Published in Image Processing On Line on 2015–12–15.
Submitted on 2014–09–19, accepted on 2015–10–09.
ISSN 2105–1232 © 2015 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2015.126

An Unsupervised Point Alignment Detection Algorithm

José Lezama[1], Gregory Randall[2], Jean-Michel Morel[3], Rafael Grompone von Gioi[4]

[1] CMLA, ENS Cachan, France (lezama@cmla.ens-cachan.fr)
[2] IIE, Universidad de la República, Uruguay (randall@fing.edu.uy)
[3] CMLA, ENS Cachan, France (morel@cmla.ens-cachan.fr)
[4] CMLA, ENS Cachan, France (grompone@cmla.ens-cachan.fr)

Abstract

In this article we present an algorithm for the detection of perceptually relevant alignments of points in 2D point patterns. Our algorithm is based on the *a contrario* detection theory. It requires no parameter tuning and has only one critical parameter, which controls the number of false detections.

Source Code

The ANSI C reviewed source code for this algorithm is available at the web page of this article[1]. Compilation and usage instructions are included in the `README.txt` file of the archive.

Keywords: A contrario; clustering; point alignments; Gestalt theory

1 Introduction

In this article we describe an algorithm for automatically finding collinear subsets within a planar set of points. This problem arises in many contexts of data analysis [13, 5, 6, 9]: alignments are among the simplest structures observable in a point set and 3D alignments are viewpoint-invariant structures (the projection of aligned points in 3D are always aligned points in 2D). The fundamentals behind this algorithm are presented in [10], and a successful application of the algorithm to the problem of vanishing points detection is presented in [9].

While it may seem that point alignments are simple structures, Figure 1 shows how complex an alignment event can be. From a purely factual point of view, the same alignment is present in the three figures. However, it is only perceived as such by most viewers in the first one. The second and the third figures illustrate two occurrences of the *masking phenomenon* discovered by gestaltists [8]: the *masking by texture*, which occurs when a geometric structure is surrounded by a clutter of randomly distributed similar objects or *distractors*, and the *masking by structure*, which happens when the structure is masked by other perceptually more relevant structures, a phenomenon also called *perceptual conflict* by gestaltists [12]. The magic disappearance of the alignment in the second and third figures can be accounted for in two very different ways. For the first one, a probabilistic *a*

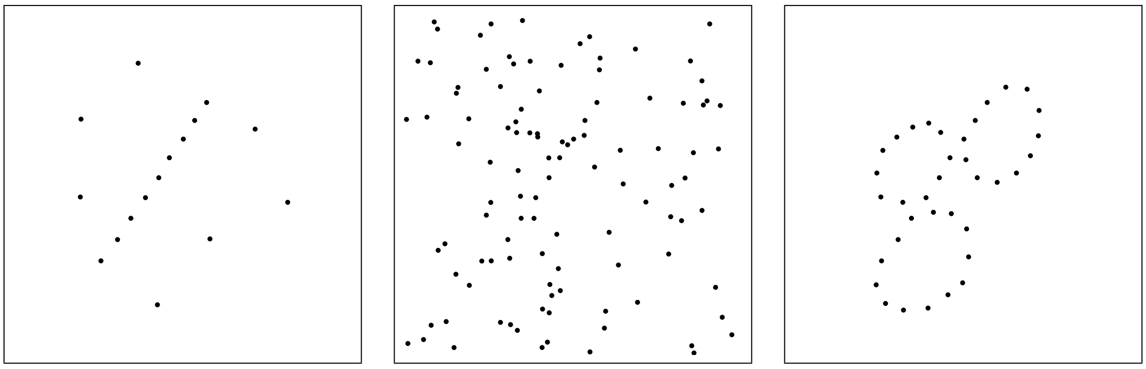

Figure 1: Illustration of the complexity of the point alignment problem: exactly the same set of aligned dots is present in the three images, but it is only perceived as such in the leftmost one. The center one is a classic "masking by texture" case and the one on the right shows a "masking by structure", often called "Gestalt conflict".

contrario model [3] can lead to a quantitative prediction. The second one is explained by the winning intervention of three more powerful grouping laws, *good continuation*, *convexity* and *closure* in the perceptual conflict [8].

Here we develop a method derived from the *a contrario* methodology proposed by Desolneux, Moisan and Morel [2, 3]. The *a contrario* framework is a mathematical formalization of the *non-accidentalness principle* proposed for perception [15, 1, 14] (sometimes called *Helmholtz principle*). In a nutshell, an observed structure is relevant if it would rarely occur by chance. This scheme has been repeatedly used in the past. In the words of David Lowe, "we need to determine the probability that each relation in the image could have arisen by accident, $P(a)$. Naturally, the smaller that this value is, the more likely the relation is to have a causal interpretation" [11, p.39]. The difference in the *a contrario* methodology is that the expectation of the number of false detections is controlled instead of the probability of observing a false detection. The resulting statistical framework provides estimates of significance similar in spirit to the methods mentioned before.

To handle the possible variety of alignments and cope with obvious objections and counterexamples, the algorithm takes into account: a) a *local* Poisson density estimation, b) an evaluation of the *regularity of the spacing* of the points in the alignment, and c) a criterion to *select the best interpretation among redundant detections.* The method comprises the search and validation of candidate point alignments. A candidate alignment consists of a thin rectangle in the image and the idea is to evaluate if the points density inside the rectangle is high with respect to the background. To provide a local estimation of the background density, a rectangular window around the rectangle will be used to count surrounding points. Also, in order to evaluate the regularity of points spacing inside a candidate rectangle, it will be divided into equally shaped intervals or boxes and the number of occupied boxes will be counted. A candidate alignment will be validated if the number of occupied boxes is statistically significant given the local point density. The last step of the algorithm consists of a redundancy reduction step, where non-maximal detections are removed.

The rest of this article is organized as follows: Section 2 describes the methodology of the alignment detector, including the redundancy reduction procedure. Section 3 describes the algorithm, with implementation details and an analysis of the computational complexity. Finally, Section 4 shows commented experiments.

