Screen-Camera Calibration Using Gray Codes

Screen-Camera Calibration Using Gray Codes

Yannick Francken, Chris Hermans, Philippe Bekaert
Hasselt University - tUL - IBBT
Expertise Centre for Digital Media
Wetenschapspark 2, 3590 Diepenbeek, Belgium
{firstname.lastname}@uhasselt.be

Abstract

In this paper we present a method for efficient calibration of a screen-camera setup, in which the camera is not directly facing the screen. A spherical mirror is used to make the screen visible to the camera. Using Gray code illumination patterns, we can uniquely identify the reflection of each screen pixel on the imaged spherical mirror. This allows us to compute a large set of 2D-3D correspondences, using only two sphere locations. Compared to previous work, this means we require less manual interventions, combined with a more robust screen pixel detection scheme. This results in a consistent improvement in accuracy, which we illustrate with experiments on both synthetic and real data.

1 Introduction

Present day personal computers come with a large variety of peripheral devices. One of the most frequently encountered devices, the webcam, has seen a significant increase in quality, while staying relatively cheap compared to other computer peripherals. This has motivated the software industry to create a new array of more demanding applications, aside from its traditional use for video conferencing. One typical example can be found in vision-based interfaces, where the user interacts with the application by simply pointing toward the screen, looking at certain regions, or making gestures that are observed by the camera [4, 14, 6].

Another category of popular new applications are shape acquisition methods, where the screen is used as a planar illuminant, while the camera captures the illuminated scene. For example, methods using a screen-camera setup have been invented for the purpose of photometric stereo [5, 12, 11], specular surface acquisition [9, 23, 27], environment matting [29], and many more applications [17, 22, 19]. The attractive feature that all these techniques have in common, is the fact that a cheap setup consisting only of off-the-shelf components can often replace otherwise expensive hardware setups, while still producing comparable results. An example of a screen-camera setup is depicted in Figure 2.

However, while being cheaper and simpler than their more expensive counterparts, most of these methods still require the camera(s) and screen(s) to be calibrated with respect to each other. We distinguish between *geometric* and *radiometric* calibration. While geometric calibration tries to find the position and orientation of the screen(s) with respect to the camera [27, 2, 10, 12], radiometric calibration establishes the relation between the light emitted by the screen and the light received by the camera [28, 11]. In this work, we will focus on geometric calibration.

If the screen is directly visible to the camera, the calibration process is straightforward: displaying a checkerboard pattern on the screen, standard calibration tools can be used to locate the screen plane [16, 3]. In case the camera and screen are facing a similar directon, a different approach is needed: a mirror is required in order to render the screen visible to the camera [27, 2, 10, 12]. In this work, we will only focus on the latter case, presenting a novel approach. The main contributions of this work are:

1. **Increased accuracy** is achieved due to an increased amount of contraints on our solution.
2. **Less manual interventions** are required, as the optimal number of mirror displacements is minimal.
3. **Robust detection** of screen reflections is achieved by employing a sophisticated masking approach.

2 Related Work

In this section we will provide a brief overview of the existing methods for screen-camera calibration, in the non-trivial case where the screen is not directly visible to the camera. However, before we start the discussion of the different approaches, we need to have a look at catadioptric systems first. Catadioptric systems are combinations

