GPU-based Good Illumination Maps Visualization

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 [1, 3, 4]. 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 is an important topic in Computational Geometry. In some applications dealing with an environment of point lights and obstacles, several lights surrounding and illuminating points are needed. In this paper we use the *good illumination* concept introduced by Canales et. al [1, 3, 4]. We will see that, in fact, 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. We also extend the drawing algorithm to the case of restricted points, for example emitting light within an angular region or/and with limited range [2].

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 (see Figure 1). The complexity of $dm(P)$ is $O(n^2)$.

*Institut d'Informàtica i Aplicacions, Universitat de Girona, Spain, {coll,mfort,nmadern,sellares}@ima.udg.es. Partially supported by grant TIN2004-08065-C02-02. Marta Fort and Narcis Madern are also partially supported by grant AP2003-4305 and BES-2005-9541 respectively.

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.

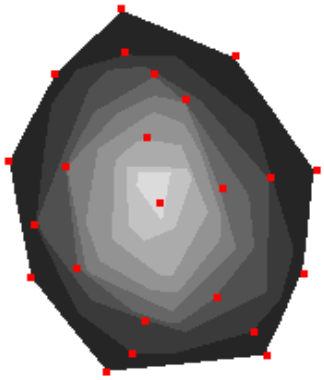


Figure 1: *Depth map of a set of points.*

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, the algorithm maps them to their dual 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 can be computed using $\min(\text{level}(v), n-\text{level}(v)+1)$. 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, that has been shown to be optimal, is $O(n^2)$ in time and space.

Since for large n the $O(n^2)$ time of Miller *et al* algorithm could be too large, in [6, 7] 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 two images formed by all the dual lines are 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 also be used for drawing the convex hull of a set of points P by introducing minor changes in 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 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 illuminate 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 3.1. *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 3.2. *If S is empty or does not intersect the convex hull $CH(P)$ then $gir_{P,S}(k) = dr_P(k)$.*

We call the set of all good illumination regions relative to P and S the good illumination map of P and S and denote it with $gim(P, S)$.

Also we denote with $gim_r(P, S)$ the restriction of $gim(P, S)$ to region r.

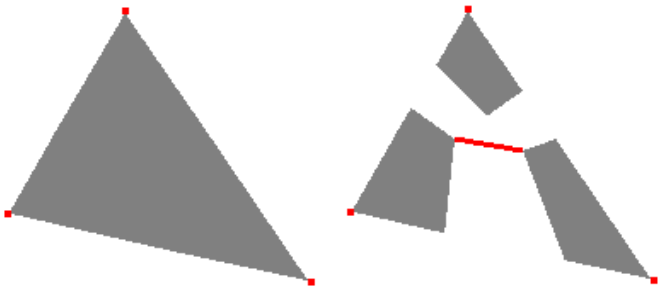


Figure 2: *In the left we have the good illumination map of a set with three points and an empty set of segments; the corresponding* 1*-good illumination region is represented in dark. In the right a segment obstacle has been added.*

4 Computing Good Illumination Maps

From now on we will focus on the non trivial case where $n \geq 3$, $m \geq 1$ and S intersects $CH(P)$.

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.

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. Denote s_0, s_1 the endpoints of s. 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 $p \in P$ and $s \in S$. All cells in $\mathcal{A}(P, S)$ are convex and 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 two cells $c \neq c'$ that are seen from the same subset of points of P may exist, it is to say with $P_c = P_{c'}$.

Theorem 4.1. *The arrangement $\mathcal{A}(P, S)$ consist of $O(n^2m^2)$ cells and each cell 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. Figure 3 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.

□

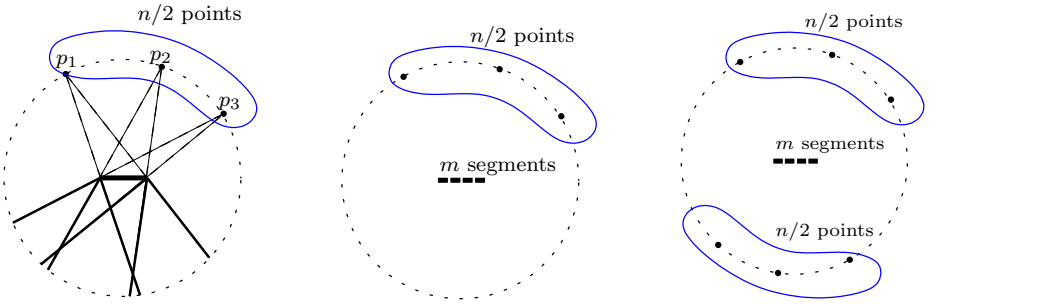
For each cell c of $\mathcal{A}(P,S)$ with illuminated points set P_c, Lemma 1 states that $gim_c(P,S)$ can be computed as $dm_c(P_c)$, the depth map of the set P_c restricted to c. Then, we have:

