https://doi.org/10.2352/ISSN.2470-1173.2017.15.DPMI-063

Stabilized High-Speed Video from Camera Arrays

Maha El Choubassi, Oscar Nestares; Intel Corporation; Santa Clara, California/USA

Abstract

We present an algorithm to get high-speed video using camera array with good perceptual quality in realistic scenes that may have clutter and complex background. We synchronize the cameras such that each captures an image at a different time offset. The algorithm processes the jittery interleaved frames and produces a stabilized video. Our method consists of: synthesis of views from a virtual camera to correct for differences in cameras perspectives, and video compositing to remove remaining artifacts especially around disocclusions. More explicitly, we process the optical flow of the raw video to estimate, for each raw frame, the disparity to the target virtual frame. We input these disparities to content-aware warping to synthesize the virtual views, significantly alleviating the jitter. Yet, while the warping fills the disocclusion holes, the filling may not be coherent temporally, leading to small jitter still visible in static/slow regions around large disocclusions. However, these regions don't benefit from high rate in high-speed video. Therefore, we extract low frame rate regions from only one camera and video composite them with the remaining highly moving regions taken by all cameras. The final video is smooth and efficiently has high frame rate in high motion regions.

Introduction

Slow motion videos are pervasive. To get these videos, we first acquire a high-speed video, i.e., a video captured at a very high frame rate, larger than the typical 30-60 frames per sec (fps). When played at slower rate, it becomes a 'slow mo' video. Such videos are popular and useful because they enable us to see details we usually miss in reality. The applications are wide including sports, science, photography, and testing and manufacturing, see Figure 1.

These videos can be captured by a specialized camera. Phantom camera [4] is the high end high-speed camera from Vision Research that captures at 25,000 fps and can go up to 1 million fps by trading off resolution. However, such cameras are expensive, and can weigh up to 1.4Kg. On the other end of the spectrum, some smartphones nowadays, such as iPhone and Samsung, have high-speed mode, making slow mo videos available to a wider population. For example, iPhone 6 can capture video at 240 fps with resolution 720 p, lower than the 1080p resolution of the regular video 60 fps [5].

However, there is special interest for high-speed video from camera arrays for the following reasons:

1. Such systems are popular, see Figure 2, and have their own merit for bringing a whole class of new applications due to the multiple cameras: computing depth, refocusing after capture, and creating panoramas.
2. In case of low frame rate individual cameras, camera arrays can reach high-speed rates, just like the specialized high-speed cameras. For already high frame rate cameras, we can multiply this rate and enable more applications.
3. In high-speed video from camera arrays, we preserve the spatial resolution, unlike the trade-off between resolution and frame rate in specialized cameras.

Given a camera array of N cameras, C_1, C_2, ..., and C_N, assume without loss of generality that each camera can capture video at F fps. Synchronizing C_1, C_2, ..., and C_N to start capturing at interleaved times t_1, t_2, ..., and t_N, we get a video that samples the time dimension N times finer at $N \times F$ fps, see Figure 3a and b. Since the cameras have different perspectives, the resulting frames are misaligned and the video is jittery. Hence, there is need for an algorithm to process these frames so that the final high-speed video is visually acceptable.

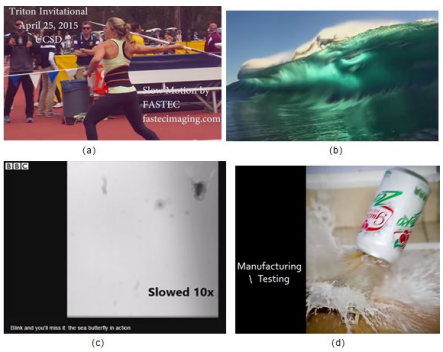

Figure 1. *(a) Slow mo in sports on YouTube (b) Slow mo footage on YouTube (c) 'Blink and you'll miss it': Highspeed video of sea snail, 10x slower [6] (d) Slow mo video in soda can quality testing courtsey of YouTube.*

Figure 2. *(a) Dell Venue 8 7000 [7] (b) Pelican camera array [8] (c) Light 16 camera-array [9] (d) Huawei P9 phone with dual cameras [10](e) HTC One phone with dual cameras [11] (f) LG G5 phone with dual cameras [12].*