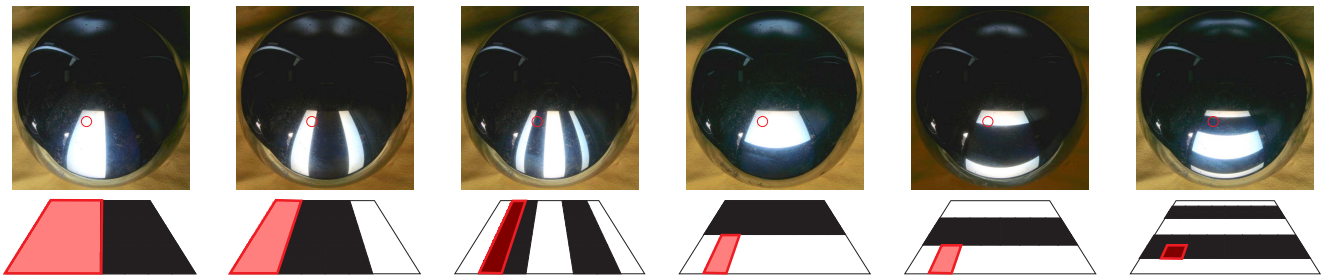

Figure 1. Gray codes uniquely identify matching camera pixels (top row) and screen pixels (bottom row). The patterns are refined step by step, narrowing down the possible regions illuminating the camera pixel. At the finest pattern level, the only remaining possibility will be the reflected pixel.

of reflective (cataoptric) and refractive (dioptric) objects, or as in this paper, a mirror and lens combination which will be used to geometrically calibrate the screen and camera. Depending on the type (plane/sphere/hyperbolic/...) and placement (central/off principal axis) of the mirror and the lens, a single or multi-viewpoint catadioptric camera is created [1, 13]. Notice that in the case of a multi-viewpoint system, it is possible to reconstruct 3D geometry from a single image because of the viewpoints created by the mirror reflections [18, 20]. In this section only two major mirror types will be referred, being a planar and a spherical one. The former produces a single viewpoint system, whereas the latter produces a multi-viewpoint system [1, 13].

Funk and Yang [12] determine the screen's position and orientation with respect to the camera using a planar surface mirror. In order to determine the location of the mirror, a calibration pattern is attached to the mirror. The camera proceeds to capture this calibration pattern, in addition to the reflected image of an additional calibration pattern emitted by the screen. The screen coordinates can then be determined by mirroring back the reflected screen pattern over the planar mirror. Bonfort et al. [2] present an alternative approach using a planar mirror, but instead of using a calibration pattern to identify the mirroring plane, they employ a mirroring hard disk platter with known interior and exterior radii. The projections of these circle boundaries yield two ellipses, which provide sufficient constraints to identify the mirroring plane. Another type of mirror is used by Tarini et al. [27] and Francken et al. [10] who use a spherical mirror. The main advantage of these methods is that no manual distance measurements are required. The only exception to this is the required knowledge of the sphere's radius, but this value is commonly provided by the manufacturer of the mirror. However, these approaches only take reflected screen corners and edges into account, resulting in a reduced accuracy of the results.

In our method, we will also employ a spherical mirror to render the screen visible. However, instead of only taking into account the reflected screen corners and edges, we will let every single screen pixel contribute to the estimated solution. This is achieved by efficiently encoding their positions by employing Gray code patterns, requiring only a small number of input images [26, 9]. In order to encode $M \times N$ screen positions, only $2(log_2 M + log_2 N)$ images are needed. Since we can generate many correspondences from a small number of input images, only two sphere placements are required. Furthermore, the feature detection is very robust, requiring no manual intervention.

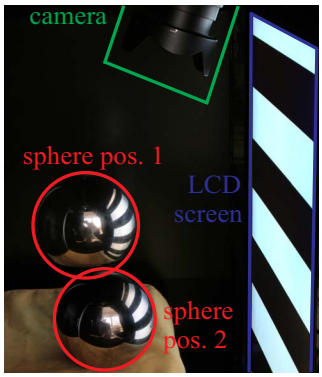

Figure 2. Our setup consists of an LCD screen and a digital still camera. The camera records Gray code patterns reflected off a spherical mirror. This procedure is executed for two different sphere locations.

3 Calibration from Gray Code Reflections

In this section, we will explain our method for screen-camera calibration, based on the analysis of reflected Gray code patterns displayed by the screen. However, before we begin our analysis, it might be useful to elaborate on the input required by our algorithm. Because we are making use of encoded illumination, we need a set of n images for each different spherical mirror. As such, when we refer to a

camera pixel for a sphere i, it is important to remember that we are referring to the n-bit illumination code stored in the set of n images associated with this sphere.

