HARDWARE SYNCHRONIZATION OF MULTIPLE KINECTS AND MICROPHONES FOR 3D AUDIOVISUAL SPATIOTEMPORAL DATA CAPTURE

[1]Yijun Jiang, [1]David Russell, [2]Timothy Godisart, [1]Natasha Kholgade Banerjee, [1]Sean Banerjee

[1]Clarkson University, Potsdam, NY, [2]Oculus Pittsburgh, Pittsburgh, PA
{jiangy, russeldj, nbanerje, sbanerje}@clarkson.edu, timothy.godisart@oculus.com

ABSTRACT

We present an approach that uses linear timecode to temporally synchronize visual data captured at 30 fps from multiple Kinect v2 sensors and audio data captured from external microphones. Existing approaches suffer from reduced frame rate due to latencies in relaying synchronization signals over a network, have inaccuracies due to ambient sound contamination when using audio for synchronization, or show multi-frame delays when using networking protocols to synchronize capture computer clocks. We align audio and color frame times to a timecode signal injected using a custom-designed hardware board to the audio processing board of each Kinect. Our approach provides synchronized capture unaffected by ambient noise or network latencies, with maximum synchronization offsets within a single frame for any number of devices. We show 3D point cloud reconstruction results with audio for a variety of single- and multi-person interactions captured using four Kinect v2 sensors and two external microphones synchronized by our approach.

Index Terms— synchronization, timecode, spatiotemporal, 3D reconstruction, audiovisual, Kinect, depth

1. INTRODUCTION

The pervasion of consumer RGB-D capture devices such as the Microsoft Kinect v1 and v2 sensors has provided significant momentum to the reconstruction of 3D spatiotemporal data from multiple sensors [1, 2, 3, 4, 5, 6, 7, 8, 9]. A principle challenge in generating spatiotemporal reconstructions from multiple Kinect v2 sensors is ensuring that the data captured by the sensors is temporally synchronized. Since a single computer supports capture from only one Kinect v2, traditional approaches perform synchronization over the network connecting the capture computers. As discussed in [6], Kinect v2 synchronization approaches such as Live Scan3D [5] which send capture requests over the network fail to achieve full frame rate capture of 30 frames per second due to network latencies. The authors of [6] provide an approach to synchronize the clocks of the capture computers for two Kinect v2 sensors using Network Time Protocol (NTP). However, as mentioned by the authors, their approach introduces a delay of up to 30 milliseconds (ms) or nearly a single frame which can degrade as the number of capture computers increases. Approaches that automatically search for cues such as a flashing light or a canonical audio signal [7] suffer from mismatches in synchronization when the cues are altered by body and object motions, spoken content, or ambient noise typical of everyday interactions.

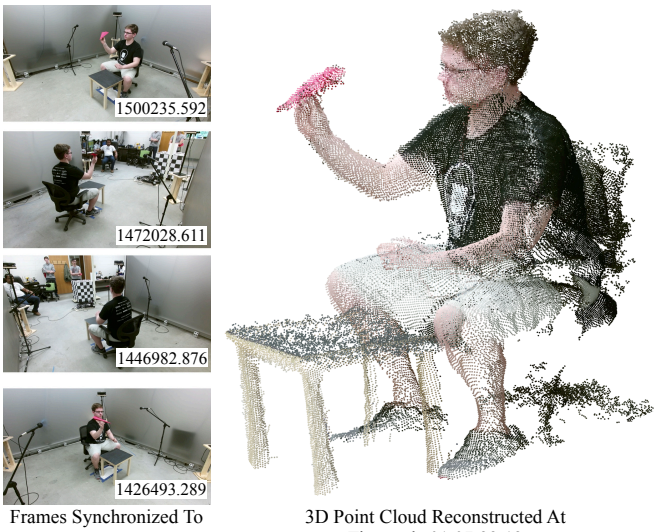

Fig. 1. Given a set of unsynchronized frames from multiple Kinect v2 sensors as shown by the different internal Kinect color frame times in milliseconds on the left, we provide an approach that uses linear timecode to synchronize the frames. Our approach enables merged 3D point cloud reconstructions from multiple sensors with sub-frame synchronization.

