Geometric Algorithms for the Analysis of 2D–Electrophoresis Gels

Alon Efrat[*] Frank Hoffmann[†] Klaus Kriegel[†‡§] Christof Schultz[†§]
Carola Wenk[†¶]

Abstract

In proteomics 2–dimensional gel electrophoresis (2–DE) is a separation technique for proteins. The resulting protein spots can be identified by either using picking robots and subsequent mass spectrometry or by visual cross inspection of a new gel image with an already analyzed master gel. Difficulties especially arise from inherent noise and irregular geometric distortions in 2–DE images. Aiming at the automated analysis of large series of 2–DE images, or at the even more difficult interlaboratory gel comparisons, the bottleneck is to solve the two most basic algorithmic problems with high quality: Identifying protein spots and computing a matching between two images. For the development of the analysis software CAROL at Freie Universität Berlin we have reconsidered these two problems and obtained new solutions which rely on methods from computational geometry. Their novelties are: 1. Spot detection is also possible for complex regions formed by several "merged" (usually saturated) spots; 2. User–defined landmarks are not necessary for the matching. Furthermore, images for comparison are allowed to represent different parts of the entire protein pattern, which only partially "overlap". The implementation is done in a client server architecture to allow queries via the Internet. We also discuss and point at related theoretical questions in computational geometry.

1 Introduction

The proteomic research deals with the systematic analysis of complete profiles of the proteins expressed in a given cell, tissue or biological system at a given time. In this field, 2D–gel electrophoresis (2–DE) is a well–established and widely used technique to separate proteins in a sample. A 2–DE gel is the product of two sequentially performed separations in acrylamide gel media: isoelectric focusing as first dimension and a separation by molecular weight as second dimension. The result of that process is a 2D pattern of spots each representing a protein. There is a variety of staining methods to display protein spots: Coomassie blue and silver-staining and / or fluorescent and radioactive labeling. Mass spectrometry and the visual (usually computer aided) comparison (matching) with already analyzed gel images of such a sample are used to identify proteins. The visual analysis of such 2–DE image series intends to identify those proteins that change their expression (size, intensity) and reflect/cause certain biochemical and biomedical conditions of an organism, see [26]. However, this

[*] Computer Science Department, University of Arizona, email:*alon@CS.Arizona.EDU*

[†] Institut für Informatik, Freie Universität Berlin, Takustr. 9, D-14195 Berlin, email:*name@inf.fu-berlin.de*

[‡] Deutsches Herzzentrum Berlin, Augustenburger Platz 1, D-13353 Berlin

[§] Supported by Deutsche Forschungsgemeinschaft, grant FL 165/4–2.

[¶] Supported by Deutsche Forschungsgemeinschaft, grant AL 253/4–3.

requires high throughput analysis tools and the major challenge is to obtain both robust and reliable algorithmic solutions that work automatically, or at least need only little user interaction. As we will see, this goal in conjunction with efficiency requirements makes it necessary to start with simple (sometimes oversimplified) modelling assumptions.

A second strand of research was initiated in [22]. Here the challenge consists of creating analysis software that is able to perform interlaboratory gel image comparisons and to deal with images from various databases via the Internet.

The intensive research on these questions emphasizes the strong demand from the practitioners for better algorithmic solutions.

The aim of this paper is to demonstrate how ideas from computational geometry help to meet these challenges in the context of the two most basic problems in gel analysis:

- **Spot detection:** Given a scanned 2–DE gel image identify spot regions.
- **Gel matching:** Given two images by their spot lists, perform a matching procedure to identify those spot pairs that correspond to each other, so that the matching reflects both geometric and spot intensity resemblance of the images.

In this paper we survey algorithmic studies and implementation work that was done while developing the 2–DE analysis software system CAROL ([9]). It contains both new results (Section 2, Subsection 3.4) and partly published ones (Section 3, [17]). Interestingly, this application problem has also lead to new theoretical questions in geometry.

In Section 2 we address the spot detection question in the following context. Originally the CAROL project was designed to answer only matching queries for gel images. It turned out that available spot detection algorithms did not adequately support the underlying philosophy that is based on the similarity of geometric point patterns. Thus the spot detection approach presented here aims at determining the point pattern of a gel as precise as possible rather than computing exact shapes and volume data of spots. We assume that each spot has approximately the shape of an axis-parallel ellipse. This is a widely accepted, but surely simplified, modeling assumption, see for example [5] or [14]. However, spots that are very close to each other may partially overlap and "merge". This can lead to rather complicated local pixel regions, as depicted in Figure 1, compare [20].

In [23] a spot detection algorithm is described that relies on a watershed transformation applied to the gradient image. The most difficult part is the interpretation of twin spots, streaks (left side in Fig.1), and so–called *complex regions* (right side in Fig.1) as unions of ellipses. This especially applies to the case of oversaturated regions in silver stained gels.

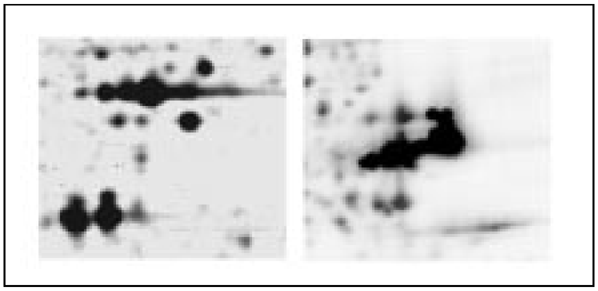

Figure 1: Twin spots, streaks, and a complex region

In fact, in [23] the latter case of interpreting complex regions was left open and in the implementation the user had to edit these complex regions by hand. Here we present an algorithmic solution to this question based on a Linear Programming (LP) formulation. To the best of our knowledge it is the first algorithmic attempt to deal with spot detection of complex oversaturated regions in a single image. An alternative approach has been studied in [6], where the information of multiple already aligned images is used for a subsequent spot detection.

There is an inherent uncertainty in the images due to the electrophoresis process itself which is highly susceptible to faults and geometric distortions. The reader may argue just by looking at the images, that there is no hope to come up with a perfect spot detection algorithm. This is certainly true. Yet we can cope with this ambiguity, at least to some extent by computing a ranked list of proposals how a complex region could be covered instead of computing only the "best" covering, which could be erroneous. Then, in the matching algorithm, we have the possibility to accept the matching of two ambiguous regions if there is a pair of proposed coverings that match.

