Covering Simple Polygonal Regions by Ellipses *

Alon Efrat† Frank Hoffmann‡ Klaus Kriegel‡ Christof Schultz‡

Abstract

We study the problem of how to cover simple polygonal rectilinear regions by a small set of axis–parallel ellipses. This question is well motivated by a special pattern recognition task where one has to identify ellipse shaped protein spots in 2–dimensional electrophoresis images. We present and discuss two algorithmic approaches solving this problem: a greedy brute force method and a linear programming formulation.

1 Detecting Spots in 2–dimensional Gel Electrophoresis Images

1.1 Gel Electrophoresis: The Application Background

In proteomics 2–dimensional gel electrophoresis (2DE) is a widely used technique to separate proteins. A 2DE gel is the product of two separations performed sequentially in acrylamide gel media: isoelectric focusing as the first dimension and a separation by molecular size as the second dimension. A two-dimensional pattern of spots each representing a protein is the result of that process. Eventually, spots are made visible by staining or radiographic methods. By analysing series of such 2DE images one hopes to identify those proteins that change their expression (size, intensity) and reflect/cause certain biochemical and biomedical conditions of an organism, see [8].

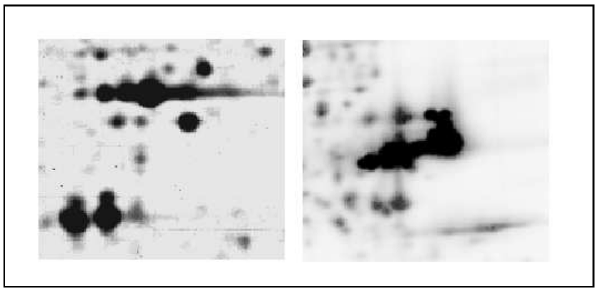

Figure 1: Twin spots, streaks and complex region

Ideally, in an gel image each spot has the shape of an axis–parallel ellipse. However, spots that are very close to each other can partially merge and form rather complicated regions.

At Freie Universität Berlin we have started a few years ago to develop a softare system CAROL (see [1]) that is able to perform local and global matching queries for gel images (given in GIF format). Its novel algorithmic idea was to avoid setting landmarks by hand. Instead, the matching between a source and a target image uses the history of the incremental Delaunay triangulation, [4], [5], of the target spots. To this end we assumed that images are already given as spot lists with each spot represented by point coordinates of its center and a real value describing its intensity, as provided e.g. by the PDQuest system.

Meanwhile it has become lucid that wrongly detected spots are the main obstacle towards a better performance of the matching tool compared to influence of pure geometric distortions.

---

*Part of a joint research project with Deutsches Herzzentrum Berlin, supported by Deutsche Forschungsgemeinschaft, grant FL 165/4–1.

†Computer Science Department, Stanford University, email:*alon@Graphics.Stanford.EDU*

‡Institut für Informatik, Freie Universität Berlin, Takustr. 9, D-14195 Berlin, email:{*hoffmann,kriegel,schultz*}@*inf.fu-berlin.de*

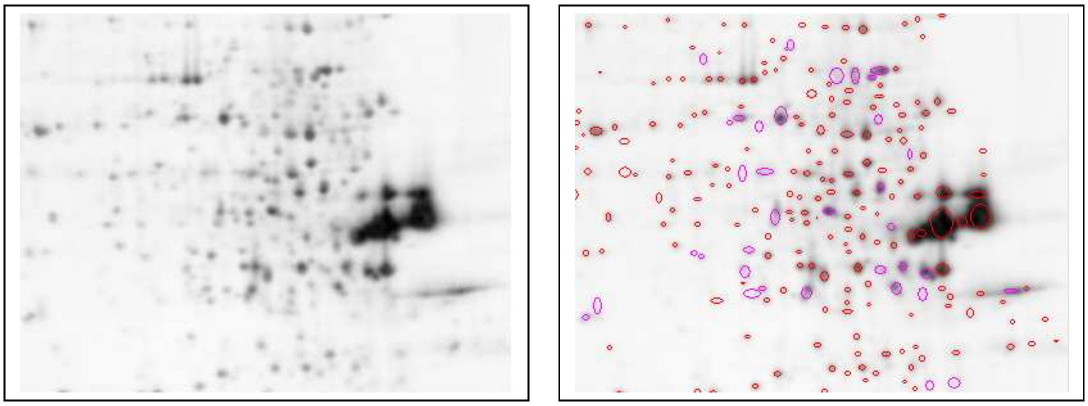

Figure 2: Part of a gel image and spots computed

That is why we decided to develop and include a new spot detection algorithm into the CAROL system to overcome these difficulties. In Figure 2 a part of a gel image is shown and aside the ellipses representing spots as computed by our spot detection algorithm.