[1]http://dx.doi.org/10.5201/ipol.2015.126

2 Method Description

2.1 Candidate Rectangles

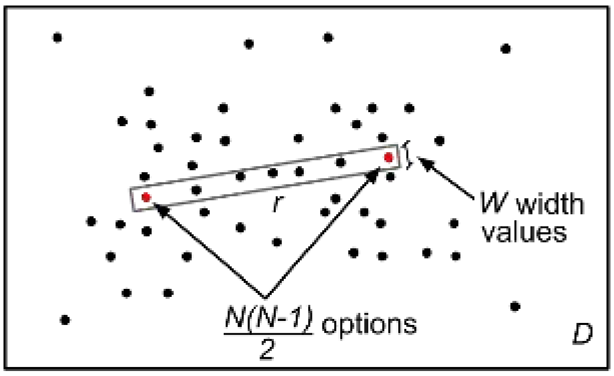

Figure 2: A schematic representation of the evaluated rectangle. In a domain with N points, and considering W possible widths, there are $\frac{N(N-1)}{2}W$ possible rectangles. In this example, $N = 49$ and 5 among them are inside the rectangle r.

The main body of the proposed algorithm consists of two parts: the search for candidate alignments and their validation. The algorithm is completed by a redundancy reduction or non-maximum suppression step.

Consider a set $\mathbf{X}$ of N 2D points defined in a domain D, see Figure 2. We are interested in detecting groups of points that are well aligned. Candidate alignments will be formed by taking every pair of points and constructing a rectangle r, whose major axis ends at each point, and with variable width. For a particular problem, one may have reasons to restrict the shape of rectangles. Nevertheless, this inquiry is not aimed at any particular application. Thus, we will rely on the following general criteria. An alignment should be an elongated structure, so a minimal ratio between the length and the width of the rectangle must be fixed. Then, a fixed number of widths must be tested, decreasing geometrically from the maximal width (the choice of a geometric series is justified by the obvious scale invariance of the detection problem). Our implementation uses a minimum length/width ratio of 10 and a geometric series of 8 width values with a factor $1/\sqrt{2}$. The total number of widths to be tested will be denoted by W.

2.2 Density Estimation

In order to evaluate if a configuration of points forms a statistically significant alignment, the method requires an estimation of the density of points in the figure. When the point density is estimated globally in the entire image domain, regions of different density can produce false detections, as shown in Figure 3(b). In this example, the global density is an average of the density of the two very distinct regions. Thus, any subset of points from the high density region will be considered very dense compared to the global density. To avoid this, the algorithm computes a *local* estimation of the points density.

For each rectangle, a local window is defined inside which the points density will be estimated, as shown in Figure 4 (left). The algorithm considers different sizes for this local window. The length of the local window is fixed to the length of the rectangle, and the width is varied. The largest window is square of side equal to the length of the alignment. Then a fixed number $L = 8$ of widths in geometric series are tried. The choice for a geometric series with factor $1/\sqrt{2}$ is again justified by the scale invariance of the detection problem.

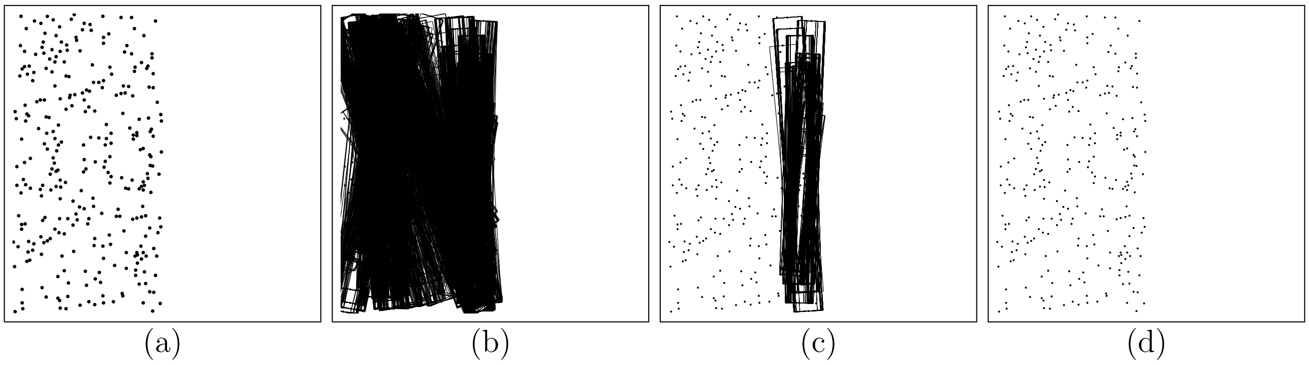

Figure 3: Local vs. global density estimation. **(a)** The set of points. **(b)** Alignments found using global density estimation. The many detected rectangles indeed have a high point density compared to the average image density used as background model. **(c)** Alignments found using local density estimation. The local density is lower on the border, hence the deceptive detection. **(d)** No alignment is found when the local density is estimated by the maximum density on both sides of the alignment. This figure, and a detailed explanation on how the detections in it were obtained can be found in [10].

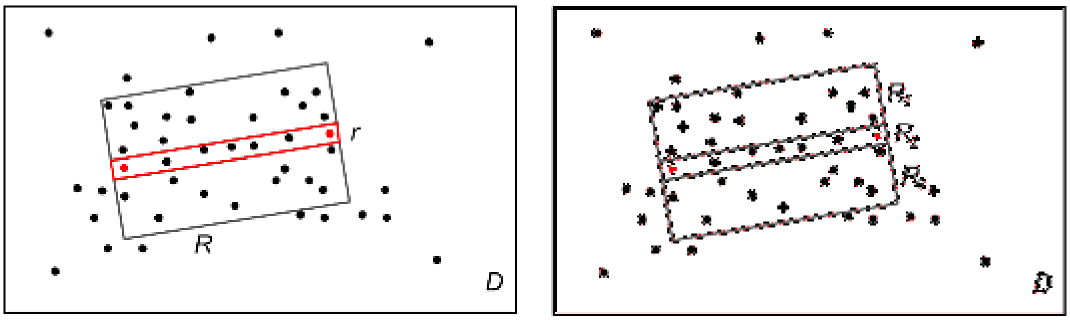

Figure 4: Left: the point density is estimated in the local window R surrounding the alignment r. Right: to refine the estimation, the density of points is measured on each side of the evaluated rectangle. The maximum of the densities in R_1 and R_3 is taken as an estimation of the point density in both R_1 and R_3. R_2 corresponds to r. This figure was taken from [10].

