Building structured lighting applications using low-cost cameras

Sibgrapi paper ID: 12

***Abstract*—Structured lighting is a computer vision technique that projects illumination patterns onto the scene to facilitate feature extraction from the captured images. The use of low-cost cameras is avoided not only due to their low image quality but mostly due to the lack of a synchronization mechanism for the illuminators. In this paper we propose a method to synchronize low-cost cameras and illuminators based on the dynamic estimation of the camera sensor exposure and number of lines. At the same time, the use of structured stroboscopic lighting is used to enhance the image quality. Starting with a coarse estimation of the sensor parameters, we developed computer vision algorithms that detect image artifacts created by the structured lighting when the illuminators are not correctly synchronized with the camera frames. The detected artifacts are used to refine the estimation of the sensor parameters and to adjust the firing of the illuminators until a clear picture is obtained. Our technique requires a simple external circuit to control the firing of the illuminators, that is adjusted by software, and allows virtually any modern digital camera to be used in structured lighting applications. We demonstrate the use of this technique in a fast 187 fps robust pupil detector that can be used for gaze interaction applications.**

I. Introduction

Shape reconstruction using coded structured light is considered one of the most reliable techniques to recover object surfaces [1]. The technique projects a light pattern and captures images from one or more cameras. The three-dimensional information is obtained from the distortions observed from the projected pattern. One of the most commonly used strategies is based on temporal coding, in which a sequence of patterns are successively projected. The codeword for a given pixel is given by the sequence of intensities for that pixel across the projected patterns [2].

The use of structured lighting generally requires precise synchronization between the light pattern and camera frames. Synchronization can be achieved using specialized hardware [3], or image processing software [4], or both [5], [6].

One way of synchronizing images from different cameras is by triggering a pulse of light. A strobe creates a virtual exposure time that is shared by all cameras. The use of stroboscopic lighting also helps to minimize some image artifacts that are created using rolling shutter cameras, such as skew and smear. For example, Theobalt et al. [7] have used stroboscopic lighting to capture high speed motion of a baseball using standard still cameras and a high power stroboscope, and Bradley et al. [8] have used stroboscopic light to synchronize an array of consumer grade cameras.

When consecutive camera frames must be illuminated by different light sources, the firing of each light must be carefully synchronized with the beginning of each frame. Such time multiplexed illumination have been used for multispectral imaging [9], object relighting [10], and pupil detection [11] using a technique called differential lighting (DL).

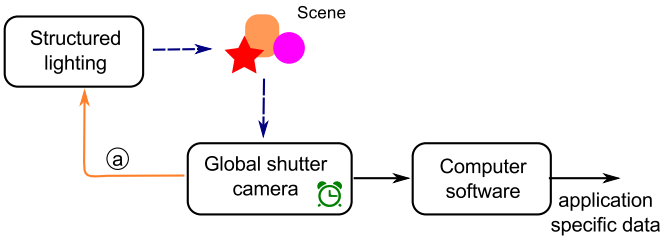


Fig. 1. Traditional setup of a structured lighting system. Link (a) provides the synchronization means between the camera and lighting. Note that the link can be implemented using the computer as intermediate, which then controls the lighting.

In a traditional structured lighting system, the camera provides the synchronization signal to fire the light sources. Figure 1 shows a block diagram of a typical system. The structured lighting can be generated by a DLP projector, in shape reconstruction systems, or spatially arranged infrared LEDs in a DL pupil detector.

We propose to replace the global shutter, synching-capable specialized camera by a low cost one. Low cost digital cameras typically employ rolling shutters and constitute the great majority of cameras being used, employed on notebooks, cellphones, tablets, and so on. Figure 2 shows a block diagram of our proposed method, called Stroboscopic Differential Lighting (SDL), that uses an external timing source controlled by an image processing software.

When lighting is not properly synchronized with the camera frames, a dark stripe artifact is create in the image, i.e., a region characterized by scanlines progressively dark until a minimum followed by lines progressively bright. The height (number of scanlines) and position of the stripe must be such that, when in sync, just a few scanlines from the top and bottom of the frame are dark, leaving the image mostly unaffected.

Our technique estimates the sensor internal parameters which allows the automatic camera configuration. The exact pattern produced on the frames for specific exposure and strobe durations are described, thus allowing the right exposure value to be set automatically.

In this paper we focus on methods to estimate the number of scanlines of the sensor and define how to set the exposure length. These parameters are essential to achieve synchronization between the rolling shutter camera and light sources. As a result of our method, an increased number of applications can

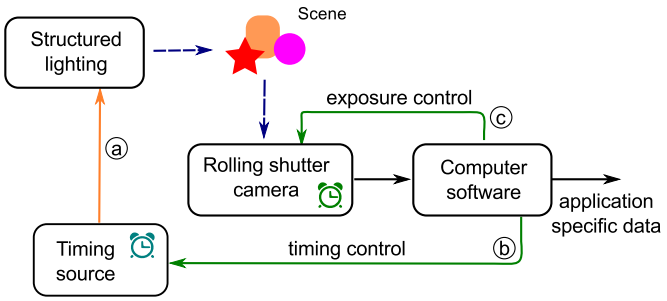


Fig. 2. Proposed setup. An external timing source replaces the camera synchronization.

be developed using practically any affordable high resolution web camera and high frame-rate action cameras.

The rest of the paper is organized as follows. The next section describes the camera model and how the parameters can be estimated using image processing techniques. We also present how an external timing source can be adjusted to trigger illuminators in synchrony to the camera. Section III presents a particular implementation of the technique, followed by Section IV, that presents experimental results using a low cost high speed camera, including a practical application. Finally, in Section V we draw the conclusions.

