Pattern Recognition Letters 12 (1991) 757-764
North-Holland

December 1991

Optimal algorithms for extracting spatial regularity in images

Andrew B. Kahng and Gabriel Robins

UCLA Department of Computer Science, Los Angeles, CA 90024-1596, USA

Revised 12 August 1991

Abstract

Kahng, A.B. and G. Robins, Optimal algorithms for extracting spatial regularity in images, Pattern Recognition Letters 12 (1991) 757-764.

Finding spatial regularity in images is important in military applications (e.g., finding rows of landmines), texture analysis, and other areas. We give an optimal $\Theta(n^2)$ algorithm for finding all maximal equally-spaced collinear subsets within a pointset in E^d. We also generalize this method to yield an optimal $\Theta(n^3)$ algorithm for determining all maximal regular coplanar lattices.

Keywords. Pattern recognition, regularity, computational geometry, combinatorial algorithms.

1. Introduction

Given a finite pointset $P \subset E^d$ with all points distinct, a subset $P' \subseteq P$ is *collinear* if $|P'| \geq 2$ and all points of P' lie on a single line. A *maximal collinear subset* (MCS) of P is a collinear subset that is not properly contained in any other collinear subset of P. The problem of finding an MCS in an image arises in line and feature detection for computer vision, and instances can occur in dimensions greater than two. To solve the MCS problem, bucketing techniques based on the Hough transform (e.g., Duda and Hart (1972), Risse (1989), Ben-Tzvi and Sandler (1990)) or other duality relations are often used. However, these methods do not give any insight into the *spatial regularity* of collinear points.

To capture the notion of regularity in an image, we call a collinear subset $P' \subseteq P$ an *equally-spaced* collinear subset if $|P'| \geq 3$ and all the points of P' are equally-spaced along their containing line. When given a pointset in the plane, it is very natural to ask "What is its largest equally-spaced collinear subset?"

Maximum Equally-Spaced Collinear Subset (MESCS) Problem. For n points in E^d, find the largest equally-spaced, collinear subset of points.

Like MCS, the MESCS problem arises in many practical applications, particularly since regularity is in many cases the distinguishing characteristic of 'interesting' regions in an image. A motivating (military) application for our work is the examination of infrared ground surveillance bitmaps to find equally-spaced collinear 'hotspots' (rows of surface landmines, fenceposts in a region perimeter, etc.) Since we wish to find contiguous se-

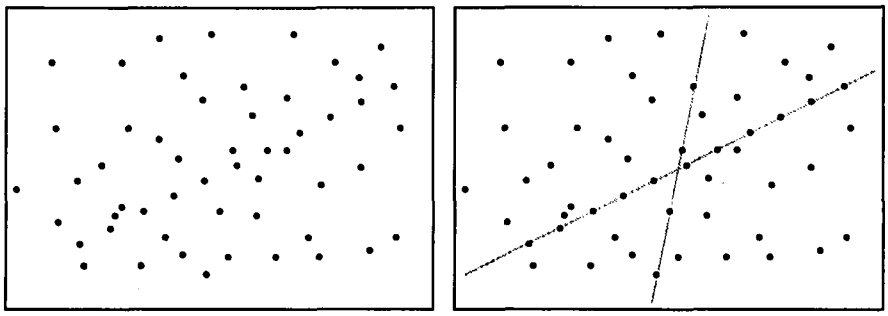

Figure 1. A sample pointset and two of its maximal equally-spaced collinear subsets.

quences of points with equal spacing, the usual spectral methods for determining gross periodicity in data cannot be used.

Often, we would like to examine *all* possible regularities in an image, and therefore require a roster of all maximal (equally-spaced) collinear subsets; the output itself can be of size $\Omega(n^2)$, as described below. In other words, we want the complete *order statistics* of the input with respect to the given problem:

All Maximal Equally-Spaced Collinear Subsets (AMESCS) Problem. Given a set of n points in E^d, find all maximal equally-spaced, collinear subsets of points.

Figure 1 illustrates a pointset and two of its maximal equally-spaced collinear subsets.

Existing algorithms for line-finding do not extend well to encompass the added difficulties of regularity detection or high-dimensional data. For example, MCS in E^d can be solved in $O(n^d)$ time using a method of Edelsbrunner (1987, pp. 278–282), but the exponential dependence on dimension is prohibitive. The work of Edelsbrunner and Guibas (1986) implies an $O(n^2)$ time algorithm for MCS in two dimensions, but this method does not generalize to higher dimensions, nor does it seem applicable to the MESCS or the AMESCS problem. Hough-style bucketing algorithms will also usually require time exponential in the dimension d or in the granularity of the bucketing. In general, we find that the current ad hoc methods for finding spatial regularity in images (e.g., in the texture classification literature) are dependent on low dimension, prescribed matching templates, or limits on the scale of features. On the other hand, our formulation of regularity detection and the algorithms proposed below are quite general.

