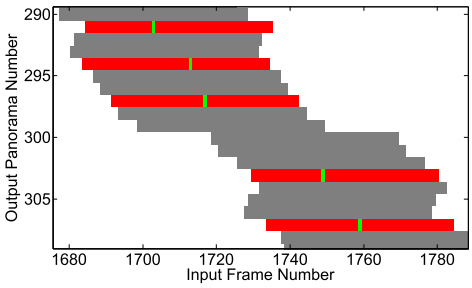

Fig. 8. Example for mapping input frames to output panoramas from sequence "Running." Rows represent generated panoramas, and columns represent input frames. Red panoramas were selected for Panoramic Hyperlapse, and gray panoramas were not used. Central frames are indicated in green.

canvas and started tracking from that frame. Any stabilization algorithm may be used as a post processing step for further fine detail stabilization. Since video stabilization reduces the FOV to be only the common area seen in all frames, starting with panoramic images mitigates this effect.

D. Cropping

Panoramas are usually created on a canvas much larger than the size of the original video, and large parts of the canvas are not covered with any of the input images. In our technique, we applied a moving crop window on the aligned panoramas. The crop window was reset whenever the stabilization was reset. In order to get smooth window movement, while containing as many pixels as possible, we find crop centers cr_i, which minimize the following energy function:

$$E = \sum \|\mathrm{cr}_i - m_i\|^2 + \lambda \sum \left\| \mathrm{cr}_i - \frac{\mathrm{cr}_{i-1} + \mathrm{cr}_{i+1}}{2} \right\|^2 \quad (7)$$

where m_i is the center of mass of the ith panorama. This can be minimized by solving the sparse set of linear equations given by the derivatives

$$\mathrm{cr}_i = \frac{\lambda(\mathrm{cr}_{i-1} + \mathrm{cr}_{i+1}) + m_i}{2\lambda + 1}. \quad (8)$$

The crop size is dependent on the camera movement and on λ. Larger λ will favor less movement of the crop window, and in order to keep it in the covered part of the canvas, it will get smaller.

E. Removing Lens Distortion

We use the method of [33] to remove lens distortion. Usually, frames are cropped after the lens distortion removal to a rectangle containing only valid pixels. However, in the case of panoramas, the cropping may be done after stitching the frames. This results in even larger FOV. An example of a cropped panoramic image after removal of lens distortion is given in Fig. 9.

We list the steps to generate Panoramic Hyperlapse in Algorithm 1.

V. Panoramic Hyperlapse of Multiple Videos

Panoramic Hyperlapse can be extended naturally to multiple input videos, as we show in this section.

Fig. 9. Same scene as in Fig. 2. The frames were warped to remove lens distortion, but were not cropped. The mosaicing was done on the uncropped frames. Notice the increased FOV compared with the panorama in Fig. 2.

Algorithm 1 Single video Panoramic Hyperlapse

Data: Single video
Result: Panoramic Hyperlapse
for *every temporal window* **do**
  find the central frame of the window;
for *every panorama candidate with center c* **do**
  for *each frame f participating in the panorama* **do**
    Calculate the transformation between f and c;
    Calculate the cost for shakiness, FOV and velocity;
Choose panoramas for the output using shortest path in graph algorithm;
Construct the panoramas;
Stabilize and crop;

A. Correspondence Across Videos

For multivideo hyperlapse, we first find corresponding frames in all other videos, for every frame in each video. We define as corresponding frame, the frame having the largest region of overlap, measured by the number of matching feature points between the frames. Any pair of frames with less than ten corresponding points is declared as nonoverlapping. We used coarse-to-fine strategy, starting from approximate candidates with skip of ten frames between each pair of matched images to find a searching interval, and then zeroing on the largest overlapping frame in that interval. It may be noted that some frames in one video may not have corresponding frame in the second video. Also note that the corresponding frame relationship is not symmetric.

We maintain temporal consistency in the matching process. For example, assuming x' and y' are the corresponding frame numbers in the second video for frame numbers x and y in the first video. If $x < y$, then we drop the match y, y' if $x' > y'$.