II. Estimating Camera Internal Parameters

Our method is able to synchronize stroboscopic lighting to a wide range of digital cameras. The method lies in setting the camera exposure such that the lighting produces a distinguishable dark stripe in the frames. Such stripe can be made invisible by moving it to a region in the sensor that is not displayed, here called invisible scanlines. We can obtain the number of such lines from the total number of scanlines (S) and the actual frame height. However, the number of scanlines is not straightforward to get, as it is related to the frame timings, and thus, resolution-dependent. For each frame height the camera supports, a different number is available. Such information is generally present in the sensor's datasheets. However, for many cameras it is hard to figure out the actual sensor employed without opening the camera. Querying the sensor parameters in execution time is not possible either. Estimating the number of scanlines allows the development of a closed-form exposure definition.

A. Exposure, gain and strobe duration

To achieve good image quality for both dark and bright illuminated scenes, cameras allow the control of a variety of parameters. They include variable optical aperture, variable gain settings, and variable integration time. In low cost cameras, such control is mostly by software. Hereafter we assume that the camera has no physical exposure means and that exposure refers only to the integration time. The exposure is represented by Δe when given in seconds and by $\mathbb{E}$ to refer to the driver specific value. Later in this paper we will discuss a method to translate between these two values.

Exposure and gain are normally manipulated together. The gain is used to amplify the data collected during integration, and is normally used to control the image brightness between two adjacent steps in exposure. Since increasing the gain of the image data amplifies the received signal, any noise in the original signal is also amplified upon increasing the gain. Usually, the camera is responsible for adjusting the exposure and gain to deliver the best quality image to the user.

In general, for constant or slow changing light sources, the exposure affects how much light enters the camera sensor. Associated to the gain, they control the final image brightness. However, when pulsed or stroboscopic light is employed, an increase in exposure not necessarily implies that more light enters the sensor. We can break the influence of those two light sources apart, and denote the intensity of a scanline as:

$$I(.) = \delta' \cdot \Delta e + \delta'' \cdot \Delta s \tag{1}$$

where $I(.)$ is the average intensity of a scanline fully illuminated by the strobe with duration Δs, δ' is the room intensity multiplier, and δ'' is the strobe intensity multiplier. Those two variables incorporate the surface reflectance of the objects imaged at the scanline, the camera conversion efficiency and gain, and the irradiation coming from the ambient and from the strobe. Note that increasing Δe does not change the intensity if the ambient light is null (i.e. $\delta' = 0$). In most CMOS sensors, photon flux conversion to photocurrent is linear, photocurrent is integrated into charge during exposure, and charge to voltage conversion is performed using linear amplifiers [12], [13]. Thus, we assume that linearly reducing Δs leads to a linearly proportional reduction in the line intensity. This concept is shown in Figure 4 and Figure 5 to describe the expected frame output under certain conditions.

B. Rolling Shutter Camera Model with Stroboscopic Lighting

The camera model adopted is similar to the one of Bradley et al. [8], with the added generalization of allowing vertical blanking lines both before and after the visible scanlines, which was introduced by Borsato et al. [14] in order to be compliant with camera chips [15], [16] and image sensor receivers [17].

The readout time $r_y^{(j)}$ of line y from frame j is given by [8]

$$r_y^{(j)} = t_0^{(j)} + \frac{y}{S} \cdot \Delta t = t_0^{(0)} + \left(j + \frac{y}{S}\right) \cdot \Delta t \tag{2}$$

where $t_0^{(j)}$ is the readout time instant of the topmost scanline in frame j. The period of the camera frame is denoted by Δt, and is defined as the interval needed to read the total number of scanlines S.

Each pixel exposure starts some fixed time before the readout time, resulting in exposure intervals temporally offset for each scanline. The moment a line y begins to be exposed for frame j is given by

$$e_y^{(j)} = r_y^{(j)} - \Delta e \tag{3}$$

where Δe is the frame exposure interval, and is upper bounded to Δt.

Figure 3 presents the capture model of a rolling shutter camera subject to stroboscopic illumination. Typically, such

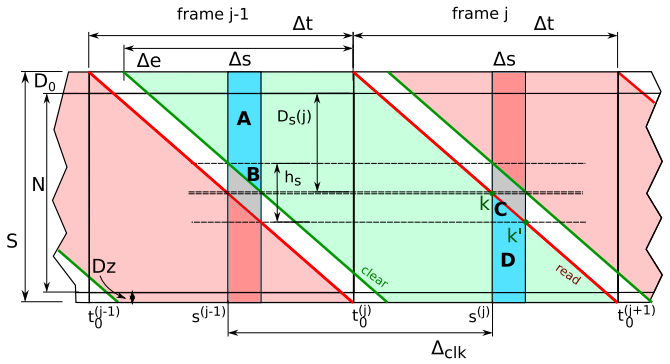


Fig. 3. Rolling shutter camera model subject to stroboscopic lighting (Adapted from [14]).

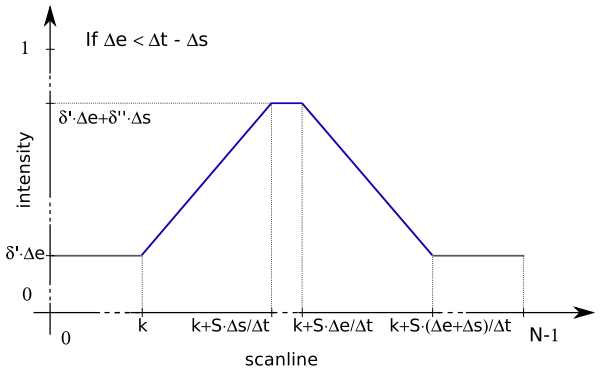


Fig. 4. Ideal intensity profile of a column of a frame captured under stroboscopic lighting for $\Delta e < \Delta t - \Delta s$.

cameras have optical black and/or dummy lines (named D_0) before the visible scanlines (N in the model). They can also have invisible lines after the image (named D_Z in the model). The invisible scanlines are not output, but count on the total frame time as any ordinary image line, so they are important for our formulations.