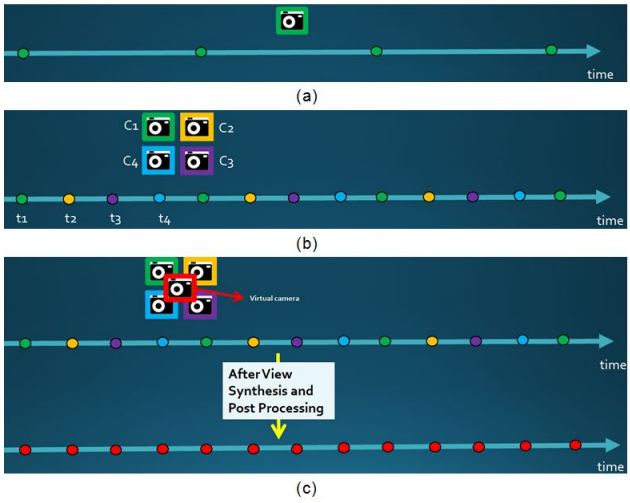

Figure 3. *(a) Individual camera capturing video at F fps (b) camera array of N cameras (e.g. N = 4). Jittery raw high-speed video. Effective frame rate=NxF fps. (c) After view synthesis and postprocessing, we have our smooth high-speed video at NxF fps.*

Previous work and Problem Statement

Our objective is to enable high-speed video on camera arrays by transforming the jittery raw interleaved video into visually plausible video. Other researchers have previously proposed using camera arrays to get high-speed video [1, 2, 3].

FourSee Camera Array [3]: the prototype has 4 cameras pointing inwards and 1 lens. The cameras capture video at interleaving times, but are physically arranged so that they all capture the same perspective. While there is no need for an algorithm to align the frames, this configuration is specific for high-speed video only, forfeiting desirable camera arrays capabilities such as depth. Contrarily, most camera arrays purposefully have different perspective for every camera to get these capabilities.

The approach in [1] is an early proof of concept stabilizing only a single depth plane using homographies and the artifacts are quite visible. Our approach is not simply based on homography and therefore has much more stabilization power.

The approach in [2] is a major improvement over [1], where the authors design a new flow algorithm and warp the images to virtual camera position. However, this solution does not address the occlusion and disocclusion areas, which are challenging and cause significant artifacts. The example 'ball.wmv' in [2] has simple plain background and does not have many objects. But for richer scenes with closer objects and big gaps in depth, the artifacts due to occlusions/disocclusions would be intolerable. In our approach, we carefully work on eliminating the artifacts as in **Compositing** section. Moreover, as mentioned in [1], errors in the flow will cause distortions, as clearly seen around the head of the player for example in 'ball.wmv'. We use confidence to put less weight on point correspondences that we don't trust, hence reducing the artifacts due to flow errors as in **Content-aware warping** section.

In this paper, we combine flow estimation, view synthesis, and video compositing to generate a high-speed video from camera arrays in realistic scenes with complex environment, where we stabilize all the depth planes and carefully remove disocclusion artifacts, without sacrificing depth capabilities or reducing spatial resolution.

Algorithm Description

Our algorithm mainly has 2 components as in Figure 4: view synthesis to align the perspective and video compositing to remove artifacts.

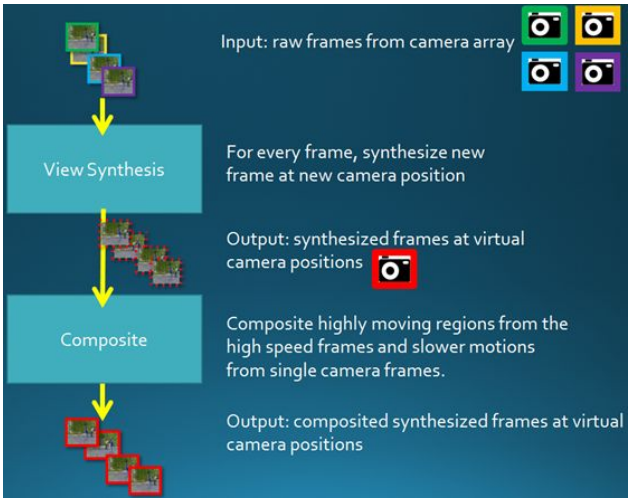

Figure 4. *High-speed algorithm. (1) view synthesis: content aware warping. (2) Posprocessing: compositing to generate the final high-speed video.*

View synthesis

We choose the virtual camera position at the array center, as close as possible to all the cameras. We use content-aware warping as a fast method to synthesize views from the virtual perspective. The result video will be much better aligned than the raw interleaved frames, but some artifacts will remain and will be addressed in later step.