In order to avoid a border effect that causes detections in the boundaries of high-density regions, as shown in Figure 3(c), the algorithm separately measures the point density on each side of the candidate rectangle, and considers the density of the local window to be the maximum lateral density, as shown in Figure 4 (right). To compute the final density estimation, the local window is divided in three parts: R_1, R_2 and R_3. R_1 is the rectangle formed by the area of the local window on one side of the alignment, R_3 is the area of the local window on the other side, and R_2 is the rectangle which forms the candidate alignment ($R_2 = r$). Next, given the set of input points $\mathbf{X}$, the algorithm counts the numbers of points M_1, M_2, and M_3 in R_1, R_2 and R_3, respectively, and defines the conservative estimate of the local number of points as

$$n^*(R, \mathbf{X}) = 2\max(M_1, M_3) + M_2. \tag{1}$$

Once the conservative counting of points is done, the points density is given by this quantity and the local window area. The aim of the algorithm is to detect if a statistically surprising number of those points are concentrated in the inner rectangle.

Note that when the rectangle to be tested lies near the border of the domain, the local window may be partly outside it, where no point information is available, leading to a wrong density estimation.

This also happens when the rectangle covers the diagonal of the domain. A symmetric extension of the point set across the domain boundary is used to estimate the point density in windows meeting the outside. The candidates are still selected among the original points.

2.3 Points Regularity

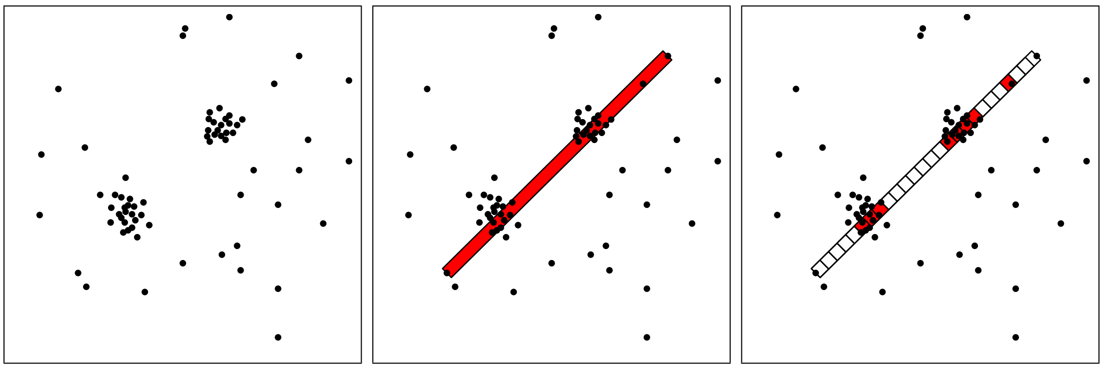
Figure 5: Left: dot pattern with two point clusters but no alignment. Center: a thin rectangle with a high point density was found, but it would constitute a false detection. Right: the algorithm divides the rectangle into boxes and counts the occupied ones, avoiding this misleading cluster effect. The occupied boxes are marked in red. In this case, no alignment is detected.

Besides a local increase of points density, the proposed method aims at recovering alignments where points are regularly spaced, or where the points density is uniform inside the inner rectangle. The idea is to counter the fact that the presence of clusters of points can produce zones of higher density but not true alignments, as shown in Figure 5. To avoid this, the final step in the construction of a candidate alignment consists in dividing the rectangle r into c equally sized boxes. Then, instead of counting the number of points inside r, the algorithm counts the number of boxes occupied by at least one point, see Figure 5 (right). If a small cluster of points occupies a box, it will only account for one occupied box, the same as if it were just one point.

A set $\mathcal{C}$ of different values are tried for the number of boxes c into which the rectangle is divided. In practice $\#\mathcal{C} = \sqrt{N}$ is set. Each candidate rectangle is divided into c boxes, with c going from 1 to $\sqrt{N}$.

2.4 Candidate Validation

In Sections 2.1 to 2.3 we described how to construct the set of candidate alignments. To summarize, candidate alignments are constructed by considering rectangles formed by every pair of points with different widths, then different widths for the local window estimation and finally different number of boxes to divide the inner rectangle into. In this section we shall describe how each candidate will be validated as a detection or not.

The validation is done using the *a contrario* framework. The *a contrario* methodology consists in assuming a null hypothesis H_0 for the data, where no detections should occur, and finding violations of this hypothesis (i.e. events that could hardly happen under the hypothesized model). In the proposed method, the null hypothesis is that the points are drawn from a uniform random distribution (a Poisson process) in the input domain. Note that this is a good hypothesis for modeling unstructured points, since their relative positions are random and independent.

The fundamental quantity of the *a contrario* approach is the Number of False Alarms (NFA) associated with an event e and a set of points $\mathbf{X}$. Given an *a contrario* hypothesis H_0 for the set of points in the domain, the NFA of an event is a bound on the expected number of occurrences of the event under H_0. Given an event e, its NFA is

$$\mathrm{NFA} = N_{tests} \cdot \mathbb{P}_{H_0}(e), \tag{2}$$

where N_{tests} is the number of tested events.

Let us now consider a candidate rectangle r inside a local estimation window R, and divided into c boxes. The algorithm first counts the number of points in the local window considering the maximum lateral density (Section 2.2), thus $n^* = \max(M_1, M_3) \times 2 + M_2$, where M_1 and M_3 are the number of points in the portion of the local window to the left and right of r respectively (R_1 and R_3), and M_2 is the number of points in r (see Figure 4). If b is the number of occupied boxes among c, then the algorithm computes the probability that at least b out of c boxes are occupied when n^* points are randomly distributed inside the local domain R. Let us call p_0 the probability of one box being occupied by at least one point

$$p_0(r, R, c) = 1 - \left(1 - \frac{S_b}{S_R}\right)^{n^*}, \tag{3}$$

where S_b and S_R are the areas of one box and of the local window R respectively, and n^* is the conservative number of points from (1). Then, the probability of at least b out of c boxes being occupied is

$$p = \mathcal{B}(c, b, p_0), \tag{4}$$

where $\mathcal{B}$ is the tail of the binomial distribution[2]. This formula corresponds to the probability of observing the event "at least b of c boxes are occupied in r, given that n^* points are randomly distributed in R".