In the model, $s^{(j)}$ denotes the moment the strobe is triggered for a given frame j, while Δs give the strobe duration. Due to the time shift of the exposure interval of adjacent lines, the stroboscopic light might affect them distinctly. While regions A and D from Figure 3 present outputs as defined in (1), regions B and C form a sort of dark stripe. If we take the intensities along a vertical line of the frame, we would obtain what we call a stripe profile (better characterized if one takes the average of all columns). The stripe profile and size h_s are dependent on Δs, Δt, Δe, and S, while the stripe vertical position, given by $D_s(j)$, is a function of the phase difference between the camera clock and the lighting.

In the following sections, we exploit the interaction among the camera parameters and lighting configurations in the formation of stripes. We developed image processing algorithms that detect the stripes and estimate the total number of scanlines S. These parameters are then used to control the firing of the lights to move the stripes to the dummy lines, thus removing the artifacts from the images.

C. Estimating the total scanlines (S)

Consider $s^{(j)} = r_k^{(j)}$, i.e. the strobe is fired when line k begins to be read in frame j. Thus the last line to receive the strobe light is $e_{\hat{k}}^{(j)} = s^{(j)} + \Delta s$. Using (3) we have $r_{\hat{k}}^{(j)} - \Delta e = s^{(j)} + \Delta s$, and accordingly $r_{\hat{k}}^{(j)} = s^{(j)} + \Delta s + \Delta e$. From the difference between the readout of line $\hat{k}$ and k, we obtain $r_{\hat{k}}^{(j)} - r_k^{(j)} = \Delta s + \Delta e$. Using (2) we have

$$t_0^{(j)} + \frac{\hat{k}}{S} \cdot \Delta t - t_0^{(j)} - \frac{k}{S} \cdot \Delta t = \frac{\hat{k} - k}{S} \cdot \Delta t = \Delta s + \Delta e \quad (4)$$

Let us define $h_s = \hat{k} - k$ as the stripe height, so the total number of scanlines can be determined as follows:

$$S = \frac{\Delta t \cdot h_s}{\Delta e + \Delta s} \quad (5)$$

where Δt, Δe and Δs are given in seconds and, S and h_s are given in scanlines.

While Δt and Δs can be easily computed, the same might not be true for Δe. As we assume no prior knowledge about the exposure unit, granularity, and minimum value, setting it to the lowest value does not imply it will be close to zero. Thus, we estimate a device and resolution dependent constant $\hat{\beta}$, as an approximation to β, such that given some $\mathbb{E}$ and Δe, it holds that $\mathbb{E} \approx \Delta e / \hat{\beta}$ (with $\mathbb{E} = \Delta e / \beta$). This approximation is computed by iterating $\mathbb{E}$ from its minimum to a value κ that leads to $h_s = N/2$. The pairs $(\mathbb{E}, h_s)$ are used to fit a model of the form $h_s = a \cdot \mathbb{E} + b$ using linear regression, where b gives an approximation to $S \cdot \Delta s / \Delta t$, and a can be used to estimate β, such that $\hat{\beta} = a \cdot \Delta t / S$. Note that this step can be skipped if the user provides the true β. For instance, in a Linux or MacOs system with an UVC compliant driver, $\beta = 0.1 \cdot 10^{-3}$ [18].

D. Estimating S for cameras on auto-exposure (AE)

When auto-exposure is enabled in low light conditions, the camera exposure time rises to its maximum. To estimate S in this case, observe that the time to integrate a line corresponds to the shift in the exposure window of the given line. This shift is a function of S and Δt. When the strobe is triggered, some scanlines are already integrating, receiving its full length. Some start integrating while the strobe is on, receiving light proportional to the exposure time. And some will start integration after the strobe is already off. This sequence is represented in Figure 5. Note that the ramp (linear slope of region B) might also be inverted, depending on the phase difference between the timing source and the start of the frame.

Assuming line k is being read at time $s^{(j)}$, and k' is being read at time $s^{(j)} + \Delta s$, $r_k^{(j)} = s^{(j)}$ and $r_{k'}^{(j)} = s^{(j)} + \Delta s$. Thus, $r_{k'}^{(j)} - r_k^{(j)} = \Delta s$. From (2), we have $\frac{k'}{S} \cdot \Delta t - \frac{k}{S} \cdot \Delta t = \Delta s$. Let h_s^* be $k' - k$, then the total number of scanlines S is

$$S = \frac{(\Delta t \cdot h_s^*)}{\Delta s} \quad (6)$$

As the exposure will be close to Δt, to produce a distinct stripe in the image, the strobe must be triggered on interleaved frames. Note that this method can be employed to compute

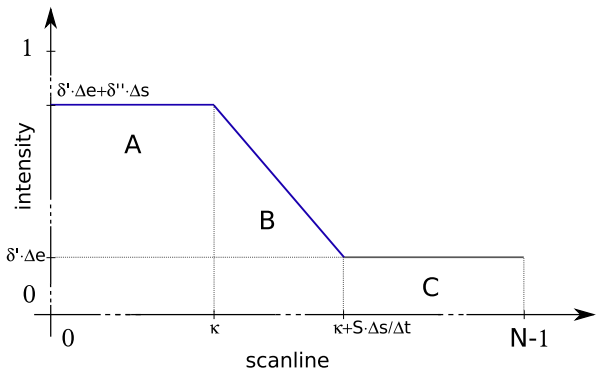


Fig. 5. Intensity profiles of a column of a frame captured under a pulse of light with $\Delta e \approx \Delta t$.