This algorithm, see [7] for details, relies on a watershed transformation applied to the gradient image. The most difficult part is how to interpret twin spots, streaks (left side in Fig. 1) and so called *complex regions* (right side in Fig. 1) as unions of ellipses. In fact, in [7] the latter case was left open and the user had to edit these complex regions by hand. To present a solution to this question is the subject of this paper. In fact, just by looking at the images it is clear that there is an inherent uncertainty in the 2DE gel images due to the electrophoresis process itself which is highly susceptible to faults and distortions.

Consequently, in the recent CAROL version we cope with that problem by maintaining a list of proposals how such an ambiguous region could be covered instead of computing only the best covering. In the matching algorithm part we then have the possibility to accept the matching of two ambiguous regions if there is a pair of proposed coverings that match.

Figure 2 shows only 15 percent of the full 2DE gel image in GIF format. The total size is 811×900 pixel and our spot detection algorithm used about 9 sec to compute a total of 553 spots.

1.2 Modelling the Covering Problem

We assume that a simply connected pixel pattern R is given. In the application it is usually a subpattern of a 100×100–square. Let ∂R denote the polygonal curve describing its boundary.

Identifying a pixel with its center point p we define the two pixel sets $\partial R_- = \{p \in R | d_\infty(p, \partial R) = 3/2\}$ and, analogously, $\partial R_+ = \{p \notin R | d_\infty(p, \partial R) = 1/2\}$. Recall that an axis–parallel ellipse is formed by all points (x, y) fullfilling the equation

$$\frac{(x-c)^2}{a^2} + \frac{(y-d)^2}{b^2} - 1 = 0 \qquad (1)$$

with parameters $a, b, c, d \in \mathbb{R}$.

Now we can formulate the approximative covering problem we are interested in from the application point of view:

For a number $0 \leqslant p \leqslant 1$ and a natural number $\rho \geqslant 1$ find a smallest possible set $\mathcal{E}$ of axis–parallel ellipses fulfilling the following conditions:

1. (Shape) For each $E \in \mathcal{E}$ the ratio of its halfaxes is in the intervall $[1/\rho, \rho]$.

2. (Covering) $\bigcup_{E \in \mathcal{E}} E$ covers at lest a $1 - p$ portion of pixels in ∂R_-.

3. (Fitting) No ellipse $E \in \mathcal{E}$ contains an element from ∂R_+.

4. (Intersection) Any pair of ellipses $E, E' \in \mathcal{E}$ intersects in at most 2 points, and, we demand that $2 \cdot \text{area}(E \cap E') \leqslant \min\{\text{area}(E), \text{area}(E')\}$.

We emphasize once more that especially condition 4 is justified by the application because two spots can only partially overlap and do not form a cross. Moreover, in both solutions presented in the next section we represent ∂R_- by a smaller sample ∂R^*_-.

2 Brute Force Solution vs. Linear Programming

2.1 Computing a Brute Force Solution

The basic idea behind the brute force solution is to generate all ellipses that fulfill condition 1 and 3. A voting scheme is then set up to select the covering.
In a first step of the pixel set ∂R_- is approximated by a carefully (heuristically) chosen subset ∂R_-^* of size about $1/6$ of the original set.
The second step in the brute force approach is to discretise the search space of ellipses. To this end we restrict the ellipses to have centers which are pixel centers and one of the halfaxes, say a, has to have multiple pixel side length. Finally, for a fixed a we restrict the second halfaxis to values from the set $\{ax^k | 0 \leqslant k\}$, in the implementation we choose $k = 20, x = \sqrt[10]{5}$.
Now, the algorithm simply computes for each center (c, d) and for each k the maximal a such that the ellipse with parameters (a, ax^k, c, d) does not intersect the set ∂R_+.
For each ellipse we store which points from ∂R_-^* it covers and their number. Eventually, in a greedy fashion we choose the covering. Assume a partial covering is already chosen. The next ellipse E is selected among all ellipses satisfying condition 4 of a covering according to the following criteria, ordered as ranked:

1. Maximal number of previously uncovered points from ∂R_-^* covered by E
2. Length of longest chain of consecutive points from ∂R_-^* covered by E
3. E minimizes maximal intersection area with an already chosen ellipse.