The matching algorithm itself is described in Section 3. Assume that a source and a target image are given by their spot lists. We first transform the lists into geometric point patterns, where a spot is represented by the ellipse center coordinates and additionally we code the real size/intensity of the spot into a single integer value. The lists can be of significantly different size and, additionally, we allow the images to show only partially overlapping sections of the gels. Thus the task is to find a maximal–cardinality one–to–one matching of the spot lists, so that this mapping respects both the geometry of the images and the relative spot intensities i.e., relatively intensive source spots should be mapped to relatively intensive target spots. The geometric matching criterion is based on similarity of imaginary edges which link spots. Two edges are similar if their length ratio and the difference of their angles formed with the x–axis are within preset tolerance bounds. Our algorithmic solution, which we call global–via–local matching, works in two stages. Firstly, we compute matchings for several small local patterns, which are chosen in a grid–like fashion. For this local matching we use a modified alignment method known from point pattern matching ([2]) combined with geometric hashing. A significant speed–up is gained by reducing the set of all source / target edges to those belonging to the history of the incremental Delaunay triangulation of the spot lists in the order of their decreasing intensities. Having sets of matching candidates for each of the local patterns, we select a subset that is consistent with the overall grid topology. The matching spot pairs computed this way later serve as "landmarks" in the second stage. Landmarks were also used in previous algorithmic solutions, but they had to be set manually. The novelty of our solution is to generate them automatically. The subsequent extension from landmarks to a maximal global matching is rather standard by local neighborhood comparisons. Since this step uses mainly geometric information, it is capable to match spots with different intensities, that correspond to significantly regulated proteins. Moreover in the extension step it is possible to exploit the proposal lists for ambigious spot regions, as computed in the spot detection. This way, we present a first step towards the simulation of how an expert would compare images by visual inspection, namely, by intertwining the spot detection with the matching process.

Figure 2 shows two 2–DE gel images originated from a transfection experiment of cultured EaHy cells with 800 (left) and 1000 protein spots. Their original size is about 23cm by 29cm. For the purpose of illustration in Figure 3 and 4 more details are shown of the small rectangular window regions marked in Figure 2. (For simplicity ellipses were replaced by circles.)

In both Sections 2 and 3 we also refer to related theoretical issues and open problems in computational geometry that could further improve our solution. In Section 4 we outline implementation issues and describe experimental results.

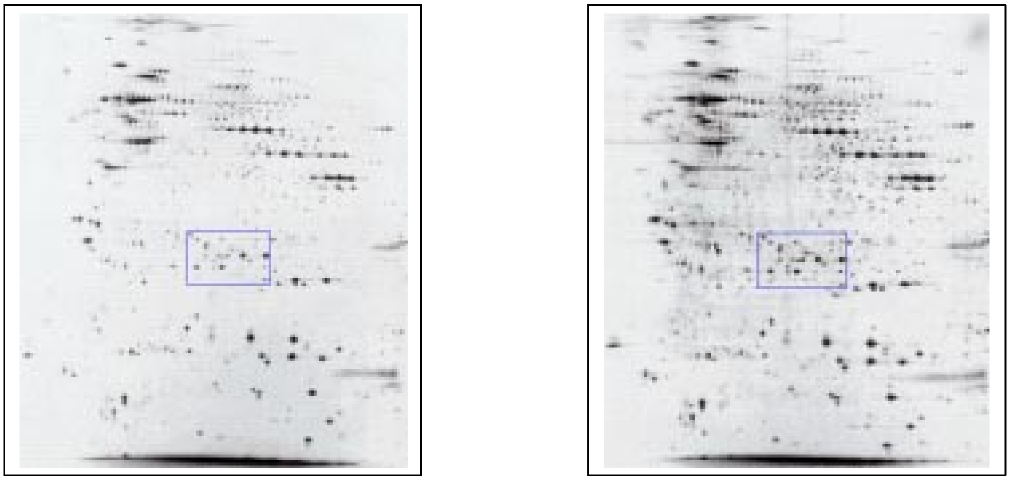

Figure 2: Two gel images of cultured EaHy cells from a transfection experiment

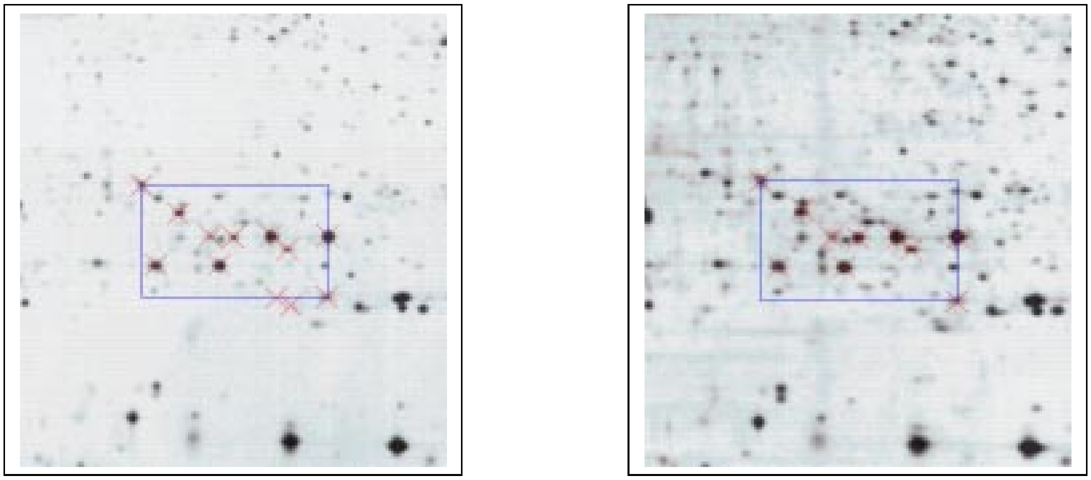

Figure 3: Detailed local images of Fig.2, a selected pattern on the left side and a partial matching

2 Spot Detection in Complex Regions as an Ellipse Covering Problem

2.1 Modeling of the Core Problem

Spot detection is to a wide extent an image processing problem. In fact, if the spots are well separated, classical methods like filtering and watershed transformation on the gradient image provide a satisfying algorithmic solution, see [23]. As mentioned above, the core combinatorial problem is the interpretation of complex pixel regions as unions of axis parallel ellipses.

The following formalization is a compromise stemming from discussions with practitioners who solve these covering instances by peer review. First, we remark that complex regions are typically saturated (black), and therefore gray level information does not help for the detection. We assume that a connected pixel pattern R is given, which is fat in the sense that there are neither short horizontal nor vertical cuts consisting of only two pixels. In the application R is usually a simply connected subpattern of a 100×100–pixel square.

