VERSITA

OPTO-ELECTRONICS REVIEW **21**(1), 23–38

DOI: 10.2478/s11772-013-0076-8

Structured light camera calibration

P. GARBAT, W. SKARBEK*, and M. TOMASZEWSKI

Faculty of Electronics and Information Technology, Warsaw University of Technology, 1 Pl. Polietchniki, 00–661 Warsaw, Poland

Structured light camera which is being designed with the joined effort of Institute of Radioelectronics and Institute of Optoelectronics (both being large units of the Warsaw University of Technology within the Faculty of Electronics and Information Technology) combines various hardware and software contemporary technologies. In hardware it is integration of a high speed stripe projector and a stripe camera together with a standard high definition video camera. In software it is supported by sophisticated calibration techniques which enable development of advanced application such as real time 3D viewer of moving objects with the free viewpoint or 3D modeller for still objects.

Keywords: 3D viewer, 3D modeller, structured light method, camera calibration.

1. Introduction

Structured light technique, is one of the most widely used techniques in practical applications of three-dimensional (3D) shape measurements, including object recognition, digital model generation, medical and biometric applications. Recently the numerous based 3D shape measurement approaches have been developed [1–6]. Nowadays, in the 3D shape measurement systems it is needed to use the following combined technical features: fast speed, high accuracy, capacity of measuring complex shapes. However, the existing techniques can satisfy some of the requirements, but not all of them. Typical shape measurement projection systems are based on visible light spectrum. That approach limited applications in various fields.

The most popular device which uses structured light method in an infrafred band is Kinect [7]. The Kinect sensor consists of an infrared laser emitter, an infrared camera and an RGB camera. The pattern projection is realized by a laser source whose beam is splited into multiple beams by a diffraction grating. This speckle pattern is captured by the infrared camera and is correlated with a reference pattern. In this method point density and accuracy are the most important factors influencing the quality of a point cloud [8]. The main intrinsic source of the error in the Kinect data are: inadequate calibration, inaccurate measurement of disparities, and the imaging geometry.

The proposed simple solution promises to satisfy the critical demands in this fields.

One of the novelties proposed in this project is the monitoring of real 3D time variable objects with application of an active shape measuring method. The proposed system is based on the temporal stripe sequence projection technique. This setup provides data of 3D objects' geometry in form of point clouds and additionally – thanks to colour camera – information about colour of objects surface for every point of the cloud in a form of three components RGB. The proposed method for determination of objects' shape is based on an analysis of a stripe shape change in a calibrated measuring setup. For a calibration of setups' measuring volume, novel cameras and projector calibration methods are developed.

As the main achievement we consider getting real time visualisation for off-shelf PC equipment. The main reason of the success is a design of several look-up tables which hide nonlinearities of the optical system and solve the 3D correspondence problem.

2. 3D shape acquisition

2.1. Overview of the proposed system

One of the important requirements associated with a 3D active acquisition system is that capture process does not disturb human eye in any way while working in contact with human being. We proposed a model of a setup for a 3D shape measurement system in which the 3D estimation could be performed in the near IR domain. In this paper, we describe an improved version of the system based on binary stripe patterns for a real-time 3-D shape measurement. This system takes full advantage of the single-chip DLP technology for rapid switching of coded fringe patterns. The set of different patterns is created by a PC. When this pattern is sent to a single-chip DLP projector, the projector emits the binary pattern in sequence repeatedly and rapidly. The detection module consists of a synchronized high speed CCD stripe camera and a colour CMOS camera, which are calibrated together. The

*e-mail: W.Skarbek@ire.pw.edu.pl

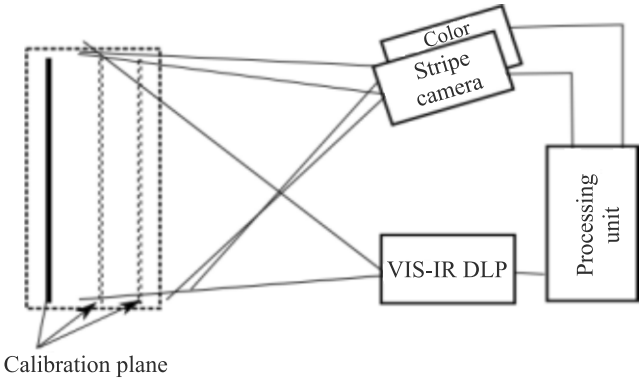


Fig. 1. General scheme of proposed 3D measurement setup.

stripe camera is used to capture the pattern images from which 3D information of the object surface is retrieved. Long-pass filter (which blocks infrared) is substituted with band-pass filter (blocking visible light and other IR radiance) to use infrared in our research. The colour HD camera is also used to take 2D colour pictures of the object at a frame rate of 25 frames/s for texture mapping.

Fig. 1 shows the layout of the proposed structured light system for a 3-D shape measurement, and Fig. 2 shows a picture of the developed hardware system. For the projection of computer-generated patterns, a single-chip DLP projector is used, which produces images based on a digital light switching technique.

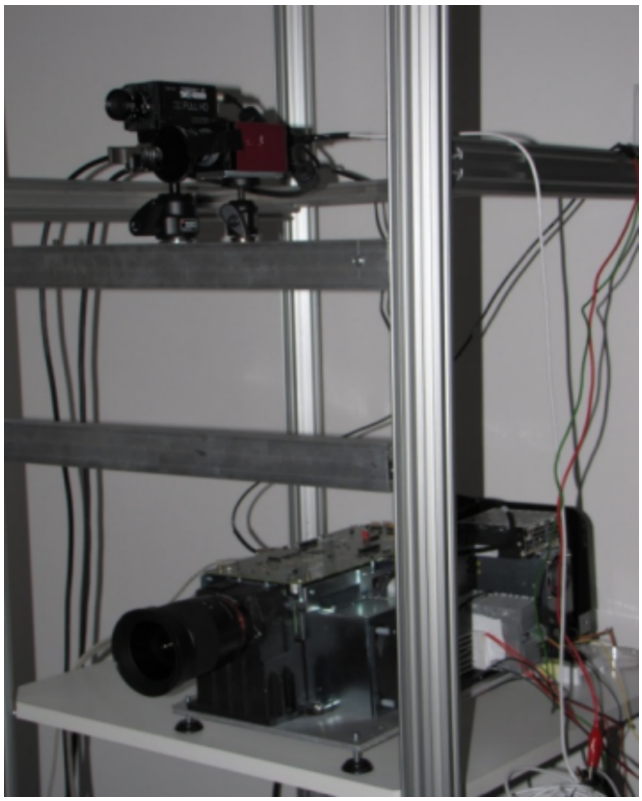


Fig. 2. General scheme of proposed 3D measurement setup.

2.2. VIS and infrared projector

The human eyes cannot detect waves in the range over 740 nm. Thus, infrared is used to avoid discomfort to the human eyes. Infrared, by itself, is classified into three ranges: near--infrared, medium infrared, far-infrared. Since CCD sensors used in digital cameras can detect near-infrared, we employ near-infrared band. The main aim of using IR light source was to improve comfort of work and to decrease influence of a colour (texture) object for data acquisition quality. Quality of data is related with accuracy of the stripe image segmentation process. We can observe a decrease in difference of intensity for individual colours in IR illumination relatively to VIS illumination, Fig. 3.

The light source element includes a very high power IR LED (850 nm) and condenser lens. To design a VIS and IR projector for a 3D reconstruction, two requirements exist as follows:

1) the projector casts a procedural stripe patterns that consists of vertical and horizontal stripes;
2) two VIS and IR wavelengths are required to calibrate procedure. Proposed solution is composed of a 2-band LED source: IR-850 nm, and Green-532 nm.

The projector was built based on a DLP platform of Texas Instrument Company. Main elements of the system are: DMD device driving setup with driving and output interfaces, LED illuminator, illuminator driving setup, power setup and optical module with lens. The optimal projection settings will depend, in part, on the frame rate capability of the camera, the type of lens and light source used in a projector, the camera and the background illumination conditions. The applied platform allows for the use of lenses with F-type mounts which makes simpler selection of lenses with a proper measuring volume. Three criteria must be taken into account while lens selection: value of measuring volume – angle, depth of a field range is strictly related to a shutter value, intensity of projected stripe images and illumination uniformity. The selection of lens is brought down to selecting the one that under given angle and depth of field allows to acquire an image of sufficient brightness. Twenty levels of difference between black and white stripe were

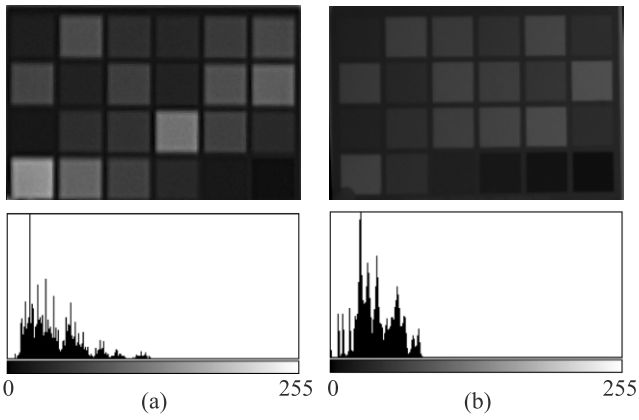
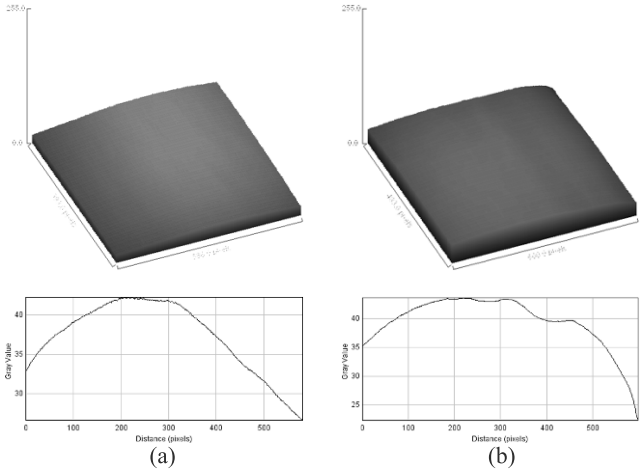


Fig. 3. Luminance modulation on colour test for two types of light source: a) VIS light source, and b) NIR (850 nm) light source.

chosen as a detection threshold. In this implementation a 35-mm lens with a 2.4-F number was chosen.