Our processing pipeline consists of three separate modules. First, for each set of images associated with a single sphere position, we estimate the 3D location of the spherical mirror. Second, we establish the bijective function between the camera pixels and the corresponding screen pixels. This relationship is uniquely defined, due to the use of encoded illumination patterns. Finally, we estimate the optimal 3D screen location by minimizing the geometric error between the reprojected 2D screen pixels and the computed reflected ray intersections. We will now discuss each module in more detail.

3.1 Spherical Mirror Detection

The first step of our algorithm consists of the detection of the sphere for each set of images in our dataset. This is done by first performing background subtraction on each set. As the reflected background texture is deformed and has a lower intensity with respect to the real background, standard background subtraction performs well. After cleaning up the resulting mask, we extract the edge pixels using standard image processing operators. This yields an ellipsoidal contour, which is fitted using a tailored RANSAC-based approach [7]. Assuming we know both the internal camera parameters (determined in a preceding preprocessing step) and the radius of the spherical mirror (defining the scale of our solution), we have sufficient data to estimate the sphere's location. This procedure is similar to the one described by Francken et al. [10].

3.2 Screen Pixel Labeling

Now we will explain how Gray codes can provide us with a unique labeling for the screen pixels, as well as how it facilitates distinguishing between screen reflections among other pixels.

3.2.1 Gray codes

As we want to reconstruct the screen's position with respect to the camera, the reflected screen pixels have to be matched to their reflections on the sphere. In order to uniquely identify each individual reflected pixel, we need to encode its 2D position on the screen. Therefore we employ Gray code illumination patterns, as they robustly and efficiently encode ten thousands of screen positions with only tens of images. An example of Gray code illumination is given in Figure 1.

The principle behind Gray code illumination is simple but effective. At each step, we need to establish if the reflected pixel is illuminated by a white or a black region of

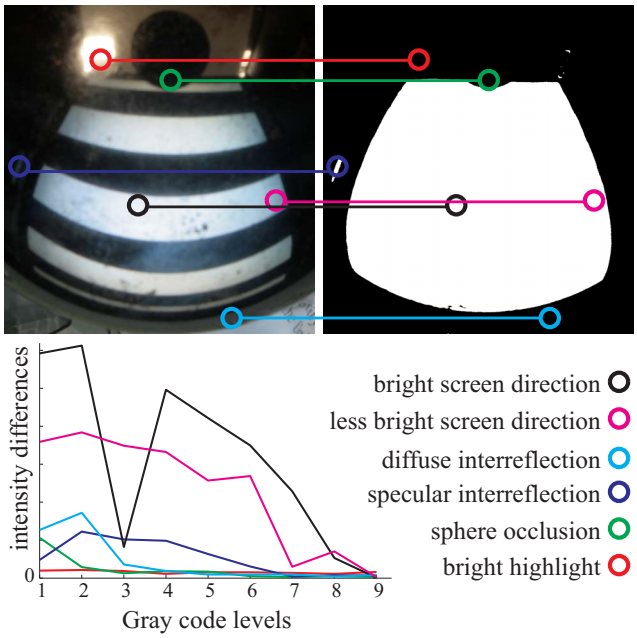

Figure 3. The mask indicates whether or not a pixel is a screen reflection. Top left: input image. Top right: generated mask. Bottom: plot of the intensity differences between scene reflections under normal and complement illumination patterns.

the screen. In order to increase the robustness of this process, we also employ the complementary illumination pattern and compare it to the original. The patterns are refined step by step, narrowing down the possible regions illuminating the pixel of interest. Because of the construction of the patterns, ultimately only one single screen pixel location will remain at the finest pattern level (Figure 1).

Because every pattern doubles the number of white and black stripes, the number of encoded screen positions will double for every extra pattern refinement that is applied. Since we also use the complementary patterns, the exact amount of input images is $2(log_2 M + log_2 N)$ where M and N represent the vertical and horizontal number of locations.

