Stripe Propagation for Color Encoded Structured Light

M. Burisch D. Guerrero Ichaso A. Kuijper
{michael.burisch, david.guerrero, arjan.kuijper}@igd.fraunhofer.de
Fraunhofer Institute for Computer Graphics Research IGD, Germany

Abstract

We present a method to improve surface reconstruction using color encoded structured light. A video camera is used to obtain a color encoded pattern projected onto the object using a projector. A multi-stage method is presented to reconstruct the surface from the captured pattern. It consists of i) a robust edge detection step, ii) a color decoding using feedback from previous stripes and iii) a propagation step to detect errors and propagate detected stripes. Using feedback and propagation we can reconstruct areas where regular color detection fails by bridging small gaps which happen due to illumination variations or noise. The reconstruction results are illustrated for human skin.

1. Introduction

Fast 3D reconstruction of surfaces from 2D views using structured light projection is an active field research as it offers the ability to capture moving objects. Typical applications include biometrics, especially face scanning, and medical applications, like breath tracking or anatomic motion analysis. Opposite to classical scanning these applications require a low scanning time to avoid motion artifacts during reconstruction.

To capture motion in 3D either a very fast sequence or a low number of patterns have to be used. A fast sequence usually utilizes three images with sinusoidal patterns, which are projected at a high frequency. By combining the captured images, the ambient and diffuse parts of the images can be removed and the phase values can be calculated for every pixel. These allow for a high spatial resolution and thus a very dense 3D reconstruction. Due to the fact that multiple images are required, those approaches suffer from fast motion, which introduces motion artifacts.

The optimal solution in terms of motion artifacts is a single image 3D reconstruction, which can be achieved by using color encoded structured light. The most widely used patterns comprise parallel lines with different colors. In contrast to phase-shifting, the decoding process is spatial, meaning that a unique position within the pattern can only be established by analyzing a local neighborhood for every position. Furthermore, ambient and diffuse distortion as well as wavelength dependent reflection and absorption are an important factor during color detection. For a successful color detection, the stripes also need a certain width. This has two important consequences: Small details in geometry cannot easily be recovered and the spatial resolution and thus the 3D reconstruction is much lower as compared to the phase-shifting approach explained above. Note, that although the spatial resolution is lower, depth resolution can still be high using sub pixel accurate detection for stripe positions.

1.1. Related Work

An overview of different patterns and their applications is given in Salvi et al. [13]. Zhang and Huang [17, 18] proposed a method for real-time reconstruction based on three rapidly projected patterns. To be robust against noise the waves are short with respect to projector resolution. Therefore, the pattern is not unique but repeats itself a number of times, which requires additional phase-unwrapping. In their initial approach they assumed a continuous surface, which was later relaxed by Zhang [16]. Weise et al. [14] proposed a similar system with dedicated support for phase-unwrapping and motion artifacts, by using methods from stereo vision and motion estimation.

Color encoded structured light systems usually use patterns containing parallel lines with different colors, although other patterns are also possible [11, 8]. Parallel lines contain all the information necessary for surface recovery. Other patterns like the grid from Chen et al. [2] add additional robustness to noise at the cost of a much lower spatial resolution. Most patterns are created as de Bruijn sequences [3]. In these patterns a sequence of n different colors uniquely identifies the position. Such systems are described for instance by Forster [6] or Fechteler et al. [5]. For a robust color detection in the presence of surface reflectance and absorption Zhang et al. [15] use dynamic programming to find an optimal stripe assignment. Fechteler and Eisert [4] introduce a probability measure for color assignment using k-means clustering of the colors. A differ-

ent approach to cope with surface properties is the use of scene adaptive patterns as suggested by Koninckx et al. [9]. However, this affects only the following images and cannot compensate problems with small bridges. For sub-pixel accurate detections of the stripe positions, the methods proposed in [4] and [10] can be used.

1.2. This work

In this paper we propose a 3D scanning system using color encoded structured light, which is able to reconstruct a surface *using only one pair of images from two cameras*. Our work is not focused on reconstruction itself, but on exploiting the captured data and reconstructing as many points as possible. Using feedback from neighboring stripes and propagation along the stripe direction during color detection and sequence decoding, the system is able to recover areas where regular approaches fail and detect false classifications. These steps are needed, as our work is motivated in capturing human skin, which has challenging surface properties with respect to reflection and absorption. First we describe our setup, then we explain our reconstruction method consisting of pattern selection, edge detection, color decoding and stripe propagation. Afterwards we evaluate the algorithm and finish with a conclusion.