Content-aware warping

As in Figure 5, content-aware warping [13, 14, 21] renders a perceptually pleasant view from the target virtual camera by first curing the noisy input target disparity or correspondences. Using a dense disparity map or sparse point correspondences is a design knob that trades off quality vs complexity. The output disparities, d_1, d_2, ..., d_V, don't need to cover the whole support of the image and can cover a sample of the original dense image pixels positions such as a regular grid of vertices. Solving for this sampled disparity is another parameter for the system designer to control the size of the problem, and hence balance computational complexity and memory requirements together with perceptual quality requirements. For this purpose, we formulate a cost function in terms of the desired disparity and minimize it as in the equation below. The solution to this optimization problem is the sought disparity, which

can be readily used to generate the synthetic view.

$$E = E_d + \alpha E_s, \quad (1)$$

where the data and the distortion terms are defined as

$$E_d(d_1 d_2, \ldots, d_V) = \sum_{polygon\ p} \sum_{n} c_n (\mathbf{w_n}^t \mathbf{d^p} - o_n)^2,$$

$$E_s(d_1 d_2, \ldots, d_V) = \sum_{polygon\ p} s_p f(\mathbf{d^p}),$$

The original disparities and their confidence are o_1, o_2, ..., o_M and c_1, c_2, ..., c_M. The confidence gives better quality results as less trustworthy samples due to flow errors contribute less to the solution. $\mathbf{d^p}$ is the vector of unknown disparity values at vertices of polygon p of the grid. o_n is the disparity at a point in p and $\mathbf{w_n}$ is the vector with interpolation weights for each vertex of p for this point, e.g., bilinear interpolation weights. $f(\cdot)$ is the perceptual distortion and s_p is the saliency weight (for less distortion in salient regions). We use sum of squared differences between disparity pairs at vertices of p for $f(\cdot)$ as in [14] and variance for s_p as in [13].

The data term constrains the desired disparities d_1, d_2, ..., d_V to be close to the input disparity. At the same time, the distortion term enforces the perceptual constraints on d_1, d_2, ..., d_V. We solve (1) for d_1, d_2, ..., d_V by converting it into a linear problem and efficiently using a sparse solver (linear PCG with Jacobi preconditioner).

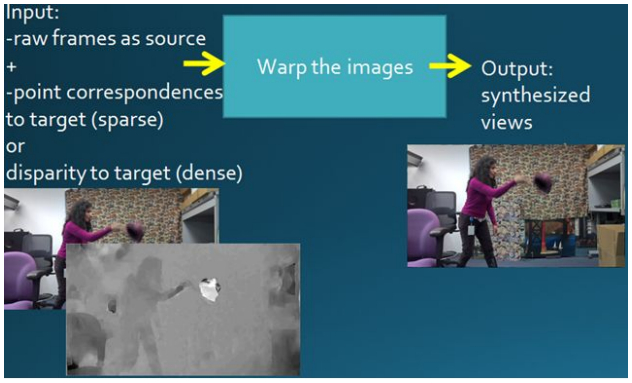

Figure 5. *Content aware warping. Input: image + sparse or dense disparity to target. Output: synthesized view from target virtual camera.*

Many variations of the equation (1) are possible depending on the application itself. For example, the authors in [13, 14, 21] use variants of (1), respectively for video stabilization, synthesizing views for autostereoscopic displays, and generating cinematographs.