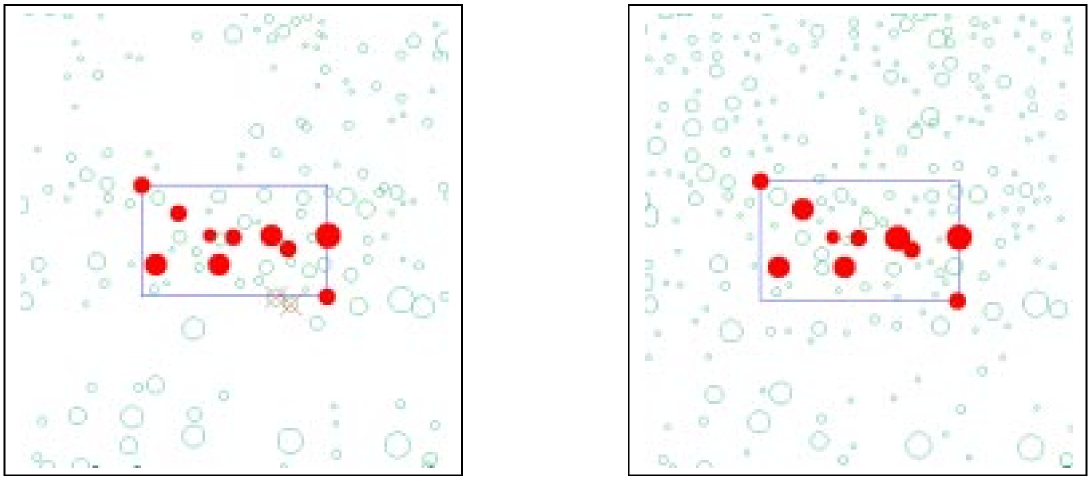

Figure 4: Spot sets detected with black spots indicating the local matching

To make use of the powerful machinery of geometric approximation algorithms we identify pixels with their center points. Then the region R can be represented by two sets of points C and F, where C is a sample of points to be *covered* (inside R) and F is a set of *forbidden* points (outside R). One can obtain these sets walking along the boundary of R and choosing points inside and outside within a small distance. This approach somehow mimics the general practice of experts who are looking for ellipses approximating long parts of the boundary of R. The advantage that both sets are of small cardinality has to be paid for by the fact that the computed cover could have some hole in the interior of the region. To avoid this one also can choose C as the set of all grid points in R (from an appropriately dense grid). Then, of course, the cardinality m of C can be quadratic in n, the size of F.

Now we can formulate the approximate covering problem from the application point of view. For numbers $0 \leq \delta, \epsilon \leq 1$ the problem is to find a smallest-cardinality set $\mathcal{E}$ of axis–parallel ellipses fulfilling the following conditions.

1. **(Fitting)** Each ellipse $E \in \mathcal{E}$ respects F, i.e., it does not intersect F.
2. **(Intersection)** The boundaries of every pair of ellipses E, E' in $\mathcal{E}$ intersects in at most 2 points, and $\text{area}(E \cap E') \leq \delta \cdot \min\{\text{area}(E), \text{area}(E')\}$.
3. **(Covering)** $\bigcup_{E \in \mathcal{E}} E$ covers at least a $(1 - \epsilon)$–portion of pixels in C.

The aim to find minimal–cardinality coverings is in accordance with Occam's razor principle, since the smallest cardinality set is, in absence of a master gel, the simplest hypothesis to explain how a complex region could have been evolved. At first sight one would require to cover all points in R, however in order to cope with noise the covering condition allows to leave a small portion of points uncovered. The fitting condition ensures that the selected ellipses do not cover much more than the region R. From the way the gel production process spreads a protein in the two dimensions, practitioners usually conclude that two spots do not form a cross. We incorporated this notion in the intersection condition.

Following the standard methodology of set cover approximation, our ellipse covering problem splits into two parts:

1. Find a basic set $\mathbf{E}$ of ellipses such that each ellipse $E \in \mathbf{E}$ fulfills the fitting property, and $R \subseteq \bigcup_{E \in \mathbf{E}} E$.

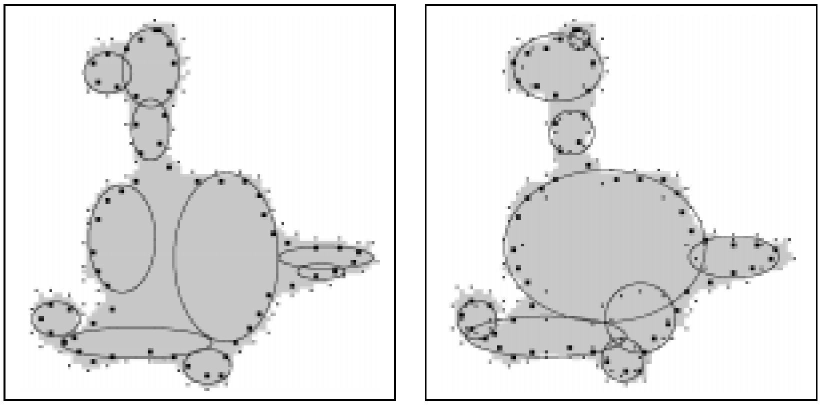

Figure 5: Ellipse covering computed by brute force method (left) and by LP approach (right)

2. Select a minimum cardinality subset $\mathcal{E} \subseteq \mathbf{E}$ which respects the intersection and covering properties.

Remark that (the boundary of) an axis–parallel ellipse can be represented by the solutions of the equation

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

with parameters $a, b, c, d \in I\!\!R$, where (c, d) is the ellipse center, and a and b are the lengths of the halfaxes. Thus a straightforward approach is to discretize the parameter space of all possible ellipses, and define $\mathbf{E}$ to be the set of all inclusion maximal ellipses fulfilling the fitting property. One way to avoid misinterpretation of streaks is to exclude ellipses of an ”extreme” shape, i.e., to require $a/b \in [1/\rho, \rho]$ for a suitable constant $\rho \geq 1$. This also reduces the complexity of $\mathbf{E}$. The selection of $\mathcal{E} \subseteq \mathbf{E}$ can then be easily done in a greedy fashion. We implemented this approach, however both runtime and the quality of the solution did not meet our expectations, see e.g. Figure 5. In Subsection 2.3 we study from a theoretical point of view how to improve on both, making use of advanced geometric data structures and techniques. In Subsection 2.2 we present an alternative LP-based approach, which avoids the explicit construction of a large basic set $\mathbf{E}$, and instead combines construction and selection of ellipses. Experimental results show that both quality and runtime of the LP-approach are superior to the simple brute force solution.

