SynCam: Capturing sub-frame synchronous media using smartphones

by

Ishit Mehta, Parikshit Sakurikar, Rajvi Shah, P J Narayanan

in

IEEE International Conference on Multimedia and Expo
(*ICME-2017*)

Report No: IIIT/TR/2017/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2017

SYNCAM: CAPTURING SUB-FRAME SYNCHRONOUS MEDIA USING SMARTPHONES

Ishit Mehta[†], *Parikshit Sakurikar*[⋉], *Rajvi Shah*[⋊], *P J Narayanan*[*]

CVIT, Kohli Center on Intelligent Systems, IIIT Hyderabad, India
[⋉†⋊]first.last@research.iiit.ac.in, [*]pjn@iiit.ac.in

ABSTRACT

Smartphones have become the de-facto capture devices for everyday photography. Unlike traditional digital cameras, smartphones are versatile devices with auxiliary sensors, processing power, and networking capabilities. In this work, we harness the communication capabilities of smartphones and present a synchronous/co-ordinated multi-camera capture system. Synchronous capture is important for many image/video fusion and 3D reconstruction applications. The proposed system provides an inexpensive and effective means to capture multi-camera media for such applications. Our co-ordinated capture system is based on a wireless protocol that uses NTP based synchronization and device specific lag compensation. It achieves sub-frame synchronization across all participating smartphones of even heterogeneous make and model. We propose a new method based on fiducial markers displayed on an LCD screen to temporally calibrate smartphone cameras. We demonstrate the utility and versatility of this system to enhance traditional videography and to create novel visual representations such as panoramic videos, HDR videos, multi-view 3D reconstruction, multi-flash imaging, and multi-camera social media.

***Index Terms*—** Synchronous Media Capture, Coordinated Capture, Multi-camera, Social Capture

1. INTRODUCTION

Over the years, smartphone cameras have continuously improved due to better optics and more compute power for post-processing. Once popular for casual photography, point and shoot digital cameras are now obsolete and replaced by smartphone cameras due to comparable image quality and ease of access. Despite the revolution in capture technology and paradigm, the primary consumer-captured media forms have not deviated much from traditional photos and videos.

Present day smartphones are powerful in the sense that they provide more functionality than conventional cameras in form of auxiliary sensors such as GPS, accelerometer, gyroscope and network connectivity. There is potential to utilize these auxiliary functionalities to enhance conventional imaging and produce new forms of compelling visual representations. In this paper, we leverage the networking capabilities of smartphones and propose a system for creating multi-camera media.

Several computer vision and image processing tasks such as object tracking, depth estimation, edge detection, etc. become significantly easier with multi-camera input. Multi-camera feeds can also be combined in interesting ways to create compelling new representations such as panorama videos, HDR videos, multi-shot videos, etc. However, synchronization and co-ordination of camera devices is an important issue while capturing dynamic real-world scenes. To exemplify, consider a panorama video of a dynamic scene created by stitching multiple video feeds. Lack of synchronization restricts automatic temporal alignment of frames. Even if cameras are triggered simultaneously, slight drift or lag can create severe artefacts in the final outcome. Our system provides a flexible framework to achieve sub-frame accurate synchronization for a heterogeneous (different make and model) system of smartphone cameras using a wireless protocol and visual marker based temporal calibration.

Traditional methods for multi-camera video synchronization can be broadly categorized into (i) capture-time synchronization and (ii) post-processing based synchronization. Capture-time synchronization can be achieved using hardware [1, 2] or software [3] triggers. Hardware triggers require a physical trigger device to be attached to the camera (operated remotely using a connected controller) and tend to be expensive. In comparison, software triggers based on Network Time Protocol (NTP) are cheaper. They rely on server-client architectures, in which the server sends capture signals over a LAN connection. However, these methods do not work well with heterogeneous systems of cameras as they do not compensate for device specific capture latencies. Also, they require extensive setups at the time of capture and rely on the quality of service of the network [4, 5, 6, 7]. In contrast, our system provides a flexible framework for accurate capture-time synchronization even for heterogeneous smartphone cameras.