After selecting a best E we update the scores of all other remaining ellipses by deleting the points covered by E. We stop augmenting ellipses when condition 2 is met.
An example is depicted in Figure 1, the region (subset of a 70×45–pixel array) shown is in fact part of the fat one from Figure 2. The set ∂R_-^* consists of 44 (shown in gray) pixels. Here, the brute force method generates about 13000 ellipses that fit into the region and which are then subject to the voting procedure. To compute the 8 ellipses in the solution takes about 3 sec on a SUN UltraSPARC 400MHz processor. The drawback of the brute force approach is obvious, too many ellipses are tested.

The followig observation is immediate from standard arguments for set covering greedy algorithm, see for example [2].
Fact: Let $\mathcal{F}$ be the family of ellipses (a, ax^k, c, d) as described above, and let k_{opt} be the minimal number of ellipses of $\mathcal{F}$, needed to cover all points of ∂R_-^* and avoiding the ones of ∂R_+. Then the number of ellipses reported by the brute force algorithm is $\leqslant k_{opt} \ln |\partial R_-^*|$.

2.2 Using an LP Approach to Generate Ellipses

Compared to the brute force approach we want to restrict the set of ellipses under consideration for covering a region. Observe that each element in the pixel set ∂R_+ forms a constraint for each ellipse in the cover, an ellipse must not contain such pixels. Since we do not want arbitrary small ellipses in the cover and we do not want degenerated halfaxes ratios as well, we can sample these 'outer' constraints by choosing every 4th point. Let us denote this sample by ∂R_+^*. On the other side we want at least a few points (say, at least three) from ∂R_-^* to be included in an ellipse.
Therefore we start from a randomly chosen triplet of mutually visible points from ∂R_-^* and ask whether there is an axis–parallel ellipse containing these points such that it does not violate an outer constraint. Once having the information that for a given triplet there is a feasible solution one can try to optimise it.
The idea how to make use of linear programming for testing feasibility is based on the observation that the parameters of an ellipse E can be transformed into variables of an LP in such a way that each of the three points which has to be covered by E adds a linear constraint to the outer constraints. The constraints are derived by putting a point $p = (p_x, p_y)$ into the ellipse formula $\frac{(p_x-c)^2}{a^2} + \frac{(p_y-d)}{b^2} - 1$ which has to be negative (positive) for points inside (outside) E. Since this is not a linear constraint we start with one of the form

$$p_x^2 x_1 - p_x x_2 + p_y^2 x_3 - p_y x_4 + x_5 \geqslant 0$$

By elementary calculation, setting $t = x_5 - \frac{x_2^2}{4x_1} - \frac{x_4^2}{4x_3}$, we derive as ellipse parameters

$$a = \sqrt{-\frac{t}{x_1}} \quad b = \sqrt{-\frac{t}{x_3}}$$

$$c = -\frac{x_2}{2x_1} \qquad d = -\frac{x_4}{2x_3}$$

Once having the existence of a feasible solution (not an explicite ellipse yet!) one wants to extend the point

