Graphics and Visual Computing 5 (2021) 200034

Contents lists available at ScienceDirect

Graphics and Visual Computing

journal homepage: www.elsevier.com/locate/gvc

Technical section

Robust marker-based projector–camera synchronization ☆

Vanessa Klein [a,*], Martin Edel [b], Marc Stamminger [a], Frank Bauer [a]

[a] Visual Computing, Friedrich-Alexander-University Erlangen-Nürnberg (FAU), Cauerstr. 11, Erlangen, 91058, Germany
[b] Independent Researcher, Germany

ARTICLE INFO

Article history:
Received 23 July 2021
Received in revised form 25 October 2021
Accepted 9 November 2021
Available online 15 November 2021

Keywords:
Projector–camera system
Software synchronization

ABSTRACT

Recording clean pictures of projected images requires the projector and camera to be synchronized. This task usually requires additional hardware or imposes major restrictions on the devices with software-based approaches, e.g., a specific frame rate of the camera. We present a novel software-based synchronization technique that supports projectors and cameras with different frame rates and at the same time tolerates camera frame drops. We focus on the special needs of LCD projectors and the effect of their liquid crystal response time on the projected image. By relying on visible marker detection we entirely refrain from taking time measurements, allowing for a robust and fast synchronization.

1. Introduction

Many applications that employ a projector also require a camera to observe the projection. For example, in active stereo vision the projector projects a known pattern onto a three-dimensional surface and the camera captures the deformed projection to reconstruct the surface geometry. Depending on the reconstruction method, the patterns thereby range from discrete black-white patterns to continuous grayscale gradients to color-coded patterns. Whereas in this example the only purpose of the projection image is to be recorded by the camera and it is thus tailored to its characteristics, in other applications the camera is not the main spectator. In the Spatial Augmented Reality (SAR) domain, projection mapping systems project arbitrary textures onto planar or three-dimensional surfaces to entertain or convey information to a user. In feedback-loop systems, such an application is complemented with a camera observing the user-oriented projection to report errors in the projection quality.

In any of these examples it is mandatory to project a sequence of images, take clean photos of them with a camera, and match the photos to the original images. However, without additional hardware to synchronize projector and camera, the devices run independently from each other and at possibly different frame rates. Since neither projecting an image nor capturing one is an instantaneous task, synchronized projection captures require adjusting the duration that a single image is projected. The shorter that duration is, the better the performance of the application.

Our contribution is a software-based synchronization between an LCD projector and a camera. The frame rates of the devices may differ but should be predominantly constant. Our technique specifically considers the slow reaction time of the LCD technology. We present multiple synchronization strategies — each suited for different application needs and robustness requirements, including a detection and support for camera frame drops. Wait durations are adaptively reduced to achieve fast results with hardware of which only the frame rates are known.

2. Related work

Synchronization between a projector and camera has been achieved in the past. Previous techniques are divided into two groups: hardware- and software-based methods.

In hardware-based approaches, an external device acts as the pulse generator which drives projector and camera. Zhang et al. [1] therefore disable the projector's timing signal and employ a microcontroller-based circuit as the trigger signal. In a similar fashion, Grundhöfer et al. [2] install customized electronics. Wissmann et al. [3] synchronize a DLP projector with the help of its two optical tracks that control the camera exposure time. Other hardware-based systems are presented in [4]–[5][6]–[7].

On the software side, Jaeggli et al. [8] present a synchronization algorithm for LCD projectors that measures delays in the setup as a preprocessing step. Although their approach is a good approximation of the constant latencies involved in their setup, correct photos of the projection cannot be guaranteed, as they do not elaborate the effects of the projector's liquid crystal response time and rely on a non-real-time system's time measurement precision. Petković et al. [9]–[10] indirectly measure the delay between projection and photo exposure start by back-calculating

☆ This article was recommended for publication by Kun Xu.

* Corresponding author.
E-mail addresses: vanessa.klein@fau.de (V. Klein), frank.bauer@fau.de (F. Bauer).

https://doi.org/10.1016/j.gvc.2021.200034

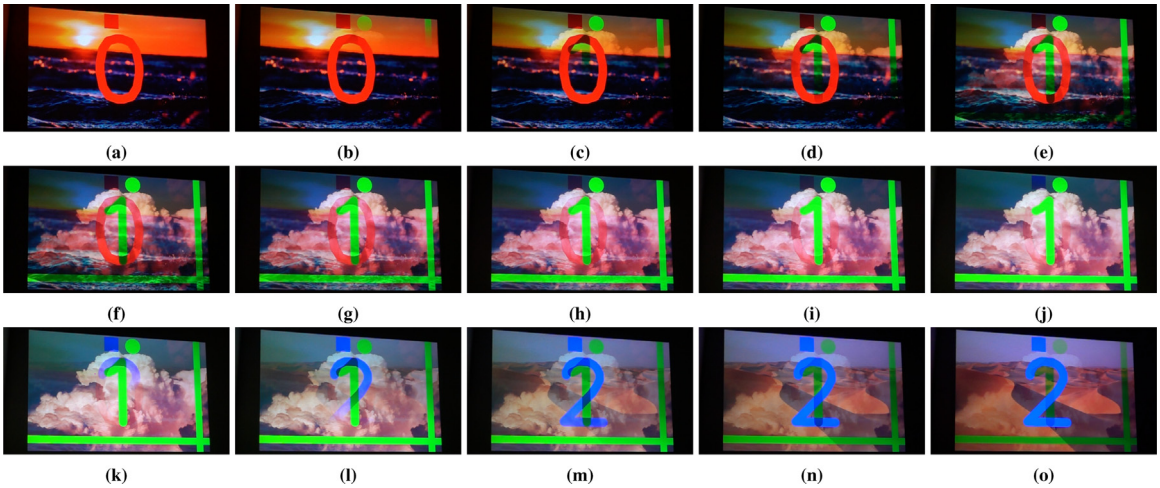


Fig. 1. Photos captured of a projected, numbered image sequence (sea, clouds, desert); projector: 60 FPS, camera: 1000 FPS, every other camera frame is depicted. Artifacts from projector frame 0 are still visible in projector frame 2 (k–o). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

linear factors from recorded gray values from a black/white calibration sequence. However, their approach implicitly expects an immediate transition from one projector frame to the next and is thus not applicable to the slow response time of LCD projectors. Their algorithm is also designed for cameras that can be triggered via software, not fixed-frame-rate cameras. El Asmi et al. [11] assume the same linear mixture model and expect projector and camera to share the same frame rate. In a related domain, Perli at al. [12] take synchronized photos of an LCD screen. Our algorithm is also based on marker detection, but we do not impose a frame-rate restriction on the camera. Altogether, their synchronization needs differ from ours in that their goal is the marker detection itself, while we aspire the best possible recording of an LCD display/projector in general. For the same reason we do not refer to related work from the field of unsynchronized captures, e.g., Moreno et al. [13].

To the best of our knowledge, this is the first software-based synchronization of projector and camera that supports unequal frame rates, considers the special needs of LCD projectors and guarantees robust results.

3. Hardware issues

Several hardware properties of LCD projectors and digital cameras complicate synchronization and have to be considered for a software-based approach. Fig. 1 depicts a numbered image sequence that is projected at 60 FPS and recorded with a 1000 FPS camera. The photos demonstrate that each projector frame is constructed from top to bottom (for this specific model) and, most importantly, that every photo after the first one is a mixture of at least two projector frames, even after the refresh has reached the bottom. Upon close inspection of Fig. 1(o) (third projected frame) one can even see traces from the first frame's red zero. These mixtures are a result of the hardware details of both the camera and the projector.

