Available online at www.sciencedirect.com
SCIENCE DIRECT®
J. Vis. Commun. Image R. 16 (2005) 233–249
JOURNAL OF VISUAL Communication & IMAGE Representation
www.elsevier.com/locate/jvci

Optimal projections onto grids and finite resolution images

J.M. Díaz-Báñez[a], F. Hurtado[b], M.A. López[c], J.A. Sellarès[d,*]

[a] *Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain*
[b] *Departament de Matemática Aplicada II, U.P.C, Barcelona, Spain*
[c] *Department of Computer Science, University of Denver, Denver, CO, USA*
[d] *Institut d'Informàtica i Aplicacions, Universitat de Girona, Girona, Spain*

Received 3 October 2002; accepted 4 November 2004
Available online 12 January 2005

Abstract

The problem of determining nice (regular, simple, minimum crossing, and monotonic) and non-degenerate (with distinct x-coordinate, non-collinear, non-cocircular, and non-parallel) orthogonal and perspective images of a set of points or a set of disjoint line segments has been studied extensively in the literature for the theoretical case of infinite resolution images [J. Vis. Commun. Image Represent. 10 (2) (1999) 155; Int. J. Math. Algorithms 2 (2001) 227; J. Vis. Commun. Image Represent. 12 (4) (2001) 387; J. Vis. Commun. Image Represent.]. In this paper we propose to extend the study of this type of problems to the case where the images have finite resolution. Applications dealing with such images are common in practice, in fields such as computer graphics and computer vision. We derive algorithms that solve three related problems, both exactly and approximately. Given a set $\mathcal{P}$ in the plane or in the space, find a graduated line, a line partitioned into unit length intervals, so that the maximum number of points of $\mathcal{P}$ that are projected into a single interval is minimized. In space we also study the variant where we want to project $\mathcal{P}$ onto a graduated plane, a plane partitioned into unit squares.

Keywords: Projections; Finite resolution images; Optimal images

* Corresponding author.
E-mail address: sellares@ima.udg.es (J.A. Sellarès).

1047-3203/$ - see front matter
doi:10.1016/j.jvcir.2004.11.001

1. Introduction

An object in two-dimensional and three-dimensional space is often represented by a set of points and line segments that act as its features. An optimal projection of an object is one that gives an image in which the features of the object, relevant for some task, are visible without ambiguity so that the image is as simple and readable as possible: no two points are projected to the same point, the projection of a segment is not reduced to a point, the number of crossings of the projection of a set of segments is minimum, no two projected points have the same x-coordinate, no three projected points are collinear, no four projected points are cocircular, no two projected segments are parallel, etc. The terms *nice* and *non-degenerate projections* refer to these requirements and to many others. The problem of determining nice and non-degenerate orthogonal and perspective projections of sets of points and disjoint line segments has been studied extensively in the literature for the theoretical case of infinite resolution images [3,6–8].

In the finite resolution model a d-dimensional point projects onto a cell (pixel) of a regular grid (all cells are disjoint but identical k-dimensional hypercubes, for $k < d$). For clarity, when the dimensionality of the grid is important, we explicitly refer to the grid as a *graduated line* or *graduated plane*, for example.

In this paper we consider orthogonal projections only. Furthermore, without loss of generality, we assume unit size grid cells (hypercubes of side length equal to one). Other cell sizes can be reduced to unit cells by scaling the input set about the origin. We study the problem of finding orthogonal projections that minimize the maximum number of points that project to the same grid cell. Specifically, we present both exact and approximation algorithms for the following problems:

1. Given a set of points in $\mathbb{R}^2$, determine a graduated line ℓ in $\mathbb{R}^2$ such that the maximum number of input points that project to the same cell of ℓ is minimum (or within a small factor of the minimum).
2. Given a set of points in $\mathbb{R}^3$, determine a graduated line ℓ or graduated plane π in $\mathbb{R}^3$ such that the maximum number of input points that project to the same cell of ℓ or π is minimum (or within a small factor of the minimum).

We make no assumptions about the precision of the input points. This precision can be lower or higher (in principle, even infinite) than the precision of the output projections, as our approach does not depend on the precision of the data. The latter could well change, for instance, due to improved methods of data acquisition, using higher resolution data types, etc., while the resolution of the output images remains the same.

Definition 1.1. Let d and k be positive integers such that $k < d$. An instance of the $\mathrm{P}(d,k)$ problem is a set of points $\mathcal{P}$ in $\mathbb{R}^d$. A solution to this instance is a k-dimensional graduated hyper-plane π such that the maximum number m of points of $\mathcal{P}$ projected to the same cell of π is minimal. We refer to m as the cost of the solution.

Other optimization problems related to projections of a point set onto a graduated line or graduated plane and depending on the number of points projected in each cell can be solved with the same approach that we use to solve problems 1 and 2.

In [1,2] a similar, but different problem is studied. Given a set of points in the plane the problem is to find an optimal set of b equally spaced parallel lines such that the maximum number of points in a region bounded by two consecutive lines is minimized. In [4] a regular grid is considered. Notice that for these problems, unlike ours, the distance between adjacent parallel lines is not fixed beforehand.

2. Two-dimensional case

The graduated line $\ell = \ell(\theta, u, v)$ is the line obtained from the x-axis by composing a rotation by angle $\theta \in [0, \pi)$ around the origin o with a translation by vector (u, v). The point (u, v) is called the origin of the graduated line (see Fig. 1). Each number $z \in \mathbb{Z}$ defines a cell $c_z = [l_z, r_z)$ in ℓ that is obtained from cell $[z, z+1)$ of the x-axis by the composite transformation described above.

In the finite resolution model the projection of a point p onto the line ℓ is the cell c of ℓ that contains the intersection p^* between the line ℓ and the line through p orthogonal to ℓ (Fig. 2).

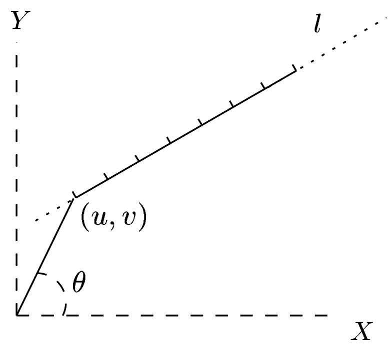

Fig. 1. Graduated line.