In order to increase the intensity of projected images illumination setup was also modified. Considering the applied method of binary stripes' projection analyzed sequentially, the sequential switching of diodes can be canceled. The illuminator proposed is made of two channels green for calibration of a HD camera and NIR for a stripe camera. Both work simultaneously, and separation is obtained on camera tracks thanks to bandpass filters. The internal timing signal of the light source is disabled. Two dichroic filters were applied in order to bring light beams of both channels to a DMD converter. Considering varying demands of LED power two separate power suppliers were used. Considering high power of LED IR diode a system of Peltier cells and mechanically cooled radiators were used for heat drainage. The LED IR diode used in illuminator emits light of 850 nm wavelength and allows for 18 W of optical power. Uniformity of illumination is the next very important aspect in the stripe detection process. The main sources of this inconvenience are projector lens and LED light source. Designed lighting module corrects non-uniformity of illumination. Fig. 4 shows effect of the source correction.

Driving setup of DMD system with created driving software allows for an introduction of stripe images generated procedurally to apparatus memory and displaying them in a correct order. Apart from the order of images in a sequence, the possibility of changing times of image displaying exists. The developed procedure library allows for driving the displaying process in a "step by step" mode. A sequence of encoded patterns, as discussed in Sect. 3, was selected for this implementation. The binary patterns are generated and uploaded into projector's unit memory. Once all patterns have been created, each one was saved as a bitmap image file with dimensions of 1024×768. The procedure of projector handling consists of four basic steps: generating a new image pattern sequence, initialization of device with setting of display frequency, loading of binary pattern images to RAM of the projector, determination of images order, starting of display sequence. Synchronizing signal is generated by a driving setup of the projector for each image from the sequence. This allows for a synchronization of the stripe camera with displayed images. To trigger the camera enables at least an output sync. A 50-μs signal, with a 10-μs delay and positive polarity to trigger the camera is used in the setup. The frame rate on the DMD is set to 200 fps.



(a) (b)

Fig. 4. Non-uniformity of projection light (illuminated surface plot and H profile): a) before, and b) after correction.

The stripe camera works in B/W mode which allows to register 200 frames per second at 640×480 resolution. The separation of visible light from IR light is done with application of a bandpass filter of characteristics shown in Fig. 5. The camera works in a trigged mode. Each projected pattern activates image acquisition operation. In the system, we use the optimized image capturing software to process a sequence of 8 images. For a single sequence the captured images are processed online (without saving into a hard disc). For each sequence index image is obtained.

A camera with HD texture allows for registration of images in 1080i50 format. Information about lumination and chrome is given by a camera in YUV format. Considering this two additional operations are required: deinterlacing and YUV – RGB conversion. Synchronization of HD camera is done according to program. The main application of data conversion forces acquisition of the following frame to a specified memory buffer.

The proposed system, based on the fast 3-D reconstruction algorithm and parallel processing technique in GPU, allows to realize a high-resolution, real-time 3D shape measurement at the frame rate of up to 25 frames/ s and a resolution of 640×480 points per frame.

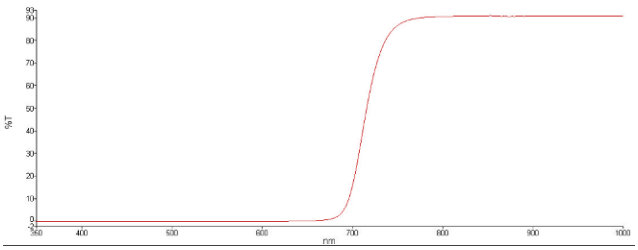


Fig. 5. Band pass filter characteristics.

3. Introduction to structured light camera calibration

Structured light is a method of light volumes' forming which is useful in surface modelling when projected on them [9–18].

Stripe images are a mean of 3D space subdivision into sectors to which a unique code can be assigned. The code can be detected (read out) in a stripe camera image which acts with the same speed as the projector while acquiring an image of the illuminated scene in the visible spectrum range.

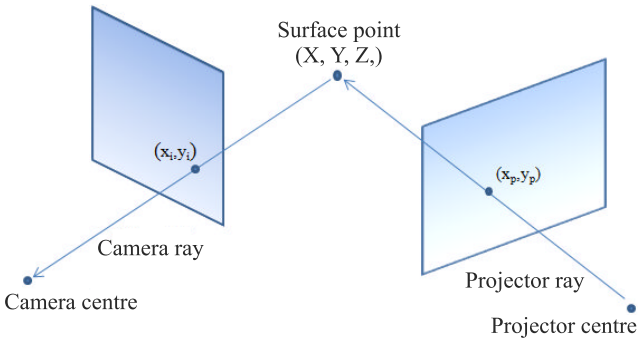


Fig. 6. 3D point registration using a single light ray.

3.1. Beginning of light structuring: single ray

The simplistic case is a *3D point registration* problem which can be described as follows:

- *Single ray design*: set a single pixel (x_p, y_p) in the graphics card frame buffer with an adequate RGB values.
- *Project the ray on a surface*: ensure that the reflected ray from an unknown point (X, Y, Z) differs in colour wrt to its spatial neighbours (Fig. 6).
- *Get an image of the surface*: make a camera single shot.
- *Detect image* (x_i, y_i) *of* (X, Y, Z): apply a colour segmentation procedure (Fig. 7).
- *Compute an approximation* $(\tilde{X}, \tilde{Y}, \tilde{Z})$ *of the point* (X, Y, Z): apply the Least Square Method to find a point of equal minimal distance from projector's ray (x_p, y_p) and camera's pixel ray (x_i, y_i).

Fig. 7. Image of a single ray projected onto a 3D model.

3.2. Plane of light rays

In practice, when a 3D scene is static, the moving light plane, i.e. the plane of light rays which enlight a curve Γ on a 3D object's surface (Fig. 8) is applied [19–27]. Then, we have four algorithmic steps for each light plane $y = y_p$:

1. *Emission of light plane* $y = y_p$.
2. *Image registration* for modelled object.

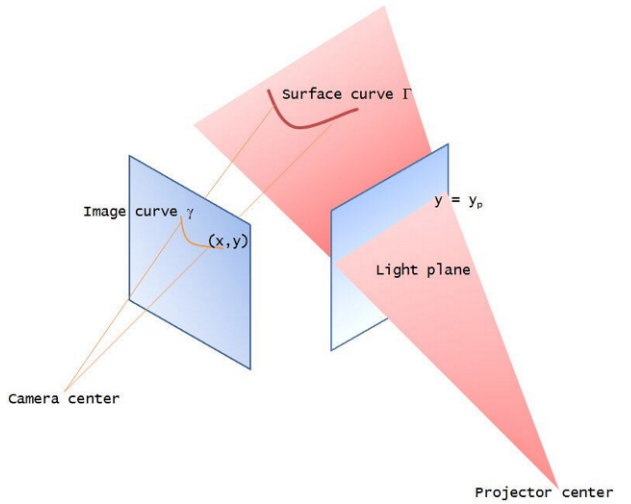


Fig. 8. Light plane projected onto a surface enlighting a curve in a 3D space.

3. *Detection of image curve* γ = the image of Γ
4. *Approximation of spatial curve* Γ

From a systemic point of view we can consider the complete 3D modelling algorithm as follows:

- Systemic description:
  1. *Input:* the plane of light rays $y = y_p$.
  2. *State:* the unknown curve Γ on object's surface.
  3. Output: the discrete curve γ.
- 3D modelling as the system identification problem:
  1. *Given:* image(s) of 3D object illuminated by structured light.
  2. *Identification of system output:* the detection of discrete curve γ in camera's image.
  3. *Identification of system in*: the detection of light plane $y = y_p$, generating γ.
  4. *Identification of system state:* the approximation of the curve Γ on the object's surface.

If we can segment out the curve γ in the image, then, the modelling problem is reduced to many 3D point registration problems using many rays.

3.3. Advancing light structuring – stripes

In case of a 3D modelling of dynamic scenes the single light plane must be replaced by many planes what makes in practice their segmentation in the image a hard task. Using different colours for light planes can help in a limited area of applications [28–33].

The light planes are generated from stripe pattern images attributed as follows:

- *Stripe*: colour (in RGB), orientation (H or V), location (first line index and width).
- *Stripe pattern* is the list of disjoint stripes of the same orientation, covering the projector's image frame.
- What is important, the stripe sector is a pyramid with its apex located within the volume of the stripe projector and with an unspecified base (Fig. 9).

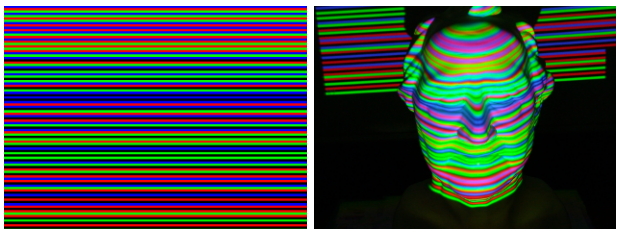


Fig. 9. Stripe pattern image and its projection onto a 3D model.

In the stripe camera, for each pixel the sector's code is readout for the stripe sector with the view pyramid determined by the pixel. Such readout is possible if in the intersection of both pyramids there is a surface reflecting projected light.

3.4. Need for stripe colour coding

Since the number of different stripe colours should be reduced in order to detect automatically which stripe pixel belongs to (out of many of the same colour) one has to arrange consecutive (in space) stripe colours in such an order that the stripe identity, say an index (Fig. 10), can be uniquely determined [34–37].

