Determining Visible Points in a Three-Dimensional Discrete Space

Grit Thürmer[1,2]*, Arnault Pousset[2], and Achille J.-P. Braquelaire[2]

[1] CoGVis/MMC – Computer Graphics, Visualization, Man-Machine Communication Group, Faculty of Media, Bauhaus-University Weimar, 99421 Weimar, Germany
thuermer@medien.uni-weimar.de

[2] LaBRI – Laboratoire Bordelais de Recherche en Informatique – UMR 5800, University Bordeaux 1
351, cours de la Libération, 33405 Talence, France
{pousset,braquelaire}@labri.u-bordeaux.fr

Abstract. A method is proposed which computes the visible points of surfaces in a 3-dimensional discrete space. The occlusion of surface points of an object by other object points is determined by shooting a discrete ray from each surface point towards the center of projection considering the intersection of the ray with other object points. Since the projection of points onto the viewing plane is done by a continuous mapping, additionally to the discrete ray, the location of the continuous projection ray is examined regarding its location to the surface points that are intersected by the discrete ray.

1 Introduction

The growing interest of computer graphics in the three-dimensional discrete space $\mathbb{Z}^3$ has led to new application fields of volume data: e.g. virtual reality in medicine [19], volume-based interactive design and sculpturing [16,11]. For such application fields, synthetic objects, i.e. geometrically defined objects in $\mathbb{Z}^3$ or data sets obtained by rastering [1,15] geometric descriptions of objects given in $\mathbb{R}^3$, are generated and the boundaries of these objects are required to be visualised.

Three principal approaches for rendering discrete objects can be distinguished: *backward* or *image-order projection*, *forward* or *object-order projection* techniques, and *hybrid techniques* [14,10,9] that combine advantages of both the backward and forward projection methods. Backward projection algorithms traverse the pixels and solve the visibility problem for each pixel by casting a ray from the viewing point through each pixel of the image into the data space. This group includes *ray-casting* [5,6,3] and *ray-tracing* [18] techniques, which have

* This work has been supported by a postdoctoral grant of the DFG (Deutsche Forschungsgemeinschaft).

been specifically developed for the rendering of volumes. Forward projection algorithms, e.g. *splatting* [17,4], traverse the three-dimensional discrete scene and project its components onto the viewing plane.

In contrast to splatting techniques, where a point of $\mathbb{Z}^3$ is represented by a kernel function in the viewing plane, we assume that a point of $\mathbb{Z}^3$ is mapped to the viewing plane such that its image is a point in $\mathbb{R}^2$. Consequently, generally the image of a discrete surface in the viewing plane is a set of scattered of points. Moreover, assuming a continuous viewing plane requires a method for the determination of visible points, which provides results independently from the resolution of the final image. Therefore, rendering methods applying the z-buffer algorithm are not suitable to solve the visibility problem. Instead, a technique is needed which determines the visible points of a scene in object space, i.e. in $\mathbb{Z}^3$. A similar problem has to be solved in discrete ray-tracing [2] when it has to be determined if a point is in shadow, i.e. if the point cannot be seen from the light source. After the visible points are mapped onto the viewing plane, the final image could be obtained by applying a technique, e.g., as proposed in [12].

In this paper, a method is proposed which determines the visible points of closed surfaces in $\mathbb{Z}^3$. Each surface forms the boundary of a discrete object. In a first step, front facing points are computed in the common way, comparing the normal vectors at the surface points with the viewing direction. The computation of occluded front facing points is performed utilising the idea of forward ray-casting: a discrete ray is shot from each surface point towards the centre of projection considering the intersection with other non-empty points, i.e. points which belong to objects of the scene. Since the actual projection of surface points onto the viewing plane is done by a continuous mapping, additionally to the discrete ray, the location of the continuous projection ray is examined regarding its location to the surface points that are intersected by the discrete ray.

The paper is organised as follows: Section 2 states the definitions used throughout the paper. In Sect. 3, at first the problem is examined in detail. Afterwards, a solution is proposed in 2D which is then extended to 3D. Experimental results are presented and discussed in Sect. 4. Finally, Sect. 5 summarises the paper.