This note presents an optimal $\Theta(n^2)$ time algorithm for solving the AMESCS problem for a pointset in arbitrary dimension. We also generalize the 'equally-spaced' notion of regularity to 2-dimensional *lattices*, and give an optimal $\Theta(n^3)$ algorithm for determining all maximal regularly-spaced planar sublattices within an arbitrary pointset in E^d.

2. Preliminaries

We establish a lower bound of $\Omega(n \log n)$ for both MCS and MESCS by reduction from the Element Uniqueness (EU) problem (i.e., determining whether a given set of numbers contains duplicates), which is known to require $\Omega(n \log n)$ time in the standard comparison model of computation (e.g., Preparata and Shamos (1985)). A lower bound of $\Omega(n^2)$ for AMESCS is established based on the output size.

Theorem 1. MCS *and* MESCS *both require at least* $\Omega(n \log n)$ *time to compute.*

Proof. Given an instance $S = (s_1, \ldots, s_n)$ of EU, we construct an instance of the 2-dimensional MCS problem by transforming each element s_i into a point $p_i = (s_i, s_i^2)$, as shown in Figure 2. The resulting MCS instance has a collinear subset with cardinality 3 or greater iff S contains duplicates, since

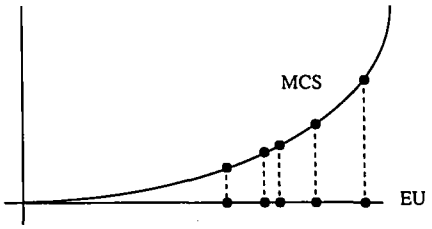

Figure 2. Transforming an instance of the Element Uniqueness (EU) problem into an instance of the Maximal Collinear Subset (MCS) problem.

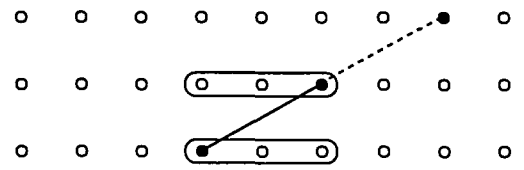

Figure 3. Example of a pointset containing a quadratic number of distinct maximal collinear equally-spaced subsets: each point in the middle third of the bottom row can be used along with an arbitrary point in the middle third of the middle row to determine a distinct collinear equally-spaced triple.

no triple of *distinct* points on a parabola can be collinear. It follows that the $\Omega(n \log n)$ lower bound for EU also holds for the MCS problem in all dimensions.

Similarly, we may reduce EU to a 1-dimensional instance of MESCS as follows. Given an instance $S = (s_1, \ldots, s_n)$ of EU, we transform each element s_i into

$$p_i = s_i + \varepsilon/2^{s_i}$$

(where ε is a positive real number smaller than the difference between any two s_i's). This transformation eliminates all arithmetic progressions that may be present in the original EU instance, since all differences $p_i - p_j$ are distinct, $i \neq j$. We then form the MESCS instance

$$P = (p_1, \ldots, p_n, p_1, \ldots, p_n);$$

P will contain 3 or more equally-spaced elements (i.e., an arithmetic progression of size 3) iff S contains duplicate elements (this arithmetic progression will be degenerate, with difference = 0). We conclude that the $\Omega(n \log n)$ lower bound for EU also holds for MESCS in all dimensions. □

Theorem 2. AMESCS *requires at least* $\Omega(n^2)$ *time to compute.*

Proof. The output of AMESCS can be of size $\Omega(n^2)$, since there can be a quadratic number of distinct collinear equally-spaced triples, as exemplified by the pointset

$$\{1, 2, 3, \ldots, n/3\} \times \{1, 2, 3\},$$

shown in Figure 3. In this example, each point in the middle third of the bottom row can be used along with an arbitrary point in the middle third of the middle row, to determine a distinct collinear equally-spaced triple. The total number of such triples is therefore $(n/9)^2 = \Omega(n^2)$. □

3. An optimal AMESCS algorithm

A naive $O(n^2 \log n)$ algorithm for AMESCS iterates through all $\binom{n}{2}$ line segments induced by the pointset, and determines how far each segment spacing can be extended to either direction within the pointset. Extending a given solution in this manner costs $O(\log n)$ time per added point (via binary search on a preprocessed, sorted version of the input). Since no segment can participate in more than one solution, we need to examine each segment only once during this process, hence the $O(n^2 \log n)$ bound.

