Distributed Smart Camera Calibration using Blinking LED

Michael Koch[1,2], Zoran Zivkovic[1], Richard Kleihorst[1] and Henk Corporaal[2]

[1] NXP Semiconductor Research, High Tech Campus 32, 5656 AE Eindhoven, The Netherlands
{michael.a.koch, zoran.zivkovic, richard.kleihorst}@nxp.com
[2] University of Technology, Den Dolech 2, 5612 AZ Eindhoven, The Netherlands
{m.a.koch@student.tue.nl, h.corporaal@tue.nl}

Abstract. Smart camera networks are very powerful for various computer vision applications. As a preliminary step in the application, every camera in the scene needs to be calibrated. For most of the calibration algorithms, image point correspondences are needed. Therefore, easy to detect objects can be used like LEDs. Unfortunately, existing LED based calibration methods are highly sensitive to lighting conditions and only perform well in dark conditions. Therefore, in this paper, we propose a robust LED detection method for the calibration process. The main contribution to the robustness of our algorithm is the *blinking* behavior of the LED, enabling the use of temporal pixel information. Experiments show that accurate LED detection is already possible for a sequence length of three frames. A distributed implementation on a truly embedded smart camera is performed. Finally, a successful spatial calibration is performed with this implemented method.

Key words: Blinking LED Detection, Distributed Calibration

1 Introduction

Due to the continuously falling prices of cameras and computing elements, the combination of these two elements on one platform, a *smart camera*, becomes more and more interesting. The resulting vision system is capable of real-time extracting information from captured images out of the scene and sharing the results to the outside world by means of (wireless) communication while its power can be drawn from a battery. Deploying a smart camera network can be very helpful for certain applications because of the different scene observations of the cameras. For a proper use of the application, all the cameras in the scene must know the (relative) location and rotation of all scene cameras with respect to a chosen origin. Also, the internal orientations of all cameras must be known. Hence, we must preliminary *calibrate* the scene cameras. During camera calibration, the intrinsic and extrinsic camera parameters are estimated [1–3].

In general for a multi-camera setup, one has to find matching pairs of image locations from different cameras of 3D world points or lines to perform the

calibration process [1, 2, 4–6]. Another approach to this calibration process is the use of dynamic silhouettes, like walking persons [7]. We will focus on the point or line correspondence search. These points or lines can include e.g. edges or corners. The estimation of the image locations of these lines or points must be as accurate as possible, preferably sub-pixel level, for reliable calibration results. To succeed in the calibration procedure, the scene must contain enough points or lines to detect, but unfortunately, they will generally differ from camera to camera because of the different viewing angles. Also, not all scenes will result in enough points or lines to detect, making calibration impossible. Therefore, a light source (e.g. a LED) as easy to detect scene object can be used, assuming we do not have an uncommon critical camera configuration [8–10]. Several LED based camera calibration methods already exist. Unfortunately, from our experiments, these methods are highly sensitive to lighting conditions. As a result, they only give a low number of false detections if performed in darkness. Hence, these methods are not well suited for a robust calibration procedure.

We present an approach to the LED detection robust against lighting conditions, enabling camera calibrations in both indoor and outdoor setups. The LED will blink at a certain rate to contribute to the robustness of the algorithm. The blinking pattern of the LED can be e.g. square-wave, sinusoidal or of some other (unique, for identification [11]) form. We will use the square-wave pattern because we initially use a single LED and hence do not yet require identification.

Basic extrinsic camera calibration is described first in Chap. 2. Then, we analyze different choices for the calibration object in Chap. 3. Blinking LED detection and implementation on a smart camera are described in Chaps. 4 and 5, respectively. In Chap. 6, experimental LED detection results with two different pattern recognition metrics and three different LED blinking frequencies are presented. Finally, an extrinsic calibration of a smart camera network is successfully performed.

2 Extrinsic Camera Calibration Procedure

The extrinsic camera parameters consist of the pose of the camera with respect to a reference coordinate system. In a camera network, one camera can be chosen as the reference and the poses of other cameras can be determined with respect to the reference camera. Because the poses of the cameras in a N-camera network, with $N > 2$, can be easily obtained by combining the appropriate pairwise poses, we will focus here on the relative pose between two cameras for simplification.