2 Definitions

The three-dimensional discrete space is constituted by $\mathbb{Z}^3$, that is the 3D array of points with integer coordinates in the Cartesian coordinate system. Two points $p(x_p, y_p, z_p)$ and $q(x_q, y_q, z_q)$ of $\mathbb{Z}^3$ where $(\mid x_p - x_q \mid \leq 1) \wedge (\mid y_p - y_q \mid \leq 1) \wedge (\mid z_p - z_q \mid \leq 1)$, are said to be *6-adjacent* if $\mid x_p - x_q \mid + \mid y_p - y_q \mid + \mid z_p - z_q \mid = 1$, *18-adjacent* if $0 < \mid x_p - x_q \mid + \mid y_p - y_q \mid + \mid z_p - z_q \mid \leq 2$, and *26-adjacent* if $0 < \mid x_p - x_q \mid + \mid y_p - y_q \mid + \mid z_p - z_q \mid \leq 3$. Points k-adjacent to p, where $k \in \{6, 18, 26\}$, are called *k-neighbours* of p. A *k-component* of a set A is a maximal subset of A in which between each pair of points p, q exists a sequence of distinct points $P = \{p = p_0, p_1, ..., p_n = q\}$ of A whereby two consecutive points p_i and p_{i+1} with $0 \leq i < n$ along the sequence are k-adjacent.

The surface $S \subset \mathbb{Z}^3$ to be rendered is assumed as a closed surface that forms the boundary of a finite object $O \subset \mathbb{Z}^3$, whereby O is a 6-connected component. The points of O are called *object points*. The surface $S \subset O$ is constituted of all points of O that are 6-adjacent to some point of $\overline{O} = \mathbb{Z}^3 - O$. Each point of S is also required to be 26-adjacent to some point of $O - S$. Note that S may consist of more than one component. The points of S are denoted as *surface points*. In contrast, the points in $O - S$ are called *inside points* and the points of $\overline{O}$ are named *empty points*.

3 Determination of Visible Surface Points

In general, a point on a closed surface is *visible* only if it is front facing, i.e. the surface in this point is oriented towards the observer, and the point is not occluded by another point.

The computation of front facing points is rather trivial and can be done as described in Sect. 3.1. In contrast, the determination of occluded points in discrete space needs further investigations. Section 3.2 describes the basic difficulties of the solution of the occlusion problem, which arise while employing the idea of discrete forward ray casting. In Sect. 3.3, an approach is introduced to determine occluded points in 2D space. Then, in Sect. 3.4 this approach is extended to 3D.

3.1 Front Facing Surface Points

The culling of back facing surface points in discrete space is done in the same way as for polygons: the cosine of the angle δ between the normal vector at a surface point and the vector representing the viewing direction is determined. If $cos(\delta) \leq 0$ the point is back facing, otherwise it is front facing.

Clearly, a normal vector associated with each point of the discrete surface is essential to cull back facing points. The normals are required to represent the local surface geometry. The determination of normal vectors at discrete surfaces is an active field of research. For example, the method introduced in [13] provides results that are suitable for the purpose of this work. However, to deal with the problem of visibility independently of the computation of the normal vectors, one could also associate the discrete surface points with the true normals of the underlying continuous surface.

3.2 Forward Ray-Casting

To determine the surface points that are not occluded by other points, the idea of forward ray-casting is utilised: shoot a discrete ray, denoted as *discrete viewing ray*, from each front facing surface point along the viewing direction and check if the ray intersects any other object point. A discrete viewing ray is the rasterization of the straight line segment between a surface point and the centre of projection.

The basic problems, which arise for the discrete viewing rays and are matter of this section, address the features of a ray in order to define a point as occluded and the connectivity of the rays. There are two basic approaches possible to solve the first problem: a point is not occluded only if its discrete viewing ray does not intersect any other object point, or the ray is allowed to be tangent to the surface, i.e. the ray may intersect other surface points but no inside point. These two approaches and their related problems are discussed below.