3.1. Camera properties

Digital cameras function by collecting light that falls on a sensor. The spatially different amounts of light are converted to digital signals and determine the brightness and color values of the pixels. During this light collection duration, the *exposure time*, the scene must remain as static as possible. Otherwise, the light from the changing scene reflects differently, mixes with the previous value, and as a result makes the photo blurry. For truly perfect photos of a projection, the projection image thus must not change during the camera's exposure time.

Aside from the inherent technological issues within digital cameras, our work focuses on cameras with a fixed frame rate, e.g., ordinary webcams. In contrast to system cameras, the user cannot control the exact moment when the photo is captured. Instead, the photos are recorded in a continuous fixed-frame-rate stream. However, although the camera might record its photos at a fixed frame rate, the application probably does not receive the photo stream as such. To transfer the recordings from the camera to the computer or a mobile device, the devices must be connected, e.g., by a USB cable. Each connection medium introduces a layer of *latency* that is presumably not constant across all frames. Measuring latencies, however, is problematic. We therefore fully abstain from time measurements to make our algorithm as broadly deployable as possible.

Finally, we have to acknowledge that even the best hardware can fail. Frame drops, i.e. missing frames, may occur as a result of problems from within the camera or simply because the frame-collecting thread was not scheduled fast enough by the operating system. We show in Section 6 that frame drops do indeed occur frequently with commodity hardware.

3.2. LCD projector properties

An LCD projector employs the Liquid Crystal Display (LCD) technology to assemble a single frame. Nowadays, an LCD projector usually has three LCD panels installed (3LCD) – one for each color channel (red, green, blue). Each LCD panel is composed as a grid, resembling the pixels, and it acts as a gateway for the projector's light source. By adjusting each grid cell's voltage, the liquid crystals (LC) in that cell twist by a voltage-dependent angle and thereby restrict the amount of light that can physically pass through. Similar to a slide show the panel blocks the light for dark pixels and passes through more light the brighter the pixels are. By superimposing the independently constructed red, green and blue images with a dichroic prism cube, each pixel contains all three colors and the projected image appears colored.

Although modern LCD panels are fast, they do not transition instantaneously from one frame to the next. To accurately measure the *response time*, the time it takes to transition from one start color value to another target value, one would need a detector that is ten times faster [14]. Clearly, such a device is not

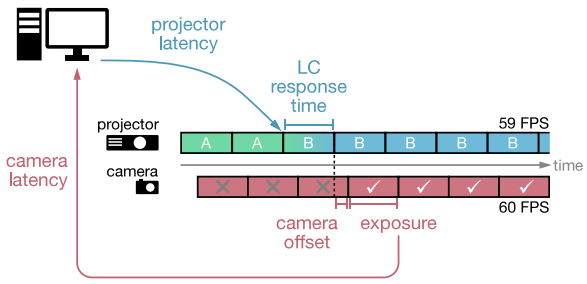


Fig. 2. Overview of all latencies that must be considered when projecting and capturing image B. In this example, the liquid crystal response time equals one full projector frame duration. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

readily available. Additionally, the response time depends significantly on the luminance start and target values, white↔black transitions usually being faster than mid-tone transitions [15]–[16]. This property makes it basically impossible to know in advance or measure the response time of a single pixel with certainty.

Techniques like Dynamic Capacitance Compensation (DCC) improve the response time by shortly applying more/less voltage than required. These improvements require that frames are buffered to precompute the optimal voltage and thus come at the cost of an increased *latency* between a frame being ready for projection and actually being projected.

Standardized by ISO 9241-305, the response time of a transition is the duration it takes for transitioning from 10% to 90% of its start and target luminance levels. In 2002, Suzuki et al. [17] report average response times between 10 ms to 40 ms, depending on the type of LCD panel and compensation method, with some transitions taking up to nearly 60 ms. Elze [15] reports response times between 5 ms to 17 ms in 2010. They explicitly state that "[t]he transition can exceed one frame". Elze and Tanner [16] later show that DCC is commonly used in modern panels, which results in average response times of less than 10 ms. Although we may expect even faster response times in future hardware, e.g., Chen et al. [18] propose a liquid crystal mixture with an average response time of 1.29 ms, LCD technology that is in use today is still limited to response times in the milliseconds range, as Fig. 1 demonstrates. With commodity projector and camera hardware also exhibiting frame rates in that range, the response time of liquid crystals must not be underestimated when synchronizing the devices.

4. Sequential Marker-based synchronization

To synchronize projector and camera in an entirely software-based manner, we consider the full duration it takes to project an image and take a photo of it. Besides the device-specific latencies, the combination of both devices introduces another unknown temporal gap between the start point of a projector frame and the start point of the respective camera frame, which we call the *camera offset*. If projector and camera share the same frame rate, the camera offset is constant, otherwise the offset changes with each frame. Thus, the duration of a single synchronized image capture task is summed up by the projector latency, liquid crystal response time, camera offset, camera exposure time and camera latency (see Fig. 2).

Even if we were to assume that the projector latency and camera latency are constant, reliably measuring all latencies with color-dependent liquid crystal response times and continuously changing camera offsets is a nearly impossible task. It is, however,

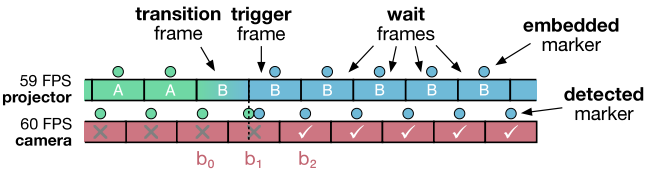


Fig. 3. To record image B, a transition frame is projected, followed by the trigger frame that contains a detectable marker. The projection is maintained with wait frames until the correct photo b_2 is received by the processing computer (which includes the camera latency).

not necessary to measure these latencies, if we reduce the synchronization problem to knowing *from the camera frame itself* to which projector frame it corresponds. We base our core algorithm on two assumptions:

1. Whenever parts of a projection image B are detectable in a camera frame, image B was (temporarily) being projected (voluntarily or involuntarily due to the LC response time afterglow) during the camera frame's exposure time.
2. Camera frames are delivered in their recording order (with arbitrary delays in between).

Under real-world conditions it is important to note that rule 1 cannot be reversed. Since no detector is perfect, not detecting traces of image B is not a guarantee that image B was not present during the exposure time.

By embedding a detectable marker (e.g., ArUco marker [19]) into the image, any camera photo of its projection is hereby labeled with that marker and a semantic connection between the projector and camera frame is established. Consequently, if the marker is detected in a camera frame, it is guaranteed that this and any subsequent camera frame capture the same image (until the projection itself changes). To take clean photos that are not affected by a previous image, the marker must be embedded only after the projector's LC response time for that image has passed. However, due to the exposure duration, the first camera frame that exhibits the marker may still include traces from the previous projection image before the projection was stable. It is thus the *second* camera image in that sequence that is the first correct recording of the projection.

Unfortunately it is impossible to predict the LC response time accurately without measuring any possible luminance transition beforehand. Elze and Tanner [16] even report different LC response times for the same transition for one of their tested LCD monitors. We therefore propose a pragmatic approach. Since a marker can be incorporated into the projection only at a regular frame update, one does not need to know the exact LC response time. Instead, the user merely has to decide after how many projector frames the projection is stable. With modern hardware that applies acceleration techniques it is reasonable to assume that a regular 60 FPS projector achieves its transitions within one frame. For out-dated hardware or safety-critical applications one should follow the slowest reported LC response time in literature or, depending on the application, decide how much damage slight traces of a previous image would do. In the following figures we depict the LC response time with a single *transition frame* but keep explanations and formulas universally valid with an arbitrary number of transition frames.

