SYNCHRONIZATION ISSUES IN SURGICAL TRAINING

Álvaro Marco, Héctor Gracia, Ángel Asensio, Carlos Guallart
José Ignacio Artigas and Roberto Casas
Instituto de Investigación e Ingeniería de Aragón (TecnoDiscap group), Universidad de Zaragoza
María de Luna 1, Zaragoza, España +34976762627
amarco@unizar.es, hgracia@unizar.es, aasensio@unizar.es, carlosguallart@gmail.com
jiartigas@unizar.es, rcasas@unizar.es

Keywords: Synchronization, Training systems, Wireless sensor networks, Video.

Abstract: Surgical training systems allow novel surgeons to acquire the required skills to successfully carry out an operation without harming a real patient. These systems emulate the situation of a real operation, replicating the information gathered by sensors, movements of the surgeon, patient response, etc. All this information must be synchronized to provide an experience to the novel surgeon as closest to reality as possible. A special case of information synchronization is when using video images from the operation. In this paper, we analyze these synchronization issues —video, movements, sensors, etc. — and show a particular case that bring all together: an endoscopic video-surgery learning system.

1 INTRODUCTION

Endoscopy or tele-surgery, are medicine techniques that require a special skill from surgeons, as they have not a direct vision of what they are doing. Learning these techniques is supported by training systems that allow the surgeon to acquire the required skills to successfully carry out an operation without harming a patient.

(Ballaro et al. 1999) propound a training system where the surgeon manipulates synthetic images of a prostate. (Gomes et al. 1999) and (Kumar et al. 2002) improve it by providing a tactile feedback of the manipulation with an artificial prostate. (Chen and Marcus, 1998) offers also a feedback with an adapted resectoscope, analyzing the anatomic tissues virtually touched and generating the opposing force.

To properly emulate real situations where the surgeon can be involved, it is necessary to collect a lot of data from different sensors in real operations. Besides that, surgical training is often supported by use of pre-recorded video images (Gambadauro and Magos, 2007).

Although most of training systems offering haptic feedback use virtual simulations instead of real video, augmented reality systems combining these videos with graphics images systems provide a better simulation environment (Botden et al. 2007). Real videos can be used for training with motorized mock-ups driving surgeon's hands while playing video-images of the operation, reproducing the movements displayed in the video. Obviously, it is needed that the information used to drive them to a position is synchronized with the related frames of the video.

In this paper, we discuss some synchronization issues we have faced during the development of a video-surgery learning utility, which implements also a wireless-synchronization between classroom workbenches.

Next section outlines the learning utility and its synchronization issues, which are discussed in sections three and four. Section five details synchronization integration in the classroom and its operation, and finally, conclusions are presented.

2 VIDEO-SURGERY LEARNING TOOL

Our application consists on a classroom for training surgeons on prostatic surgery, whose system layout is presented in figure 1, and consists of several mock-ups. All the mock-ups are wireless connected, which allows an easy deployment and configuration of the classroom.

One of them —master—, includes a motion monitoring system which extracts all the movements the instructor surgeon makes; this one is called sensing mock-up. The others —slaves—, transmit these previously recorded movements to the practice surgeon so he can first learn the instructor movements and, after that, be evaluated by comparing his movements with the ones made by the instructor; this ones are called motor mock-up.

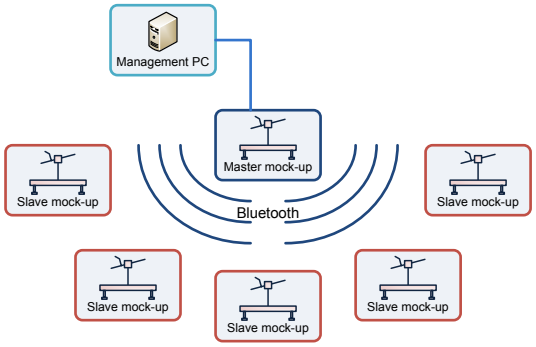

Figure 1: Video-surgery classroom. Master mock-up is video-synchronized and slave mock-ups are wireless-synchronized.

In order to get a real-like feedback, a big video display is used. At this moment, it seems clear there is heavy need of synchronization for this system, so video and captured instructor movements are correctly recorded and, this way, the learning and evaluation processes are synchronized with the video displayed.

It is also necessary to propagate this synchronization through the different motor mock-ups so all of them reproduce the movements at the same time the action is being taken at the video they see.

In short, we can group synchronization in two terms:

a) *Video-synchronization*, between the pre-recorded images and the surgeon's movements.
b) *Wireless-synchronization* between the mock-ups.

In the next sections, these synchronization issues are discussed in deep.