Viewing Rays do not Intersect any Object Point. The following assumption would be the simplest approach when solving the visibility problem in $\mathbb{Z}^3$ by forward ray-casting: a point is not occluded if its discrete viewing ray does not intersect any object point. However, this assumption is too strict. Frequently, the viewing ray of a surface point close to the contour of a surface is almost tangent to the surface, i.e. the ray intersects surface points but no inside point. Examples for this fact are shown in Fig. 1(a) and (b) for orthographic and perspective projection, respectively. For simplification, these examples are given in 2D. Clearly, the viewing rays shown in the Fig. 1 intersect other surface points. The condition of non-occlusion, as stated above, would lead to a missing of such points. A similar problem has been reported by Delfosse et al. [2] for shadow rays in discrete ray-tracing. They solve the problem by considering the real boundary of the underlying continuous object. This approach is not suitable for our work since we consider only the discrete objects, without any assumption on the underlying continuous objects. Consequently, we need a further specification of the condition for non-occluded points to allow viewing rays to be tangent to the surface, so that the discrete viewing ray of a visible point may intersect other surface points.

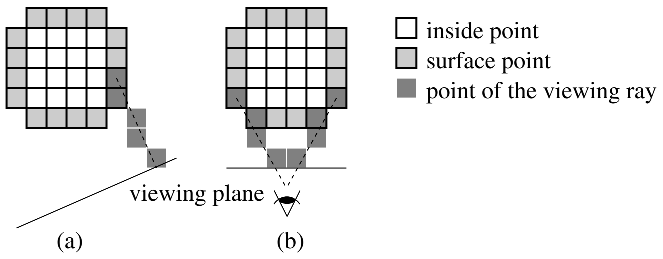

Fig. 1. Discrete viewing rays intersecting a surface point for (a) orthographic projection and (b) perspective projections

Connectivity of Viewing Rays. If there is no further restriction on the topology of the surface, i.e. S is defined as in Sect. 2, the discrete viewing rays have to be 6-connected to avoid a traversal of the object by the rays while intersecting only surface points. In such cases, the viewing rays are not tangent to the surface. An example in 2D is shown in Fig. 2(a). For the points p and q, a 26-connected viewing ray would pass through the object, hitting only surface points.

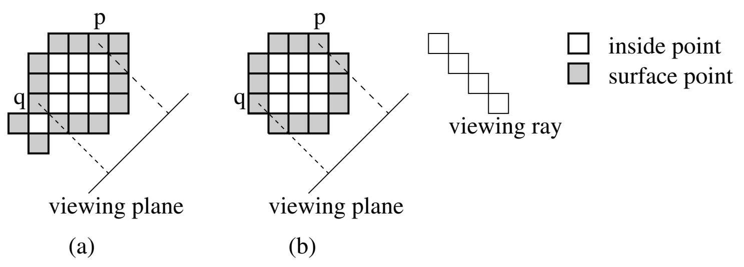

Fig. 2. 26-connected viewing rays and (a) a surface with no restriction and (b) a surface satisfying the definition from discrete topology

The generation of 26-connected rays is twice as fast than that of 6-connected rays since a 6-connected ray consists of about twice as many points as the 26-connected ray. For optimisation purposes, a technique proposed in [18] can be applied to speed up the usage of 6-connected rays: to traverse the empty space between the objects of a scene quickly, the rays are 26-connected until they come close to an object. Then the connectivity of the rays changes to 6-connectivity. Nevertheless, 26-connected rays would be preferred for practical applications because of performance reasons.

Assume S satisfies a surface definition as known from discrete topology [8,7]. Then S can be viewed as the discrete analog of a closed two-manifold surface, and S is the minimal set which separates $\mathbb{Z}^3 - S$ into two non-empty 6-connected sets. Thus, each point of S is 6-connected to some point of $O - S$ and to some point of $\overline{O}$. Experiments have shown, that 26-connected viewing rays are sufficient if S has these properties. Fig. 2(b) shows an example for this case in 2D. It is beyond the scope of this paper to proof this fact. However, this will be a matter of future work. In Sect. 4, results are shown for both the cases 6-connected and 26-connected viewing rays.