2. Method

Our setup is composed of two color cameras with a Bayer filter and a DLP projector as illustrated in Fig. 1. The cameras are aligned to capture objects at a distance of 1.5m in front of them with a baseline of approximately 30cm. The projector is located between both cameras and projects the pattern onto the object. The cameras are assumed to be calibrated, thus their intrinsic and extrinsic parameters are known. As we use two cameras for reconstruction, no calibration of the projector is required. To retrieve a 3D model, a pattern is projected onto the object. The depth information is then encoded into the deformation of the observed stripes. To calculate the depth value, a correspondence between the captured and projected stripe has to be established. Having a stripe number for every captured position, the 3D points can be reconstructed by triangulation between the two cameras. To acquire the stripe number, at least five consecutive stripes have to be decoded. This is not always possible, as there might be small details in the geometry or color miss classifications. To detect such errors and reconstruct small parts, feedback and propagation are used during reconstruction. In our method we use three steps: 1) edge detection, 2) stripe decoding, and 3) stripe propagation. In the following we describe these steps in more detail, starting with the description of the selected pattern.

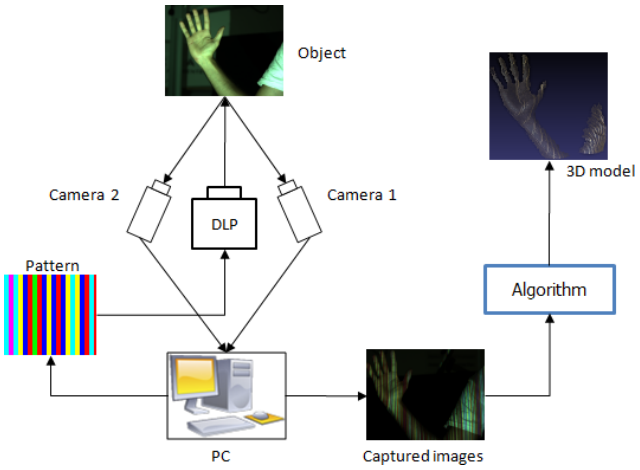


Figure 1: System setup: Two color cameras and a projector.

2.1. Pattern

The projected pattern should allow a robust detection. Therefore, a pattern consisting of vertical stripes with binary enabled colors (a channel is either on or off) is used. However, as the projector only provides three basic colors (red, green, and blue) there can only be eight different colors (red, green, blue, cyan, magenta, yellow, black, and white) fulfilling this requirement. To reconstruct discontinuous surfaces a de Bruijn sequence [3] consisting of six colors is used. The uniqueness window contains five stripes, meaning that within the sequence every five consecutive stripes can be labeled with a unique sequence number. This sets the minimum reconstructible object size. The colors used are red, green, blue, cyan, magenta, and yellow. Neither black nor white is used, as they exacerbate stripe detection. This can be seen in Fig. 2, in which the pattern is projected on a white flat board, which is one of the best scenarios possible. However, it can be seen that the intensity in each channel for white color is significantly higher than for other colors. Due to those intensity differences this does not only affect techniques like automatic contrast stretching, but has effect on neighboring stripes as well. This is visualized in Fig. 3, where the pattern is projected onto human skin. A white stripe *boosts* the intensity of all color channels. In neighboring stripes this effect has vanished, and so, even though a color channel stays enabled, the captured intensity is noticeably lower. Furthermore, this effect is also wavelength dependent and as a result the color with the highest measured intensity is not necessarily projected as such. Black stripes are not used to avoid false positives in unlighted areas. For best results during edge detection, the additional requirement of at least two changing color channels between consecutive stripes, as suggested in [4], is added. This additional constraint causes improved edge

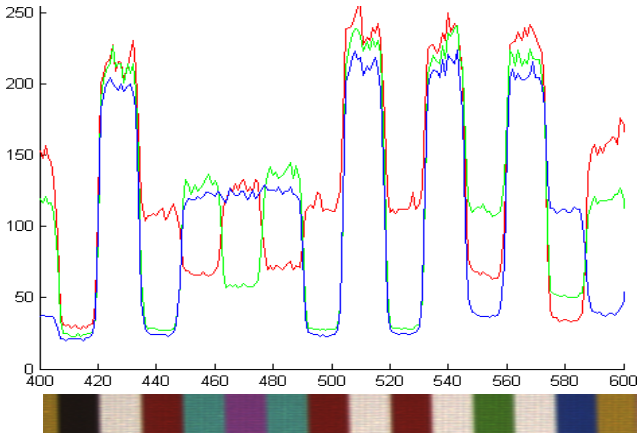


Figure 2: Above: RGB signal vs. pixel position. Below: The captured image, projected onto a white board. The color at white stripes has a much higher intensity than in colored stripes.