In this section, we develop an optimal method for the AMESCS problem in arbitrary dimension. Our method is based on a solution of the 1-dimensional AMESCS problem, i.e., finding all maximal arithmetic progressions in a set of numbers.

First, consider the very restricted 1-dimensional AMESCS variant which looks only for equally-spaced *triples* of points, i.e., arithmetic progressions of length three.[1] We may find *all* equally-spaced triples as follows. First, sort the input using

[1] Using the same reductions as in the proof of Theorem 1, we can show that this problem, as well as the decision problem of determining whether a given pointset contains a collinear triplet, also has an $\Omega(n \log n)$ lower bound. Interestingly, for neither of these apparently much simpler decision problems is an $o(n^2)$ time algorithm known (Edelsbrunner, O'Rourke and Seidel (1986)).

$O(n \log n)$ time, yielding a sorted list $p_1, p_2, \ldots, p_n$. Next, assume that we have a pointer to position $A = i$ in the sorted list, and that p_A is the leftmost point of a triple. We maintain two pointers B and C to positions in the sorted list, with initially $B = i+1$ and $C = i+2$. Let $X_1(p_A)$ denote the coordinate (on the x_1 axis) of the point p_A. If

$$X_1(p_B) - X_1(p_A) > X_1(p_C) - X_1(p_B),$$

we increment C by 1, otherwise we increment B by 1. Whenever the two differences are equal, we record the equally-spaced triple (p_A, p_B, p_C). Because the pointers B and C simply march along the sorted list, we find all equally-spaced triples with $X_1(p_A)$ as leftmost component using linear time; iterating over $i = 1, 2, \ldots, n-2$ will report all equally-spaced triples within $O(n^2)$ time. This method is asymptotically optimal, since by Theorem 2 inputs can have up to a quadratic number of triples.

The main idea is that the AMESCS problem can be solved by overlapping these equally-spaced triples in order to determine all maximal equally-spaced collinear subsets. This is accomplished by constructing a graph where for each reported equally-spaced triple (p_A, p_B, p_C) we create the nodes $\langle A, B\rangle$ and $\langle B, C\rangle$ and the edge $(\langle A, B\rangle, \langle B, C\rangle)$. Each node in this graph has degree at most two, so the edge set and vertex set both have size $O(n^2)$. Connected components in this graph correspond to maximal equally-spaced collinear subsets in the original pointset, and any linear-time algorithm for determining connected components can be used over this graph[2] to yield all maximal equally-spaced subsets within $O(n^2)$ time.

To solve AMESCS in higher dimensions, we sort the pointset by the first coordinate only; i.e., we project onto the x_1 axis. Without loss of generality, we can assume that no two points have the same x_1 coordinate.[3] We then proceed to solve the 1-dimensional AMESCS problem for the sorted, projected pointset, as outlined above. Equally-spaced triples in the pointset will correspond to equally-spaced triples in the projection. Although some equally-spaced triples in the projection will not correspond to actual equally-spaced triples in the pointset, checking for such spurious triples requires only constant time per triple in any fixed dimension. Since the number of equally-spaced triples is bounded by $\binom{n}{2}$ in all dimensions, our algorithm will run in time $O(n^2)$ in any fixed dimension. Our optimal AMESCS algorithm for an arbitrary finite pointset $P \subset E^d$ is thus as given in Figure 4.

4. Generalization to equally-spaced lattices

In this section, we generalize the notion of 'equally-spaced' regularity to two dimensions by considering lattices that are determined by linear combinations of a pair of linearly independent vectors.

Definition. Given fixed d-dimensional vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are linearly independent, a *lattice* $L(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ is a pointset of the form

$$\{v \mid v = j\boldsymbol{a} + k\boldsymbol{b} + \boldsymbol{c} \text{ for all integers } j \text{ and } k\}.$$

The lattice point v is said to have *lattice coordinates* (j, k).

Definition. Given a fixed lattice $L(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$, a finite coplanar pointset L' is a *sublattice* of L if $L' \subseteq L$.

