A LED-Based IR/RGB End-to-End Latency Measurement Device

Markus Billeter* Chalmers University
Gerhard Röthlin† Disney Research
Jan Wezel ‡ Disney Research
Daisuke Iwai § Osaka University
Anselm Grundhöfer ¶ Disney Research

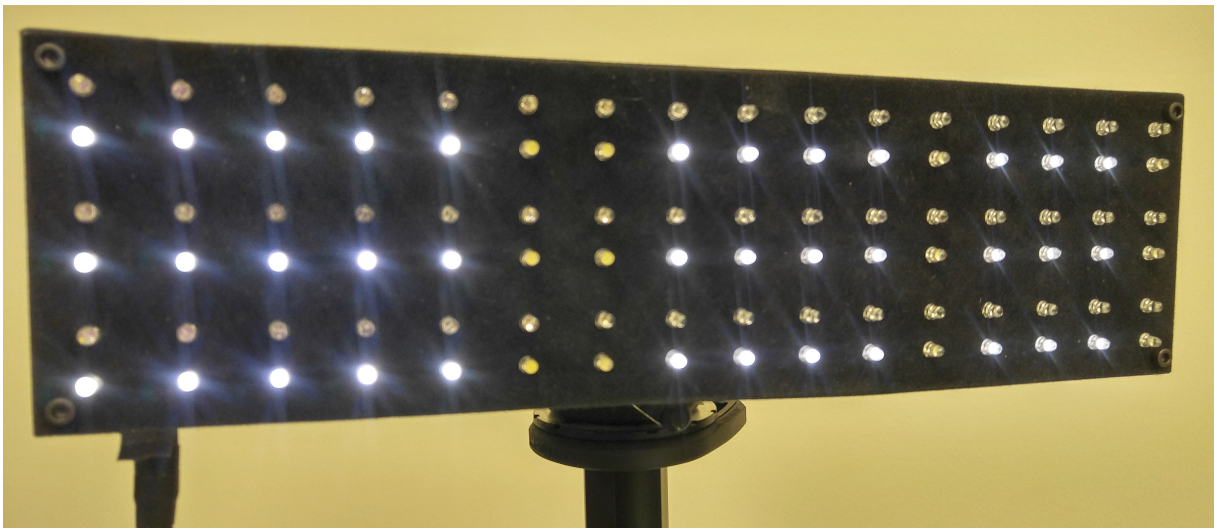

Figure 1: The developed prototypical latency measurement device. It consists of three rows of 16 white and IR LED pairs (lower row: visible light, upper row: IR) emitting time stamps encoded as gray codes. Encoding and LED control is carried out using an Arduino micro controller. Decoding is done by an external camera.

ABSTRACT

Achieving a minimal latency within augmented reality (AR) systems is one of the most important factors to achieve a convincing visual impression. It is even more crucial for non-video augmentations such as dynamic projection mappings because in that case the superimposed imagery has to exactly match the dynamic real surface, which obviously cannot be directly influenced or delayed in its movement. In those cases, the inevitable latency is usually compensated for using prediction and extrapolation operations, which require accurate information about the occurring overall latency to exactly predict to the right time frame for the augmentation. Different strategies have been applied to accurately compute this latency. Since some of these AR systems operate within different spectral bands for input and output, it is not possible to apply latency measurement methods encoding time stamps directly into the presented output images as these might not be sensed by used input device.

We present a generic latency measurement device which can be used to accurately measure the overall end-to-end latency of camera-based AR systems with an accuracy below one millisecond. It comprises a LED-based time stamp generator displaying the time as a gray code on spatially and spectrally multiple locations. It is controlled by a micro-controller and sensed by an external camera device observing the output display as well as the LED device at the same time.