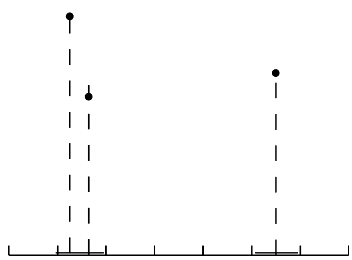
Fig. 2. Projections in the finite resolution model.

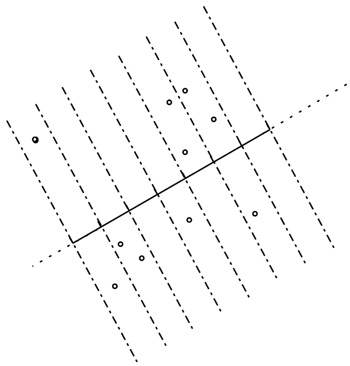

Fig. 3. Buckets.

A region of the plane bounded by two lines orthogonal to ℓ through consecutive cell boundaries is called a *bucket* and the participating lines are the *bucket boundaries* (see Fig. 3). Equivalently, a bucket is the locus of points in the plane that project to the same cell.

Remark 2.1. Let ℓ be a graduated line with unit normal n and unit direction vector d. Let ℓ' be the graduated line obtained by translating ℓ by $zd + rn$, for any $z \in \mathbb{Z}$ and $r \in \mathbb{R}$. Then lines ℓ and ℓ' determine the same bucket sets.

Definition 2.1. Two graduated lines whose bucket sets produce exactly the same partition of a point set $\mathcal{P}$ are said to be equivalent with respect to $\mathcal{P}$.

A solution to an instance $\mathcal{P}$ of $\mathrm{P}(2,1)$ is a graduated line ℓ that minimizes the maximum number of points of $\mathcal{P}$ in a bucket determined by ℓ. Clearly, if ℓ is an optimal solution so is any graduated line equivalent to ℓ.

Lemma 2.1. *Let ℓ be a solution to an instance $\mathcal{P}$ of* $\mathrm{P}(2,1)$. *There exists a graduated line ℓ_0 equivalent to ℓ such that a point of $\mathcal{P}$ lies on the boundary of a bucket of ℓ_0.*

Proof. Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be a point set. For $1 \leqslant i \leqslant n$ let p_i^* denote the projection of p_i onto ℓ, and d_i, the distance from p_i^* to the left endpoint l_i of the cell containing p_i^*. Suppose that $d_j = \min_{1 \leqslant i \leqslant n}\{d_i\}$. Then the translation that moves point l_j to point p_j transforms line ℓ to the desired line ℓ_0. $\square$

Our algorithm to solve $\mathrm{P}(2,1)$ is based on two key ideas:

(1) As a consequence of Lemma 2.1, we only need to consider graduated lines that have a bucket boundary passing through a point p_i of $\mathcal{P}$. Furthermore, from Remark 2.1, we can assume that p_i is the origin of the graduated line.
(2) Let ℓ be a graduated line with origin p_i. Every rotation of ℓ by an angle θ about its origin produces a partition of $\mathcal{P}$ into buckets. Since $\ell = \ell(\theta, u, v)$ and $\ell = \ell(\pi + \theta, u, v)$ produce equivalent partitions, it suffices to consider angles in $[0, \pi)$. A radial sweep of ℓ about p_i produces a finite number of different partitions of $\mathcal{P}$. As ℓ rotates from $\theta = 0$ to π, the current partition changes only at angles where the left boundary line of a bucket

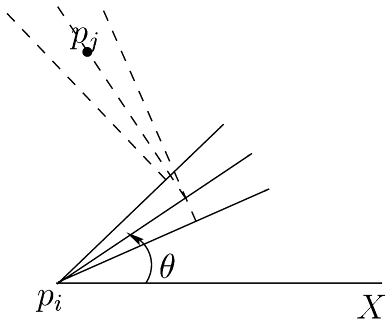

Fig. 4. For angle θ point p_j changes of bucket.

passes through a point p_j of $\mathcal{P}$, $p_j \neq p_i$. At this instant, p_j changes from one bucket to the adjacent one (see Fig. 4). Consequently, in searching for the optimal solution for a fixed origin, it suffices to examine the finite collection $\mathcal{A}$ of these angles.

The algorithm, which we call **MGL2**, proceeds in two steps:

(1) For each point p_i of $\mathcal{P}$ perform a radial sweep around p_i to find the graduated line ℓ_i with origin p_i that minimizes the maximum number of points of $\mathcal{P}$ that fall in the same bucket. Call this minimum number m_i.
(2) Report the line ℓ that corresponds to $\min\{m_1, \ldots, m_n\}$.

We now elaborate on Step 1. Consider the iteration with the sweep centered at p_i. Without loss of generality assume that $p_i = o = (0,0)$ as, otherwise, a simple translation of $\mathcal{P}$ makes this true. Let $C(p,r)$ denote the circle of radius r centered at p, and $\ell(k,\theta)$, the line obtained by rotating the boundary line $x = k$ about o by angle θ. We wish to determine the value(s) of θ for which an input point p_j lies on $\ell(k,\theta)$. Each such value corresponds to an event of the sweep. Note that if $d(o,p_j) < k$ the rotated line cannot possibly contain p_j. Otherwise, the above equation has one or two solutions, depending on whether p_j lies on the boundary or outside, respectively, of the circle $C(o,k)$. In fact, the solutions correspond to the lines through p_j tangent to the circle. The points of tangency (hence the values of θ) can be found as the intersection of the circle $C(p_i,k)$ with circle $C(m,\frac{1}{2}\,\mathrm{dist}\,(p_i,p_j))$ where m is the midpoint of segment p_ip_j, as illustrated in Fig. 5. Note that we may substitute θ by $\theta - \pi$ if $\theta \geqslant \pi$ as the sweep is performed in the range $[0,\pi)$.

This procedure, which yields at most two events, is repeated for integer values of k that satisfy $|k| \leqslant \mathrm{dist}(p_i,p_j)$. Once all events have been collected they are processed in ascending order by θ. Before the first event, the "status" of the sweep is initialized by computing the number of points that belong to each bucket for a graduated line with $\theta = 0$ and the largest such count is identified. Each event modifies the status by updating these bucket counts and computing the largest count. The minimum m_i of these counts corresponds to the best graduated line ℓ_i centered at p_i. The complete algorithm is described in Fig. 6.