B. Creation of Multivideo Panorama

Once the corresponding frames have been identified, we initiate the process of selecting central frames. This process is done independently for each video, as described in Section IV with the difference that for each frame in the temporal window ω, we now collect all corresponding frames from all the input videos. For example, in an experiment with n

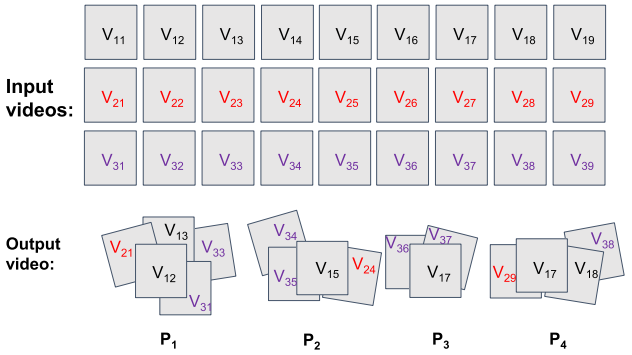

Fig. 10. Creating a multivideo Panoramic Hyperlapse. The first three rows indicate three input videos with frames labeled V_{ij}. Each frame P_i in the output panoramic video is constructed by mosaicing one or more of the input frames, which can originate from any input video.

input videos, up to $(n \cdot |\omega|)$ frames may participate in each central frame selection and mosaic generation process. The process of panorama creation is repeated for all temporal windows in all input videos. Fig. 10 outlines the relation between the Panoramic Hyperlapse and the input videos. Note that the process of choosing central frames for each camera ensures that the stabilization achieved in multivideo Panoramic Hyperlapse is similar to the one that would have been achieved if there were only a single camera. The mosaic creation can only increase the sense of stabilization because of increased FOV.

C. Sampling

After creating panoramas in each video, we perform a sampling process similar to the one described in Section IV-B, the difference being that the candidate panoramas for sampling come from all the input videos. The graph creation process is the same, with the nodes now corresponding to panoramas in all the videos. For the edge weights, apart from the costs as mentioned in the last section, we insert an additional term called *cross-video* penalty. Cross-video terms add a switching penalty, if in the output video there is a transition from panorama with central frame from one video to a panorama with central frame that comes from some other video. Note that the FOE stabilization cost in the edge weight aims to align the viewing angles of two (or three) consecutive frames in the output video and is calculated similarly, irrespective of whether the input frames originated from single or multiple videos.

The shortest path algorithm then runs on the graph created this way and chooses the panoramic frames from all input videos. We show a sample frame from one of the output videos generated by our method in Fig. 11. Algorithm 2 gives the pseudocode for our algorithm.

It may be noted that the proposed scheme samples the central frames judiciously on the basis of EgoSampling , with the quality of the chosen output mosaics being a part of the optimization. This is not equivalent to generating mosaics from individual frames and then generating the stabilized output, in the same way as in the case of single video scenario, fast-forward followed by stabilization is not equivalent to EgoSampling.

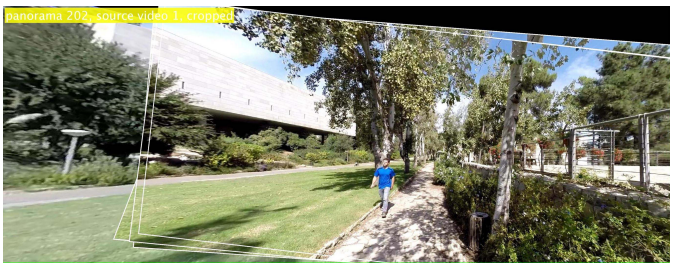

Fig. 11. Multivideo output frame. All rectangles with white borders are frames from the same video, while the left part is taken from another. Notice the enlarged FOV resulting from using frames from multiple videos.

Algorithm 2 Multivideo Panoramic Hyperlapse

Data: Multiple videos
Result: Panoramic Hyperlapse
Preprocess: temporally align videos (if necessary); calculate homographies between matching frames in different videos;
for *each video* **do**
  Find central frames and calculate cost similar to the single video case;
Calculate cross-video cost ;
Choose panoramas for the output using shortest path in graph algorithm;
for *each panorama with center c* **do**
  for *every frame f from c's video participating in the panorama* **do**
    warp f towards c;
    for *frames f' aligned with f in other videos* **do**
      warp f' towards c using chained homography f'-f-c;
  Construct the panoramas;
Stabilize and crop;

VI. Experiments

In this section, we give implementation details and show the results for EgoSampling as well as Panoramic Hyperlapse. We have used publicly available sequences [12], [34]–[36] as well as our own videos for the demonstration. The details of the sequences are given in Table I. We used a modified (faster) implementation of [4] for the LK [37] optical flow estimation. We use the code and calibration details given by [8] to correct for lens distortion in their sequences. Feature point extraction and fundamental matrix recovery is performed using VisualSFM [38], with GPU support. The rest of the implementation (FOE estimation, energy terms, and shortest path) is in MATLAB. All the experiments have been conducted on a standard desktop PC.