A straightforward parametrization of the relative pose is to use three values for the rotation angles and another three values for the translation in the x, y and z directions. The translation is described by a three dimensional vector T and the rotation angles can be used to define a 3×3 rotation matrix R.

The common starting point is that n landmarks, for which the 3D positions are unknown, are observed in both camera images. The resulting image point correspondences are denoted by two sets of 2D image points $\{\mathbf{x_i} \leftrightarrow \mathbf{x'_i},\ i = 1 \dots n\}$.

The problem of finding the relative pose of the cameras and the positions of the 3D landmarks is an optimization problem with $6 + 3n$ parameters (6 for the extrinsic pose parameters plus 3 for each landmark). By assuming Gaussian noise, a Maximum Likelihood estimate of the parameters can be obtained. However, this optimization problem is non-linear and hard to solve. It is possible to directly find a solution that approximates the Maximum Likelihood solution by using the so called essential matrix and e.g. the *Normalized Eight Point Algorithm* [1, 4, 6] or the *Five Point Algorithm* [12].

The essential matrix is the algebraic representation of the intrinsic projective geometry between two views. The essential matrix E is a 3×3 matrix of rank two. If a 3D space point $\mathbf{X}$ is imaged as $\mathbf{x}$ on the image plane of the first camera and as $\mathbf{x'}$ in the second camera's image plane, then the image points satisfy the relation $\hat{\mathbf{x}}'^T E \hat{\mathbf{x}} = 0$ where $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are the normalized image coordinates of $\mathbf{x}$ and $\mathbf{x}'$, respectively. The optimal E is computed after which the relative rotation and translation can be recovered from this matrix [5]. The reconstruction is up to a scale factor since the same scene if scaled by a scale factor gives the same 2D image projections of the 3D points. To recover the scale of the scene additional information is needed about certain metric scene distances, for example the distance between two points in the scene.

3 Object Choice

It is difficult to calibrate a camera network automatically since it is hard to match image points or lines between very different views. For robustness, we require scene adaption from the user who will put in a scene object, creating easy to detect points or lines in the camera's viewing frustums.

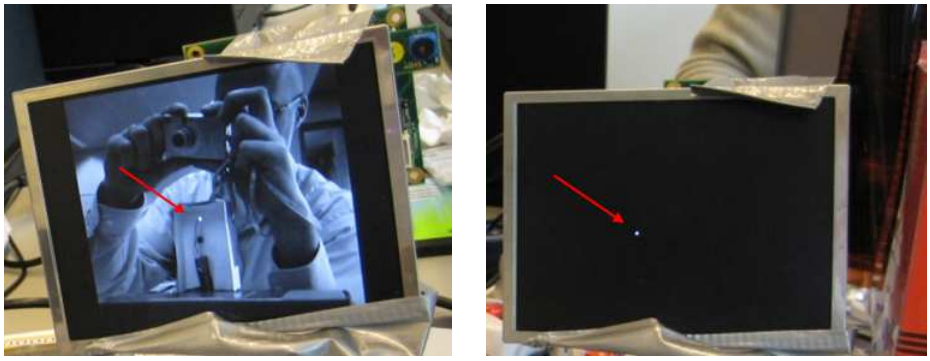
The use of an object with known lengths has the advantage that the scale factor of the calibration process can be eliminated. But, such an object does not scale, i.e. the object will appear very small in the camera image plane if the distance to the camera sensor is too far, giving high length estimation errors. A possible solution is to use a bigger, scaled version of the object, but that is not very user friendly. Another solution is to make sure the distance between the object and the camera is bounded, but this is not practical.

A light source, in contrast, is easily detected by multiple cameras as one 3D scene point, assuming that the radiation angle and radiant intensity of the light source are high enough. The distance to the camera sensor now is not an issue anymore, because we can fit an appropriate point spread function (e.g. Gaussian), resulting in a sub-pixel precise estimation of the light source centre.

An incandescent light bulb can be used for this purpose, giving good results at far distances. However, due to its high energy consumption and heat production, it does not lend itself for a portable, battery-powered device. A LED (Light Emitting Diode), for decennia available in a variety of colors and models (e.g. flashing), is better suited for battery use due to its low energy consumption. If we assume the radiation angle and the intensity output power of the LED are high enough, a LED will perform perfectly as a camera calibration tool [13]. For

robustness, it is preferable that we use an easy-to-detect LED. An infrared LED color can therefore be a proper choice. Every digital camera not equipped with an infrared light blocker is sensitive to this invisible light.