S for a camera on manual exposure as well, by adjusting its exposure to maximum (close to Δt). Note that β is not required in this method.

E. Automatically adjusting the exposure

We propose a procedure to adjust the exposure automatically, such that a particular stripe is created. The total number of scanlines S is assumed to be known from the application of the methods described in Section II-C. We also assume that β, a mapping coefficient from the driver exposure to seconds, is also known (it is approximated by $\hat{\beta}$).

Assuming $s^{(j-1)} = e_k^{(j)}$, i.e, the strobe is set when the exposure of line k begins, the last line to receive light from the strobe is $k' = k + \frac{S \cdot \Delta s}{\Delta t}$. To obtain a line with no illumination, the next strobe must illuminate line $k'+1$. Using (3), $r_k^{(j)} - e_k^j = \Delta e$ for any given k, and by (2), $r_{k'+1}^{(j)} - e_{k'}^j = \Delta e + \frac{1}{S} \cdot \Delta t$.

Line $k'+1$ is readout at $r_{k'+1}^j$, and we want it to coincide with the next strobe $s^{(j)} = s^{(j-1)} + \Delta_{clk}$. Thus,

$$s^{(j-1)} + \Delta_{clk} - e_{k'}^j = \Delta e + \frac{1}{S} \cdot \Delta t. \tag{7}$$

Replacing $s^{(j-1)}$ by $e_k^{(j)}$, and then $e_k^{(j)} - e_{k'}^{(j)}$ by $-\Delta s$ in (7), we have

$$\Delta e = \Delta_{clk} - \Delta s - \frac{1}{S} \cdot \Delta t. \tag{8}$$

With such exposure, stripes with one scanline not illuminated by any strobe are expected. The stripe height is $h_s = 1 + \frac{2 \cdot S \cdot \Delta s}{\Delta t}$.

F. Adjusting the timing source period and phase

Previous software synchronization methods focused on the detection of a stripe induced on the image due to a particular combination of the camera and stroboscopic lighting parameters [14]. In [14], the stripe is detected as the strongest response of a gradient operator, orthogonal to the stripes. By tracking the stripe over time, the method is capable of adjusting the timing source to minimize the stripe translation (i.e. achieve synchronization). One limitation of this approach is the use of a fixed size kernel, which produces noisy stripe localizations depending on the camera frequency. Another limitation is the adjustment of the timing source using fixed steps, limiting the resolution of the synchrony achieved.

Our method resolves these limitations by first estimating the optimal gradient filter size for any setup (camera, resolution, sampling frequency), improving the stripe detection. We then iterate the timing source adjustment using variable steps proportional to the drift in phase in the given interval, increasing the resolution of the synchrony to the one of the timing source.

G. Using rolling shutter cameras with structured lighting systems

To use a low-cost rolling shutter camera in structured lighting applications, we must configure the camera frequency, exposure and gain, as would be normally necessary for any camera model. The exposure is set according to (8) for the subsequent adjustment of the timing source (Section II-F).

If the camera model is unknown, the driver is checked for UVC compliance. If compliant, β does not need to be calculated, as it is standardized. Then, the total number of lines for the particular frame height is estimated by either the procedure described in Section II-C or II-D. If the camera supports only auto exposure, estimating S is limited to the procedure described in Section II-D. Note that when estimating S for a camera in AE mode, the ambient light must be as such as to keep the exposure at maximum.

Once the timing source is adjusted, the structured lighting application is allowed to run as if a true synchronization from the camera is available, but instead, provided by the timing source. As the clocks are subject to small deviations in frequency over time and also due to the limited resolution of the timing source, drift corrections might be necessary from time to time.

III. Implementation

To estimate the stripe height (h_s) we compute the distance between consecutive edges on the average column image. We assume the intensity profile of an average column image containing a stripe consist of an almost uniform segment. The column image consists of a sequence of intensities $\alpha_0, \alpha_1, ..., \alpha_{N-1}$. The piecewise profile is defined by the $k+1$ edges which segment data with differing behaviors, where $z_0 = 0, z_1, ...z_{k-1}, z_k = N-1$ are the lines on the original image corresponding to the edge positions. The edges define intervals $m_1, ..., m_k$, such that $m_i = [z_{i-1}, z_i)$ follows a linear model M_i.

The goal is to approximate the intensity profile with k piecewise linear functions $M_i(x)$. Assuming k is known, each line segment can be defined as follows:

$$M_i(x) = \frac{\alpha_{z_i} - \alpha_{z_{i-1}}}{z_i - z_{i-1}}(x - z_{i-1}) + \alpha_{z_{i-1}} \quad for \quad z_{i-1} \leq x \leq z_i \tag{9}$$

where $i = 1, \ldots, k$.

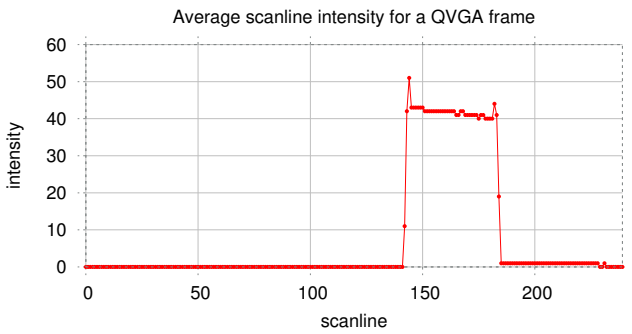


Fig. 6. Average column intensity of a frame captured from a PS3 Eye Camera at 187fps, minimum exposure and using a strobe with length $\Delta s = 800\mu s$.

