Exploiting Time-Multiplexing Structured Light with Picoprojectors

Mario Valerio Giuffrida\textsuperscript{a}, Giovanni Maria Farinella\textsuperscript{a}, Sebastiano Battiato\textsuperscript{a}, Mirko Guarnera\textsuperscript{b}

\textsuperscript{a}IPLab, DMI, University of Catania, Viale Andrea Doria, 6, Catania, Italy; \textsuperscript{b}STMicroelectronics, Stradale Primosole 50, Catania, Italy

ABSTRACT
When a picture is shot all the information about the distance between the object and the camera gets lost. Depth estimation from a single image is a notable issue in computer vision. In this work we present a hardware and software framework to accomplish the task of 3D measurement through structured light. This technique allows to estimate the depth of the objects, by projecting specific light patterns on the measuring scene. The potentialities of the structured light are well-known in both scientific and industrial contexts. Our framework uses a picoprojector module provided by STMicroelectronics, driven by the designed software projecting time-multiplexing Gray code light patterns. The Gray code is an alternative method to represent binary numbers, ensuring that the hamming distance between two consecutive numbers is always one. Because of this property, this binary coding gives better results for depth estimation task. Many patterns are projected at different time instants, obtaining a dense coding for each pixel. This information is then used to compute the depth for each point in the image. In order to achieve better results, we integrate the depth estimation with the inverted Gray code patterns as well, to compensate projector-camera synchronization problems as well as noise in the scene. Even though our framework is designed for laser picoprojectors, it can be used with conventional image projectors and we present the results for this case too.

Keywords: Structured light, Gray code patterns, Time-multiplexing, Picoprojector.

1. INTRODUCTION
Every time we look at a picture, we are actually looking at a 2D projection of the 3D world surrounding us. The third dimension, the depth, is lost once a picture is taken by a conventional camera device. Finding the correct estimation of the missing dimension is a challenging problem in computer vision and several successful techniques were developed in the past years\textsuperscript{[2]}-\textsuperscript{[4]} Exploiting the objects’ distance, different tasks can be accomplished, such as 3D object reconstruction\textsuperscript{[2]}-\textsuperscript{[4]}, object recognition\textsuperscript{[3]}-\textsuperscript{[4]}, pedestrian detection\textsuperscript{[3]}-\textsuperscript{[4]} and so on.

Most of the applications related with depth estimation, the employed camera system needs external hardware. A way to calculate the object’s distance is to use a laser beam to measure the photon’s time-of-flight\textsuperscript{[5]} needed to go from the source and come back to a receiver. For example, the recent LG D722 mobile phone is equipped with an infrared laser, which is used by the camera application to detect distances to focus the scene faster. Another method to estimate the depth in a scene is the stereo vision\textsuperscript{[6]} which uses two pictures of the same scene with a small horizontal (or vertical) displacement to get an estimation about the distances of the objects within the scene. Usually stereo vision is achieved with two cameras placed one next to each other. Stereo vision exploits the parallax effect, which gives to the observer the perception that farther objects (like mountains) move slower than nearer ones. A huge amount of disparity between two corresponding points in the stereo images imply a nearer distance from the camera, whereas small displacement is related with farther objects. The key idea in stereo vision is to find the correct correspondence between a given point in one image with respect to the other

\textsuperscript{[2]} Exploiting Time-Multiplexing Structured Light with Picoprojectors

Further author information: (Send correspondence to M.V. Giuffrida)

M.V. Giuffrida: valerio.giuffrida88@gmail.com
G.M. Farinella: gfarinella@dmi.unict.it
S. Battiato: battiato@dmi.unict.it
M. Guarnera: mirko.guarnera@st.com
Table 1: Difference between classic binary code and Gray code representation for the first 8 decimal numbers. The two columns headed as H report the hamming distance between the i-th row and the i+1 row for both binary and Gray code. In the latter case, the hamming distance is always constant.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary Code</th>
<th>H</th>
<th>Gray code</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>1</td>
<td>000</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>2</td>
<td>001</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>1</td>
<td>011</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>3</td>
<td>010</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>1</td>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>2</td>
<td>111</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>1</td>
<td>101</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>-</td>
<td>110</td>
<td>-</td>
</tr>
</tbody>
</table>