Next, we introduce the notion of lattice 'cells':

Definition. Four points of a given fixed lattice L define a lattice *cell* if their lattice coordinates are of the form

$$\{(j, k), (j+1, k), (j, k+1), (j+1, k+1)\};$$

the lattice cell itself is said to have *cell coordinates* (j, k). Four points in a sublattice $L' \subseteq L$ constitute a cell in L' if they also constitute a cell in L.

[2] Converting from this 'ordered-pair' (edge list) representation to the standard adjacency-list representation is accomplished in time proportional to the size of the graph: we sort the edge list by its first component, then scan the sorted edge list, grouping all edges with the same first component into an adjacency list for that vertex. Since edge components are pairs of vertex/point indices (i.e., integers between 1 and n), a linear-time bucket sort may be used.

[3] E.g., rigidly rotate the pointset by a tiny angle θ so as to make all of the x_1 coordinates unique.

AMESCS Algorithm for arbitrary dimension

Input: An arbitrary finite pointset P in E^d
Output: All maximal collinear equally-spaced subsets of P

Rotate P if necessary so that all x_1 coordinates are unique
$P = (p_1, \ldots, p_n)$ = **Sort** P by x_1 coordinate
$G = (V, E) = (\emptyset, \emptyset)$
For $A = 1$ to $n - 2$ **Do**
  $B = A + 1$
  $C = A + 2$
  Until $C > n$ **Do**
    If p_A, p_B, p_C are collinear and equally-spaced **Then**
      $V = V \cup \{\langle A, B\rangle, \langle B, C\rangle\}$
      $E = E \cup \{(\langle A, B\rangle, \langle B, C\rangle)\}$
    If $X_1(p_B) - X_1(p_A) > X_1(p_C) - X_1(p_B)$ **Then** $C = C + 1$ **Else** $B = B + 1$
Convert G into adjacency-list format (using bucket sort)
Output the connected components of G

Figure 4. An optimal $O(n^2)$ algorithm for the AMESCS problem in arbitrary dimension.

Definition. Two cells p and q of a given lattice are *neighbors* (denoted $p \Diamond q$) if their respective cell coordinates (j_1, k_1) and (j_2, k_2) satisfy

$$|j_1 - j_2| + |k_1 - k_2| = 1.$$

The relation $\Diamond$ over the cells of a sublattice induces a graph structure:

Definition. Given a fixed lattice L, the *cell graph* of a sublattice $L' \subseteq L$ is defined by $G(L') = (V, E) = (\{\langle u\rangle | u \text{ is a cell in } L'\}, \{(\langle u\rangle, \langle v\rangle) | u \text{ and } v \text{ are cells in } L', \text{ and } u \Diamond v\})$.

Definition. A set of points L' in the plane is said to be *regularly-spaced* with respect to some fixed lattice L if

(i) L' is a sublattice of L,

(ii) every point in L' belongs to some cell of L', and

(iii) the cell graph of L' is connected.

A regularly-spaced subset is *maximal* if it is not a proper subset of any other regularly-spaced subset.

We may now define the following problem:

Maximum Regularly-Spaced Subset (MRSS) Problem. Given a pointset in the plane, find its largest regularly-spaced subset.

In particular, for a variety of applications we are interested in finding *all* order statistics with respect to regularity in two dimensions (i.e., all maximal regularly-spaced subsets):

All Maximal Regularly-Spaced Subsets (AMRSS) Problem. Given a pointset in E^d, find all of its maximal regularly-spaced coplanar subsets.

Figure 5 illustrates a 2-dimensional instance of AMRSS and one of its maximal regularly-spaced subsets. It is easy to show a worst-case upper bound of $O(n^3)$ on the output size of AMRSS:

Theorem 3. *The sum of the sizes (i.e., number of points) of all maximal regularly-spaced subsets embedded in a given set of points is bounded by* $O(n^3)$.

Proof. A non-collinear triple of points is both necessary and sufficient to completely determine exactly three structures (i.e., there are only three ways to complete the parallelogram). Thus the number of distinct lattice structures induced by a given pointset is bounded by $3 \cdot \binom{n}{3} = O(n^3)$. Recall the definition of a lattice cell above; since any three points can belong to at most three different cells, it follows that a pointset contains at most $O(n^3)$ cells. Because a cell uniquely determines a lattice structure, no cell can appear more than once in the output. Therefore, the size of the output is $O(n^3)$. □