Viewing Rays may Intersect other Surface Points. Allowing the discrete viewing rays to intersect surface points but no inside point may cause problems. It cannot be ensured that the projection of the visible points leads to a correct image. The problem is illustrated in Fig. 3. Consider a convex discrete object O, i.e. O is the discrete representation of a convex continuous object $\tilde{O}$, located in the viewing space with coordinate axes x and y. The image of O, denoted O', is the projection of O onto the x-axis. If viewing rays of visible points may hit any surface point, the point $p \in O$ would be visible. Since p bounds the set of visible points of O shown in light grey in Fig. 3(a), one would expect that the image p' of p bounds O', i.e. it should be the point of O' with the smallest x-value in Fig. 3(a). This case is illustrated for the continuous object $\tilde{O}$ in Fig. 3(b). However for the discrete object in Fig. 3(a), after the projection of the visible points the image q' of the visible point q forms the boundary of O' instead. This problem arises because discrete rays are applied to determine the visible points in $\mathbb{Z}^3$ and, in contrast, a continuous mapping is used for the projection of the visible points onto the viewing plane. Therefore, a criterion is needed which ensures that points like p are excluded from the set of visible points while allowing viewing rays to intersect other surface points.

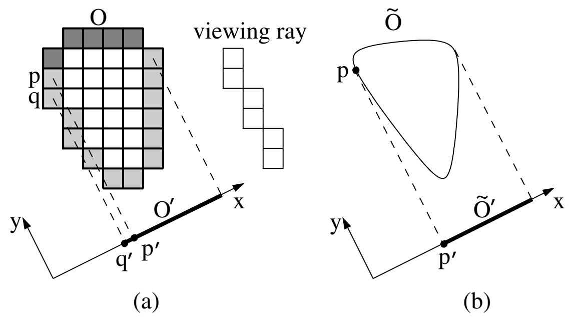

Fig. 3. Problem arising after mapping visible points onto the viewing plane: (a) for the discrete case, the projection of the point p does not form an outline of the image as it does in (b) the corresponding continuous case

3.3 Visibility in 2D

To solve the problem described in the previous section, a condition is introduced to exclude points from the set of visible surface points whose projection would lead to artefacts in the rendered image. This problem may arise only for surface points whose discrete viewing rays intersect other surface points.

At first, the problem is examined in 2D. Consider a front facing surface point p associated with a discrete viewing ray r_p, which intersects only empty points or other surface points. In case r_p intersects any inside point, p is not visible. The point p is not occluded if for each surface point q along r_p the continuous projection ray $\tilde{r}_p$ of p does not intersect O. Note that $\tilde{r}_p$ is the continuous representation of r_p. If $\tilde{r}_p$ is referred to as a vector subsequently then it is considered as the normalised vector representing the projection ray of p pointing towards the centre of projection.

Assume a continuous surface patch S_q of differential size with a normal vector N_q located at a surface point $q \in r_p$ and check the location of $\tilde{r}_p$ with respect to S_q. The vector N_q is assumed to be oriented towards $\overline{O}$. Thus, it gives us some notion on which side of S_q the inside of the object O is located. Consequently, it can be determined if $\tilde{r}_p$ crosses the inside of O in the neighbourhood of q.

The formal realization of this test is described in the following and is illustrated in Fig. 4. The straight line defined by $\tilde{r}_p$ separates the 2D space into two half-planes. This is illustrated in Fig. 4(a). By translating the normal vector N_q into p, it can be checked if $\tilde{r}_p$ intersects O in the neighbourhood of q depending on which half-plane $p + N_q$ belongs to and where q is located. If $p + N_q$ and q are in different half-planes, p is not occluded by q. This case is shown Fig. 4(b). The line through q represents the surface patch S_q in q. The projection ray of p goes along the outside of the surface in the neighbourhood of q. This is additionally illustrated by the projections p', q' and N_q' of p, q and N_q, respectively, onto the viewing plane. If $p + N_q$ belongs to the same half-plane containing q or $q \in \tilde{r}_p$, p is not visible. In the first case, the projection ray of p goes along the inside of the object in the neighbourhood of q as illustrated in Fig. 4(c). In the latter case, the projections of p and q are identical, but q is closer to the observer and, therefore, it occludes p.