Therefore, another way to compute the object’s distance is the employment of the *structured light*. This technique replaces one of the camera in the stereo vision system with an image projector, allowing to determine the depth in the scene by using specific light patterns. According how those patterns deform on top of the objects’ surface, it is possible to calculate the amount of distance between the objects and the camera. The Microsoft Kinect is a piece of technology that successfully implements this kind of technique to detect the players in the scene. In a nutshell, structured light encodes pixel information inside the light patterns. When the measuring scene is acquired by a camera, each point belonging to the pattern is triangulated with the pixel position in order to get the depth information. Light patterns are useful to establish a correspondence for a given point between the image plane and the projector plane. There are different patterns that can be used as structured light, summarized into the following groups: direct coding, spatial neighborhood and time-multiplexing. In the first case, it is used a light pattern which each pixel is univocally labeled by a codeword. In spatial neighborhood, the pixel codeword is retrieved by the information taken from near points around it. In the last case, a set of specific patterns is projected in different instant of time. The codeword for each pixel is retrieved from all the patterns shot in the measurement scene.

In this paper we present a hardware and software framework, which is able to make depth estimation through time-multiplexing structured light patterns. The hardware setup includes a laptop, a webcam and a picoprojector, which is used to project patterns onto the measuring scene to be reconstructed. In our test we employed Gray code patterns, which is an alternative representation to the normal binary numeral system. The peculiarity of Gray code binary number representation is that two successive numbers have *always* hamming distance equal to 1 (Table 1). The main reason in using Gray code patterns for structured light is due to its robustness to the bit flipping, that could be caused by the presence of noise in the scene. Another advantage in using Gray code light patterns is because it is possible to achieve high performance in depth estimation. In order to achieve an even better depth estimation, we add an inverted Gray code patterns to the acquisition sequence. Those patters, used beside the not-inverted Gray code, will be used to compensate noise in the scene.

The remaining of this paper is organized as follow: in Section 2 a review about depth estimation through structured light techniques is discussed. In Section 3 the theoretical background of the proposed method is presented, whereas in Section 4 we show both the hardware and software framework used in the experiments and results will be shown in Section 5. Finally, Section 6 presents the conclusions and some suggestions for future works.

2. RELATED WORKS

As briefly introduced earlier, there are many ways to achieve the depth estimation task. We will focus the literature review on the methods we are interested in, namely the ones involving the structured light. Following the taxonomy in *Salvi et al.* we present some the state-of-the-art structured light methods in the following subsections.
2.1 Direct coding

Direct coding uses light patterns to establish a codeword for each pixel on the measuring scene. One of the former method using it is the Rainbow Range Finder by Tajima et al.\textsuperscript{13} They use a white light and a lens to obtain the rainbow pattern, since white light is a summation of all of the light spectrum. Because the pattern is bounced by a mirror, it is required a rigorous calibration system camera-projector (mirror in this case) to correctly estimate the distance.\textsuperscript{14} In Sato\textsuperscript{15} this idea was improved by replacing the white lamp system with an LCD projector (Figure 1). The author shifts the rainbow phase by 1/3, shooting to the objects three slightly different patterns. In this way they achieved two important results: better performances with small objects and compensation with the spectral reflectance of colored surfaces.\textsuperscript{1,15} The aforementioned methods are not suitable for moving objects, although perform well just in static conditions. A functional example of Rainbow Range Finder working on dynamic scene was proposed by Geng.\textsuperscript{16}

2.2 Spatial neighborhood

Differently than direct coding, spatial neighborhood encodes all the information in one pattern and retrieves the codeword of a pixel from a set of points around it. Because of this peculiarity, most of the spatial neighborhood patterns are more sophisticated than direct coding strategies (and time-multiplexing as well, as it will be shown afterward), making the decoding stage more expensive in term of computational cost.\textsuperscript{1} In order to find the right correspondence between points in the image into the projector plane, Maruyama et al.\textsuperscript{17} proposed a pattern made by randomly interrupted slits (Figure 2a). When decoding the position of a specific point, the length of the correspondent segment is measured and correlated with the other ones in the pattern. Since there could be more than a valid matching in correlation, the correspondence is improved done by using 6 adjacent segments. Because of the randomness of the slit patterns and the position of six adjacent segments, the matching stage is likely to be unique. One of the drawback of this method is that the segment’s length could be subjected to measurement error, due to the distance between the camera and the objects placed in the measuring scene.