We present an approach that uses linear timecode (LTC) to address the challenge of temporally synchronizing multiple Kinect v2 sensors for 3D point cloud reconstruction as shown in Figure 1. The novelty of our approach is to leverage the audio input that is ubiquitous on most consumer devices to transfer linear timecode as a global synchronization signal in the absence of generator locking ports. We insert a low-cost custom-designed hardware synchronization board between the Kinect microphones and the audio processing

978-1-5386-1737-3/18/$31.00

Fig. 2. Our approach enables synchronized capture of 3D motions such as the subject folding the paper from multiple Kinect sensors at 30 frames per second. Timecode in `hour:minute:second:frame` format is shown above each frame.

board that transmits an input timecode signal to the audio processing board. Our approach automatically decodes the output signal from the audio processing board as timecode values. We synchronize multiple Kinect v2 sensors by searching for correspondences between the times at which color frames are captured by the sensors, and the times corresponding to the audio frames captured at each timecode value.

Our approach allows the Kinects to capture 3D motions such as those shown in Figure 2 at their full frame rate of 30 frames per second. Our approach is unaffected by background noise or network latencies. As shown by our results, the maximum offsets in synchronization are no more than a single frame, irrespective of the number of devices used in capture. By using linear timecode, we can synchronize any number of Kinects to each other, and to other devices that accept timecode through a dedicated port or through audio input. We demonstrate multi-device synchronization using four Kinect v2 sensors and two external microphones.

The audio output from the Kinect v2 sensor shows features characteristic of off-the-shelf devices, e.g., active cancellation of repetitive patterns such as timecode signals treated as background noise, and occasional signal corruption due to imperfect processing by the Kinect hardware. Our approach provides two contributions to extract a clean audio signal from the noisy output. To suppress active noise cancellation, we install a set of relays on the hardware synchronization board that swap the timecode data at regular intervals for random audio signals from the microphones over short time periods. To repair the corrupted timecode signal, we automatically estimate corrected timecode values by examining consecutive pairs of decoded timecode and audio frame times.

2. RELATED WORK

With the pervasion of low-cost RGB-D sensors such as the Microsoft Kinect, many approaches have used multiple Kinect v1 sensors to provide merged 3D reconstructions of human activities [1, 2, 3, 4]. Since up to four Kinect v1 sensors can be connected to one computer, approaches that use the Kinect v1 leverage low inter-process communication to perform software synchronization of the Kinect v1 sensors.

Unlike the Kinect v1, a single computer supports one Kinect v2 sensor. Approaches that synchronize multiple Kinect v2 sensors traditionally use a networked architecture since the devices lack generator locking or timecode ports. The approach of Live Scan3D [5] synchronizes Kinect v2 sensors by sending a capture request over the network and waiting for a response prior to relaying the next request. However, the approach is unable to handle full frame rate capture due to sensor start time delays and network latencies. In [6], the authors synchronize a pair of Kinect v2 sensors by using Network Time Protocol (NTP) and recording Kinect frame and thread time stamps. However, their approach introduces a maximum lag of 30 ms or 1 frame, which can degrade when the number of devices is increased. The authors of [7] use an audio signal to perform post-capture synchronization of multi-Kinect data. However, ambient noise typical of everyday environments can cause temporal misalignments.

Due to the move toward using larger densities of Kinect sensors and similar off-the-shelf capture devices [10, 7, 5, 11, 12, 4, 13, 14, 15], accurately synchronizing dense arrays of sensors becomes vital to create temporally aligned 3D reconstructions. Our hardware synchronization approach handles any number of Kinect v2 sensors, and works with any device with a timecode port or microphone jack. In this paper we demonstrate the versatility of our system by synchronizing four Kinect v2 sensors and two microphones.

