Modelling 3D Objects using 2D Sketches through Radial Renderings of Curvature Maps

Frode Eika Sandnes[1,2] and Evelyn Eika[1]

[1]Department of Computer Science, Faculty of Technology, Art and Design, OsloMet–Oslo Metropolitan University, Oslo, Norway
[2]Westerdals Oslo School of Art, Communication and Technology, Oslo, Norway
frodes@hioa.no, Evelyn.Eika@hioa.no

Abstract. Modelling 3D objects is challenging; often special software skills are required. This paper explores a new method for experimenting with 3D modelling using two-dimensional drawings. These drawings use coloured areas to dictate the rate of curvature. The curvature images are rendered in a radial manner from the centre to the sides. The method allows complex 3D shapes to be modelled. There is no need to employ any new software program as any arbitrary 2D painting application can be used to sketch objects.

Keywords: sketching, 3D-modelling, 2D hand drawings, design, ideation

1 Introduction

Three-dimensional models are used in areas such as computer graphics, design, and computer games. Usually 3D models are composed using 3D modelling software. Such software is often considered hard to use [1]. The input of 3D artefacts is needed using 2D input devices such as keyboards, mice, and drawing tablets. Time can be an important factor, especially in design where ideas emerge quickly, and the user wants to capture the idea as a sketch before it is forgotten [2]. Three-dimensional modelling is considered much harder than 2D visualizations [3] or other types of modelling including graphs used in scheduling [4, 5], configuration management [6], and interaction analysis [7]. Some user interfaces are cognitively demanding [8, 9, 10].

Several approaches have been proposed for specifying shapes in 3D using 2D representations [11, 12]. One simple approach is to use silhouettes [13] that are rotated to get the desired 3D shape. Others have experimented with two-dimensional curves which subsequently are used to specify shapes in 3D [14]. Researchers have attempted to automatically interpret and convert flat projective sketches of 3D objects into 3D models [15, 16, 17, 18]. However, this is a challenging problem as it is hard to interpret the exact location of a line in 3D-space. Another approach is to move the viewer using a tablet computer where the user draws 3D sketches on the tablet from various angles; by shifting the viewing angle, the model can be corrected and refined in real time [19]. Obtaining 3D modelling through several views of a 2D model from different angles of observation have also been attempted [20].

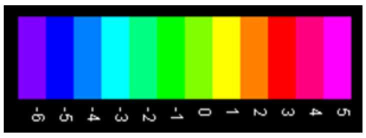


Fig. 1. Colour-to-level coding.

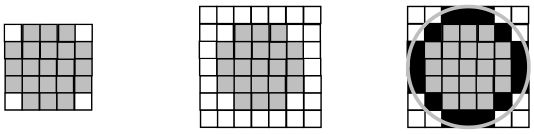


Fig. 2. Radial rendering using an expanding square with a disc test. Grey, white, and black pixels represent processed, unprocessed, and currently processed pixels, respectively.

One image can also be used to modulate the surface of a shape. In one approach, reliefs are added to arbitrary shapes using line drawings of the reliefs [21]. It is not always necessary to obtain a 3D model of an object, as it is sufficient to give the impression of a 3D object. This is particularly relevant for sketching where panoramic sketches are used to give 3D dimensional view from a single point [22, 23, 24, 25, 26].

Shading has also been used to define shape [27, 28, 29]. Several shading-based techniques have been proposed. Direct height maps use the grey-level to denote the height on a contour map, while illumination-based models interpret the image shape lit by a light source. Colour has also been used to define height [30, 31] as it is easier to determine a hue than the absolute level of grey [32, 33, 34, 35].

This study uses colour maps to specify levels of curvature instead of height. This allows complex shapes to be specified. This is like the classic turtle graphics in the logo programming language [36], which allows young children to construct mathematically complex shapes. For instance, a circle is drawn by the three instructions 1) one step forward, 2) turn one degree, 3) repeat. The turtle graphics concept has also more recently been extended to three dimensions [37]. However, unlike 3D turtle graphics, which rely on textual instructions, the current method uses visual instructions.