2.2 Using an LP Approach to Generate Ellipses

In the implementation we assumed that the region R is simply connected and rectilinear. However, it is straightforward how to extend the algorithms to arbitrary polygonal regions. Aiming at better runtimes, C and F are samples of those points inside and outside of R which are close (one pixel distance) to the boundary of R. The basic rule of the sampling heuristic we apply is to choose a denser sample for C (resp. F) in convex (resp. concave) boundary parts with high curvature.

Observe that each element in F forms an outer constraint for each ellipse in the cover because of the fitting condition. On the other hand, we want at least a few points (say, at least three) from C to be included in an ellipse of the covering. Therefore, we start from a randomly chosen triplet of mutually visible points from C and ask whether there is an axis–parallel ellipse containing these points so that

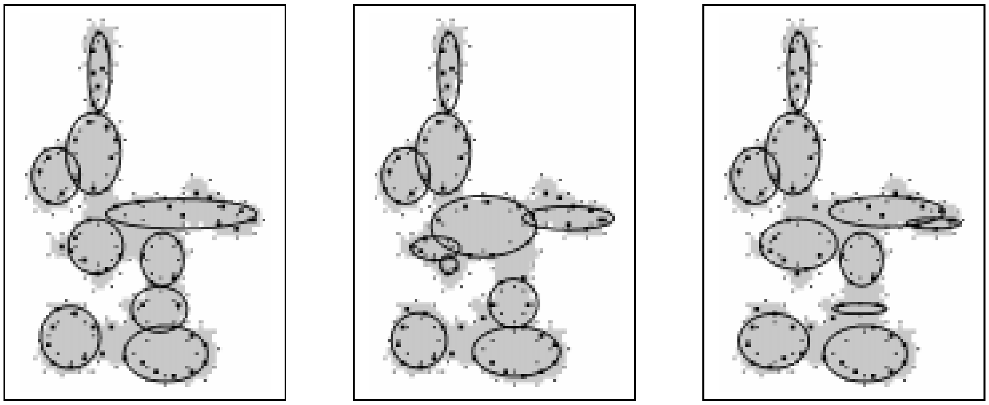

Figure 6: Three covering proposals for a complex region generated by the LP–approach

it does not violate an outer constraint. Once having the information that for a given triplet there is a feasible solution one can efficiently extend the covered sample subset by an LP–based approach.

We start from 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 have to be covered by E adds a linear constraint to the outer constraints. The constraints are derived by plugging a point $p = (p_x, p_y)$ into the ellipse formula $\frac{(p_x-c)^2}{a^2} + \frac{(p_y-d)^2}{b^2} - 1$ which has to be zero (negative, positive) for points on (inside, outside) E. Since this is not a linear constraint we start with one of the form

$$p_x x_1 - p_y^2 x_2 + p_y x_3 - x_4 = p_x^2 \qquad \text{(resp. } <, >\text{)} \tag{2}$$

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

$$a = \sqrt{t} \qquad b = \sqrt{\frac{t}{x_2}} \qquad c = \frac{x_1}{2} \qquad d = \frac{x_3}{2x_2} \qquad . \tag{3}$$

Given the existence of a feasible solution (not an explicit ellipse yet!) one wants to extend the point set that can be covered. Clearly, it makes sense to add and check first the neighbors of the already covered sample points. This is also done in a random way. However, it is 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 of ellipses is very small and does not allow to add a new sample point. To implement the necessary backtracking step, we have adapted a simplified version of the so–called Metropolis methodology, see for example [19], where it has been used for the generation of large cliques in graphs, a situation similar to ours.

This random Markov-chain like process is organized in rounds. With probability $q > 1/2$ 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" in which we delete a randomly chosen point from the already covered point set. After a fixed number of rounds we actually compute an explicit candidate ellipse to be included next into the partial covering. Remark that the shape of such an ellipse is almost completely determined by the covered points and by the outer constraints. To arrive at a unique solution, the LP-solver chooses that one with halfaxes ratio closest to 1. This is repeated a constant number of times and we select the candidate that covers the maximum number of points; ties are broken randomly.

There remains to explain the incorporation of the intersection condition into the LP–approach. We delete already covered points from C. For each selected ellipse E we add new "outer" constraints to

F by sampling the interior of E. These new points prevent later chosen ellipses from having large intersections with E. Again, this scheme is run until the covering condition is met. In the example depicted in Figure 6 the black pixels represent C. 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 F. Remark that, willing to spend more runtime, the quality of the covering can be improved by choosing a denser sample. Observe that, apart from the differences in the three coverings depicted in Figure 6, they in fact represent very similar point patterns when considering the centers of the ellipses only.

2.3 Related Theoretical Issues

We want to discuss the ellipse covering problem from a more theoretical point of view. As in Subsection 2.1 let a connected pixel pattern R be given. We identify a pixel with its center, and assume that pixel centers lie on an ϵ-grid. Let F be the set of grid points not in R that have distance ϵ to R. Let $n = |F|$, which yields $|R| = O(n^2)$. In this subsection we consider the case that the set C of points to be covered consists of all pixel centers in R, thus $R = C$.

The *ellipse covering problem* is to find a collection $\mathcal{E} = \{E_1, \ldots, E_k\}$ of minimal cardinality of axis-parallel open ellipses such that the union $U = \bigcup E_i$ covers R, and respects F. An ellipse *respects* F if it contains no point of F in its interior, however possibly on its boundary.

This covering problem is a simplified version of our original covering problem. It shares the fitting condition, and a special case ($\epsilon = 0$) of the covering condition. Note that this restriction can easily be relaxed by including ϵ into the stopping condition of the algorithm described below. Nevertheless it is still a challenge to incorporate an additional requirement like the intersection condition into the approximate set covering framework.

The problem of (approximately) covering a shape with ellipses is strictly related to the problem of exact covering of a shape with rectangles, which was shown to be NP-complete [12]. It is also related to the problem of covering a shape with strips [1], and to the range covering problem in a hypergraph [8]. Thus, in the general setting there is not much hope for finding a polynomial time algorithm. However it is possible to employ the Brönnimann & Goodrich paradigm [8] in order to obtain a polynomial time algorithm that computes a covering with ellipses defined by grid points, such that the cardinality of the cover is $k_{\text{opt}} \log k_{\text{opt}}$, where k_{opt} is the cardinality of the optimal-size solution. The runtime of a straightforward implementation of the paradigm is $O(n^5 k_{\text{opt}} \log n)$. In the sequel we present a faster implementation which runs in $\tilde{O}(n^3)$, where $\tilde{O}$ denotes a variant of the O-notation which subsumes polylogarithmic factors.