To obtain the NFA, we need to multiply the probability computed above by the number of tests N_{tests}. To obtain the number of tests, recall that we first consider each pair of points to generate a candidate alignment, this gives a total of $N(N-2)/2$ pairs. Then, we consider W different widths for the rectangle r and L different sizes of the local estimation window (Section 2.2). Finally, we consider C different numbers of boxes into which r is divided (Section 2.3). Thus, the number of tested events is

$$N_{tests} = \frac{N(N-1)}{2} \cdot W \cdot L \cdot C. \tag{5}$$

Having computed the previous quantities, the NFA of the alignment event becomes

$$\mathrm{NFA}(r, R, c, \mathbf{X}) = \frac{N(N-1)}{2} \cdot W \cdot L \cdot C \cdot \mathcal{B}(c, b(r, c, \mathbf{X}), p_0(r, R, c)). \tag{6}$$

Here one would like to keep the alignments that are statistically significant: those with low NFA, or in other words those who are unlikely to happen by chance under the *a contrario* model. Thus, a threshold ε is set on the NFA. The algorithm computes the NFA for each evaluated rectangle and validates only those with NFA $\leq \varepsilon$. Usually the value of ε is set to 1, which means that in average, only one false detection will occur in a random data set.

2.5 Redundancy Reduction

Suppose a significant alignment is formed by a set of points A taken from $\mathbf{X}$. It is possible (and very likely) that a subset of those points, B, constitutes a significant alignment as well. To obtain a valid final result, this redundancy should be eliminated and only the most significant detections be kept. To this end, the following Masking Principle is introduced:

[2] $\mathcal{B}(n, k, p) = \sum_{j=k}^{n} \binom{n}{j} p^j (1-p)^{n-j}$

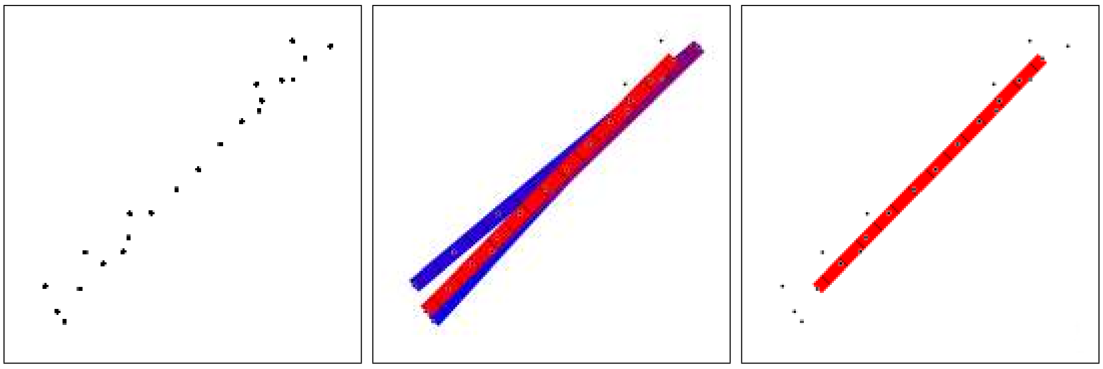
Figure 6: Redundant detections. Left: point pattern. Center: alignments found by the alignment detector. Red means the most meaningful and blue the least meaningful detections. Right: result of the redundancy reduction process. The alignments in the back cease to be meaningful when the points from the most significant alignment are removed. This figure was taken from [10].

Definition 1 (Masking Principle)**.** *A meaningful structure B will be said "masked by a structure A" if B is no longer meaningful when evaluated without counting its building elements belonging to A. In such a situation, the structure B is not retained as detected.*

In short, a meaningful structure should be detected *if* it is not masked by any other more meaningful detected structure. A procedural way to attain this result is to validate alignments one by one, starting by the one with smallest NFA (most meaningful). Before accepting a new alignment, it is checked that it is not masked by any one of the previously validated alignments. To check if an alignment B is masked by another alignment A, the algorithm recomputes the NFA of B using its same structure (rectangle r_B, divided into c_B boxes and local window R_B) but removing the points belonging to A. If the new NFA is no longer significant ($\leq \varepsilon$), then B is said to be masked by A. An example result of this procedure is shown in Figure 6.

3 Algorithm

3.1 Main Algorithm

Algorithm 1: Main body of the algorithm

```
input  : A set $\mathbf{X}$ of $N$ points [$W = 8$, $L = 8$, $\mathcal{C} = \{1 \cdots \lfloor\sqrt{N}\rfloor\}$ , $\varepsilon = 1$]
output: A list $\mathbf{m}$ of non-redundant point alignments
1 Create a symmetric extension of $\mathbf{X}$;
2 $\mathbf{l}$ = detect_alignments($\mathbf{X}$);
3 $\mathbf{m}$ = redundancy_reduction($\mathbf{l}$);
```

- Line 1: The purpose of this line is to create a data structure containing a symmetric reflection of the points in the input domain. This will be used when the local density estimation window falls outside the input domain. In that case, the points in the extension will be used for density estimation. Note that they will not be used to create candidate alignments.
- Line 2: Described in Algorithm 2
- Line 3: Described in Algorithm 3