Existing LED detection methods for calibration do not work properly due to their high sensitivity for lighting conditions. An approach to this challenge is to use a visible light filter in front of the camera to suppress all the visible light, see Fig. 1. In Fig. 1a, no filtering is applied, but in Fig. 1b, a couple of developed color films of an analog camera are held against the camera lens as a simple but good working spectral high-pass infrared filter. As one can see, all the visible light is suppressed and only the infrared light spot is visible after filtering. Unfortunately, because the camera should also be used for the application which can need the complete visible light spectrum, this light filter is not desirable. After calibration, the light filter should be removed; in a camera network where the cameras can be hard to reach, this definitely is unpractical. Therefore, in this paper, we analyse the robust detection of a *blinking* LED for the calibration process, so that we can omit the use of the unpractical light filter in combination with the existing non-blinking LED detection methods.



a. Without IR filter b. With IR filter against camera lens

Fig. 1. Scene containing IR LED. At the back, there is a smart camera containing one camera while at the front, the image plane of that camera is displayed on the LCD display.

4 Blinking LED Pattern Recognition

For the blinking LED detection, we start with a sequence of N observed pixel intensity values $\mathbf{p} = [p_1\ p_2\ \ldots\ p_N]$. We assume that the LED is moving slowly with respect to the camera's frame rate, i.e. we assume to have at least N measurements of the LED at the same pixel (actually it may be 2x2 pixels, if we would downscale from VGA to QVGA). After a frequency domain conversion and an alias term elimination in Sect. 4.1, we calculate the distance of the spectral

components with a signature in Sect. 4.2. Finally, a classifier function q is defined which creates a Boolean output *true* if a valid LED sequence was observed.

4.1 Frequency Analysis

Because the LED is blinking, the intensity values of the LED containing pixels will continuously alternate in magnitude. As a result, a frequency pixel analysis can be performed, having the advantage that it is robust to changing light conditions (which only will result in a scaling of the spectral component amplitudes).

The elements p_k of the time-domain pixel intensity sequence are discrete-time samples of the time-continuous signal $\tilde{p}(t)$ via the relation $p_{(k+1)} = \tilde{p}\,(kT_s)$, where $k = 0, 1, \ldots, (N-1)$, with sampling frequency $\omega_s = 2\pi f_s = \frac{2\pi}{T_s}$ and f_s equal to the camera's frame rate. According to Nyquist-Shannon's sampling theorem, we assume $f_s \geq 2f_{LED}$, where f_{LED} equals the blinking frequency of the LED.

We will use the N-point Discrete Fourier Transform per pixel sequence for frequency analysis. As a result, we will convert the discrete time, periodic pixel intensity sequence into a discrete frequency, periodic sequence as

$$\mathbf{P} = \left[\, \mathcal{F}_* \left([p_i]_{i=1,2,\ldots,N} \right) \right] = \left[\, P_1 \; P_2 \; \ldots \; P_N \,\right] \quad , \tag{1}$$

where $\mathcal{F}_*$ depicts the discrete Fourier transform. The angular frequencies of the sequence's individual spectral components equal

$$\omega_{P_i} = \frac{2\pi\,(i-1)}{NT_s} \quad . \qquad i = 1, 2, \ldots, N \tag{2}$$

Because the camera pixel samples cannot be infinitely small in practice, instead of $\tilde{p}(t)$, a 'spread out' version $a(t) * \tilde{p}(t)$ will be sampled [14], with $a(t)$ equal to

$$a(t) = \frac{1}{\tau} rect\left(\frac{t}{\tau}\right) = \begin{cases} \frac{1}{\tau}, |t| \leq \frac{\tau}{2} \\ 0, \text{ elsewhere} \end{cases} \quad , \tag{3}$$

where τ, which must be smaller than the sampling interval T_s, equals the pixel integration time of the camera sensor used. As a result, every frequency element of (1) will be multiplied with the Fourier transformed representation of (3), i.e.

$$A(j\omega) = sinc\left(\frac{\omega\tau}{2}\right) = \begin{cases} \frac{2sin\left(\frac{\omega\tau}{2}\right)}{\omega\tau}, \omega\tau \neq 0 \\ 1, \qquad \omega\tau = 0 \end{cases} \quad . \tag{4}$$

We can hence state that in the practical case, (1) becomes

$$\mathbf{P}' = \left[P_i \; sinc\left(\frac{\omega_{P_i}\tau}{2}\right) \right]_{1 \leq i \leq N} \quad . \tag{5}$$

The time-domain input values are real, so the following frequency symmetry rule holds for the frequency sequence: $P'_{(N-i)} = P'^{*}_{(i+2)}$, $i = 0, 1, \ldots, (N-2)$.