3 VIDEO SYNCHRONIZATION

In this case, a little misalignment between images and movement —below several milliseconds— is admissible, but playing must be fluid and without cuts that will hinder learning. If the training system core runs in a PC, a possibility for synchronizing surgeon's movements with video is to play the video in the PC, and to stream the movement information to the mock-up —through a serial port, by TCP/IP…—, but common multi-task Operating Systems (OS) can lead to cuts or undesired delays, so a real-time OS will be needed to do that.

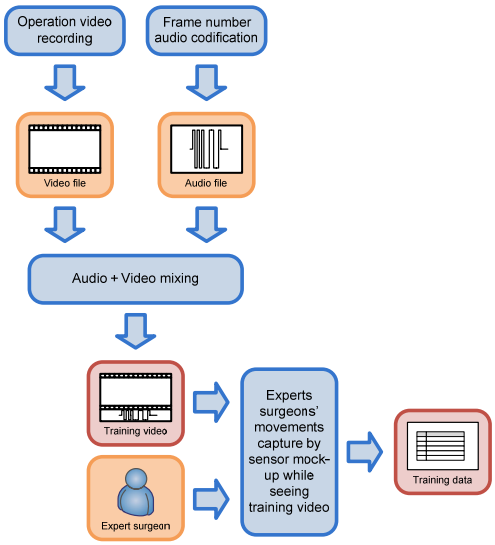

Figure 2: Block diagram for video synchronization. Expert surgeon's movements are captured and time-stamped with the corresponding frame number of the operation video.

To properly synchronize the video and the data, we decide to integrate the data directly into the video stream. AVI file formats allow integrating multiple streams into the same file (e.g. the video, audio, subtitles, chapters… of a movie). A possibility is to integrate a data stream with the movement information like a "subtitle", but we need also to extract that information and to stream it to the mock-ups, outside the PC, which can lead to an "asynchronous playing". The easiest way to avoid that is recurring to the audio channel. Audio is send by the OS to the sound card, and is synchronized with the video, so if we code our data in the audio track, the mock-ups can listen to the sound card and get the data synchronously with the video. Moreover, we could record the file in a DVD and play the video on a DVD player without the need of a PC.

In addition to that, there is another big issue about how to compound the operation information and the video images to get a synchronized training data set. If all the information of the operation is obtained "online", while the operation is carried out, as all the sensors must be synchronized, the training data set can be generated directly.

However, if the information related to the surgeon's movements cannot be real-time acquired,

an offline synchronization is required. The way to do this is that an experimented surgeon, while seeing the video images, replicates the movements over a sensor mock-up. That mock-up captures the movement information, and synchronizes it with the video.

Again, streaming the movement data to or from a PC can lead to a bad synchronization. The ideal solution would be to directly record the movement data on the audio track of the video, but this is not trivial. Fortunately, there is an easiest method. Strictly, we don't need to extract the data directly from the video, and it's enough by knowing the timestamp of the video, or the frame number.

For the movement data capture, we have coded the frame number of the video in the audio track, and the sensor mock-up listen to the video the expert surgeon is viewing. When the capture starts, the surgeon reproduces the movements displayed on the operation, and the mock-up store them together with the frame numbers listened. This way, we obtain a training data set where each frame of the video has synchronized information relative to the movements of the expert surgeon.

3.1 Frame Number Codification

To code the frame number into the audio track, we have generated an audio wave with a Manchester code, suitable for binary data transmissions. The data rate will be constrained by the audio characteristics and the sound card hardware. With a sampling frequency of 44.100 Hz, and using two samples to codify each bit (one sample for both high and low part of the wave), will give us a maximum data rate of 22.050 bps, but resulting sound wave cannot be correctly played by every sound card. A safer ratio of six samples per bit allows 7.350 bps, enough to codify more than 200 frame numbers per second with 32 bits resolution.

Frame number codes must be placed in the audio wave at the exact moment the frame is displayed in the video, which can be easily done with any video editing tool. To prevent errors when decoding the audio wave, a hamming code of distance three is also applied to the frame number.

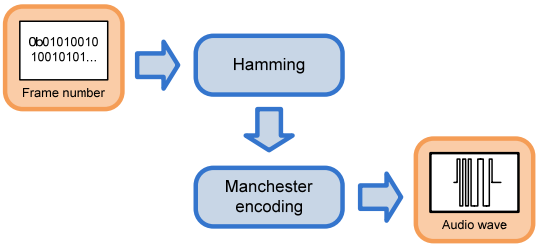

Figure 3: Frame number codification.

4 WIRELESS COMMUNICATION AND SYNCHRONIZATION

Once we have established synchronization between video and movements on the master mock-up, this synchronization must be propagated to the slave mock-ups, so we need that they are wireless synchronized. Every mock-up calculates its local time basing on its own oscillator, and these timings tend to diverge one from another. This is caused by the lack of precision on their oscillators, as there can be errors from 20 ppm to 100 ppm. The more time they keep running their clocks free, the bigger the misalignment will be.