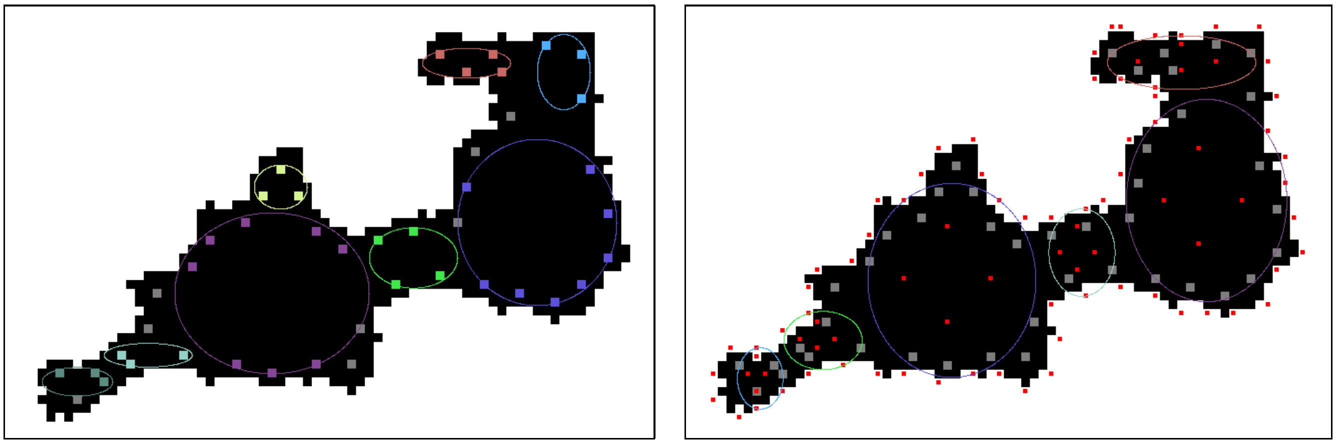

Figure 3: Ellipse covering computed by brute force method (left) and by LP approach (right)

set that can be covered. We started from the observation that an ellipse in a meaningful covering touches the boundary in at most 3 different places. Therefore it makes sense to add and check first the neighbors of the already chosen points. This can also be done in a random way. However, it should should also be clear that one can easily get stuck, when there is a very restrictive partial solution, in the sense that we have still a feasible system but the actual solution set is very small and does not allow to add a new point. To implement the necessary backtracking step, we have adapted a simplified version of the so called Metropolis methodology, see for example [6]. This random process is organized in rounds. With probability q we have a round 'extend', that is we choose and try to add a random neighbor; with probability $1 - q$ we have a round 'backtrack' and we delete a randomly chosen point from the already covered point set. After a fixed number of rounds we actually compute an explicite ellipse. Among all possible ellipses we choose that one yielding a halfaxes ratio closest to 1.

The only thing that remains to explain is how to incorporate condition 4 into the LP–approach. As before we delete already covered points from ∂R^*_-. But now for each selected ellipse E we add four new 'outer' constraints to ∂R^*_+. These are given by the mid points of the four halfaxes of E and they prevent later chosen ellipses from having large intersections with E. Again, this scheme is run until condition 2 is met.

In the example depicted in Figure 3 the gray pixels represent ∂R^*_-. The dots outside the region describe the initial outer constraints, within the chosen ellipses the additional outer constraints are also indicated. Together they form the final set ∂R^*_+. In the implementation we used the LP solver developed in [3].

References

[1] CAROL, Software system for Matching 2DE Gel Images, *http://gelmatching.inf.fu–berlin.de*

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms* MIT Press, 1990.

[3] B. Gärtner, S. Schönherr, An Efficient,Exact, and Generic Quadratic Programming Solver for Geometric Optimization, submitted

[4] F. Hoffmann, K. Kriegel and C. Wenk, Matching 2D Patterns of Protein Spots, in Proceedings 14th Annual ACM Symposium on Computational Geometry, June 7–10 1998, 231–239

[5] F. Hoffmann, K. Kriegel and C. Wenk, An applied pattern matching problem: comparing 2D patterns of protein spots, Discrete Applied Mathematics 93 (1999), 75–88

[6] M. Jerrum, Large Cliques Elude the Metropolis Process, Random Structures and Algorithms 3 (4), 1992, 347–359

[7] K.–P. Pleißner, F. Hoffmann, K. Kriegel, C. Wenk, S. Wegner, A. Sahlström, H. Oswald, H. Alt, E. Fleck, New algorithmic approaches to protein spot detection and pattern matching in two–dimensional electrophoresis gel databases, Electrophoresis 20 (1999), 755–765

[8] M. R. Wilkins, K. L. Williams, R. D. Appel, D. F. Hochstrasser (Eds.) *Proteome Research: New Frontiers in Functional Genomics*, Springer-Verlag Berlin Heidelberg New York, 1997