3. MULTI-KINECT V2 HARDWARE SYNCHRONIZATION

Our hardware synchronization approach uses linear timecode (LTC) to synchronize the Kinects and microphones. We choose LTC over repetitive signals such as sine or square waves since LTC acts as a natural 30 fps counter, and provides hour, minute, second, and frame values for each video or audio frame. While Gray code is a possible choice of counting signal, LTC has the advantage of having 16 parity bits of the form 0011 1111 1111 1101 that are used to prevent decoding errors from propagating in time. We generate LTC using a Denecke SB-3 generator, and amplify it using a Pelco CM9760-MDA amplifier to handle increasing number of capture devices. We capture environment audio using the Behringer microphones. To synchronize the microphone audio, we transmit the amplified timecode signal to two Alto ZMX52 mixers. Each mixer outputs two-channel audio as shown in Figure 3, where the right and left channels consist respectively of the timecode signal and the audio from the corresponding microphone.

To synchronize the Kinects using LTC, we re-purpose the audio channel on the Kinect sensors as a carrier for the timecode signal. We transmit the amplified timecode signal to a custom-designed hardware synchronization board installed on the Kinect as shown in Figure 4. The timecode signal is applied across the pins LTC+ and LTC-. We remove the wires to the microphone and audio processing board connections on

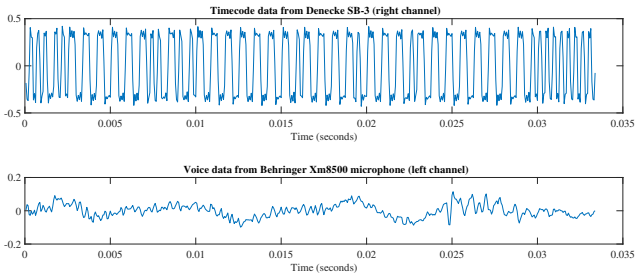

Fig. 3. Timecode and microphone data is captured as a mix consisting of the timecode signal in the right channel and microphone audio in the left channel. The timecode frame and value on the top is corresponded to the microphone data frame on the bottom.

the factory Kinect v2 sensor and connect them to the points on the timecode board marked 'To mics' and 'To Kinect audio processing board' respectively. The synchronization board is compactly housed behind the Kinect, and preserves the original functionality of the sensor.

As shown in Figure 5(a), the timecode appears as a self-repeating signal and triggers active noise cancellation on the Kinect to curb the signal after 300 seconds. The noise cancellation cannot be disabled by the Kinect SDK. To circumvent noise cancellation, we install four Omron G6k-2P relays on the hardware synchronization board to switch between microphone audio data and timecode data as shown in Figure 4. As shown in Figure 5(b), we trigger the relays every 2 minutes by sending a 5 volt signal across the pins 5V and ground in Figure 4. The relays intersperse the timecode data with 15 seconds of audio from the Kinect microphones, which appears as a random signal to the noise cancellation hardware. Our approach suppresses active noise cancellation for captures as long as 60 minutes.

We decode the recorded timecode captured by the Kinect v2 sensors and the microphones using the `libltc` library [16]. The decoded results for the i^{th} Kinect consist of timecode values in `hour:minute:second:frame` format stored in the time-series vector $\mathbf{v}_i$, and corresponding audio frame times stored in time-series vector $\mathbf{a}_i$. Signal corruption due to imperfect processing may cause single timecode values to be decoded incorrectly. Gaps in timecode values occur when the relays are cycled to circumvent the active noise cancellation. For the i^{th} Kinect we check if the timecode values $\mathbf{v}_i[p]$ and $\mathbf{v}_i[p+1]$ are a frame apart, i.e., if

$$\mathbf{v}_i[p+1] = \mathbf{v}_i[p] + 1. \tag{1}$$

If the condition in Equation (1) holds, we leave the timecode values $\mathbf{v}_i[p]$ and $\mathbf{v}_i[p+1]$ unchanged. Otherwise, we fix incorrect timecode values in $\mathbf{v}_i$ using the internal audio frame times stored in $\mathbf{a}_i$. We check whether the next audio frame time is ahead of the current audio frame time by one frame or $t = 33.33$ ms within a tolerance of $\epsilon = 3$ ms to account for