Actually such an arrangement leads to a colour encoding scheme. To define such a code we select which is the stripe window size and, then, for each image pixel its colour code is a sequence of colours which occur in its neighbourhood. For instance the colour of the stripe the pixel belongs together with $n-1$ colours for the next $n-1$ stripes (in axis order with warping for the trailing stripes).

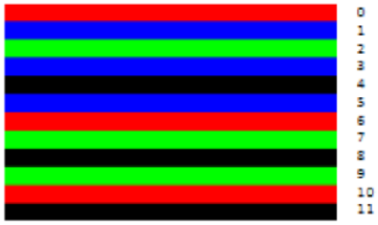


Fig. 10. Example of colour stripe pattern which represents a modified De Bruijn code.

3.5. Gray patterns

Interfering of stripe colours with objects colours what occurs in the scene image makes the stripe colour identification error prone. Hence, the idea of temporal pattern sequence arises in order to code colours in time, independently for each pixel in the stripe image [24,38–40]. The most popular approach is the use of Gray code (Fig. 11) which exhibits the following property: *each pixel has either the same colour for each stripe pattern or belongs to an edge between colour stripes in only one stripe image.*

The definition of the Gray code could be given by an inductive scheme:

- Gray code of length n over alphabet $\sum_s \doteq \{0,\ldots,s-1\}$ is a one-to-one mapping $G_{s,n}\colon [0,s^n) \to \sum_s^n$ for which

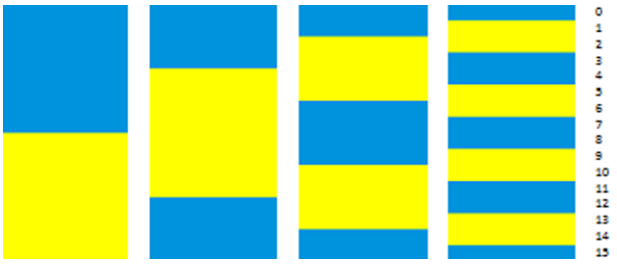


Fig. 11. Stripe images representing Gray code for colours and patterns.

$G_{s,n}(i)$ and $G_{s,n}(i+1 \bmod s^n)$ differ in exactly one position.

- Domain of $G_{s,n}$ is the set of indexes for items to be encoded.
- Item to be encoded – for instance a stripe of light.
- $\Rightarrow$ Gray code of length one: The code $G_{s,1}(i) \doteq i, i \in \sum_s$ is Gray code of length one.

$\Rightarrow$ Inverse Gray code: Let $G'_{s,n}$ be an inverse Gray code defined wrt to Gray code $G_{s,n}$ as follows:

$$G'_{s,n}(i) = G_{s.n}(s^n - 1 - i), \quad i \in [0, s^n).$$

Then $G'_{s,n}$ is the Gray code over the alphabet $\sum_s$

Binary Gray codes in 2D tables:

- item index is array column index;
- code position is array row index.

- Example for $s = 2,\ n = 1$:

$G_{2,1}$	[0]	[1]
[0]	0	1

- Example for $s = 2,\ n = 3$:

$G_{2,3}$	[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
[0]	0	0	0	0	1	1	1	1
[1]	0	0	1	1	1	1	0	0
[2]	0	1	1	0	0	1	1	0

3.6. Departure from Gray codes for calibration and modelling

In the section 5 we derive another code to define stripes with better properties than Gray code patterns.

For our camera more important is pixel dependent thresholding for the accurate camera image binarisation. The repetition factor will characterize the admissible code.

Another property is a balance between white and black areas in the stripe pattern image. It is achieved by the symmetrical offset in the table of admissible codes. This leads us to valid codes used for calibration and modelling.

There are three correspondences as the results of the structured light camera calibration:

1. (stripe camera pixel, valid code index) $\mapsto$ depth index;
2. (stripe camera pixel, depth index) $\mapsto$ video camera pixel;
3. (stripe camera pixel, depth index) $\mapsto$ Z coordinate of 3D point.

Since optical distortions of cameras contribute with high share, and projector with less impact, the above correspondences are highly nonlinear. Therefore, the number of modelling parameters is intractable. Hence, they are approximated by interpolation process using experimentally established correspondence tables.

4. Homographic method for depth calibration

4.1. Homographic context of the calibration process

The conceptual side view of the projector screen and video devices (projector, video, and stripe cameras) is given in Fig. 12.

The calibration process consists of D depth data registration steps. At each step, we shift the movable projector screen to the predefined depth positions indexed by $d_0, \ldots, d_{D-1}$. The top view of the experimental setup is illustrated in Fig 13.

The coordinate frame $[q_x, q_y, q_z]$ linked in the above picture to the depth d_3 presents the orientation of the projector screen wrt to the video image plane. The symbol t stands for the location of the plane reference point wrt to camera origin point.

In our approach depth planes are used together with light planes for 3D space fine subdivision. Ideally, they should be evenly spaced, and parallel to each other and to the video image plane too. However, in practice we are not able to ensure these requirements. To correct the arising errors in depth measurements, we find for each screen location its pose, i.e. the orientation and the location of its reference points wrt the video camera coordinate system. The use of plane pose is deferred to 3D modelling step and/or on-line 3D viewing.

The pose of the calibration plane is found by using the homographic method which handles the perspective projections between physical planes and image projection planes.

In this section we explore the details of the homographic method for the pose identification, and for the vision intrinsic parameters identification, as well.

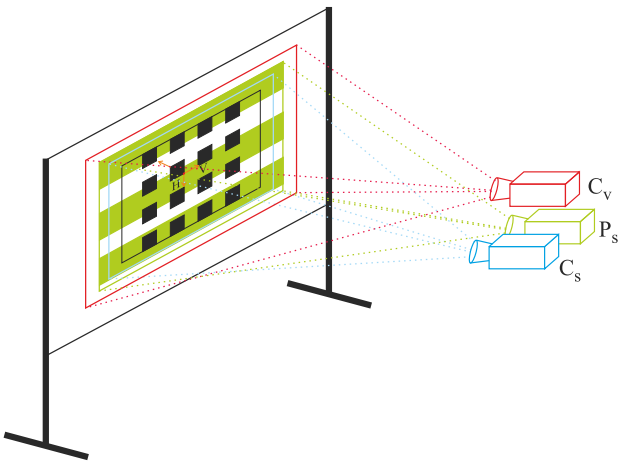


Fig. 12. Calibration setup: fixed positions of the stripe camera C_S, the video camera C_V, and the stripe projector P_S – the movable calibration screen.

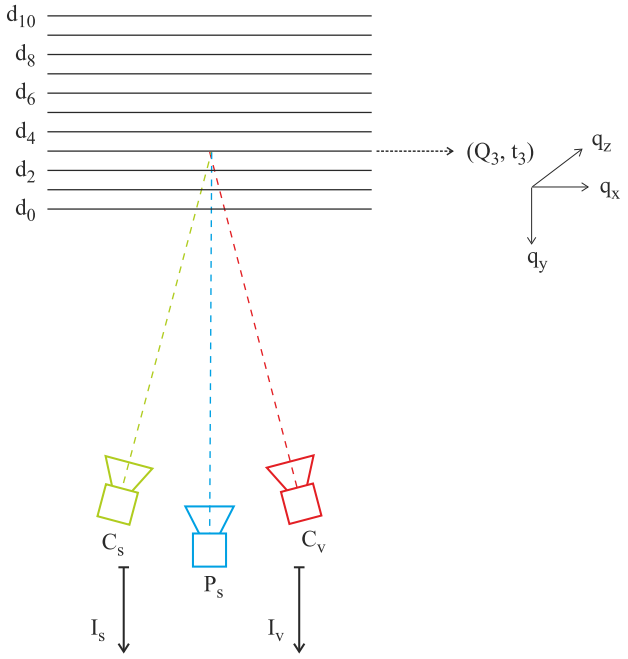


Fig. 13. Top view of calibration screen locations wrt to the video devices, where I_S is the stripe image, I_V is the video image.

While developing an independent implementation of the calibration for the new vision system, the structured light camera is, we explored apparently novel options of the relevant algorithms including: corner detection, corner indexing, linear, and nonlinear optimization of homographic constraints.

Part of the material discussed here is well known, but its new way of presentation could be of tutorial value for readers.

4.2. Homography basics

Homography is a projective mapping between a plane located in a 3D space and the projective plane. Introducing uniform coordinates on both planes, the homography can be represented by a matrix $H \in R^{3\times 3}$ [41].

There is a simple formula for the matrix H, if the scene plane is equipped with the orthogonal coordinate system (Q,t), the defined axis versors $Q = [q_x, q_y, q_z]$ and the translation vector t, all expressed wrt coordinate system of the projective plane:

$$H' = [q_x, q_y, t].$$

If projective plane is covered by a pixel grid, then, the homography matrix H transforming to pixel coordinates has the obvious form:

$$H = KH' = K[q_x, q_y, t] \doteq [h_x, h_y, h_z]$$

$$h_x \doteq \begin{bmatrix} h_{xx} \\ h_{yx} \\ h_{zx} \end{bmatrix}, \quad h_y \doteq \begin{bmatrix} h_{xy} \\ h_{yy} \\ h_{zy} \end{bmatrix}, \quad h_z \doteq \begin{bmatrix} h_{xz} \\ h_{yz} \\ h_{zz} \end{bmatrix},$$

where $K \in R^{3\times 3}$ provides conversion from projective plane coordinates to pixel coordinates.

In camera terms the matrix K is called intrinsic parameters' matrix and its identification is a part of camera calibration process.