First observe that we can restrict our search to *maximal ellipses*. An ellipse E is *y-maximal* if (1) its center is in a grid point of R, it (2) does not intersect F in its interior, (3) among all ellipses with the same center and width E has the largest height. x-maximal ellipses are defined analogously. An ellipse is *maximal* if it is either x-maximal or y-maximal. Note that each maximal ellipse has a point of F on its boundary. A point of F on the boundary of an ellipse E is called a *dominator* of E. Clearly, we can assume that the optimal cover consists of maximal ellipses.

Let $\mathbf{E}$ be the set of all possible maximal ellipses with centers in R, and width or height defined by grid points. There are are $O(n^2)$ points p in R, and each p gives raise to $O(n)$ maximal ellipses centered at p, since there are $O(n)$ different possible widths and heights. Thus $|\mathbf{E}| = O(n^3)$. The rest of this section is dedicated to explain how to find a collection $\mathcal{E} = \{E_1 \ldots E_k\} \subseteq \mathbf{E}$ of cardinality $k = O(k_{\text{opt}} \log k_{\text{opt}})$ whose union respects F and covers R. Note that $k_{\text{opt}} = O(n)$ since one can choose a thin ellipse covering all points in a row or column of R between two points in F.

The Brönnimann & Goodrich approach assumes that k_{opt} is known, thus we assume in the sequel that k_{opt} is given. It can be determined afterwards by applying unbounded binary search. The overview of the Brönnimann & Goodrich algorithm is as follows: It gives a weight $w(E)$ to each ellipse $E \in \mathbf{E}$, initially all set to be 1. The algorithm consists of *rounds*. In each round a random sample $\mathcal{E}$ of size $ck_{\text{opt}} \log k_{\text{opt}}$ of the ellipses of $\mathbf{E}$ is picked, where the probability of an ellipse to be picked is proportional to its weight, and c is a constant. Next the algorithm checks if $\mathcal{E}$ covers R. If this is not the case, we find a point $q \in R$ which is not covered (arbitrarily chosen if there are more than one), and double the weight of each $E \in \mathcal{E}_q$, where $\mathcal{E}_q \subseteq \mathbf{E}$ is the set of ellipses containing q. Then the algorithm continues with the next round and stops when a cover is found. Since the Vapnik-Chervonenkis dimension of the problem is clearly finite, it follows from [8] that the expected number of rounds is $O(k_{\text{opt}} \log n)$.

In a somehow similar fashion to [1], we construct a data structure that enables us to (implicitely) modify the weight of all ellipses in $\mathcal{E}_q$. We describe the process for y-maximal ellipses. x-maximal ellipses can be handled in an analogous manner.

For $p \in R$ let $e_w(p)$ denote the y-maximal ellipse of width w whose center is in p. Let r be a row of grid points of R, and let w be a fixed width. Using the property that the boundaries of $e_w(p_1)$ and $e_w(p_2)$, $p_1, p_2 \in r$, intersect at most once above r, it is straightforward to show (c.f. [13]) that for a given $s \in F$ the region $\{p \in r | s \text{ is a dominator for } e_w(p)\}$ forms a connected interval on r. Similarly it follows that for $q \in R$ the set $I_{w,r}(q) := \{p \in r | q \in e_w(p)\}$ is also an interval on r. Now consider a partition of r into maximal connected intervals, such that in each interval the dominator that determines $e_w(p)$ is the same. It follows that the complexity of this partition is $O(n)$. Using standard divide and conquer we can compute this partition in $O(n \log n)$ time. In the first stage of the algorithm we compute and store these partitions for all rows and all widths. This requires $O(n^3)$ space and $O(n^3 \log n)$ preprocessing time. For a given $q \in R$ and width w we can now compute the interval $I_{w,r}(q)$ using binary search along r in $O(\log n)$ time as follows: For a $p \in r$ we locate p in the partition of r, and compute $e_w(p)$ in constant time. If $q \notin e_w(p)$ we can deduce if (i) for each $p' \in r$ to the right of p, $q \notin e_w(p')$, or (ii) for each $p' \in r$ to the left of p, $q \notin e_w(p')$.

After picking a random sample $\mathcal{E} \subseteq \mathbf{E}$ of ellipses, we need to see if $\mathcal{E}$ covers R. For this, we compute $\cup\mathcal{E}$ explicitly, (using for example a line sweep technique). Since the size of $\mathcal{E}$ is $O(k)$ the complexity of this union is $O(k^2)$, and it can be computed in time $O(k^2 \log k)$. Using point location we can find the first point $p \in R \setminus \cup\mathcal{E}$ in $O(n^2 \log k)$ time.

In order to maintain the weight of the ellipses efficiently, we do not construct the weight of each ellipse in $\mathbf{E}$ explicitly. Instead, for each row r and each possible fixed width w we construct a binary balanced search tree $\mathcal{T}_{r,w}$ on the grid points of r, sorted by their x-coordinate. Each leaf $v \in \mathcal{T}_{r,w}$ is associated with the maximal ellipse of width w whose center is (at the point) v. Each inner node $\mu \in \mathcal{T}_{r,w}$ maintains a factor ω_μ, and the sum of the weights of all leaves in its subtree (up to common factors on the path to the root). The weight of a leaf is the product of all factors on the path to the root. For a node μ let $I_\mu \subseteq r$ be the set of all points corresponding to leaves in the subtree at μ. In order to double the weights of all maximal ellipses of width w whose center lie on $I_{w,r}(q)$, we double the factor ω_μ for each node μ for which $I_\mu \subseteq I_{w,r}(q)$, but $I_{father(\mu)}$ is not contained in $I_{w,r}(q)$. Analogous to standard segment trees, we see that we need to update only $O(\log n)$ factors and sums of weights, and that computing the weight of a specific maximal ellipse is doable in $O(\log n)$. Picking the random subset is done as follows. We pick a tree at random, according to the sum of weights of leaves of the tree which is stored in the root. Next we compute a random path in the tree from the root to a leaf, where at an inner node the probability of branching to the left or the right child depends on the stored sum of weights of the leaves in the subtrees. The ellipse we pick is the maximal ellipse associated with the leaf that ends the path. Thus picking one random ellipse needs $O(\log n)$ time.