Index Terms: H.5.2 [HCI]: User Interfaces—Benchmarking;

1 INTRODUCTION

Calculating the overall latency of AR systems is an important task when gathering the required information to asses the overall system performance, especially in situations where real-time feedback is required. This is in particular important for spatial augmentations such as dynamic projector-camera (procams) systems since, as opposed to video-see-through augmentations, the real world impression obviously cannot be delayed to match the augmentation. In these systems often the camera is not capturing the same region of the electromagnetic spectrum which is used for projection since then the projection would interfere with the surface information. Examples for that are IR based tracking systems.

Optical see-through AR systems in general require an extremely low overall latency to guarantee a convincing impression to the user. Ng et al. [8] show that human perception cannot on average perceive a delay if the end-to-end latency is below $6.04ms$ with a standard deviation of $4.33ms$. If the delay is significantly higher than this time delta, the augmentation appears to be delayed which greatly reduces the visual quality as well as the user's performance in accomplishing specific tasks.

Therefore, it is important to use sophisticated prediction methods to accurately estimate what exactly has to be rendered to compensate for the inevitable latency. Adjusting the parameters of such predictors, however, requires the accurate knowledge of the overall end-to-end latency of the system.

In this paper we propose a generic latency measurement method, which uses a configurable LED-based time stamp generator to enable an accurate latency measurement (cf. Figure 1) without interfering

*e-mail: markus@newq.net
†e-mail:gerhard@disneyresearch.com
‡e-mail:jan.wezel@disneyresearch.com
§e-mail:daisuke.iwai@sys.es.osaka-u.ac.jp
¶e-mail:anselm@disneyresearch.com

with the system's internal processing. It can be tuned to specific wavelengths, and temporal accuracies and has been proven to generate accurate measurements within different configurations.

2 Related Work

Since optical see-through and spatial augmentations have the purpose to directly superimpose the real world with additional, computer generated content within interactive frame rates, it is crucial to display them with minimal, hopefully imperceptible latency. This also guarantees a high visual quality leading to a significantly improved immersion, but also guarantees that the users are able to accomplish specific tasks with the required accuracy.

The overall latency of such systems stems from several sources [9]:

- Camera exposure, readout and transmission
- Image processing
- Output image generation
- Synchronization between image generation and display device
- Internal delay inside of the display device

While some of these steps can be optimized by using more sophisticated algorithms and faster processing hardware, others, such as the exposure time of the camera cannot be arbitrarily reduced due to physical limitations. Thus, there will always be an unavoidable latency in the system, which needs to be taken into account when rendering the augmentation. Unlike video–see-through AR applications, delaying of the input real-world data as by Bajura and Neumann [2] is obviously not possible for optical see-through applications. In other words, not only the relative latency between different devices [4] has to be taken care of, but also the absolute latency of the system has to be exactly known to enable an accurate prediction of the augmentation [1, 5, 7]. Experimental methods for reducing the latter were presented by several researchers [6, 11, 12, 13] using specialized hardware components. Both methods are not able to be used with standard camera-display processing pipelines.

End-to-end latency within camera-based AR systems has already been measured before by either using a pulse generator driving an LED which is sensed by an oscilloscope [4], or by displaying time-encoding blobs on a display [10]. Although these methods are able to quite accurately measure the end-to-end latency of such systems, they partially require specialized hardware such as oscilloscopes, which require further processing for automated measurements or can only work when spectral bands for input (camera) and output (display) devices overlap. We present a hardware device which overcomes the latter limitations and only requires simple image processing methods to calculate the latency with a highly precise accuracy.

3 Method

To overcome the limitation that display and camera need to share the same spectral bands, we developed an external hardware device for latency measurements. The device encodes pre-defined time stamps as gray code patterns displayed via LEDs which are emitting light in the required spectral bands. In our case this is a pair of LEDs, one emitting in the visible spectrum and another one with a wavelength of 850*nm* (near IR). An image of the developed prototype hardware is shown in Figure 1.