Extracting at least five pixels (x, y) in the image I which are images of the points (X, Y) from the scene plane is enough to find the homography for this plane, as the one projective correspondence provides two homogenous equations wrt unknown h_{ab}, $a, b = x, y, z$.

$$Xh_{xx} + Yh_{xy} + h_{xz} - xYh_{zx} - xYh_{zy} - xh_{zz} = 0$$
$$Xh_{yx} + Yh_{yy} + h_{yz} - yYh_{zx} - yYh_{zy} - yh_{zz} = 0.$$

Having $n \geq 5$ point-pixel correspondences $(X_i, Y_i) \mapsto (x_i, y_i)$, $i = 1, \ldots, n$, the uniform set of equations $A_u h = 0$ is obtained with the matrix $A_u \in R^{2n \times 9}$

$$A_u^t \doteq \begin{bmatrix} X_1 & 0 & \ldots & X_n & 0 \\ Y_1 & 0 & \ldots & Y_n & 0 \\ 1 & 0 & \ldots & 1 & 0 \\ 0 & X_1 & \ldots & 0 & X_n \\ 0 & Y_1 & \ldots & 0 & Y_n \\ 0 & 1 & \ldots & 0 & 1 \\ -x_1X_1 & -y_1X_1 & \ldots & -x_nX_n & -y_nX_n \\ -x_1Y_1 & -y_1Y_1 & \ldots & -x_nY_n & -y_nY_n \\ -x_1 & -y_1 & \ldots & -x_n & -y_n \end{bmatrix}$$

There is also more known nonuniform approach, i.e. the equation $Ah = b$, which assumes normalization of $h_{zz} = 1$. Then, the equation matrix $A_n \in R^{2n \times 8}$ consists of the first eight columns of the matrix A_u and the right hand column vector $b \in R^{2n}$:

$$b = [x_1, y_1, \ldots, x_n, y_n]^t.$$

The uniform set of equations is solved by kernel analysis of the matrix A_u which, in turn, is performed using singular value decomposition (SVD) method:

$$A_u \overset{SVD}{=} U \Sigma V^t, \quad V = [v_1, \ldots, v_9]$$

The kernel of A_u is spanned by the right singular vectors v_i corresponding to zero singular values. The point-pixel correspondence is consistent if the last singular value is close to zero, and the second to the last is large comparing to the last one. Then, the l_2 normalized solution is either v_9 or $-v_9$.

In case of nonuniform method the pseudo-inversion is used for the matrix A_n. The pseudo inverse A_n^+, applied to b, defines the solution v:

$$Av = b \rightarrow v = A^+ b$$
$$A = U \Sigma V^t \rightarrow A^+ = V \Sigma U^t,$$

where

$$\Sigma_{ii}^+ = \begin{cases} \Sigma_{ii}^+ & \text{if } \Sigma_{ii} \neq 0.0 \\ 0.0 & \text{otherwise} \end{cases}.$$

In case of error pixel measurements, the good approach is selection of the minimal subset of points with the minimal error of data reconstruction. For the inverse method the minimal subset of points consists of four points creating a non-degenerated quad. We check the correctness of the four-point subset by the function `isQuad` which verifies whether all of $\binom{4}{3} = 4$ three point subsets are valid triangles with integer coordinates:

$$\left|(X_i - X_j)(Y_k - Y_j) - (X_k - X_j)(Y_i - Y_j)\right| < 1$$

where the set $\{i, j, k\}$ is any subset of $\{0,1,2,3,4\}$

In case of five points (suitable for kernel method) we check existing of $\binom{5}{3} = 10$ triangles and, then, the above condition is verified for $\{i, j, k\}$ being subset of $\{0,1,2,3,4\}$.

If we get from an image analysis the corresponding points $(X_i, Y_i) \mapsto (x_i, y_i)$, $i = 1, \ldots, n$, to find the homography, it may happen that some of them are located in the image areas where optical distortions make the homographic model invalid.

Since a priori the valid pixel area is unknown we guess a circle $(x_0, y_0; r_0)$ to bound the camera points. The subpixel accuracy of the obtained model should confirm our guess, provided that camera pixels within the correspondence are found with subpixel accuracy. Rejecting pixels actually implies rejecting relevant corrspondences.

Another aspect before we find the homographic model is scaling point and pixel coordinates to the intervals $[-0.5, +0.5]$. It will reduce the numerical error of matrix pseudo inverse and singular value decomposition.

The following steps handle data scalings:

1. Parameters of bounding box for accepted pixels:
$$m_x = \min_i[x_i], M_x = \max_i[x_i]$$
$$m_y = \min_i[y_i], M_y = \max_i[y_i]$$
$$d_x = M_x - m_x; \; d_y = M_y - m_y$$
$$c_x = (M_x + m_x)/2; \; c_y = (M_y + m_y)/2.$$
2. Pixel scaling for $i = 1, \ldots, n$:
$$x_i \leftarrow \frac{x_i - c_x}{d_x}, \quad y_i \leftarrow \frac{y_i - c_y}{d_y}.$$
3. Pixel de-scaling matrix:
$$S_{xy}^{-1} = \begin{bmatrix} d_x & 0 & c_x \\ 0 & d_y & c_y \\ 0 & 0 & 1 \end{bmatrix}.$$
4. Parameters of bounding box for accepted points:
$$m_X = \min_i[X_i], M_X = \max_i[X_i]$$
$$m_Y = \min_i[Y_i], M_Y = \max_i[Y_i]$$
$$d_X = M_X - m_X; \; d_Y = M_Y - m_Y$$
$$c_X = (M_X + m_X)/2; \; c_Y = (M_Y + m_Y)/2.$$

5. Point scaling for $i = 1, \ldots, n$:

$$X_i \leftarrow \frac{X_i - c_X}{d_X}, \quad Y_i \leftarrow \frac{Y_i - c_Y}{d_Y}.$$

6. Point scaling matrix:

$$S_{xy} = \begin{bmatrix} \frac{1}{d_X} & 0 & -\frac{c_X}{d_X} \\ 0 & \frac{1}{d_Y} & -\frac{c_Y}{d_Y} \\ 0 & 0 & 1 \end{bmatrix}.$$

Preprocessing of input data before the homography implies the need for postprocessing of the output matrix. Let S_{xy} transform pixel domain, while S_{XY} transforms the point domain:

$$[x', y', 1]^t = S_{xy}[x, y, 1]^t$$
$$[X', Y', 1]^t = S_{XY}[X, Y, 1]^t.$$

Then, the identified homography H' in the transformed domains satisfies the equation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \lambda H' \begin{bmatrix} X' \\ Y' \\ 1 \end{bmatrix}.$$

Hence,

$$S_{xy} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda H' S_{XY} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}.$$

Therefore,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda S_{xy}^{-1} H' S_{XY} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}.$$

In conclusion the post-processing step has the form of:

$$H \leftarrow S_{xy}^{-1} H' S_{XY}.$$

We identify the best homography for the given correspondences using a greedy method wrt the randomly permuted sequence.

4.3. Homography from a calibration image

In order to get the pose of the projector screen at the given depth wrt to the video camera, we are obliged to use the classical pattern in the form of a regular grid of black squares printed on the white paper. The detected (in the video camera image) and indexed corners of squares determine the calibration points used for computing of the homography, and next, to find the pose of the screen.

The idea of imaging the scene calibration points is shown in Fig. 14:

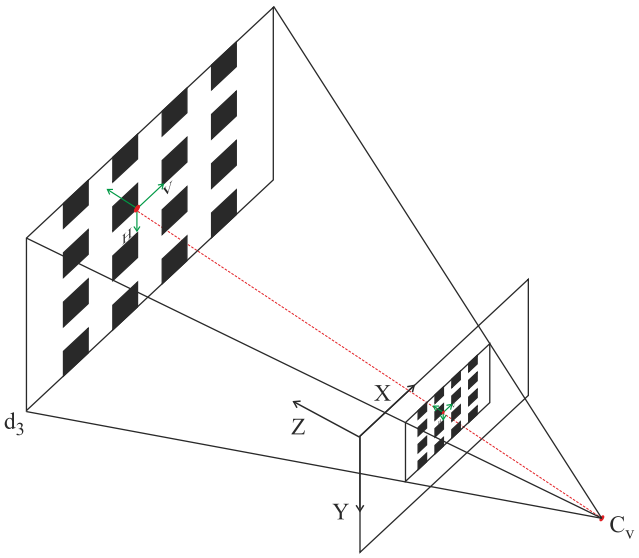


Fig. 14. Idea of imaging scene calibration point.

There are two problems we have to solve before the homography can be computed on the basis of correspondence of square corners on the screen plane and the square corners in the camera image:

1. The corners detection problem: the pixel location in the camera image should be detected for each visible square corner of the calibration pattern sticked to movable projector screen.
2. The corners indexing problem: to each pixel representing detected corner the corner from the calibration scene should be assigned to.

The first problem we solve using Harris corner detector as a compromise between the accuracy, completeness, and computational complexity factors [42].

The second problem is actually the special case of the correspondence problem between scene points and their images [43]. The name of indexing is used since the scene corners are naturally indexed in the following way: if the calibration square occupies the row r and the column c of the regular calibration pattern, then, (r, c) are indexes of the square which for corners are extended by bits 00 for LU (left upper) corner, 10 for RU (right upper) corner, 01 for LD (left down) corner, and 11 for RD (right down) corner.

4.4. From homography to plane coordinate frame

Let the camera intrinsic parameters K and the homographic matrix H from the scene plane to the camera image plane be given. We are looking for the unknown coordinates frame (Q, t).