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

Theorem 4.2. *The good illumination map of P and S can be computed in $O(n^4m^2)$ time.*

Proof. We associate a set P_c to each cell c of $\mathcal{A}(P,S)$. First, by using a topological sweep [5, 9], $\mathcal{A}(P,S)$ and the associated sets P_c can be computed in $O(n^3m^2)$ time. Next, for each cell c we compute $dm_c(P_c)$ by intersecting the convex cell c with the depth contours determined by $dm(P_c)$. This spends a time of $O(n^2)$ per cell (see section 2.1). Thus, the time needed to compute $gim(P,S)$ is $O((nm)^2n^2) = O(n^4m^2)$.

□



a) $O(n^2)$ regions b) $O(n^2m^2)$ regions c) $O(n^2m^2)$ regions seeing $O(n)$ points

Figure 3: *Worst case $\mathcal{A}(P,S)$ configuration.*

5 Drawing Good Illumination Maps

In this section we describe a method for drawing good illumination maps using GPU capabilities. 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 the boundary, interior to $CH(P)$, of all shadow regions $sr(p,s)$, $p \in P$, $s \in S$ in white 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. We paint this pixel with the current color and then we visit its four surrounding pixels. If the visited pixel is white (belongs to the boundary) we store its neighbors in a waiting list and we continue visiting and painting pixels until we have visited a entire cell. While there are pixels in the list, we take the first pixel and if it is black we

repeat the process from this position, otherwise the pixel is rejected. By doing this 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 by taking the interior pixel of c and drawing the shadow regions defined by the pixel and the m segments of S in white on a black screen. By doing this, a point p illuminates the cell c if its corresponding pixel is black. We obtain the set P_c by checking if the pixel corresponding to each point in 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 inputs of this 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 Extension to the case of restricted illumination

We consider that a point p is *restricted* when is only emitting light within an angular region or/and with limited range [2]. We denote $rr(p)$ the *restricted region* illuminated by p when any segment obstacle is considered.

In order to draw good illumination maps with restricted points it is only necessary to modify the arrangement $\mathcal{A}(P, S)$ and the computation of the shadow regions. Besides of painting the lines of all shadow regions of $\mathcal{A}(P, S)$, we must also draw the contours of the restricted illumination regions associated to each point in P. We can now find the cells in $\mathcal{A}(P, S)$ just as it is described previously in this Section. Finally we have to change the way to know if a point in P illuminates each cell c. For each point $p \in P$ we paint all shadow regions $sr(p, S)$ and the exterior of $rr(p)$ in white over a black screen and we read one pixel from each cell c. If the read pixel is white, it means that c is not illuminated by p. In Figure 8 we can see two examples of Good Illumination Maps with restricted point lights.

5.2 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 2GB of RAM and a GeForce 7800 GTX/PCI-e/SSE2 graphics card using a screen resolution of 500x500 pixels.

In Figures 5 and 4 we can observe the time needed by our algorithm in some particular cases. Figure 4 shows the time spent by our algorithm by increasing the number of points for several number of segment obstacles. Each one of the lines represents the time for a fixed number of segments. Points and segments are placed using the worst case configuration (see Figure 4). Figure 5 shows the time spent by the algorithm in a configuration of points and segments placed randomly. Observe that in this case the running time increases slower than in the worst case configuration.

Figures 2, 6 and 7 show some examples of good illumination maps obtained using our implementation. In these figures the points are colored in a grey gradation according to its good illumination depth (black corresponds to level one), however pure white color shows level zero.

6 Future Work

We are improving the implementation of the approximated algorithm in order to obtain a better performance.