Because we are not going to use any phase information and only need the non-DC spectral value magnitudes, we may discard the redundant numbers in (5). As a result, feature vector $\mathbf{P'}$ reduces approximately 50%. The final resulting feature vector containing only unique values equals

$$\mathbf{P''} = \left[|P_i| sinc\left(\frac{\omega_{P_i}\tau}{2}\right)\right]_{2 \leq i \leq \lfloor \frac{N}{2} \rfloor + 1} \quad . \tag{6}$$

4.2 Classification

In this paper, we will analyze the (easy) Euclidean and the (more complicated, but possibly better result giving) Mahalanobis distance measure functions for determining distance d between $\mathbf{P''}$ and $\mathbf{P_s} = \left[P_{s_1} \; P_{s_2} \; \ldots \; P_{s_{\lfloor \frac{N}{2} \rfloor}} \right]$ which equals the signature vector, i.e. the mean values of the individual frequency components of all possible LED containing pixels. To be more precise, every mean element P_{s_j} equals the mean value of column j of a $M \mathrm{x} \lfloor \frac{N}{2} \rfloor$ sized dataset matrix $\mathbf{X}$, containing all possible LED pixel intensity sequences (with one sequence per row), as

$$P_{s_j} = \frac{1}{M} \sum_{i=1}^{M} X_{ij} \quad , \qquad j = 1, 2, \ldots, \left\lfloor \frac{N}{2} \right\rfloor \tag{7}$$

where X_{ij} equals the element of matrix $\mathbf{X}$ at row i and column j.

The Euclidean distance equals the "ordinary" distance between two points, whereas the Mahalanobis distance is a distance metric which is scale-invariant (i.e. not dependent of measurement scaling) and which takes into account dataset correlations; to this purpose, we also need the covariance matrix Σ of $\mathbf{X}$.

After distance d is determined, classifier function q is defined as:

$$q(d) = \begin{cases} 1 = true, & d \leq th \\ 0 = false, & d > th \end{cases} \quad . \tag{8}$$

with th the internally constant, optimized threshold value. When this threshold is varied, we can tighten or loosen the decision if a valid LED pixel sequence was detected.

5 Smart Camera Implementation

In this chapter, we describe the architecture of the used smart camera and how the algorithm from the LED detection can be implemented on this platform.

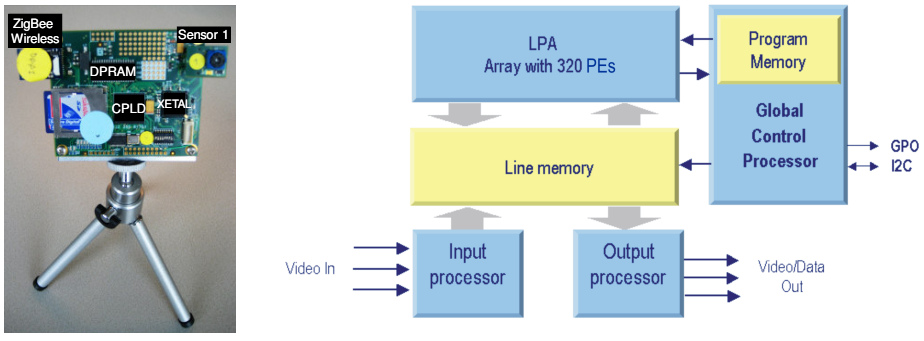
5.1 WiCa

WiCa stands for Wireless Camera and this wireless smart camera 'mote' is built in such a way that it can operate stand-alone or in a network of cameras [15].