Fig. 3 visualizes the basic algorithm. For a new image B the projection starts with as many transition frames as required; no marker is included yet. After the transition phase a marker that is unique to B is embedded into the image which is then projected as the *trigger frame*. The projection is kept stable by continuing to project the image (*wait frames*) until it is successfully recorded. Eventually, a camera frame b_1 contains the detectable marker for

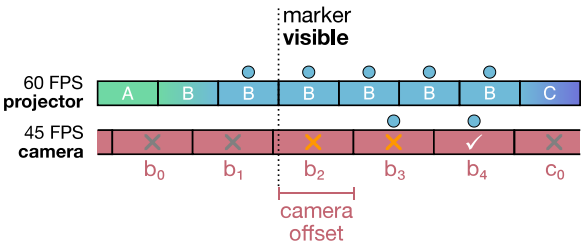


Fig. 4. In the unrealistic worst case, the marker is first detected in camera frame b_3, since the exposure for b_2 started before the marker became visible.

the first time. This is the trigger for the application to select the next incoming camera frame b_2 as the correct result. Note that although in this example b_0 and b_1 contain the marker from the previous image A due to slow LC response times, it would be equally possible that the marker is not detectable anymore.

5. Pipelined synchronization

The algorithm from the previous section is simple and robust towards frame drops, since the projection of the image and its marker is maintained until the correct photo is received. However, in many applications the next image to project does not depend on the photo of the previous one. For example, when projecting a Gray code sequence to perform 3D scanning, the images' pattern content is determinable from the start. In that case, a lot of time is wasted by projecting an unnecessary multitude of wait frames, which produces a redundant supply of their photos (every camera frame after b_2 in Fig. 3). Their amount is predominantly determined by the projector and camera latency. To overcome these, the idea is to send the next image(s) to the projector before the current image was recorded yet, knowing that the camera will still capture the image in the near future. The idea of pipelining images is not new and was already discussed by Jaeggli et al. [8]. Instead of measuring the time that can be saved, we apply a theoretical worst-case analysis of how many wait frames it takes to know with certainty that an image will be recorded successfully. We then modify the algorithm to automatically adapt to an average-case wait frame count.

5.1. Worst-case analysis

The number of required wait frames depends on both the projector's ability to project a detectable marker and the camera to capture a photo in which the marker is detectable. For the worst-case analysis we make some unrealistic assumptions that go to the very limits of how cameras and LCD projectors operate. We assume that…

- …the trigger marker becomes visible only at the very end of the trigger frame.
- …the marker is detectable in a camera frame only if the marker was visible for the entire exposure time.

5.1.1. Wait frame count

With these worst-case assumptions, the worst-case camera offset for detecting the marker is almost the full duration of a camera frame, i.e., the frame's exposure starts just a fraction before the marker becomes visible (see b_2 in Fig. 4). Camera frame b_3 is thus the first frame to detect the marker and b_4 therefore the successful recording. Hence, the worst-case wait frame count is the number of projector frames spanning three camera frames (b_2, b_3, b_4):

$$W = \left\lceil \frac{3 \cdot F_p}{F_c} \right\rceil \tag{1}$$

where F_p and F_c denote the frame rate (in FPS) of the projector and camera respectively. By projecting the worst case number of wait frames it is guaranteed, for every possible camera offset, that a correct photo of the projection image will be recorded in the future. Therefore the next image C may be sent into the projection pipeline immediately afterwards.

5.1.2. Validation

The worst-case analysis is only valid if the frame rates of projector and camera are constant. Once a camera frame is dropped, the worst-case wait frame count is not sufficient anymore. If b_4 was dropped in Fig. 4, the next camera frame would already contain the next image C. It is deceptive to try and detect such cases to reject a photo by implementing an additional new marker for the successive transition frame(s) of C. Remember from Section 4 that not detecting a marker is never a guarantee for the marker not actually being present in a photo. To make the synchronization frame-drop-robust, it is necessary to prevent such false-negative failure detection cases. Therefore, we do not seek to detect errors but to validate correct recordings and in case of doubt discard a correct camera frame. It is up to the application to decide if a rejected recording is ignored or if the corresponding image is scheduled again for projection. In any case it is important to notify the application that an error occurred. For the sake of completeness we mention that one can achieve a partial frame-drop-robustness by increasing the wait frame count to cover as many camera frame drops as required. However, in case there is one more consecutive frame drop than expected, the error goes unnoticed. Moreover, the additional wait frames quickly add up when allowing multiple frame drops in a row and are thus time-costly.

Without reliable time measurements to detect frame drops we instead build a validation process on the marker detection to prove that a frame was recorded correctly. It is not sufficient to check if the marker is part of the supposedly correct photo to validate it. Depending on the LC response time it is possible that the marker is still detectable in frame c_0 (see Fig. 4), although the camera frame is already recording mostly image C. To support frame drops and occasionally malfunctioning marker detectors, we define the supposedly correct camera frame as the *candidate*. A candidate frame must be validated before being declared the result photo. Therefore, if the marker is once again detected after recording the candidate, the candidate is validated, otherwise invalidated. To reliably detect the marker this additional time, the worst-case wait-frame equation (see Eq. (1)) must be updated as follows:

$$W_v = \left\lceil \frac{4 \cdot F_p}{F_c} \right\rceil \tag{2}$$

Additionally, it is necessary to extend the projection by a fourth kind of frame, the *end frame*, without any marker. The projection sequence must contain as many end frames as transition frames. They thus resemble a transition time for the projected marker — from presence to complete absence. The end frames guarantee that the validating frame is the only camera frame that can possibly contain (traces of) the marker and the next image C already. Any validated candidate before must therefore contain only image B.

In Fig. 5 the worst-case scenario is extended by the additional wait and end frames. After the marker is detected for the first time in camera frame b_3, the next frame b_4 automatically becomes the candidate. The marker is once again detected in frame b_5, which validates b_4 as the correct result. If b_4 was dropped, the next frame delivered after b_3 is b_5 and becomes the candidate. It then depends on the marker detection of b_6 if it is validated. Similarly, in case b_4 and b_5 were dropped, the validation of

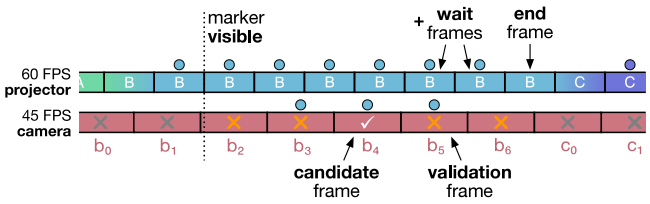


Fig. 5. The example from Fig. 4 extended by additional wait and end frames. Since the marker is detected in b_5, it validates the candidate b_4.

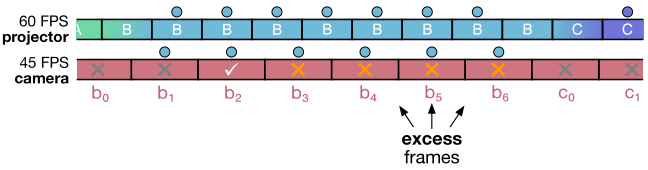


Fig. 6. Without frame drops and under realistic conditions the frames b_4, b_5 and b_6 are redundant.

candidate b_6 depends on c_0. Since the marker from the last wait frame might still be visible due to slow LC response times, it is important that the end frame maintains image B such that b_6 could be correctly validated (in contrast to the false-negative scenario described above).