2 Method

The method proposed herein relies on a visual shaping language where the user provides a regular 2D image to render the 3D shapes. The visual language uses different colours to specify shape. The colours are organized according to the colour wheel where warm colours indicate positive curvatures and cold colours indicate negative curvatures (see Fig. 1). Yellow-green represents neutral curvature, but this can also be represented using white. The further a colour is from yellow-green on the colour wheel, the steeper the curvature is. The shaping starts at the origin of the shaping image and is processed radially outwards. Black lines are used to deflect the radial curvature.

Moreover, a second image can be used to specify the texture on the curve, where each pixel in the shape image is mapped to the corresponding image in the texture image. White pixels in the texture image are interpreted as transparency and can therefore be used to make holes in the shape or control the shape of the edges.

2.1 Pre-processing

First, a check is performed to determine if the shape image has the same dimensions as the texture image. If they are different, the texture image is resized to match the size of the texture image. Next, the shape image is quantized into N_{colors} discrete colours where *hue'* is the quantized hue and *hue* is the original hue using:

$$hue' = \frac{\lfloor hue \cdot N_{colors} \rfloor}{N_{colors}} \tag{1}$$

2.2 Radial Rendering

The method defines by convention that the rendering starts from the centre of the curvature map, that is, [C_i, C_j], where

$$C_i = \frac{W}{2}, C_j = \frac{H}{2} \tag{2}$$

and *W* is the width of the image and *H* is the image height. The image is then traversed by the area of a disc with increasing radius in steps of 1 from $R_{min} = 0$ to

$$R_{max} = \sqrt{{C_i}^2 + {C_j}^2} \tag{3}$$

For each radius *r*, the 8×*r* pixels *i*, *j* lying on the square around the centre point are added to the list of points to be processed *P* (see Fig. 2). That is,

$$[C_i + t, C_j + r], [C_i + r, C_j + t], [C_i - t, C_j - r], [C_i - r, C_j - t], t \in [-r..r] \tag{4}$$

In other words, these points represent a growing square with side $2r$. For all the points in the list *P* a disc-test is performed: If the distance between the point and the centre is not larger than *r*, the point is removed from *P* and the point is rendered. That is

$$r \geq \sqrt{(C_i - i)^2 + (C_j - j)^2} \tag{5}$$

where *i*, *j* is a given point. This procedure ensures that points are processed in increasing radius, and that no points are missed. Moreover, no trigonometric functions are needed.

2.3 Rate-of-Curvature

The hue of each pixel in the shape image is interpreted as follows. First, the use is converted from radians to the interval -1..1 where the origin is located at yellow-green or π/2, that is

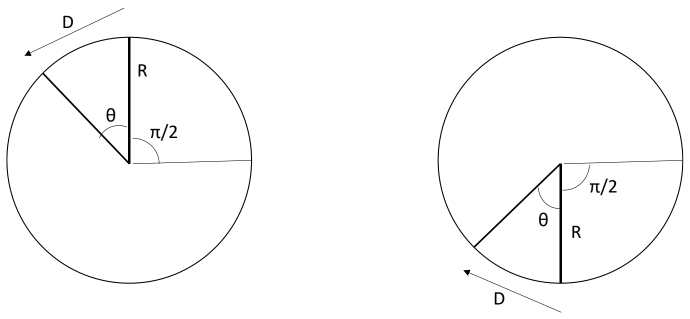


Fig. 3. Negative curvature (left) and positive curvature (right) defined in terms of circle radius.