Altogether we need $O(n^3 \log n)$ preprocessing time to construct the trees and the partitions. In each round, of which there are $O(k_{\text{opt}} \log n)$ many, we pick a random sample in $O(k_{\text{opt}} \log k_{\text{opt}} \log n)$ time, compute the union and find an uncovered point q which needs $O((k^2 + n^2) \log k)$ time, and compute for all w and r the interval $I_{w,r}(q)$, and update the weights of the points in $I_{w,r}(q)$, which can be done in $O(n^2 \log n)$ time. This yields a total runtime of $\tilde{O}(n^3)$, where $\tilde{O}$ denotes a variant of the O-notation which subsumes polylogarithmic factors. Thus we obtained the following theorem:

Theorem 1 *Let R, F and $\mathbf{E}$ be as above, and let $n = |F|$. Then in expected time $\tilde{O}(n^3)$ one can find a set $\mathcal{E} \subseteq \mathbf{E}$ of ellipses whose union covers R and avoids F. The cardinality of $\mathcal{E}$ is $O(k_{opt} \log k_{opt})$, where k_{opt} is the cardinality of an optimal solution.*

Note that the total runtime is dominated by the $\tilde{O}(n^3)$ term for the preprocessing. We are optimistic that, using 2-dimensional planar subdivisions into regions with the same dominator, and constructing only those parts being accessed in the trees, we can obtain a runtime of $\tilde{O}(n^2 k_{\text{opt}})$. This is the subject of our ongoing research.

3 Computing Global via Local Matchings

3.1 Modeling the Matching Problem

Assume that a source image S and a target image T are given by their spot lists. Recall that after the spot detection a “spot” is simply a vector $(x(s), y(s), i(s))$ consisting of its nonnegative point coordinates $(x(s), y(s))$ in the Euclidean plane and a positive real number $i(s)$ describing its intensity.

By this simplification of spots we can treat the geometric matching task as an approximate point pattern matching problem, see [2] for a survey on this field. There, the general setting for our bottleneck matching type problem is as follows. For a real number $\varepsilon \geq 0$, a group of admissible transformations $\mathcal{A}$ (e.g. translations, rigid motions and/or scalings) and a metric d, one seeks a bijection f between as large as possible subsets $S' \subseteq S$ and $T' \subseteq T$, so that there exists a transformation $g \in \mathcal{A}$ for which $d(g(s), f(s)) \leq \varepsilon$ for every $s \in S'$. Thus, the transformation g describes the geometric resemblance between S' and T'. Additionally, we want that the intensity $i(s)$ for $s \in S'$ resembles the intensity of $f(s)$. However, this is only the general framework for our solution. There are several major difficulties stemming from the inherent noise in the electrophoresis process:

1. 2–DE images, even of the same probe, can differ significantly and the offset vectors for matching spots are only locally almost equal.
2. The spot resolution of the images can be different. Moreover, given a correct matching the intensity orderings of matched spots in both images are similar but not identical.
3. It often happens that one has to compute a matching of images that only partially “overlap”. Previous algorithmic solutions assume that both images show the same frames, or at least there are preset landmark pairs that allow to initialize the matching procedure. We neither want to choose the matching frames nor landmarks by hand.

To cope with these problems we choose the following 2–stage approach, which we call the global-via-local matching paradigm. See Figure 7 for a flow chart.

1. In a first step we compute a rather dense set of landmarks. This is realized via local matchings. A local matching is the registration of a small local source pattern (consisting of locally most

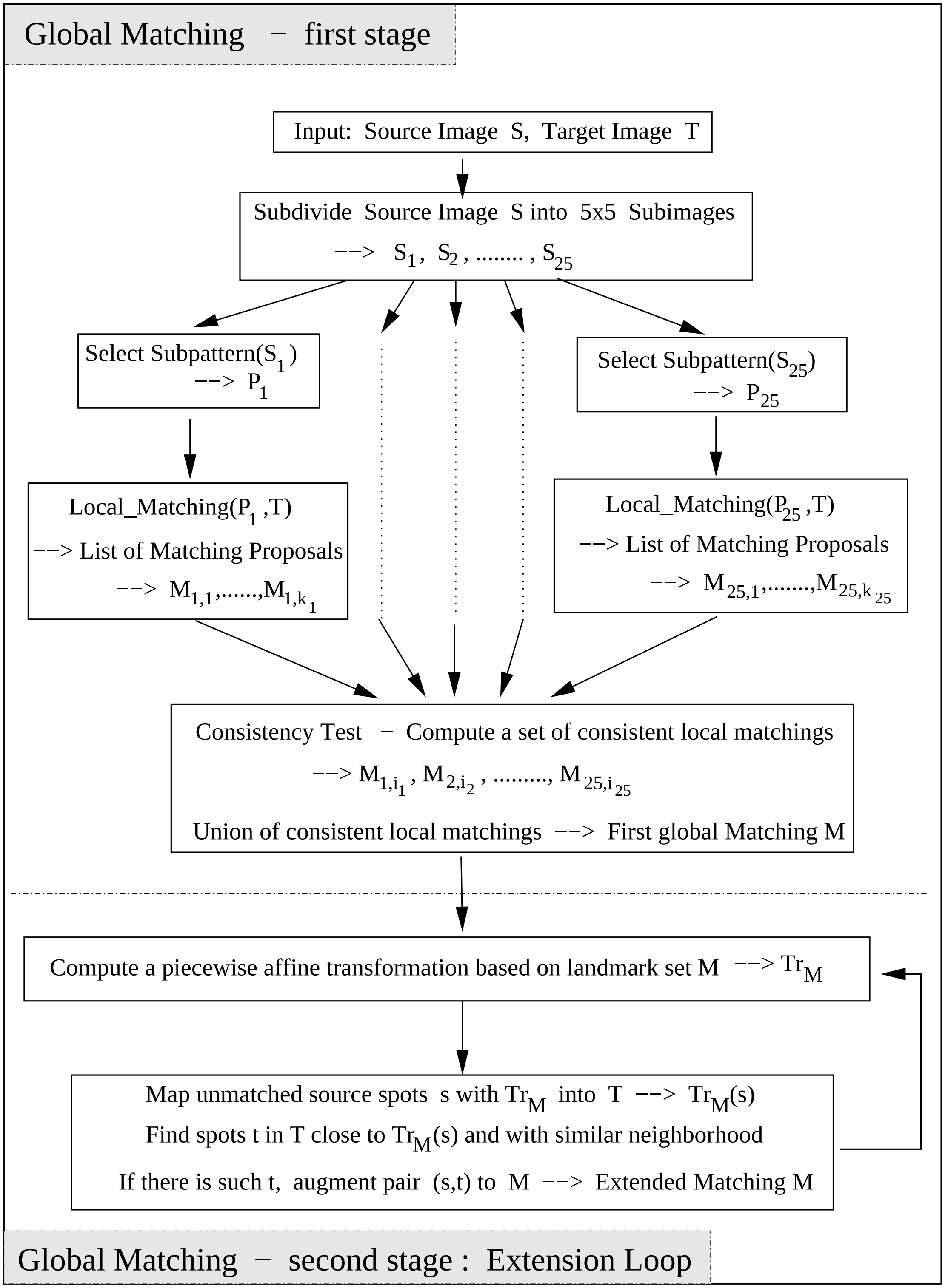
Global Matching − first stage
Input: Source Image S, Target Image T
Subdivide Source Image S into 5x5 Subimages
−−> S_1, S_2, ... , S_{25}
Select Subpattern(S_1)
−−> P_1
Select Subpattern(S_{25})
−−> P_{25}
Local_Matching(P_1,T)
−−> List of Matching Proposals
−−> $M_{1,1}$,......,M_{1,k_1}
Local_Matching(P_{25},T)
−−> List of Matching Proposals
−−> $M_{25,1}$,.......,$M_{25,k_{25}}$
Consistency Test − Compute a set of consistent local matchings
−−> M_{1,i_1}, M_{2,i_2}, ..., $M_{25,i_{25}}$
Union of consistent local matchings −−> First global Matching M
Compute a piecewise affine transformation based on landmark set M −−> Tr_M
Map unmatched source spots s with Tr_M into T −−> $Tr_M(s)$
Find spots t in T close to $Tr_M(s)$ and with similar neighborhood
If there is such t, augment pair (s,t) to M −−> Extended Matching M
Global Matching − second stage : Extension Loop