The camera consists of basically four components; one or two VGA color image sensors, the Xetal IC3D SIMD processor for low-level image processing [16], a general purpose processor for intermediate and high-level processing and control, and a communication module. Both processors are coupled using a Dual Port RAM (DPRAM) that enables them to work in a shared workspace on their own processing pace, see Fig. 2a.

Xetal IC3D Processor The IC3D is a member of the NXP Xetal family of SIMD processors, see Fig. 2b. The video input and output processors are capable of streaming in and out 3 digital video signals to the internal memory. The heart of the chip is formed by the Linear Processor Array (LPA) with 320 Processing Elements (PEs). Each of these PEs have simultaneous read and write access within one clock-cycle to memory positions in the parallel memory. Both the memory address and the instruction of the PEs are shared in SIMD sense. All PEs can also read the memory data of their left and right neighbors directly. At the extremes of the linear array, the inputs and outputs of the PEs are optionally coupled or mirrored. The PEs have downloadable instructions ranging from arithmetic and single-cycle multiply-accumulate to compound instructions. In addition to these, there are conditional guarding instructions, enabling data-dependent operations. Data paths are 10-bits wide. Each PE has two word registers and a flag register.

The line memory stores 64 lines of 3200 bits. Pixels of the image lines are placed in an interlaced way on this memory. So QVGA (320x240) images result in 1 pixel per processor, VGA (640x480) in 2 pixels per processor, etc. The GCP (Global Control Processor) is a processor dedicated to control the IC3D and to do some global DSP operations on the data. It takes care of video synchronization, program flow and also communicates with the LPA and the outside world.



(a) (b)

Fig. 2. Wireless smart camera development board (WiCa 1.1, (a)) and the "IC3D" architecture (b), a member of the NXP "Xetal" family of SIMD chips. The WiCa board has dimensions 8x7cm, is powered by 4 AA batteries and also has a second camera sensor port not used here.

5.2 Implementation

The algorithm to be implemented will work in a detect-and-track way. First, the camera's image plane is searched for the blinking LED. After this search, the LED position is tracked.

We assume a sequence length of $N = 8$ and QVGA video mode. Due to the memory size constraint we could not store 8 complete frames; therefore we store eight half frames of 160x240 pixels into DPRAM. This action is performed real-time. Then, from each stored half frame, we read 3 videolines from DPRAM and put them in internal linememories which are next searched for the LED. Reading all 8 half frames from DPRAM takes two frametimes, mainly because of the latency of the DPRAM access, whereas a single 320 pixel linememory LED search takes approximately 15% of the available IC3D processor time per videoline. If the LED was not found in the first half of the QVGA space, the same procedure is followed for the second half of the QVGA space. As a result, the LED detection of the complete QVGA space takes at most $(8+2)\cdot 2 = 20$ frames which equals, with a 30 fps frame rate, 0.67 seconds. The total IC3D resource usage equals approximately 52% program memory, 74% registers, 42% coefficients and almost all linememories. Because we actually work in VGA mode but downscale to QVGA, where some filtering is applied, little movement of the LED during detection is allowed. Also, if the used processor was a Xetal II [17] with its 2048 linememories, we could omit the use of DPRAM, saving in total 12 frametimes for the complete QVGA LED search which then takes 8 frames which equals, with a 30 fps frame rate, 0.27 seconds.

LED position tracking is currently performed by a search around the previous found LED position(s) where the 160x240 pixel window is horizontally centered around the previous found LED position.

If the LED is successfully found and currently been tracked, a ready signal will be passed to the other smart camera(s). If every camera is ready, the capturing will be started in a synchronized way where the positions of the LED at equidistant times will be saved per smart camera. We assume the calibrating scene cameras are a priori temporal synchronized with respect to each other. After storing the needed amount of positions, these locations can be read out and used, making use of the camera network, to perform the calibration process in a distributed way.

6 Experimental Results

We performed experiments with a WiCa smart camera, containing the IC3D Xetal processor and a CMOS VGA camera with a 30fps frame rate. The blinking frequencies of the IR LED were equal to $\{\frac{30}{4}, \frac{30}{3}, \frac{30}{2.2}\}$ Hz. This way we obeyed the Nyquist-Shannon theorem, which states that $f_{LED} \leq \frac{30}{2}$ Hz. Because one period of blinking of the LED takes $\frac{f_s}{f_{LED}} = \frac{30}{f_{LED}}$ frames, the minimum pixel sequence lengths equal $N_{min} = 4$ for $f_{LED} = \frac{30}{4}$ Hz and $N_{min} = 3$ for the other two blinking frequencies. For the LED dataset creation the LED was held at

different distances to the camera sensor with different lighting conditions and camera settings like brightness and contrast. These distances ranged from 48 cm till 2.33 m.