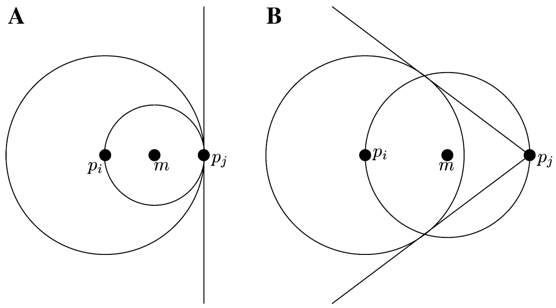

Fig. 5. Tangent lines between an input point p_j and a circle correspond to boundary lines which, when rotated, contain p_j; (A) one solution and (B) two solutions.

The complexity of **MGL2** is dominated by the sorting step of line 13. Each execution of this step requires $O(|\mathcal{A}| \log |\mathcal{A}|)$ time and $O(|\mathcal{A}|)$ space. If $t_i = \max_{1 \leqslant j \leqslant n} \{\text{dist}(p_i, p_j)\}$, then $|\mathcal{A}| = O(t_i n)$ and the ith iteration of the algorithm requires $O(t_i n \log t_i n)$ time. Thus, the total execution time is $O(tn^2 \log tn)$, where $t = \max\{t_i\}$ is the diameter of $\mathcal{P}$. The space required is $O(tn)$. We have proven the following:

Theorem 2.1. *An instance $\mathcal{P}$ of problem* $P(2,1)$ *can be solved in* $O(tn^2 \log tn)$ time and $O(tn)$ *space, where t is the diameter of $\mathcal{P}$.*

algorithm MGL2($\mathcal{P}$)
1 **for** $i \leftarrow 1$ **to** n **do**
2 $\quad\mathcal{A} \leftarrow \emptyset$
3 $\quad$set all bucket counts to 0
4 $\quad$**for** $j \leftarrow 1$ **to** n **do**
5 $\qquad q_j \leftarrow p_j - p_i$
6 $\qquad$increment the count of bucket $\lfloor x(q_j) \rfloor$ by 1
7 $\qquad d_j \leftarrow \text{dist}(p_i, p_j)$
8 $\qquad$**for** $k \leftarrow 0$ **to** $\lfloor d_j \rfloor$ **do**
9 $\qquad\quad$**for each** point of tangency t between q_j and circle $C(o, k)$
10 $\qquad\qquad \theta \leftarrow \text{angle}(t)$
11 $\qquad\qquad$if $\theta \geq \pi$ **then** $\theta \leftarrow \theta - \pi$
12 $\qquad\qquad \mathcal{A} \leftarrow \mathcal{A} \cup \{\theta\}$
13 $\quad$Sort $\mathcal{A}$ in ascending order
14 $\quad m_i \leftarrow$ maximum number of points in a bucket of $\ell(0, p_i)$
15 $\quad$**for each** event θ of $\mathcal{A}$ **do**
16 $\qquad$compute bucket count of $\ell(\theta, p_i)$
17 $\qquad m \leftarrow$ maximum number of points in a bucket of $\ell(\theta, p_i)$
18 $\qquad$**if** $m < m_i$ **then** $m_i \leftarrow m$
19 report $\min\{m_1, \ldots, m_n\}$
end

Fig. 6. Algorithm for solving an instance of $P(2,1)$. A bucket is denoted by the x-coordinate of its left boundary line before rotation.

2.1. Approximate solutions to P(2, 1)

In this section we describe approximation algorithms that compute answers that are guaranteed to be within a small constant factor of the optimal.

When the origin (u, v) of the graduated line is fixed, and the only degree of freedom is the rotation angle θ, we obtain a restricted instance AP(2, 1) of problem P(2, 1). When the angle θ of the projection line is fixed, but we are allowed to shift the projection line over itself, we obtain a restricted instance SP(2, 1) of problem P(2, 1). It is natural to consider solutions to the restricted instances AP(2, 1) and SP(2, 1) as approximations to the solution of P(2, 1).

The solution to problem AP(2, 1) can be obtained by performing one iteration of Step 1 of **MGL2**, using $p = (u, v)$ (instead of p_i) as the designated origin.

Theorem 2.2. *An instance of problem* AP(2, 1) *can be solved in* $O(tn \log tn)$ time and $O(tn)$ *space, where* $t = \max_{1 \leqslant j \leqslant n}\{\mathrm{dist}(p, p_j)\}$.

Let us consider problem SP(2, 1), whose solution is simplified by the following simple observation:

Remark 2.2. Shifting a graduated line ℓ by s is equivalent to shifting it by $s \bmod 1$.

Remark 2.2 allows us to concentrate on positive shifts smaller than 1 (the pixel size) when solving SP(2, 1). A simple algorithm is obtained as follows. Without loss of generality assume that $\theta = 0$, i.e., that we are projecting onto the x-axis. We can start with an arbitrary graduated line ℓ_0, say, $\ell_0 = \ell(0, 0, 0)$. Let x_i^* denote the x-coordinate of the projection p_i^* of p_i and $\Delta_i = x_i^* \bmod 1$. In other words, Δ_i is the distance from p_i^* to the left boundary of its pixel on l_0. For notational convenience, we define $\Delta_0 = 0$ and $\Delta_{n+1} = 1$. Sort $\langle \Delta_1, \ldots, \Delta_n \rangle$ in ascending order and relabel the values so that $i < j \Rightarrow \Delta_i \leqslant \Delta_j$. Now let $\ell_i = \ell(0, \Delta_i, 0)$, i.e., the line obtained from ℓ_0 by shifting it Δ_i units to the right. In turn and in order, compute the cost of ℓ_1, then the cost of ℓ_2, and so on to ℓ_n, keeping track of the minimum which is reported at the end.

Remark 2.3. Let $\Delta_i \leqslant \delta < \Delta_{i+1}$, $0 \leqslant i \leqslant n$. Then, line $\ell(0, \delta, 0)$ is equivalent to line $\ell(0, \Delta_i, 0)$.

This remark implies that if we gradually and continuously shift ℓ_0 to the right, changes to the partition into buckets of P occur only when shifting by some Δ_i. In other words, it suffices to consider shifts $\Delta_1, \ldots, \Delta_n$ when looking for an optimal answer. This, in turn, implies that the algorithm is correct.

Remark 2.4. The cost of $\ell(0, \Delta_{i+1}, 0)$ can be computed from the cost of $\ell(0, \Delta_i, 0)$ in $O(1)$ time.