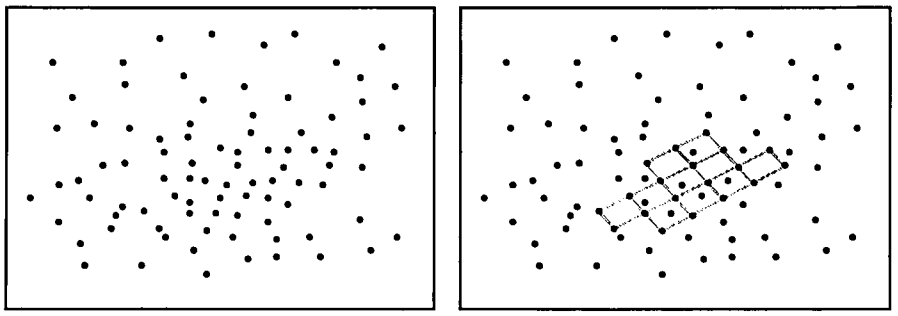

Figure 5. A sample 2-dimensional pointset and one of its maximal regularly-spaced subsets.

We may also show a worst-case lower bound of $\Omega(n^3)$ for any AMRSS algorithm:

Theorem 4. *The* AMRSS *problem requires at least* $\Omega(n^3)$ *time to solve.*

Proof. Consider the pointset

$$\{1, 2, 3, \ldots, n/2\} \times \{0, 1\},$$

where the output of AMRSS will contain each of the cells determined by a triple of points as shown in Figure 6. In this example, any two points in the left half of the bottom row, along with an arbitrary point in the left half of the top row, will determine a distinct regularly-spaced subset. It follows that in the worst case the output can be of size at least

$$\frac{n}{4}\binom{n/4}{2} = \Omega(n^3),$$

and this gives the desired lower bound on the time complexity of any algorithm for the AMRSS problem. □

The remainder of this section develops an optimal $\Theta(n^3)$ time algorithm for the AMRSS problem. We begin by considering the problem in two dimensions.

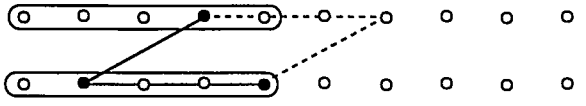

Figure 6. Example of a pointset containing $\Omega(n^3)$ distinct maximal regularly-spaced subsets: any pair of points in the left half of the bottom row, along with an arbitrary point in the left half of the top row, will determine a distinct regularly-spaced subset.

As in the discussion of AMESCS, we assume that the x_1 coordinates of the input points are unique (again via a rigid rotation if needed). Create the set T of all $\binom{n}{2}$ segments defined by pairs of input points, sorted by three keys:

(i) the length of the segment,
(ii) the slope of the segment, and
(iii) the x_1 coordinate of the (lower) left endpoint of the segment.

This preprocessing/sorting phase requires $O(n^2 \log n)$ time and allows us to determine the complete list of segments having a given length and slope at logarithmic cost per inquiry. Moreover, this returned list, which is contiguous in the sorted list T, will already be sorted by x_1 coordinates of the left endpoints (i.e., the third sort key). We use t_i to denote the ith segment in the sorted segment list T, and use $X_1(t)$ to denote the x_1 coordinate of the left endpoint of segment t.

Our algorithm proceeds by detecting pairs of equal-length parallel edges, since these define lattice cells. For each segment $t_i \in T$, we extract the (already-sorted) list Q of all $O(n)$ segments which have the same length and slope as t_i, and which also have left endpoint x_1 coordinate greater than or equal to $X_1(t_i)$ (note that $Q_1 = t_i$). Analogous to our solution for AMESCS, the idea is to find all pairs of adjacent congruent cells, and then overlap these pairs to determine maximal connected groups of cells.[4]

[4] Our algorithm actually outputs all maximal groups which contain two or more cells. To output isolated cells as well, one may simply list all pairs of segments which have identical slope and length; we showed earlier that the total number of such pairs is $O(n^3)$.

A pair of adjacent congruent cells is determined by three segments which have the same length and slope, and whose left endpoints form an equally-spaced triple of points. Assume that t_i is the leftmost segment of the three parallel segments that define a pair of adjacent congruent cells. Let $A = i$, and maintain two pointers B and C to segments in Q, with initially $B = i+1$ and $C = i+2$. If

$$X_l(Q_B) - X_l(Q_A) > X_l(Q_C) - X_l(Q_B),$$

we increment C by 1, otherwise we increment B by 1. Whenever the two differences are equal, the corresponding triple of segments (A, B, C) defines a pair of adjacent congruent cells, and we record this event. As in the AMESCS solution, we require linear time to find all pairs of adjacent congruent cells for which t_i is the leftmost of the three segments defining the pair of cells; iterating over all $\binom{n}{2}$ segments detects all pairs of adjacent congruent cells within $O(n^3)$ time. Since by Theorem 4 inputs can have a cubic number of adjacent cell pairs, this approach is asymptotically optimal.