5.1.3. Timeout invalidation

The last camera frame that can possibly contain a detectable marker of an image is a frame that starts exposure right before the end of the last projector end frame (second inverted worst-case assumption). There is thus an upper limit (timeout) of camera frames that can validate a candidate:

$$m = \frac{1 + W + E}{F_p} \qquad (3)$$
$$\text{limit} = 1 + \lceil F_c \cdot m \rceil - 2$$

E denotes the number of end frames (equals the number of transition frames). m describes the longest possible duration (best case) that the marker of an image is visible in the projection, spanning the trigger frame, wait and end frames. To consider the above mentioned last camera frame that can possibly contain the marker, $F_c \cdot m$ is rounded up. Additionally, the timeout must include the earliest photo ($+1$) that can possibly contain the marker, ending a fraction after the beginning of the trigger frame. If more than `limit` camera frames were received after candidate nomination (-2) and neither could validate it, the candidate is ultimately invalidated. In case a marker of a future image is detected without having validated the candidate of the last image first, the candidate and all potentially skipped recording tasks in between are also invalidated. By linking the timeout to the marker detection, the algorithm also becomes robust towards changes in the camera latency.

Since the timeout is based on having received the candidate frame, it is necessary to add filler images at the end of the entire image sequence. Otherwise it is, in theory, possible that the last frame never finishes. If, for example, the last image to project and record is invalidated, it takes an entire projector latency to project the image again, during which time the projected content is undefined. Since this duration is unknown it is impossible to set a fixed timeout. With bad luck it is then possible that frame drops cause the system to never detect one of the image's markers. The system would then search for the last image in an endless loop, since no successive image's marker would report the skip. Filler images keep the pipeline running in such a case and report the skip.

5.2. Adaptive algorithm

It is obvious that the worst-case analysis wastes a lot of time to cover unrealistic hardware behaviors. In Fig. 5 the marker synchronization fails to recognize the frames b_2, b_3, b_5 and b_6 as valid frames due to a lot of algorithmic overhead. To speed the general algorithm up we cannot make assumptions about the marker detector. Instead, we exploit the validation process to estimate how many projector frames can be saved. After validating a candidate we perform a retrospective analysis that counts how many camera frames after the validation frame would also be able to validate the candidate. Fig. 6 depicts a more realistic scenario than the worst-case assumption. Camera frame b_1 is the first to contain a detectable marker, thus assigning b_2 the candidate role. After validation through b_3, the frames b_4, b_5 and b_6 all contain the detectable marker and could also act as the validation frame. In that case, the number of excess camera frames is 3. Failures are recorded with a negative count (candidate invalidation: -1, image skipped entirely: -3).

The number of excess frames varies with each projected image and is predominantly influenced by the camera offset and dropped frames. Estimating the actual required number of wait frames must therefore be determined by considering multiple images. We base the optimal number of frames to analyze on the idea that failures are acceptable if it takes equal or less time to project the failures again than projecting the entire sequence with enough wait frames to prevent failures in the first place. As an example, we consider an image sequence where each of the seven images took seven projector frames in total. Since all candidates were successfully validated, the total frame cost of the sequence is $6 \cdot 7 = 42$. If, instead, the images were projected only with six projector frames each, enough frames are saved to compensate for one failure that must be projected again (assuming that the second projection of the failure image is successful): $6 \cdot 6 + 1 \cdot 6 = 42$. Hence, we define the window size of images to investigate with regard to excess frames as the current total projector frame count plus one:

$$S = T + 1 + W_i + E + 1 \qquad (4)$$

where T, W_i and E are the currently configured numbers of transition, wait and end frames respectively. If the last S images contain only a single failure, we assume to have reached the optimal wait frame count. If more/less failures are detected, the wait frame count must be increased/decreased. In that case, the S images are sorted by their excess frame count and the number of wait frames is updated based on the excess frames X of the second worst image (to allow for a single failure in the future):

$$W_{i+1} = \begin{cases} W_i + \max\left(1, \left\lfloor \frac{-X \cdot F_p}{F_c} \right\rfloor\right) & \text{for failures} > 1 \\ \max\left(0, W_i - \left\lceil \frac{X \cdot F_p}{F_c} \right\rceil\right) & \text{for failures} < 1 \\ W_i & \text{else} \end{cases} \qquad (5)$$

Too many failures guarantee a wait frame count increase ($W_i +$ max$(1, \ldots)$) and conservatively (floor function) add the missing frames. In contrast, a lack of failures is approached boldly (ceil function) by reducing the wait frame count.

Due to the projector latency and unfinished recording tasks still in the pipeline, extra care must be taken to exclude the images' excess frames from the analysis that were recorded with an old configuration of wait frames. If the required window size of S excess frame counts is not yet reached after a wait frame count change, we still monitor the situation of the available data.

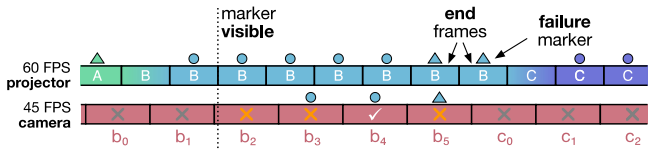


Fig. 7. b_4 is validated by detecting the failure marker in b_5.

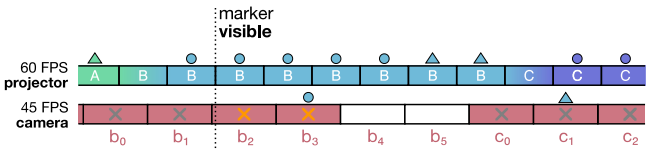


Fig. 8. c_0 is incorrectly validated by c_1.

If more than a single failure occurred in this set of reduced size, we speed up the reaction time by computing the new increased wait frame count immediately.

5.3. Failure marker algorithm

The adaptive algorithm aims at reducing wait frames to save time. Although this reduction includes the additional wait frames for validation, the end frames remain a constant overhead, especially if many transition frames are configured. However, requiring as many end frames as transition frames is not necessary if the marker detector fails early enough after the projector stops projecting it. In this section we present another adaptive algorithm that breaks some of our initial rules and is thus slightly less robust, but might still be a sufficiently robust heuristic for many applications.

For this section we assume that a marker detector is more likely to detect a marker that is voluntarily projected than a marker that is involuntarily projected (by means of the LC response time). It must thus not be possible that the detection fails while the marker is projected and then succeeds afterwards, when the marker is only barely visible due to the LC response time. Under these conditions we modify the validation from Section 5.1.2 as follows: Project the image with the original wait frame count from Eq. (1). Then, instead of padding the projection with marker-less end frames, embed a new failure marker into the end frames (unique per image). Furthermore, instead of repeating the end frames as many times as transition frames exist, repeat it only for so long that there is a high chance of detecting it once:

$$E_f = \left\lceil \frac{F_p}{F_c} \right\rceil \tag{6}$$

Similarly to before, a candidate is then validated by another subsequent camera frame if that frame contains the detectable main or failure marker. However, additionally, the candidate itself must not contain the failure marker. A candidate is invalidated if it contains the failure marker or if the failure marker could not be detected in a subsequent frame up to the adjusted timeout limit. In Fig. 7, the candidate frame b_4 does not contain the failure marker. The failure marker is detected in the next frame, b_5, which validates the candidate. In case b_4 is dropped, b_5 becomes the candidate. If the failure marker is correctly identified, it is immediately invalidated. If the detection fails, it is still possible that the failure marker is detected in c_0, which would also correctly validate b_5. If it is not detected, the initial assumption from above holds and there is no subsequent camera frame containing the detectable failure marker, which invalidates b_5 after the timeout.