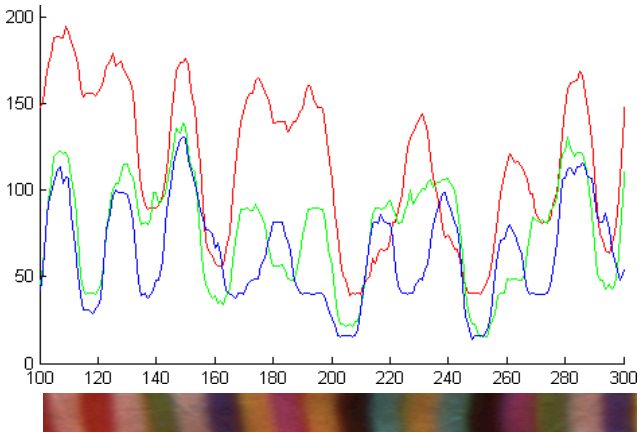


Figure 3: Above: RGB signal vs. pixel position. Below: The captured image, projected onto a hand. Notice for instance the signal at $x = 140$ or $x = 160$, where a color detection easily fails due to influence from the neighboring white stripes.

detection, especially in scenes with more challenging reflectance, such as human skin.

2.2. Edge Detection

The captured images are first filtered with a 3×3 low pass filter to reduce noise. This will also smooth the stripes, reducing the number of false edges detected. As a downside, some stripe edges will be blurred too much and thus not be detected. However, the balance between these cases and the effective reduction of noise results in a better overall performance in our experiences. To avoid interference from the background, a simple mask is applied during the edge detection. This mask is computed from the difference between a background image and the captured image. After applying the filter and the mask, the image contrast is enhanced by stretching each channel's histogram so that 1% of the data is saturated.

Once the images have been enhanced, edges have to be detected. Simple approaches using first- and second-order derivatives could be implemented, but both methods are highly sensitive to noise. To achieve good results in the presence of noise, a Canny edge detection filter [1] is used. The filter is applied to every channel of the RGB separately. This will result in three binary images which have connected edges detected by the filter. Only edges that belong to the projected stripes are of interest. As they are assumed to be roughly vertical in the image, horizontal edges should be eliminated, since they will mainly be due to noise. Such edges typically result from small geometrical features of the object. To eliminate edges in the undesired direction, a morphological opening on the binary images using a 1×3 vertical mask is performed. This mask eliminates pixels which are perpendicular to the assumed stripe direction. Unfortunately, this has the side effect of also erasing stripe edges excessing a certain angle due to surface deformation. This is not undesired, though: the reconstruction quality decreases as the angle between camera ray and surface normal increases. So removing those stripes also prevents low quality points to a certain degree. After performing the opening on the image, still some traces of noise are left. For further reduction, very small connected segments (in the order of a few pixels) are also erased. The detected edges in each channel are now to be merged, by creating an image E, where the three binary detected edge images E_r, E_g and E_b are summed up:

$$E_r, E_g, E_b \in \{0,1\}^{width \times height} \tag{1}$$

$$E_{sum} = E_r + E_g + E_b \tag{2}$$

It is easy to see that the resulting image gives a weighted edge detection. However, there will be many duplicate entries for the same edge. In order to eliminate those and form a final edge image E, the neighbors of all pixels in E_{sum} are checked in horizontal direction up to a distant equivalent to half the *width of the stripes*. The expected width of the stripes in pixel s can be estimated from the optical properties of the projector and the cameras. The projector has a resolution x_{res} and projects an image at distance z_0 with a width of w_0. Using Eqs. (3) and (4) the expected width s of a n-pixel wide stripe in the camera with focus f_u at distance z to the object can be calculated as