If no two Δ_i's are the same, only one point changes buckets when comparing the partition of $\ell(0, \Delta_{i+1}, 0)$ with that of $\ell(0, \Delta_i, 0)$. Otherwise, the cost of computing $\ell(0, \Delta_i, 0)$ can be easily amortized among multiple shifts with the same value to yield the required result. This remark implies that the algorithm is efficient. The most expensive step is sorting the candidate shifts.

We have proven the following:

Theorem 2.3. *An instance of problem* $\mathrm{SP}(2,1)$ *can be solved in* $\mathrm{O}(n\log n)$ *time.*

An interesting property of problem $\mathrm{SP}(2,1)$ is that the costs of any two candidate graduated lines cannot differ by more than a factor of 2.

Lemma 2.2. *Let* ℓ *be a projection with cost* C *and let* C' *be the cost of a projection* ℓ' *obtained by shifting* ℓ *by an arbitrary amount. Then,* $\frac{1}{2} \leqslant \frac{C'}{C} \leqslant 2$.

Proof. It suffices to concentrate on positive shifts smaller than the pixel size. This means that, after an arbitrary shift, the worst thing that can happen is that the contents of two buckets are combined into one, thus at most doubling the cost of the more expensive of the two. Since no bucket contains more than C points, we have $C' \leqslant 2C$. Conversely, the cost of a bucket could be reduced by shifting. For some $0 \leqslant i \leqslant j$, a bucket with j points will contribute, after shifting, i and $j-i$ points, respectively, to two neighboring buckets. This implies that for some $0 \leqslant i \leqslant C$, $C' \geqslant \max(i, C-i) \geqslant \lceil C/2 \rceil$ which completes the proof. □

An interpretation of this lemma suggests that in solving $\mathrm{P}(2,1)$, it is more important to find a good angle than it is to find a good shift. We now prove:

Theorem 2.4. *Let* C *be the cost of an optimal solution to an instance* $\mathcal{P}$ *of* $\mathrm{P}(2,1)$ *and let* t *denote the diameter of* $\mathcal{P}$. *There is an algorithm that runs in* $\mathrm{O}(tn\log tn)$ *time that finds an answer with cost no bigger than* $2C$.

Proof. Let $\ell = \ell(\theta, u, v)$ be an optimal solution to $\mathrm{P}(2,1)$. Let $\ell_i = \ell(\alpha_i, x_i, y_i)$ be the best solution found during the ith iteration of Step 1 of algorithm **MGL2**, i.e., while using (x_i, y_i) as the pivot point. Clearly, $\mathrm{cost}(\ell_i) \leqslant \mathrm{cost}(\ell(\theta, x_i, y_i))$ as ℓ_i is optimal for pivot (x_i, y_i). Furthermore, for some $s \geqslant 0$, there is a line $\ell' = \ell(\theta, x_i + s\cos\theta, y_i + s\sin\theta)$ which is equivalent to the optimal ℓ. Since the supporting lines of $\ell(\theta, x_i, y_i)$ and ℓ' are one and the same, Lemma 2.2 implies that $\mathrm{cost}(\ell(\theta, x_i, y_i)) \leqslant 2\mathrm{cost}(\ell')$. Thus, we have $\mathrm{cost}(\ell_i) \leqslant \mathrm{cost}(\ell(\theta, x_i, y_i)) \leqslant 2\mathrm{cost}(\ell') = 2\mathrm{cost}(\ell)$. □

Notice that most of the useful work of **MGL2** is already done during the first iteration! Later iterations can only yield small improvements, if any.

3. Three-dimensional case

3.1. Graduated lines

The graduated line $\ell = \ell(\theta, \varphi, u, v, w)$ is the line obtained from the x-axis by composing a rotation by angle $\pi - \varphi$, $\varphi \in [-\pi/2, \pi/2]$ around the y-axis, a rotation by angle θ, $\theta \in [0, 2\pi]$ around the z-axis and a translation by vector (u, v, w). The point (u, v, w) is called the origin of the graduated line ℓ. Each number $z \in \mathbb{Z}$ defines a cell $c_z = [l_z, r_z)$ in ℓ that is obtained from cell $[z, z+1)$ of the x-axis by the composite transformation described above. The projection of a point p onto the line ℓ is the cell

c of ℓ that contains the intersection p^* between the line ℓ and the plane that passes through p and is orthogonal to ℓ.

A region of space consisting of all points that project to the same cell is called a *bucket*. A bucket is bounded by two planes orthogonal to ℓ through consecutive cell boundaries.

Remark 3.1. Let ℓ be a graduated line with unit direction vector d and let n be any unit vector normal to ℓ. Let ℓ' be the graduated line obtained by translating ℓ by $zd + rn$, for any $z \in \mathbb{Z}$ and $r \in \mathbb{R}$. Then lines ℓ and ℓ' determine the same bucket sets.

A solution to an instance $\mathcal{P}$ of $\mathrm{P}(3,1)$ is a graduated line that minimizes the maximum number of points of $\mathcal{P}$ that lie in the same bucket. As before, two graduated lines whose bucket sets produce the same partition of a point set $\mathcal{P}$ are said to be equivalent with respect to $\mathcal{P}$.

Lemma 3.1. *Let ℓ be a solution to an instance $\mathcal{P}$ of* $\mathrm{P}(3,1)$. *There exists a graduated line ℓ_0 equivalent to ℓ with a point of $\mathcal{P}$ lying on a boundary plane of a bucket of ℓ_0.*

We are going to describe an algorithm, which we call **MGL3**, to solve $\mathrm{P}(3,1)$. The algorithm is based, as algorithm **MGL2**, on two key ideas:

(1) As a consequence of Lemma 3.1, it suffices to consider graduated lines that have a bucket boundary passing through a point p_i of $\mathcal{P}$. Furthermore, from Remark 3.1 we can assume that p_i is the origin of the graduated line.
(2) Each graduated line with origin $p_i = o$ determines a partition of $\mathcal{P}$ into buckets and the set of all such graduated lines produces a finite number of different partitions of $\mathcal{P}$. The set of all graduated lines is obtained from the x-axis by composing a rotation by angle $\pi - \varphi$ around the y-axis and a rotation by angle θ around the z-axis. During such motions, the current partition changes only at angles θ, φ where the left boundary plane of a bucket passes through a point p_j of $\mathcal{P}$, $p_j \neq p_i$. At this instant, p_j changes from one bucket to the adjacent one. Consequently, in searching for the optimal solution for a fixed origin, it suffices to consider the finite collection of curves $\mathcal{C}$ of the two-dimensional space parametrized by the angles θ and φ for which such event occurs, since in any cell of the arrangement determined by $\mathcal{C}$ the partition of $\mathcal{P}$ remains constant.