Getting depth: cascade tracking flow

As in Figure 5, for every frame captured by C_1, C_2, ..., and C_N, content-aware warping needs as input both RGB and also some depth information, in the form of dense target disparity map or sparse point correspondences to the virtual camera. This depth information is represented by the original disparities o_1, o_2, ..., o_M and their confidence in equation (1). We could capture RGBD data by grouping the N cameras into N/2 stereo pairs. However, in this way, the resulting high-speed video will have rate $N/2 \times F$ fps, i.e., half the possible speedup in frame rate. Instead of sacrificing half the cameras for RGBD capture, our method uses all the N cameras for high-speed capture, estimates the target disparity map or point correspondences as explained below, and obtains the full speedup of factor N.

As in Figure 6, we we track across the frames from C_1, C_2, ..., and C_N and compute the bidirectional flow and estimate the confidence with cross check. Depending on desired balance of quality vs computational complexity, the tracking result can be either sparse trajectories, for example using Kanade-Lucas-Tomasi features tracker [15, 16], or dense flow maps. We assume dense flow without loss of generality. The same steps are applicable to sparse tracking. We next cascade the flows and use the interpolation

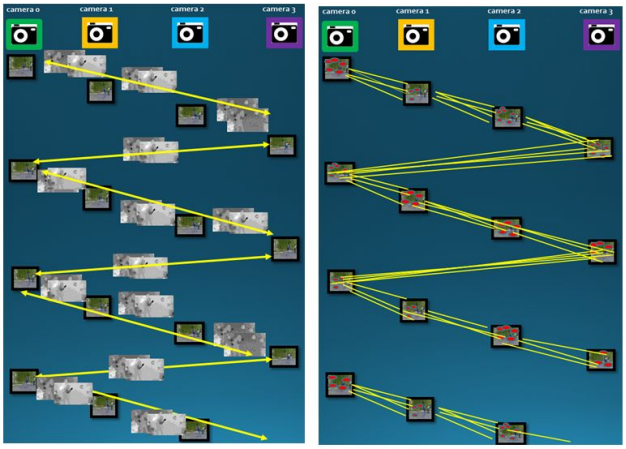

Figure 6. *Track across frames from C_1, C_2, ..., and C_N. (a) dense forward and backward flow. (b)sparse track trajectories.*

to estimate the disparity from every frame to its target 'to-be-synthesized' frame from the virtual camera.

Detailed example. Assume we have $N = 4$ cameras on a 2x2 grid. Because we are using all the cameras for high-speed capture, the cameras are capturing at interleaving times. Hence at any capturing instance, only one of the cameras captures a frame and all the remaining cameras are inactive. For example, in Figure 7, at time t, only camera 3 (purple) captures and $F3$ is the frame captured by it. None of the other cameras captures at this time, including camera 0 (green). Still, we need to estimate the disparity map $F3 \rightarrow \hat{F}$ from $F3$, the actual frame of camera 3 frame (purple) at time t, to $\hat{F}$, the hypothetical frame that would have been captured by camera 0 at same time t.

For this purpose, in Figure 8a, we cascade the backward flows $F4 \rightarrow F3 \rightarrow F2 \rightarrow F1 \rightarrow F0$ to estimate the flow $F4 \rightarrow F0$. The cascading is done by accumulating the flow and forward mapping ,e.g., using bilinear interpolation. Next, we scale this flow by $0.25 = 1/4$ ($N = 4$ cameras) in this case as in Figure 8b, to estimate the flow from $F4 \rightarrow \hat{F}$. Generally, the scalar is c/N, where c is

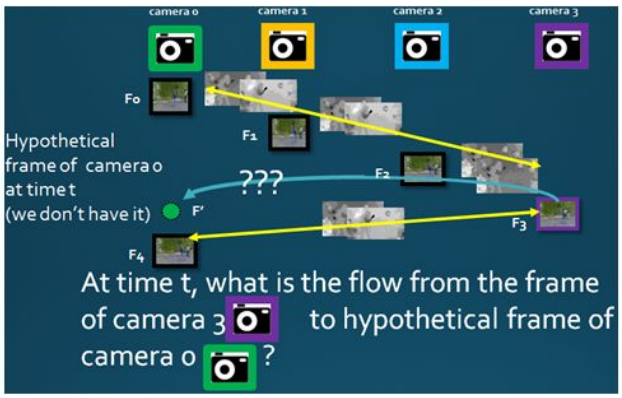

Figure 7. *Need disparity $F3 \rightarrow \grave{F}$ from camera 3 frame at time t to the uncaptured hypothetical frame of camera 0 at time t.*