$$H(x) = \begin{cases} \frac{\pi}{2} - x, \; x < \frac{3\pi}{2} \\ \frac{5\pi}{2} - x, \; x \geq \frac{3\pi}{2} \end{cases} \tag{6}$$

where *x* is the hue and *H*(*x*) is the converted scale. The shape value is discretized using

$$h(x) = \left\lfloor \frac{H(x)}{N_{colors}} \right\rfloor \tag{7}$$

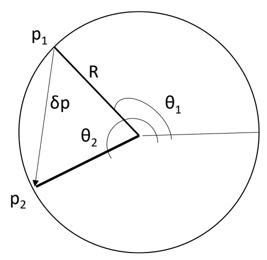
The shape value is used to define the rate of curvature. This rate is defined as the radius of the circle that yields the curvature (see Fig. 3). The radii are related to the dimension of the image. Therefore, medium curvature is defined as a circumference equal to the image width, that is, *W*=*D*, while steeper curvatures are achieved with circles with circumferences of half the width, quarter the width, etc., namely, *W*/2, *W*/4, etc. Less steep curvatures are achieved with circumferences twice and quadruple that of the image width, namely, $2W$, $4W$, etc. The circumference can thus be defined as

$$D = W2^m, \text{ and } m = \frac{N_{colors}}{4} - |h(x)| \tag{8}$$

This is based on the assumption that half of the colours denote negative curvatures and the other half of the colours denote positive curvatures. For each of these halves, half of the colours denote circles with circumferences smaller than the image width and the other half larger than the image width. Since the circumference of a circle is *D*=2π*R*, the radius *R* is thus

$$R = s\frac{W2^m}{2\pi} \tag{9}$$

where *s* is the sign of *h*(*x*). Note that the curvatures increase exponentially.



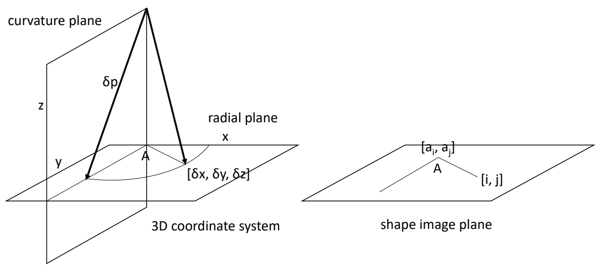


Fig. 4. Position change due to curvature.

Fig. 5. Transforming a point in the curvature plane to the modelling space using the angle in the radial plane.

2.4 Curvature Computation

The corresponding point x, y, z in 3D space given a point i. j on the curvature image is computed as follows. First, the correct anchor point a_i, a_j is determined. If no other anchor points are defined, the origin of the radial rendering is the anchor point, namely, C_i, C_j. Then, the curve image is scanned along the line from i, j to the anchor point a_i, a_j, and the position of the first transition between two different colours are recorded and used as a new anchor point a_i, a_j. Since this point is located closer to the centre, it will already have been processed with the 3D coordinates x_a, y_a, z_a at an angle of θ_a. Note that the first processed pixel, which is the centre, is set to x=0, y=0, z=0, $\theta=\pi/2$. The length between the current point and the anchor point on the image is named D. Eq. (9) is used to compute the radius R of the circle, which defines the slope. The slope at point i. j in radians is therefore

$$\theta = \theta_a + \frac{D}{R} \tag{10}$$

This is because the circumference of the circle is given by $2\pi R$ and that 360 degrees represent 2π radians. The factor 2π thus cancels out in the denominator and nominator of the fraction. Next, the points on the circle given by the two angles θ and θ_a are computed assuming the centre at the origin, namely

$$p_1 = [R\cos\theta, R\sin\theta] \tag{11}$$
$$p_2 = [R\cos\theta_a, R\sin\theta_a] \tag{12}$$

The relative distance δp travelled is thus $\delta p = p_2 - p_1$ (see Fig. 4). The point δp is then rotated into the 3D coordinate system using $\delta x = \delta p_x \sin A$, $\delta y = \delta p_x \cos A$, and $\delta z = \delta p_y$. The orientation A is given by the angle made up by the vector from the anchor point to the current point in the curvature image, namely

$$A = atan2(j - a_j, i - a_i) \tag{13}$$

The final 3D point is therefore, $x = x_a + \delta x$, $y = y_a + \delta y$, and $z = z_a + \delta z$. If the angle for a given length D is 0, the new point is simply computed instead using

$$\delta p = [D \cos L\,, D \sin L], \text{ where } L = \frac{\theta_a}{|\theta_a|}\left(|\theta_a| + \frac{\pi}{2}\right) \tag{14}$$

since zero curvature change is a straight line (see Fig. 5). Note that the angle of the line is perpendicular to the corresponding angle on the circle.