Post-processing based synchronization methods leverage audio cues [8] or visual features based temporal alignment [9, 10]. However, the effectiveness of visual feature based

Supported by [⋉]TCS Research Scholarship Programme and [⋊]Google India PhD fellowship.

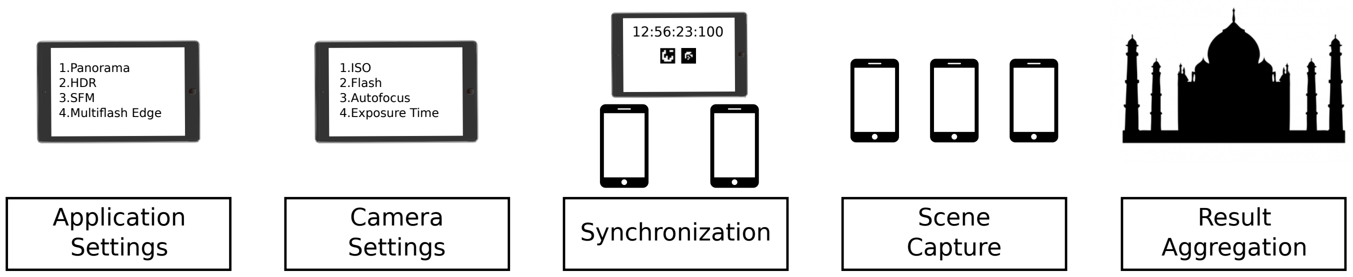


Fig. 1. System workflow. Each of the blocks represent different stages in the workflow of capturing media using a heterogenous system of uncalibrated camera phones.

alignment heavily depends on the scene elements and camera movements. Audio-based synchronization is more common in commercial video editing tools such as Adobe Premiere Pro, Final Cut Pro, and Adobe After Effects. Our system provides accurate capture-time synchronization, eliminating the need for time consuming and error-prone post-processing for synchronization.

We achieve sub-frame synchronization in two steps: (i) by compensating for system time lag using NTP, (ii) by compensating for device-specific shutter lag estimated using a marker-based calibration clock displayed on an LCD device (phone/tablet/monitor). The synchronization protocol in our work is closest to [11]. However, they use a homogeneous set of smartphones (of the same make and model) in their experiments and require custom hardware in the form of an array of LEDs. We provide an end-to-end system to capture multi-camera synchronized image sequences without the need of custom hardware and show its flexibility and usefulness by capturing and compositing a number of compelling examples.

Summarily, the contributions of this paper are, *(i)* a universal application to control heterogeneous camera phones with fine control over their camera parameters. *(ii)* a flexible and easy-to-use coordination and synchronization framework that is useful for a number of computer vision and creative authoring applications. In the following section, we provide an overview of the system and explain the workflow from user's perspective. In Section 3, we describe our synchronization protocol with supporting statistics. We discuss example applications with results in Section 4 and finally conclude the paper with a discussion on future work.

2. SYSTEM OVERVIEW AND WORKFLOW

The proposed system has a server-client fan out architecture, where clients are Android-based smartphones participating in the capture event and the server is responsible for managing the participating clients. Each client runs an Android application which provides control over several camera parameters. Upon installation of the application, the participating smartphones are registered on the server. Apart from the client and server devices, we use an additional smartphone/tablet to display the marker based clock for device-lag estimation. We now discuss the end-to-end workflow (depicted in Figure 1) of our system with a brief overview of the individual steps/modules.

Application Settings SynCam allows each participating smartphone camera to select the application configuration corresponding to the capture event. Each camera is then setup using the default settings for the event, which are decided based on the type of media such as HDR video, Panorama video, multi-flash imaging, etc.

Camera Settings For creative authoring applications, it is crucial that the user can control the capture quality and style. Using our app, a user can configure the camera parameters for each device independently at the time of registering the event. Before capture time, a user can change ISO settings, exposure time, flash, switch to front camera, and change focus distance. Depending upon the media application, the app also provides a default setting of camera parameters for all devices. For example, to capture video panoramas, all cameras are set by default to video mode, with focus at infinity, flash turned off, ISO set at 400, and exposure locked. The user is free to use the default settings or reconfigure, depending on the requirements.