3.2 Point Alignment Detector

Algorithm 2: Point alignment detector

```
input : A set $\mathbf{X}$ of $N$ points $[W = 8,\ L = 8,\ \mathcal{C} = \{1 \cdots \lfloor\sqrt{N}\rfloor\},\ \varepsilon = 1]$
output: A list $\mathbf{l}$ of point alignments

1  for $i = 1$ to $N$ do
2      for $j = 1$ to $i - 1$ do
3          $l \leftarrow$ distance$(\mathbf{x}_i, \mathbf{x}_j)$;
4          $w \leftarrow l/10$;
5          for 1 to $W$ do
6              $r \leftarrow$ rect$(\mathbf{x}_i, \mathbf{x}_j, w)$;
7              $w_L \leftarrow l$; // width of the local window
8              for 1 to $L$ do
9                  $R_1 \leftarrow$ local-win-left$(\mathbf{x}_i, \mathbf{x}_j, w_L)$;
10                 $R_3 \leftarrow$ local-win-right$(\mathbf{x}_i, \mathbf{x}_j, w_L)$;
11                 Count $M_1$, $M_2$, $M_3$ the number of points in $R_1$, $r$ and $R_3$ respectively;
12                 Compute $n^*(R, \mathbf{X})$ [Equation (1)];
13                 for $c \in \mathcal{C}$ do
14                     Divide $r$ into $c$ equal boxes;
15                     Compute $p_0(r, R, c)$ [Equation (3)];
16                     Count $b(r, c, \mathbf{X})$, the number of occupied boxes;
17                     Compute NFA$(r, R, c, \mathbf{X})$ [Equation (6)];
18                     if NFA$(r, R, c, \mathbf{X}) \leq \varepsilon$ then
19                         $\mathbf{l} \leftarrow r$;
20                     end
21                 end
22                 $w_L \leftarrow w_L/\sqrt{2}$;
23             end
24             $w \leftarrow w/\sqrt{2}$;
25         end
```

Algorithm 2 describes the pseudocode for the search and validation procedure. Following are some additional comments:

- Line 6: r is a rectangle whose main axis is the segment $\overline{\mathbf{x}_i\mathbf{x}_j}$ and whose width is w.
- Lines 9 and 10: R_1 and R_3 are depicted in Figure 4 (right).
- Line 11: The two points defining the statistical test must not be counted. In the present implementation, the counting of points inside the rectangles is based on the horizontal and vertical distances of each point to the segment $\overline{\mathbf{x}_i\mathbf{x}_j}$. A symmetric extension of the points is used when the local window falls outside the input domain (see Section 2.2). As a reference, the point counting procedure is done in the following way:

```
    m1 = 0;
    m2 = 0;
    m3 = 0;

    dx = xj-xi;
    dy = yj-yi;
    len = sqrt(dx*dx+dy*dy);

    for(m=0; m<N; m++)
    {
      if( m==i || m==j ) continue; /* do not count i and j */

      /* compute coordinates relative to alignments */
      x =  dx * (points[2*m]-x1) + dy * (points[2*m+1]-y1);
      y = -dy * (points[2*m]-x1) + dx * (points[2*m+1]-y1);

      /* count local window points */
      if( x < 0.0 || x >= len ) continue; // horiz. outside R
      if( y < -l/2.0 || y > l/2.0 ) continue; // vert. outside R
      if( y < -w/2.0 ) ++m1; // R_1 count
      else if( y > w/2.0 ) ++m3; // R_3 count
      else ++m2;
    }
```

Where `(xi,yi)` and `(xj,yj)` are the coordinates of the pair of points $\mathbf{x}_i$ and $\mathbf{x}_j$ forming the alignment, `N` is the total number of points, `w` and `l` are the widths of the rectangle and local window respectively. `points` is an array of length $2N$ containing all points coordinates, such that `points[2*m]` and `points[2*m+1]` are the horizontal and vertical coordinates of the m$^{\text{th}}$ point respectively. Please note that this code is a simplified version of the source code, so variable names in the source code might differ.

- Line 13: Recall that $\mathcal{C}$ contains the natural numbers from 1 to $\lfloor\sqrt{N}\rfloor$. However, in the present implementation, once the number of points inside the alignment is known, the algorithm starts from $c = n/2$, where n is the number of points inside the alignment and goes up to $c = min(2n, N)$. This is only an acceleration heuristic and does not affect the detector performance.

- Line 15: Note that n^*, the conservative estimation of points in R, is required to compute p_0.

- Line 16: When the rectangle is divided into c boxes, two half-boxes are left at each extreme of the alignment, so that c full boxes remain in the middle (see Figure 7). There are two reasons for this. One is that the points defining the statistical test must not be counted in the test. The second reason is that if there are c points perfectly spaced inside the alignment, then these would fall exactly in the center of each of the c boxes. As a reference, the present implementation counts the number of occupied boxes in the following way:

```
    box = len / (c+1); // takes into account one half-box at each extreme

    for(m=0; m<n_alignment; m++)
    {
      b = floor( (x[m] - box/2.0) / box );
      if( b>=0 && b<c ) ++occupied[b];
    }
```

Where `len` is the length of the alignment, `n_alignment` is the number of points inside the alignment, `x` is an array of length `n_alignment` containing horizontal distances inside the alignment and `occupied` is an array of length `c` initialized with zeros that will contain a positive integer where a box is occupied or zero otherwise. Please note that this code is a simplified version of the source code, so variable names in the source code might differ.

- Line 17: To simplify the computation of the NFA value, Equation (6), our implementation computes a bound to the tail of the binomial distribution instead of its actual value. Given n

independent Bernoulli random variables with parameter p, its sum $S = X_1 + \cdots + X_n$ follows the binomial distribution. It is easy to obtain [3] an upper bound to the tail of this distribution using Hoeffding's inequality [7]

$$\mathcal{B}(n,k,p) = \mathbb{P}(S \geq k) \leq \left(\frac{p}{k/n}\right)^k \left(\frac{1-p}{1-k/n}\right)^{n-k}.$$

The following code computes the previous approximation and returns the logarithm with base 10 of the result:

```
static double log_bin(int n, int k, double p)
{
  double r = (double) k / (double) n;
  if( r <= p ) return 0.0;
  else if( n == k) return k * log10(p);
  else return k * log10(p/r) + (n-k) * log10( (1-p)/(1-r) );
}
```

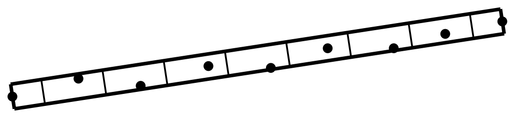

Figure 7: A detail of the counting of occupied boxes. Two half boxes are placed at the extremes of the rectangle, and only the full boxes in the middle count. This is because the two points forming the alignment must not be counted in the statistical test since they are the ones that define the test.

3.3 Redundancy Reduction

Algorithm 3 presents the pseudocode for the redundancy reduction procedure. Following are some additional comments:

- Lines 4 and 7: Event A is defined by the rectangle r_A, local window R_A and the number of boxes c_A. Event B is defined by the rectangle r_B, local window R_B and the number of boxes c_B.
- Line 8: A new set of points $\mathbf{X}'$ is considered, by removing all points belonging to alignment A (all points in r_A).
- Line 10: The NFA of B is re-evaluated using the set of points $\mathbf{X}'$. If the new NFA is greater than ε, then B is no longer significant when the points from A are removed: A masks B.
- Line 14: This condition means that the current alignment was not masked by any event in the current output list (all of which are more significant than itself). If the condition is satisfied, the current alignment is appended to the output list.