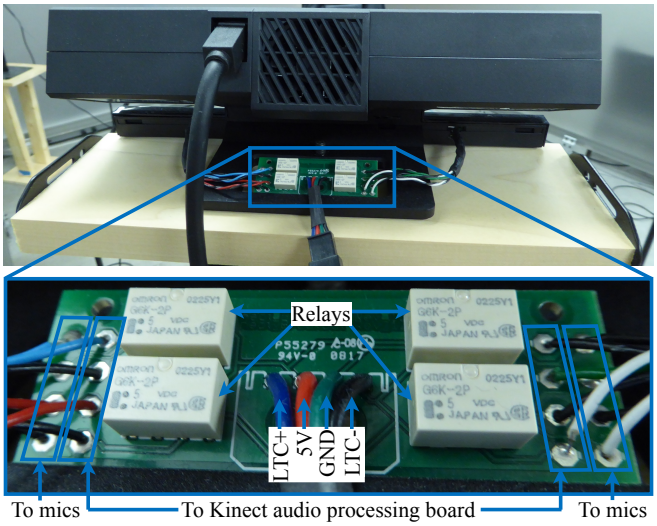

Fig. 4. We install a compact hardware synchronization board housed behind the Kinect, that does not affect sensor usability. The board accepts LTC signals from the timecode generator through the LTC+ and LTC- pins. The four Omron G6K-2P relays are triggered by the 5V input and send either timecodes or microphone signals to the Kinect audio processing board.

millisecond level differences in capture times, i.e., if

$$|(\mathbf{a}_i[p+1] - \mathbf{a}_i[p]) - t| \leq \epsilon. \tag{2}$$

Equation (2) holding represents a decoding error for the timecode $\mathbf{v}_i[p+1]$, in which case we correct timecode $\mathbf{v}_i[p+1]$ by adding one frame to $\mathbf{v}_i[p]$. Equation (2) being violated represents a gap of frames missing due to the relay cycling. In this case, we compute the number of missing frames G as

$$G = \left((\mathbf{a}_i[p+1] - \mathbf{a}_i[p])\,/t\right) - 1. \tag{3}$$

We insert G timecodes $\mathbf{v}_i[p]+1$, $\mathbf{v}_i[p]+2$, $\cdots$, $\mathbf{v}_i[p]+G$ between $\mathbf{v}_i[p]$ and $\mathbf{v}_i[p+1]$, and G audio frame times $\mathbf{a}_i[p]+t$, $\mathbf{a}_i[p]+2t$, $\cdots$, $\mathbf{a}_i[p]+Gt$ between $\mathbf{a}_i[p]$ and $\mathbf{a}_i[p+1]$.

While rare, operating system bottlenecks may cause the sensors to drop color frames. We estimate missing color frames for proper correspondence between timecode values and color frame times. We store the color frame times for the i^{th} Kinect in vector $\mathbf{c}_i$. For the q^{th} color frame time in $\mathbf{c}_i$ we check whether the next color frame time is ahead by t ms within a tolerance of ϵ, i.e., if

$$|(\mathbf{c}_i[q+1] - \mathbf{c}_i[q]) - t| \leq \epsilon. \tag{4}$$

If the condition in Equation (4) is violated, we insert G dropped color frame times $\mathbf{c}_i[q]+t$, $\mathbf{c}_i[q]+2t$, $\cdots$, $\mathbf{c}_i[q]+Gt$ between $\mathbf{c}_i[q]$ and $\mathbf{c}_i[q+1]$, where G is computed using Equation (3) by replacing $\mathbf{a}_i$ with $\mathbf{c}_i$ and p with q.