The algorithm proceeds in two steps:

(1) For each point p_i of $\mathcal{P}$ find the graduated line ℓ_i of origin p_i that minimizes the maximum number of points of $\mathcal{P}$ that fall in the same bucket. Call this minimum number m_i.
(2) Report the line ℓ that corresponds to $\min\{m_1, \ldots, m_n\}$.

We now elaborate on Step 1. For $p_i = o$, consider the iteration with the motion of the x-axis that generates all the graduated lines with origin p_i. Let $S(p,r)$ denote the sphere of radius r centered at p. Denote by $\pi(k,\theta,\varphi)$ the plane obtained by rotating the plane $x = k$ by an angle $\pi - \varphi$ around the y-axis and then by an angle θ around the z-axis. We wish to determine the values of θ, φ for which an input point p_j lies on $\pi(k, \theta, \varphi)$. Each such pair of values θ, φ corresponds to an event of the motion. Note that if $d(p_i, p_j) < k$ the rotated plane cannot possibly contain p_j. For $|k| \leqslant \text{dist}(p_i, p_j)$, the solutions of the above equation correspond to the planes through p_j tangent to the sphere $S(p_i, k)$. The points of tangency determine a curve $X_{j,k}$ of the two-dimensional space defined by the angles θ, φ that can be found as the intersection circle $C_{j,k}$ between the sphere $S(p_i, k)$ and the sphere $S(m, \frac{1}{2}\text{dist}(p_i, p_j))$ where m is the midpoint of segment p_ip_j. See Fig. 7 for an example.

This procedure is repeated for integer values of k that satisfy $|k| \leqslant \text{dist}(p_i, p_j)$. Since in the bijection between points of the two-dimensional space determined by angles θ, φ of coordinates (θ, φ) and points of the unit sphere centered at p_i of polar coordinates (θ, φ), the curve $X_{j,k}$ is in correspondence with a circle on this unit sphere, the maximum number of intersection points between any couple of curves $X_{j,k}$ and $X_{j',k'}$ is bounded by 2. We break each $X_{j,k}$ curve, at its points of vertical tangency, into x-monotone curves. Finally, we construct the arrangement determined by the collection $\mathcal{C}$ of all such x-monotone curves.

We initialize the number of points that belong to each bucket for a graduated line with $\theta = \varphi = 0$ and the largest such count is identified. Then, starting in the cell that corresponds to $\theta = \varphi = 0$, we make a breadth first search in the dual graph of the arrangement of $\mathcal{C}$. For each new cell of the arrangement we update the bucket counts and we compute the largest count. The minimum m_i of these counts is reported as the best graduated line ℓ_i centered at p_i. The complete algorithm is summarized in Fig. 8.

The complexity of **MGL3** is dominated by the construction of the arrangement of line 12. The arrangement of a collection of n x-monotone curves, each pair of which intersect in at most 2 points, can be computed using an incremental approach in time $O(n\lambda_4(n))$ and space $O(n^2)$, where $\lambda_4(n)$ is roughly linear in n [9]. In our case, since there are $t_i = \max_{1 \leqslant j \leqslant n}\{\text{dist}(p_i, p_j)\}$ curves in $\mathcal{C}$, the construction of the arrangement of line 12 requires $O(t_i n \lambda_4(t_i n))$ time and $O(t_i^2 n^2)$ space. Thus, the total execution time of the algorithm is $O(tn^2\lambda_4(tn))$, where $t = \max\{t_i\}$ is the diameter of $\mathcal{P}$. The space required is $O(t^2n^2)$. We have proven the following result:

Theorem 3.1. *An instance $\mathcal{P}$ of problem* $P(3,1)$ *can be solved in* $O(tn^2\lambda_4(tn))$ *time and* $O(t^2n^2)$ *space, where t is the diameter of $\mathcal{P}$.*

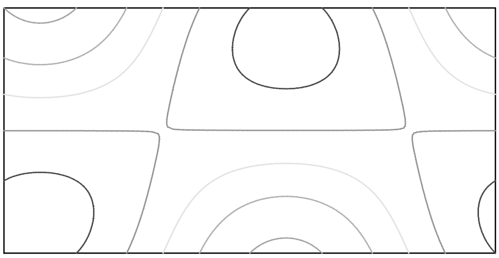

Fig. 7. Example of $X_{j,k}$ curves.

```
algorithm MGL3(𝒫)
1  for i ← 1 to n do
2      𝒞 ← ∅
3      set all bucket counts to 0
4      for j ← 1 to n do
5          q_j ← p_j − p_i
6          increment the count of bucket ⌊x(q_j)⌋ by 1
7          d_j ← dist(p_i, p_j)
8          for k ← 0 to ⌊d_j⌋ do
9              for each circle of tangency C_{j,k} between q_j and sphere S(o, k)
10                 X_{j,k} ← angle(C_{j,k})
11                 𝒞 ← 𝒞 ∪ {X_{j,k}}
12     Construct the arrangement determined by 𝒞
13     m_i ← maximum number of points in a bucket of ℓ(0, 0, p_i)
14     for each cell c of the arrangement determined by 𝒞 do
15         select angles θ, φ associated with cell c
16         compute bucket count of ℓ(θ, φ, p_i)
17         m ← maximum number of points in a bucket of ℓ(θ, φ, p_i)
18         if m < m_i then m_i ← m
19 report min{m_1, ..., m_n}
end
```

Fig. 8. Algorithm for solving an instance of P(3,1). A bucket is denoted by the x-coordinate of its left boundary plane before motion.

An alternative approach for solving P(3,1) was suggested by an anonymous referee and works as follows. We know that it suffices to consider graduated lines that have a bucket boundary passing through the origin $o = p_i$. One only needs to find the direction of the solution line ℓ. For any other point p_j, we need to determine the critical orientations of ℓ in which p_j projects to an endpoint of a bucket boundary. Now p_j projects to the point at distance k from the origin when the direction of ℓ traces a circle on the sphere of directions $\mathbb{S}^2$, whose center is the direction op_j and whose opening angle β satisfies $\cos\beta = k/\text{dist}(o,p_j)$. Putting $t_{i,j} = \text{dist}(o,p_j)$, we obtain $\lfloor t_{i,j} \rfloor$ concentric circles, and we collect them for each p_j still keeping $p_i = o$. We thus get an arrangement of $\sum_j \lfloor t_{i,j} \rfloor \leqslant t_i n$ (non-maximal) circles on $\mathbb{S}^2$, where $t_i = \max_{1 \leqslant j \leqslant n}\{\text{dist}(p_i,p_j)\}$. This arrangement of circles can be computed in $O(t_i n \lambda_4(t_i n))$ time with an incremental algorithm [9].