The purpose of the device (*L* in Figure 2) is to emit time stamps with the desired temporal resolution in the spectral band visible to the input camera *I* as well as in the spectral range of the output device *O* of the system whose latency is to be measured. In one example, the LEDs at 850*nm* are visible to an IR tracking camera,

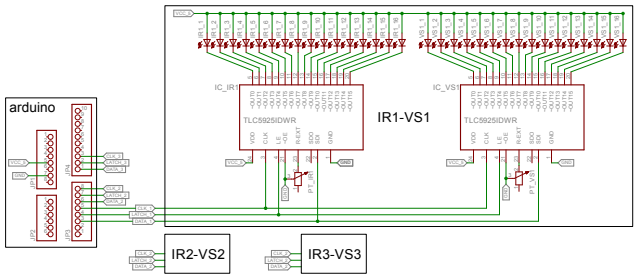

Figure 3: Board Schematic

and the visible output overlaps with the output spectrum of standard RGB-based projection/display devices.

This device is placed within the view-frustum of *I*. Since the small areas of common LEDs approximate point light sources, the device is not required to be placed exactly within the focal plane of *I*. During measurement, the captured image of *L* is cropped to the region of interest which contains the device data and displayed at the end of the processing pipeline at a freely defined area on *O*.

An external camera *E* is used to capture the displayed image of the device and, at the same time, the actual device displaying the current time stamp. Out of these captured images, the time codes and thereby the latency can be automatically decoded using homography warps, blob detection and standard gray code decoding (Section 3.3).

3.1 Prototype

The presented method was prototypically realized as a small hardware device and encoding/decoding algorithms were implemented as described in the following.

3.1.1 Hardware

The LED Clock hardware is built around an Arduino UNO, using a 16 MHz Atmel ATmega 328P. It provides more than sufficient clock resolution for our requirements. The Arduino communicates with three pairs of TLC5925 16-bit shift registers using three GPIO pins each. Each shift register drives one line of 16 LED-pairs (see Figure 1). Since one infrared and one visible LED line form a combined logic line displaying the same code, they do not have to be controlled separately.

Two GPIO Pins are used via standard Arduino `digitalWrite` and `shiftOut` API calls, carrying the data and a per-bit write clock. Once 16 bits are written into the register, the third GPIO Pin is used to trigger the register write, updating all LEDs at the same time. This, in combination to the grey code property of only ever changing the state of one bit per step, ensures that no invalid states are visible.

Potentiometers connected to the shift registers control the current provided to each set of LEDs, allowing fine-tuning of the brightness according to the requirements of the IR and visible light cameras. Obviously the spectral ranges of the LEDs could also be varied depending on the actual setup.

3.1.2 Software

For displaying the time stamps as binary on/off combinations of LEDs, we encoded them using a standard "Binary Reflected Gray Code". Besides the fact that it is easy to generate, compared to direct binary encoding, this gray code encoding has the significant advantage that it guarantees that only a single bit changes between successive codes, which makes it much more insensitive against errors [3].

In our current prototype we are using three LED lines instead of one since we want to ensure that even a camera that captures with at