While the timecode values in $\mathbf{v}_i$ are corresponded to the audio frame times in $\mathbf{a}_i$, the color frame times in $\mathbf{c}_i$ lack a

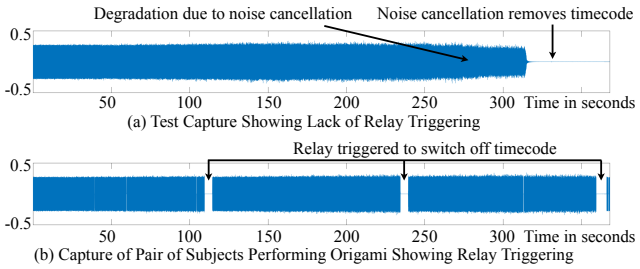

Fig. 5. (a) Timecode captured without relays being triggered. After 250 seconds the signal begins to degrade. After 300 seconds, the signal is removed due to noise cancellation. (b) With relay triggering the signal persists after 300 seconds.

one-to-one correspondence to $\mathbf{v}_i$. We obtain correspondences between the timecode values and color frame times by comparing the values in $\mathbf{c}_i$ to the audio frame times in $\mathbf{a}_i$. For the p^{th} timecode in $\mathbf{v}_i$ we find frame $\mathbf{c}_i[q]$ in $\mathbf{c}_i$ where

$$\mathbf{c}_i[q] \geq \mathbf{a}_i[p] - \epsilon \wedge \mathbf{c}_i[q] < \mathbf{a}_i[p+1] - \epsilon. \quad (5)$$

We set location $\mathbf{c}'_i[p]$ in a vector $\mathbf{c}'_i$ to the color frame time $\mathbf{c}_i[q]$. The vector $\mathbf{c}'_i$ now contains color values with one-to-one correspondence to the timecode values in $\mathbf{v}_i$.

Due to differences in the start times of the Kinects, the first timecode value $\mathbf{v}_i[1]$ differs across the sensors. To temporally align N Kinects, where N is the number of sensors, we determine a reference Kinect r such that $r = \arg\max_i \mathbf{v}_i[1]$. The reference Kinect represents the sensor that starts the latest. For each reference timecode $\mathbf{v}_r[p]$, we find the location m_i in the timecode vector $\mathbf{v}_i$ of the i^{th} Kinect sensor, such that $\mathbf{v}_i[m_i] = \mathbf{v}_r[p]$. We return the corresponding color frame time $\mathbf{c}'_i[m_i]$ for the i^{th} Kinect, and create a tuple $(\mathbf{v}_r, \mathbf{c}'_1[m_1], \mathbf{c}'_2[m_2], \cdots, \mathbf{c}'_N[m_N])$ for all N Kinects. The tuple consists of the color frame times for all sensors synchronized to each timecode value. We align each external microphone to the Kinects by obtaining the audio frame number for the microphone corresponding to each timecode value from the reference Kinect.

Our timecode validity check and correction approach is flexible enough to account for changes in noise cancellation trigger time. If the cancellation time is longer than two minutes, there is no change in our approach, as the relay triggering will continue circumventing noise cancellation. If the cancellation time tc is shorter than two minutes, there is a period of $(2 - t_c)$ minutes where the timecode signal is incorrectly decoded due to signal corruption. However, once the relay is triggered, the correct timecode values re-appear. Our approach uses the timecode values after relay trigger to correct the timecode values in the $(2 - t_c)$ minute interval corresponding to the corrupted timecode signal and in the 15 second interval corresponding to the relay trigger.

Fig. 6. Two views of 3D point clouds for winning frames of a rock-paper-scissors (rps) duel.