The following equation is used to compute the position of the edge points $z_1, \ldots, z_{k-1}$, such that the overall squared error e is minimized:

$$\min_{z_1,\ldots,z_{k-1}} \left\{ e = \sum_{x=0}^{N-1} [\alpha_x - M_i(x)]^2 \right\} \tag{10}$$

Figure 6 shows the intensity profile of a frame with minimum exposure subject to stroboscopic illumination. Spikes around the bright scanlines are evidenced, result of the camera post processing to increase sharpness. We will assume every camera employs such a filter, with varying levels of intensity.

To reduce the artifacts introduced by the camera, we use a non-continuous piecewise segmentation technique in which the optimum number of edges is estimated from the data. Using the dynamic programming method from Lemire [19], we are able to detect the correct edges even when the number of edges is overestimated (due to noise). The method uses a mixed model (piecewise linear and constant) which was shown to reduce the fitting error [19].

We considered the camera clock and the timing source to have slightly different periods. This means that the difference $t_0^{(j)} - s^{(j)}$ changes with j. Therefore, the stripe produced seems to move top-down or bottom-up. A frame is useful for the estimation of S only if the whole stripe profile is on the visible scanlines. The adaptive segmentation can then be used on the average column image to estimate the edges that separate the segments, and the polynomial degree of each slice can be used to identify the relevant sections to detect the stripe height. Linear regression is used on a particular segment if it does not agree with the model.

IV. Experimental Results

We used a Sony PS3 Eye [20] camera in our experiments. It is a low cost and high speed camera capable of capturing frames with 240 lines at 187 fps. While higher resolutions are possible at lower speeds, we favored speed against resolution as the synchronization becomes more challenging as the frame period is reduced. The timing source was implemented using an Arduino board [21], a low cost micro-controller board which provided a timer resolution of 62.5 ns adjustable by software.

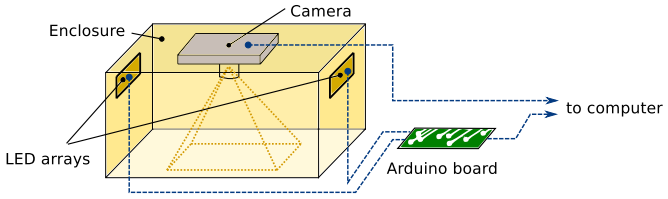


Fig. 7. Setup to assess the estimation methods.

The results are organized in four parts. The first is related to the estimation of the camera parameters, necessary for the software synchronization. The second assess the accuracy of the stripe estimated using a simple filter given the parameters estimated in the first part. The third assess the synchronization, given the camera parameters are known. Lastly, a pupil detection application using structured lighting is presented. Discussions are drawn in each section on the strengths and weakness of the methods.

A. Camera parameters estimation

The PS3 Eye camera provides ground truth for most of its parameters, such as S, D_0 and D_z. However, as it is not UVC compliant, the β is unknown and varies with both the frame rate and frame height.

To avoid taking the camera apart, visible stroboscopic light was used. Two LED lights were arranged on opposite sides of a polystyrene enclosure as seen in Figure 7. The material was used to reflect the light and provide an even illumination. The camera was positioned at an aperture on the enclosure, with the lens facing a wall illuminated from two directions simultaneously.

The procedure from Section II-G was followed to estimate S and β. For the camera at 30 fps and using $\Delta s = 5$ ms, the estimated $S = 270.78 \pm 1.00$ scanlines and $\beta = 2.44e^{-5} \pm 1.32e^{-7}$. At 187 fps and $\Delta s = 2.5$ ms, the estimated $S = 273.54 \pm 1.34$ scanlines and $\beta = 0.29e^{-5} \pm 2.46e^{-7}$. The scanline results represent a deviation from ground truth of 2.59% and 1.60%, respectively. Figure 8 shows the complete results for varying Δs.

Note that independent of the strobe length and frame rate, our implementation seems to underestimate S. To investigate this possible bias of the method, we computed Δs using different exposures, as shown in Figure 9a. The y-intercept of the first order regression on Δs allows the estimation of the scanline period (and then S), while the slope corresponds to β. Note in Figure 9b that the stripe height estimated using $\mathbb{E} = 0$ is more accurate. The underestimation bias seen in Figure 8 can be explained by the non-linearity of the stripe when captured by the host computer, as can be seen in Figure 10. Figure 10 shows a series of stripe profiles obtained with the gain at minimum, where the points represent the edges that define the line segments as computed by the piecewise segmentation. The edges computed for each profile have different colors. Note the stripes differ from Figure 5 in which a single non-zero slope segment is expected. This affects the placement of the edges, which in turn are responsible for the bias observed in Figure 8.