an integer $1 \leq c \leq N-1$, depending on the target time t. The scaling is justified since $F0$ and $F4$ are adjacent frames captured by camera 0 and we assume that the motion between frames F0 and F4 can be approximated as linear. Equivalently, we are approximating each trajectory in a camera C_j stream as piecewise linear. To get our estimated flow $F3 \rightarrow \grave{F}$, the last step is in Figure 8c. We cascade the forward flow $F3 \rightarrow F4$ and the estimated backward flow $F4 \rightarrow \grave{F}$. The result is the estimated disparity map $F3 \rightarrow \grave{F}$ in Figure 8.[1]

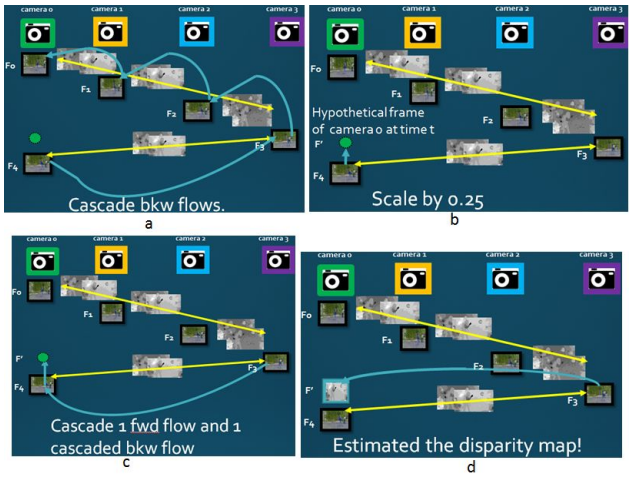

Figure 8. *(a) Cascade the backward flows from $F4 \rightarrow F3 \rightarrow F2 \rightarrow F1 \rightarrow F0$. The result is estimated flow $F4 \rightarrow F0$ (b) Scale the resulting flow of (a) by 0.25. The result is the estimated flow $F4 \rightarrow \grave{F}$, from $F4$ to hypothetical frame of camera 0 at time t. (c) Cascade the forward flow $F3 \rightarrow F4$ and the resulting flow of (b) $F4 \rightarrow \grave{F}$. (d) The result flow of (c) is $F3 \rightarrow \grave{F}$.*

Applying the same process, we estimate the disparity maps from frame $F3$, the frame captured at time t by camera 3, to the hypothetical frames that would have been captured by all the remaining N-1 cameras, in this case 3 cameras: camera 0, camera 1, and camera 2. At this point, we can estimate the desired target disparity map from frame $F3$ to the virtual camera position as a weighted sum of all these estimated flows (the weight is $1/N$) as in Figure 9. Using this weight, the target virtual camera is at the center of the camera array. Now, our content-aware warping has what it needs to generate the synthesized views from the virtual camera position.

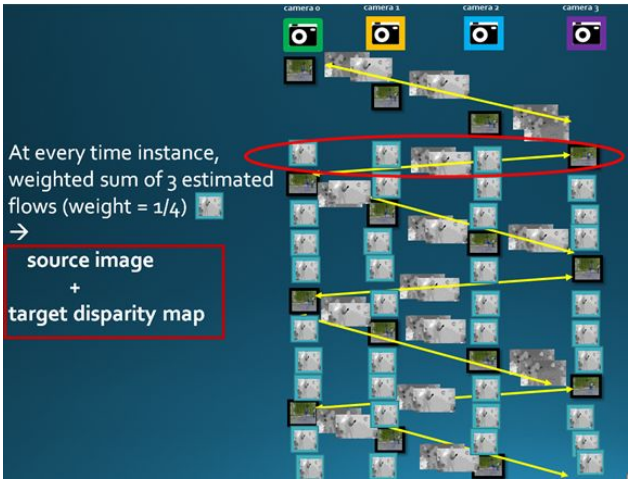

Figure 9. *Get the estimate of the target disparity map: weighted sum (weight = 1/N) of estimated flows from Figure 8.*