We derive the relevant formulas:

1. The starting equation:

$$[h_x, h_y, h_z] = K[q_x, q_y, t].$$

2. The X axis vector q_x:

$$q_x = K^{-1} h_x.$$

3. The Y axis vector q_y:

$$q_y = K^{-1} h_y .$$

4. The Z axis vector q_z:

$$q_z = q_x \times q_y$$

5. The coordinates change matrix Q – from scene plane to camera projective plane:

$$Q = [q_x, q_y, q_z]$$

6. The translation vector t:

$$t = K^{-1} h_z$$

7. The coordinate frame of scene plane: (Q, t).

In the seminal Zhang's paper [41], the author used only scaling by $\lambda \doteq \|q_x\|^{-1}$ assuming equality of norms $\|q_x\| = \|q_y\|$. Since usually it is not valid assumption for comparison with our general correction procedure, we redefine λ to the inverse of average length

$$\lambda = \frac{2}{\|q_x\| + \|q_y\|}.$$

Since the unit vectors could be nonorthogonal, the matrix Q could be nonorthogonal, too. We apply the SVD decomposition $Q = U \sum V^T$ to find the nearest to Q (in Frobenius norm) orthogonal matrix $Q' = UV^t$:

$$Q = U \sum V^T \rightarrow Q' = \arg\min_R \|Q - R\| = UV^t ,$$

where R is any rotation matrix ($R^t R = I$, $\det(R) = 1$).

5. Stripe patterns

Any sequence of k binary images defines a code image including at most 2^k different binary codes. Since we want to use such coding to subdivide the 3D space into identical parts, the corresponding areas in code image, so called level sets, should be identical, too. The simplest areas of this kind are horizontal or vertical stripes of the equal height L_h or the width L_w. For structured light camera calibration we need more fine 3D space subdivision and, then, square areas obtained as intersections of horizontal and vertical stripes will fit.

High speed stripe projectors require saving all stripe images in its memory. Then, its capacity determines the upper bound $k_{\max}$ for the total number of horizontal k_h and vertical k_v stripe images:

$$k_h + k_v \le k_{\max}.$$

This bound is used in calibration application and for off-line 3D modelling of still objects. In practice $k_h, k_v < 10$ and $k_{\max} > 1000$. Therefore, the above condition is satisfied in the contemporary technology state of art.

In 3D player we use only one-directional stripes, and, then, the speed $f_{ps}^{(s)}$ of the stripe projector and the speed $f_{ps}^{(v)}$ of the video (texture) camera, measured in actually registered image frames per second, matter:

$$k \le \frac{f_{ps}^{(s)}}{f_{ps}^{(v)}},$$

where $k = k_h$ or $k = k_v$.

Obviously, for the projector with resolution $W \times H$ the further constraints for k_h and k_v:

$$k_h \ge \log_2 H, \quad k_v \ge \log_2 W.$$

Here, we have assumed use of the finest stripes of one pixel width. However, in practice, the registration of borderline pixels is error prone. Since both sides of border can be affected, the borderline could be wider than one pixel. Therefore, perfect codes are expected at ratio at least $100(L-2)/L\%$. For instance, for the stripe width of L = 4, 10, 20 we get 50%, 80%, 90% respectively.

For system calibration we need higher ratio of code recognition and single line resolution of code image. By trading effective code image resolution with calibration time we display the same stripe image L times. At i-th time each image is displayed at offset i rows (in case of horizontal stripes) and at offset i columns (in case of vertical stripes).

Since projector's memory is enough large we can keep all original images and their offset images together for stripe width L:

$$Lk_h + Lk_v \le k_{\max} \rightarrow L \le \frac{k_{\max}}{k_h + k_v}.$$

For $k_v \le k_h = 10$, $k_{\max} = 2000$ we have $L \le 100$. With $f_{ps}^{(s)} = 200$ we get not more than 10 seconds to collect all calibration data at the fixed relative position of the video camera and the projector screen. Therefore, even for such extreme case mechanical movements of the devices determine the whole time of calibration procedure.

In particular case if $W = 640 = 5 \cdot 2^7$, $H = 480 = 3 \cdot 5 \cdot 2^5$, then L = 5, 10, 20 are feasible and the corresponding number of horizontal stripes $K_h = H/L$ = 96, 48, 24 and the number of vertical stripes $K_v = W/L$ = 128, 64, 32.

Since each stripe has a unique code, the number of codes $C_h \ge K_h$ and $C_v \ge K_v$ and they are determined by the construction of code table which, in turn, will depend on the number of bits $k_h (k_v)$, and the zero-one repetition factor r. For instance r = 2 means that in each bit-string of code table we have at least two zeros and at least two ones.

Code table construction is based on the following reasoning:

1. The light emitted from the projector is always an additional light in the 3D scene.
2. Therefore, we cannot distinguish all zeros from all ones.
3. Repetition condition on the level r: the quantity of zeros $\ge r$ and the quantity of ones $\ge r$.
4. For k = 4 stripe images the code table (codebook) consists of the entries: 0011, 0101, 0110, 1001, 1010, 1100.

5. In general for k, r we get the number C_{kr} of codes:

   a) if $r = 1$, then $C_{k1} = 2^k - 2$,

   b) if $r = 2$, then $C_{k2} = 2^k - 2k - 2$,

   c) if $r = 3$, then $C_{k3} = 2^k - k(k+1) - 2$:

$$C_{k3} = 2^k - 2 - 2k - \binom{k}{2} - \binom{k}{k-2} = 2^k - 2 - k - k^2 = 2^k - k(k+1) - 2.$$

6. For $r = 2$, one of the reasonable binarisation procedure is defined by thresholding wrt $T(i, j)$

$$T(i, j) = \frac{g_{l_1}(i, j) + g_{l_{k-2}}(i, j)}{2},$$
$$g_{l_0}(i, j) \le g_{l_1}(i, j) \le \ldots \le g_{l_{k-1}}(i, j),$$

where $g_l(i, j)$ jest l-th gray scale value in sorted k values at pixel (i, j) while the indexes are defined by using argsort function

$$[l_0, l_1, \ldots, l_{k-2}, k_{k-1}] = \text{argsort}[g_0, g_1, \ldots, g_{k-2}, g_{k-1}].$$

It is reasonable to make stripe generating independent of the projector image resolution. To this goal for each stripe single bit is assigned. Then, we get the colour pixels in stripe by scaling and copying the bit according to the required resolution.

Therefore, the constructor of object stripes defines the list of valid codes tables – one table per stripe image. The process of generating valid codes follows the definitions:

1. k – codeword length;
2. $K_{\max}$ – the number of all possible codewords of length k: $K_{\max} = 2^k$;
3. r – zero-one repetition factor ($2r < k$);
4. admissible codeword – it consists of at least r zeros and r ones;
5. C_{kr} – the number of admissible codes of length k, with repetition factor r;
6. valid codeword – the admissible codeword with index between the first (f) and the last (l) indexes in the sequence of admissible codes;
7. K – the number of valid codes:

$$K = l - f + 1, \quad K \le C_{kr};$$

8. The symmetric range of valid codes if:

$$f \doteq \left\lfloor \frac{C_{kr} - K}{2} \right\rfloor, \quad l \doteq f + K - 1.$$

In our approach K is a compromise between the projector image resolution and the stripe width. Having K we find the first and the last valid codes from the above condition on the symmetric range.

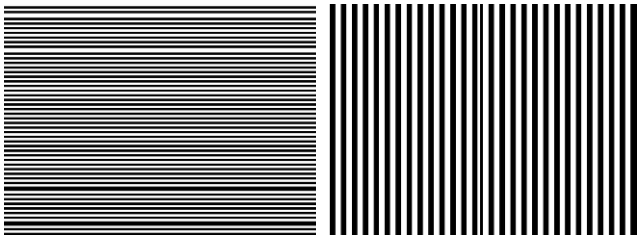


Fig. 15. Example of horizontal and vertical pattern with the repetition factor $r = 2$ and the stripe size $L = 10$.

In Fig. 15 we observe the nonuniform black and white stripes' layout and the balance between the white and black areas.

6. Stripe image analysis

The primary goal of the stripe image analysis is the definition of index image which is a smoothed version of usually used code image.

6.1. Index image definition

Context of image index:

1. The 3D scene is illuminated in turn by k stripe images (patterns) of the same orientation.
2. Optionally stripe images are interleaved with reference images switching between two images: all white F_{255}, F_{255}^c and all black pixels F_0.
3. Stripe camera images are registered in gray scale for each illumination.
4. Binarisation is performed according to equations described in the section 5.
5. However if registered by camera reference images F_0^c, are used, then binarisation of the pixel (x, y) in stripe images F^c is performed online by using the following equation:

$$\left|F^c(x, y) - F_0^c(x, y)\right| > \left|F^c(x, y) - F_{255}^c(x, y)\right| \rightarrow B(x, y) = 1.$$

The codeword $C(x, y)$ at pixel (x, y) is, then, computed incrementally:

$$C(x, y) \leftarrow C(x, y) + B(x, y) * w_i,$$

where w_i is the weight of i-th pattern which depends on the total number k of patterns for the given direction ($k = k_v$ or $k = k_h$):

$$w_0 = 2^{k-1}, \; w_{k-1} = 1, \; w_i = w_{i-1}/2$$

for $i = 1, \ldots, k-1$.

6. For each pixel its codeword is converted to its index, i.e. the position of the codeword in the table of valid codes.
7. For visualization, the index could be scaled to a gray scale value or simply be the index of a pseudo colours' table.

In Fig. 16 we observe the perfect index images which correspond to planar objects in a 3D space.

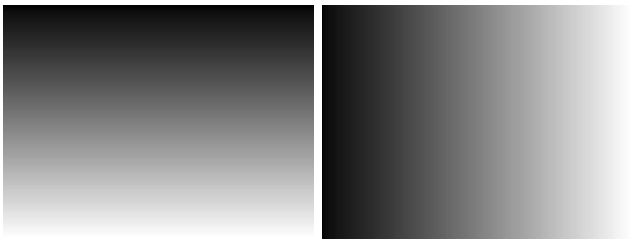


Fig. 16. Perfect index images for perfect planes.

On-line implementation of index tables (also for various sub-stripe offsets) essentially depends on type of interface the video camera offers. The call-back type leads to a more complex logic of the algorithm than the call-forward type. The cameras' drivers used in our experiments were of call--back type.

6.2. Virtual corners extractor

An overlay of vertical and horizontal index images registered by a stripe and a video camera produces a virtual quadrilateral mesh which can be used either for the system calibration wrt to depth information or during on-line 3D free--point views creation.