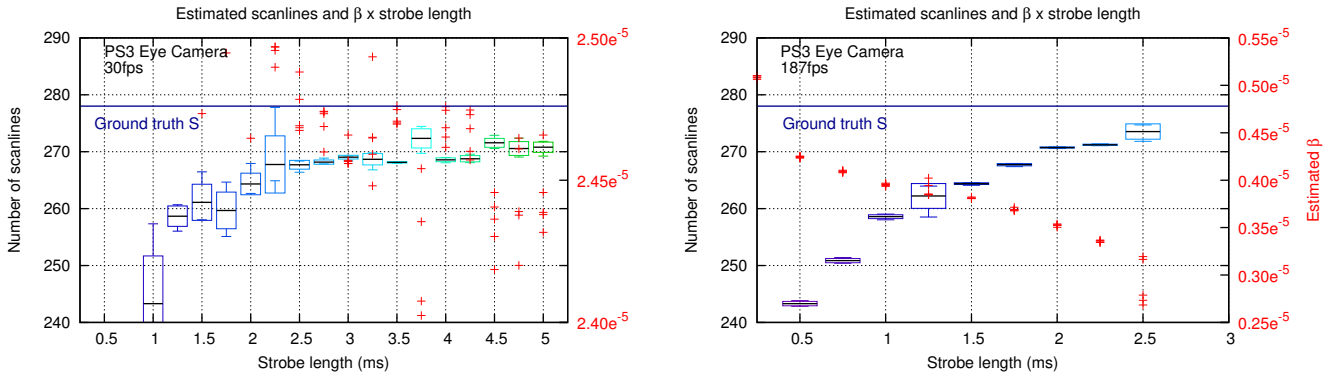
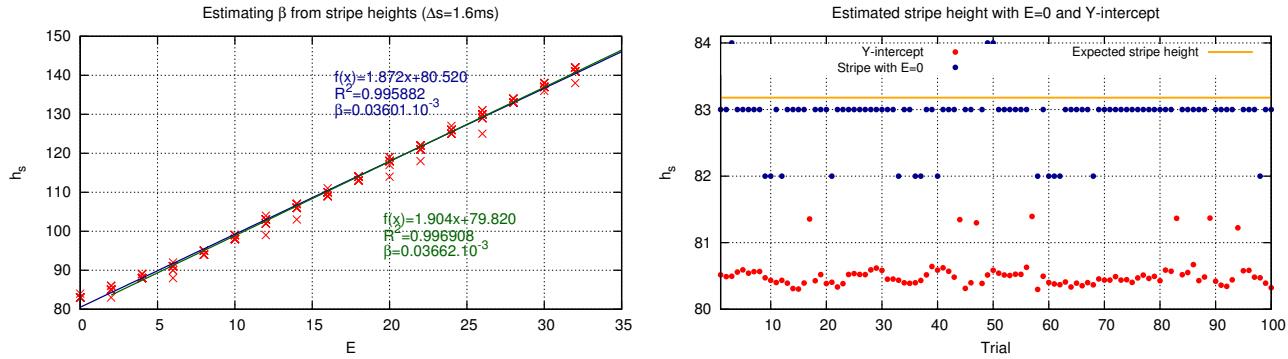


Fig. 8. Estimated total scanlines and β in function of strobe length Δs using the method described in Section II-C. Five trials per Δs.



a) b)

Fig. 9. Estimating β for the PS3 Eye camera at QVGA 187fps. a) First order regression on the stripe height for different exposures (one trial). b) The y-intercept for one hundred trials and the stripe height estimated with $\mathbb{E} = 0$.

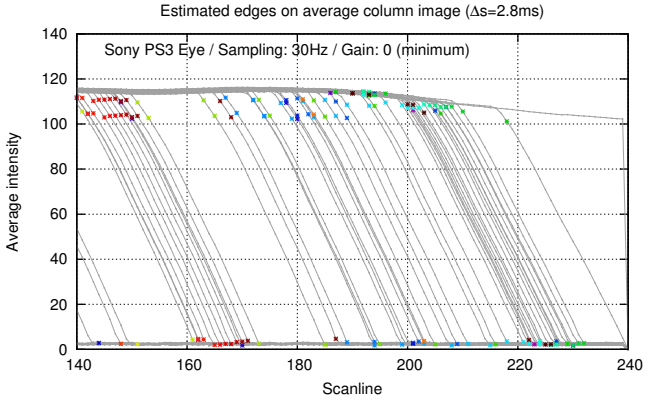


Fig. 10. Estimating the edges using a maximum of seven segments for the PS3 Eye camera with minimum gain. Several stripe profiles from distinct frames shown. The edges computed for each profile are color-coded.

The method described in Section II-D was also tested. It allows the estimation of the scanlines in cameras with the limitation of supporting only auto exposure. For the camera at 30 Hz and $\Delta s = 2.8$ ms, the computed h_s after 50 trials was 21.12 ± 1.36 scanlines, resulting in $S = 251.4$ scanlines. At 187 Hz and $\Delta s = 1.6$ ms, the estimated h_s was 80.63 ± 5.5 scanlines, resulting in $S = 268.9$ scanlines. These results represent a deviation from ground truth of 9.5% and 3.2%, respectively. Note that if we assume a similar error to the one reported in Figure 9b of 2.5 scanlines, the estimated S rises to 277.23, reducing the error down to 0.27%.

B. Stripe detection accuracy

A rolling shutter camera with the exposure configured as in (8) imaging a scene illuminated with stroboscopic lighting produces images with a dark stripe that translates horizontally according to the phase difference between the camera clock and the timing source, that feeds the lighting. Its detection is necessary for the timing source adjustment (Section II-F), as the phase difference is computed from the stripe translation over time. A robust detection method is to convolve the image orthogonally to the stripe orientation with a gradient operator [14]. Moreover, as we are able to predict the stripe height, a custom filter can be exploited to improve the results according to the camera and resolution in use.

To assess the improvements in the position detection accuracy of the stripe, we compared the detected stripe position against ground truth using 704 images of a robust structured lighting pupil detector [14]. The experiment comprehends convolving the images with different kernels, including a fixed size one (an approximation of a derivative of Gaussian of size

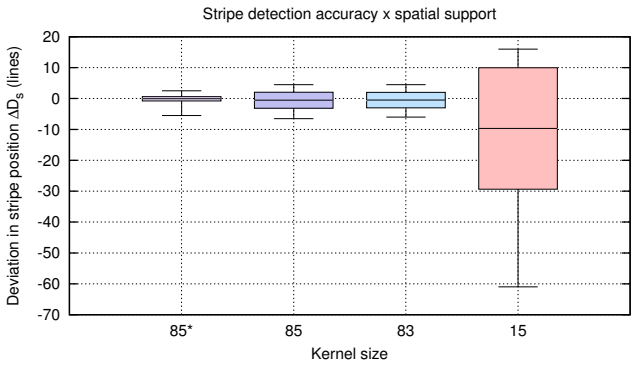


Fig. 11. Estimated stripe positions given different spatial supports. * indicate a kernel with only the two extreme values non-zero.