Similar to the original adaptive approach (see Section 5.2), the excess frames for each image are counted until a failure marker is detected. If the failure marker is detected in the trigger or candidate frame, negative excess frames are stored (-2 and -1 respectively). It is important to note that, even without frame drops, in some hardware configurations an unfortunate camera offset can lead to skipping the failure marker entirely for an image (e.g., both devices 60 FPS, camera offset ca. 50 %). We count such failures like a regular camera-offset- or frame-drop-caused failure with an excess frame count of -1. However, for devices with identical frame rate the situation should be monitored and, in case of need, the end frame count increased by one to prevent an endless loop. If an image was skipped entirely, it is rated with an excess frame count of -3.

Fig. 8 visualizes the aforementioned flaw in the algorithm. Due to two consecutive frame drops (b_4 and b_5), the candidate c_0 is incorrectly validated by c_1, since the failure marker detection failed for c_0, but succeeded for c_1. Checking c_0 for the main marker would also be no reliable remedy for the problem, if the transition frame count were larger than 1 in that example.

6. Results

We tested our algorithms with two 3LCD projectors and cameras, each connected to a standard Windows 10 workstation:

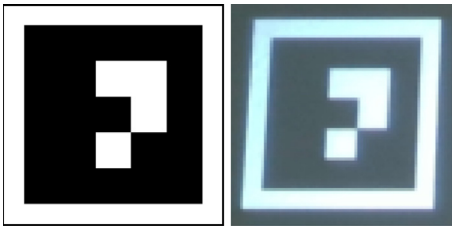
- `NEC@60`: projector `NEC NP-P451WG`, manufactured January 2015, 1280 × 800 pixels, 60 FPS.
- `Epson@59`: projector `Epson EB-L510U`, manufactured presumably 2019, 1920 × 1200 pixels, 59 FPS
- `Realsense@30/@60`: camera `Intel Realsense SR300`, 1920 × 1080 pixels at 30 FPS or 1280 × 720 pixels at 60 FPS
- `Svpro@120`: camera `Svpro DESVUSBFHD01MSFV USB Camera Module`, 640 × 480 pixels at 120 FPS

The photos of Fig. 1 were recorded with a Sony RX100 V camera. Since it does not support streaming directly to a computer, it was not further included in our tests.

6.1. Method

For each projector/camera-pair and synchronization strategy we conducted 100 experiments by projecting and recording 100 images each (without filler images). For the algorithms that rely on candidate validation, images were projected anew immediately after detecting invalidations. We embedded and detected ArUco markers [19] with the OpenCV library [20]. The number of distinct ArUco markers was restricted to 10 for practical reasons. This indirectly limits our pipeline synchronization algorithms to tolerating consecutive frame drops for at most 9 (4 with failure markers) entire image projection and recording tasks. To make the marker detection as fast as possible and not have the LC response time of the previous marker interfere, we alternated between two marker positions within the image for each two distinct consecutive markers. We noticed that the marker detection failed occasionally but repeatedly with the same images. Upon further inspection it seems like the default ArUco marker detection from OpenCV is impaired by some artifacts of the LC response time of the previous image (see Fig. 9), if the number of transition frames was too low. The markers' background was therefore permanently kept white.

Since the ArUco marker detection on a full-resolution camera image can exceed the camera frame time, on start-up we automatically determine the marker region of interest (ROI) in the camera image by projecting markers at both positions. The



(a) Original Marker (b) Projected Marker

Fig. 9. The marker from (a) is clearly visible for a human spectator in the projected image (b), but could not be automatically detected.

ROI is then derived from the corner detection in the first camera image that includes both markers. This initial ROI detection time (about 250 ms) is not included in our statistics. From this point on, the markers are searched for only in the ROI, which keeps the run-time detection time negligibly below 1 ms.

The projectors had a warm-up time of at least 1 h to increase the LCD-panel temperature, which in turn decreases the LC response time [21]. For the `Realsense@30/@60` camera we found that less than 1 % of all experiments exhibit frame drops (duration between two consecutive camera frames exceeding 150 % of the expected duration). To illustrate the basic algorithmic performance, we have excluded those outlier runs from the statistics and replaced them with runs without frame drops. In contrast, the `Svpro@120` camera exhibits a particularly high frame drop rate (see Fig. 10). During a 1-minute period the camera delivered only 4413 frames instead of expected 7200. While small deviations can be explained by rounding errors and imprecise time measuring, the histogram shows peaks around 13 ms and 24 ms, suggesting up to two consecutive frame drops on a regular basis. The histograms' distributions also demonstrate the need of software-based synchronization to support varying camera frame rates. On a non-real-time system we cannot determine whether the distribution of received frame rates is a result of an actual varying frame rate of the camera itself or if it is a varying accumulation of latencies from the camera connection to the receiving thread. While our pipeline methods expect a constant camera frame rate for the basic algorithmic design (see Section 5.1.1), the validation process guarantees correct recordings even if the camera frame rate varies. Please see the supplementary video as a demonstration of our method.

6.2. Evaluation

Fig. 1 depicts photos from the `NEC@60` projector and suggests that a single transition frame is not sufficient for the projector. We could confirm this assumption with the `Svpro@120` camera and found that even two transition frames are insufficient (see Fig. 11). In some rare images one can see artifacts of the previous image in the validated candidate frame. We could not find such artifacts anymore after projecting three transition frames. In contrast, one transition frame appears to be sufficient for the `Epson@59` projector. We show timing statistics for multiple settings with both projectors for reference (see Tables 1 and 2).

Depending on the hardware, configured number of transition frames and synchronization strategy, the 100 payload images are projected and recorded with 3.97 FPS to 14.84 FPS. Generally speaking, the sequential synchronization (see Section 4) is the slowest of our tested synchronization techniques. It is, however, difficult to quantify its performance for a general-purpose setup, since the performance mostly depends on the involved projector and camera latencies. The relations change if the camera is significantly slower than the projector, since both the worst-case estimate and the validation process are based on the camera frame rate. Moreover, the validation process becomes more time-costly the more transition frames are configured, which directly influences the number of end frames. If these effects add up (e.g., `NEC@60/Realsense@30`, 3 transition frames), the sequential synchronization performs better than pipeline techniques with validation. It is, however, outperformed by the failure-marker synchronization, which exhibits a constant validation overhead, regardless of the transition frames.

The worst-case synchronization (without validation) demonstrates the pipeline speed-up, being consistently faster than the sequential approach. The performance becomes drastically worse when adding the candidate validation to the method, being usually the second slowest or even slowest technique. In return, the results are robust. Although we filtered the frame-drop runs from the `Realsense@30/60` experiments, the total number of projection tasks (n) is consistently slightly higher (e.g., `Epson@59/Realsense@30`: 100.08) than the expected 100. This deviation may imply rare marker detection failures or that the real camera frame rate (before transferring the image via cable or network) is not constant, which impacts the worst-case analysis.

The wait frame overhead of the worst-case methods is reduced by the adaptive approach, making it generally fast. Though, it cannot compensate for the additional validation cost and is thus mostly slower than the worst-case algorithm without validation. The performance can only be improved further by cutting costs on the validation, demonstrated by the failure-marker synchronization. The method is usually the fastest or on par with the worst-case technique without synchronization, while providing more robust results.

A notable exception to the aforementioned performance advantage of the adaptive algorithm is the `Svpro@120` camera.