2.5 Complex Anchor Points

Black is used to indicate anchor points, which are processed radially, meaning that they have only effect on pixels shadowed by the anchor point relative to the previous anchor point. First, a list of all black points is complied. Next, to find the anchor point of a given point *i, j*, it is first determined if there is clear sight to the center. If there is clear sight to the center, that is, the line from the current point to the center does not intersect any black pixel, the center point is used as the anchor. Otherwise, the nearest black pixel on the processed disk is found. If there is an even closer pixel in the list of all pixels, the center point is used as the anchor. However, if the closest point on the disk of processed points is the nearest, it is used as the anchor.

3 Modelling case studies

The models demonstrated herein were visualized using CloudCompare [38]—a tool for visualizing point clouds [39]. Fig. 6 shows basic features of the modelling approach based on a uniform curvature map. The top illustration shows part of a spherical surface modelled using yellow representing a small positive curvature and hence a large radius. The second shape is a full sphere with negative curvature modelled using a uniform cyan image. Clearly, it is easy to model spheres with the approach. The bottom shape shows both the full sphere from the previous example and a smaller sphere modelled using a steep positive curvature (magenta). The sphere makes several revolutions (or shells) although these are not visible. The two spheres are side by side since one is curving negatively and the other positively from the same origin.

Fig. 7 shows a ring. A simple uniform orange shape map is used to achieve a sphere with one revolution. Next, the ring is cut out from the sphere using the yellow texture map. Next, a variation on the same theme is provided below where the ring is punched with holes and red and orange speckles are added.

Fig. 8 shows enhancement using a texture map. The top image shows the resulting curved grid, where the grid is drawn directly on the texture map and the holes are made using white. Note that the relationship between coordinates in the image plane and the sphere surface is not as simple as with geographical coordinates of latitude and longitude [40, 41]. Some experimentation is needed to map textures to shapes. The second example does not have holes but illustrates how the striped image is mapped. It is also possible to block the 2D-curving by adding a black line as illustrated in Fig. 8 right.

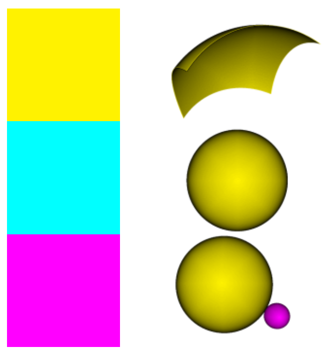


Fig. 6. Simple curvature, low curvature, medium curvature (full sphere), high negative curvature (small sphere).

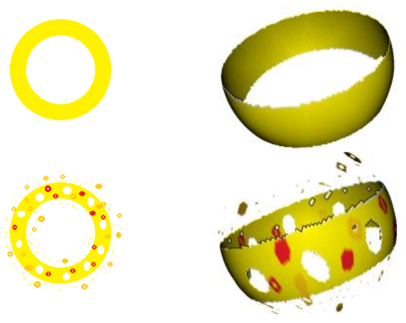


Fig. 7. Ring.

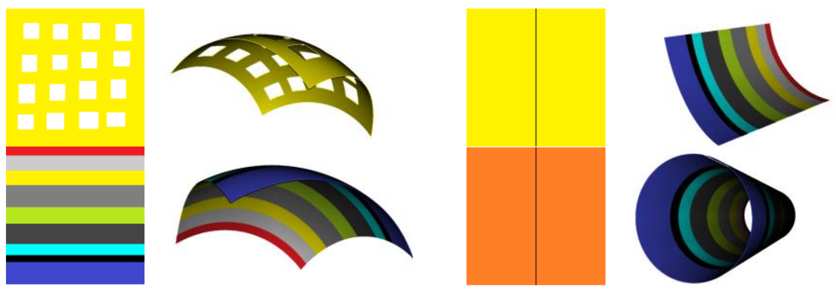


Fig. 8. Adding texture to basic shapes, including transparency (left) and cylinders (right).