Figure 7: Flow chart of the matching algorithm.

intensive spots) in the target. For a single pattern the result is typically not unique, yielding different matching proposals. But for a set of grid–like located local source patterns we can choose a consistent common registration (i.e., matching) by comparing the underlying transformations, which should be similar for neighboring patterns.

2. A second step implements the extension of the landmark matching to neighboring spots. This is done by a variant of the standard neighborhood graph comparison. Moreover, in this step we take advantage of the extra information concerning ambiguous regions that was gathered in the spot detection preprocessing.

Next we formalize the geometric matching criterion for the registration of a local pattern $P \subset S$ in the target T.

We call two line segments $\overline{ss'}$ and $\overline{tt'}$ (λ, α)–*similar* if the absolute difference of their angles with the x–axis is smaller than α and for their lengths we have:

$$1 - \lambda \leq |\overline{ss'}|/|\overline{tt'}| \leq 1 + \lambda \tag{4}$$

Two point patterns $P \subset S$ and $Q \subset T$ (λ, α)–match if there is a bijection f between the point sets so that $\overline{ss'}$ and $\overline{f(s)f(s')}$ are (λ, α)–similar for all $s, s' \in P$. In sum, from the geometric point of view we want to find (λ, α)–matchings f between as large as possible subpatterns $P' \subset P$ and local target patterns Q, compare also [21].

As described in [17], we cannot model intensity resemblance by directly comparing $i(s)$ and $i(f(s))$, instead we rank, in both source and target, the spot intensities into 10 groups and require that the matching respects those ranks within a tolerance of 1.

3.2 How to Compute Local Matchings

In this subsection we present an efficient algorithm which computes partial approximate matchings of a pattern P in a target point set T. The algorithm has to find maximal subsets $P' \subseteq P$ (at least 50 %), corresponding subsets $Q \subseteq T$ and transformations t approximating the matching between P' and Q. Firstly, we discuss the transformation class of translations, but, we always keep in mind possible generalizations to homothetic transformations (translations plus scalings). In the runtime analysis k and n will denote the number of points in P and in T, respectively.

The algorithm is basically a twofold refinement of the naive alignment approach which computes a geometric matching by aligning a pattern edge $e \in P$ with a similar edge e' in T. If one wants to find full matchings of P, it is sufficient to fix e and search for all similar target edges. However, looking also for partial matchings, one has to check all pattern edges against all edges in T. Each edge similarity induces a translation $t_{e,e'}$ which maps the midpoint of e onto the midpoint of e'. Then candidate points for a matching can be found by nearest neighbor queries for the k points of the transformed pattern $t_{e,e'}(P)$. Hence, the runtime is bounded by $O(k^3 n^2 \log n)$ which combines the $k^2 n^2$ edge comparisons with the $O(k \log n)$ time to compute a matching. However, the algorithm is much faster in practice because most of the edges similarity tests are answered negatively.

Remark 1: It is not hard to construct examples where the nearest neighbor $q \in T$ of a transformed pattern point does not satify the requirements of a geometric matching whereas another neighbor q' does. Therefore all neighbors within a certain distance have to be checked.

Remark 2: Looking only for matchings approximated by translations it would be sufficient to check all transformations defined by point pairs $(p, q) \in P \times T$. However, such an approach is not very suitable in practice because it is not extendable to homothetic transformations and, moreover, the resulting $O(k^2 n \log n)$ time bound is tight.

Remark 3: In [18] a randomized version of the alignment method has been studied. One of the results is a $O(kn^2 \log n)$ Monte Carlo algorithm computing all partial matchings which match at least a constant fraction of P.

Instead of randomization we make use of the intensity resemblance which should hold for a majority of spots. To this end we order our point sets according to decreasing intensity. A triangulation of a point set X in the plane is called *Delaunay triangulation* if for each triangle in the triangulation its circumcircle contains only the three triangle points. One can construct such a triangulation in an incremental way by inserting points one by one in the given order, compare [15]. The insertion of a point can destroy some of the previous edges and add some new edges. The *history* $\text{Hist}(X)$ of the *incremental triangulation* is the set of all Delaunay edges which occur at some time in this process.
There are two results from Computational Geometry which together make up a fast and very robust heuristic.

Theorem 2 (see [3]) *Let X be a point set with an intensity ordering. Assume that a pattern $Q \subset X$ consists of the most intensive points in a rectangle R. Consider a triangle Δ occuring during the incremental Delaunay triangulation of Q. If the circumcircle of Δ is contained in R, then each edge of Δ belongs to Hist(X).*

This implies that under ideal assumptions (of nearly identical images) it suffices to consider all similarities between edges from $\text{Hist}(P)$ and $\text{Hist}(T)$. In practice however one cannot start from these assumptions: the matching is only approximate (local distortions), sometimes partial (missing spots), and moreover the intensity orders can vary (spots of regulated proteins). Nevertheless the Delaunay Triangulation approach is still suitable for the following three reasons:

1. The algorithm below is designed in such a way that it is not necessary to have for all edges of $\text{Hist}(P)$ similar counterparts in $\text{Hist}(T)$.

2. Augmenting $\text{Hist}(T)$ with all flip edges which occur in the incremental triangulation we obtain the so called extended history $\text{Hist}^*(T)$. This increases the chance to find similar edges; see [17] for some experimental results.

3. Even if the intensities of some spots in P differ heavily from their corresponding spots in T, there is a high chance to find the partial matching of the remaining pattern points. Based on this partial matching, geometrically corresponding spots with different intensities can be found in the extension stage, see Subsection 3.4.