$$w(z) \approx n \cdot \frac{z}{x_{res}} \cdot \frac{w_0}{z_0}, \tag{3}$$

$$s \approx f_u \cdot \frac{w(z)}{z}. \tag{4}$$

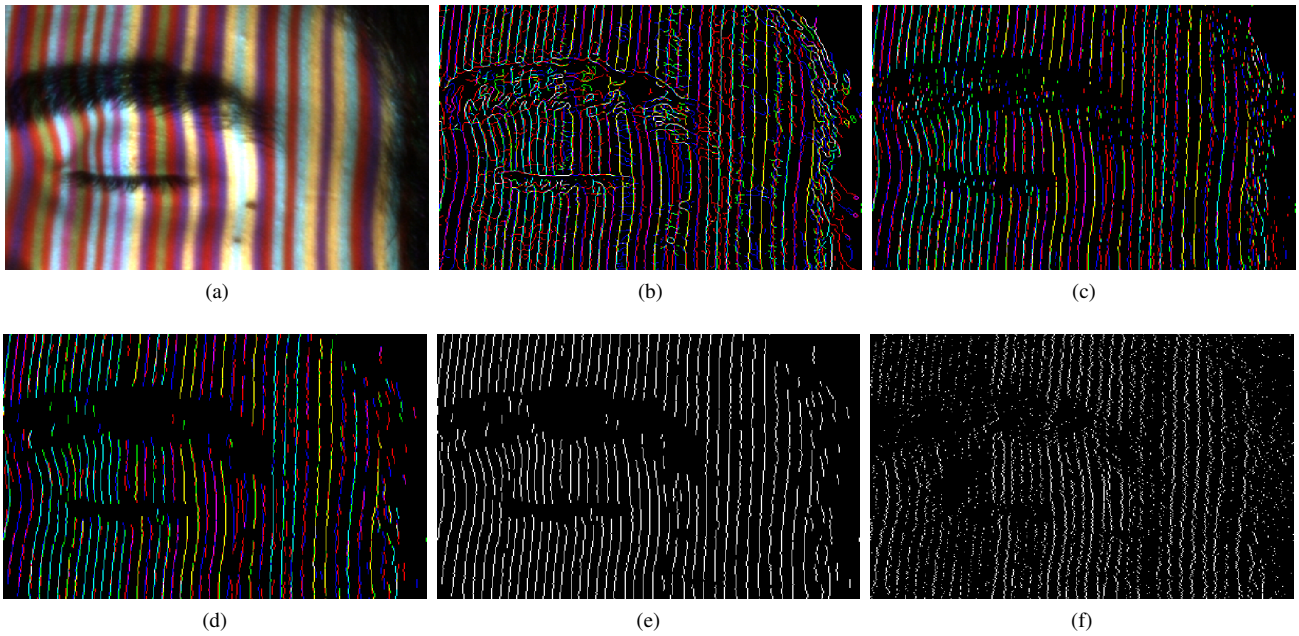


Figure 4: (a) Captured image. (b) Edges detected after applying a Canny filter to each color channel. (c) Edges left after performing opening in each channel. (d) Edges left after erasing small segments. (e) Final edge image. (f) Edge detection using second-order derivative.

If there is any pixel in the interval with a value of two or three, its position is marked in E. If all of them have a value of one, the mean position is calculated and the respective pixel set to one. In the end we obtain a final edge image with weighted values depending on how reliable the edges are. The results of these operations are shown in Fig. 4.

2.3. Stripe Decoding

After the edge detection, the stripe color and sequence number have to be decoded. The first step is to find *decision points* where we want to analyze the image. Theoretically, these points should be in the middle of every projected stripe. Using the edge image E we add decision points in between two detected edges. In the next steps, these points will be used to detect the color and decode the pattern, checking to which stripe they belong and assigning them their unique sequence number. The coordinates of these points is the information we need to reconstruct a 3D model. Care has to be taken for edges in E, that do not belong to stripe edges. To account for this, the distance between two consecutive edges has to be the expected stripe width with a 50% margin of tolerance. Moreover, the last edge distance is stored and taken into account, so if the distance between edges slowly increases or decreases in small steps, they are also considered as correct. If the distance between consecutive decision points is outside the margins, it is marked. Those jumps happen as a consequence of occlusions or discontinuities and are not considered in further processing. Although some valid points will be removed, this also helps to assure a high quality reconstruction. Stripes on an orthogonal plane with respect to the camera view will all be valid. Only areas with a large angle between the normal and the camera ray will be rejected, as they lead to large or small stripes.

The pattern has a uniqueness window of five stripes. This means that once five stripes are decoded correctly, the position of each stripe can be found in the original pattern. As the projected pattern is known, there is an expectation about the next color. This information is used during the color detection process. Elaborating on this knowledge, two different methods are applied: a *ratio* and a *threshold* approach. The ratio method is used when there is no information about the expected color of the stripe, for instance after a discontinuity. In this method the RGB values of the pixel are compared as described in [5]. A vector with the ratios $\frac{r}{g}$, $\frac{r}{b}$, $\frac{g}{r}$, $\frac{g}{b}$, $\frac{b}{r}$, $\frac{b}{g}$ is calculated. The two biggest ratios are selected and the respective colors are assumed to be enabled. If for instance $\frac{r}{b}$ and $\frac{g}{b}$ are the two biggest ratio, the resulting color is (1, 1, 0). This is the primary method, as it is robust with respect to brightness and illumination changes. However, in dark images this methods can fail during selection of the second biggest ratio. It is important to note that it is impossible to use a method that will work under any situation and, because of that, there will always be er-

rors. Our experiences confirm that this method is reliable in most circumstances. After decoding five sequent colors, we use the knowledge of the next expected color and verify the detection. There are two possibilities in case of a difference between them: false detection or false expectation, for instance due to discontinuities in the surface.

