Good illumination maps


Marta Fort
Universitat de Girona


J. Antoni Sellarès
Universitat de Girona

Good Illumination Maps

Narcís Coll* Marta Fort * Narcís Madern* J. Antoni Sellarès*

Abstract

Given a set P of points (lights) and a set S of segments (obstacles), the good illumination of a point q relative to P and S, describes the relationship between q and the distribution of the points in P from which q is illuminated taking into account the effect of the segments of S. A point q is t-well illuminated relatively to P and S if and only if every closed halfplane defined by a line through q contains at least t points of P illuminating q. The greater the number t the better the illumination of q. The good illumination depth of q is the maximum t such that q is t-well illuminated relatively to P and S. The good illumination map is the subdivision of the plane in good illumination regions where all points have the same fixed good illumination depth. In this paper we present algorithms for computing and efficiently drawing, using graphics hardware capabilities, the good illumination map of P and S.

1 Introduction

Given a set P of points, the location depth of a point q relative to P describes, intuitively, the relationship of q to the distribution of the points in P. A depth region is the locus of all points with the same fixed location depth. The *depth map* of P is the subdivision of the plane in depth regions.

The notion of illumination or visibility has been the main topic for a lot of different works in Computational Geometry. In this paper we use the *good illumination* concept based on Canales et. al [1, 4]. In some applications dealing with an environment of a set of point lights and a set of segment obstacles, it is not sufficient to have one point illuminated, is necessary to have several lights surrounding and illuminating the point. We will see that, in fact, the good illumination combines two well studied concepts: illumination with obstacles and location depth. In [4], Canales studied 1-good illumination when the lights are located in the exterior of a convex polygon and 2-good illumination when lights are located at the vertices of a simple (convex or non-convex) polygon.

In this paper we extend the study of good illumination to the general case of a set P of points (lights) and a set S of segments (obstacles). The good illumination map is the subdivision of the plane in regions whose points have the same good illumination relative to P and S. Drawing the good illumination map of P and S helps to visualize the distribution of the points of P relative to the segments of S. We present algorithms for computing and efficiently drawing, using graphics hardware capabilities, the good illumination map of P and S.

2 Depth Maps

Let P be a set of n points. The *location depth* of an arbitrary point q relative to P, denoted by $ld_P(q)$, is the minimum number of points of P lying in any closed halfplane defined by a line through q. The k-th depth region of P, represented by $dr_P(k)$, is the set of all points q with $ld_P(q) = k$. For $k \geq 1$, the external boundary $dc_P(k)$ of $dr_P(k)$ is the k-th depth contour of P. The *depth map* of P, denoted $dm(P)$, is the set of all depth regions of P. The complexity of $dm(P)$ is $O(n^2)$. This bound is tight, for example, when all points of P are in convex position. We denote $dm_r(P)$ the restriction of $dm(P)$ to a planar region r.

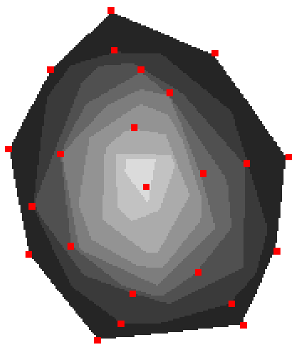

Figure 1: *Depth map example. The location depth of a point increases with the whiteness.*

2.1 Computing Depth Contours

Miller *et al* [8] present an algorithm for computing the depth contours for a set of points that makes an extensive use of duality, and proceeds as follows. Given a set P of points, it maps them to their dual

*Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, {coll,mfort,nmadern,sellares}@ima.udg.es. Partially supported by grant TIN2004-08065-C02-02

arrangement of lines. Then a topological sweep is applied to find the planar graph of the arrangement and its vertices are labeled with their levels (the number of dual lines above them). The depth of a vertex is $\min(\text{level}(v), n-\text{level}(v)+1)$. Finally, for a given k, $dc_P(k)$ is computed by finding the lower and upper convex hulls of the vertices at depth k. Each such vertex corresponds to a half-plane in the primal plane, and $dc_P(k)$ is the boundary of the intersection of these halfplanes (which might be empty, in that case $dc_P(k)$ does not exist). The complexity of this algorithm is $O(n^2)$ in time and space, that has been shown to be optimal.