Algorithm 3: Redundancy reduction (Masking Principle)

```
input  : A list $\mathbf{l}$ of all significant alignments
output: A list $\mathbf{m}$ of maximally significant alignments
$\mathbf{l} \leftarrow$ sort($\mathbf{l}$);  // by NFA, lowest to highest
$\mathbf{m}[0] \leftarrow \mathbf{l}[0]$;
for $i = 1 \rightarrow$ length($\mathbf{l}$) $- 1$ do
    $B = \mathbf{l}[i]$ ;
    masked = false;
    for $j = 0 \rightarrow$ length($\mathbf{m}$) $- 1$ do
        $A = \mathbf{m}[j]$;
        $\mathbf{X}' = \{\mathbf{x} \mid \mathbf{x} \notin r_A\}$ ;
        if NFA$(r_B, R_B, c_B, \mathbf{X}') > \varepsilon$ [Equation (6)] then
            masked = true ;
            break;
        end
    end
    if masked == false then
        $\mathbf{m} \Leftarrow \mathbf{l}[i]$;
    end
end
```

3.4 Computational Complexity

The proposed method consists of an exhaustive search for candidates and their validation, including the subsequent redundancy elimination. Algorithm 2 describes the exhaustive search. The number of tests performed is the theoretical number in (5). To obtain the total complexity, this number must be multiplied by the complexity of a single test. To compute the NFA the algorithm needs to evaluate whether each of the N points belongs to a box or not. Thus, the complexity of a single test is proportional to the number of points N. Finally, the total complexity of the exhaustive search is $O(\sqrt{N}N^3)$, where the $\sqrt{N}$ comes from the set of number of boxes tested (C). In the final redundancy reduction step, described in Algorithm 3, the validated candidates (n_{val}) need to be compared to those already selected as final detections (n_{out}). This final step has a complexity of $O(n_{val}n_{out}N)$, typically much smaller than the complexity of the exhaustive search.

All in all, the method has a complexity of $O(\sqrt{N}N^3)$. This polynomial time limits the number of points that can be handled in practice. With current computers a few thousands points can be processed in a few minutes. One possible strategy for acceleration is computing a fast list of candidates, instead of all the possible pairs of points. To this end, a Gaussian mixtures algorithm such as [4] may be used as an efficient heuristic to propose candidates.

4 Experiments

Figure 8 shows the results of the algorithm in the examples shown in Figure 1. In Figure 8(a), the detector correctly found the alignment. In Figure 8(b), where the alignment is masked by noise, no detection was found because the added points satisfy exactly the H_0 model, and the original alignment contains too few points to stand out in the noise. In Figure 8(c), the alignment found by the algorithm is correct, but as discussed before, does not correspond to the most common interpretation by a human observer. A natural way of handling this problem would be to detect the

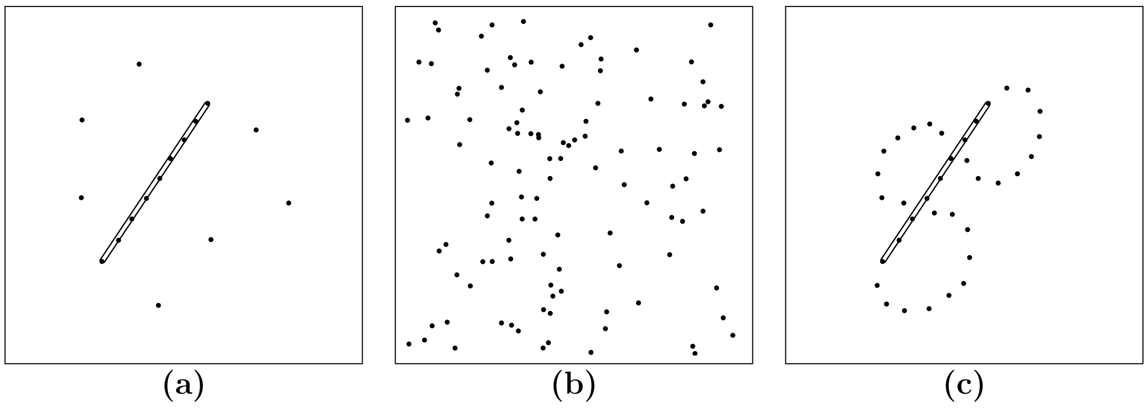

Figure 8: Results of the alignment detector in the examples of Figure 1.

"curves" by good continuation and then forcing a global interpretation by methods similar to our masking methodology, that would discard the detected alignment as "masked" by the curves.

Figure 9 shows the results of the proposed algorithm for a few more example data sets. The two leftmost examples contain only uniformly and independently distributed random points generated using MATLAB's random number generator. No detection was produced in the random points, regardless of the number of points in the dataset, which is perceptually correct. The three alignments in the rightmost figure were found. Note how the method correctly handled the redundant detections. Some further results of the method, with increasing difficulty, are shown in Figure 10, where the figures were correctly solved. Notice how the very low relative density alignments in Figure 10 (right) were correctly detected.

Figure 11 presents a case illustrating one important feature of the redundancy reduction procedure or "masking principle". The advantage of this approach is that one point can belong to more than one meaningful structure. For example, let us consider one point at the intersection of one horizontal and one vertical alignment. When the points from one alignment are removed from the other to check if it is masked, the second alignment loses one point but it can still be meaningful. Thus, all the vertical and horizontal alignments are kept.

In Figure 12, we present example figures that show the limitations of the algorithm. All the alignments in Figure 12(a) were found; however, the redundancy reduction step did not select the candidates covering the complete alignments. Also, a global interpretation of the figure is lacking but requires other detection tools. In Figure 12(b) the presence of a large cluster masks an alignment: it increases artificially the number of tests and short segments are no longer detected. Handling this example probably implies a round cluster detector but also a recursive approach: once a cluster is detected and removed, the algorithm should easily detect both alignments. Finally, two of the structures in Figure 12(c) are slightly curved, rendering them inaccurate as alignments; they were therefore not detected. Again, the detection of good continuation may provide a solution. In short, no alignment detection algorithm can be fully satisfactory *per se*; it requires the interaction (and conflicts) with other feature detectors, namely cluster and curve detectors.

