Optimal Algorithms for Determining Regularity in Pointsets (extended abstract)

Andrew Kahng and Gabriel Robins

UCLA CS Dept., Los Angeles, CA 90024-1596

Abstract

The problem of discerning spatial regularity in images yields elegant geometric problem formulations. Applications include texture analysis or finding rows of landmines, fenceposts, etc. We pose recognition of regularity as the problem of finding all maximal equally spaced collinear subsets of a pointset in E^d, and give an optimal $\Theta(n^2)$ time algorithm. We generalize this method to yield an optimal $\Theta(n^3)$ time algorithm for determining all maximal regular planar lattices within a pointset in E^d. Extensions to near-collinearity and near-equal spacing are also possible within this framework.

1 Introduction

Given a finite pointset $P \subset E^d$ with all points distinct, a subset $S \subseteq P$ is *collinear* if $|S| \geq 2$ and all points of S 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. Bucketing techniques based on the Hough transform [2] [7] [1] or other duality relations are often used for the MCS problem. However, such 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 subset $S \subseteq P$ an *equally spaced* collinear subset if $|S| \geq 3$ and all the points of S are equally spaced along the 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.

The MESCS problem has many practical applications, such as the examination of infrared ground surveillance bitmaps to find equally spaced collinear "hotspots" (rows of surface landmines, fenceposts in a region perimeter, etc.) Often, we would like to examine all possible regularities in an image, and therefore require a roster of all maximal equally spaced collinear subsets, i.e., the complete ***order statistics*** of the input with respect to the MESCS problem:

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

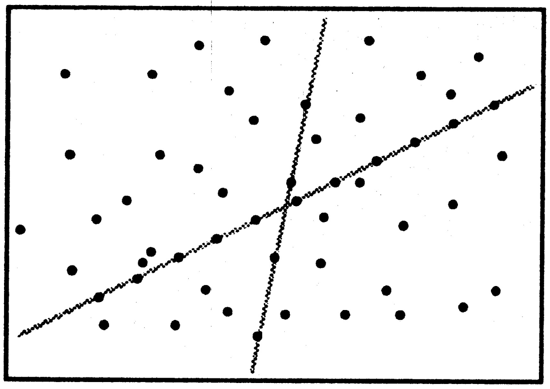

Figure 1: Two maximal equally spaced collinear subsets.

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 in [3], but the exponential dependence on dimension is expensive, and it is not clear that the method can satisfy the "equally spaced" constraint within the same time complexity. The work of Edelsbrunner and Guibas [4] also 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 also usually require time exponential in the dimension d or in the granularity of the bucketing. In general, methods in combinatorial geometry for finding spatial regularity in images (e.g., for texture classification) are dependent on low dimension, prescribed matching templates, or limits on the scale of features. In contrast, 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 two-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 problem (i.e., determining whether a given set of integers contains duplicates), which is known to require $\Omega(n \log n)$ time in the standard comparison model of computation [6], while a lower bound of $\Omega(n^2)$ for AMESCS is established based on the output size. Proofs of these results may be found in the full report [5]:

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

Theorem 2: AMESCS requires at least $\Omega(n^2)$ time.

3 Solving AMESCS Optimally

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.

This section gives an optimal algorithm for the AMESCS problem in arbitrary dimension. The method is based on a solution of the one-dimensional problem, i.e., finding all maximal arithmetic progressions in a set of numbers.

First, consider the very special one-dimensional variant which looks for an equally spaced *triple* of points, i.e., an arithmetic progression of length three (we can show that this problem, as well as the decision version of determining whether a given pointset contains a collinear triplet, also has an $\Omega(n \log n)$ lower bound). We find *all* equally spaced triples as follows. First, sort the input using $O(n\ log\ n)$ time, yielding a list $x_1, x_2, \ldots x_n$. Next, assume that the point at position $A = i$ in the sorted list is the leftmost point of a triple. We maintain pointers (B and C) to points in the sorted list, initially with $B = i+1$ and $C = i+2$. If $(B - A) > (C - B)$, we increment C by 1, otherwise we increment B by 1. Whenever the two differences are equal, we record the equally spaced triple (A,B,C). Because pointers B and C simply march along the sorted list, we find all equally spaced triples with x_i as leftmost component in linear time. Iterating over $i = 1, 2, \ldots n$ returns all equally spaced triples in $O(n^2)$ time, and since inputs can have a quadratic number of triples, this is asymptotically optimal.