Synchronization In the next step, the participating devices are calibrated to achieve sub-frame synchronization. Using NTP, we achieve a coarse level of synchronization (in hundreds of milliseconds) across all devices. To achieve sub-frame synchronization, we estimate the shutter-lag for each participating devices using a fiducial marker-based system. These two steps ensure that the capture event starts at the same time on all participating devices. Section 3 explains each step of the synchronization module in detail.

Scene Capture Different multi-cam visual media requires different camera setup configurations. Section 4 illustrates a few examples. HDR video requires the cameras to be static relative to each other whereas video captured by multiple people in a social setting cannot have such a restriction. Before capture, the user arranges the cameras in a spatial layout

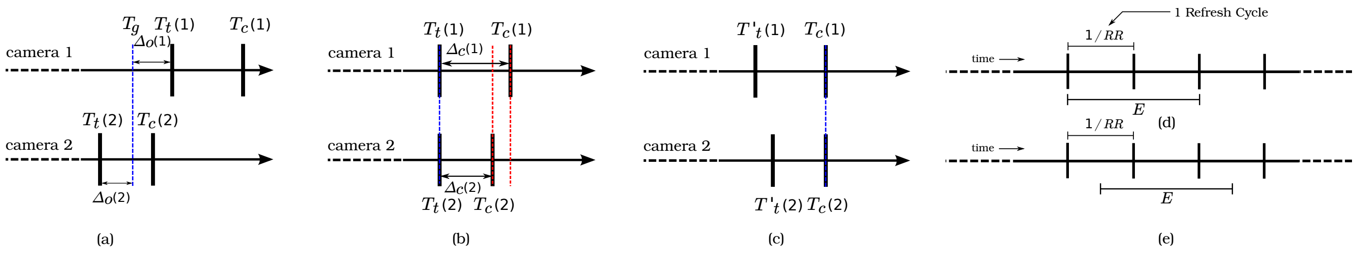


Fig. 2. Different stages of synchronization. (a) shows the initial configuration without synchronization. (b) shows the alignment after adjusting trigger time. (c) shows the ideal result after adjusting capture time. (d) and (e) show the relationship between exposure time and refresh cycle with (d) representing 2 timestamps visible and (e) shows 3 timestamps visible with same exposure time.

which is according to the type of media application selected in the second stage. The synchronized capture event is executed once each device is spatially setup according to the desired camera array configuration.

Aggregation Post-capture, the captured media (image/video) files are sent to the server using REST API. The files are stored in a central repository for easy user access. For applications where human intervention is not necessary (such as panorama video, HDR video, edge-detection, etc.), the system automatically aggregates the captured media as programmed. Representative snapshots of the final outcomes for a few such applications are shown in Section 4.

3. SYNCHRONIZATION

In order to achieve sub-frame synchronization among capturing devices, we need to first sync the device clock with the global time scale of the server. We also need to compensate for the shutter lag of each device which is the offset between the time of initiation of a capture request and actual capture time of a frame. There are three timestamps central to the synchronization system:

1. **Global Trigger Time** T_g - The global time at which the capture event is to be executed according to the server.
2. **Device Trigger Time** $T_t(i)$ - The device time at which the capture request is executed on device i.
3. **Device Capture Time** $T_c(i)$ - The device time at which the camera exposure starts on the device i. For a video, this timestamp represents the time at which the exposure starts for the first frame.