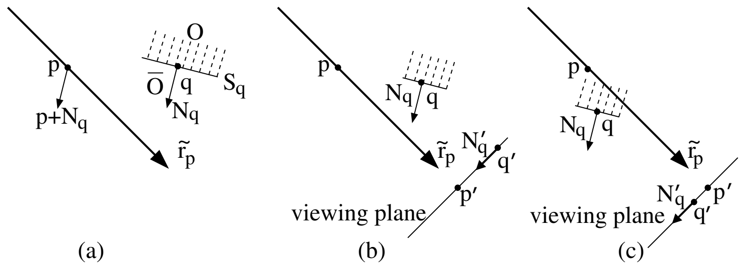

Fig. 4. (a) Subdivision of the space by $\tilde{r}_p$ into half-planes, (b) and (c) location of q in the half-planes with respect to the normal vector N_q of q

The two cases discussed above do not cover a normal vector N_q having the same direction like $\tilde{r}_p$, i.e. $\tilde{r}_p = N_q$ or $\tilde{r}_p = -N_q$. If the normal vectors associated with the discrete surface points reflect the local surface configurations correctly, one can expect for $\tilde{r}_p = -N_q$, that r_p hits also some inside point of the object since it crosses S with respect to N_q from the outside to the inside. If $\tilde{r}_p = N_q$, r_p would pass through S in q from the inside to the outside and should also hit some inside point. In fact, these cases could not appear for a discrete viewing ray which is tangent to a closed surface. If they arise nevertheless in a practical application, we assume for these cases that q occludes p as presented above.

3.4 Visibility in 3D

For the extension of the approach described in the previous section to 3D, an additional dimension for the location of the local surface in q with respect to $\tilde{r}_p$ has to be taken into account.

Consider the surface patch S_q in q and its projection onto the viewing plane. Without loss of generality, assume the viewing plane is located in q. (More generally, the viewing frustum is assumed to be bounded by a front clipping plane located in q and parallel to the projection plane.) Unless the direction of N_q or $-N_q$ is equal to the projection ray $\tilde{r}_q$ of q, the intersection of the projection plane and S_q leads to a curve γ_q with a normal vector N_q' that is the projection of N_q. This is illustrated in Fig. 5(a). Note, that for $\tilde{r}_p = N_q$

and $\tilde{r}_p = -N_q$, we make the same assumption as in 2D, i.e. p is occluded by q. For orthographic projection $\tilde{r}_p = \tilde{r}_q$. For perspective projection, the difference between $\tilde{r}_p$ and $\tilde{r}_q$ are very small since q is intersected by r_p. Thus, we neglect this difference and assume in the following additionally if $\tilde{r}_q = N_q$ or $\tilde{r}_q = -N_q$ that q occludes p.

To determine if $\tilde{r}_p$ intersects the surface in q, we translate the tangent line at γ_q to p'. This line separates the viewing plane into half-planes and enables a similar approach as described for 2D to determine if q occludes p: if $p' + N'_q$ and q' are in the same half-plane, q occludes p. If they are in different half-planes, q does not occlude p. The distance d shown in Fig. 5(a) between p' and q' depends on the location of p and q in 3D and is limited since p and q belong to the same discrete viewing ray, which in turn is the rasterization of the projection ray. Consider the straight line that contains p and is perpendicular to the tangent line at γ_q. If d is neglected, we obtain on this line the same scenario as illustrated on the viewing plane of Fig. 4(c) and (d).