The main idea is that the AMESCS problem can be solved by overlapping these triples in order to determine all maximal equally spaced chains. This is accomplished by constructing a graph where for each reported equally spaced triple $[x_i,\ x_j,\ x_k]$ we create the nodes $< i,j >$ and $< j,k >$ and the edge $(< i,j >\ ,\ < j,k >)$. 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 to determine connected components in this graph will yield all maximal equally spaced subsets within $O(n^2)$ time.

```
Rotate $P$ if needed so all $x_1$ are unique;
$S = P$ sorted by $x_1$ coordinates;
$G = (V, E) = (\emptyset, \emptyset)$;
For $A = 1$ to $n - 2$ do
  $B = A + 1$;
  $C = A + 2$;
  Until $C > n$ do
    If $S_A$, $S_B$, $S_C$ are collinear
       and equally-spaced Then
          $V = V \cup \{< A, B >, < B, C >\}$;
          $E = E \cup \{(< A, B >, < B, C >)\}$;
    If $x_1(S_B) - x_1(S_A) > x_1(S_C) - x_1(S_B)$
       Then $C = C + 1$ Else $B = B + 1$;
Convert G to adjacency-list format (bucket sort);
Output the connected components of G.
```

Figure 2: Optimal $O(n^2)$ algorithm for the AMESCS problem.

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 (E.g., rigidly rotate the pointset by a tiny angle θ so as to make all of the x_1 coordinates unique). 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 triplets in the projection will not correspond to actual equally spaced triplets, checking for spurious triples requires only constant time per triple in any fixed dimension. Since the number of equally spaced triplets is bounded by $\binom{n}{2}$ in all dimensions, our algorithm will run in time $O(n^2)$ for any fixed dimension. Our optimal AMESCS algorithm for an arbitrary pointset $P \subset E^d$ is formally given in Figure 2.

4 Generalization to Lattices

In this section, we generalize the notion of "equally-spaced" regularity to two dimensions. First, we consider lattices which are given by the intersection points of two (different-sloped) families of equally-spaced parallel lines. These families are formally specified by two linear equations in general form with integer parameters j and k, respectively; an added constraint $ad \neq bc$ on the constants in the linear equations insures that the two families are not parallel.

Definition: A *lattice* $L(x_0, y_0, a, b, c, d)$ for fixed real absolute constants x_0, y_0, a, b, c, and d, $ad \neq bc$, is a pointset of the form $\{(x,y) \mid a(x-x_0)+b(y-y_0) = j \text{ and } c(x-x_0)+d(y-y_0)=k$ for some integers j and $k\}$; the lattice point (x,y) has *lattice coordinates* (j,k).

Definition: Given a fixed lattice $L(x_0, y_0, a, b, c, d)$, a set of points in the plane is a *sublattice* of L if it is a subset of 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 $\{(j,k),(j+1,k),(j,k+1),(j+1,k+1)\}$, and the cell itself has *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.

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

We may define a graph structure over the cells in a sublattice which is induced by the neighbor relation:

Definition: Given a fixed lattice L, the *cell graph* of a sublattice $L' \subseteq L$ is defined by $G(L') = (V,E) = (\{<p> \mid p \text{ is a cell in } L'\}, \{(<p>,<q>) \mid p \text{ and } q \text{ are cells in } L', p \diamond q\})$.

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. Now we may state the following problems:

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

Again, we want to find *all* order statistics:

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

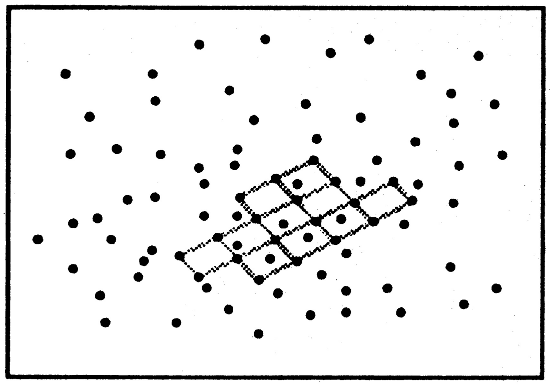
Figure 3: A maximal regularly-spaced subset.

Figure 3 illustrates a pointset 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, and also a worst-case lower bound of $\Omega(n^3)$ for any AMRSS algorithm [5]:

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 $O(n^3)$.

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

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. 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, and sort T by three keys: (a) segment length, (b) the slope of the segment, and (c) 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, the 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). Denote the x_1 coordinate of the left endpoint of the segment t_i by $e(t_i)$.