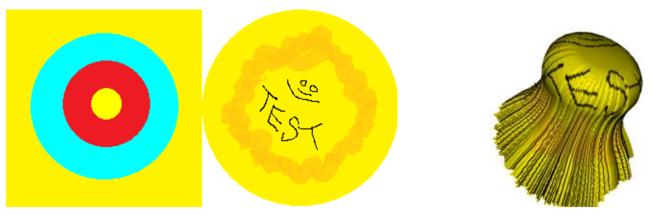


Fig. 9. Chess piece.

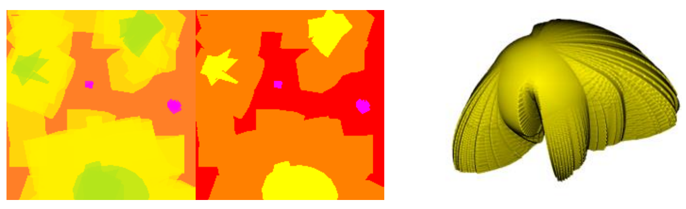


Fig. 10. Arbitrary shape contour, quantized shape contour and resulting shape.

This line signals that the radial rendering from the single centre point is replaced by a set of anchor points resulting in a horizontal rendering, that is, middle-left, middle-right. The top illustration shows the result yielding part of a tube, and the bottom a tube.

Fig. 9 shows a simple chess piece. It is constructed using consecutive coloured rings. First, yellow is used to achieve a flat top, and red is used to make a round shaping downwards. Next, a cyan ring bends the shape outwards, and finally yellow is used to make the piece point slightly inwards. A circular texture map is used to cut off the corners. The example reveals some computational inaccuracy in the proof-of-concept implementation as the surface jaggedness increased with the number of colour transitions. The last example shows the result of using an arbitrary shape map in Fig. 10 (left). This shape map is drawn in Microsoft Paint. The middle and right images show the colour quantization and the resulting blob.

4 Conclusions

The modelling method explored herein allows the modelling of certain classes of curved shapes and does not have the same limitations as height-based modelling. The method facilitates easy experimentation of complex 3D modelling using ordinary 2D drawing programs as there is no need to learn new software tool. However, it is not intuitive or easy to control the resulting shapes based on curvature maps; therefore, it is not as suitable as height-based modelling for controlling the resulting shape. It may thus be more suitable as an experimental, exhibition [42, 43], and educational tool. The prototype yields undesirable aliasing effects due to the curvature maps. Future work may include applying various image- [44] and video processing [45, 45] to the input images.

References

1. Black, A.: Visible planning on paper and on screen: The impact of working medium on decision-making by novice graphic designers. Behaviour & Information Technology 9, 283-296 (1990)
2. Sandnes, F.E., Jian, H.L.: Sketching with Chinese calligraphy. Interactions 19, 62-66 (2012)
3. Eika, E., Sandnes, F.E.: Authoring WCAG2.0-compliant texts for the web through text readability visualization. In: Antona, M., Stephanidis, C. (eds.) UAHCI 2016. LNCS, vol. 9737, pp. 49–58. Springer, Cham (2016)
4. Sandnes, F. E., Sinnen, O.: A new strategy for multiprocessor scheduling of cyclic task graphs. International Journal of High Performance Computing and Networking 3, 62-71 (2005)
5. Rebreyend, P., Sandnes, F. E., Megson, G. M.: Static multiprocessor task graph scheduling in the genetic paradigm: A comparison of genotype representations. Research re-port no. 98-25. Ecole Normale Superieure de Lyon, Laboratoire de Informatique du Par-allelisme, Lyon, France (1998)
6. Sandnes, F. E.: Scheduling Partially Ordered Events in a Randomised Framework: Empirical Results and Implications for Automatic Configuration Management. In: Proceedings of LISA, pp. 47-62. USENIX (2001)