Therefore, a stripe detection in a camera image for illuminated scene is of crucial importance. For calibration intersections of stripe edges, so called corners, are also useful.

For corners and stripe edge detection the popular method is Harris filter which is a detector of abrupt changes in a natural image. However, the index image is a map of code image. The nature of stripe code is its stability within the stripe. The changes within the stripe mean error code detection inside the stripe. Moreover, changes between stripes in a calibration planar scene equal to 1, and to small integer for natural scenes.

To watch the results in Fig. 17 the detected corners are indicated by red colour within the vertical index image.

In Fig. 17 the perfect corner extraction is observed. This is possible only if a perfect image binarisation takes place what in real camera images is not true. Copying with imperfection is the main research goal. In this case, we simply avoid detection of stripe corners in favour of stripe inner pixels which exhibit much higher succes rate of detection.

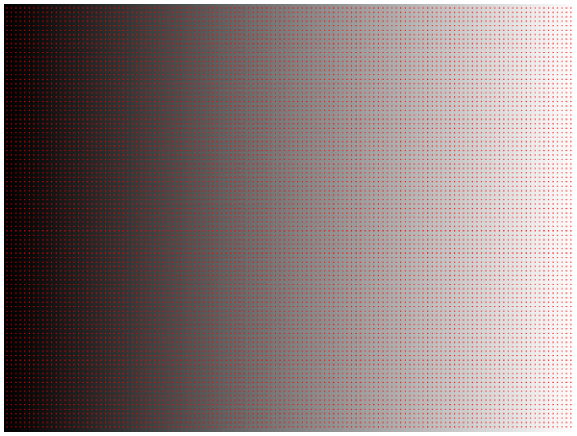


Fig. 17. Perfect corners for perfect index images.

6.3. Building sub-stripe index tables

Stripe patterns with fixed offset determine index tables which can be combined into one index table replacing stripe index with sub-stripe index.

In our experiments the sub-stripe is identical with the single line (either horizontal or vertical) of the stripe pattern. Since in the last stripe of the pattern, the sub-stripes are not defined, the final number of stripes equals:

$$K_v^s = L\cdot(K_v - 1), \quad K_h^s = L\cdot(K_h - 1).$$

We assign the index i_{ss} to the sub-stripe j of the stripe i_s according the equation:

$$i_{ss} = L\cdot i_s + j, \ \ j \in [0, L), \ \ i_s \in [0, K).$$

Since $KL = D$ is the resolution in the given direction (for instance $K = K_v$, $D = W$, for vertical stripes), the sub-stripe index is identical to line index in the given direction (for instance column line for vertical stripe pattern).

Hence, offset patterns increase the resolution of corresponding blocks' pixels in stripe and video cameras, what results in higher accuracy of the calibration process.

Note that sub-stripes are virtual objects. Stripes of width one are in practice undetectable, as most errors occur on stripe edges and in the lower than projector stripe camera resolution, which leads to perfect noisy image.

Since errors occur at pixels where the code changes, they are not taken into account at stripe membership evaluation.

In Table 1 we assume that error codewords can occur only at border pixels, i.e. those which are located at the beginning or at the end of stripes in camera image. The wrong codeword follows from wrong binarisation, while for wrong binarisation, there are two error sources:

1. Border pixels discretization – unavoidable random distribution of light energy for pixels located exactly on a narrow stripe edge line.
2. Wide border edges due to lack of camera focus.

While the first type of error does agree with the border error assumption, the second source may propagate errors further than one pixel apart of the border line. To avoid the errors of the second type we apply the decimation (sub-sampling by the factor two) for the video image. The code indexes which can be trusted are underlined in the upper part of Table 1.

6.4. Correcting wrong offset indexes

Suppose that in the above table the unsure index entries (those which are not underlined) are replaced by the maximum possible index for the given representation precision (we use unsigned 16-bit data words).

Consider now any pixel of j-th sub-stripe of the stripe with index i and its index value $iL + j$ for offsets $j = 0, \ldots,$

Table 1. Observed in time index values in the sub-stripes of a stripe (upper) and time intervals of reliable index detection.

		time ⟶							
	↑ $i-1$	(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)
i	[0]	i	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	i
	[1]	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	i	$i+1$
	[2]	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	i	$i+1$	$\underline{i+1}$
	[3]	$\underline{i}$	$\underline{i}$	$\underline{i}$	$\underline{i}$	i	$i+1$	$\underline{i+1}$	$\underline{i+1}$
	[4]	$\underline{i}$	$\underline{i}$	$\underline{i}$	i	$i+1$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$
	[5]	$\underline{i}$	$\underline{i}$	i	$i+1$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$
	[6]	$\underline{i}$	i	$i+1$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$
	[7]	i	$i+1$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$	$\underline{i+1}$
	↓ $i+1$								

j=3

0_i 1_i 2_i 3_i 4_i 5_i 6_i 7_i ⟼ $(i,0)$

0_i 1_i 2_i 3_i 4_i 5_i 6_i 7_{i+1} ⟼ $(i,1)$

0_i 1_i 2_i 3_i 4_i 5_i 6_{i+1} 7_{i+1} ⟼ $(i,2)$

0_i 1_i 2_i 3_i 4_i 5_{i+1} 6_{i+1} 7_{i+1} ⟼ $(i,3)$

0_i 1_i 2_i 3_i 4_{i+1} 5_{i+1} 6_{i+1} 7_{i+1} ⟼ $(i,4)$

0_i 1_i 2_i 3_{i+1} 4_{i+1} 5_{i+1} 6_{i+1} 7_{i+1} ⟼ $(i,5)$

0_i 1_i 2_{i+1} 3_{i+1} 4_{i+1} 5_{i+1} 6_{i+1} 7_{i+1} ⟼ $(i,6)$

0_i 1_{i+1} 2_{i+1} 3_{i+1} 4_{i+1} 5_{i+1} 6_{i+1} 7_{i+1} ⟼ $(i,7)$

$L-1$. If everything is perfectly registered, then we expect that the first $L-j$ index tables exhibit the index i at this pixel while the final j get the index $i+1$. For real life stripe images the unsure position may get the values different than i and $i+1$. However, for offsets $j \in [0, L/2)$, we expect that majority of indexes (votes for) are equal to i while for $j \in (L/2, L)$ the majority of votes is for $i+1$. If $j = L/2$ the majority principle cannot be used.

Let for the pixel (x, y) the index $i_{\max}(x, y) \in [0, K)$ receives the maximum votes:

$$i_{\max}(x,y) = \arg\max_i \left|\{j : I_j(x,y) = i\}\right|.$$

Let the most frequent index occupies the time interval $[j_{\min}(x,y),\ j_{\max}(x,y)]$.

We say that the index $I_j(x,y)$ is the wrong index (denoted by ε) if and only if:

$$\left|I_j(x,y) - i_{\max}(x,y)\right| > 1$$

or

$$I_j(x,y) = i_{\max}(x,y) - 1 \text{ and } j_{\max} < j$$

or

$$I_j(x,y) = i_{\max}(x,y) + 1 \text{ and } j_{\min} > j.$$

Identification of sub-stripes requires corrections of wrong indexes at borders.

The lower part of Table 1 illustrates the change of a code index for eight sub-stripes $j = 0,\ldots,7$ of the given stripe i. The column t represents the calculated indexes at time $t = 0,\ldots,7$ for all eight sub-stripes. The time intervals where for the given sub-stripe the registered code indexes are trustful, are outlined by boxes.

The concept of trustful indexes can help us in a simple correction process which is performed as a pixel-wise operation. To understand the idea, consider any pixel (x, y) and its offset indexes and quantisation error

$$i_k \doteq I_k(x,y),\ \ q_k \doteq Q_k(x,y),\ \ k = 0,\ldots,L-1.$$

Then from the Table 1 we see that for any sub-stripe there are two runs of trusted indexes (one of them empty for sub-stripes $j = 0, L-1$) separated (modulo L) by two untrusted values.

We formulate the following decision rules to recover wrong indexes for any pixel (x, y), with k denoting time moment when a change of index at (x, y) occurs:

1. If $0 < k < L-1$ and $I_k(x,y) = I_{k+1}(x,y) = \varepsilon$, then $j = L - k - 1$, $I_k(x,y) = I_{k-1}(x,y)$, and $I_{k+1}(x,y) = I_k(x,y) + 1$.
2. If $I_0(x,y) = I_1(x,y) = \varepsilon$, then $j = L-1$, $I_0(x,y) = I_{L-1}(x,y) - 1$, and $I_1(x,y) = I_{L-1}(x,y)$.
3. If $I_0(x,y) = I_{L+}(x,y) = \varepsilon$, then $j = 0$, $I_0(x,y) = I_{L-1}(x,y) = I_1(x,y)$.
4. If $I_k(x,y) = \varepsilon$ and $I_{k+1}(x,y) = I_{k-1}(x,y) + 1$, then if $Q_{k-1}(x,y) < Q_{k+1}(x,y)$, then $j = L-k-1$, and $I_k(x,y) = I_{k-1}(x,y)$ otherwise $j = L-k$ and $I_k(x,y) = I_{k+1}(x,y)$.[1]

[1] To be used in diagrams, this condition is denoted by W_k.