A different way to generate spatial neighborhood patterns is the employment of pseudo-random algorithms, such as the De Bruijn sequence.\textsuperscript{15} This kind of sequence can generate pseudo-random strings $S$ of length $|S| = k^n$ from an alphabet $A = \{a_1, a_2, \ldots, a_k\}$ of $k$ elements. The only constraint imposed in De Bruijn sequence is that every possible substring of $n$ elements has to appear only once in $S$. There are many methods to generate a De Bruijn sequence. One of them is to build a De Bruijn graph and find either a Hamiltonian path (passing through all the vertexes) or an Eulerian path (passing through all the edges) on it. In Lavoie et al.\textsuperscript{2} they use an alphabet $A$ of $k = 5$ elements $A = \{R = \text{Red}, G = \text{Green}, C = \text{Cyan}, Y = \text{Yellow}, M = \text{Magenta}\}$ and each substring is $n = 5$ elements long. The blue color was not included in the alphabet, because camera is not able to capture
it well. With this configuration they generate a pseudo-random sequence of $|S| = 5^3 = 125$ elements, where the alphabet’s elements are the colors assigned to a grid pattern, as it is shown in Figure 2b. The matching process is done by exploiting the window property for each subsequence $n = 5$ long appears only once in $S$, it is sufficient to look up in $S$ for the sequence decoded from the camera (both in horizontal and vertical) to find the correct location inside the pattern.

Another successful technique to achieve depth estimation through structured light is the Microsoft Kinect device, which it achieves the depth estimation task in real time. The device includes a RGB camera, an infrared light emitter and an infrared sensor. The IR emitter transilluminates a transparency, which depicts a pseudo-random sparkle pattern on it (Figure 2c). A processor is involved to elaborate the input image and extract the three-dimensional map.\cite{11}

2.3 Time-multiplexing

In the time-multiplexing approaches, a sequence of patterns is projected onto the scene and the codeword of each pixel is gathered from all the values in the projected patterns. According to the survey in Salvi et al., this technique tends to outperform other category methods in term of accuracy. Since several specific patterns are projected onto the measurement scene, they are typically simple (e.g., binary patterns). Moreover, since the set of basis employed in those techniques is easy, the decoding step is usually fast and very accurate. Time-multiplexing method uses a coarse-to-fine paradigm, that is a light pattern projected at time $t + 1$ improves the

*Microsoft Kinect includes other features, but they are not essential for depth estimation.
pattern. This is the coarsest one and it is the MSB of the codeword. The first bit of the codeword for the red area is 0 (black region).

(b) $2^1$ pattern. In this case, the second bit of the codeword for the red region is 1, because it falls onto the white region.

(c) $2^2$ pattern. As in the previous case, the red area is on the white region and the third bit is coded with 1.

(d) $2^3$ pattern. In this case, the LSB is encoded with 0, since the red area falls in the black region of the light pattern.

Figure 3: An example of Binary patterns and codeword for the region marked in red. In this case, the region of interest is coded with the sequence 0110$_2$. This is the idea behind the binary coding.

estimation of a given pixel determined at time $t$. In the past years several different patterns were developed for the distance estimation task through time-multiplexing light pattern.