Additionally, there is a per-device offset $\Delta_o(i) = T_g - T_t(i)$, between the device i's clock and server time. The initial unsynchronized configuration is shown in Figure 2(a) where the $T_t(i)$ values for the devices may be misaligned. The capture time of a device $T_c(i)$ is ideally expected to be the same as the trigger time $T_t(i)$ of the same device. However, this is not the case as there is a varying shutter lag for capture due to the various processes running on the smartphone at that instant

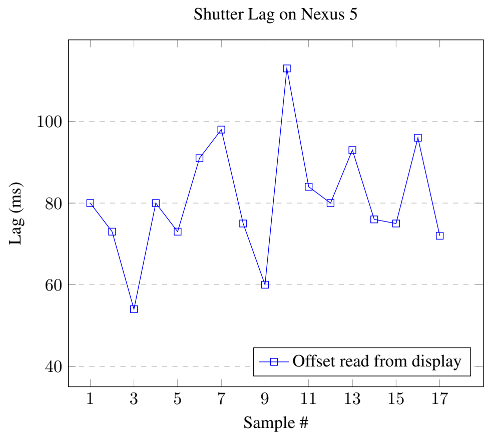


Fig. 3. Shutter lag observed on a Nexus 5 camera. It can be observed that the shutter lag varies over multiple capture samples.

– as shown in Figure 3. $\Delta_c(i) = T_c(i) - T_t(i)$ represents camera latency or shutter lag. As it varies from device to device, we synchronize the smartphone cameras according to $T_c(i)$ rather than $T_t(i)$. This ensures that each frame across multiple cameras begins exposure at the same global time.

3.1. Adjusting Trigger Time

We first calibrate the participating devices such that their trigger times align with the global trigger time T_g. This involves estimation of the $\Delta_o(i)$ for each device. We setup a stratum 4 NTP clock server and all devices use this clock as a reference. Using the NTP [3] protocol, the error in adjusting the clocks is capped by half of RTT. On a university network, we find the average RTT delay to be 11 msec. This contributes a maximum network time error (e_N) of ±5.5 msec from the global trigger time. Figure 2(b) shows the state of the system after adjusting the trigger time. It can be seen that the trigger time $T_t(i)$ for each device is now aligned with the global trigger time T_g.

3.2. Adjusting Capture Time

To calculate the shutter lag $\Delta_c(i) = T_c(i) - T_t(i)$ for each device, we use the LCD of the server device which displays

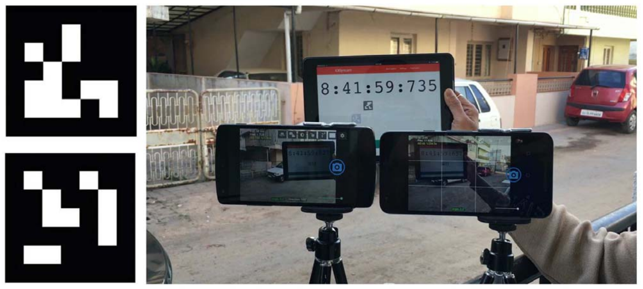


Fig. 4. Fiducial Markers – each of which represent time in milliseconds on the reference clock, are displayed on the LCD screen. Camera phones to be calibrated, capture images of the clock and markers on the screen. A dictionary of 1000 such markers is generated such that there is maximum possible inter-marker distance.

the reference clock time at milliseconds resolution. We register a capture event for all the participating devices. The LCD is kept in the field of view of all cameras. The reference clock displayed on the LCD refreshes every 16.67 msec as the LCD has a refresh rate of 60 Hz. We use high timestamp resolution and read timestamps autonomously. Instead of using digits to represent time, we use 1000 fiducial markers as shown in Figure 4. Each of the markers correspond to a specific millisecond. The distinctiveness of the markers makes them easily recognizable under varying lighting conditions.

Each camera captures an image of the LCD displaying the server time. The server time is shown using numeric digits upto the last second while markers are used to display the number of milliseconds. Multiple markers are visible in the captured image owing to finite exposure time of the frame (The relationship is shown in Figure 2(d,e)). Each marker gets activated at the time it represents and stays active for the duration of the refresh cycle. The number of markers k visible in an image is related to the refresh rate RR of the display and the exposure time of the camera E by,

$$k \in \{x, x+1\}$$

where, $x = \lceil RR \cdot E \rceil$. To detect and recognize the markers in an image as shown in Figure 4 we use the method described in [12]. The timestamp information of each visible marker in the image is then identified. Each marker can be visible for one full refresh cycle ($1/RR$) or for a smaller duration. The duration for which the marker is visible is proportional to the intensity of the marker in the image. We calculate the normalized intensity of each marker to estimate the amount of time for which the marker was visible during the exposure. We compute intensity of a marker as the mean of intensity values of white pixels in the marker. If the intensity of a marker i is I_i, then the relative intensity α_i of the marker is,

$$\alpha_i = \frac{I_i}{I_{max}} \quad \text{where, } I_{max} = \max_i I_i$$

	Nexus 5	Nexus 5x	Moto E
Adjusted trigger time	19.1	16.5	17.6
Adjusted capture time	11.35	10.63	14.12