3.2. Approximate solutions to P(3,1)

A good approximate solution to an instance of P(3,1) can be obtained in a manner similar to that used to approximate P(2,1). In other words, a single iteration of the loop of line 4 results in a solution with cost no higher than twice the optimal. Once the orientation of the graduated line is fixed, further shifting cannot change the cost of the answer by a factor higher than 2. Since the argument is essentially the same as that used in Section 2.1 we provide only the main result:

Theorem 3.2. *Let C be the cost of an optimal solution to* $\mathrm{P}(3,1)$. *There is an algorithm that runs in* $\mathrm{O}(tn\lambda_4(tn))$ *time that finds an answer with cost no bigger than* $2C$.

3.3. Graduated planes

The graduated plane $\pi = \pi(\alpha, \theta, \varphi, u, v, w)$ is the plane obtained from the xy-coordinate plane by composing a rotation by angle $\alpha \in [0, \pi]$ around the z-axis, a rotation by angle $\pi - \varphi$, $\varphi \in [-\pi/2, \pi/2]$ around the y-axis, a rotation by angle $\theta \in [0, 2\pi]$ around the z-axis, and a translation by vector (u, v, w). The point (u, v, w) is the *origin* of the graduated plane π. The graduated lines obtained by transforming the x and y-coordinate lines with the composite transformation described above are called the graduated axes of π.

Each pair of numbers $z_1, z_2 \in \mathbb{Z}$ define a cell c_{z_1,z_2} in π obtained from the cell $[z_1, z_1 + 1) \times [z_2, z_2 + 1)$ of the xy-plane, by the composite transformation described above.

The projection of a point p onto the plane π is the cell c of π that contains the intersection p^* between the plane π and the line through p orthogonal to π. A region of space consisting of all points that project to the same cell is called a *bucket*. A bucket is bounded by four planes orthogonal to π that pass through the four boundaries of a cell.

Remark 3.2. Let π be a graduated plane with unit normal n and let d_x and d_y be unit direction vectors of the graduated axes of π. Let π' be the graduated plane obtained by translating π by $z_x d_x + z_y d_y + rn$, for any $z_x, z_y \in \mathbb{Z}$ and $r \in \mathbb{R}$. Then planes π and π' determine the same bucket sets.

A solution to an instance $\mathcal{P}$ of $\mathrm{P}(3,2)$ is a graduated plane that minimizes the maximum number of points of $\mathcal{P}$ that lie in the same bucket. As before, two graduated planes whose bucket sets produce the same partition of $\mathcal{P}$ are said to be equivalent with respect to $\mathcal{P}$.

Lemma 3.2. *Let* π *be a solution to an instance* $\mathcal{P}$ *of* $\mathrm{P}(3,2)$. *There exist a graduated plane* π_0 *equivalent to* π *and two points* p_i, p_j of $\mathcal{P}$ *so that* p_i *lies on a bottom boundary plane of a bucket of* π_0 *and* p_j *lies on a left boundary plane of a bucket of* π_0. *In some cases, point* p_i *lies on a left-bottom boundary line of a bucket of* π_0.

Proof. Let $\mathcal{P} = \{p_1, \ldots, p_n\}$ be a point set and $\mathcal{P}^* = \{p_1^*, \ldots, p_n^*\}$, where p_i^* is the projection of p_i onto π, $1 \leqslant i \leqslant n$. Denote c_k the cell containing p_k^*, $1 \leqslant k \leqslant n$. For each p_k^* of $\mathcal{P}^*$ let db_k be the distance from p_k^* to the bottom side b_k of the cell c_k and let dl_k be the distance from p_k^* to the left side l_k of the cell c_k. Suppose that $db_i = \min_{1 \leqslant k \leqslant n}\{db_k\}$ and $dl_j = \min_{1 \leqslant k \leqslant n}\{dl_k\}$. The transformation obtained by composing the translation on the direction of the y-axis that moves the point p_i^* onto line b_i and the translation on the direction of the x-axis that moves the point p_j^* onto line l_j moves plane π to the desired plane π_0. In the special case in which $i = j$, the point $p_i = p_j$ belongs to a bottom-left bucket boundary line. □

Next we sketch an algorithm **MGPG** to solve P(3, 2). First we consider the general case and then the special case as described in Lemma 3.2.

3.3.1. General case

(1) As a consequence of Lemma 3.2 it suffices to determine graduated planes that have a bottom bucket boundary passing through a point p_i of $\mathcal{P}$ and a left bucket boundary passing through a point p_j of $\mathcal{P}$, with $p_i \neq p_j$. Furthermore, from Remark 3.2, we can assume that p_i belongs to the bottom side of a cell c of the graduated plane. If we denote by p_j^* the projection of p_j onto the graduated plane, then the origin of the graduated plane "sees" p_i and p_j^* under a right angle. Consequently we can assume that the origin of the graduated plane is the point where the line through the bottom side of c and the plane containing the left bucket boundary passing through p_j intersect.
We discuss briefly how to fulfill these conditions for each pair of points p_i and p_j. Without loss of generality we can assume that $p_i = o$. Consider the graduated plane $\pi = \pi(\alpha, \theta, \varphi, 0, 0, 0)$ through p_i and suppose that the movement that transforms the xy-plane onto the plane π also transforms the xz-plane and the yz-plane onto planes π' and π'', respectively. Denote by π''_j the plane through point p_j parallel to π''. Then, if $\text{dist}(o, \pi''_j) < 1$, we take as origin (u, v, w) of the final graduated plane the point $\pi \cap \pi' \cap \pi''_j$ and, consequently u, v, w depend only on $(\alpha, \theta, \varphi)$.
(2) Each of these final graduated planes determines a partition of $\mathcal{P}$ into buckets and the set of all such graduated planes produces a finite number of different partitions of $\mathcal{P}$. During the motion of the xy-coordinate plane, the current partition changes only at angles α, θ, φ where the bottom boundary plane passes through a point p_k of $\mathcal{P}$ or the left boundary plane passes through a point $p_{k'}$ of $\mathcal{P}$, p_k, $p_{k'} \neq p_i, p_j$. At these instants, p_k or $p_{k'}$ change from one bucket to the adjacent one. Consequently, in searching for the optimal solution for a pair of point p_i,p_j, it suffices to consider the finite collection of surfaces $\mathcal{S}$ of the three-dimensional space determined by the angles $\alpha \in [0, \pi]$, $\theta \in [0, 2\pi]$, $\varphi \in [-\pi/2, \pi/2]$ for which such events occur, since in any cell of the arrangement determined by $\mathcal{S}$ the partition of $\mathcal{P}$ remains constant.