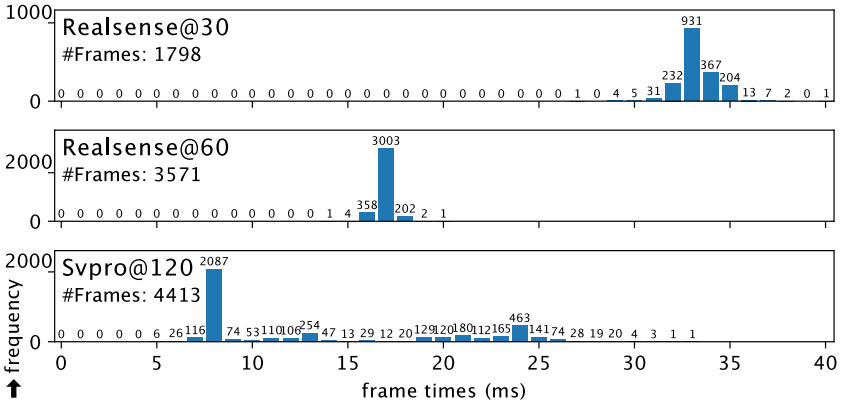


Fig. 10. Frame duration times histogram of the cameras tested over a 1-minute period.

Table 1
Epson@59 statistics. valid.: if a validation was performed, W^*: initial wait frame count, C: avg. percentage of wait frame count changes (increase/decrease), n: avg. number of projection tasks, T: avg. test run duration, subscripts: number of configured transition frames.

Epson@59/Svpro@120								
Strategy	Valid.	W^*	C_1 ↑/↓ (%)	n_1	T_1 (s)	C_2 ↑/↓ (%)	n_2	T_2 (s)
Sequential	–	–	–	100.00	12.864	–	100.00	14.397
Worst case	✗	2	0.00/0.00	100.00	6.738	0.00/0.00	100.00	8.402
Worst case	✓	2	0.00/0.00	100.24	8.423	0.00/0.00	100.11	11.736
Adaptive	✓	2	5.35/5.53	119.56	9.215	3.79/3.91	112.70	12.699
Adaptive	✓	1	5.82/5.20	121.83	9.326	4.47/3.72	114.92	12.874
Fail. marker	✓F	2	0.58/1.21	104.52	7.188	2.79/3.62	113.03	9.417
Epson@59/Realsense@60								
Strategy	Valid.	W^*	C_1 ↑/↓ (%)	n_1	T_1 (s)	C_2 ↑/↓ (%)	n_2	T_2 (s)
Sequential	–	–	–	100.00	12.288	–	100.00	13.955
Worst case	✗	3	0.00/0.00	100.00	8.407	0.00/0.00	100.00	10.076
Worst case	✓	4	0.00/0.00	100.20	11.872	0.00/0.00	100.11	15.187
Adaptive	✓	4	2.32/3.19	113.05	9.844	0.00/1.00	100.26	12.380
Adaptive	✓	2	2.65/2.65	115.42	9.606	0.04/0.03	100.44	12.002
Fail. marker	✓F	3	0.40/1.25	103.25	7.377	0.09/1.02	101.44	8.918
Epson@59/Realsense@30								
Strategy	Valid.	W^*	C_1 ↑/↓ (%)	n_1	T_1 (s)	C_2 ↑/↓ (%)	n_2	T_2 (s)
Sequential	–	–	–	100.00	16.706	–	100.00	16.935
Worst case	✗	6	0.00/0.00	100.00	13.392	0.00/0.00	100.00	15.132
Worst case	✓	8	0.00/0.00	100.08	18.496	0.00/0.00	100.16	21.823
Adaptive	✓	8	0.09/1.07	100.73	14.986	0.08/1.05	100.40	18.234
Adaptive	✓	6	0.11/0.37	100.57	14.335	0.12/0.36	100.60	17.673
Adaptive	✓	5	0.07/0.22	100.49	13.049	0.09/0.15	100.50	16.601
Adaptive	✓	4	0.76/0.06	102.56	13.161	0.81/0.09	102.82	16.753
Fail. marker	✓F	6	0.07/1.11	100.30	12.460	0.01/1.00	100.14	15.691

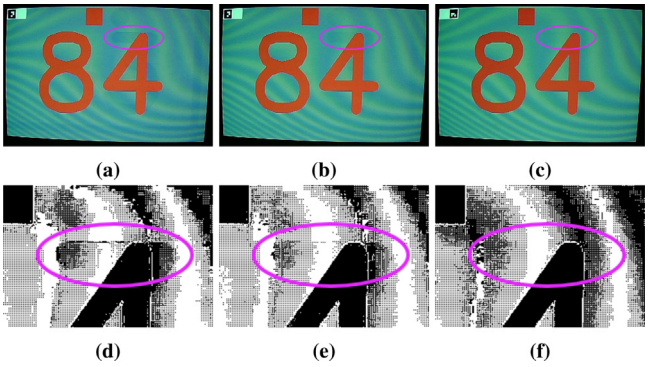


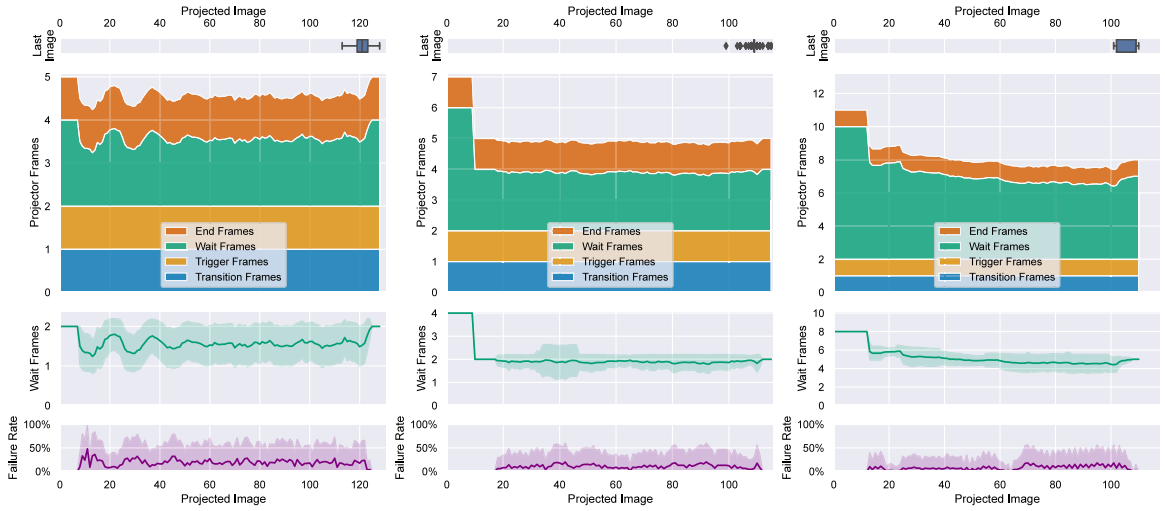
Fig. 11. Validated NEC@60 candidate frames, recorded with the Svpro@120 camera, with one (a), two (b) and three (c) transition frames. Traces from the previous image (with number 83) are visible (enlarged binary threshold image below) if the number of transition frames is insufficient (a, b).

Since the frame drop rate is particularly high, the adaptive approach loses speed by adjusting the wait frame count repeatedly. These adjustments also lead to a high rate of failures which manifest themselves as a high rate of image repetitions, for example NEC@60/Svpro@120 with 1 transition frame: 122.22 image projection tasks for originally 100 payload images. In more than 11% of images, the wait frames had to be adjusted either up (5.83%) or down (5.94%). Fig. 12(a) visualizes the temporal course. The average number of wait frames fluctuates between 1 and 2 and there is a generally high number of failures throughout all images. In contrast, Fig. 12(b) shows a clear convergence to 2 wait frames with the Realsense@60 camera. Since the convergence indirectly depends on the camera frame rate, convergence slows down with a lower frame rate (see Fig. 12(c)). To strip the initial convergence overhead from the time table, we included timings with an presumably optimal converged wait frame count (W_0) for each adaptive test run. In case the asymptote was not clearly visible from the graph, we included the closest integer asymptotes. As expected, if convergence is achievable, the total average duration decreases.