6.1 Classification Experiments

We created two datasets $\mathbf{x}$ and $\mathbf{y}$ containing M_x and M_y time-domain pixel intensity sequences with sequence length N respectively, where we require M_x and M_y to be high enough for giving accurate estimation results. Dataset $\mathbf{x}$ contains blinking LED sequences and dataset $\mathbf{y}$ random (non-LED) sequences (50% taken from dynamic scenes to represent a random pixel as close as possible). Next, we split dataset $\mathbf{x}$ in $\mathbf{x_1}$ and $\mathbf{x_2}$, resulting in a training and a test dataset, containing M_{x_1} and M_{x_2} sequences, respectively. All odd observation sequences are placed in $\mathbf{x_1}$ and all even sequences are stored in $\mathbf{x_2}$. In our experiments, datasets $\mathbf{x}$ and $\mathbf{y}$ contain thousands of sequences.

Covariance Matrix and Signature Vector Estimation To determine the values of the covariance matrix Σ and the signature vector $\mathbf{P_s}$, from training dataset $\mathbf{x_1}$, the equivalent frequency domain representation $\mathbf{X_1}$ is obtained by making use of (6). Because we will use a CMOS camera sensor with a very low pixel integration time which results in a negligible low τ value, all the *sinc* terms in (6) can be approximated by one. Now, from $\mathbf{X_1}$, the covariance matrix can be calculated. Also, $\mathbf{P_s}$ can be calculated with (7).

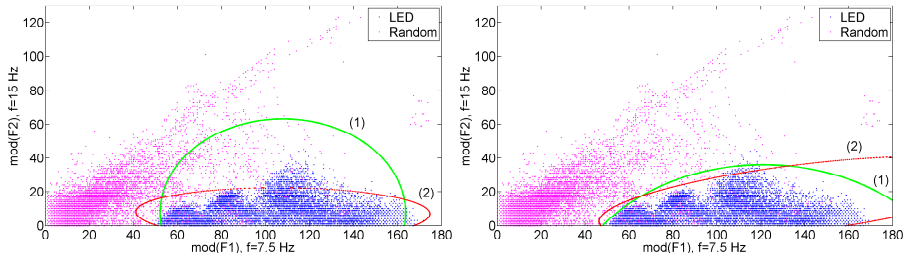
Classification Now, we can determine all the distances d between the frequency representation of all the sequences of test dataset $\mathbf{x_2}$ and the signature vector with the Euclidean or Mahalanobis distance function. This frequency representation is obtained by making use of (6) and equals $\mathbf{X_2}$. If we then input all distances into function q in (8), with a to be decided optimal threshold value, and calculate the true positive rate, this number should be close to unity. We can define this *True Positive Rate* as

$$TPR_{_{LED}} = \frac{1}{M_{x_2}} \sum_{i=1}^{M_{x_2}} q\left(dist\left(\left[X_{2_{i1}}, X_{2_{i2}}, \ldots, X_{2_{i\left\lfloor \frac{N}{2} \right\rfloor}} \right], \mathbf{P_s} \right) \right) \quad . \tag{9}$$

Likewise, the *False Positive Rate* of the frequency representation of the random dataset $\mathbf{Y}$ should be close to zero. This rate is defined as

$$FPR_{_{random}} = \frac{1}{M_y} \sum_{i=1}^{M_y} q\left(dist\left(\left[Y_{i1}, Y_{i2}, \ldots, Y_{i\left\lfloor \frac{N}{2} \right\rfloor} \right], \mathbf{P_s} \right) \right) \quad . \tag{10}$$

When we vary threshold th of (8), these both rates will change. For every threshold value, the set (FPR,TPR)=(x,y) depicts an unique point in the two-dimensional *Receiver Operating Characteristic (ROC)* curve. We will use this binary classification model to find the optimal threshold value satisfying our needs.



a. Without optimization b. With non-linear optimization

Fig. 3. Spectral components LED and random pixel sequence, $f_{LED} = 7.5$ Hz, $N = 4$. The green circle (1) depicts the optimal Euclidean distance whereas the red ellipse (2) depicts the optimal Mahalanobis distance.