4. RESULTS

We show 3D point cloud reconstructions for a variety of single- and multi-person activities captured by the four Kinect v2 sensors and two external microphones in Figures 2, 6, and 8. Timecode values are indicated above the point cloud for each frame. To perform 3D reconstruction, we obtain the extrinsic transformations relating the depth cameras of all Kinects by calibrating them using a checkerboard cube imaged in the infrared domain. We use the Kinect SDK to reconstruct 3D points and merge the points by applying the extrinsic camera transformations. We perform ground plane removal using RANSAC and outlier removal by identifying points whose distance to their k nearest neighbors is greater than a threshold similar to the approach of [5].

Our approach captures hand and finger poses in activities such as the rock-paper-scissors game in Figure 6, the guitar player in Figure 8(b), the chair building contest in Figure 9, the origami teaching session in Figure 10, and the curving of the wrist in the last frame as the player tracks the ping pong ball in Figure 8(a). Our approach also captures intricate paper folding operations such as opening and flattening pockets in Figure 10(a), pulling folds out to make an origami boat as in Figure 10(b), and curving and creasing paper in Figure 2 and Figure 8(c) to build the paper plane in Figure 1.

The synchronization between the cameras allows tracking of progress in competitive activities such as the chair building contest in Figure 9, and multi-viewpoint visualization of cooperative activities such as the teacher assisting the student in Figure 10(c). As shown in the accompanying video, the guitar capture in Figure 8(b) is synchronized to the external microphone audio. The supplementary material shows original color images for all captures.

Light Synchronization: Figure 7 shows a comparison of our timecode synchronization approach against synchronizing the sensors using light patterns. We generate a light pattern by turning a high intensity LED light on and off two times. We perform light synchronization by computing the absolute values of the differences between successive images. In ideal

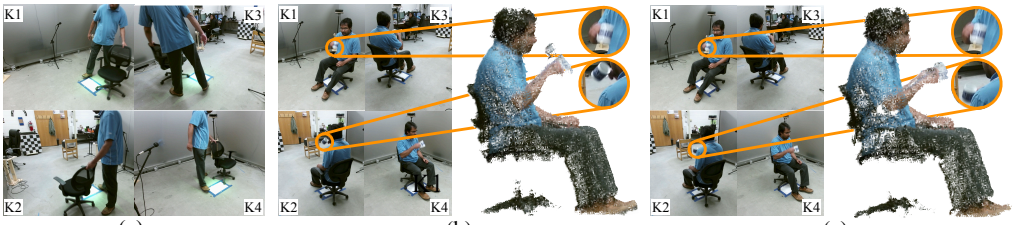

Fig. 7. (a) Temporal synchronization approaches that use light patterns cause misalignment if the subject moves while the light shines. (b) Misaligned coffee mug due to mismatches from Kinects 2, 3, and 4 to Kinect 1. (c) Our approach embeds timecode directly into the Kinect audio to provide synchronized alignment across all frames.

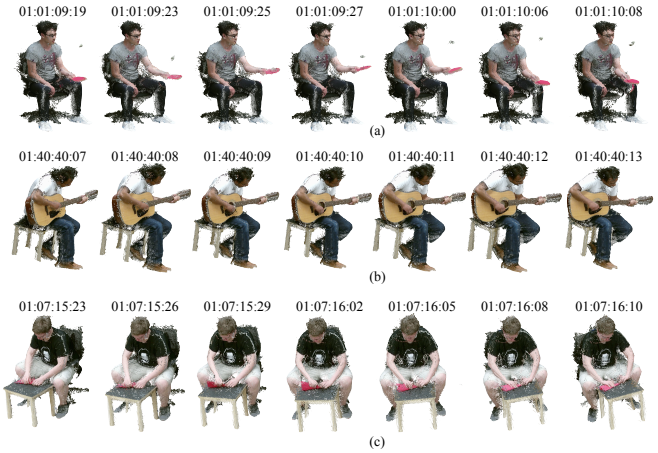

Fig. 8. (a) Frames from player tossing a ping-pong ball with a paddle. (b) A musician playing the guitar showing hand movements across the sound hole. The supplementary video provides the audio track of the guitar synchronized to the 3D reconstruction of the musician. (c) Subject from Figure 2 folding the plane body in half.