A. EgoSampling

We show results for EgoSampling on eight publicly available sequences. For the four sequences for which we have camera calibration information, we estimated the motion direction based on epipolar geometry. We used the FOE estimation

TABLE I

SEQUENCES USED FOR THE FAST-FORWARD ALGORITHM EVALUATION. ALL SEQUENCES WERE SHOT IN 30 fps, EXCEPT "RUNNING," WHICH IS 24 fps AND "WALKING11," WHICH IS 15 fps

Name	Src	Resolution	Num Frames
Walking1	[12]	1280x960	17249
Walking2	[39]	1920x1080	2610
Walking3	[39]	1920x1080	4292
Walking4	[39]	1920x1080	4205
Walking5	[36]	1280x720	1000
Walking6	[36]	1280x720	1000
Walking7	–	1280x960	1500
Walking8	–	1920x1080	1500
Walking9	[39]	1920x1080	2000
Walking11	[36]	1280x720	6900
Walking12	[4]	1920x1080	8001
Driving	[35]	1280x720	10200
Bike1	[12]	1280x960	10786
Bike2	[12]	1280x960	7049
Bike3	[12]	1280x960	23700
Running	[34]	1280x720	12900

method as a fallback when we could not recover the fundamental matrix. For this set of experiments, we fix the following weights: $\alpha = 1000$, $\beta = 200$ and $\gamma = 3$. We further penalize the use of estimated FOE instead of the epipole with a constant factor $c = 4$. If camera calibration is not available, we used the FOE estimation method only and changed $\alpha = 3$ and $\beta = 10$. For all the experiments, we fixed $\tau = 100$ (maximum allowed skip). We set the source and sink skip to $D_{\text{start}} = D_{\text{end}} = 120$ to allow more flexibility. We set the desired speed up factor to $10\times$ by setting K_{flow} to be ten times the average optical flow magnitude of the sequence. We show representative frames from the output for one such experiment in Fig. 5. Output videos from other experiments are given at the project's Web site: http://www.vision.huji.ac.il/egosampling/.

1) Running Times: The advantage of EgoSampling is in its simplicity, robustness, and efficiency. This makes it practical for long unstructured egocentric videos. We present the coarse running time for the major steps in our algorithm below. The time is estimated on a standard Desktop PC, based on the implementation details given above. Sparse optical flow estimation (as in [4]) takes 150 ms per frame. Estimating F-Mat (including feature detection and matching) between frame I_t and I_{t+k}, where $k \in [1, 100]$ takes 450 ms per input frame I_t. Calculating second-order costs takes 125 ms per frame. This amounts to a total of 725 ms of processing per input frame. Solving for the shortest path, which is done once per sequence, takes up to 30 s for the longest sequence in our data set ($\approx 24K$ frames). In all, running time is more than two orders of magnitude faster than [8].

2) User Study: We compare the results of EgoSampling, first- and second-order smoothness formulations, with naïve fast-forward with $10\times$ speedup, implemented by sampling the input video uniformly. For EgoSampling the speed is not directly controlled but is targeted for $10\times$ speedup by setting K_{flow} to be ten times the average optical flow magnitude of the sequence.

We conducted a user study to compare our results with the baseline methods. We sampled short clips (5–10 s each) from the output of the three methods at hand. We made sure the clips start and end at the same geographic location. We showed each of the 35 subjects several pairs of clips, before stabilization, chosen at random. We asked the subjects to state which of the clips is better in terms of stability and continuity. The majority (75%) of the subjects preferred the output of EgoSampling with first-order shakiness term over the naïve baseline. On top of that, 68% preferred the output of EgoSampling using second-order shakiness term over the output using first-order shakiness term.

To evaluate the effect of video stabilization on the EgoSampling output, we tested three commercial video stabilization tools: 1) Adobe Warp Stabilizer; 2) Deshaker[1]; and 3) YouTube's video stabilizer. We have found that YouTube's stabilizer gives the best results on challenging fast-forward videos.[2] We stabilized the output clips using YouTube's stabilizer and asked our 35 subjects to repeat process described above. Again, the subjects favored the output of EgoSampling.