Using this feedback it is also possible to correct errors. If an error is detected, the second detection approach based on a local threshold for every channel is used. The threshold used is the arithmetic mean of each channel's signal in a window of the expected width of the stripes. This method has been compared to the results from using Otsu's method [12] for threshold estimation. There, the intra-class variance of two classes (in our case the color channels) is minimized. However, the latter method is much slower and the improvement only marginal and so the first method is preferred. The RGB value of the decision point is compared to the thresholds and the color channels enabled or disabled accordingly. Since this method is generally used in bad conditions (the first one already failed), there is also a margin of error for the threshold. The margin is set as 10% of the maximum value that the RGB signal assumes in the window:

$$r_{max} = \max_{x=-\frac{s}{2}..\frac{s}{2}} red(x) \tag{5}$$

$$th_{margin} = 0.1 \cdot \max(r_{max}, g_{max}, b_{max}) \tag{6}$$

This comes into play if the expected and the decoded color still differ. Each channel is compared to the margin and flipped when necessary:

$$v_{red_{new}} = \begin{cases} v_{red} & |v - th_{mean_{red}}| > th_{margin} \\ 1 - v_{red} & else \end{cases} \tag{7}$$

To illustrate this, Fig. 5 shows a point x_d where the *ratio* method fails. In the example, the actual color of the stripe is green, that is, (0,1,0). The RGB values of the pixel are (61, 69, 45) and so the ratios are (0.88, 1.36, 1.13, 1.53, 0.74, 0.65). Since the two biggest ratios are $\frac{r}{b}$ and $\frac{g}{b}$ this will be detected as yellow (1,1,0). Using the feedback from previous steps, green is expected and thus we apply the second method. As illustrated in Fig. 5, the thresholds for each channel are (74.78, 57.56, 53.56) with an error margin of 6.9. When comparing the RGB values with this thresholds, the resulting color is green, (0,1,0) which is the expected color.

Experiments using the *threshold* method first and falling back to the *ratios* method have been performed as well. However, practical results contradict this approach; using the methods in the former order provides better stripe decoding. This supports the presumption that the *ratios* method is more robust against brightness issues. If both methods fail to detect the expected color, decoding is restarted at the current stripe.

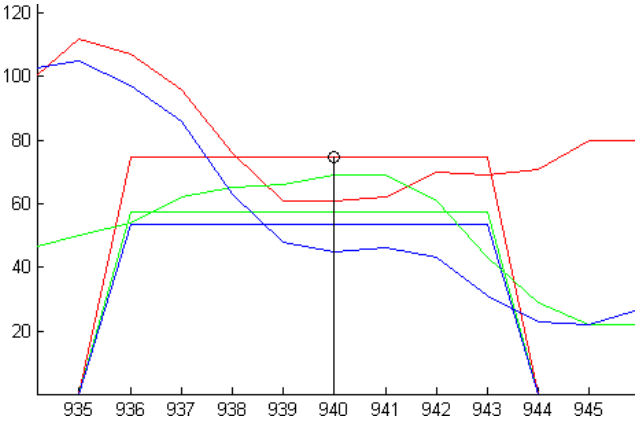


Figure 5: RGB signal and corresponding thresholds for decision point $x_d = 940$.

2.4. Stripe Propagation

When all positions have been decoded, the data is theoretically ready for 3D reconstruction. However, the quality of the data can be highly improved. For this purpose we have developed a propagation method, in which additionally errors can be detected and corrected, gaps can be filled and lines can be expanded to areas where detection was impossible before.

This fixing step approaches the lines from a different point of view. Up until now, stripe detection was done on a line-by-line basis. However, by design we know that stripes should be composed of connected pixels. Consequently, we should recover complete stripes by using an *oriented region growing* approach. Stripes are traced along their direction to check that there are no false detections and, when possible, the stripes are extended and gaps are filled.

This strategy provides a much higher level of error detection since each stripe is not just considered in a single point anymore. When tracing a stripe, two scenarios can occur: i) neighboring pixels have a different sequence number, or ii) the end of a stripe is reached due to undetectable colors or gaps in the geometry.

In the first scenario the sequence number between two connected points is different. Then either the current or the previous pixel is incorrectly assigned, or the object is discontinuous at this position. In both cases there are two sets of differently labeled points. To ensure that the most likely one is chosen, the following rules are applied:

1. Compare length of both sequences. If one is short and the other much longer (say a factor of three), the latter is assumed to be correct and the former to be wrong.

2. Compare the underlying color of both sequences. If they correspond to the same color, the estimated pixel position in the image that corresponds to that sequence

number according to an average stripe width is estimated. The one closest to the current position is assumed to be correct.

3. In case both colors are different, they are compared to see which one is more likely to be correct for the current pixel by searching the one with a minimum *distance* to the pixel values. The distance between the current pixel $\vec{p} \in [0, 255]^3$ and an expected color $\vec{c} \in \{0, 1\}^3$ is calculated according to Eq. (11):

$$p_{min} = \min(p_r, p_g, p_b) \tag{8}$$

$$p_{max} = \max(p_r, p_g, p_b) \tag{9}$$

$$\vec{v} = p_{min} \cdot (1 - \vec{c}) + p_{max} \cdot \vec{c} \tag{10}$$

$$distance(\vec{p}, \vec{c}) = \frac{1}{3} \sum_{i \in \{r,g,b\}} |v_i - p_i| \tag{11}$$