For each such segment t_i, we extract the already-sorted list Q of all $O(n)$ segments having the same length and slope, and with left endpoint x_1 coordinate greater than or equal to $e(t_i)$ (i.e., $Q_1 = t_i$). Analogous to our solution for AMESCS, the main idea is to find all pairs of adjacent congruent cells, and then overlap these pairs to determine maximal connected groups of cells.

A pair of adjacent congruent cells is determined by three segments having the same length and slope, and with left endpoints forming an equally-spaced triple. Assume that t_i is the leftmost segment A in a pair of adjacent congruent cells, and maintain two pointers B and C to segments in Q, with initially $B = 2$ and $C = 3$. If $e(Q_B) - e(Q_A) > e(Q_C) - e(Q_B)$, 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 use linear time to find all pairs of adjacent congruent cells with t_i as the leftmost of the three segments defining the pair of cells; iterating over all $\binom{n}{2}$ segments t_i reports all pairs of adjacent congruent cells within $O(n^3)$ time. Since inputs can have a cubic number of adjacent cell pairs, this is asymptotically optimal.

Finally, we solve the AMRSS problem by overlapping the cell pairs to determine all maximal connected groups of congruent cells. This is accomplished by constructing a graph where for each reported pair of cells C_i and C_j we create the nodes $< i >$ and $< j >$ and the edge $(< i > , < j >)$. Each node in this graph has degree of 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 now 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, again assuming without loss of generality that all x_1 and x_2 coordinates are distinct, 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 again requires only constant time per cell pair for any fixed dimension. Since the number of congruent adjacent cells is bounded by $O(n^3)$ in all dimensions, the algorithm runs in time $O(n^3)$ for any fixed dimension. Thus, the optimal algorithm for AMRSS of an arbitrary pointset $P \in E^d$ is given formally in Figure 4.

5 Conclusion

We address the problem of finding geometric regularity in pointsets in arbitrary dimension. We prove lower bounds and give an optimal algorithm for computing all order statistics of equally-spaced collinear subsets. This generalizes to yield a lower bound and an optimal algorithm for determining all maximal regularly-spaced planar subsets in all dimensions. Extensions to near-equal spacing or near-collinearity are possible within this framework.

```
Project $P$ onto the $x_1 - x_2$ plane;
Rotate $P$ if needed so all $x_1$, $x_2$ are unique;
$T' = \{(p,q) \mid p \in P, q \in P\}$;
$T = T'$ sorted by the three keys:
    1) $dist(p,q)$, 2) $slope(p,q)$, 3) $min_{x_1}(p,q)$;
$G = (V,E) = (\emptyset, \emptyset)$;
For $t \in T$ do
  $Q = \{s \in T \mid slope(s) = slope(t),\ x_1(s) > x_1(t),$
      $length(s) = length(t)\}$ sorted by $x_1$ coord;
  $A = 1$; $B = 2$; $C = 3$;
  Until $C > |Q|$ do
    If $Q_A$, $Q_B$, $Q_C$ define 2 congruent adjacent cells
      Then $V = V \cup \{<A,B>, <B,C>\}$;
           $E = E \cup \{(<A,B>, <B,C>)\}$;
    If $e(Q_B) - e(Q_A) > e(Q_C) - e(Q_B)$
      Then $C = C + 1$ Else $B = B + 1$;
Convert G to adjacency-list format (bucket sort);
Output the connected components of G.
```

Figure 4: An optimal $O(n^3)$ algorithm for the AMRSS problem.

References

[1] Ben-Tzvi, D. and M. B. Sandler (1990). A Combinatorial Hough Transform. *Pattern Recog. Lett.* 11, 167-174.

[2] Duda, R. and P. Hart (1972). Use of the Hough Transform to Detect Lines and Curves in Pictures. *Communications of the ACM* 15(1), 11-15.

[3] Edelsbrunner, H. (1987). *Algorithms in Computational Geometry*, Springer-Verlag, Berlin.

[4] Edelsbrunner, H. and L. J. Guibas (1986). Topologically Sweeping an Arrangement. *Proc. ACM Symposium on Theory of Computing* , 389-403.

[5] Kahng, A. and G. Robins (1990). *technical report.* CSD-900045, UCLA CS Department.

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

[7] Risse, T. (1989). Hough Transform for Line Recognition: Complexity of Evidence Accumulation and Cluster Detection. *Computer Vision* 46, 327-345.