3) Quantitative Evaluation: We quantify the performance of EgoSampling using the following measures. We measure the deviation of the output from the desired speedup. We found that measuring the speedup by taking the ratio between the number of input and output frames is misleading, because one of the features of EgoSampling is to take large skips when the magnitude of the optical flow is rather low. We therefore measure the effective speedup as the median frame skip.

An additional measure is the reduction in epipole jitter between consecutive output frames (or FOE if F-Matrix cannot be estimated). We differentiate the locations of the epipole (temporally). The mean magnitude of the derivative gives us the amount of jitter between consecutive frames in the output. We measure the jitter for our method as well for naïve $10\times$ uniform sampling and calculate the percentage improvement in jitter over competition.

Table II shows the quantitative results for frame skip and epipole smoothness. There is a huge improvement in jitter by our algorithm. We note that the standard method to quantify video stabilization algorithms is to measure crop and distortion ratios. However, since we jointly model fast-forward and stabilization such measures are not applicable. The other method could have been to postprocess the output video with a standard video stabilization algorithm and measure these factors. Better measures might indicate better input to stabilization or better output from preceding sampling. However, most stabilization algorithms rely on trajectories and fail on resampled video with large view difference. The only successful algorithm was YouTube's stabilizer, but it did not give us these measures.

4) Limitations: One notable difference between EgoSampling and traditional fast-forward methods is that the number of output frames is not fixed. To adjust the effective speedup, the user can tune the velocity term by setting different values to K_{flow}. It should be noted, however, that not all speedup factors are possible without compromising the stability of the output.

[1] http://www.guthspot.se/video/deshaker.htm

[2] We attribute this to the fact that YouTube's stabilizer does not depend upon long feature trajectories, which are scarce in sub-sampled video as ours.

TABLE II

FAST-FORWARD RESULTS WITH DESIRED SPEEDUP OF FACTOR 10 USING SECOND-ORDER SMOOTHNESS. WE EVALUATE THE IMPROVEMENT AS A DEGREE OF EPIPOLE SMOOTHNESS IN THE OUTPUT VIDEO (COLUMN 5). THE PROPOSED METHOD GIVES A HUGE IMPROVEMENT OVER NAÏVE FAST-FORWARD IN ALL BUT ONE TEST SEQUENCE (SEE FIG. 12 FOR THE FAILURE CASE). NOTE THAT THE ACTUAL SKIP (COLUMN 4) CAN DIFFER A LOT FROM THE TARGET IN THE PROPOSED ALGORITHM

Name	Input Frames	Output Frames	Median Skip	Improvement over Naïve $10\times$
Walking1	17249	931	17	283%
Walking11	6900	284	13	88%
Walking12	8001	956	4	56%
Driving	10200	188	48	−7%
Bike1	10786	378	13	235%
Bike2	7049	343	14	126%
Bike3	23700	1255	12	66%
Running	12900	1251	8	200%

Fig. 12. Failure case for the proposed method showing two sample frames from an input sequence. The frame-to-frame optical flow is mostly zero because of distant view and (relatively) static vehicle interior. However, since the driver shakes his head every few seconds, the average optical flow magnitude is high. The velocity term causes us to skip many frames until the desired K_{flow} is met. Restricting the maximum frame skip by setting τ to a small value leads to arbitrary frames being chosen looking sideways, causing shake in the output video.

For example, consider a camera that toggles between looking straight and looking to the left every ten frames. Clearly, any speedup factor that is not a multiple of 10 will introduce shake to the output. The algorithm chooses an optimal speedup factor, which balances between the desired speedup and what can be achieved in practice on the specific input. Sequence "Driving" (Fig. 12) presents an interesting failure case.

Another limitation of EgoSampling is to handle long periods in which the camera wearer is static; hence, the camera is not translating. In these cases, both the fundamental matrix and the FOE estimations can become unstable, leading to wrong cost assignments (low penalty instead of high) to graph edges. The appearance and velocity terms are more robust and help reduce the number of outlier (shaky) frames in the output.

B. Panoramic Hyperlapse

In this section, we show experiments to evaluate Panoramic Hyperlapse for single as well as multiple input videos. To evaluate the multiple videos case (Section V), we have used two types of video sets. The first type is the videos sharing similar camera path on different times. We obtained the data set of [39] suitable for this purpose. The second type is the videos shot simultaneously by number of people wearing cameras and walking together. We scanned the data set of [36] and found videos corresponding to a few minutes of a group walking together toward an amusement park. In addition, we choreographed two videos of this type by ourselves. We will release these videos upon paper acceptance. The videos were shot using a GoPro3+ camera. Table I gives the resolution, FPS, length, and source of the videos used in our experiments.