In the second scenario the end of the decoded stripe is reached. In order to extend it, the edges are needed. These are known from the previous step, although sometimes edges are broken. If just one pixel is missing, the edge from the next position is used. If both edges are found, a decision points is added in the middle as shown in Fig. 6. Once the position is set, the color is decoded. Since the expected color and its sequence number are known, the process is the same as in the previous section: first run the *ratio* method and if it fails the *threshold* method. If the color can be decoded correctly, the pixel is added and the stripe extended. As stripes should be continuous, the new point has to be connected to the previous one. If this is not possible, gap filling starts. During gap filling the same sequence number is searched for along the direction of the stripe. The search range is determined as a fraction of the number of pixels the stripe already has. The idea is to avoid connecting stripes which are not connected, but separated by a discontinuity. Therefore, in case the stripe is very long, only a maximum number of pixels is to be checked. In case a pixel with a different sequence number is found, no gap filling is performed. However, if the gap is small, the current stripe has a minimum length, and the pixel belongs to a very short sequence, then the gap is still filled and the sequence ignored. Whenever the same sequence number is found after a gap, both coordinates get connected by a line between both ends. As it cannot be assumed that every point belongs to a stripe and not to an occlusion, the regular color detection methods for every point on the line apply. An example for the detection and propagation of stripes is shown in Fig. 7. In case the stripe cannot be extended, the current stripe is assumed to end there and we start with the next one.

3. Evaluation

Tests have been carried out to evaluate the performance of our new approach. Point reconstruction is done by tri-

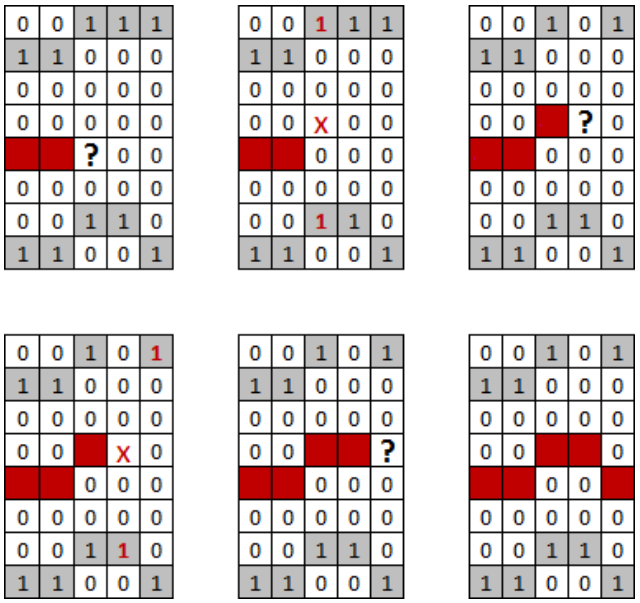
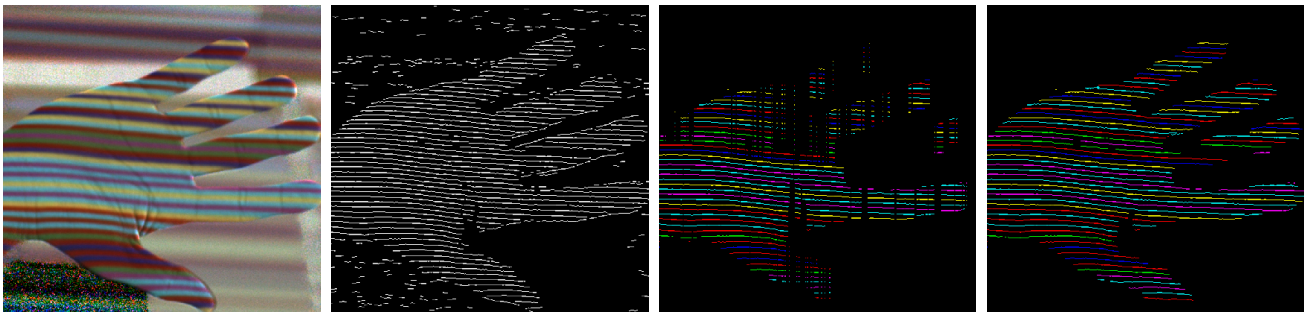


Figure 6: A step by step example of the stripe growing method. For every column to grow, the edges (marked as 1) are found and the middle point added as a new decision point. If it is valid (color and connectivity), the current sequence number is assigned.

angulation between the two cameras. For every detected stripe point in one camera image the best match in the second camera is found. Using the intrinsic and extrinsic camera parameters 3D positions are calculated using the linear triangulation method described in [7]. As stated before, the focus of this work is on exploiting the captured data and reconstructing as many points as possible, and not on the reconstruction itself.