Since for large n the $O(n^2)$ time of Miller *et al* algorithm could be prohibitive, in [7, 6] an algorithm is presented that draws, using graphics hardware capabilities, an image of the depth contours as a set of colored pixels, where the color of a pixel is its depth value. The algorithm consists of two steps: in the first step, the input point set P is scan-converted to lines in the dual image plane. The algorithm runs on two bounded duals due to the finite size of the dual plane, in order to guarantee that all intersection points of the lines lie in this finite region. Since each dual plane is discrete, it is possible to compute the level of each pixel by drawing the region situated above every dual line of P, incrementing by one the stencil buffer for each region. In the second step, the image of the dual lines is scanned, and for each pixel on a dual line the corresponding primal line at the appropriate depth is rendered as a colored 3D graphics primitive using the z-buffer. The depth of each primal line is easily determined from the stencil buffer value and the line color must be distinct for each depth. The resulting rendered image (see Figure 1) contains the depth contours of the point set P as the boundaries between colored regions. This method can be used also for drawing the convex hull of a set of points P. It is only necessary to change a bit the second step. When we scan the dual images, if a pixel has a level greater than zero we rasterize its primal line with depth one and using always the same color.

3 Good Illumination Maps

Let P be a set of n points and S be a set of m segments. We will assume that no point in P belongs to the interior of a segment in S. The free space F_S relative to S is the complement of S. Given two points $q \in F_S$ and $p \in P$, we say that point p illuminates q if the interior of the segment with endpoints p and q remains completely inside F_S. A point q is t-well illuminated relatively to P and S if and only if every closed halfplane defined by a line through q contains at least t points of P that illuminates q. The good illumination depth of q relative to P and S, denoted by $gid_{P,S}(q)$, is the maximum t such that q is t-well illuminated relatively to P and S.

Lemma 1 *If P_q denotes the subset of points of P illuminating q, then $gid_{P,S}(q) = ld_{P_q}(q)$.*

The k-th good illumination region relative to P and S, denoted $gir_{P,S}(k)$, is the set of all points q with $gid_{P,S}(q) = k$. Observe that $gir_{P,S}(k)$ can be formed by several convex connected components (see Figure 2).

Lemma 2 *If S is empty or does not intersect the convex hull $CH(P)$ then $gir_{P,S}(k) = dr_P(k)$.*

We call *good illumination map* of P and S, denoted $gim(P,S)$, the set of all good illumination regions relative to P and S. Also we denote $gim_r(P,S)$ the restriction of $gim(P,S)$ to region r.

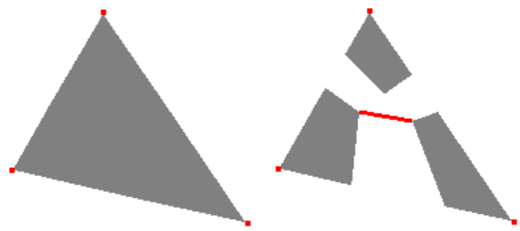

Figure 2: *In the left we have the good illumination map of a set with three points and an empty set of segments; the corresponding* 1*-good illumination region is represented in dark. In the right a segment obstacle has been added.*

4 Computing Good Illumination Maps

The Lemma 1 induces a way to compute $gim(P,S)$. First we decompose the free space F_S into illumination regions so that all points in a single connected such region are illuminated exactly by the same points in P. Then in each illumination region we compute the depth map of its illuminating points.

Denote s_0, s_1 the endpoints of the segment $s \in S$. Given a point $p \in P$ and a segment $s \in S$, the shadow region of s with respect to p, denoted $sr(p,s)$, is the set of points non illuminated from p when we consider segment s as an obstacle.

When $p \notin s$, $sr(p,s)$ is the region delimited by the segment s, the ray of origin s_0 and direction $\overrightarrow{ps_0}$ and the ray of origin s_1 and direction $\overrightarrow{ps_1}$. When p is an endpoint of s, for example s_0, the shadow region $sr(p,s)$ is the ray of origin p and direction $\overrightarrow{ps_1}$.

Let $\mathcal{A}(P,S)$ be the arrangement determined by the family of all shadow regions $sr(p,s)$ interior to $CH(P)$, for all $p \in P$, $s \in S$. All points in a cell c of $\mathcal{A}(P,S)$ are seen from exactly the same subset P_c of points of P. Observe that may exist two cells $c \neq c'$

that are seen from the same subset of points of P, it is to say with $P_c = P_{c'}$.