7. Sandnes, F. E.: Evaluating mobile text entry strategies with finite state automata. In: Proceedings of the 7th international conference on MobileHCI 2005, pp. 115-121. ACM (2005)
8. Sandnes, F.E., Jian, H.L.: Pair-wise variability index: Evaluating the cognitive difficulty of using mobile text entry systems. In: International Conference on MobileHCI 2004. LNCS, vol. 3160, pp. 347-350. Springer Berlin Heidelberg (2004)
9. Berget, G., Sandnes, F.E.: Do autocomplete functions reduce the impact of dyslexia on information searching behaviour? A case of Google. J. Am. Soc. Inf. Sci. Technol. 67, 2320–2328 (2016)
10. Sandnes, F.E., Lundh, M.V.: Calendars for Individuals with Cognitive Disabilities: A Comparison of Table View and List View. In: Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility, ACM, pp. 329-330 (2015)
11. Olsen, L., Samavati, F.F, Sousa, M.C., Jorge, J.A.: Sketch-based modeling: A survey. Computers & Graphics 33, 85-103 (2009)
12. Kondo, K.: Interactive geometric modeling using freehand sketches. J. Geom. Graphics 13, 195-207 (2009)
13. Tai, C.L., Zhang, H., Fong. J.C.K.: Prototype modeling from sketched silhouettes based on convolution surfaces. Computer Graphics Forum 23, 71-83 (2004)
14. Das, K., Diaz-Gutierrez, P., Gopi, M.: Sketching free-form surfaces using network of curves. In: Proceedings of Eurgraphics Workshop on Sketch-Based Interfaces and Modeling, pp. 127-134, The Eurographics Association (2005)
15. Varley, P.A.C., Martin, R.R, Suzuki, H.: Can machines interpret line drawings. In: Proceedings of Eurgraphics workshop on sketch-based interfaces and modelling, pp. 107-116, The Eurographics Association (2004)
16. Naya, F., Jorge, J., Conesa, J., Contero, M., Gomis, J. M.: Direct modeling: from sketches to 3D models. In: Proceedings of the 1st Ibero-American Symposium in Computer Graphics SIACG, pp. 109-117, (2002)
17. Matondang, M.Z., Mardzuki, S., Haron. H.: Transformation of engineering sketch to valid solid object. In Proceedings of Intl. Conf. of The 9th Asia Pacific Industrial Engineering & Management Systems Conference and The 11th Asia Pacific Regional Meeting of International Foundation for Production Research, pp. 2707-2715, (2008)
18. Tolba, O., Dorsey, J., McMillan, L.: Sketching with projective 2d strokes. In: Proceedings of the 12th annual ACM symposium on User interface software and technology, pp. 149-157, ACM (1999)
19. Xin, M., Sharlin, E., Sousa, M.C.: Napkin sketch: handheld mixed reality 3D sketching. In: Proceedings of the 2008 ACM symposium on Virtual reality software and technology, ACM (2008)
20. Triki, O., Zaharia, T.B., Preteux, F.J.: 3D virtual character reconstruction from projections: A NURBS-based approach. In: Electronic Imaging 2004, International Society for Optics and Photonics (2004)
21. Kolomenkin, M., Leifman, G., Shimshoni, I., Tal, A.: Reconstruction of relief objects from line drawings. Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE, 2011.
22. Sandnes, F.E.: Communicating Panoramic 360 Degree Immersed Experiences: A Simple Technique for Sketching in 3D. In: UAHCI, LNCS, vol. 9738, pp. 338-346, Springer (2016)
23. Sandnes, F.E.: PanoramaGrid – A Graph Paper Tracing Framework for Sketching 360-degree Immersed Experiences. In Proceedings of the International Working Conference on Advanced Visual Interfaces AVI 2016, pp. 342-343, ACM (2016)
24. Sandnes, F.E., Huang, Y.P.: Translating the viewing position in single equirectangular panoramic images. In: Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC 2016), pp. 389–394. IEEE (2016)