From thousands of test runs with the Realsense@30/@60 we could only identify a few dozens experiments with frame drops. Interestingly, if a frame drop did occur, there were usually multiple drops in the test run. For most of the runs with validation we cannot identify a deviation from the frame-drop-less averages. This seems reasonable, as the validation methods tolerate frame drops and chances are that the dropped frame was not critical (e.g. during a transition frame). There are a few exceptions, e.g., Epson@59/Realsense@30, 1 transition frame, adaptive synchronization, 7 frame drops: n: 101, time: 16.149 s. Compared to the frame-drop-less experiments (14.986 s), it converged slower, only after 21 images (regular: 13 to 14), and only to 6 wait frames, whereas many other runs converged lower (4). These exceptions further back our assumption that sporadic frame drops are correctly identified and processed by the synchronization methods with validation, but may slow down the pipeline.

6.3. Discussion

From our results we draw multiple lessons, the most important one being: The best variant of marker-based synchronization depends on the setup and the application itself. If a user does not have any prior knowledge about the projector and camera hardware but requires robust results, the adaptive synchronization is a good all-rounder. In case the risk of slight color deviations from a next image is bearable, the failure-marker synchronization provides the best cost-benefit ratio. While we did not witness such deviations in our tests, they are, in theory, possible and might occur more easily with high-speed cameras. The worst-case synchronization methods are more suited for applications that do not allow for changing the order of the images but tolerate the occasional failure, either by an invalid result (no validation) or by skipping a result (invalidation of a candidate). The sequential method remains the best option if changes in the order of images



(a) NEC@60/Svpro@120 (b) NEC@60/Realsense@60 (c) NEC@60/Realsense@30

Fig. 12. Overview of the adaptive test runs with 1 transition frame (image index 0-based). Half-transparent areas depict the standard deviation. Note the different scaling of the y-axis. (a) After the initial worst-case wait-frame setting (2), the number fluctuates between 1 and 2. (b) The optimal wait frame count converges to 2. (c) The slower the camera, the wider the excess frames window, the slower the convergence. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
NEC@60 statistics. valid.: if a validation was performed, W^*: initial wait frame count, C: avg. percentage of wait frame count changes (increase/decrease), n: avg. number of projection tasks, T: avg. test run duration, subscripts: number of configured transition frames.

NEC@60/Svpro@120											
Strategy	Val.	W^*	C_1 ↑/↓ (%)	n_1	T_1 (s)	C_2 ↑/↓ (%)	n_2	T_2 (s)	C_3 ↑/↓ (%)	n_3	T_3 (s)
Sequent.	–	–	–	100.00	12.138	–	100.00	14.128	–	100.00	16.879
Worst C.	✗	2	0.00/0.00	100.00	6.747	0.00/0.00	100.00	8.442	0.00/0.00	100.00	10.096
Worst C.	✓	2	0.00/0.00	100.14	8.444	0.00/0.00	100.84	11.885	0.00/0.00	102.64	15.486
Adaptive	✓	2	5.83/5.94	122.22	9.451	4.27/4.37	114.83	13.042	2.91/2.92	110.59	16.545
Adaptive	✓	1	6.29/5.63	124.26	9.523	4.97/4.16	117.46	13.277	3.60/2.53	112.97	17.034
Fail. M.	✓F	2	0.55/1.38	103.69	7.125	2.70/3.53	111.95	9.387	1.11/2.07	104.74	10.654
NEC@60/Realsense@60											
Strategy	Val.	W^*	C_1 ↑/↓ (%)	n_1	T_1 (s)	C_2 ↑/↓ (%)	n_2	T_2 (s)	C_3 ↑/↓ (%)	n_3	T_3 (s)
Sequent.	–	–	–	100.00	11.364	–	100.00	13.280	–	100.00	14.840
Worst C.	✗	3	0.00/0.00	100.00	8.528	0.00/0.00	100.00	10.137	0.00/0.00	100.00	11.882
Worst C.	✓	4	0.00/0.00	100.17	12.060	0.00/0.00	100.14	15.152	0.00/0.00	100.42	18.707
Adaptive	✓	4	1.85/2.76	110.07	9.614	0.16/1.15	101.08	12.512	0.02/1.01	100.30	15.742
Adaptive	✓	2	2.22/2.23	111.81	9.222	0.18/0.17	101.27	12.157	0.05/0.04	100.46	15.251
Fail. M.	✓F	3	0.50/1.48	101.76	7.291	0.30/1.28	101.49	9.041	0.03/1.02	100.82	10.615
NEC@60/Realsense@30											
Strategy	Val.	W^*	C_1 ↑/↓ (%)	n_1	T_1 (s)	C_2 ↑/↓ (%)	n_2	T_2 (s)	C_3 ↑/↓ (%)	n_3	T_3 (s)
Sequent.	–	–	–	100.00	15.560	–	100.00	16.758	–	100.00	18.984
Worst C.	✗	6	0.00/0.00	100.00	13.456	0.00/0.00	100.00	15.127	0.00/0.00	100.00	16.776
Worst C.	✓	8	0.00/0.00	100.19	18.508	0.00/0.00	100.24	21.868	0.00/0.00	100.15	25.186
Adaptive	✓	8	2.09/2.46	107.47	15.042	2.34/2.57	108.68	19.150	0.35/1.26	101.38	22.521
Adaptive	✓	6	2.44/1.76	108.70	14.631	2.62/1.85	109.69	18.704	0.58/0.41	102.14	21.976
Adaptive	✓	5	2.23/1.27	108.71	14.193	2.67/1.38	110.81	18.252	0.96/0.48	103.88	20.825
Adaptive	✓	4	3.53/1.39	113.63	14.460	4.31/1.76	117.50	19.050	1.84/0.45	106.82	21.292
Fail. M.	✓F	6	0.72/1.63	103.10	12.486	0.00/1.00	100.06	15.670	0.94/1.53	102.97	16.109

are not allowed and the results must be robust. In case of exceptionally low latencies in the setup or a slow camera and high LC response time, the sequential method usually outperforms other techniques.

The synchronization performance is directly tied to the projector and camera frame rate and (at least for color-correctness) the LC response time. Additionally, the utilized marker detection algorithm must process the incoming camera photos faster than new photos are delivered, e.g., by defining a region of interest. For applications that expect a real-time synchronized-camera-image input stream, it is necessary to employ hardware that is fast enough to compensate for the synchronization overhead (e.g., 270 FPS devices for an approx. 60 FPS stream with failure-marker synchronization). As of today, the required speed might not be achievable yet with LCD technology. In such cases, there is a trade-off between color-correctness and performance.

7. Limitations

We have designed our synchronization to be widely applicable. Our only two technical prerequisites are that camera frames arrive in their correct order (frame drops are allowed) and that the frame rates of projector and camera are known (usually provided by the operating system) to determine the initial wait frame count. Though, we expect our adaptive algorithm to handle

wrongly assumed frame rates as well, since the validation process adjusts the wait frame count automatically to match a different frame rate.

Our greatest limitation is the embedded marker. The marker must be visible for the camera, which implies that it is also visible for any other spectator apart from the camera, which may impair the projection immersion. The problem is solvable if projector and camera communicate via a wavelength invisible to humans. For example, if the projector is equipped with an internally synced infrared (IR) projector and the camera equally features IR vision too, the marker can be embedded in the IR image only.

Furthermore, depending on the marker, it may also restrict the projection surface. For an ArUco marker to be robustly detectable, the surface should be as planar and tilted towards the camera as possible. However, the restriction is only imposed on the area of the projector image where the markers are embedded. For a structured-light scan of a three-dimensional object, for example, a small box can be included into the scene where only the markers are projected.