Compositing

Generally, view synthesis algorithms [13, 14, 17, 18] need to carefully address regions around depth boundaries of close objects with large gaps in depth. These regions are prone to artifacts due to large disocclusion when changing the perspective in the new synthesized view. For example, some view synthesis methods may generate holes in the synthesized frames in these areas, as in [19], and a post processing step of hole filling/inpainting [20] is needed. However, content-aware warping does not generate holes as it implicitly fills the disocclusion holes by stretching the texture around them. Looking at each individual frame, it will be perceptually acceptable and the implicit hole filling is sufficient. But while the resulting video of interleaved synthesized views will be much better aligned than the raw interleaved frames, it will still need some post processing (Figure 13 and 14). Indeed, we will see flickering in the disocclusion regions surrounding objects that are:

- static or very slowly moving, and
- close to the camera, and
- with big gap depth from their background.

While stretching the texture is sufficient for good visual quality in individual frames coming from C_1, C_2, ..., and C_N, there is no guarantee that this implicit hole filling is coherent across temporally consecutive frames coming from different cameras C_i and C_j. Particularly in static regions or very slowly moving regions, viewers expect the consecutive frames to be exactly the same or very similar.

[1] Note that even if we directly compute the flow $F4 \rightarrow F0$ instead of estimating it, we still need the individual shorter flows, e.g., $F1 \rightarrow F2$, to finally get $F3 \rightarrow \grave{F}$. The same applies to estimating the flow from frame $F3$ to all the remaining N-1 cameras.

But in the high-speed video, consecutive frames come from different cameras and the implicit impainting from content-aware warping slightly differs from one frame to the next. Due to this slight difference, our eye will notice this alternating filling between the frames as the high-speed video plays. Contrarily, in significantly moving regions, we will not see this artifact, as the consecutive frames are anyway expected to be significantly different in these regions and we will not have this alternating jittery filling.

But in static or slowly moving regions, we don't actually need high-speed video. We only need high frame rate in the highly moving regions. Therefore, our algorithm chooses one camera as reference and gets the static/slow regions from this camera as in Figure 10 and 11. In the final video, every frame is composited of the static/slow regions from single camera video and the moving regions from the full highspeed video. Our method not only fixes the visible distortions, but also leads to a video with adaptive frame rate, varying based on amount of motion.

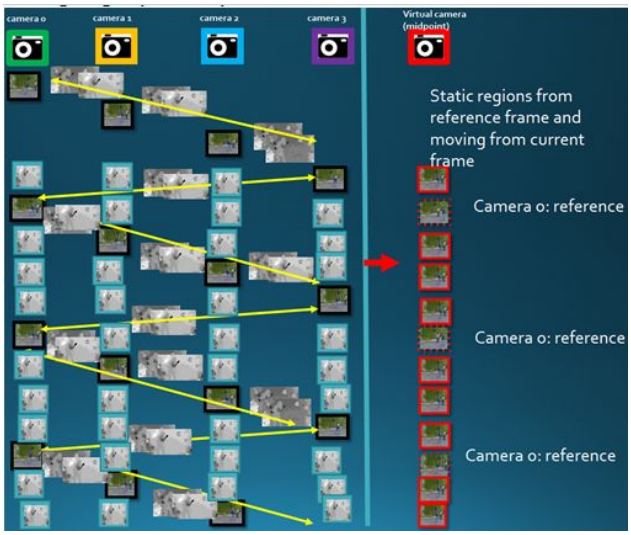

Figure 10. *Compositing: get the static/slow regions from frames of reference camera and the remaining highly moving regions from all the cameras.*

Getting the mask and blending

In every synthesized view, we need to identify the highly moving regions and the remaining regions. Analyzing the temporal gradient of the video as shown in Figure 11, we can get a mask indicating the highly moving regions. The temporal gradient needs to be cured to get rid of noise and get connected components together. For this purpose, we can use classical morphological image processing. At the end, we get a mask that partitions the synthesized view into highly moving regions vs the other static/slow regions. Having the masks, we generate the final video by compositing. For every synthesized view and mask, we generate a new image composited from 2 regions as in Figure 12:

- get the highly moving regions indicated in the mask from the synthesized view itself, and
- get all the remaining regions, static/slow regions, from the closest precedent synthesized frame from the reference camera.