Name	K1 Offset	K2 Offset	K3 Offset	K4 Offset
paper-plane	14.89 ± .48	-2.30 ± .49	1.02 ± .53	24.65 ± .42
tea	10.86 ± .47	26.60 ± .53	-2.16 ± .53	21.09 ± .36
rps	28.53 ± .53	9.87 ± .42	6.88 ± .41	.83 ± .48
ping-pong	26.30 ± .53	7.36 ± .43	4.63 ± .49	-1.82 ±.41
chair-contest	20.56 ± .48	6.81 ± .53	-.31 ± .52	16.01 ± .53
guitar	22.52 ± .49	14.71 ± .53	8.13 ± .53	19.96 ± .51
table-build	5.32 ± .53	26.53 ± .53	10.43 ± .43	22.73 ± .49

Table 1. Offsets from timecode in ms for various captures.

conditions, when there is no subject movement, the largest absolute differences occur four times when the light transitions from on to off or off to on. We detect the top four peaks in the absolute differences, and use the last peak corresponding to the switching off of the LED light to synchronize all cameras. However, in the capture shown in Figure 7, the subject moves in the scene during the switching on and off of the light as shown in Figure 7(a). We show the effect of subject

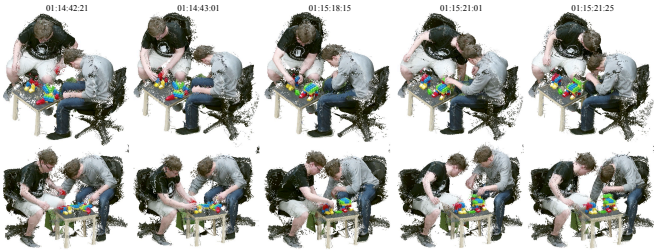

Fig. 9. Two views of 3D point clouds from five frames of a contest to build a chair out of toy blocks. While the contestant on the left chooses to add one piece at a time to his model, the one on the right uses several blocks at once.

motion when using light synchronization in Figure 7(b). In the 3D reconstruction, the coffee mug appears misaligned as Kinects 2, 3, and 4 are mismatched from Kinect 1 by 5, 1, and 2 frames respectively. We show the 3D reconstruction using our approach in Figure 7(c), where the coffee mug is correctly aligned. Unlike light synchronization, our approach is unaffected by subject motions as we embed the timecode into the audio recorded by the Kinect.

Offsets: Since the Kinects lack a trigger to start them at a desired time, their start-times can introduce offsets in the synchronization of up to one frame. Table 1 shows the offsets in milliseconds of each sensor from the closest timecode value for seven captures. Small negative values arise from a frame being captured just before the timecode within the tolerance ϵ. The uncertainties represents frame-to-frame differences in the internal times taken by each Kinect sensor to capture a frame of data. Internal times typically vary between 32.5 ms and 34.5 ms. The maximum synchronization offset is 28.76 ms between Kinects 2 and 3 for the tea capture, which is less than 33.33 ms or one frame.

5. DISCUSSION

We have provided an approach that uses linear timecode to synchronize multiple Kinect v2 sensors using hardware synchronization boards installed on the Kinects. Our approach

Fig. 10. Teacher teaching student how to build an origami boat. (a) Top view showing opening of pocket to flat out into square. (b) Front view showing opening of boat flaps. (c) Teacher assisting student to puff out the boat. (d) Two views of final models.

enables the Kinect sensors to capture at their full frame rate. Unlike existing approaches, our system is not affected by network latencies or background noise. Our hardware synchronization board is low-cost and does not impact the original functionality of the sensor.