A successful set of structured light patterns is the binary code, which is used for the first time by Posdamer et al. The technique projects $m$ patterns encoding up to $2^m$ values in binary representation. The Most Significant Bit (MSB) of the sequence is retrieved by the first pattern (Figure 3a), detecting in which area a given pixel falls into. The operation is repeated until the Least Significant Bit (LSB), the number $m$ (Figure 3d) and the final codeword is composed. Figure 3 presents an example where, across all the $m = 4$ patterns, the codeword associated with the red area is 0110$_2 = 6_{10}$. This approach was improved by Inokuchi et al. using Gray code patterns, namely another binary representation convention. The conversion from classic binary code to Gray code is done by bitwise XOR operation: let $b = (b_0, b_1, \ldots, b_{m-1})$ be a binary sequence such that $b_i \in \{0, 1\}$ and let $\hat{b} = (0, b_1, b_2, \ldots, b_{m-2})$ be the $b$ sequence shifted by one position to the right. The conversion of the binary number $b$ to the correspondent Gray code sequence $g$ is $g = (g_0, g_1, \ldots, g_{m-1}) = b \text{ XOR } \hat{b}$. Each value $g_i$ is calculated by the XOR between the $b_i$ and $b_{i-1}$, for each $i = 1, 2, \ldots, m - 1$. The MSB remains the same for both of the sequences. For example, the conversion from binary code to Gray code for the number $47_{10} = 101111_2$ is $101111 \text{ XOR } 010111 = 111000$. The inverse conversion is calculated differently: let $g_i = b_i \text{ XOR } b_{i-1}$ be the result of the Gray code conversion from a binary digit and one wants to find the
The inverse conversion sequence $b$ is obtained by $g$ applying the XOR each bit in $g$ with the previous operation outcome $b = g_i \text{ XOR } b_{i-1}, i = 1, 2, \ldots, m - 1$. Even in this case the MSB $b_0 = g_0$ remains the same in both $g$ and $b$. For the sake of clarity, the backward conversion of 111000 in Gray code to binary code is 111000 XOR 010111 = 101111. The conversion mask is the same for both conversion and it is a right shift of the binary code representation of the number to convert. Figure 4 shows the difference between binary code and Gray code number from 0 to 15. This generates a binary tree, which gives a visual overview on how the numeral systems expands in depth. Gray code results to be more robust to noise, as well as it is easy to implement, requiring just a few computational resources.

3. THEORICAL BACKGROUND

In order to have a better depth estimation, it is required to calibrate the camera-projector system. Camera calibration is a numerical method that obtains intrinsic and extrinsic parameters. Intrinsic camera parameters are related with the inner configuration of the device itself, such as focal length, image sensor format and principal point. Extrinsic parameters refer to the 3D geometry of the world. As it is shown in Figure 1, camera and projector are placed in two different locations and rotated in opposite directions. Camera calibration comes from stereo vision but since the projector replaces a camera, the same theory is applied for the camera-projector system. In Falcao et al. the authors addresses the problem separately: they first calculate the camera calibration parameters with Zhang’s algorithm. Then, they solve the ray-plane intersection to find the projector’s parameters. When a camera-projector system is well calibrated, one gets the following achievement:

1. object’s scale factor;
2. absolute distance of the object from the camera;
3. image rectification (Figure 5).

If we neglect scale factor and absolute distance to the objects in the scene, the last remaining point is the image rectification. In our case the scale factor is not considered, because it is not important to know the absolute size of a scanned object. Still, it is not even required how many meter it is far away an object. Rectification is important to solve image distortion generated by the projection onto the camera plane. Moreover, because we are dealing with the picoprojector, which are as small as a wallet, the camera can be physically placed on top of the it, minimizing the perspective distortion of the image onto the camera plane. Nevertheless, this trivial configuration can still cause a perspective distortion in the camera plane and we solve this problem by applying a homography matrix $H$. The homography matrix transforms a set of point laying in a plane into anther one:

$$
\begin{bmatrix}
    u' \\
    v' \\
    w'
\end{bmatrix}
= 
\begin{bmatrix}
    h_{11} & h_{12} & h_{13} \\
    h_{21} & h_{22} & h_{23} \\
    h_{31} & h_{32} & h_{33}
\end{bmatrix}
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}
$$

(1)
Suppose to have a set of points $P = \{p_1, p_2, \ldots, p_n\}$ laying in a plane $\Pi$, the source plane, and another set of points $P' = \{p'_1, p'_2, \ldots, p'_n\}$ laying $\Pi'$, the destination plane, such that $p_i = (u_i, v_i, w_i)$ and $p'_i = (u'_i, v'_i, w'_i)$ 2-D points expressed in homogeneous coordinates. A homography matrix $H \in \mathbb{R}^{3 \times 3}$ is a transformation matrix mapping points from $\Pi$ into $\Pi'$. In Equation 1, $H$ is the homography matrix, $p$ is a point in $\Pi$ and $p'$ is the corresponding point in $\Pi'$. The homography matrix $H$ can be computed by Direct Linear Transform (DLT).