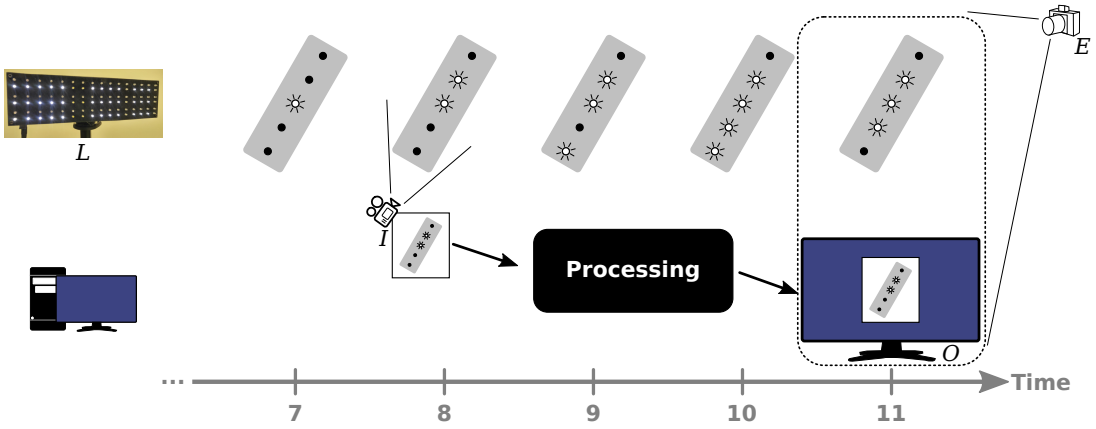

Figure 2: Overview of the latency measurement setup. The top row shows state of the hardware prototype (for simplicity, we show only a single row of LEDs). The bottom row represents the system whose latency we wish to measure. The system captures an input image that includes the clock from the top row, performs its processing and displays the captured clock image together with its results. Both the displayed clock image and the live clock are captured by an external camera. The external camera's image is then examined – the systems latency is equal to the difference of two times encoded in the image (in this case $11-8=3$ ticks).

least a third of the clock frequency still records valid images. With a delta of $750\mu s$ between consecutive clock ticks, this allows us to use an observing camera (E in Figure 2) with an exposure time of around $2ms$ or less, which fits well to our measurement hardware. Obviously this configuration can easily be adopted for other needs.

Listing 1 shows the LED control algorithm we implemented on an Arduino UNO for our prototype.

3.2 External Recording

To acquire the time difference between the captured and displayed image of the LED device and the current time, an external camera is used to capture both time stamps, i.e. the real physical device as well as the projected image of it, within the same image. This can be an arbitrary camera, but should be able to be configured to an exposure time similar to the frame rate of the display device. If that is not the case, the displayed image of the captured LED image has to be spatially switched similar to the work of Sielhorst et al. [10] in each consecutive frame n, such that the display's frame rate times n exceeds the camera's exposure time.

3.3 Decoding Software

The decoding process is implemented as follows. First, we compute the maximum image by taking the maximum intensity value at each pixel over the whole sequence. Because the LEDs of L (raw LEDs) and the projected dots of the captured LEDs (re-projected LEDs) are all visible in the maximum image, we manually assign the four corners of each region of raw and re-projected LEDs. The positions of the corners are used to rectify the LED regions in each captured image by applying the homography transformation. Second, at each frame, we measure the intensity value of each LED in the rectified image, in which the position of the LED is pre-defined. To achieve a measurement robust to camera noise, we average the intensities over a small region around the LED. Then, we decode the gray code of each LED line by applying a simple thresholding process to the averaged intensities, where different threshold values are applied between the raw and re-projected LEDs. The decoded gray codes are then converted to decimal values representing time. Finally, we calculate the delay as the difference between the sum of decoded time values of raw LED lines and those of re-projected LED lines.

Algorithm 1 Pseudo code for encoding the time stamps and driving the LEDs.

```
rows ← 3
codeBits ← 16
quantum ← 750                                   ▷ microseconds

procedure BINARYTOGRAYCODE(binary)
    grayCode ← binary ⊕ (binary/2)              ▷ ⊕ is XOR
    return grayCode
end procedure

procedure DISPLAYGRAYCODE(row, code)
    pins ← outputPins_row
    DIGITALWRITE(pins.storageClock, LOW)
    DIGITALWRITE(pins.shiftClock, LOW)
    for b ← 0, codeBits/8 do
        codeByte ← code/2^(b*8) mod 256          ▷ extract byte b
        SHIFTOUT(pins.data, pins.shiftClock, codeByte)
    end for
    DIGITALWRITE(pins.storageClock, HIGH)
end procedure

procedure MAIN
    row ← 0
    codeIdx ← []
    for ever do
        delay ← quantum − MICROS mod quantum
        DELAYMICROSECONDS(delay)
        code ← BINARYTOGRAYCODE(codeIdx_row)
        DISPLAYGRAYCODE(row, code)
        codeIdx_row ← codeIdx_row + 1 mod 2^codeBits
        row ← (row + 1) mod rows
    end for
end procedure
```