3.2.2 Reflection Mask

Using the Gray code patterns, we will get a code for every single camera pixel. Of course, only the reflections of the screen have to be extracted for use in the next step of our algorithm. A naive method to accomplish this would consist of illuminating the sphere with a complete black and a complete white pattern, subtract the first captured image from the second one, and threshold the result. This way reflections of alternative constant light sources would be filtered out. However, this is not optimal as there may still be

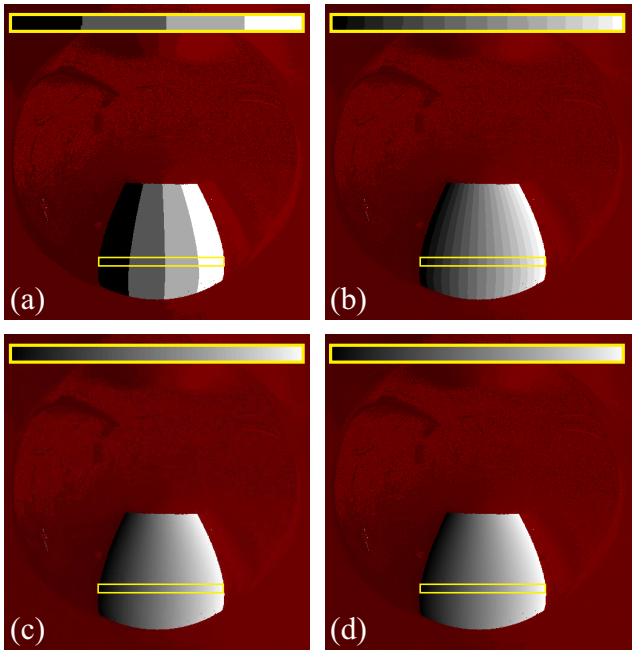

Figure 4. Detected screen positions for different pattern refinement levels, blended with the obtained reflection mask. Decoded horizontal 2D screen positions are expressed relative to the screen dimensions. Number of vertical light patterns used: (a) two, (b) four, (c) six and (d) eight.

other bright reflections caused by high albedo diffusers.

In our method we present a more sophisticated masking technique that selects only sufficiently specular screen reflections. It is inspired by a recently proposed glossiness acquisition method, which employs Gray codes in a similar setup [8]. In order to limit our mask to screen reflections only, we will observe the intensity differences between their complements for each pattern refinement level. As has been shown in the work of Ramamoorthi et al. [25], reflections can be seen as a convolution of the incoming light pattern and the BRDF kernel of the material. Because of this property, the stripe pattern will be blurred away when a certain pattern refinement level has been reached, as the BRDF kernel will be larger than the pattern stripes . Thus, the reflection of a pattern and its complement tend to converge at a certain pattern refinement level, which means that their intensity differences will converge to zero. This is the key idea behind the masking algorithm, which will be combined with the information from the mask already obtained from the background subtraction step.

First, we determine the per pixel intensity differences by subracting the complement pattern from the original one. This is followed by determining the pattern refinement level at which all the absolute values of the intensity differences stay bellow a certain threshold. Intuitively we now have a number indicating the size of the BRDF kernel, where a low number indicates a large kernel (diffuse reflections) and a high number indicates a narrow kernel (specular reflection). Simply thresholding this "glossiness" number gives a robust estimate of the useful screen reflections.

The previous procedure is illustrated in Figure 3. Our method performs well in the presence of alternative constant light sources, high albedo diffusers, and under all viewing angles with respect to the LCD screen. Our method cannot deal with specular interreflections, so we avoid other shiny objects beside the spherical mirror in our setup. In Figure 4 we show a combination of the obtained reflection mask and detected screen positions of the (extended) data set depicted in Figure 1.