Points in $\Pi$ and $\Pi'$ are used to build a matrix $A_i$, as it is shown in (2). By stacking all the $A_i$ matrices and let $A$ be the new matrix generated by this operation, we have to solve the linear system in (3). The solution of the linear system is the column vector $h$, which is a vectorization of the homography matrix. Vector $h$ is calculated by Single Value Decomposition $A = U \Sigma V^*$, by taking the eigenvector in $U$ corresponding to the least eigenvalue. Equation 3 has solution iff $n \geq 4$ and points $P$ and $P'$ are not collinear in their planes respectively.

\[
A_i = \begin{bmatrix}
-u_i & -v_i & -1 & 0 & 0 & u'_i & u'_i v_i & u'_i \\
0 & 0 & 0 & -u_i & -v_i & -1 & v'_i & v'_i \\
\end{bmatrix}
\]

\[
Ah = \begin{bmatrix}
A_1 \\
A_2 \\
\vdots \\
A_n
\end{bmatrix}
\begin{bmatrix}
h_1 \\
h_2 \\
\vdots \\
h_n
\end{bmatrix} = 0
\]

To achieve our goal to determine object’s distance from the camera, we have to calculate the disparity map. Let $I_L$ and $I_R$ be the left and right image of the Figure 5 respectively and let $p = (u, v)$ and $p' = (u', v')$ be a point in $I_L$ and $I_R$ respectively corresponding to the same point in the 3D space. Supposing that the images are taken by two cameras horizontally located with a certain amount of distance, the disparity map $D$ is defined as the difference between $u$ and $u'$: $D(p) = |u - u'|$. When the difference tends to 0, the distance of the object tends to be infinitely far away; whereas when the difference tends to infinity, object tends to be very close to the camera. In our case, because we have placed the camera right on top of the projector, the disparity map is calculated between $v$ and $v'$ instead. In order to find the correspondence between a point $p$ in $I_L$ to a point $p'$ in $I_R$, similarity metrics can be used, such as Sum of Squared Differences (SSD), Absolute Sum of Differences (ASD), Normalized Cross Correlation (NCC) and many more. This techniques are expensive in terms of computational cost and this is the reason leading us in employing structured light. Since a camera-projector system was employed, the value $v'$ is replaced by the codeword of the pixel in that point, as it is extracted in Figure 3.
4. PROPOSED FRAMEWORK

In this section we explain both hardware and software equipments of the proposed system. The hardware framework is presented first, as it is depicted in Figure 6. On the software framework, it is presented the set of tools we used in order to determine the homography matrix, to project pattern onto the measuring scene, and to calculate the disparity map.

4.1 Hardware framework

The hardware setup is composed by three devices: a workstation, a webcam and a projector (Figure 6). For this project we used a netbook Acer A0715h with a GNU/Linux operating system. The camera device is a generic RGB webcam with resolution $640 \times 480$ pixels. The projector can be any kind of projection device, connected to the computer with the external VGA port. In our case, we employed a picoprojector provided by STMicroelectronics Catania and we placed the webcam on top of it. The picoprojector has a resolution $800 \times 600$, exploiting LED lasers as color light source and a MEMS mirror technology to raster the scene with laser spot. The concept is very similar to the conventional CRT displays. A similar solution is also provided by Microvision, and we used their picoprojector device in our tests as well (Figure 9a).