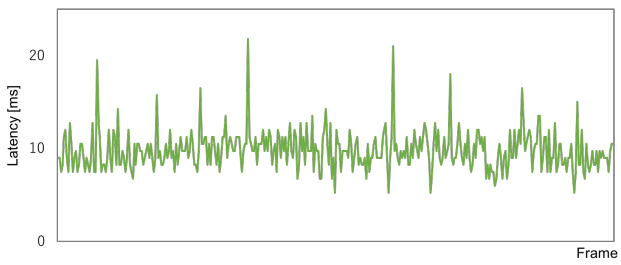

Figure 4: Measured overall system latencies of the proposed method (setup #1)

3.4 Mismatches in frame rates

It is possible that the frame rate of the input and output devices mismatches, or that the processing requires more time than a single output frame. In such cases, one input image with a certain time stamp may be displayed for multiple output frames. We can indicate this by adding (in software) markers for each output frame the image is being displayed. Our markers take the shape of small red dots.

The decoding software can detect these markers and determine the age (in output frames) of the time stamp. This enables several additional options for analysis. On one hand, a minimal latency for the system can be determined by only considering time stamps with an age of one – this is the latency the system could achieve if neither the input capture rate nor the processing rate were a bottle-neck. Considering all time stamps regardless of their age gives the average observed latency.

4 Evaluation

To evaluate accuracy as well the flexibility of our proposed latency measurement device, we tested our hardware prototype with two different setups.

1. A high-speed projector-camera system consisting of an Allied Vision Bonito IR camera capturing at $1300Hz$ and a customized Christie Mirage 4K35 3-chip DLP projector running at $480Hz$.

2. An LCD running at $60Hz$ in combination with an USB 3.0 Ximea xIQ camera set to its maximum frame rate and a shutter time of $1ms$

To externally record the LED device as well as the displayed image of it, we used a Sony RX 100 IV camera capturing images at a frame rate of $1000Hz$ for a sequence of 2 seconds. After recording these images, they were processed as described in Section 3.3.

For the first system, the average system latency was measured on average with $9.8ms$ with a standard deviation of $2.1ms$ (cf. Fig. 4).

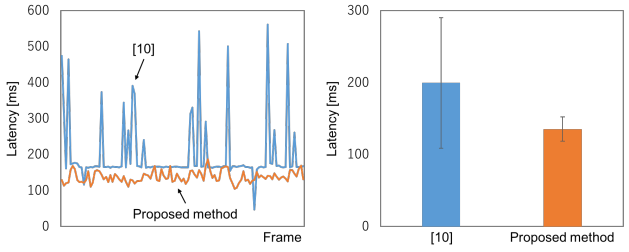

Figure 5: Measured overall system latencies of a previous method [10] and the proposed method: (left) raw latency values, and (right) averages and standard deviations. (setup #2)

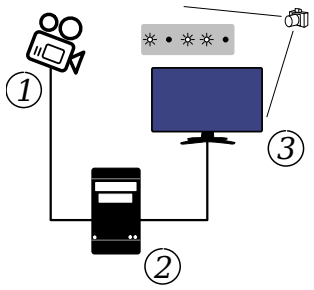

Figure 6: Our system measures the latency of the path $1 \rightarrow 2 \rightarrow 3$ using an external camera that observes the clock and display device simultaneously. In comparison, the system by Sielhorst et al. [10] measures the round trip latency of $2 \rightarrow 3 \rightarrow 1 \rightarrow 2$.