In the general case, algorithm **MGPG**, similarly to the previous ones, proceeds in two steps:

(1) For $p_i, p_j \in \mathcal{P}$, $1 \leqslant i < j \leqslant n$ find the graduated plane $\pi_{i,j}$ that minimizes the maximum number of points of $\mathcal{P}$ that fall in the same bucket. Call this minimum $m_{i,j}$.
(2) Report the plane π that corresponds to $\min\{m_{i,j} | 1 \leqslant i < j \leqslant n\}$.

During Step 1, for each $1 \leqslant i < j \leqslant n$, it is necessary to consider an arrangement determined by a finite collection of surfaces $\mathcal{S}$ of a three-dimensional space. The number $t_{i,j}$ of surfaces in $\mathcal{S}$ is in $O(\max\{\max_{1 \leqslant m \leqslant n} \text{dist}(p_i, p_m), \max_{1 \leqslant m \leqslant n} \text{dist}(p_j, p_m)\})$. Since we only need to encode the adjacency relationship among three-dimensional cells of the arrangement we compute the vertical decomposition of the arrangement and then we compute the adjacency relationship of cells in the decomposition [10]. Then we initialize the number of points that belong to each bucket for a graduated plane with $\alpha = \theta = \varphi = 0$ and the largest such count is identified. Finally we traverse $\mathcal{S}$ by using the adjacency relationship of the cells in its vertical decomposition. For each new cell of the arrangement we update the bucket counts and we compute the largest count. The minimum $m_{i,j}$ of these counts is reported as the best graduated plane $\pi_{i,j}$.

In the general case, the complexity of **MGPG** is dominated by the construction of the arrangement of surfaces. The number of cells of the vertical decomposition of an arrangement of n surfaces in $\mathbb{R}^3$ is $O(n^2\lambda_q(n))$, where q is a constant depending on the maximum degree of the surfaces, and the vertical decomposition of the arrangement can be computed in randomized expected time $O(n^{3+\epsilon})$, using the random-sampling technique [5]. In our case, the vertical decomposition of $\mathcal{S}$ can be constructed in $O((t_{i,j}n)^{3+\epsilon})$ time and $O((t_{i,j}n)^2\lambda_q(t_in))$ space. Thus, if $t = \max\{t_{i,j}\}$ we have the following result:

An instance $\mathcal{P}$ of problem $P(3,2)$ in the general case can be solved in $O(n^2(tn)^{3+\epsilon})$ expected time and $O((tn)^2\lambda_q(tn))$ space.

3.3.2. Special case

(1) As a consequence of Lemma 3.2, it suffices to consider graduated planes that have a bottom-left bucket boundary plane passing through a point p_i of $\mathcal{P}$. Furthermore, from Remark 3.2 we can assume that p_i is the origin of the graduated plane.
(2) Each graduated plane with origin $p_i = o$ determines a partition of $\mathcal{P}$ into buckets and the set of all such graduated planes produces a finite number of different partitions of $\mathcal{P}$. In searching for the optimal solution for a fixed origin, it suffices to consider a finite collection of surfaces $\mathcal{S}$ of the three-dimensional space parametrized by the angles $\alpha \in [0,\pi]$, $\theta \in [0, 2\pi]$, $\varphi \in [-\pi/2, \pi/2]$.

In the special case, algorithm **MGPG** proceeds in two steps:

(1) For $p_i \in \mathcal{P}$ find the graduated plane π_i that minimizes the maximum number of points of $\mathcal{P}$ that fall in the same bucket. Call this minimum m_i.
(2) Report the plane π that corresponds to $\min\{m_i | 1 \leqslant i \leqslant n\}$.

During Step 1, for each $1 \leqslant i \leqslant n$, it is necessary to consider an arrangement determined by a finite collection of surfaces $\mathcal{S}$ of a three-dimensional space. As explained for the general case, using the results of [5], the optimal α, θ, and φ can be found in $O((tn)^{3+\epsilon})$ expected time and $O((tn)^2\,\lambda_q(tn))$ space, where t is the diameter of $\mathcal{P}$. Thus we have the following result:

An instance $\mathcal{P}$ of problem $P(3,2)$ in the special case can be solved in $O(n(tn)^{3+\epsilon})$ expected time and $O((tn)^2\lambda_q(tn))$ space.

Since time complexity of algorithm **MGPG** is determined by the general case, we have:

Theorem 3.3. *An instance $\mathcal{P}$ of problem $P(3,2)$ can be solved in $O(n^2(tn)^{3+\epsilon})$ expected time and $O((t\,n)^2\lambda_q(tn))$ space.*

3.4. Approximate solutions to $P(3,2)$

Using an approach similar to that of Section 2.1 we can use an arbitrary origin (u_1, v_1, w_1) to derive an approximation algorithm whose cost differs from the optimal by no more than a constant factor and whose complexity is lower than that of **MGPG**. We start by showing that shifting a graduated plane over itself changes the quality of the solution by at most a factor of 4.

Lemma 3.3. *Let d_x and d_y be unit direction vectors for the axes of a graduated plane π with cost C. Let C' be the cost of a graduated plane π' obtained by shifting π by vector $r_x d_x + r_y d_y$, for any $r_x, r_y \in \mathbb{R}$.* Then $\frac{1}{4} \leqslant \frac{C'}{C} \leqslant 4$.