This is a familiar matter on wireless sensor networks. Creating a common temporal reference using wireless communication capabilities has been widely studied keeping in mind the energy, cost and size limitations of the devices used in wireless sensor networks (Sivrikaya and Yener, 2004). The regular clock corrections needed to keep wireless networks synchronized are usually performed by exchanging reference messages time-stamped with the reference time. The more accurate that timestamp is the higher accuracy the synchronization achieves. Some protocols that achieve high synchronization accuracy with a reduced traffic load are the TMSP —Timing-sync Protocol for Sensor Networks— (Ganeriwal et al., 2003) or the FTSP —Flooding Time Synchronization Protocol— (Maróti et al., 2004). Their main advantage is that they can gain access to the MAC layer, so they can precisely timestamp messages when they pass through the lower layers.

According to the time-analysis performed by Maróti et al, the most problematic delays when transmitting messages over a wireless link are those from the send, receive and access processes (see figure 4). Besides bigger than propagation time, they are not deterministic, so they have a big influence on synchronization accuracy.

This way, methods which can access to MAC layer and precisely timestamp messages, such as TMSP or FTSP, can achieve a high accuracy. However, the use of standard wireless hardware such as ZigBee or Bluetooth —as in our case— to ease deployment, block access to lower layers, preventing from a precise timestamp.

Figure 4: Times on sending, accessing, propagating and receiving reference messages. Propagation time is negligible versus send, access or reception times.

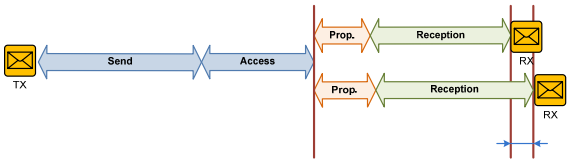

Figure 5: Misalignment on reception time. The non deterministic times on the sender side are eliminated.

In order to reduce as much as possible the uncertainties in this process, a receiver-receiver synchronization scheme is used. There is a common time reference for every member of the network, which is propagated through all the mock-ups using a broadcast message sent by the network coordinator (whose local hour is taken as the global time). After that, each mock-up receives this global hour and thus, can correct its own clock with the just received information.

With this receiver-receiver message synchronization method, medium access time variation is avoided (fig.5), so the biggest part of the non-deterministic error is eliminated. This method compares the local hours when the master mock-up and the different slaves receive the same message, so they can refer their local time to the master global time and hence, correct it. This is possible, because the uncertainty associated with the time involved in sending the message affects the same way both receivers, and so, doesn't have influence on the overall timing error. The propagation can be assumed as equal for the two nodes (because the distance is not significant enough to cause a measurable time difference in the propagation of a radio signal).

Thus, synchronize the clock of the mock-up i implies estimate and compensate its clock skew s_i and offset k_i. The most used procedure to perform these adjustments is broadly described in literature (Sivrikaya et al. 2004, Maróti et al. 2004, Elson et al. 2002, Cox et al. 2005). There is a reference clock which all the mock-ups will be synchronized to (t_r). A sync-point is defined as a pair of timestamps collected at the same time t_k in the reference node and in the node that want to be synchronized: $\{t_i^k, t_r^k\}$. Once each mock-up stores several sync-points at different instants, the offset (k_i*) and slope (s_i*) differences with the reference are calculated using linear regression:

$$s_i^* = \frac{\sum_k \left(t_r^k - \bar{t}_r\right)\left(t_i^k - \bar{t}_i\right)}{\sum_k \left(t_r^k - \bar{t}_r\right)^2}, \qquad (1)$$

$$k_i^* = \bar{t}_i - s_i^* \bar{t}_r .$$

This way, every node can estimate the global time (tr*) from its local clock:

$$t_r^* = \frac{t_i - k_i^*}{s_i^*}. \qquad (2)$$

Table 1 shows the results obtained when using the below described process to correct local times using Bluetooth and ZigBee technologies compared to other synchronization methods.

By these resynchronizations, clocks can be kept with a misalignment considerably lower than the precision required to cover our timing correction goals. The response time of the mock-up kinematics is considerably bigger than 1 ms, so achieving this misalignment between the different mock-up clocks is more than enough for this application.

Table 1: Misalignment on reception time. Results obtained by using different synchronization methods.