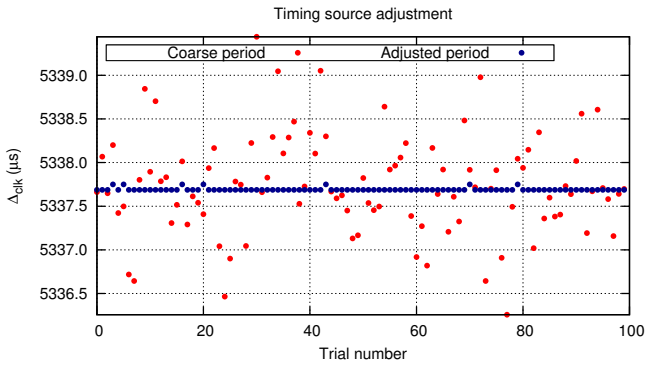


Fig. 12. Adjustment of the timing source on repeated trials for the PS3 Eye camera at 187 Hz.

15 [14]), and a first order function with unitary slope, defined as follows:

$$kernel(i) = i - \lfloor h_s/2 \rfloor \tag{11}$$

where $||kernel|| = 2 \cdot \lfloor h_s/2 \rfloor + 1$.

The convolutions were performed using the average column images, and the boxplots from Figure 11 show the position of the stripes detected as the strongest sign change of the convolution with respect to the ground truth, except for the kernel of size 15, in which the peak was selected. We call this difference ΔD_s. In Figure 11, the kernel sizes of 85 and 83, correspond, respectively, to the expected stripe height using the true number of scanlines (S = 278) and the number estimated using the procedure from Section II-C with $\Delta s = 2$ ms, according to (11).

With the fixed-size kernel, we obtained $\Delta D_s = -9.69 \pm 19.7$ lines, while for the kernels with size 85 and 83, we obtained $\Delta D_s = -0.55 \pm 2.58$ and $\Delta D_s = -0.50 \pm 2.50$ lines, respectively. The results indicate that despite the biased estimation of S, the stripe position estimation using convolution is little affected. The adequate spatial support improves the result with respect to a smaller fixed kernel, as expected. Though the use of large kernels might be computationally expensive, for our particular images, using a kernel with only two non-zero values provided better results ($\Delta D_s = -0.08 \pm 0.69$ lines).

C. Software synchronization

Our method proposes to replace the synchronization signal from a high-end camera by a signal generated by an external clock adjusted by software we call timing source. To evaluate the adjustment, we performed an experiment that consists of adjusting the timing source as described in Section II-F, and then monitor the phase difference over time.

Three volunteers were asked to wear a head-mount pupil detector [14] while the timing source was adjusted. The device employs asynchronous infrared stroboscopic lighting to capture eye images at 187 Hz using a low cost PS3 Eye camera [20]. In total, the experiment was repeated one hundred times. The timing source was initialized with a coarse estimation of the camera period based on the delivery rate to the computer in every trial. Then, the timing source were adjusted by tracking the stripe position. The stop criterion employed was to wait the equivalent in time to 200 frames (a little more than one second at 187 Hz) for one line of stripe translation. Figure 12 shows the results for both the coarse estimation and the adjusted period.

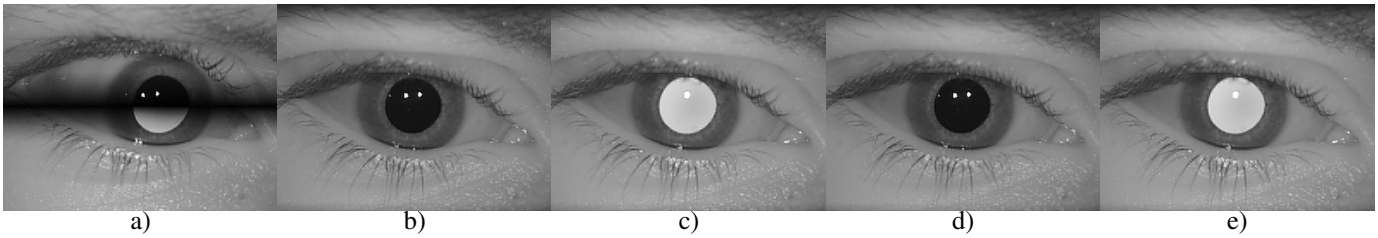
To assess the resulting synchronization, we measured the movement of the stripe produced on the frames over time. On each trial, after adjusting the timing source, we tracked the stripe translation for the equivalent of at least 10 lines, which resulted in about 11 seconds of video data per trial. With respect to the processing time, the stripe detection took on average $405 \pm 35.7 \mu s$. Note that as the stripe translates slowly after the timing source adjustment, the stripe can be detected in intervals of more than one second to reduce computations.

The results of this experiment indicate that, despite the initial biased stripe estimation, the timing source can be accurately synchronized at a low computational load. The slow drifts can be corrected by software adjustments, scheduled to run on large intervals of more than 10 seconds for a camera at 187 Hz.

D. Structured lighting pupil detection

To test the performance of the technique with a real application, we have used our method to build a robust pupil detector. The timing source is responsible for triggering two groups of infrared LEDs that are alternately activated. One group is arranged on-axis with respect to the camera to produce bright pupil images and the other off-axis, to produce dark pupil images, similar to the method presented in [11]. The pupil is detected by determining the high contrast pixels between the dark and bright pupil images.