TABLE III

COMPARING FOV: WE MEASURE CROPPING OF OUTPUT FRAME OUTPUT BY VARIOUS METHODS. THE PERCENTAGES INDICATE THE AVERAGE AREA OF THE CROPPED IMAGE FROM THE ORIGINAL INPUT IMAGE, MEASURED ON TEN RANDOMLY SAMPLED OUTPUT FRAMES FROM EACH SEQUENCE. THE SAME FRAMES WERE USED FOR ALL THE FIVE METHODS. THE NAÏVE, EGOSAMPLING (ES), AND PANORAMIC HYPERLAPSE(PH) OUTPUTS WERE STABILIZED USING YOUTUBE STABILIZER [16]. REAL-TIME HYPERLAPSE [10] OUTPUT WAS CREATED USING THE DESKTOP VERSION OF THE HYPERLAPSE PRO APP. THE OUTPUT OF HYPERLAPSE [8] IS ONLY AVAILABLE FOR THEIR DATASET. WE OBSERVE IMPROVEMENTS IN ALL THE EXAMPLES EXCEPT "WALKING2," IN WHICH THE CAMERA IS VERY STEADY

Name	Exp. No.	Naive	[10]	[8]	ES	PH
Bike3	S1	45%	32%	65%	33%	99%
Walking1	S2	52%	68%	68%	40%	95%
Walking2	S3	67%	N/A	N/A	43%	66%
Walking3	S4	71%	N/A	N/A	54%	102%
Walking4	S5	68%	N/A	N/A	44%	109%
Running	S6	50%	75%	N/A	43%	101%

C. Implementation Details

We have implemented Panoramic Hyperlapse in MATLAB and run it on a single PC with no GPU support. For tracking, we use MATLAB's built in SURF feature points detector and tracker. We found the homography between frames using RANSAC. This is a time-consuming step, since it requires calculating transformations from every frame, which is a candidate for a panorama center, to every other frame in the temporal window around it (typically $\omega = 50$). In addition, we find homographies to other frames that may serve as other panorama centers (before/after the current frame), in order to calculate the shakiness cost of a transition between them. We avoid creating the actual panoramas after the sampling step to reduce runtime. However, we still have to calculate the panorama's FOV as it is part of our cost function. We resolved to create a mask of the panorama, which is faster than creating the panorama itself. The parameters of the cost function in (6) were set to $\alpha = 1 \cdot 10^7$, $\beta = 5 \cdot 10^6$, $\gamma = 1$ and $\lambda = 15$ for the crop window smoothness. Our $cross - video$ term was multiplied by the constant 2. We used those parameters both for the single and multivideo scenarios. The input and output videos are given at the project's Web site.

D. Runtime

The following runtimes were measured with the setup described in the previous section on a 640×480 resolution video, processing a single input video. Finding the central images and calculating the shakiness cost takes 200 ms per

(a) (b) (c) (d)

Fig. 13. Comparing FOV of hyperlapse frames, corresponding to approximately same input frames from sequence "Bike1." For best viewing, zoom to 800%. (a) Original frame and output of EgoSampling. (b) Output of [8]. Cropping and rendering errors are clearly visible. (c) Output of [10] suffering from strong cropping. (d) Output of our method, having the largest FOV.

TABLE IV

EVALUATION OF THE CONTRIBUTION OF MULTIPLE VIDEOS TO THE FOV. THE CROP SIZE WAS MEASURED TWICE: ONCE WITH THE SINGLE VIDEO ALGORITHM, WITH THE VIDEO IN THE FIRST COLUMN AS INPUT, AND ONCE WITH THE MULTIVIDEO ALGORITHM

Name	Exp. No.	Ours Single	Number of Videos	Ours Multi
Walking2	M1	67%	4	140%
Walking5	M2	90%	2	98%
Walking7	M3	107%	2	118%

Fig. 14. Comparing FOV of panoramas generated from single-video (left) and multivideo (right) Panoramic Hyperlapse. Multivideo Panoramic Hyperlapse is able to successfully collate content from different videos for enhanced FOV.

frame, each. Calculating the FOV term takes 100 ms per frame on average. Finding the shortest path takes a few seconds for the entire sequence. Sampling and panorama creation takes 3 s per panorama, and the total time depends on the speed up from the original video, i.e., the ratio between number of panoramas and length of the input. For a typical 10× speed, this amounts to 300 ms. The total runtime is 1.5–2 s per frame with an unoptimized MATLAB implementation. In the multi-input video cases, the runtime grows linearly with the number of input sequences.