Finally, we solve the 2-dimensional AMRSS problem by overlapping the cell pairs to determine all maximal connected groups of congruent cells. For each reported pair of cells u and v, we create nodes $\langle u \rangle$ and $\langle v \rangle$ and the edge $(\langle u \rangle, \langle v \rangle)$ in a graph. Each node in this graph has degree at most four, so the edge set and vertex set are both of size $O(n^3)$. Connected components in this graph correspond to maximal regularly-spaced subsets in the original pointset, and we can determine these using any linear-time connected components algorithm, after the edge list is converted into an adjacency list representation, as described above.

To solve AMRSS in higher dimensions, we project the input onto the x_1-x_2 plane, assuming without loss of generality that all x_1 and x_2 coordinates are distinct, and then solve the 2-dimensional AMRSS problem for the projected pointset. Congruent adjacent cells will correspond to congruent adjacent cells in the projection, and checking for spurious cells in the projection requires only constant time per cell pair in any fixed dimension. Since the number of congruent adjacent cells is bounded by $O(n^3)$ in all dimensions, the algorithms runs in time $O(n^3)$ in any fixed dimension. The optimal algorithm for AMRSS of an arbitrary pointset $P \subset E^d$ is given in Figure 7.

5. Conclusion

We have given an optimal algorithm and lower bounds for computing all order statistics of equally-spaced collinear subsets within a pointset in arbitrary dimension. The methods generalize to yield an optimal algorithm for determining all maximal regularly-spaced coplanar subsets in all dimensions. A longstanding open question remains whether even the existence of a single collinear triple can be detected in subquadratic time. It is

AMRSS Algorithm for arbitrary dimension

Input: A pointset P in E^d
Output: All maximal regularly-spaced subsets of P

Project P onto the x_1-x_2 plane
Rotate P if necessary so that all x_1, x_2 coordinates are unique
T = **Sort** $\{t = (p, q) \mid p \in P, q \in P\}$ by 3 keys: i) length(t), ii) slope(t), iii) $X_l(t)$
$G = (V, E) = (\emptyset, \emptyset)$
For $t \in T$ **Do**
  Q = **sort** $\{s \in T \mid$ slope(s) = slope(t), length(s) = length(t) $\}$ by $X_l(s)$
  $(A, B, C) = (1, 2, 3)$
  Until $C > |Q|$ **Do**
    If Q_A, Q_B, Q_C define two congruent adjacent cells **Then**
      $V = V \cup \{\langle A, B \rangle, \langle B, C \rangle\}$
      $E = E \cup \{(\langle A, B \rangle, \langle B, C \rangle)\}$
    If $X_l(Q_B) - X_l(Q_A) > X_l(Q_C) - X_l(Q_B)$ **Then** $C = C + 1$ **Else** $B = B + 1$
Convert G into adjacency-list format (using bucket sort)
Output the connected components of G

Figure 7. An optimal $O(n^3)$ algorithm for the AMRSS problem in arbitrary dimension.

also open whether an equally-spaced triple can be detected in subquadratic time; in this note we showed that quadratic time suffices to detect *all* equally-spaced triples. Finally, although we have shown that quadratic time suffices to detect *all* maximal collinear equally-spaced subsets, it is not known whether the detection of even a single maximal collinear equally-spaced subset can be accomplished in subquadratic time.

References

Ben-Tzvi, D. and M.B. Sandler (1990). A combinatorial Hough transform. *Pattern Recognition Letters* 11, 167–174.

Duda, R. and P. Hart (1972). Use of the Hough transform to detect lines and curves in pictures. *Comm. ACM* 15(1), 11–15.

Edelsbrunner, H. (1987). *Algorithms in Computational Geometry*. Springer, Berlin.

Edelsbrunner, H., J. O'Rourke and R. Seidel (1986). Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Comput.* 15, 341–363.

Edelsbrunner, H. and L.J. Guibas (1986). Topologically sweeping an arrangement. *Proc. ACM Symposium on Theory of Computing*, 389–403.

Preparata, F.P. and M.I. Shamos (1985). *Computational Geometry: An Introduction*. Springer, New York.

Risse, T. (1989). Hough transform for line recognition: complexity of evidence accumulation and cluster detection. *Computer Vision* 46, 327–345.