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

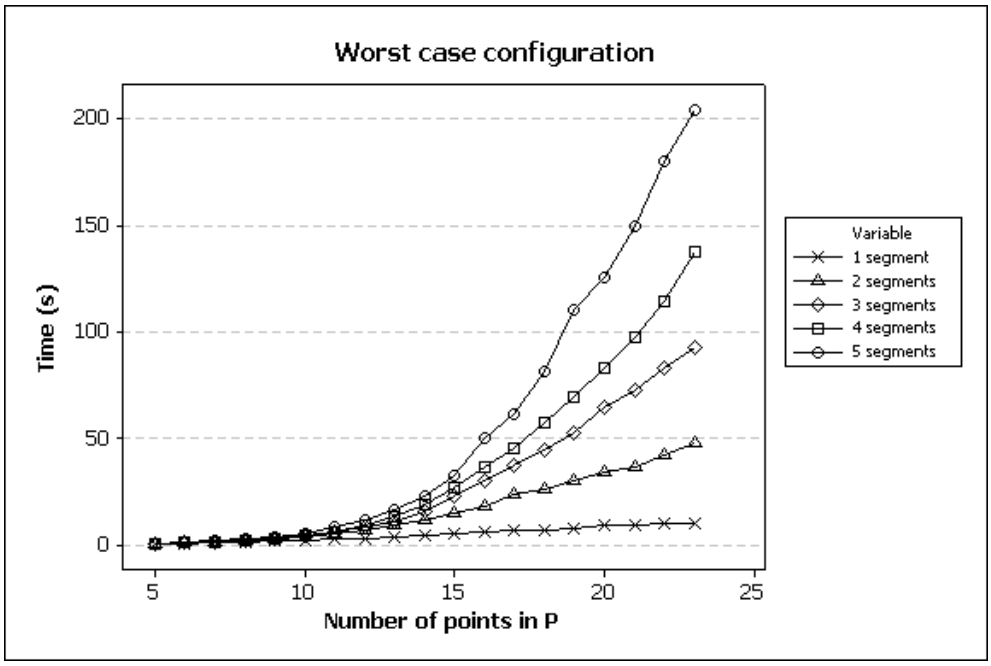


Figure 4: *Time spent by our algorithm by increasing the number of points for several number of segment obstacles. Points and segments are placed using the worst case configuration (see Figure 4).*

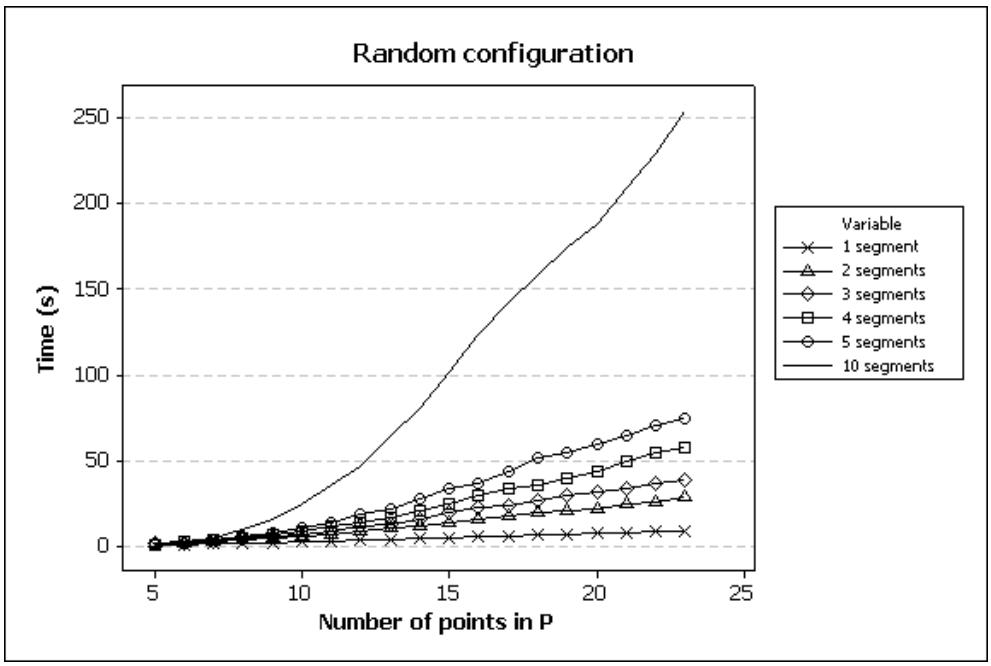


Figure 5: *Time expended by our algorithm by increasing the number of points for several number of segment obstacles. Points and segments are placed randomly.*