As these numbers indicate, this system was extremely tuned for minimal latencies. Another, more generic system, was tested in the second evaluation (Fig. 5). Here the overall system latency was measured with $135.2ms$ with a standard deviation of $16.8ms$. As it can be seen, the used USB3 interface of the camera seems to add a significant delay to the system. Since the latter system fully operates in the visible spectrum, we used it to compare our method to the one presented in [10] with which we measured an average latency of $199.3ms$ with a standard deviation of $90.4ms$ using exactly the same hardware and software combination. As it can be seen in the diagram, severe outliers tend to occur with their method, even after a careful adjustment of the camera. Because of that, our method is able to generate measurements with a significantly lower standard deviation than the related work. But even when ignoring the outliers, the latency measured by our system is on average lower compared to one by the other system [10]. This is expected, since our system measures the latency from the input camera (labeled 1 in Figure 6) to the display (labeled 3 in Figure 6), whereas the other system measures the round trip from the computer (labeled 2 in Figure 6) to the display, the input camera back to the computer. The latency measured by our system thus better represents the latency that we seek to measure.

5 Conclusion

In this paper we presented a LED-based time stamp device which enables an accurate latency measurement within camera-based AR systems. The system can be easily configured for the given accuracy requirements and spectral sensitivities ranging from the ultraviolet up to the mid infrared thermal spectrum as long as LEDs are available for the desired range. Since the latency is measured by an external camera, the only required system overhead is the read-out of the region in which the LED device is captured by the camera of the AR-system and the displaying of these pixels on the output device which can be arbitrary, for example a LCD screen, an OLED device or a projector. As shown in the evaluation, the system can easily be used within different environments in either overlapping or non-overlapping spectral bands. In the future we will further investigate its practical applicability to other spectral ranges.

References

[1] R. T. Azuma. *Predictive Tracking for Augmented Reality*. University of North Carolina at Chapel Hill, 1995.

[2] M. Bajura and U. Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–60, Sept. 1995.

[3] R. W. Doran. The gray code. *J. UCS*, 13(11):1573–1597, 2007.

[4] M. C. Jacobs, M. A. Livingston, and A. State. Managing latency in complex augmented reality systems. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, I3D '97, pages 49–ff., 1997.

[5] J. Knibbe, H. Benko, and A. D. Wilson. Juggling the effects of latency: Software approaches to minimizing latency in dynamic projector-camera systems. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15 Adjunct, pages 93–94, 2015.

[6] P. Lincoln, A. Blate, M. Singh, T. Whitted, A. State, A. Lastra, and H. Fuchs. From motion to photons in 80 microseconds: Towards minimal latency for virtual and augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1367–1376, 2016.

[7] S. Miyafuji and H. Koike. Ballumiere: Real-time tracking and projection system for high-speed flying balls. In *SIGGRAPH Asia 2015 Emerging Technologies*, SA '15, pages 2:1–2:1, 2015.

[8] A. Ng, J. Lepinski, D. Wigdor, S. Sanders, and P. Dietz. Designing for low-latency direct-touch input. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, pages 453–464, 2012.

[9] D. Schmalstieg and T. Hollerer. *Augmented Reality: Principles and Practice*. Addison-Wesley Professional, 2016.

[10] T. Sielhorst, W. Sa, A. Khamene, F. Sauer, and N. Navab. Measurement of absolute latency for video see through augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–4, 2007.

[11] T. Sueishi, H. Oku, and M. Ishikawa. Robust high-speed tracking against illumination changes for dynamic projection mapping. In *2015 IEEE Virtual Reality (VR)*, pages 97–104, March 2015.

[12] Y. Watanabe, G. Narita, S. Tatsuno, T. Yuasa, K. Sumino, and M. Ishikawa. High-speed 8-bit image projector at 1,000 fps with 3 ms delay. In *Proceedings of The International Display Workshop*, pages 1064–1065, 2015.

[13] F. Zheng, T. Whitted, A. Lastra, P. Lincoln, A. State, A. Maimone, and H. Fuchs. Minimizing latency for augmented reality displays: Frames considered harmful. In *Mixed and Augmented Reality (ISMAR), 2014 IEEE International Symposium on*, pages 195–200, Sept 2014.