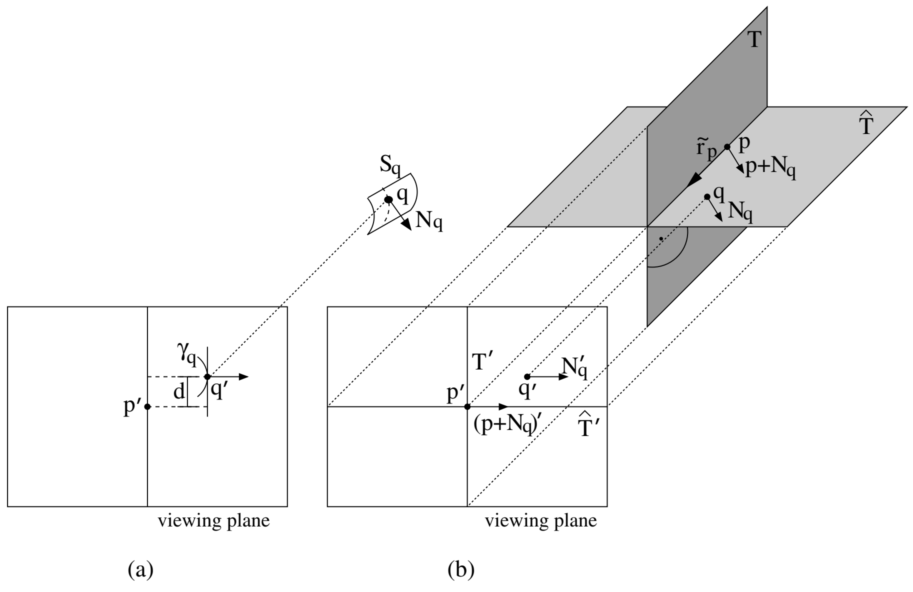

Fig. 5. Computation of the plane T

For practical applications, we perform the test described above directly in 3D. For this, a plane is needed in 3D that separates the space into half-spaces. Clearly the plane, denoted with T in the following, must contain $\tilde{r}_p$ and the projection T' of T must be perpendicular to N'_q. Then T can be computed as follows: determine a plane $\hat{T}$ containing the vectors $\tilde{r}_p$ and N_q, i.e. $\hat{T} : \tilde{r}_p \times N_q$. Then T is the plane perpendicular to $\hat{T}$ containing $\tilde{r}_p$. More formally: $(T \perp \hat{T}) \wedge (\tilde{r}_p \in T)$. This is illustrated in Fig. 5(b). The point q occludes p, if q and $p + N_q$ are in the same half-space or $q \in T$. If they are in different half-spaces, p is not occluded by q.

4 Experimental Results

Experimental results are shown below for the orthographic as well as the perspective projection. The normal vectors for the examples were computed employing the method introduced in [13]. The images in Fig. 6 show the results for two spheres using 26-connected viewing rays. The spheres were rastered in an array of 75^3 points. The occlusion was checked only for front facing points. The results shown in Fig. 6 were obtained by applying our method (upper row), by not allowing the discrete viewing rays to hit any object point (middle row) and, finally by allowing the discrete viewing rays to hit any surface point without any further restriction (lower row). Clearly for the second case, many points particularly along the outline of the objects are missed. This leads to a gap between the two spheres. In the last case, consider the part of the outline of the sphere in the foreground which occludes the other sphere: especially for perspective projection in the example, points of the occluded sphere were projected "into" the image of the occluding sphere. In contrast, this problem does not arise for our method. The lines near the outlines of the images in the upper and lower row of Fig. 6 are visible points which are very close to each other in the viewing plane such that these points are represented in the final image by neighbouring pixels. If more than one visible point would be represented by a pixel in the image, the point closest to the observer is viewed.

Figure 7 shows the necessity of the application of 6-connected discrete viewing rays, depending on the properties of the surface. The two cuboids were rastered in an array of 50^3 points. The problem arising for 26-connected rays along the left edge of the cuboid in the foreground corresponds with the situation illustrated in Fig. 2(a) by the point p: viewing rays of points of the occluded cuboid pass through the surface so that they are projected "into" the image of the occluding cuboid. This can be avoided by using 6-connected viewing rays instead.

5 Summary

The determination of the visibility of points is a major task of rendering. In this paper we have proposed a technique which determines the visibility of points on a surface in $\mathbb{Z}^3$, which is defined as the boundary of an object. The method introduced works in object space and employs the basic idea of discrete forward ray-casting. The visibility of each point is computed in $\mathbb{Z}^3$ considering a continuous mapping for the projection of the points onto the viewing plane. This assumption makes the results independent of the resolution of the final image and enables an application of the method, e.g., for discrete ray-tracing to solve the shadow problem. We have shown by examples, that our method is suitable to solve the visibility problem and thus can be employed for rendering techniques such as those proposed in [12].

A remaining problem of this work, that is matter of future work, is a proof that 26-connected rays are sufficient if the surfaces to be rendered satisfy certain conditions. This would lead to a performance increase by using 26-connected rays instead of 6-connected rays for our method.