Theorem 3 *If n is the number of points of P and m is the number of segments of S, the arrangement $\mathcal{A}(P,S)$ has $O(n^2m^2)$ cells and each cell of $\mathcal{A}(P,S)$ has $O(n)$ illuminating points.*

Proof. Each shadow region is bounded by two rays and one segment, and the convex hull $CH(P)$ has $O(n)$ edges. Then, the arrangement $\mathcal{A}(P,S)$ has $O(nm)$ lines and $O((nm)^2)$ cells. By using a topological sweep [5], $\mathcal{A}(P,S)$ and the points illuminating each cell can be computed in $O((nm)^2)$ time. Figure 4 proves that this upper bound is tight. In a) we can see a segment placed in the diameter of a circle and $n/2$ light points p_i placed on the circle and above the segment. The point p_i is put so that one ray of its shadow region intersects $i-1$ rays of all other shadow regions inside the circle and the free space. Then, the number of cells of the line arrangement is $\Omega(\sum_{i=1}^{n/2}(i-1)) = \Omega(n^2)$. In b) the segment is split in m segments. Since we have the same properties of a) for each one of the m segments, the new line arrangement has $\Omega((nm)^2)$ cells. In c) we have placed $n/2$ light points on the circle and under the segments. This placement assures that there are $\Omega((nm)^2)$ cells interior to the $CH(P)$ that see a minimum of $n/2$ illuminating points. Consequently $O(n^2m^2)$ is a well fitted upper bound of the $\mathcal{A}(P,S)$, and $O(n)$ a well fitted upper bound of the illuminating points of each cell.

□

For each cell c of $\mathcal{A}(P,S)$ with illuminated points set P_c, the Lemma 1 assures that $gim_c(P,S)$ can be computed as $dm_c(P_c)$, the depth map of the set P_c restricted to c. Then, we have:

$$gim(P,S)) = \bigcup_{c\in\mathcal{A}(P,S)} dm_c(P_c)\,.$$

Theorem 4 *The good illumination map of P and S can be computed in $O(n^4m^2)$ time.*

Proof. Theorem 3 demonstrates that $\mathcal{A}(P,S)$ has $O(n^2m^2)$ cells. For each cell c, as we have seen in the previous sections, we compute $dm_c(P_c)$ by intersecting c with $dm(P_c)$. This spends a time of $O(n^2)$ (see section 2.1). Thus, the upper bound of the time needed to compute $gim(P,S)$ is $O((nm)^2n^2) = O(n^4m^2)$.

□

5 Drawing Good Illumination Maps

In this section we describe a method for drawing good illumination maps using GPU capabilities.

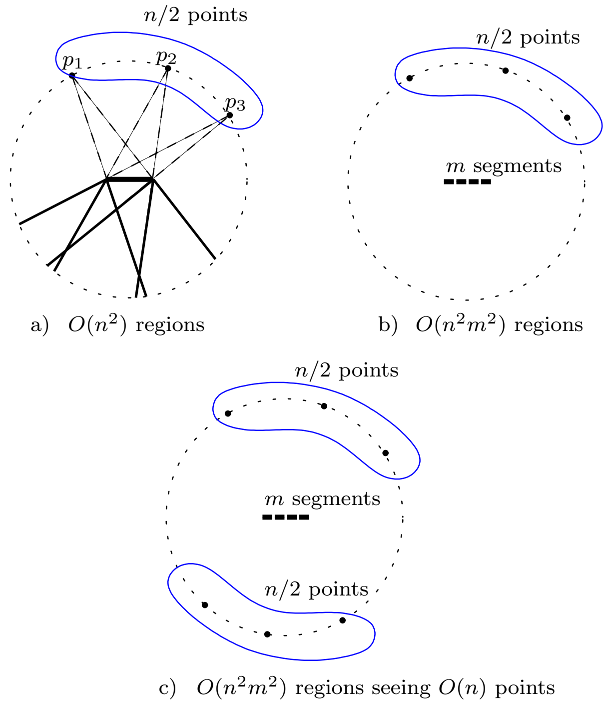

Figure 3: *Upper bound configuration.*

The method, based on the fact that $gim(P,S)) = \bigcup_{c\in\mathcal{A}(P,S)} dm_c(P_c)$, proceeds in two steps.

First step. We start drawing $CH(P)$ on a black screen, as described in Section 2.1, and we store it in a texture. Next we rasterize in white the boundary, interior to $CH(P)$, of all shadow regions $sr(p,s), p \in P, s \in S$ and we transfer the *frame buffer* to a matrix in the CPU so that each element represents a pixel. Then we find all the cells of $\mathcal{A}(P,S)$ using a CPU based growing method as follows. We take any black pixel of the matrix and we choose an unused color. Then we visit its four surrounding pixels and we paint each such a pixel with the current color. If the visited pixel is white (belongs to the boundary) we store it in a *waiting list* and we continue visiting and painting pixels until we have visited a entire cell. While there are pixels in the waiting list we take the first waiting pixel and we repeat the process from this position. In this way we *paint* each cell with a different color. During the process we store an interior pixel of each cell and its color. Finally, for each cell c we determine the set P_c of points of P illuminating c. To this purpose we take the interior pixel of c and we draw in white on a black screen the shadow regions defined by the pixel and the m segments of S. By doing this, a point p illuminates the cell c if its corresponding pixel is black. We use the *readPixels* function to obtain the set P_c by checking if the corresponding pixel of each one of the n points of P is colored black. Moreover, we assign a distinct color to each different subset P_c

so that all cells illuminated by P_c will get the same color. By doing this we ensure that we paint the same depth map at most once in the second step.