3.3 3D Reconstruction

In the final phase of our algorithm, our goal is to estimate the optimal 3D screen location, with respect to the camera. For the remainder of this section, we will assume the position of the sphere to be known, as it can be estimated using the method described by Francken et al. [10].

3.3.1 Reflected Ray Intersections

In the previous phases, we have established a bijective function between each screen pixel x and its observed camera pixel for each set of images associated with sphere position i. If we use the camera's internal parameters to backproject this camera pixel, we can locate its intersection point p_i on sphere i. Once this point is known, we can compute the associated reflected ray $l_i : p_i + t\vec{r_i}$, where $\vec{r_i}$ stands for the reflection direction and t represents a scalar value. Combining the information from all reflected rays associated with a single Gray code / screen pixel, we estimate the 3D position for each individual screen pixel x, minimizing the combined point-line distance $d(x, l^i)$.

$$d(x, l^i) = \frac{\|\vec{r^i} \times (p_i - x)\|}{\|\vec{r^i}\|} \quad (1)$$

$$= \left\| \left(\frac{[\vec{r^i}]_\times}{\|\vec{r^i}\|} \right) x - \left(\frac{[\vec{r^i}]_\times}{\|\vec{r^i}\|} p_i \right) \right\| \quad (2)$$

As can be seen in Equation 2, this is a least-squares minimization of the form $\|Ax - b\|$, which can be solved using the normal equations $(A^T A)x = A^T b$. If $A^T A$ is invertible, $x = (A^T A)^{-1} A^T b$ provides us with the solution.

3.3.2 Plane Estimation

Once we have a point cloud of estimated 3D screen pixel locations, the next step is to enforce the constraint that all

pixels are part of the same planar illuminant. As such, we need to estimate the 3D plane $\pi = (a, b, c, d)^T$ which minimizes the point-plane distance $d(\pi, x) = (\pi \cdot x)/|\pi|$. We initiate our estimation with a RANSAC search for a good intial parameter guess, followed by Levenberg-Marquardt minimization with distance measure $d(\pi, x)$.

3.3.3 Grid Estimation

Once we have a rough estimate of the plane on which the screen should be located, the next step consists of finding the exact 3D coordinates for each screen pixel. In this final step of the 3D reconstruction, we use the final contraint for our estimation problem: the pixels are part of a rigid grid structure. As such, there exists a 2D-to-3D similarity transformation M, which maps each 2D screen pixel location u as close as possible to its 3D counterpart x. Assuming we can estimate this optimal M for each given plane π, this gives rise to a second Levenberg-Marquardt plane estimation routine, minimizing the point-point distance $d(Mu, x)$.

Similarity Transformation M In order to guarantee good numerical stability for the estimation of M, we have to perform numerical pre-conditioning on both point sets [15]. As we want to apply the same normalization procedure on both $\{u\}$ and $\{x\}$, we compute the similarity transformation in a 2D reference frame. For this purpose, we first transfer the 3D plane π to the xy-plane using transformation T_{3D}.

$$T_{3D} = \begin{bmatrix} R_3 & -R_3\bar{x} \\ 0^T & 1 \end{bmatrix} \tag{3}$$

In this equation, $\bar{x}$ is the mean vector of the 3D point cloud, and R_3 rotates plane normal n_π to plane normal n_{xy}. Note that the transformation moves the mean $\bar{x}$ to the origin, facilitating subsequent normalization procedures. We follow up this transformation by a projection onto xy-plane coordinates.

$$P_{xy} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

Finally, after this transformation, we rescale the resulting point set $\{x''\}$ to map the average distance $\overline{\|x''\|}$ to $\sqrt{2}$.

$$S_{3D} = \begin{bmatrix} \sqrt{2}/\overline{\|x''\|} & 0 & 0 \\ 0 & \sqrt{2}/\overline{\|x''\|} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

In a similar fashion, we need to normalize the 2D screen pixel set. First we need to convert the screen resolution (eg. 1280×1024 pixels) into a physical resolution (eg. 474×297 mm).