Sync Method	Average	Worst case
Sender – Receiver		
Zigbee (Motes-2.4GHz) *[Cox 2005]*	14.9 µs	61.0 µs
TPSN (Motes-916MHz) *[Ganeriwal 2003]*	16.9 µs	44.0 µs
FTSP (Motes-433MHz) *[Maróti 2004]*	1.4 µs	4.2 µs
Receiver – Receiver		
RBS (Motes) *[Elson 2002]*	29.1 µs	93.0 µs
RBS (Bluetooth)	4.5 µs	18.0 µs
RBS (ZigBee)	22.2 µs	52.0 µs

5 CLASSROOM INTEGRATION

Once we know how to perform video-synchronization and wireless-synchronization, it is time to integrate them into the classroom. In the layout shown in figure 1, the sensor mock-up — master— is connected to the PC, which controls video playing. The mock-ups form a Bluetooth piconet that allows communicate them and keep them synchronized with an absolute error below 1 millisecond, small enough for the classroom requirements.

For synchronizing the data with the video, we have generated an audio file in Matlab, containing a frame number every 10 milliseconds, and mixed it with the video containing the images of the operation for training. In the configuration of the mock-ups, it is possible to set an amount of frame

numbers to drop out when decoding audio, reducing the effective frame rate.

There are two basic operating modes: capture and playing. Combinations of them allow recording movements of the expert surgeon and reproducing them for training the novel surgeon, capturing movements of the novel surgeon for evaluating, guiding, etc.

In the capture mode, the surgeon reproduces the movements related to the video sequence. The sensor mock-up receives the audio in the training video, and decodes the frame numbers, which are periodically distributed. When a frame number is decoded, the mock-up captures the values of the position encoders and stores them together with the frame number. When the capture ends, the mock-up sends the data to the PC, where can be saved as the training data file related to the video, or be evaluated with a previously stored data.

In the playing mode, all the motor mock-ups previously store the training data, and wait for the start command from the sensor mock-up. When the PC starts video, the sensor mock-up decode the first frame number, and timestamp it with the global hour. Then, it broadcasts that information to the motor mock-ups, which can compute the timestamp for the next frames, and synchronize playing.

The sensor mock-up periodically broadcast frame-time pairs for prevent errors, and when there is an unexpected value in the frame number sequence, which means a change in the video playing (pausing the video for an explanation, advance the video, looping some technique… etc.).

6 CONCLUSIONS

In this paper we have discussed two different synchronization issues we have faced during the development of a video-surgery learning utility.

Video synchronization was performed by a cost effective and simple method recurring to the audio channel. It allows accurate synchronization without the need of a complex system. Even it is possible to eliminate a PC by using a dedicated video player and controlling playing from the sensor mock-up.

Wireless synchronization between mock-ups was also analyzed using a similar criterion of wireless sensors networks. We use a synchronization protocol over Bluetooth (we also tested ZigBee with similar results) that largely achieves our requirements. The method avoids accessing the lower layers of the protocol while performing similar accuracy as others that use the MAC layer.

With the strategies described in this article, we conclude in a surgical training classroom in which images displayed will be correctly synchronized with the sensor information and the mock-up movements, so the novel surgeons can acquire the needed skills in a real-like environment without harming any patient. This is obtained at low cost by using off-the-shelf components to build up this surgical classroom.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish Ministry of Science and Technology under CICYT project numbers TIC2003-07766 and TIN2006-15617-C03-02.

REFERENCES

Ballaro, A., et al., 1999. A computer generated interactive transurethral prostatic resection simulator. In *Journal of Urology*, 162 (5), pp. 1633-1635.

Gomes, M.P.S.F., et al., 1999. A computer-assisted training/monitoring system for TURP structure and design. In *IEEE Trans. Information Technology in Biomedicine,* 3 (4), pp. 242-251.

Kumar, P.V.S., et al., 2002. A computer assisted surgical trainer for transurethral resection of the prostate. In *Journal of Urology*, 168 (5), pp. 2111-2114.

Chen, E., Marcus, B., 1998. Force feedback for surgical simulation. In *Proceedings of the IEEE*, pp. 524-530.

Gambadauro, P., Magos, A., 2007. Digital video technology and surgical training. In *European Clinics in Obstetrics and Gynaecology*, 3 (1), pp. 31-34.

Botden, S.M.B.I., et al., 2007. Augmented versus Virtual Reality Laparoscopic Simulation: What Is the Difference? In *World Journal of Surgery*, 31 (4), pp. 764-772.

Sivrikaya, F., Yener, B., 2004. Time Synchronization in Sensor Networks: A Survey. In *IEEE Network*. 18 (4), pp. 45-50.

Ganeriwal, S., et al., 2003. Timing-sync protocol for sensor networks. In *Proceedings of the 1st International conference on Embedded networked sensor systems*, pp. 138-149.

Maróti, M., et al., 2004. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pp. 39-49.

Elson, J. et al., 2002. Fine-Grained Time Synchronization using Reference Broadcasts. In *Proc. 5th Symp. Op. Sys. Design and Implementation*, pp. 147–163.

Cox, D. et al., 2005. Time Synchronization for ZigBee Networks. In *Proceedings of the 37th Southeastern Symposium on System Theory (SSST2005)*, pp. 135–138.