25. Sandnes F.E., Eika, E.: Head-Mounted Augmented Reality Displays on the Cheap: A DIY Approach to Sketching and Prototyping Low-Vision Assistive Technologies, In: Antona, M., Stephanidis, C. (eds.) UAHCI 2017, LNCS, vol. 10278, pp. 168-186, Springer, (2017)
26. Sandnes, F.E.: Sketching 3D Immersed Experiences Rapidly by Hand through 2D Cross Sections. In: Auer, M.E. (eds.) REV2017, LNNS, vol 22, pp. 1001-1013, Springer (2017)
27. Gingold, Y., Zorin, D.: Shading-based surface editing. ACM Transactions on Graphics 27 (2008)
28. Zhang, R., Tsai, P.S., Cryer, J.E., Shah, M.: Shape-from-shading: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 21, 690-706 (1999)
29. Gouraud, H.: Continuous shading of curved surfaces. IEEE Transactions on Computers C–20, 623–629 (1971)
30. Sandnes, F.E., Lianguzov, Y., Rodrigues, O.V., Lieng, H., Medola, F.O., Pavel, N.: Supporting Collaborative Ideation Through Freehand Sketching of 3D-Shapes in 2D Using Colour. In: Luo Y. (eds.) CDVE 2017, LNCS, vol. 10451, pp. 123-134, Springer (2017)
31. Sandnes, F.E., Lianguzov, Y.: Quick and Easy 3D Modelling for All: A Browser-based 3D-Sketching Framework, iJOE 13, 120-127 (2017)
32. Sandnes, F.E.: Understanding WCAG2.0 color contrast requirements through 3D color space visualization. Stud. Health Technol. Inform. 229, 366–375 (2016)
33. Sandnes, F.E.: On-screen colour contrast for visually impaired readers: Selecting and exploring the limits of WCAG2.0 colours. In: Black, A., Lund, O., Walker, S. (eds.) Information design: research and practice, pp. 405–416, Routledge (2016)
34. Sandnes, F.E., Zhao, A.: A contrast colour selection scheme for WCAG2.0-compliant web designs based on HSV-half-planes. In: Proceedings of SMC2015, pp. 1233–1237. IEEE (2015)
35. Sandnes, F.E., Zhao, A.: An interactive color picker that ensures WCAG2.0 compliant color contrast levels. Procedia Comput. Sci. 67, 87–94 (2015)
36. Solomon, C.J., Papert, S.: A case study of a young child doing Turtle Graphics in LOGO. In: Proceedings of the national computer conference and exposition, pp. 1049-1056, ACM (1976)
37. Verhoeff, T.: 3D turtle geometry: artwork, theory, program equivalence and symmetry. International Journal of Arts and Technology 3, 288-319 (2010)
38. Girardeau-Montaut, D.: CloudCompare-Open Source project. OpenSource Project (2011).
39. Gomez, J.V., Sandnes, F.E.: RoboGuideDog: guiding blind users through physical environments with laser range scanners. Procedia Comput. Sci. 14, 218–225 (2012)
40. Sandnes, F.E.: Where was that photo taken? Deriving geographical information from image collections based on temporal exposure attributes. Multimedia Systems 16, 309-318 (2010)
41. Sandnes, F.E.: Determining the geographical location of image scenes based on object shadow lengths. Journal of Signal Processing Systems 65, 35-47 (2011)
42. Huang, Y.P., Wang, S.S., Sandnes, F.E.: RFID-based guide gives museum visitors more freedom. IT Professional Magazine 13, 25 (2011)
43. Huang, Y. P., Chang, Y. T., Sandnes, F. E.: Ubiquitous information transfer across different platforms by QR codes. Journal of Mobile Multimedia 6, 3-13 (2010)
44. Huang, Y.P., Chang, T.W., Chen, Y.R., Sandnes, F.E.: A back propagation based real-time license plate recognition system. International Journal of Pattern Recognition and Artificial Intelligence 22, 233-251 (2008)
45. Huang, Y.P., Chiou, C.L., Sandnes, F.E.: An intelligent strategy for the automatic detection of highlights in tennis video recordings. Expert Systems with Applications 36, 9907-9918 (2009)
46. Huang, Y.P., Hsu, L.W., Sandnes, F.E.: An intelligent subtitle detection model for locating television commercials. IEEE Trans. Man Cybern. B 37, 485–492 (2007)