E. Evaluation

The main contribution of Panoramic Hyperlapse to the hyperlapse community is the increased FOV over existing methods. To evaluate it we measure the output resolution (i.e., the crop size) of the baseline hyperlapse methods on the same sequence. The crop is a side effect of stabilization: without crop, stabilization introduces "empty" pixels to the FOV. The cropping ensures to limit the output frame to the intersection of several FOVs, which can be substantially smaller than the FOV of each frame depending on the shakiness of the video.

The crop size is not constant throughout the whole output video, and hence it should be compared individually between output frames. Because of the frame sampling, an output frame with one method is not guaranteed to appear in the output of another method. Therefore, we randomly sampled frames for each sequence until we had ten frames that appear in all output methods. An example is shown in fig. 13. For a panorama, we considered its central frame. We note that the output of [8] is rendered from several input frames, and does not have any dominant frame. We therefore tried to pick frames corresponding to the same geographical location in the other sequences. Our results are summarized in Tables III and IV. It is clear that in terms of FOV we outperform most of the baseline methods on most of the sequences. The contribution of multiple videos to the FOV is illustrated in Fig. 14.

The naïve fast-forward, EgoSampling, and Panoramic Hyperlapse outputs were stabilized using YouTube stabilizer. Real-time Hyperlapse [10] output was created using the desktop version of the Hyperlapse Pro. app. The output of Hyperlapse [8] is only available for their data set.

a) Failure case: On sequence Walking2 the naïve results get the same crop size as our method (see Table III). We attribute this to the exceptionally steady forward motion of the camera, almost as if it is not mounted on the photographer head while walking. Obviously, without the shake Panoramic Hyperlapse cannot extend the FOV significantly.

F. Panoramic Hyperlapse From Multiple Videos

Fig. 14 shows a sample frame from the output generated by our algorithm using sequences "Walking 7" and "Walking 8." Comparison with panoramic hyperlapse generated from single video clearly shows that our method is able to assemble content from frames from multiple videos for enhanced FOV. We quantify the improvement in FOV using the crop ratio of the output video on various publicly and self-shot test sequences. Table IV gives the detailed comparison.

Multivideo Panoramic Hyperlapse can also be used to summarize contents from multiple videos. Fig. 15 shows an example panorama generated from sequences "Walking 5" and "Walking 6" from the data set released by [36]. While a lady is visible in one video and a child in another, both persons appear in the output frame at the same time.

When using multiple videos, each panorama in the Panoramic Hyperlapse is generated from many frames, as much as 150 frames if we use three videos and a temporal

Fig. 15. Panoramic Hyperlapse: Left and middle: two input spatially neighboring frames from different videos. Right: output frame generated by Panoramic Hyperlapse. Blue lines: frames coming from the same video as the middle frame (Walking6). White lines: frames from the other video (Walking5). Notice that while a lady can be observed in one and a child in another, both are visible in the output frames. The stitching errors are due to misalignment of the frames. We did not have the camera information for these sequences and could not perform lens distortion correction

window of 50 frames. With this wealth of frames, we can filter out some frames with undesired properties. For example, if privacy is a concern, we can remove from the panorama all frames having a recognizable face or a readable license plate.

VII. Conclusion

We propose a novel frame sampling technique to produce stable fast-forward egocentric videos. Instead of the demanding task of 3D reconstruction and rendering used by the best existing methods, we rely on simple computation of the epipole or the FOE. The proposed framework is very efficient, which makes it practical for long egocentric videos. Because of its reliance on simple optical flow, the method can potentially handle difficult egocentric videos, where methods requiring 3D reconstruction may not be reliable.

We also present Panoramic Hyperlapse, a method to create hyperlapse videos that have a large FOV. While in EgoSampling we drop unselected (outlier) frames, in Panoramic Hyperlapse, we use them to increase the FOV in the output video. In addition, Panoramic Hyperlapse naturally supports the processing of multiple videos together, extending the output FOV even further, as well as allowing the consumption of multiple such videos in less time. The large number of frames used for each panorama also allows the removal of undesired objects from the output.