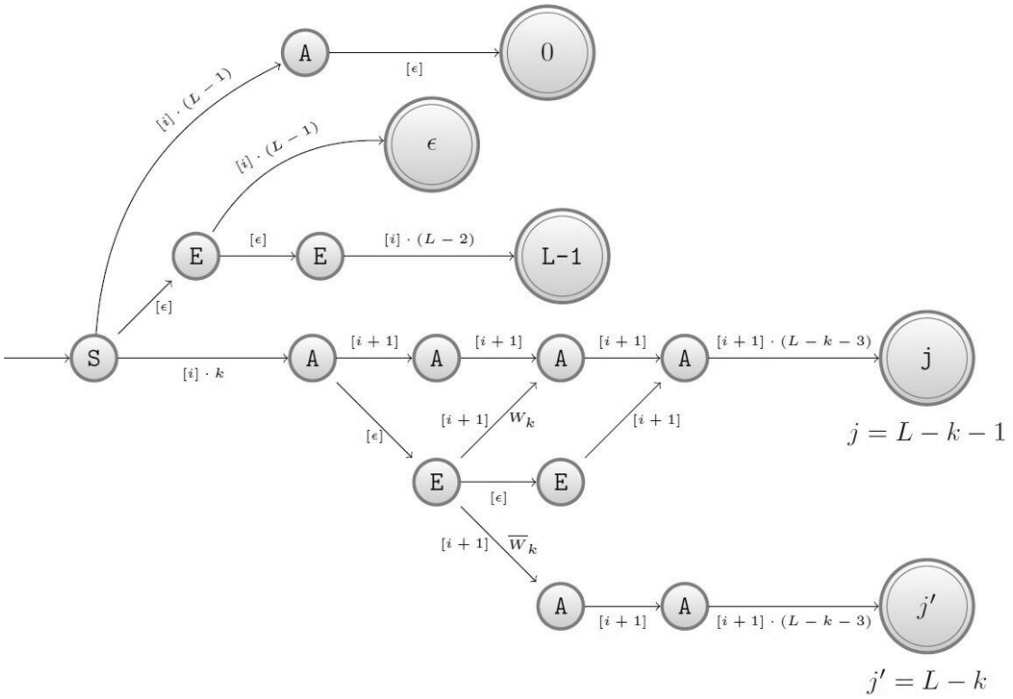


Fig. 18. State machine to make fine indexing for sub-stripes.

Then, we can detect the sub-stripes using the state machine defined in Fig. 18. It presents only sub-stripe detection. In the diagram we use the following notations for definitions of a symbol string:

1. $[s]$ is the single occurrence of symbol s,
2. $[s] \cdot E$ is the symbol s repeated times the value of expression E.

Having corrected the index images we determine the sub-stripe index images I_s which will be useful at the design of calibration tables, described in the next section:

$$I^s(x, y) = L \cdot I_0(x, y) + j(x, y), \tag{2}$$

where I_0 and j are determined by the above decision rules.

7. Calibration tables for 3D viewing

The calibration of a structured light camera is a procedure which prepares data structures for basic applications such as 3D viewer and 3D modeller. Real time viewing requires fast identification of 3D points for which video and stripe camera images are available. To this goal, the following correspondences are established and stored in relevant tables:

1. $d = S_d[i_s, y_s, x_s]$ is the depth index for a 3D point (X, Y, Z) which is observed at pixel (x_s, y_s) of the stripe camera image I_S with the stripe index i_s.
2. $x_v = S_x[d, y_s, x_s]$ is x coordinate of the video camera pixel in the image I_V where the point (X, Y, Z) is observed.
3. $y_v = S_y[d, y_s, x_s]$ is y coordinate of the video camera pixel in the image I_V where the point (X, Y, Z) is observed.
4. $Z = S_Z[d, y_s, x_s]$ is Z coordinate of the observed point.

7.1. Design of calibration tables S_d, S_x, S_y

In the Sect. 4.4, the index image has been extended to sub-stripe precision, i.e. there are L times more possible indexes now and pixels with the same horizontal and vertical indexes create small connected groups. The centroids of these groups for pixels of video camera will be used to define the tables S_x, S_y.

We denote four lists of sub-stripe index images obtained for all calibration planes $d = 0, \ldots, D_{\max} - 1$ as follows:

1. $I^s_{vs}[d]$ is the index for stripe camera image of vertical stripes for the d-th calibration plane.
2. $I^s_{hs}[d]$ is the index for stripe camera image of horizontal stripes for the d-th calibration plane.
3. $I^s_{vv}[d]$ is the index for video camera image of vertical stripes for the d-th calibration plane.
4. $I^s_{hv}[d]$ is index for video camera image of horizontal stripes for the d-th calibration plane.

Let us take any pixel (x_s, y_s) of stripe camera with the sub-stripe indexes (h, v) for the calibration plane with index d:

$$h = I^s_{hs}[d](x_s, y_s), \quad v = I^s_{vs}[d](x_s, y_s).$$

Let now Z_{hv} be the set of pixels in the video camera with the same sub-stripe indexes (h, v):

$$Z_{hv} \doteq \{(x, y)\} : h = I^s_{hv}[d](x_s, y_s), \quad v = I^s_{vv}[d](x_s, y_s).$$

We assign to the pixel (x_s, y_s) the centroid (x_v, y_v) of the set Z_{hv}:

$$(x_v, y_v) \doteq \frac{\sum_{(x,y) \in Z_{hv}} (x, y)}{|Z_{hv}|}.$$

Then, the relevant entries of the calibration tables are filled up

$$S_d\left[\lfloor h/L \rfloor, y_s, x_s\right] \doteq d$$
$$S_x[d, y_s, x_s] \doteq x_v$$
$$S_y[d, y_s, x_s] \doteq y_v.$$

7.2. Design of calibration table S_Z

Having the transition from stripe camera pixels (x_s, y_s) to video camera pixels (x_v, y_v), we can replace the depth index d by the true coordinate Z of the corresponding 3D point.

To this goal we need to know the video camera parameters which enable to determine two inversions:

1. The distortion compensation mapping D^{-1}.
2. The matrix K^{-1} i.e. the inverse of the intrinsic parameters matrix K:

$$K^{-1} = \begin{bmatrix} a_x & 0 & g_x \\ 0 & a_y & g_y \\ 0 & 0 & 1 \end{bmatrix}.$$

The steps to fill the table entry $S_Z[d, y_s, x_s]$ are as follows:

1. Readout the corresponding pixel in the video camera:

$$x_v = S_x[d, y_s, x_s], \quad y_v = S_y[d, y_s, x_s].$$

2. Change coordinates from pixel to camera coordinate system:

$$x' = a_x x_v + g_x, \quad y' = a_y y_v + g_y.$$

3. Compensate the optical distortion of the video camera:

$$(x, y) = D^{-1}(x', y').$$

4. Establish the Z coordinate of the point which belongs to the calibration plane with the index d, the translation c_d, and the normal vector r_z^d with the ray passing through the point $(x, y, 1)$:

$$Z[x, y, 1] r_z^d = c_d^t r_z^d \rightarrow Z = \frac{c_d^t r_z^d}{[x, y, 1] r_z}.$$

where c_d is the translation vector for the calibration plane.

7.3. 3D viewer for structured light camera

The calibration tables S_d, S_x, S_y, S_Z together with intrinsic parameters of video camera (including distortion compensation mapping) allows for textured visualization of a dynamic scene in real time.

The viewer of real 3D scenes which is to be implemented in contemporary graphics hardware needs [44, 45]:

1. The specification of geometry:
   a) the vertexes are defined by the mapping of stripe camera pixels:

$$(x_s, y_s) \mapsto (x_v, y_v) \mapsto (X, Y, Z);$$

   b) the geometric primitives are build as a strip of standard, pixel-based, triangles in the stripe image domain.
2. The specification of texture:
   a) the texture image is video camera image;
   b) the texture coordinates are pixel coordinates (x_v, y_v) in video camera image which are mapped from stripe camera pixel (x_s, y_s).
3. The specification of a vertex shading program (shader) which implements the following steps for any input vertex attribute data (x_s, y_s):
   a) find stripe index for stripe image pixel (x_s, y_s) from index image I^s:

$$i_s = I^s[y_s, x_s];$$

   b) read out the plane index d:

$$d = S_d[i_s, y_s, x_s];$$

   c) establish 3D point coordinates (X, Y, Z):
      i) read out the corresponding video pixel (x_v, y_v):

$$x_v = S_x[d, y_s, x_s], \quad y_v = S_y[d, y_s, x_s];$$

      ii) change to camera coordinate system:

$$x' = a_x x_v + g_x, \quad y' = a_y y_v + g_y;$$

      iii) compensate the distortion:

$$(x, y) = D^{-1}(x', y');$$

      iv) read out the Z coordinate:

$$Z = S_Z[d, y_s, x_s];$$

      v) compute the X, Y coordinates:

$$X = x \cdot Z, \quad Y = y \cdot Z.$$

4. The specification of a geometry shader which simply removes the triangles with wrongly specified vertexes.
5. The specification of a fragment (pixel) shader is trivial as it draws the colour from the video image as the value $I_V[x_v, y_v]$.

In Fig. 19 there are rendered images of a static scene (chair, teabox, bottle) from different angles and distances.

In Fig. 20 we observe the sample of four rendered images of a dynamic scene with included effects of the virtual camera.

8. Conclusions

The proposed system, based on the fast 3D reconstruction algorithm and parallel processing technique in GPU, allows to realize high-resolution, real-time 3D shape measurement at a frame rate of up to 25 frames/s and a resolution of 640×480×256 in a 3D space together with 1440×1080 surface ovelaid, video resolution.

The usage of GPU computational power enabled implementation of real time viewer of complexity reduced to less

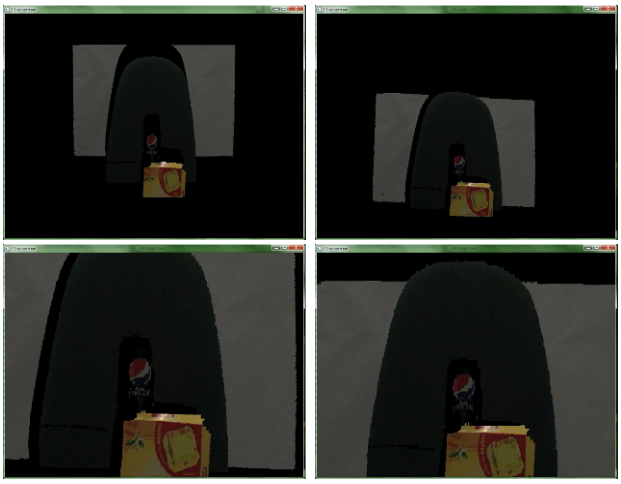


Fig. 19. Example of rendered static scene from different viewpoints.

than 15% of CPU time on contemporary, off shelf PC computers.