To evaluate the performance of the approach different body parts (face, hand, back and front torso) have been scanned and the number of reconstructed points between a direct reconstruction and the propagation method were compared. A number of tests have been carried out for different objects poses, giving similar results for the same type of object. For large flat objects only small improvements in the range of 3% to 6% can be achieved. However for more challenging objects like hands or the face improvements of around 20% to 30% are possible. Some typical results are given in Table 1. As can be seen in Fig. 8a, most scans will have certain parts which cannot be reconstructed correctly. Reasons for this include small bridges in geometry, illumination variations along the object's surface or noise, which can cause false color detections. Depending on the object's complexity and surface properties a significant increase in reconstructible points can be achieved (Fig. 8e). Aside from this, the method also reduces falsely classified sequence numbers by reassigning them. It is interesting to



(a) (b) (c) (d)

Figure 7: (a) Captured image after preprocessing. (b) Detected edges. (c) Decoded stripe positions and colors. (d) Stripes after propagation.

Object	#points	#propagated	Increase
Hand (Fig. 7)	29.531	36.305	23%
Bust	70.346	83.698	18%
Back (Fig. 8a+c+e)	104.824	110.900	5%
Face (Fig. 8f+g)	17.959	23.945	33%

Table 1: Improvement of reconstructible points by using stripe propagation.

observe that after using the propagation step, data from the fingers of the hand (Fig. 7) can be retrieved, which was impossible before, due to the uniqueness window. This shows the potential of this approach to achieve good reconstructions even in noisy or small regions.

4. Conclusion

We have presented a novel method for data exploitation during 3D reconstruction using color encoded structured light. The color decoding uses knowledge from previous stripes to increase the detection performance. In a propagation step decoded stripe numbers are expanded along the stripe direction to compensate for small bridges in geometry, noise and classification errors. Depending on surface complexity we were able to achieve a significant increase in the amount of reconstructible points. This is especially useful in scenarios, where each scan has to stand on its own, for instance in motion analysis. Different human body parts have been scanned to evaluate the performance of the algorithm.

The method itself can be improved by restricting Canny's edge tracing to follow only a limited set of directions avoiding further steps for rejection of horizontal edges. Future work deals with the optimization of the projected pattern to compensate for local variations in reflectance and absorption, i.e. the implementation of a scene adaptive pattern. A common problem during color detections is the inhomogeneous intensity in the captured images. Using temporal feedback we work on projecting a pattern that gives a homogeneous result.

References

[1] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986. 3

[2] S. Chen, Y. Li, and J. Zhang. Vision processing for realtime 3-d data acquisition based on coded structured light. *IEEE Trans. on Image Processing*, 17(2):167–176, Feb. 2008. 1

[3] N. de Bruijn. *A Combinatorial Problem.*, volume 49. Nederlandse Akademie v. Wetenschappen, 1946. 1, 2

[4] P. Fechteler and P. Eisert. Adaptive color classification for structured light systems. In *IEEE CVPRW 2008*, pages 1–7, June 2008. 1, 2

[5] P. Fechteler, P. Eisert, and J. Rurainsky. Fast and high resolution 3d face scanning. In *IEEE ICIP 2007*, volume 3, pages III –81–III –84, Oct. 2007. 1, 4

[6] F. Forster. A high-resolution and high accuracy real-time 3d sensor based on structured light. In *3DPVT 2006*, pages 208–215, June 2006. 1

[7] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, New York, NY, USA, 2000. 6

[8] H. Kawasaki, R. Furukawa, R. Sagawa, and Y. Yagi. Dynamic scene shape reconstruction using a single structured light pattern. In *IEEE Conf. on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008. 1

[9] T. Koninckx, A. Griesser, and L. Van Gool. Real-time range scanning of deformable surfaces by adaptively coded structured light. In *Proc. Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003.*, pages 293 – 300, oct. 2003. 2

[10] H. Li, R. Straub, and H. Prautzsch. Fast subpixel accurate reconstruction using color structured light. In *Proc. of the Fourth IASTED International Conference on Visualization, Imaging, and Image Processing*, pages 396–401. ACTA Press, September 2004. 2