Short pulses of light are used to improve the image quality as well as to allow only one frame to be illuminated when synchronized with the camera. Before adjusting the timing source, most images look like Figure 13a. After the adjustment, the difference of adjacent frames is used as an estimation of the pupil after thresholding. Note the dark lines at the top of the images in Figures 13b to e, that correspond to part of the stripe. These lines do not compromise the application.



a) b) c) d) e)

Fig. 13. Images of a volunteer using a head-mounted camera with attached structured lighting. a) Image with stripe due to a phase difference between the camera clock and timing source. b-e) Images captured in sequence with the lighting synchronized by software using our method.

V. Conclusions

In this paper we have shown how to build multiplexed structured lighting computer vision systems using low cost digital cameras based on rolling shutters and with no external synchronization mechanism. The proposed technique uses short light pulses that are, most likely, captured during a single frame. The pulses are triggered by a software-controlled timing source, which combined with a particular camera exposure, produce a visible dark stripe in the frames. Our method detects the stripe that is used to adjust the timing source. Because the synchronization is made by software, one advantage of our technique is that it can be used with most modern off-the-shelf digital cameras that do not offer external synchronization mechanisms. Another advantage of our technique is the reduction of motion blur due to the use of short ligh pulses.

In an experiment using a low cost PS3 Eye Camera at 187 Hz, the stroboscopic lighting was successfully adjusted to the camera frame rate. The phase difference between the camera and the timing source was responsible for no more than two lines of stripe movement per second in frames with 240 lines, i.e., less than 1% drift per second.

We have demonstrated the use of the technique in an active lighting pupil detector application that requires explicit synchronization of two light sources and the camera frames. The use of active lighting in this case greatly simplifies the image processing task.

References

[1] J. Salvi, S. Fernandez, T. Pribanic, and X. Llado, "A state of the art in structured light patterns for surface profilometry," *Pattern Recognition*, vol. 43, no. 8, p. 26662680, 2010.

[2] J. Salvi, J. Pags, and J. Batlle, "Pattern codification strategies in structured light systems," *Pattern Recognition*, vol. 37, no. 4, p. 827849, 2004, agent Based Computer Vision.

[3] P. Agrafiotis, A. Georgopoulos, A. Doulamis, and N. Doulamis, "Precise 3D measurements for tracked objects from synchronized stereo-video sequences," *Lecture Notes in Computer Science*, pp. 757–769, 2014.

[4] A. Whitehead, R. Laganiere, and P. Bose, "Temporal synchronization of video sequences in theory and in practice," in *WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, vol. 2. IEEE, 2005, pp. 132–137.

[5] B. Evans, B. Parslow, D. O'Carroll, and S. Wiederman, "Quantifying asynchrony of multiple cameras using aliased optical devices," in *Image Processing Theory, Tools and Applications (IPTA), 2015 International Conference on*, Nov 2015, pp. 567–572.

[6] L. Hou, S. Kagami, and K. Hashimoto, "Frame synchronization of high-speed vision sensors with respect to temporally encoded illumination in highly dynamic environments," *Sensors*, vol. 13, no. 4, pp. 4102–4121, 2013.

[7] C. Theobalt, I. Albrecht, J. Haber, M. Magnor, and H.-P. Seidel, "Pitching a baseball: tracking high-speed motion with multi-exposure images," *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3, pp. 540–547, 2004.

[8] D. Bradley, B. Atcheson, I. Ihrke, and W. Heidrich, "Synchronization and rolling shutter compensation for consumer video camera arrays," in *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*. IEEE, 2009, pp. 1–8.

[9] J.-I. Park, M.-H. Lee, M. D. Grossberg, and S. K. Nayar, "Multispectral imaging using multiplexed illumination," in *ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.

[10] A. Wenger, A. Gardner, C. Tchou, J. Unger, T. Hawkins, and P. Debevec, "Performance relighting and reflectance transformation with time-multiplexed illumination," in *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3. ACM, 2005, pp. 756–764.

[11] C. H. Morimoto, D. Koons, A. Amir, and M. Flickner, "Pupil detection and tracking using multiple light sources," *Image and vision computing*, vol. 18, no. 4, pp. 331–335, 2000.

[12] A. El Gamal, "High dynamic range image sensors," in *Tutorial at International Solid-State Circuits Conference*, vol. 290, 2002.

[13] S. Kavusi and A. El Gamal, "Quantitative study of high-dynamic-range image sensor architectures," in *Electronic Imaging 2004*. International Society for Optics and Photonics, 2004, pp. 264–275.

[14] F. Borsato, F. Aluani, and C. Morimoto, "A Fast and Accurate Eye Tracker Using Stroboscopic Differential Lighting," in *Computer Vision Workshop (ICCVW), 2015 IEEE International Conference on*, 2015, pp. 110–118.

[15] OmniVision Technologies, Inc., *OV7725 Color CMOS VGA OmniPixel2TM CAMERA CHIP Sensor*, Mar. 2007.

[16] ON Semiconductor, *AR0130: 1/3-Inch CMOS Digital Image Sensor*, Nov. 2014. [Online]. Available: http://www.onsemi.com/pub/Collateral/AR0130CS-D.PDF

[17] Texas Instruments Incorporated, *SN65LVDS324 1080p60 Image Sensor Receiver*, Mar. 2015. [Online]. Available: http://www.ti.com/lit/ds/sllsed9a/sllsed9a.pdf

[18] USB Implementers Forum, "Universal Serial Bus Device Class Definition for Video Devices," p. 130, Jun 2005.

[19] D. Lemire, "A Better Alternative to Piecewise Linear Time Series Segmentation," in *SIAM Data Mining 2007*, 2007.

[20] C. Sony, "PlayStation Eye Camera," 2014. [Online]. Available: http://us.playstation.com/ps3/accessories/playstation-eye-camera-ps3.html

[21] Arduino.cc, "Arduino leonardo webpage," [Online, 10/jun/2017], 2017. [Online]. Available: http://arduino.cc/en/Main/arduinoBoardLeonardo