$$S_{res} = \begin{bmatrix} w_{phys}/w_{res} & 0 & 0 \\ 0 & h_{phys}/h_{res} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

After this aspect ratio correction, we relocate the new point set's mean $\overline{u'}$ to the origin.

$$T_{2D} = \begin{bmatrix} 1 & 0 & -\overline{u'}_x \\ 0 & 1 & -\overline{u'}_y \\ 0 & 0 & 1 \end{bmatrix} \tag{7}$$

As finally, we normalize the 2D $\{u''\}$ coordinates, in a similar fashion to the normalization we performed on the 3D $\{x''\}$ coordinates.

$$S_{2D} = \begin{bmatrix} \sqrt{2}/\overline{\|u''\|} & 0 & 0 \\ 0 & \sqrt{2}/\overline{\|u''\|} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8}$$

After we have transformed both $\{x\}$ and $\{u\}$ to a common reference frame, we can employ standard techniques to estimate the 2D-to-2D similarity transformation H that minimizes the geometric error $d(Hu''', x'')$. After this minimization, we can concatenate all matrices to find our final 2D-to-3D similarity transformation.

$$M = (S_{3D}P_{xy}T_{3D})^{-1}HS_{2D}T_{2D}S_{res} \tag{9}$$

As such, all the obtained calibration information is compressed in a single matrix M, allowing for the direct transformation of 2D screen pixels into 3D locations by a simple matrix multiplication: $x = Mu$.

4 Results

In this section we will discuss the results produced by our technique. Initially, we will determine the relation between the number of pattern refinements and the error of the produced calibration. Afterwards, we will analyze the influence of the sphere location on the quality of the results. Results from both real-world and virtual data sets are provided, and are compared to previous work.

In order to determine the correctness of the estimated screen positions, we perform a ground truth evaluation. Our ground truth constitutes a virtual set of images created with the POV-Ray [24] ray tracer. The accuracy of the produced solution is is measured by the sum of squared distances between the estimated screen corners and their corresponding ground truth values.

4.1 Pattern Refinements

The accuracy of our method is largely dependent on the number of estimated ray-ray intersections. As the number of pattern refinements is directly related to the number of possible intersection points, an increase in pattern refinements will result in a higher accuracy. This is illustrated in Figure 5, where we have plotted the error in function of

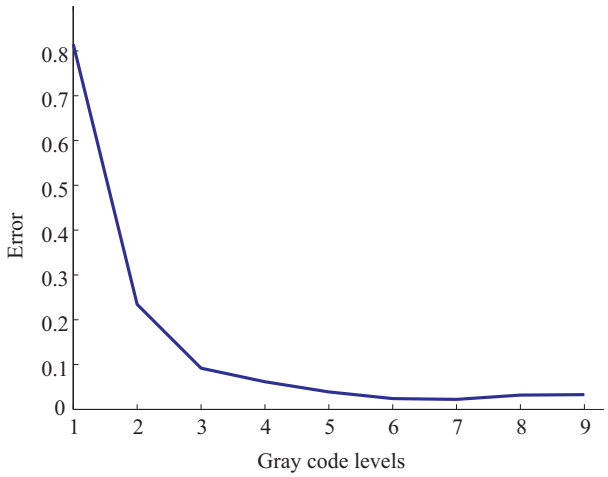

Figure 5. Error in function of the maximum applied pattern refinement level.

the pattern refinement level. At a certain point in the function, the extra refinements will no longer yield any improvements. On the contrary, the results start to deteriorate as the least significant bits of the detected code are no longer detected, lowering the chance to find exactly this code in other images.

4.2 Sphere Placement

The quality of our results not solely depends on the chosen pattern refinement level, but also on the the number of sphere displacements as well as the positioning of the sphere. From our experiments, we have concluded that only two sphere locations are required, as this case produces the lowest error values. In contrast to previous methods, the error does not decrease and converge when adding extra, less optimal, sphere locations. As can be seen in Table 1, where we compare the error values of two sphere locations in terms of their distance to the camera, the sphere locations closest to the camera produce the best results. Additionally, in order to maximize the accuracy of the sphere location detection algorithm, we place the center of the sphere on the principal axis of the camera. Finally, a minimum distance between two different locations is preserved, avoiding degenerate configurations.