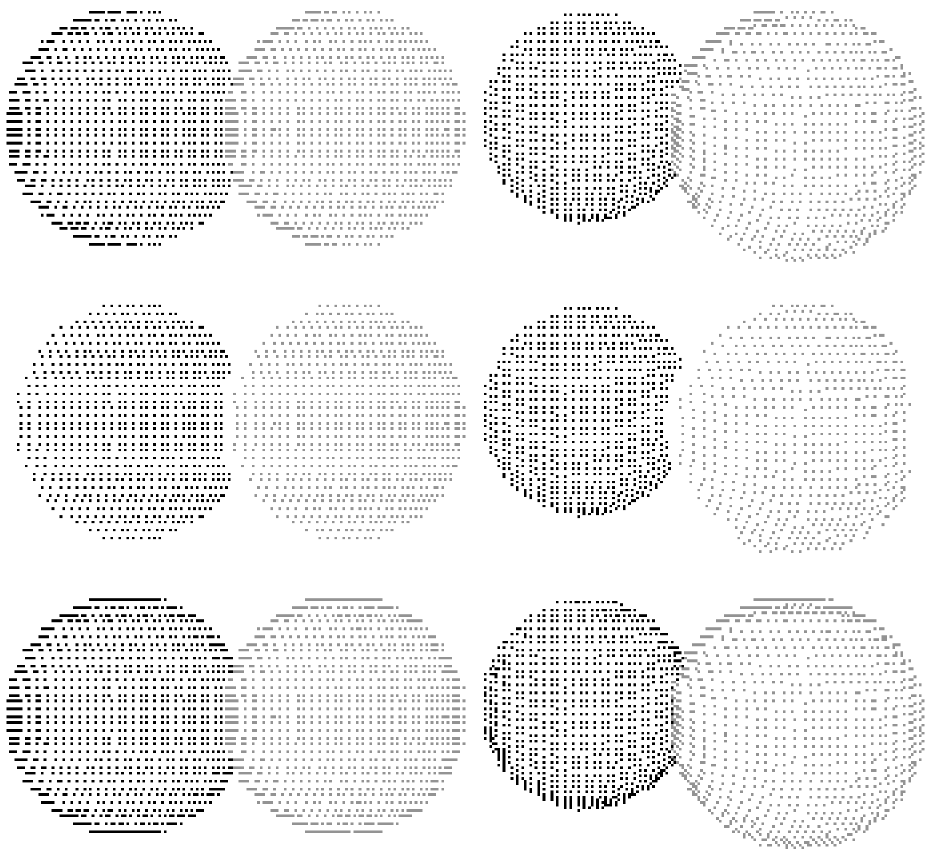

Fig. 6. Visible points of two spheres determined for orthographic (left column) and perspective projection (right column) by allowing discrete viewing rays to hit other surface points only if the projection rays do not intersect the object (upper row), not allowing intersections of any object point (middle row), and allowing intersections of any surface point (lower row) by the discrete viewing ray

References

1. Cohen, D., and Kaufman, A. Scan-conversion algorithms for linear and quadratic objects. In *Volume visualization*, A. Kaufman, Ed. IEEE Computer Society Press, 1991, pp. 280–301. 171
2. Delfosse, J., Hewitt, W. T., and Mériaux, M. An investigation of discrete ray-tracing. In *Proc. of the 4th Colloquium on Discrete Geometry for Computer Imagery (DGCI'94)* (Grenoble (France), 1994), pp. 65–74. 172, 174
3. Ke, H. R., and Chang, R. C. Sample buffer: A progessive refinement ray-casting algorithm for volume rendering. *Computers and Graphics 17*, 3 (1993), 277–283.