References

[1] Z. Lu and K. Grauman, "Story-driven summarization for egocentric video," in *Proc. CVPR*, Jun. 2013, pp. 2714–2721.
[2] Y. J. Lee, J. Ghosh, and K. Grauman, "Discovering important people and objects for egocentric video summarization," in *Proc. CVPR*, Jun. 2012, pp. 1346–1353.
[3] J. Xu, L. Mukherjee, Y. Li, J. Warner, J. M. Rehg, and V. Singh, "Gaze-enabled egocentric video summarization via constrained submodular maximization," in *Proc. CVPR*, Jun. 2015, pp. 2235–2244.
[4] Y. Poleg, C. Arora, and S. Peleg, "Temporal segmentation of egocentric videos," in *Proc. CVPR*, Jun. 2014, pp. 2537–2544.
[5] Y. Poleg, A. Ephrat, S. Peleg, and C. Arora, "Compact CNN for indexing egocentric videos," in *Proc. WACV*, Mar. 2016, pp. 1–9.
[6] K. M. Kitani, T. Okabe, Y. Sato, and A. Sugimoto, "Fast unsupervised ego-action learning for first-person sports videos," in *Proc. CVPR*, Jun. 2011, pp. 3241–3248.
[7] M. S. Ryoo, B. Rothrock, and L. Matthies, "Pooled motion features for first-person videos," in *Proc. CVPR*, Jun. 2015, pp. 896–904.
[8] J. Kopf, M. Cohen, and R. Szeliski, "First-person hyper-lapse videos," in *Proc. SIGGRAPH*, Jul. 2014, vol. 33. no. 4, Art. no. 78.
[9] N. Petrovic, N. Jojic, and T. S. Huang, "Adaptive video fast forward," *Multimedia Tools Appl.*, vol. 26, no. 3, pp. 327–344, 2005.
[10] N. Joshi, W. Kienzle, M. Toelle, M. Uyttendaele, and M. F. Cohen, "Real-time hyperlapse creation via optimal frame selection," in *Proc. SIGGRAPH*, 2015, vol. 34. no. 4, Art. no. 63.
[11] Y. Poleg, T. Halperin, C. Arora, and S. Peleg, " EgoSampling: Fast-forward and stereo for egocentric videos," in *Proc. CVPR*, Jun. 2015, pp. 4768–4776.
[12] J. Kopf, M. Cohen, and R. Szeliski, *First-Person Hyperlapse Videos—Supplemental Material*. [Online]. Available: http://research.microsoft.com/en-us/um/redmond/projects/hyperlapse/supplementary/index.html
[13] B. Xiong and K. Grauman, "Detecting snap points in egocentric video with a Web photo prior," in *Proc. ECCV*, Sep. 2014, pp. 282–298.
[14] F. Liu, M. Gleicher, H. Jin, and A. Agarwala, "Content-preserving warps for 3D video stabilization," in *Proc. SIGGRAPH*, Aug. 2009, Art. no. 44.
[15] S. Liu, Y. Wang, L. Yuan, J. Bu, P. Tan, and J. Sun, "Video stabilization with a depth camera," in *Proc. CVPR*, Jun. 2012, pp. 89–95.
[16] M. Grundmann, V. Kwatra, and I. Essa, "Auto-directed video stabilization with robust L1 optimal camera paths," in *Proc. CVPR*, Jun. 2011, pp. 225–232.
[17] F. Liu, M. Gleicher, J. Wang, H. Jin, and A. Agarwala, "Subspace video stabilization," in *Proc. SIGGRAPH*, Jan. 2011, Art. no. 4.
[18] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," in *Proc. SIGGRAPH*, Jul. 2013, Art. no. 78.
[19] S. Liu, L. Yuan, P. Tan, and J. Sun, "SteadyFlow: Spatially smooth optical flow for video stabilization," in *Proc. CVPR*, Jun. 2014, pp. 4209–4216.
[20] A. Goldstein and R. Fattal, "Video stabilization using epipolar geometry," in *Proc. SIGGRAPH*, Aug. 2012, Art. no. 126.
[21] Y. Matsushita, E. Ofek, W. Ge, X. Tang, and H.-Y. Shum, "Full-frame video stabilization with motion inpainting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, pp. 1150–1163, Jul. 2006.
[22] W. Jiang and J. Gu, "Video stitching with spatial-temporal content-preserving warping," in *Proc. CVPR Workshops*, Jun. 2015, pp. 42–48.
[23] Y. Hoshen, G. Ben-Artzi, and S. Peleg, "Wisdom of the crowd in egocentric video curation," in *Proc. CVPR Workshops*, Jun. 2014, pp. 587–593.
[24] I. Arev, H. S. Park, Y. Sheikh, J. Hodgins, and A. Shamir, "Automatic editing of footage from multiple social cameras," *ACM Trans. Graph.*, vol. 33, no. 4, Jul. 2014.
[25] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2003.
[26] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. ECCV*, Sep. 2014, pp. 834–849.
[27] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. ICRA*, May 2014, pp. 15–22.
[28] D. Sazbon, H. Rotstein, and E. Rivlin, "Finding the focus of expansion and estimating range using optical flow images and a matched filter," *Mach. Vis. Appl.*, vol. 15, no. 4, pp. 229–236, Oct. 2004.
[29] O. Pele and M. Werman, "Fast and robust earth mover's distances," in *Proc. ICCV*, Sep. 2009, pp. 460–467.
[30] E. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959.
[31] R. Szeliski, "Image alignment and stitching: A tutorial," *Found. Trends Comput. Graph. Vis.*, vol. 2, no. 1, pp. 1–104, Jan. 2006.
[32] L. Zelnik-Manor and P. Perona, "Automating joiners," in *Proc. 5th Int. Symp. Non-Photorealistic Animation Rendering*, Aug. 2007, pp. 121–131.
[33] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 5695–5701.
[34] *Ayala Triangle Run With GoPro Hero 3+ Black Edition*, accessed on Sep. 12, 2014. [Online]. Available: https://www.youtube.com/watch?v=WbWnWojOtIs