In this experimental section we have used synthetic datasets that show clearly the behavior of the algorithm, but similar effects can be observed in real data [9].

Acknowledgement

Work supported by the European Research Council (advanced grant Twelve Labours n° 246961) and the Office of Naval research (ONR grant N00014-14-1-0023).

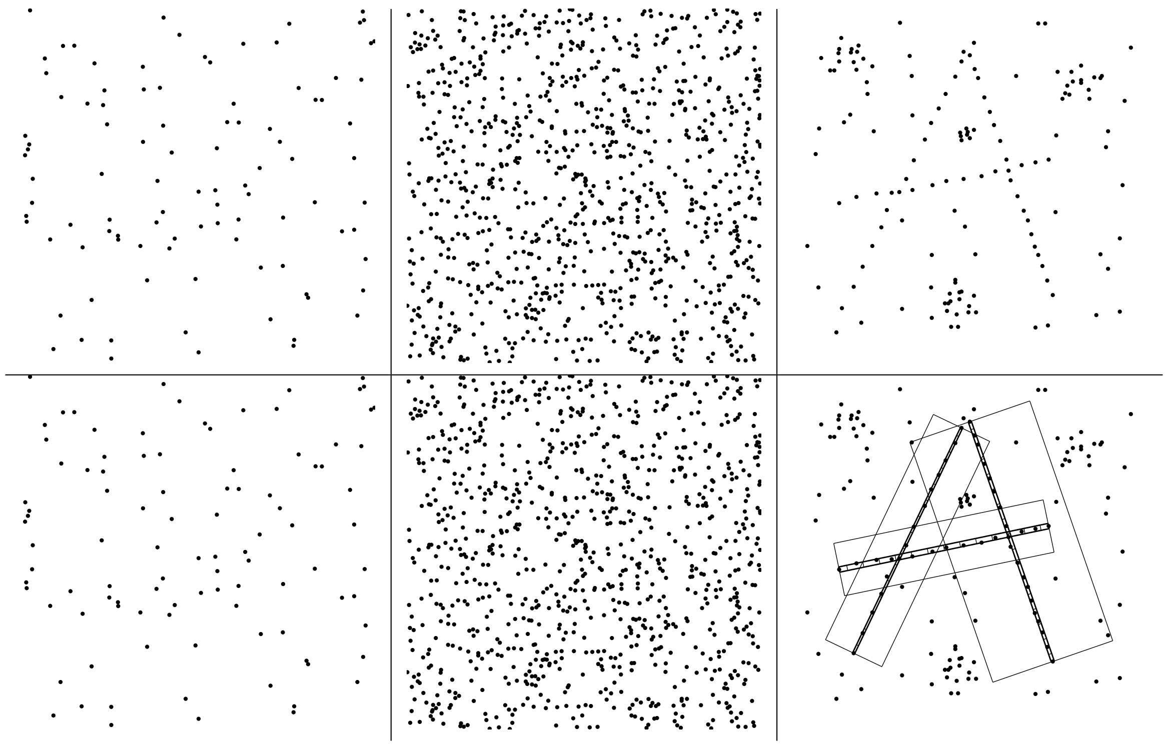

Figure 9: Top row: input points. Bottom row: result of the proposed algorithm. Each detection is represented by a thin rectangle divided into boxes, and surrounded by the local window.

Image Credits

All images by the authors.

References

[1] M. K. Albert and D. D. Hoffman, *Genericity in spatial vision*, in Geometric Representations of Perceptual Phenomena: Arts. in Honor of Tarow Indow's 70th Birthday, D. Luce, K. Romney, D. Hoffman, and D'Zmura M., eds., Erlbaum, 1995, pp. 95–112.

[2] A. Desolneux, L. Moisan, and J.-M. Morel, *Meaningful alignments*, International Journal of Computer Vision, 40 (2000), pp. 7–23. http://dx.doi.org/10.1023/A:1026593302236.

[3] ——, *From Gestalt Theory to Image Analysis, a Probabilistic Approach*, Springer, 2008. ISBN 9780387743783.

[4] M. A. T. Figueiredo and A. K. Jain, *Unsupervised learning of finite mixture models*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (2002), pp. 381–396. http://dx.doi.org/10.1109/34.990138.

[5] P. Hall, N. Tajvidi, and Malin P. E., *Locating lines among scattered points*, Bernoulli, 12 (2006), pp. 821–839. ISSN 13507265.

[6] Ø. Hammer, *New statistical methods for detecting point alignments*, Computers & Geosciences, 35 (2009), pp. 659–666. http://dx.doi.org/10.1016/j.cageo.2008.03.012.