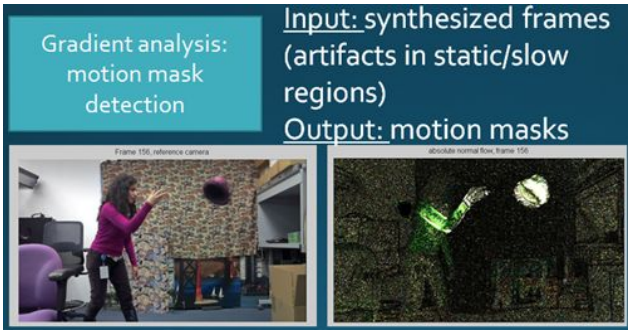

Figure 11. *Getting the motion mask from the temporal flow.*

Many compositing techniques exist to ensure smooth transitions, we use the classical Poisson method of [22].

Consequently, our algorithm generates a high-speed video that is perceptually pleasant. At the same time, this video is highly adequate for compression, as it uses the available bandwidth wisely. It effectively has an adaptive frame rate, efficiently tailored to the amount of motion in the video, with low frame rate in static/slow regions and high frame rate in high-speed regions. This approach also solves the artifacts due to the impact of cameras differences. For example, the individual cameras in the camera array may have slight differences in their sharpness level, which may cause artifacts in the static and slow moving regions as above.

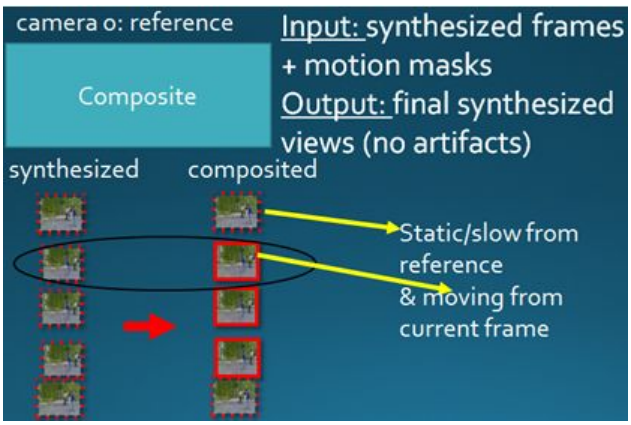

Figure 12. *Compositing: getting the views of the final highspeed video.*

Results

For our experiments, we acquire video with our camera array prototype, consisting of a planar grid of 2x2 cameras as in Figure 2b. Each camera captures 1920x1080 frames at F = 30 fps, which are rectified and input to our algorithm. We captured realistic scenes with objects at different depth layers, which can be static or moving, and have complex background. Our videos also have higher resolution than the videos in [1, 2], necessitating more accurate alignment as artifacts are much more visible at higher resolution. Please download and see 'stabilizedHighSpeed', 'umbrella',

'hat','throwHat', 'jump', 'volleyBall', and 'movingCamera' 4x slow mo results from this link[2]. The bottom of Figure 13 shows difference of 2 consecutive raw frames, captured with the camera-array static. Except for expected large magnitude on moving umbrella, most other highlighted areas illustrate misalignment due to different perspectives. After view synthesis, top of Figure 14 shows great improvement, but disocclusion areas still have artifacts (red arrows). Finally, compositing leads to much more stable video as in bottom of Figure 14, except for minor remaining artifacts. The scene 'movingCamera' is captured in handheld mode. Unlike our compositing method, our view synthesis component is directly applicable in presence of camera motion. But the need for compositing is less critical in this case, since the camera motion alleviates the flickering artifacts around disocclusions for the same reasons in **Compositing** section. Briefly, the adjacent frames in the video are anyway different even in the static regions due to the moving camera, hence independently filling the disocclusions across frames is much less noticeable.

Figure 13. *Top: rectified frame from example scene. Bottom: difference of 2 consecutive frames.*

Conclusion and Future Work

Our algorithm scales up the video rate of the camera array to factors larger than the individual cameras' rate. With our algorithm, camera-arrays can also capture high quality high-speed video, matching cameras dedicated for this sole purpose. In the future, we will optimize the algorithm to be real-time. We can also explore solving for the motion masks as solving for labels in a graph-cut problem similarly to [21, 23]. While this approach is heavier computationally, it would give more accurate masks and even higher quality results. Finally, we need to extend the compositing component to accommodate camera motion and allow for handheld use.