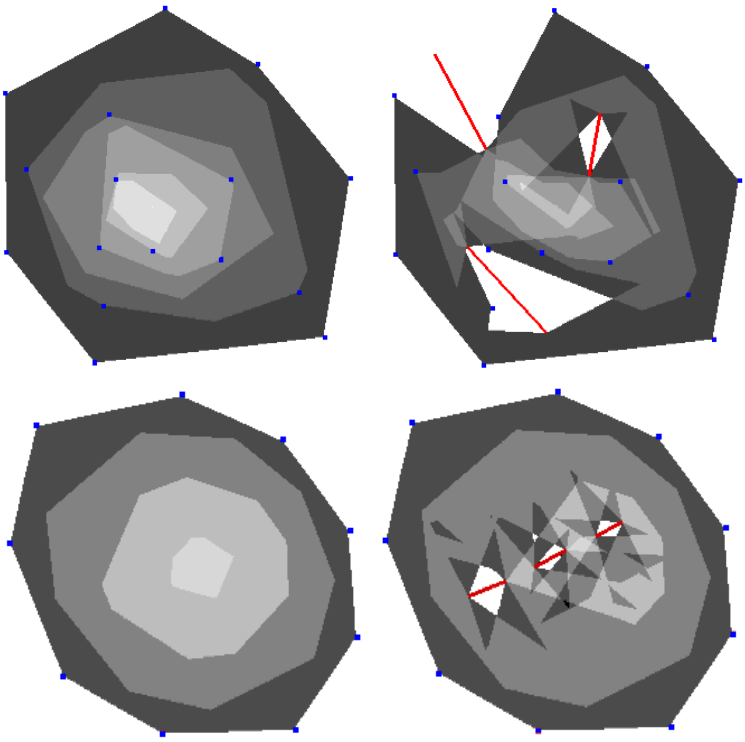


Figure 6: *Good illumination maps of two different configurations of points and segments. On the left we show the depth maps of the points. On the right we show the good illumination maps of the points and the segments.*

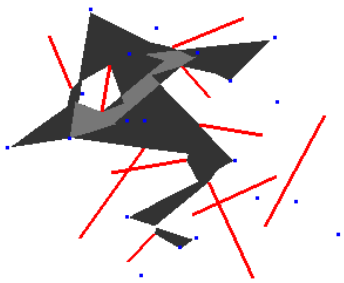


Figure 7: *Good illumination map from a set of twenty light points and a set of eleven segment obstacles.*

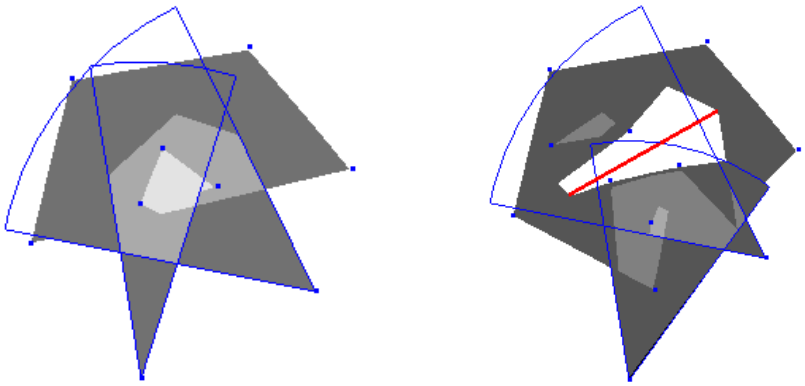


Figure 8: *Good illumination map examples from a set with restricted point lights.*

We want to extend our method to obtain good illumination maps in polyhedral terrains from a set of points or restricted lights. In this case the obstacles would be the faces of the terrain and the point lights would be placed on or over these faces.

An other future work is the study and implementation of the idea in a 3-dimensional space, where a point light could be placed in any position of the space and the obstacles could be triangles.

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 and I. Matos. Good Θ-illumination of Points. *Proc. 23rd European Workshop on Computational Geometry*, (2007), 61–64.

[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), 165–194.

[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.

[9] E. Rafalin, D. Souvaine, I. Streinu. Topological Sweep in Degenerate cases. *Proc. of the 4th international workshop on Algorithm Engineering and Experiments*, ALENEX 02, in LNCS 2409, Springer-Verlag, Berlin, Germany, (2002), 155–156.