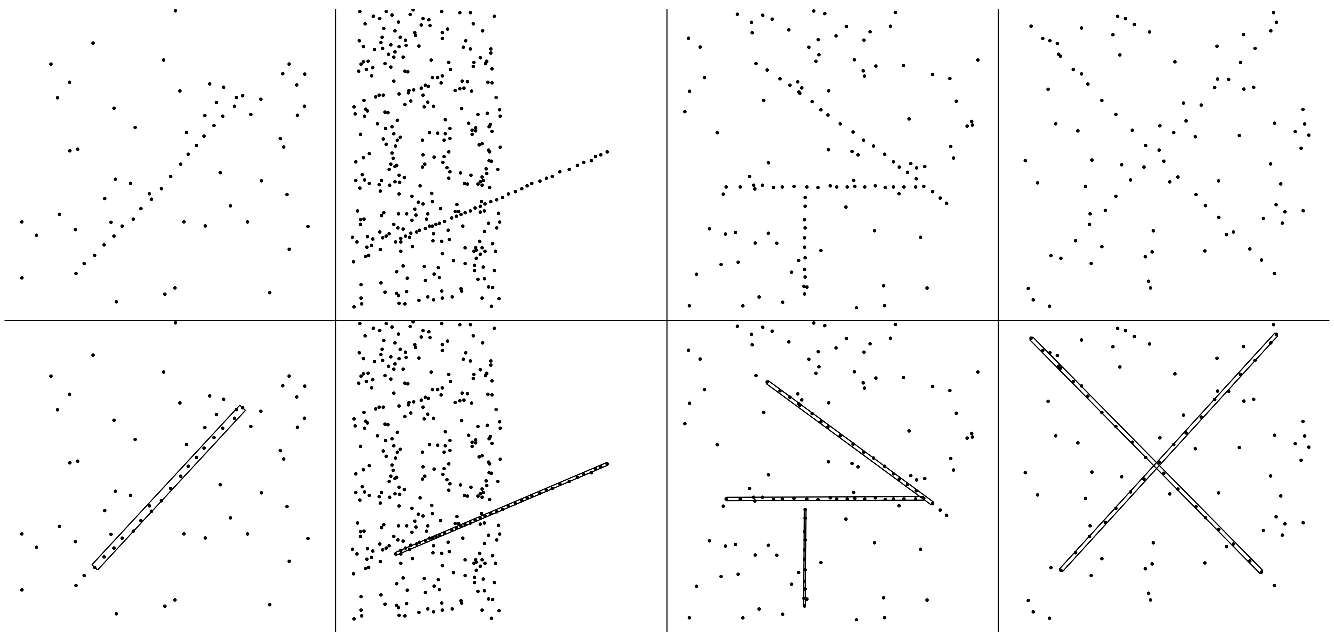
Figure 10: Additional results of the proposed algorithm. Top row: input points. Bottom row: result of the algorithm. The local window and the boxes are not drawn.

[7] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, Journal of the American Statistical Association, 58 (1963), pp. 13–30. http://dx.doi.org/10.1080/01621459.1963.10500830.

[8] G. Kanizsa, *Organization in Vision*, Praeger, 1979.

[9] J. Lezama, R. Grompone von Gioi, G. Randall, and J. M. Morel, *Finding vanishing points via point alignments in image primal and dual domains*, in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 509–515. http://dx.doi.org/10.1109/CVPR.2014.72.

[10] J. Lezama, J. M. Morel, G. Randall, and R. Grompone von Gioi, *A contrario 2d point alignment detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 37 (2015), pp. 499–512. http://dx.doi.org/10.1109/TPAMI.2014.2345389.

[11] D. Lowe, *Perceptual Organization and Visual Recognition*, Kluwer Academic Publishers, 1985. ISBN 146132551X, 9781461325512.

[12] W. Metzger, *Laws of Seeing*, The MIT Press, 2006 (originally 1936).

[13] C. G. Small, *Techniques of shape analysis on sets of points*, International Statistical Review, 56 (1988), pp. 243–257. http://dx.doi.org/10.2307/1403352.

[14] J. Wagemans, *Perceptual use of nonaccidental properties*, Canadian Journal of Psychology, 46 (1992), pp. 236–279. http://dx.doi.org/10.1037/h0084323.

[15] A. P. Witkin and J. M. Tenenbaum, *On the role of structure in vision*, in Human and Machine Vision, J. Beck, B. Hope, and A. Rosenfeld, eds., Academic Press, 1983, pp. 481–543. http://dx.doi.org/10.1016/B978-0-12-084320-6.50022-0.

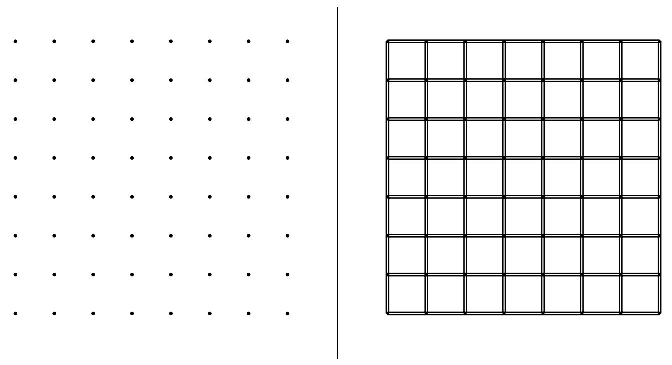

Figure 11: Left: input: an 8x8 grid of points. Right: alignments detected by the algorithm, after the redundancy reduction procedure. The proposed "masking principle" allows one point to belong to more than one structure.

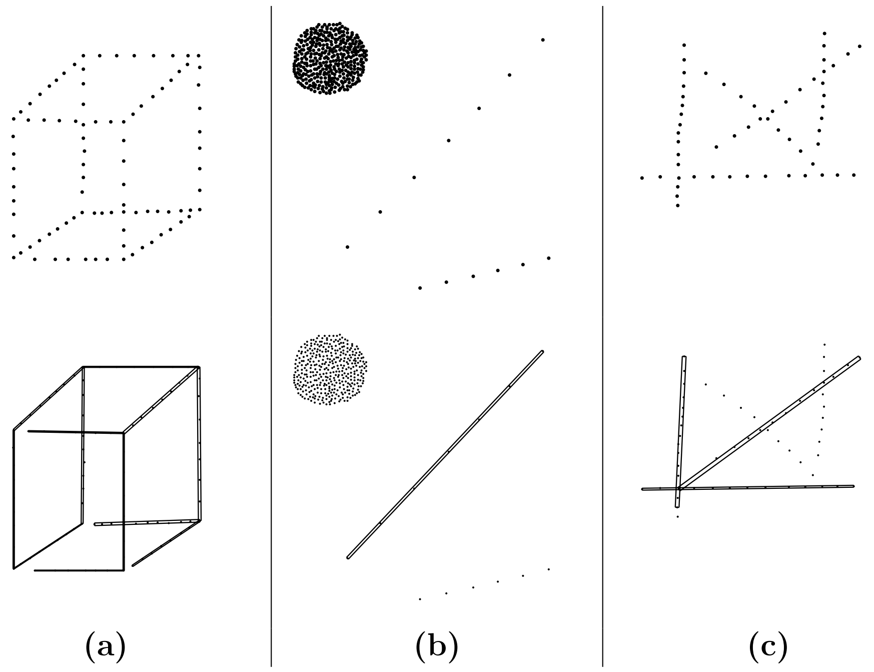

Figure 12: Examples imperfectly solved by the proposed algorithm. The local window and the boxes are not drawn. This figure was taken from [10].