A side-by-side comparison with previous methods employing a spherical mirror and our method is also performed. As it turns out, our error is approximately 7 to 8 times lower compared to the most recent method [10], and we only require 2 different sphere locations instead of 6/7.

Since no ground truth is available for our real-world setup, we have measured the vertical and horizontal edges of the LCD screen. In order to verify the accuracy of our method for real-world data, our error metric consist of a simple comparison of the measured screen edge lengths and the distances between the estimated screen corners. For a screen of 474 $\times$ 297 (measured in millimeters), we compute dimensions of 478.7 $\times$ 300.4, suggesting an accuracy up to a few millimeters. The method of Francken et al. [10] produces dimensions of 449.8 $\times$ 286.7, suggesting an accuracy up to a few centimeters.

5 Conclusions

In this paper, we have presented a novel method for performing screen-camera calibration, using a spherical mirror and screen pixel encoding using Gray code patterns. Because it is based on robust feature detection, and geometric error minimization, we have provided a solid solution to the calibration problem of these setups. We have shown our method is more accurate, requires a reduced number of manual interventions, and detects screen reflections more robustly than previous methods.

6 Future Work

As cameras are often focussed on the scene in front of the setup, and not on the reflected screen, we currently need to refocus the camera. This is not desirable since we then alter the internal camera parameters implying a recalibration. Therefore, we are now looking for alternative pattern sequences which are more insensitive to blur due to an out of focus camera.

In the near future, we will also compare the use of Gray code patterns to other codification methods in order to limit the number of required recordings. In concreto, we expect the use of gradient patterns [21] can reduce the number of input images from 30-35 to less than 10, reducing the calibration time even more.

Acknowledgements

Part of the research at EDM is funded by the ERDF (European Regional Development Fund) and the Flemish government. Furthermore we would like to thank our colleagues for their help and inspiration.

References

[1] S. Baker and S. Nayar. A theory of single-viewpoint catadioptric image formation. *IJCV*, 35(2):175–196, Nov 1999.

[2] T. Bonfort, P. Sturm, and P. Gargallo. General specular surface triangulation. In *ACCV*, volume 2, pages 872–881, Jan 2006.

[3] J.-Y. Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, 2006.

Far														0.901
												0.856	0.986	0.860
											0.704	0.704		0.826
												0.729		0.622
									0.374	0.487		0.557		
								0.435	0.397	0.470	0.495	0.551		
							0.317	0.346	0.404	0.434		0.469	0.545	0.668
						0.306	0.320		0.342	0.384	0.502	0.503	0.484	0.549
					0.245	0.246	0.278	0.306	0.374	0.369	0.434	0.467	0.555	0.494
				0.189		0.232		0.301	0.278		0.372	0.429	0.461	
↑			0.156	0.170	0.179	0.223	0.240	0.255	0.304	0.321	0.340	0.370	0.427	
		0.107	0.123	0.138	0.162	0.178	0.194	0.216	0.241	0.274	0.274		0.360	0.366
Near	0.075	0.085	0.099	0.111	0.130	0.138	0.163	0.178	0.202	0.217	0.224	0.264	0.341	0.279
	Near	→												Far

Table 1. Error for different combinations of 2 out of 14 sphere locations. The spheres all lie on the principal axis on evenly spaced distances from the camera. Spheres near the camera yield the lowest error. Missing values are due to a low number of Gray code correspondences between both image sets, resulting in an insufficient number of estimated 3D screen pixels.

[4] Y. Chen, C. Su, J. Chen, C. Chen, Y. Hung, and C. Fuh. Video-based eye tracking for autostereoscopic displays. *Optical Engineering*, 40:2726, 2001.