4.2 Software framework

The software was coded in C language with the use of OpenCV. Since the projector is detected as a secondary display device, our software drives the projection process by opening a fullscreen window in the projector area. The software offers the following features:

1. project, acquire and save a sequence of Gray code patterns;
2. from a sequence of ordered sequence of acquisitions, it is able to provide the depth map;
3. using the depth map, it provides the point cloud in the 3D space.

During the pattern projection/acquisition phase, the first projected pattern is a white light, which is used as test pattern in order to find the region of interest to compute the homography matrix. The user has to set manually the four corner points belonging to the projector area. This is the only case where the user interaction is required and it is made as simple as possible, since the user has to select from a set of suggested points prompted on the screen (Figure 7). Suggested points are detected through Harris corner detection algorithm. Since DLT requires at least four distinct points to compute the homography matrix, the corner points selected from the white light pattern are enough to determine the matrix $H$. The corners, starting from the top-left
one in clockwise direction, are respectively mapped to the following point in the destination plane: $p'_1 = (0,0)$, $p'_2 = (w,0)$, $p'_3 = (w,h)$ and $p'_4 = (0,h)$, where $w$ and $h$ are the projector width and height respectively. After the white light is projected, a sequence of $m$ Gray code patterns are projected in sequence. Afterwards, the same sequence of inverted Gray code patterns are projected too. This kind of patterns have the same characteristics of the one discussed in Section [3] but the black color and the white color are inverted instead. By shooting a sequence of inverted Gray code light patterns, it makes possible to obtain more accurate results, because in this way the noise due to the camera sensor, ambient conditions, and picoprojector color distortions can be compensated.

When a pattern sequence is acquired, each frame is stored in different image file, used to compute the disparity map. After the rectification of the picoprojector’s region of interest, all the images will have a new dimension $w' \times h'$, smaller than the one captured originally. Moreover, since Gray code codification encodes $2^m$ values, where $m$ is the number of projected patterns, $h'$ could be different than $2^m$. A small choice of $m$ would lead to inaccurate estimation, whereas a big value $m$ would lead to aliasing problem, because a dense amount of lines could not be rendered properly. We recall that we are calculating the disparity map with vertical displacement: in case one prefers to deal with the horizontal one, the original projector width $w$ and the rectified region of interest width $w'$ have to be considered instead. The codeword is generated by gathering the contribution of Gray code patterns and the inverted ones. Let $I_k$ be an image taken by the camera when normal Gray code pattern was projected, and Let $I'_k$ be the same scene, but the correspondent inverted Gray code pattern were projected instead, for all $k = 1, 2, \ldots, m$. The threshold image $T_k$ is calculated as follows:

$$T_k(u, v) = \begin{cases} 
0 & I_k(u, v) - I'_k(u, v) \leq \tau \\
1 & \text{otherwise} 
\end{cases}$$

where $\tau$ is a fixed threshold. Therefore, the codeword $g$ for a generic pixel $p = (u,v)$ is trivially obtained by concatenation of all the thresholded images $g = (T_1(u,v), T_2(u,v), \ldots, T_m(u,v))$. Then, $g$ can be converted in binary representation and the disparity map can be finally computed. As it has been discussed above, rectified picture and projector have not the same size. In order to have accurate results, it would be necessary to normalize the coordinate systems. The adopted solution is to normalize with respect to the maximum obtainable value, namely the image height for the rectified pictures and $2^m$ for the codewords. In conclusion, the disparity between
a generic point $p$ in $I$ and its binary codeword $b$ is:

$$D(u, v) = \frac{v}{2^m} - \frac{\text{dec}(b)}{2^m - 1}$$

where $\text{dec}$ converts binary number into the decimal representation.

5. EXPERIMENTAL RESULTS

Firstly, we tested our software framework using a conventional image projector with $1024 \times 768$ pixels resolution. A disturbing factor of this test is the presence of the webcam’s shadow in the scene (Figure 8a). Both the camera and the projector have two different fields of view. Hence, the projector has to be placed several meters behind the camera to cover the statue. For this reason, the camera is placed between the projector and the measuring scene, casting a shadow onto the measuring scene. In Figure 8b it is shown the obtained disparity map: blue regions are far away, whereas red regions are near to the camera. The horizontal stripes appearing in the disparity map are caused by the homography applied to all of the frame acquired by the camera.