Table 1. Camera offsets in ms. The offsets represent the mean of absolute error $|T_c(i) - T_g|$

Using this relative intensity, we can estimate the time at which the camera exposure ends and thus estimate the capture time $T_c(i)$ by:

$$T_c(i) = t + \frac{\alpha_t}{RR} - E$$

where, t is the timestamp represented by the most recent marker visible in the image and α_t is the relative intensity of the corresponding marker. $t + \frac{\alpha_t}{RR}$ represents the time at which the exposure ends, from which E is subtracted to get the time at which the exposure starts.

Figure 2(c) shows the final calibration configuration where the $T_c(i)$ values are temporally aligned. Table 1 shows the average error in alignment of $T_c(i)$ values over 30 capture events, confirming sub-frame synchronization. After synchronization the actual trigger time for a device i is set to:

$$T'_t(i) = T_t(i) - \Delta_c(i) = 2 \cdot T_t(i) - t - \frac{\alpha_t}{RR} + E$$

4. RESULTS

Panorama Sub-frame synchronization of multiple smartphone cameras enables high quality panorama stitching for scenes with active subjects. Figure 6(a) shows a still frame from a video panorama stitched using a network of four heterogeneous mobile phones. The devices used for the experiment are synchronized using the algorithm described in Section 3 and a time-synchronized video capture event is registered using the event interface. During event registration, the exposure, FPS and auto-focus for each device is fixed and set to locked mode. The phones are placed such that they remain fixed for the duration of the capture. Using a stitching pipeline similar to [13], the final video panorama is stitched frame by frame.

HDR To capture scenes with dynamic subjects in HDR, we use a pair of synchronized cameras with a large overlap in their field of view. Figure 7 shows a resultant frame from an HDR video captured using two low dynamic range videos, one with 1/33 s exposure time and the other with 1/100 s. Both cameras are placed close to each other with roughly the same vertical and horizontal alignment. We register an event to capture videos using the two cameras and create a frame-by-frame HDR video of the scene. The FPS resolution for both the cameras is locked at 30fps to ensure frame synchronization despite of the differences in exposure time. We use the Enhanced Correlation Coefficient(ECC) alignment [14] algorithm to register frames from one video to the corresponding frames in the other. We use the generalized random walks

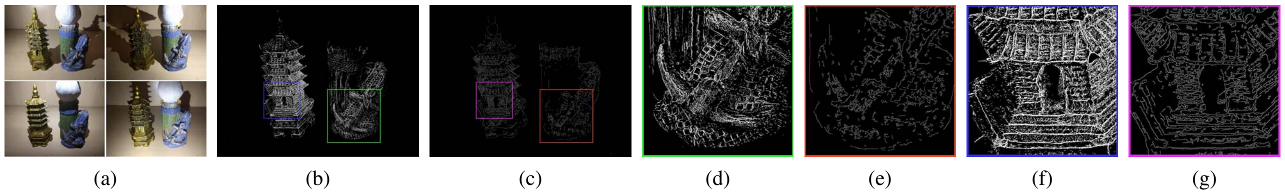
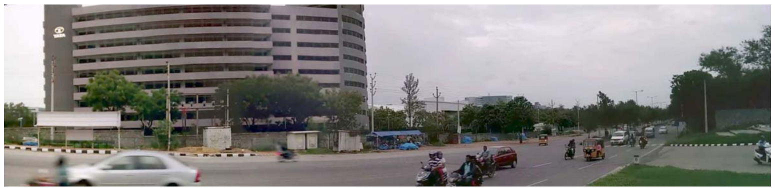
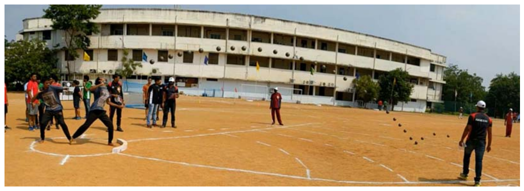