Blinking LED Pattern Recognition For $f_{LED} = 7.5$ Hz and $N = 4$, we have exactly two unique frequency components F_1 and F_2 per pixel frame sequence, enabling 2D visualization of the LED pixel sequence and the random sequence; see Fig. 3a. Also, the optimum threshold values of both distance metrics are plotted. Notice that both the circle and ellipse have the same centre which is the calculated mean value of both frequency components. Also note that e.g. for $f_{LED} = 7.5$ Hz and $N = 8$, there are 4 unique frequencies.

Euclidean and Mahalanobis Distance In our case, the optimum threshold value equals the value for which we achieve maximum accuracy in the ROC binary classification. This accuracy equals the proportion of the true results, both true positives and true negatives. As a formula,

$$ACC = \frac{TPR_{LED} M_{x_2} + (1 - FPR_{random})\, M_y}{M_{x_2} + M_y} \quad . \tag{11}$$

As a result, the maximum accuracy of the ROC curves of all three blinking frequencies for both distance metrics and all possible $N \leq 15$ are summarized in the three plots of Fig. 4.

Metric Comparison and Discussion We can observe that Euclidean outperforms Mahalanobis due to the optimization for *only* the threshold value. When we would also optimize for the circle centres, better results can be expected; i.e. that Mahalanobis will perform as least equally well as Euclidean. For the given example with the 2D visualization, a non-linear optimization is performed, see Fig. 3b. Indeed, higher accuracy values were obtained, but because this optimization was very time-consuming on a state-of-the-art PC already for $N = 4$, this optimization is not further elaborated.

Accurate LED detection is already possible for small N with both distance metrics. In the search for the lowest N with an unfailing classifier, we see that this holds for $f_{LED} = 7.5$ Hz, $N = 8$ and the Euclidean distance metric.

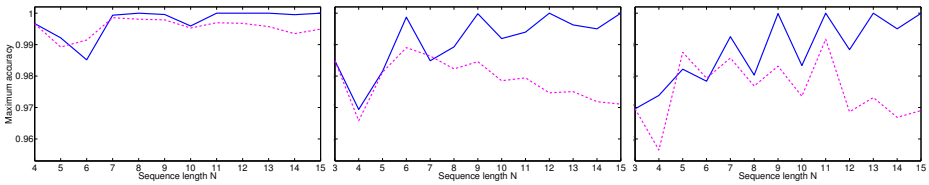
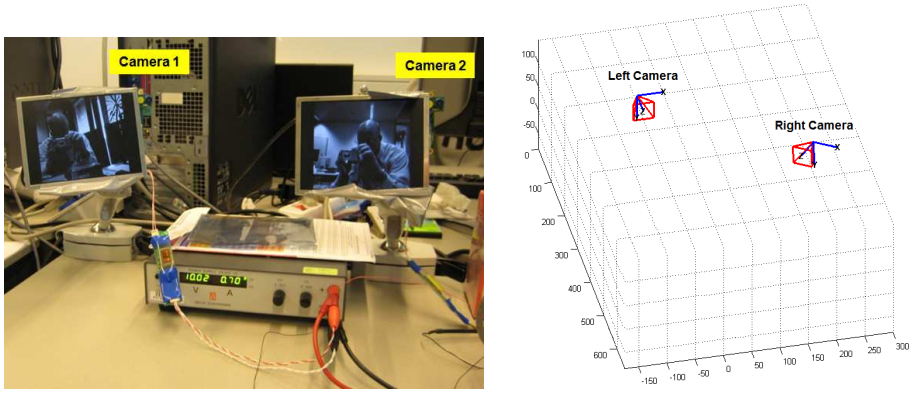


Fig. 4. Non-optimized maximum accuracy functions for LED blinking frequencies $\frac{30}{4}$ Hz (left), $\frac{30}{3}$ Hz (middle) and $\frac{30}{2.2}$ Hz (right) for all possible $N \leq 15$. The blue solid lines and the dashed magenta lines depict the Euclidean and Mahalanobis distance metrics, respectively.