[35] *GoPro Trucking!—Yukon to Alaska 1080p*, accessed on Aug. 10, 2014. [Online]. Available: https://www.youtube.com/watch?v=3dOrN6-V7V0
[36] A. Fathi, J. K. Hodgins, and J. M. Rehg, "Social interactions: A first-person perspective," in *Proc. CVPR*, Jun. 2012, pp. 1226–1233.
[37] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. IJCAI*, vol. 2. Apr. 1981, pp. 121–130.
[38] C. Wu. *Visualsfm: A Visual Structure From Motion System*, accessed on Aug. 3, 2013. [Online]. Available: http://ccwu.me/vsfm/
[39] Y. Hoshen and S. Peleg, "An egocentric look at video photographer identity," in *Proc. CVPR*, Jun. 2016, pp. 4284–4292.

Tavi Halperin received the B.Sc. degree in mathematics and computer science and the M.Sc. degree in computer science from Hebrew University of Jerusalem, Jerusalem, Israel, in 2012 and 2015, respectively, where he is currently working toward the Ph.D. degree in computer vision with the Department of Computer Science.

His research interests include egocentric videos and computational geometry.

Yair Poleg received the Ph.D. degree from Hebrew University of Jerusalem, Jerusalem, Israel, in 2015.

His Ph.D. research involved the analysis of egocentric video. He devotes most of his time to his startup company, Ayyeka, Jerusalem, where he is involved in developing industrial Internet-of-Things solutions.

Chetan Arora received the Ph.D. degree in computer science from Indraprastha Institute of Information Technology (IIIT) Delhi, New Delhi, India, in 2012.

Since 2014, he has been an Assistant Professor with the Computer Science Department, IIIT Delhi. He has spent over 10 years in industry, where he co-founded three startups, all working on computer vision products coming out of the latest research ideas. He has authored over 20 papers in top computer vision journals and peer-reviewed conferences. He has been actively involved in the area of computer vision for persons with disabilities and has organized multiple workshops to promote the same.

Dr. Arora has served as an Area Chair at the ICVGIP 2016 and the Program Chair for the Workshop on Assistive Vision held with the ACCV 2016.

Shmuel Peleg received the Ph.D. degree in computer science from University of Maryland, College Park, MD, USA in 1979.

In 1981, he became a Faculty Member with Hebrew University of Jerusalem, Jerusalem, Israel, where he is currently a Professor of Computer Science. He has authored over 150 technical papers in computer vision and image processing, and holds 22 U.S. patents. His technologies provided the technical foundations to several startup companies.

Dr. Peleg served as an Editor and a Committee Member of numerous international journals and conferences, and was the General Co-Chair of international conferences including the ICPR 1994, the CVPR 2011, and the ICCP 2013. He is a General Co-Chair of the CVPR 2018.