Unlike the Kinect v1 sensor, the Kinect v2 uses a time-of-flight camera to measure depth. Our testing has shown that a large number of Kinect v2 sensors can be used without interference as long as the sensors do not directly face each other. In future work, we will investigate the effects of interference on the accuracy of the 3D reconstruction by using a high density array of Kinect v2 sensors. By using linear timecode we enable the capture of 3D audiovisual spatiotemporal data from any device which accepts timecode through a native timecode port or an external microphone jack. In future work, we will channel the timecode signals to consumer devices such as GoPros and smartphones to create dense low-cost multi-view capture environments.

6. ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation (NSF) grant #1730183. We thank Yaser Sheikh, Tomas Simon, and Taiki Shimba for insightful discussions.

7. REFERENCES

[1] Jing Tong, Jin Zhou, Ligang Liu, Zhigeng Pan, and Hao Yan, "Scanning 3d full human bodies using kinects," *IEEE TVCG*, 2012.

[2] Bernhard Kainz, Stefan Hauswiesner, Gerhard Reitmayr, Markus Steinberger, Raphael Grasset, Lukas Gruber, Eduardo Veas, Denis Kalkofen, Hartmut Seichter, and Dieter Schmalstieg, "Omnikinect: Real-time dense volumetric data acquisition and applications," in *VRST*, 2012.

[3] Angelos Barmpoutis, "Tensor body: Real-time reconstruction of the human body and avatar synthesis from rgb-d," *IEEE Trans. Cybernetics*, 2013.

[4] Dimitrios S Alexiadis, Dimitrios Zarpalas, and Petros Daras, "Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras," *IEEE Trans. Multimedia*, 2013.

[5] Marek Kowalski, Jacek Naruniec, and Michal Daniluk, "Live Scan3D: A Fast and Inexpensive 3D Data Acquisition System for Multiple Kinect v2 Sensors," in *IEEE 3DV*, 2015.

[6] Vahid Soleimani, Majid Mirmehdi, Dima Damen, Sion Hannuna, and Massimo Camplani, "3d data acquisition and registration using two opposing kinects," in *IEEE 3DV*, 2016.

[7] Dimitrios S Alexiadis, Anargyros Chatzitofis, Nikolaos Zioulis, Olga Zoidi, Georgios Louizis, Dimitrios Zarpalas, and Petros Daras, "An integrated platform for live 3d human reconstruction and motion capturing," *IEEE TCSVT*, 2017.

[8] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros, "Efficient model-based 3d tracking of hand articulations using kinect," in *BMVC*, 2011.

[9] Zhou Ren, Jingjing Meng, Junsong Yuan, and Zhengyou Zhang, "Robust hand gesture recognition with kinect sensor," in *ACMMM*, 2011.

[10] Logan Jeya and John Zelek, "Kinectscenes: Robust real-time rgb-d fusion for animal behavioural monitoring," in *IEEE CRV*, 2016.

[11] Hassan Afzal, Djamila Aouada, David Font, Bruno Mirbach, and Björn Ottersten, "Rgb-d multi-view system calibration for full 3d scene reconstruction," in *IEE ICPR*. IEEE, 2014.

[12] Dimitrios Alexiadis, Dimitrios Zarpalas, and Petros Daras, "Fast and smooth 3d reconstruction using multiple rgb-depth sensors," in *VCIP*, 2014.

[13] Stephan Beck and Bernd Froehlich, "Volumetric calibration and registration of multiple rgbd-sensors into a joint coordinate system," in *IEEE 3DUI*, 2015.

[14] Stephan Beck, Andre Kunert, Alexander Kulik, and Bernd Froehlich, "Immersive group-to-group telepresence," *IEEE TVCG*, 2013.

[15] Andrew Maimone and Henry Fuchs, "Encumbrance-free telepresence system with real-time 3d capture and display using commodity depth cameras," in *IEEE ISMAR*, 2011.

[16] "POSIX-C Library for LTC," https://x42.github.io/libltc/.