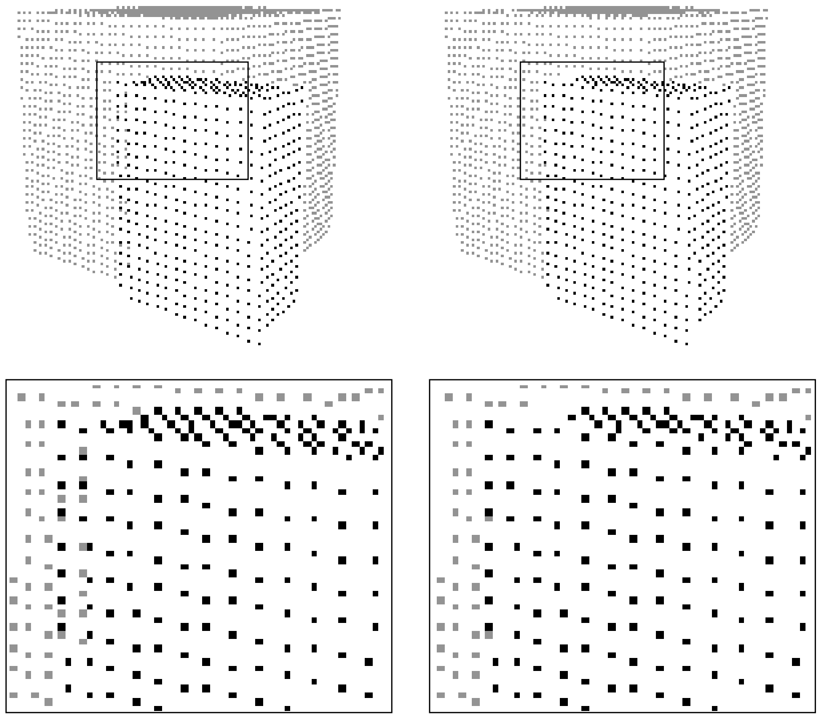

Fig. 7. Visible points obtained from 26-connected discrete viewing rays (left) and 6-connected viewing rays (right), and an enlargement of a detail (lower row) showing the upper left corner of the cuboid in the foreground

171

4. Laur, D., and Hanrahan, P. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics 25*, 4 (1991), 285–288. 172
5. Levoy, M. Display of surfaces from volume data. *IEEE Computer Graphics and Applications 8*, 5 (1988), 29–37. 171
6. Levoy, M. Efficient ray tracing of volume data. *ACM Transactions on Graphics 9*, 3 (1990), 245–261. 171
7. Malgouyres, R. A definition of surfaces of Z^3: A new 3D discrete Jordan theorem. *Theoretical Computer Science 186* (1997), 1–41. 175
8. Morgenthaler, D. G., and Rosenfeld, A. Surfaces in three-dimensional digital images. *Information and Control 51* (1981), 227–247. 175
9. Mueller, K., and Yagel, R. Fast perspective volume rendering with splatting by utilizing a ray-driven approach. In *Proc. of Visualization '96* (San Francisco, CA, 1996), pp. 65–72. 171
10. Nehlig, P., and Montani, C. A discrete template based plane casting algorithm for volume viewing. In *Proc. of the 5th Colloquium on Discrete Geometry for*

Computer Imagery (DGCI'95) (Clermont-Ferrand (France), 1995), pp. 71–81. 171

11. Shareef, N., and Yagel, R. Rapid previewing via volume-based solid modeling. In *Proc. of Solid Modeling '95* (1995), pp. 281–292. 171
12. Thürmer, G. Rendering surfaces of synthetic solid objects. *submitted for publication*. 172, 179
13. Thürmer, G., and Wüthrich, C. A. Normal computation for discrete surfaces in 3D space. *Computer Graphics Forum (Proc. Eurographics'97) 16*, 3 (1997), C15–C26. 173, 179
14. Upson, C., and Keeler, M. V-buffer: Visible volume rendering. *Computer Graphics (SIGGRAPH'88) 22*, 4 (1988), 59–64. 171
15. Wang, S. W., and Kaufman, A. E. Volume sampled voxelisation of geometric primitives. In *Proc. of Visualization '93* (1993), IEEE Computer Society Press, pp. 78–84. 171
16. Wang, S. W., and Kaufman, A. E. Volume-sampled 3D modeling. *IEEE Computer Graphics and Applications 14*, 5 (1994), 26–32. 171
17. Westover, L. Footprint evaluation for volume rendering. *Computer Graphics (SIGGRAPH'90) 24*, 4 (1990), 144–153. 172
18. Yagel, R., Cohen, D., and Kaufman, A. Discrete ray tracing. *IEEE Computer Graphics and Applications 12*, Sep. (1992), 19–28. 171, 175
19. Yagel, R., Stredney, D., Wiet, G. J., Schmalbrock, P., Rosenberg, L., Sessanna, D. J., and Kurzion, Y. Building a virtual environment for endoscopic sinus surgery simulation. *Computers and Graphics 20*, 6 (1996), 813–823. 171