Second step. For each cell $c \in \mathcal{A}(P,S)$ we draw $dm_c(P_c)$ using the algorithm described in Section 2.1 that draws depth contours. In order to paint only the pixels inside c we use a fragment shader. The input of the fragment shader are the arrangement $\mathcal{A}(P,S)$ represented as a texture and the color assigned to P_c. The fragment shader only paints a pixel (x,y) if the color in the position (x,y) of the texture representing $\mathcal{A}(P,S)$ is equal to the color of c, since in this case the pixel is inside the cell c.

5.1 Results

We have implemented the proposed method using C++ and OpenGL, and all the tests and images have been carried out on a Intel(R) Pentium(R) D at 3GHz with 1Gb of RAM and a GeForce 7800 GTX/PCI-e/SSE2 graphics board.

Figures 2, 4 and 5 show some examples of good illumination maps obtained using our implementation.

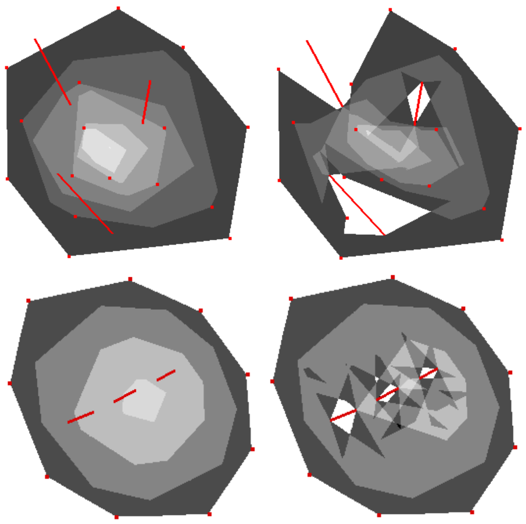

Figure 4: *Good illumination map examples. On the left we show the depth map of the light points. On the right we show the good illumination map of the light points and the obstacle segments.*

6 Future Work

We are extending our work on good illumination to the case of points modelling source lights of restricted illumination, for example emiting light within an angular region or/and with limited range.

We are also studying the possibility of developing a more efficient algorithm that, as in the case of depths maps, will work entirely in dual space.

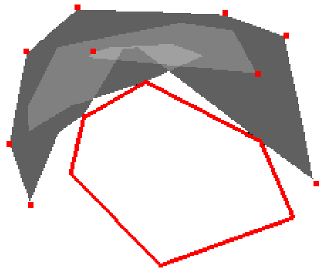

Figure 5: *Good illumination map with a convex obstacle polygon.*

References

[1] M. Abellanas, S. Canales and G. Hernández. Buena iluminación, *Actas de las IV Jornadas de Matemática Discreta y Algorítmica*, (2004), 239-246.

[2] M. Abellanas, A. Bajuelos, G. Hernández and I. Matos. Good Illumination with Limited Visibility. *Proc. International Conference of Numerical Analysis and Applied Mathematics*, Wiley-VCH Verlag, (2005), 35-38.

[3] M. Abellanas, A. Bajuelos, G. Hernández, I. Matos and B. Palop. Minimum Illumination Range Voronoi Diagrams. *Proc. 2nd International Symposium on Voronoi Diagrams in Science and Engineering*, (2005), 231-238.

[4] S. Canales. Métodos heurísticos en problemas geométricos, Visibilidad, iluminación y vigilancia. *Ph.D. thesis, Universidad Politécnica de Madrid*, 2004.

[5] H. Edelsbrunner and L. Guibas. Topological sweeping an arrangement. *J. Comput. System. Sci.* 38 (1989), 165194.

[6] Fischer I. and Gotsman C. Drawing Depth Contours with Graphics Hardware. Proceedings of Canadian Conf. on Comp. Geometry, (2006), 177–180.

[7] S. Krishnan, N. Mustafa, and S. Venkatasubramanian. Hardware-assisted computation of depth contours. *Proc. thirteenth ACM-SIAM symposium on Discrete algorithms*, (2002), 558–567.

[8] K. Miller, S. Ramaswami, P. Rousseeuw, J.A. Sellarès, D. Souvaine, I. Streinu and A. Struyf. Fast implementation of depth contours using topological sweep. *Statistics and Computing*, (2003), 13:153–162.