Proof. First, we can assume that $0 \leqslant r_x, r_y < 1$ as shifting π by $(r_x \bmod 1)d_x + (r_y \bmod 1)d_y$ results in a plane equivalent to π'. Thus, after shifting, the worst that can happen is that the contents of four buckets are combined into one, at most quadrupling the cost of the solution. Since no bucket of π has more than C points we have $C' \leqslant 4C$. Conversely, the cost of a bucket can be reduced by shifting. Take a bucket of π with C points. After shifting, these points will be distributed among at most four buckets. At least one of these buckets must have no fewer than $\lceil C/4 \rceil$ points which implies that $C' \geqslant \lceil C/4 \rceil$. □

Now let $\pi_0 = \pi(\alpha_0, \theta_0, \varphi_0, u_0, v_0, w_0)$ be an optimal solution for an instance of $P(3,2)$. Also let $\pi_1 = \pi(\alpha_1, \theta_1, \varphi_1, u_1, v_1, w_1)$ be the best solution over all possible values of α, θ, and φ for an arbitrary but fixed origin (u_1, v_1, w_1). Clearly, $\mathrm{cost}(\pi_1) \leqslant \mathrm{cost}(\pi(\alpha_0, \theta_0, \varphi_0, u_1, v_1, w_1))$ as π_1 is optimal for (u_1, v_1, w_1). Furthermore, for some (u', v', w'), there is a graduated plane $\pi' = \pi(\alpha_0, \theta_0, \varphi_0, u', v', w')$ through (u_1, v_1, w_1) which is parallel and equivalent to the optimal π_0. Since $\pi(\alpha_0, \theta_0, \varphi_0, u_1, v_1, w_1)$ can be obtained from π' by a translation along the direction vectors of the axes of π', Lemma 3.3 implies that $\mathrm{cost}(\pi(\alpha_0, \theta_0, \varphi_0, u_1, v_1, w_1)) \leqslant 4\mathrm{cost}(\pi')$. Thus, we have $\mathrm{cost}(\pi_1) \leqslant \mathrm{cost}(\pi(\alpha_0, \theta_0, \varphi_0, u_1, v_1, w_1)) \leqslant 4\mathrm{cost}(\pi') = 4\mathrm{cost}(\pi_0)$. We still need to find the optimal projection for a fixed (u_1, v_1, w_1). For this purpose we create an arrangement of hyper-surfaces parametrized by α, θ, and φ, in a manner similar to that outlined in the previous section. All points in the same cell of the arrangement correspond to graduated planes with the same origin that produce the same partition of $\mathcal{P}$. Using the results of [5] the optimal α,θ, and φ can be found in $O((tn)^{3+\epsilon})$ expected time, where t is the diameter of $\mathcal{P}$. We have proven the following result.

Theorem 3.4. *Let C be the cost of an optimal solution to* $P(3,2)$. *There is an algorithm that runs in expected time* $O((tn)^{3+\epsilon})$ *that finds an answer with cost no bigger than* $4C$.

4. Future work

The algorithms described in this paper are based on identifying classes of equivalent graduated lines or planes that define bucket sets that produce exactly the same partition of the input $\mathcal{P}$. Other optimization problems related to projections of a point set onto a graduated line or plane and depending on the number of points projected in each cell can be solved using basically the same approach. Examples include finding an orthogonal projection that maximizes the number of cells with at least one projected point of $\mathcal{P}$ or one that minimizes the total number of collisions, i.e., pairs of points projected to the same cell. Furthermore, when choosing between two projections with the same cost it is probably desirable to choose the one that separates the projected points from different buckets as much as possible. Thus, as suggested by an anonymous referee, it would also be interesting to consider the problem of maximizing $D = \sum_{i,j} d(p_i^*, p_j^*)$, where $d(p_i^*, p_j^*)$ denotes the distance, measured in pixels, between the projections of p_i and p_j. We think this problem can be solved using an approach similar to the one described in this paper.

It remains as future work to state the bounds of our algorithms not just in terms of n and t, but in terms of the distribution of distances between the points of $\mathcal{P}$, since the bounds can be much smaller for dense or random sets.

Acknowledgments

The authors thank G. Toussaint and A. Mesa for organizing the workshop at which this research was initiated. We are indebted to an anonymous referee for many valuable remarks. We also thank Sergio Cabello for comments leading to improve some results related to problem P(3, 2). The first author was supported in part by Grant BFM2000-1052-C02-01. The second author was supported in part by Grants MEC-DGES-SEUID PB98-0933, SGR1999-0356, and SGR2001-0224. The third author was supported in part by the National Science Foundation under Grant DMS-0107628. The fourth author was supported in part by Grants TIC2001-2392-C03-01 and DURSI 2001SGR-00296.

References

[1] P.K. Agarwal, B.K. Bhattacharya, S. Sen, Improved algorithms for uniform partitions of points, Algorithmica 32 (4) (2002) 521–539.

[2] T. Asano, T. Tokuyama, Algorithms for projecting points to give the most uniform distribution with applications to hashing, Algorithmica 9 (1993) 572–590.

[3] P. Bose, M.F. Gómez, P. Ramos, G. Toussaint, Drawing nice projections of objects in space, J. Vis. Commun. Image Represent. 10 (2) (1999) 155–172.

[4] P. Bose, A. Maheshwari, P. Morin, J. Morrison, The grid placement problem, in: Proceedings of the 17th European Workshop on Comp. Geometry, 2001, pp. 74–77.

[5] B. Chazelle, H. Edelsbrunner, L.J. Guibas, M. Sharir, A singly-exponential stratification scheme for real semi-algebraic varieties and its applications, in: Proceedings of the 16th Internat. Colloq. Automata Lang. Program, Lecture Notes Comput. Sci., vol. 372, Springer-Verlag, 1989, pp. 179–192.

[6] F. Gómez, F. Hurtado, J.A. Sellarès, G. Toussaint, On degeneracies removable by perspective projection, Int. J. Math. Algorithms 2 (2001) 227–248.

[7] F. Gómez, F. Hurtado, J.A. Sellarès, G. Toussaint, Nice perspective projections, J. Vis. Commun. Image Represent. 12 (4) (2001) 387–400.

[8] F. Gómez, S. Ramaswami, G. Toussaint, On computing general position views of data in three dimensions, J. Vis. Commun. Image Represent. 13 (4) (2002) 401–424.

[9] M. Sharir, P.K. Agarwal, Davenport–Schinzel Sequences and their Geometric Applications, Cambridge University Press, Cambridge, 1995.

[10] M. Sharir, Recent developments in theory of arrangements of surfaces, in: Lecture Notes in Computer Science, vol. 1738, Springer-Verlag, 2000, pp. 1–21.