The marker must also be identifiable by a robust detector. Our algorithms are designed to tolerate false-negatives (wrongfully not detecting an existing marker), which are mostly handled like camera frame drops. However false-positives (wrongfully detecting a marker that is not there) would quickly break the synchronization.

Finally, we have demonstrated that the LC response time is an issue for color-correct photos of the projection, since the time may exceed a single projector frame. Though, we must currently rely on the user to configure the correct number of transition frames. This makes the problem tangible for a user, since no exact time measurement must be entered but only a rough approximation in projector frame units. However, even if an exact time measurement was known, the marker detection dependency limits our algorithms' reaction times to multiples of projector frames, which may waste time in the future, when transition times might be significantly faster than the projector's frame rate.

8. Conclusion

We have presented multiple marker-based algorithms to synchronize a projector and camera. Depending on the application's needs and the setup, either speed or robustness can be favored. The adaptive strategy is a compromise of both, providing robust results, while estimating the best performance possible. Our algorithms are specifically designed for LCD projectors, although we assume that they are adjustable towards other kinds of projectors too, e.g., DLP projectors, by reducing or omitting the transition frames. However, other additional prerequisites may apply in that case, e.g., the camera's frame rate being a multiple of the DLP projector's color wheel rotation frequency to avoid rainbow effects.

For future work, we see the need of automating the decision-making on the required number of transition frames, e.g., by establishing a mathematical model of the projector's LC response time [22].

Declaration of competing interest

One or more of the authors of this paper have disclosed potential or pertinent conflicts of interest, which may include receipt of payment, either direct or indirect, institutional support, or association with an entity in the biomedical field which may be perceived to have potential conflict of interest with this work. For full disclosure statements refer to https://doi.org/10.1016/j.gvc.2021.200034.

Although this work is not funded by any third party, we want to disclose that one of the authors was in the past employed by a company (Bosch Sensortec) that develops projector systems.

Acknowledgment

We would like to thank Johannes Walz for his preliminary studies on cameras and LCD projectors for this work.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.gvc.2021.200034.

References

[1] Zhang S, Huang PS. High-resolution, real-time three-dimensional shape measurement. Opt Eng 2006;45(12):123601. http://dx.doi.org/10.1117/1.2402128.
[2] Grundhofer A, Seeger M, Hantsch F, Bimber O. Dynamic adaptation of projected imperceptible codes. In: 2007 6th IEEE and ACM international symposium on mixed and augmented reality. Nara, Japan: IEEE; 2007, p. 1–10. http://dx.doi.org/10.1109/ISMAR.2007.4538845.
[3] Wissmann P, Schmitt R, Forster F. Fast and accurate 3D scanning using coded phase shifting and high speed pattern projection. In: 2011 International conference on 3d imaging, modeling, processing, visualization and transmission. Hangzhou, TBD, China: IEEE; 2011, p. 108–15. http://dx.doi.org/10.1109/3DIMPVT.2011.21.
[4] Rusinkiewicz S, Hall-Holt O, Levoy M. Real-time 3D model acquisition. ACM Trans Graph 2002;21(3):438–46. http://dx.doi.org/10.1145/566654.566600.
[5] Cotting D, Naef M, Gross M, Fuchs H. Embedding imperceptible patterns into projected images for simultaneous acquisition and display. In: Third IEEE and ACM international symposium on mixed and augmented reality. Arlington, VA, USA: IEEE; 2004, p. 100–9. http://dx.doi.org/10.1109/ISMAR.2004.30.
[6] Vieira M, Velho L, Asla Sa, Carvalho P. A camera-projector system for real-time 3D video. In: 2005 IEEE computer society conference on computer vision and pattern recognition - workshops. vol. 3, San Diego, CA, USA: IEEE; 2005, p. 96. http://dx.doi.org/10.1109/CVPR.2005.385.
[7] Atif M, Lee S. Adaptive frame rate pattern projection for structured light 3D camera system. In: 2017 IEEE international conference on multisensor fusion and integration for intelligent systems. Daegu: IEEE; 2017, p. 482–7. http://dx.doi.org/10.1109/MFI.2017.8170367.
[8] Jaeggli T, Koninckx TP, Gool LV. Online 3D acquisition and model integration. In: IEEE Int. Workshop on Projector–Camera Systems in Conjunction with ICCV 2003. 2003.
[9] Petkovic T, Pribanic T, Donlic M, D'Apuzzo N. Software synchronization of projector and camera for structured light 3D body scanning. In: Proceedings of the 7th international conference on 3d body scanning technologies. Lugano, Switzerland: Hometrica Consulting - Dr. Nicola D'Apuzzo; 2016, p. 286–95. http://dx.doi.org/10.15221/16.286.
[10] Petković T, Pribanić T, Donlic M, D'Apuzzo N. Multi-projector multi-camera structured light 3D body scanner. In: Proc. of 3DBODY.TECH 2017 - 8th int. conf. and exh. on 3D body scanning and processing technologies. 2017, p. 319–26. http://dx.doi.org/10.15221/17.319.
[11] El Asmi C, Roy S. Fast unsynchronized unstructured light. In: 2018 15th Conference on computer and robot vision. Toronto, ON, Canada: IEEE; 2018, p. 277–84. http://dx.doi.org/10.1109/CRV.2018.00046.
[12] Perli SD, Ahmed N, Katabi D. PixNet: interference-free wireless links using LCD-camera pairs. In: Proceedings of the sixteenth annual international conference on mobile computing and networking. Chicago, Illinois, USA: ACM Press; 2010, p. 137. http://dx.doi.org/10.1145/1859995.1860012.
[13] Moreno D, Calakli F, Taubin G. Unsynchronized structured light. ACM Trans Graph 2015;34(6):1–11. http://dx.doi.org/10.1145/2816795.2818062.
[14] Video Electronics Standards Association. Flat panel display measurements standard version 2.0. 2001, p. 332, URL: https://vesa.org.
[15] Elze T. Achieving precise display timing in visual neuroscience experiments. J Neurosci Methods 2010;191(2):171–9. http://dx.doi.org/10.1016/j.jneumeth.2010.06.018.
[16] Elze T, Tanner TG. Temporal properties of liquid crystal displays: Implications for vision science experiments. In: Krekelberg B, editor. PLoS One 2012;7(9):e44048. http://dx.doi.org/10.1371/journal.pone.0044048.
[17] Suzuki S, Suzuki M, Takizawa H, Nakanishi N. Response time evaluation for LCD display modes and its relationship to moving image perception. In: Wu MH, editor. San Jose, CA; 2002, p. 85–92. http://dx.doi.org/10.1117/12.463794.

[18] Chen H, Peng F, Gou F, Lee YH, Wand M, Wu ST. Nematic LCD with motion picture response time comparable to organic LEDs. Optica 2016;3(9):1033. http://dx.doi.org/10.1364/OPTICA.3.001033.
[19] Garrido-Jurado S, Muñoz-Salinas R, Madrid-Cuevas F, Marín-Jiménez M. Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognit 2014;47(6):2280–92. http://dx.doi.org/10.1016/j.patcog.2014.01.005.
[20] Bradski G. The OpenCV library. Dr. Dobb's J Softw Tools 2000.
[21] Liang H, Badano A. Temporal response of medical liquid crystal displays. Med Phys 2007;34(2):639–46. http://dx.doi.org/10.1118/1.2428403, URL: http://doi.wiley.com/10.1118/1.2428403.
[22] Adam P, Bertolino P, Lebowsky F. Mathematical modeling of the LCD response time. J Soc Inf Disp 2007;15(8):571. http://dx.doi.org/10.1889/1.2770857.