Fig. 5. Depth-edge detection using multi-flash coordinated capture. (a) shows the input images that were taken with phones acting as flash units kept in four positions around a central camera. The central camera is triggered consequently with different lighting conditions produced by the "flash" phones. (b) is the resultant image showcasing the detected edges which are significantly better than (c) obtained using canny edge detection. (d), (e), (f) and (g) highlight the difference between (b) and (c).



(a) An example frame of a video panorama with dynamic subjects.



(b) Multi-Frame Composition. A photomontage created using multiple frames of a video panorama.

Fig. 6. Media obtained by stitching synchronized videos captured by multiple phones.

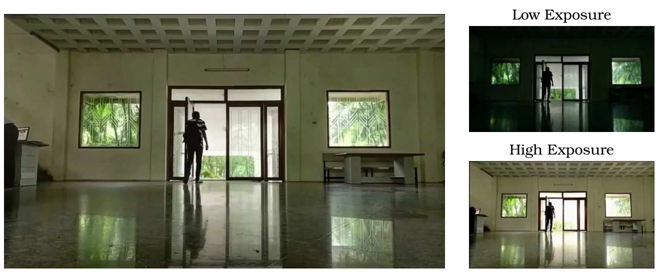


Fig. 7. HDR composition of a frame using two synchronous cameras with different exposure times.

algorithm [15] for temporally coherent multi-exposure fusion. Bilateral filter is used to temporally smoothen out variation in parameters.

Multi-Flash Edge Detection Our system also enables novel multi-camera compositions such that a few devices can be used as illumination units while the others capture images. We show a synchronized time-sequential capture application for multi-flash imaging for depth edge detection as described in [16]. We use an array of five cameras, where the central camera is the capture unit and the four surrounding cameras are used as flash triggers in a sequential manner. Using our synchronization algorithm, the central camera captures an image at exactly the same instant when each of the flash units illuminate the scene. Using the top, left, bottom and right illuminated images of the scene, we can isolate the depth and texture edges in the scene at a much better resolution than applying image-processing techniques such as Canny/Sobel edge detection over no-flash images, as described in [16]. Figure 5 shows the result on a scene with two objects having varying degrees of textural details.

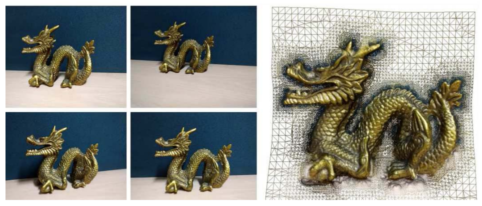


Fig. 8. Dense 3D model obtained using 5 cameras. The figure shows 4 out of the 5 simultaneously captured images alongwith the generated 3D mesh model.

Dense 3D Reconstruction Structure from motion uses multiple images of an object or a scene to create elaborate 3D point-cloud reconstructions. We use our system to demonstrate an outside-in capture framework. We capture five images of a small 3D scene and reconstruct a dense 3D model. Using these images, we obtain a sparse point cloud using the Bundler toolkit [17] and synthesize a dense point cloud using CMVS/PMVS(Clustering Views for Multi-view Stereo/Patch-based Multi-view Stereo) algorithms [18][19]. Poisson surface reconstruction [20] is then applied to generate a smooth 3D mesh as shown in Figure 8.

Multi-Cam Social Media Multi-camera images and videos can also be used for creative social media. Multiple colocated viewers capturing a synchronized videos of an event can cycle through different viewpoints and can composite different shots from different viewpoints during post-processing. An illustration can be found on our project page.