[2] https://drive.google.com/open?id=0B3-CYFBmsrsBT2RiS3AtdkxySlE

Figure 14. *Top: difference of 2 consecutive synthesized frames. Bottom: difference of 2 consecutive composited frames in the final video.*

References

[1] B Wilburn, N Joshi, V Vaish, M Levoy, and M Horowitz, High Speed Video Using a Dense Camera Array, CVPR 2004

[2] B Wilburn, N Joshi, V Vaish, E Talvala, E Antunez, A Barth, A Adams, M Horowitz, and M Levoy, High performance imaging using large camera arrays, Siggraph 2005.

[3] H G Dietz, FourSee TDCI Multi-Camera, University of Kentucky, Aggregate.Org online technical report, http://aggregate.org/DIT/FourSee/ 2015

[4] https://www.phantomhighspeed.com/

[5] http://improvephotography.com/30019/iphone-6-camera-depth-review/

[6] http://www.bbc.com/news/science-environment-35586935

[7] http://www.cnet.com/products/dell-venue-8-7000/

[8] http://www.pelicanimaging.com/technology/mobile.html

[9] http://www.gizmag.com/light-camera-combines-16-sensors/39764/

[10] http://www.cnet.com/products/huawei-p9/

[11] http://www.computerworld.com/article/2476104/smartphones/in-pictures–here-s-what-the-htc-one-s-dual-cameras-can-do.html

[12] http://www.androidcentral.com/lg-g5-hands-on

[13] F Liu, M Gleicher, H Jin, and A Agarwala, Content-

preserving warps for 3D video stabilization, ACM Transactions on Graphics 2009
[14] N Stefanoski, O Wang, M Lang, P Greisen, S Heinzle, and A Smolic, Automatic View Synthesis by ImageDomain-Warping, IEEE Trans on Image Proc 2013
[15] BD Lucas and T Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. International Joint Conference on Artificial Intelligence 1981
[16] C Tomasi and T Kanade. Detection and Tracking of Point Features. Carnegie Mellon University Technical Report CMU-CS-91-132, 1991
[17] S Pujades, F Devernay, and B Goldluecke. Bayesian view synthesis and image-based rendering principles, CVPR 2014
[18] CL Zitnick, SB Kang, M Uyttendaele, S Winder, and R Szeliski, High-quality video view interpolation using a layered representation, Siggraph 2004
[19] J Gautier, O Le Meur, and C Guillemot, Depth-based image completion for view synthesis, 3DTV 2011
[20] M Bertalmio, G Sapiro, V Caselles, and C Ballester, Image inpainting, Siggraph 2000
[21] J Bai, A Agarwala, M Agrawala, and R Ramamoorthi. Selectively De-Animating Video, Siggraph 2012
[22] P Perez, M Gangnet, and A Blake, Poisson Image Editing, ACM Trans. Graph. 2003
[23] P Bhat, CL Zitnick, N Snavely, A Agarwala, M Agrawala, M Cohen, B Curless, and SB Kang, Using Photographs to Enhance Videos of a Static Scene, EGSR, 2007

Author Biography

Maha El Choubassi received her BE in Computer and Communications Engineering from the American University of Beirut (2003) and her Master's and PhD in electrical engineering from University of Illinois at Urbana-Champaign (2005 and 2008). She is currently a research scientist at Intel Corporation in Santa Clara, CA. Her work focuses on image/video processing and computer vision.

Oscar Nestares received his M.S. (1994) and Ph.D. (1997) in Electrical Engineering from Universidad Politecnica de Madrid. After that he was a Fulbright Visiting Scholar at Stanford University and consultant at Xerox PARC (1998-2000) and a tenured Research Scientist at the Institute of Optics, Spanish National Research Council (2000-2003). In 2003 he joined Intel Labs focusing on statistical approaches to video enhancement (de-noising, super-resolution, stabilization), its applications to consumer electronics, workload analysis, and computational photography.