As second test, we used the picoprojector and the experiments were run inside a dark room with controlled ambient light condition. The Microvision PicoP picoprojector is presented in Figure 9a, where a board with a webcam was placed on top of the picoprojector. We tested the picoprojector-camera system inside a completely
dark room. Unlike the case with the conventional projector based on DLP technology, the pattern projected by the picoprojector, using laser scanning technology, is affected by two problems: the first one is the flickering (darker band moving cyclically from the top to the bottom) and the latter one is the pincushion distortion, both of them are shown in Figure 9b. The pincushion phenomenon is more relevant in projector using biaxial mirror technology (single MEMS scanning both horizontal and vertical direction), then projector uses two different mirrors for horizontal and vertical scanning. The optical distortion is solved by homography matrix, whereas the flickering phenomenon was compensated with the inverted Gray code pattern contribution.

In Figure 10 we show how our framework works on two acquisition processes. In the first case, we placed in the measuring scene a plastic cup (Figure 10a), in order to test whether the white color of the surface affects the codeword extraction. Using the inverted Gray code patterns, we compensate the dark band due to the flickering, as well as we can successfully detect white surfaces. The depth map of this test is shown in Figure 10b. In the second instance we put a toy in the scene, placing it closer to the camera. Due to this configuration, the toy appears closer in Figure 10d (shades of red), than the plastic cup in Figure 10b (shades of blue).

6. CONCLUSIONS

In this paper we presented a fully-functionally hardware and software framework to estimate the depth map in a scene. This problem arises several times in computer vision in different context and they were proposed different methods to achieve this purpose. One of the widely methods to detect the distance of objects in a scene is to employ the structured light, which is inspired from the stereo vision system. In the case of structured light, one of the camera in the stereo system is replaced by a projector, which is driven to shoot special light pattern. By
analyzing how the light patterns react to the objects they met, it is possible to calculate an estimation of the distance between them and the camera.

The proposed framework is based upon the time-multiplexing method proposed by Inokuchi et al.\textsuperscript{22} using Gray code binary light patterns. Because several patterns have to be projected, time-multiplexing methods cannot be used in moving scene, that is objects have to be always in the same position. Nevertheless, binary patterns outperforms more accurate results than other structured light methods discussed earlier. The hardware framework provided by STMicroelectronics (Figure 9a) is made by two devices: a picoprojector, namely a small device able to project images from a workstation, and a webcam mounted on top of it. We also tested our software with a conventional image projector. The experiments involving the picoprojector were run inside a room with controlled ambient light condition. Unlike standard projectors, the images shot by the picoprojector are affected by two disturbance factors: flickering and optical distortion. We dealt with flickering, by employing inverted Gray code patterns, beside the normal ones. Whereas, the optical distortion was solved by applying the homography matrix during image rectification. Experimental results of the proposed method are shown in Figure 8, where we tested our software with a conventional image projector. Still, in Figure 10 we show two example of object acquisition through the picoprojector. The provided results have two problems, which lead to wrong depth estimation: the former one is related to the shadow casting, which mark a determined area with a totally black shape. The latter problem is the presence of horizontal stripes, due to the employment of homography matrix, which do not completely compensate the pincushion distortion.

It is possible to improve the proposed approach in different way, even considering the case in using a conventional image projector. For instance, it would be necessary to avoid detect shadow areas and skip them during te depth estimation. Focusing in the picoprojector case, it would be important to provide a better way to rectify the image, by compensating the optical pincushion distortion, as it is shown in Figure 9b. Furthermore, the picoprojector-camera system would need a webcam with a higher resolution, since the rectified image is small than the picoprojector’s resolution. Because the homography matrix is applied to a region of interest of the original image, it reduces the actual image resolution in that area. Since we tested that picoprojector can be employed for structured light, it would be worthy to test them for moving scene, in order to provide a depth map in real time situations. This should be done in conjunction with a port of the software framework into an embed device (e.g., Raspberry), in order to have a portable device able to calculate depth map in real-time.

REFERENCES