Multi-Frame Compositing Interesting image composites and photomontages from videos can be created by combining visual elements from several frames into one single image. This can be achieved for a larger field of view by using multiple synchronized video streams. To demonstrate this with an example, we capture a dynamic scene using 2 cameras with overlapping FOVs – from which a video panorama is created. By extracting a few frames spaced out evenly on the timeline, a photomontage is created using the method described in [21] as shown in Figure 6(b).

5. CONCLUSION

In this work, an end-to-end multi-camera capture framework for smartphones is introduced, which does not require any specialized hardware. This goes beyond the existing software systems which calibrate multi-camera feeds in post-production. It also improves upon the state-of-the-art hardware solutions which do not account for shutter lag. The proposed framework is able to achieve sub-frame synchronization using NTP and a marker-based timestamp detection algorithm. The suggested system can serve as a standard platform for multi-mobile capture applications.

References

[1] Ningwen Liu, Yunfeng Wu, Xianxiang Tan, and Guoji Lai, "Control system for several rotating mirror camera synchronization operation," in *Int'l Congress on High-Speed Photography and Photonics*, 1997.

[2] "Triggertrap - smartphone powered camera remotes," `http://www.triggertrap.com/`.

[3] David L Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, 1991.

[4] Lukas Ahrenberg, Ivo Ihrke, and Marcus Magnor, "A mobile system for multi-video recording," in *European Conference on Visual Media Production (CVMP)*, 2004.

[5] Piyush Kumar Rai, Kamal Tiwari, Prithwijit Guha, and Amitabha Mukerjee, "A cost-effective multiple camera vision system using firewire cameras and software synchronization," in *Proceedings International Conference on High Performance Computing*, 2003.

[6] Tomáš Svoboda, Hanspeter Hug, and Luc Van Gool, "Viroomlow cost synchronized multicamera system and its self-calibration," in *Joint Pattern Recognition Symposium*, 2002.

[7] Georgios Litos, Xenophon Zabulis, and Georgios Triantafyllidis, "Synchronous image acquisition based on network synchronization," in *Proceedings CVPR Workshop*, 2006.

[8] Prarthana Shrestha, Mauro Barbieri, Hans Weda, and Dragan Sekulovski, "Synchronization of multiple camera videos using audio-visual features," *IEEE Transactions on Multimedia*, 2010.

[9] Ahmed Elhayek, Carsten Stoll, Kwang In Kim, H-P Seidel, and Christian Theobalt, "Feature-based multi-video synchronization with subframe accuracy," in *Joint DAGM (German Association for Pattern Recognition) and OAGM Symposium*, 2012.

[10] Tiago Gaspar, Paulo Oliveira, and Paolo Favaro, "Synchronization of two independently moving cameras without feature correspondences," in *European Conference on Computer Vision*, 2014.

[11] Richard Latimer, Jason Holloway, Ashok Veeraraghavan, and Ashutosh Sabharwal, "Socialsync: Sub-frame synchronization in a smartphone camera network," in *European Conference on Computer Vision*, 2014.

[12] S Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, 2014.

[13] Matthew Brown and David G Lowe, "Automatic panoramic image stitching using invariant features," *International journal of computer vision*, 2007.

[14] G.D. Evangelidis and E.Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[15] R. Shen, I. Cheng, J. Shi, and A. Basu, "Generalized random walks for fusion of multi-exposure images," *IEEE Transactions on Image Processing*, 2011.

[16] Ramesh Raskar, Kar-Han Tan, Rogerio Feris, Jingyi Yu, and Matthew Turk, "Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging," in *ACM transactions on graphics (TOG)*. ACM, 2004.

[17] Noah Snavely, Steven M Seitz, and Richard Szeliski, "Photo tourism: exploring photo collections in 3d," in *ACM transactions on graphics (TOG)*, 2006.

[18] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski, "Towards internet-scale multi-view stereo," in *CVPR*, 2010.

[19] Yasutaka Furukawa and Jean Ponce, "Accurate, dense, and robust multi-view stereopsis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2010.

[20] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe, "Poisson surface reconstruction," in *Proceedings Eurographics symposium on Geometry processing*, 2006.

[21] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen, "Interactive digital photomontage," in *ACM Transactions on Graphics (TOG)*, 2004.