6.2 Spatial Two Camera Calibration

A spatial two camera calibration is performed using the method described in this paper. See Fig. 5 for the setup and the estimated extrinsic 3D scene.



a. Scene containing two smart cameras

b. Reconstructed scene

Fig. 5. Two-camera extrinsic calibration process with the real and estimated scene.

7 Conclusion

A method for the calibration of a smart camera network was presented in this paper. For robustness, a blinking infrared LED was used as a scene object. For the blinking LED pattern recognition, we performed a pixel sequence frequency analysis using Euclidean and Mahalanobis distance metrics as cost functions. Various LED blinking frequencies are analyzed. Experiments show that for a low sequence length $N \in \{3, 4\}$, an accurate blinking LED detection is possible using both metrics with all investigated LED blinking frequencies. If we want an unfailing LED detector with the lowest possible N, the best result from our

experiment is given by letting f_{LED} equal 7.5 Hz with $N = 8$, using the Euclidean distance metric. We notice that better results could be obtained if we optimize not only for the threshold value, but also for the mean values of the spectral components. A distributed implementation on an embedded smart camera is performed. Finally, for a two-camera setup, a successful extrinsic calibration is performed using this implementation.

References

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Second Edition. Cambridge University Press, Cambridge, UK, 2003.

[2] A. Barton-Sweeney, D. Lymberopoulos, and A. Savvides, "Sensor Localization and Camera Calibration in Distributed Camera Sensor Networks", in *Broadband Communications, Networks and Systems*, pp. 1-10, 2006.

[3] Matlab Camera Calibration Toolbox, Caltech Computational Vision, *http://www.vision.caltech.edu/bouguetj/calib_doc/*, July 2008.

[4] R. Hartley, "In Defense of the Eight-Point Algorithm", in *IEEE Tr. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580-593, June 1997.

[5] B. Horn, "Relative Orientation", in *International Journal of Computer Vision*, vol. 4, no. 1, pp. 59-78, 1990.

[6] H.C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene From Two Projections", *Nature*, vol. 293, pp. 133-135, 10 Sept 1981.

[7] S.N. Sinha,M. Pollefeys and L. McMillan, "Camera Network Calibration from Dynamic Silhouettes", in *IEEE Proceedings of Computer Vision and Pattern Recognition*, vol. 1, pp. I-195–I-202, 2004.

[8] T. Svoboda, D. Martinec and T. Pajdla, "A Convenient Multi-Camera Self-Calibration for Virtual Environments", in *Presence: Teleoperators and Virtual Environments*, vol. 14, issue 4, August 2005.

[9] H.C. Longuet-Higgins, "The Reconstruction of a Scene From Two Projections - Configurations That Defeat the Eight-Point Algorithm", in *IEEE Proceedings of the First Conference on Artificial Intelligence Applications*, December 1984.

[10] S.J. Maybank, "The Projective Geometry of Ambiguous Surfaces", in *Ph. Tr.: Phys. Sc. and Eng.*, vol. 332, no. 1623, pp. 1-47, July 16, 1990.

[11] H. Yamazoe, A. Utsumi et al., "Geometrical and Temporal Calibration of Multiple Cameras by Using LED Markers for Image Synthesis", in *ICAT*, 2004.

[12] D. Nister, "An Efficient Solution to the Five-Point Relative Pose Problem", in *IEEE Tr. on Pattern An. and Machine Int.*, vol. 26, no. 6, pp. 756-777, 2004.

[13] B. Shirmohammadi and C. Taylor, "Self Localizing Smart Camera Networks and their Applications to 3D Modeling", in *ACM Sensys / First Workshop on Distributed Smart Cameras*, October 2006.

[14] B. Girod, R. Rabenstein et al., *Signals and Systems*, Chichester, Wiley, 2001.

[15] R. Kleihorst, B. Schueler and A. Danilin, "Architecture and Applications of Wireless Smart Cameras (Networks)", in *Proceedings ICASSP*, 2007.

[16] R. Kleihorst et al., "Xetal: A Low-Power High-Performance Smart Camera Processor", in *IEEE Int. Sym. on Circ. and Syst.*, vol. 5, pp. 215-218, 2001.

[17] A.A. Abbo, R.P. Kleihorst et al., "Xetal-II: A 107 GOPS, 600 mW Massively Parallel Processor for Video Scene Analysis", in *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 192-201, January 2008.