The CPU memory usage is dominated by frame buffers' size for stripe and video cameras. The extra memory is necessary for index image. However, the significant memory consumption occurs in GPU: besides the index image, and the video image (refreshed 25 fps), we store tables S_d, S_y, S_y, s, together bounded by 0.25 GB.

In the conducted experiments with 3D viewer of structured light camera (SCC) certain measurement errors were found. Significant part of them would dissapear if we could increase the projector's illumination. Another part is due to sparse sampling of depth volume – currently, there are only 20 calibration screen locations, but it can be increased up to 200 locations. However, the errors coming from depth edges are unovoidable and they have higher impact on the final image quality. Their partial removal is possible by increasing the projector power, more accurate binarisation scheme, and finally in a postprocessing step.

Fig. 20. Example of the rendered dynamic scene from different viewpoints.

The main achievement is the real time visualisation of HD resolution in colour and, potentially, SD resolution in a 3D space.

The main reason of the success is the novel design of several look-up tables which hide nonlinearities of the optical system and solve the 3D correspondence problem.

Acknowledgements

This work was supported by the Ministry of Science and Higher Education under the grant no. N N516 415738.

References

1. S.J Koppal, S. Yamazaki, S.G. Narasimhan, "Exploiting dlp illumination dithering for reconstruction and photography of high-speed scenes", *Int. J. Comput. Vision* **96**, 125–144 (2012).
2. V.C. Paquit, K.W. Tobin, J.R. Price, and F. Mèriaudeau, "3D and multispectral imaging for subcutaneous veins detection", *Opt. Express* **17**, 11360–11365 (2009).
3. W. Gao, L. Wang, and Z.Y. Hu, "A flexible method for structured light system calibration", *Opt. Eng.* **47**, 083602 (2008).
4. Q.A. Li, M. Biswas, M.R. Pickering, M.R. Frater, "Accurate depth estimation using structured light and passive stereo disparity estimation", *IEEE Int. Conf. Image Process.*, Brussels, pp. 969–972 (2011).
5. http://en.wikipedia.org/wiki/Structured_light
6. G. Wiora, "High resolution measurement of phase-shift amplitude and numeric object phase calculation", *Proc. SPIE* **4117**, 289–299 (2000).
7. Microsoft Kinect. Available online: http://www.xbox.com/en-us/kinect/ (accessed on).
8. K. Khoshelham, S.O. Elberink, "Accuracy and resolution of kinect depth data for indoor mapping applications", *Sensors* **12**, 1437–1454 (2012).
9. J. Ghring, "Dense 3-d surface acquisition by structured light using off-the-shelf components", *Proc. SPIE Videometrics and Optical Methods for 3D Shape Measurement* **4309**, 220–231 (2001).
10. E. Horn and N. Kiryati, "Toward optimal structured light patterns", *Image Vision Comput.* **17**, 87–97 (1999).
11. E. Trucco, R.B. Fisher, A.W. Fitzgibbon, and D.K. Naidu, "Calibration, data consistency and model acquisition with laser stripers", *Int. J. Computer Integrated Manufacturing* **11**, 293–310 (1998).
12. R.J. Valkenburg and A.M. McIvor, "Accurate 3d measurement using a structured light system", *Image Vision Comput.* **16**, 99–110 (1998).
13. J.L. Posdamer and M.D. Altschuler, "Surface measurement by space-encoded projected beam systems", *Comput. Graph. Image Process.* **18**, 1–17 (1982).
14. Z.J. Geng, "Rainbow 3-dimensional camera: New concept of high-speed 3-dimensional vision systems", *Opt. Eng.* **35**, 376–383 (1996).
15. C. Wust and D.W. Capson, "Surface profile measurement using colour fringe projection", *Mach. Vision Appl.* **4**, 193–203 (1991).

16. T. Pajdla, "Bcrf – binary-coded illumination range finder reimplementation", *Technical report KUL/ESAT/MI2/9502*, Katholieke Universiteit Leuven, ESAT, Leuven, 1995.
17. P. Lavoie, D. Ionescu, and E. Petriu, "A high precision 3D object reconstruction method using a colour coded grid and nurbs", *Proc. Int. Conf. Image Analysis and Processing*, pp. 370–375, Venice, 1999.
18. J. Tajima and M. Iwakawa, "3-D data acquisition by rainbow range finder", *Proc. IEEE Int. Conf. Pattern Recogn.*, pp. 309–313, Atlantic City, 1990.
19. D. Bergmann, "New approach for automatic surface reconstruction with coded light", *Proc. SPIE Remote Sensing and Reconstruction for Three-Dimensional Objects and Scenes*, Vol. **2572**, pp. 2–9, San Diego, 1995.
20. M. Ito and A. Ishii, "A three-level checkerboard pattern (TCP) projection method for curved surface measurement", *Pattern Recogn.* **28**, 27–40 (1995).
21. S. Kiyasu, H. Hoshino, K. Yano, and S. Fujimura, "Measurement of the 3-D shape of specular polyhedrons using an m-array coded light source", *IEEE T. Instrumentation and Measurement* **44**, 775–778 (1995).
22. S. Inokuchi, K. Sato, and F. Matsuda, "Range-imaging for 3-D object recognition", *Int. Conf. Pattern Recogn.*, pp. 806–808, Montreal, 1984.
23. W. Krattenthaler, K.J. Mayer, and H.P. Duwe, "3D-surface measurement with coded light approach", *Proc. 17th Meeting of the Austrian Association for Pattern Recognition on Image Analysis and Synthesis*, Vol. 12, pp. 103–114, 1995.
24. K. Sato, "Range imaging based on moving pattern light and spatio-temporal matched filter", *IEEE Int. Conf. Image Process.* 1, pp. 33–36, Lausanne, 1996.
25. T. Monks and J. Carter, "Improved stripe matching for colour encoded structured light", *5th Int. Conf. Computer Anal. Images and Patterns*, pp. 476–485, Budapest, 1993.
26. E.M. Petriu, Z. Sakr, S. H. J. W., and A. Moica, "Object recognition using pseudo-random colour encoded structured light", *Proc. IEEE 17th IEEE Instrumentation and Measurement Technology Conference*, Vol. 3, pp.1237–1241, Baltimore, 2000.
27. H. Morita, K. Yajima, and S. Sakata, "Reconstruction of surfaces of 3-d objects by m-array pattern projection method", in *IEEE Int. Conf. Comput. Vision*, pp. 468-473, Tampa, 1988.
28. J. Salvi, J. Pages, and J. Batlle, "Pattern codification strategies in structured light systems", *Pattern Recogn.* **37**, 827–849 (2004).
29. C. Chen, Y. Hung, C. Chiang, and J. Wu, "Range data acquisition using colour structured lighting and stereo vision", *Image Vision Comput.* **15**, 445–456 (1997).
30. J. Salvi, J. Batlle, and E. Mouaddib, "A robust-coded pattern projection for dynamic 3d scene measurement", *Int. J. Pattern Recogn. Lett.* **19**, 1055–1065 (1998).
31. P. Griffin, L. Narasimhan, and S. Yee, "Generation of uniquely encoded light patterns for range data acquisition", *Pattern Recogn.* **25**, 609–616 (1992).
32. I. Ishii, K. Yamamoto, K. Doi, and T. Tsuji, "High-speed 3D image acquisition using coded structured light projection", *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 925–930, San Diego, 2007.
33. L. Zhang, B. Curless, and S.M. Seitz, "Rapid shape acquisition using colour structured light and multi-pass dynamic programming", *Int. Symp. on 3D Data Processing Visualization and Transmission*, Padova, 2002.
34. P. Fechteler and P. Eisert, "Adaptive colour classification for structured light systems", *IET J. Comput. Vision* **3**, 49–59, 2009.
35. J. Pages, J. Salvi, C. Collewet, and J. Forest, "Optimised De Bruijn patterns for one-shot shape acquisition", *Image Vision Comput.* **23**, 707–720 (2005).
36. D. Caspi, N. Kiryati, and J. Shamir, "Range imaging with adaptive colour structured light", *Pattern Anal. Machine Intel.* **20**, 470–480 (1998).
37. K.L. Boyer and A.C. Kak, "Colour-encoded structured light for rapid active ranging", *IEEE T. Pattern Analy. and Machine Intel.* **9**, 14–28 (1987).
38. H. Fredricksen, "A survey of full length nonlinear shift register cycle algorithms", *Society of Industrial and Applied Mathematics Review* **24**, 195–221 (1982).
39. P. Vuylsteke and A. Oosterlinck, "Range image acquisition with a single binary-encoded light pattern", *IEEE T. Pattern Anal. and Machine Intel.* **12**, 148–163 (1990).
40. O. Hall-Holt and S. Rusinkiewicz, "Stripe boundary codes for real-time structured-light range scanning of moving objects", in *8th IEEE Int. Conf. Comput. Vision*, pp. 359–366, Vancouver, 2001.
41. Z. Zhang, "A Flexible New Technique for Camera Calibration", *IEEE T. Pattern Anal. and Machine Intel.* **22,** 1330–1334 (2000).
42. C. Harris and M. Stephens, "A combined corner and edge detector", *Proc. 4th Alvey Vision Conf.*, Manchester, 1998.
43. O. Faugeras, *Three-Dimensional Computer Vision*, edited by MIT Press, Cambridge, 1993.
44. M. Pharr and G. Humphreys, *Physically Based Rendering*, edited by Morgan-Kauffman, Burlington, 2004.
45. G. Sansoni, S. Lazzari, S. Peli, and F. Docchio, "3d imager for dimensional gauging of industrial workpieces: state of the art of the development of a robust and versatile system", *Int. Conf. Recent Advances in 3-D Digital Imaging and Modeling*, pp. 19–26, Ottawa, 1997.