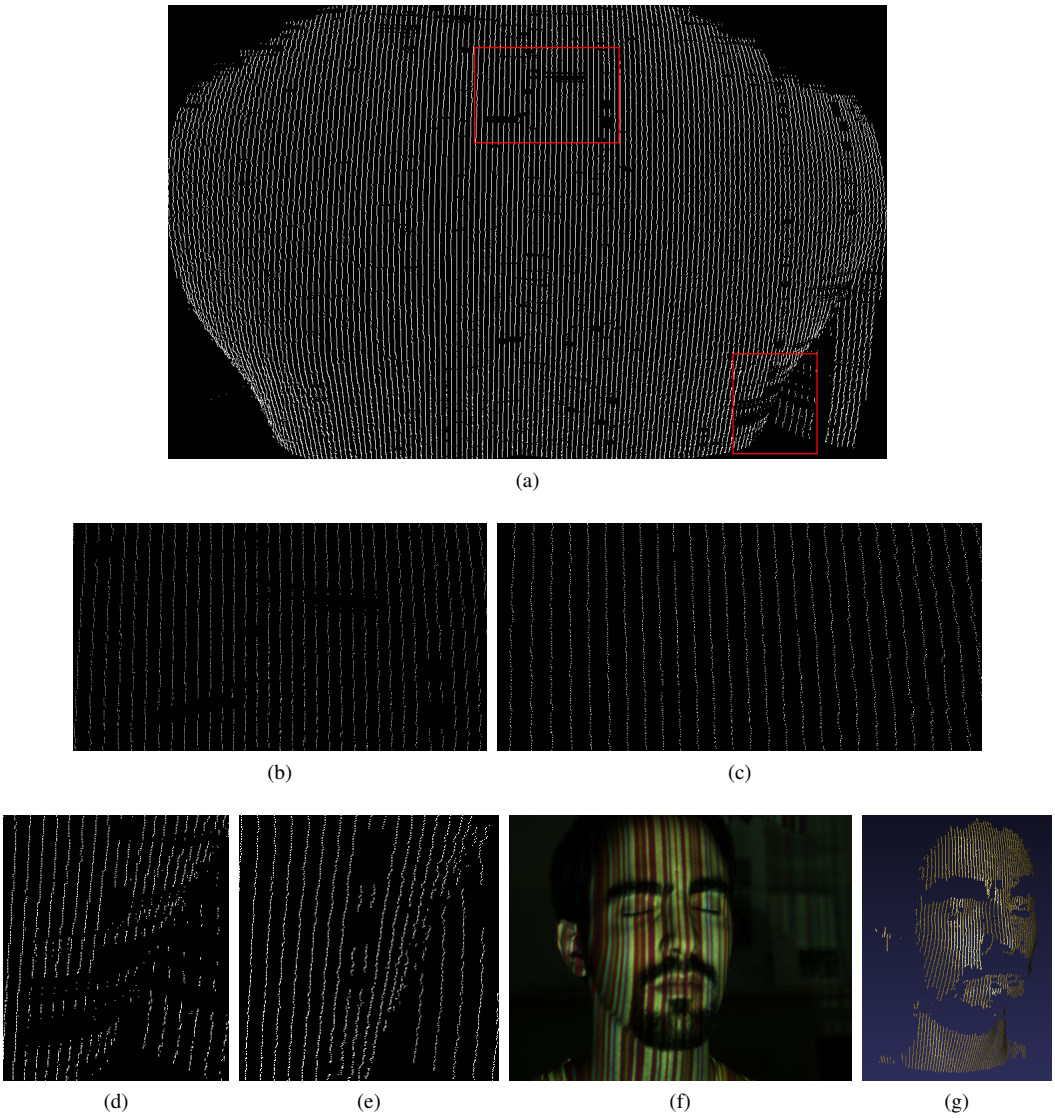


Figure 8: (a) Reconstructed back before propagation. (b) and (d) Magnification of highlighted areas. (c) and (e) Reconstruction after propagation. (f) Facial image with color pattern and (g) the facial reconstruction

[11] H. Li, R. Straub, and H. Prautzsch. Structured light based reconstruction under local spatial coherence assumption. In *3DPVT 2006*, pages 575–582, Washington, DC, USA, 2006. IEEE Computer Society. 1

[12] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man and Cybernetics*, 9(1):62–66, January 1979. 5

[13] J. Salvi, J. Pags, and J. Batlle. Pattern codification strategies in structured light systems. *Pattern Recognition*, 37(4):827 – 849, 2004. Agent Based Computer Vision. 1

[14] T. Weise, B. Leibe, and L. Van Gool. Fast 3d scanning with automatic motion compensation. In *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR '07.*, pages 1–8, June 2007. 1

[15] L. Zhang, B. Curless, and S. Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *3DPVT 2002*, pages 24–36, 2002. 1

[16] S. Zhang. Recent progresses on real-time 3d shape measurement using digital fringe projection techniques. *Optics and Lasers in Engineering*, 48(2):149 – 158, 2010. Fringe Projection Techniques. 1

[17] S. Zhang and P. Huang. High-resolution, real-time 3d shape acquisition. In *Conference on Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04.*, pages 28–28, June 2004. 1

[18] S. Zhang and P. S. Huang. High-resolution, real-time three-dimensional shape measurement. *Optical Engineering*, 45(12):123601, 2006. 1

Citation:

Burisch, Michael; Guerrero Ichaso, David; Kuijper, Arjan:
Stripe Propagation for Color Encoded Structured Light.
In: Fifth International Symposium on 3D Data Processing, Visualization and Transmission. Proceedings [online].
[cited 17 January 2011] Available from: http://campwww.informatik.tu-muenchen.de/3DPVT2010/data/media/e-proceeding/index.html, 2010, 8 p.