[5] J. J. Clark. Photometric stereo with nearby planar distributed illuminants. In *CRV*, page 16, 2006.

[6] M. Dumont, S. Maesen, S. Rogmans, and P. Bekaert. A prototype for practical eye-gaze corrected video chat on graphics hardware. In *International Conference on Signal Processing and Multimedia Applications*, pages 236–243, Porto, Portugal, Jul 2008.

[7] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM archive*, 24:381 – 395, 1981.

[8] Y. Francken, T. Cuypers, T. Mertens, and P. Bekaert. Gloss and normal map acquisition using gray codes. In *SIGGRAPH ASIA sketches*, New York, NY, USA, 2008. ACM.

[9] Y. Francken, T. Cuypers, T. Mertens, J. Gielis, and P. Bekaert. High quality mesostructure acquisition using specularities. In *CVPR*, pages 1–7. IEEE, 2008.

[10] Y. Francken, C. Hermans, and P. Bekaert. Screen-camera calibration using a spherical mirror. In *CRV*, pages 11–20, 2007.

[11] Y. Francken, C. Hermans, T. Cuypers, and P. Bekaert. Fast normal map acquisition using an lcd screen emitting gradient patterns. In *CRV*, pages 189–195. IEEE, May 2008.

[12] N. Funk and Y.-H. Yang. Using a raster display for photometric stereo. In *CRV*, pages 201–207, 2007.

[13] C. Geyer and K. Daniilidis. Catadioptric projective geometry. *IJCV*, 45(3):223–243, 2001.

[14] D. Gorodnichy, S. Malik, and G. Roth. Nouse Use your nose as a mouse-a new technology for hands-free games and interfaces. *Vision Interface*, pages 354–361, 2002.

[15] R. I. Hartley. In defense of the eight-point algorithm. *TPAMI*, 19(6):580–593, 1997.

[16] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[17] Z. Hatab, J. McNeil, and S. Naqvi. Sixteen-megabit dynamic random access memory trench depth characterization using two-dimensional diffraction analysis. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures*, 13:174, 1995.

[18] S. Kuthirummal and S. K. Nayar. Multiview radial catadioptric imaging for scene capture. *SIGGRAPH*, Jul 2006.

[19] K. N. Kutulakos and E. Steger. A theory of refractive and specular 3d shape by light-path triangulation. *IJCV*, 76(1):13–29, 2008.

[20] D. Lanman, M. Wachs, G. Taubin, and F. Cukierman. Reconstructing a 3d line from a single catadioptric image. In *3DPVT*, pages 89–96, Washington, DC, USA, 2006. IEEE.

[21] W.-C. Ma, T. Hawkins, P. Peers, C.-F. Chabert, M. Weiss, and P. Debevec. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *EGSR*, Jun 2007.

[22] N. J. W. Morris and K. N. Kutulakos. Reconstructing the surface of inhomogeneous transparent scenes by scatter-trace photography. In *ICCV*, 2007.

[23] D. Nehab, T. Weyrich, and S. Rusinkiewicz. Dense 3D reconstruction from specularity consistency. In *CVPR*. IEEE, Jun 2008.

[24] POV-Ray. `http://www.povray.org/`, 2009.

[25] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH*, pages 117–128, New York, NY, USA, 2001. ACM.

[26] J. Salvi, J. Pages, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827–849, April 2004.

[27] M. Tarini, H. P. A. Lensch, M. Goesele, and H.-P. Seidel. 3d acquisition of mirroring objects. *Graphical Models*, 67(4):233–259, Jul 2005.

[28] B. Wilburn, H. Xu, and Y. Matsushita. Radiometric calibration using temporal irradiance mixtures. In *CVPR*, pages 1–7, 2008.

[29] D. E. Zongker, D. M. Werner, B. Curless, and D. H. Salesin. Environment matting and compositing. In *SIGGRAPH*, pages 205–214, 1999.