The following fact implies an estimate on the expected size of $\text{Hist}(T)$.

Theorem 3 (see [24]) *Let X be a random permutation of n points. Then the incremental Delaunay triangulation of X can be computed in $O(n \log n)$ expected time. The expected number of edges in Hist(X) is $O(n)$.*

Therefore the expected size of $\text{Hist}^*(X)$ is also linear. Summing up, the restriction of the alignment approach to the comparison of edges from $\text{Hist}^*(P)$ and $\text{Hist}^*(T)$ implies a matching algorithm with expected time complexity $O(k^2 n \log n)$.

In fact, although this first improvement reduces the computation time remarkably, there is still a lot of wasted work because for any edge similarity $e \sim e'$ the algorithm attempts to construct a matching approximated by translation $t_{e,e'}$ in $O(k \log n)$ time. However, most of these attempts are bound to fail. We call a transformation $t_{e,e'}$ *good* if it approximates a matching of at least half of the points in P and *bad* otherwise.

The standard approach of how to avoid testing bad transformations is geometric hashing. Originally hashing has been worked out for the exact matching problem. The idea is to collect first all candidate transformations in a hash table. Note that in contrast to bad transformations the good ones occur at least $\binom{k/2}{2}$ times. In the end, the good transformations can be selected and, moreover, for each of them the corresponding matching has to be computed only once.

Here we develop a special variant of geometric hashing. We replace the hash table by a *scoring scheme* which uses a regularly spaced $m \times m$ grid over T with m^2 variables counting scores for all grid nodes. We discuss later how to choose m. This structure is suitable to compute also approximate matchings and it beats hash tables in both time and storage complexity. One has to pay for these advantages by accepting a certain heuristic flavor in this method. Let c_P denote the center of the bounding box of the pattern P. Again we start with all translations $t_{e,e'}$ defined by similar edges $e \in \text{Hist}^*(P)$ and $e' \in \text{Hist}^*(T)$. We store $t_{e,e'}$ adding scores to the four nodes of the grid cell which contains $t_{e,e'}(c_P)$. It is straightforward that good transformations yield clusters of points which in turn are represented by high scores in the neighboring grid nodes. Altogether we get an expected runtime of $O(n \log n + kn + m^2 + |\mathcal{T}_{\text{good}}|\, k \log n)$, where $|\mathcal{T}_{\text{good}}|$ denotes the number of good transformations, i.e., the number of computed partial matchings. The first and second term estimates the triangulation construction and the edge comparisons. Since the last term is of smaller order the number of grid nodes, i.e. m^2, plays the key role in the algorithm's analysis. Finer grids can display the cluster positions more precisely than coarser grids (with less nodes).

There is one more trick that allows to have both rather coarse grids and a sufficient precision of cluster positions. Each translation representative $t_{e,e'}(c_P)$ subdivides its grid cell into four rectangles. Instead of unit scores, each of the four grid nodes adds to its current score an amount proportional to the area of the opposite rectangle. This way the precise position of a single point $t_{e,e'}(c_P)$ can be recomputed from its scores. Let $\text{Score}(i,j)$ be the total score accumulated in grid node (i,j) after probing all edges from $\text{Hist}^*(P)$. All local maxima in the grid that are greater than a threshold value depending on $|P|$ are considered to correspond to potential matching locations.
Eventually, we can approximate the actual center (i_c, j_c) of the vector cluster stemming from a local maximum at node (i,j) by computing a weighted average of the scores at (i,j) and all scores at neighboring grid nodes:

$$(i_c, j_c) = \frac{\sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} \text{Score}(k,l) \cdot (k,l)}{\sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} \text{Score}(k,l)}. \tag{5}$$

The generalization from translations to homothetic transformations is straightforward. (λ, α)–similarity has to be replaced by α–similarity (the angle between the edges is at most α) and for each α–similar pair (e, e') the transformation $t_{e,e'}$ is defined by the scaling factor $|e'|/|e|$ and the translation mapping the midpoint of the scaled e onto the midpoint of e'. Consequently, the two–dimensional scoring scheme has to be replaced by a three–dimensional structure with the third dimension representing the scaling factors. In our implementation we used an exponential scale of the third axis where the factor between two grid units is λ.

Finally we have to compute the local matchings corresponding to good transformations $t \in \mathcal{T}_{\text{good}}$. To this end the whole pattern P is transformed: $t(P) = \{t(p) \mid p \in P\}$. Each transformed point $t(p)$

marks a position in the target image and we are seeking for matching candidates $p' \in T$ close to this position. As mentioned earlier the nearest neighbor to a given position is not always the correct matching partner. We call an incorrect matching of a point p with a point p' which is close to the position $t(p)$ a *switch*. To illustrate this problem suppose that there are two very close spots p_1, p_2 in the pattern, p_1 on top of p_2. Since the correct matching partners $p'_1, p'_2 \in T$ are also very close it can happen that the lower target point p'_2 is closer to the position $t(p_1)$ than the top point p'_1. Here the naive nearest neighbor strategy would decide for the switch (p_1, p'_2). Clearly, one can make the correct decision and avoid the switch by comparing the neighborhoods: Both, p_1 and p'_1 have a close neighbor below, whereas, p'_2 has not. Consequently, p_1 should be matched with p'_1 rather than with p'_2, although p'_2 is closer to the expected position $t(p_1)$.

To implement this idea we introduce small boxes $B(p)$ centered at the positions $t(p)$ and consider all $p' \in T \cap B(p)$ as potential matching candidates for p. The favourite selection depends on the neighborhood similarity of p and p'. Here the neighborhoods are defined by the Delaunay triangulation histories which had been computed during the preprocessing of the source and target point sets. This way there are no additional expenses for the neighborhood computation. It is essential to remark that the reliability of neighborhood-based decision depends on the correctness of the spot detection. If the protein spots p'_1 and p'_2 in the above example are so strongly overlapping that they were detected as one single spot p', the neighborhood comparison cannot help. At this point the different proposals computed in the spot detection come into play. Omitting all spots stemming from ambiguous regions in the local matching procedure one can improve the reliability of the matching result. Consequently it will be necessary to compute the matchings of spots from ambiguous regions in a postprocessing.

3.3 Assembling a Global Matching from Local Matchings

The local point pattern matching described in the previous subsection is the central module of our algorithm. We compute the global matching of two images S and T in several stages. First the source image is segmented into equally sized, rectangular subimages. In our implementation we used a regular 5×5-grid for this subdivision. Then for each of the 25 subimages the pattern formed by its 12 most intensive unambiguous spots is selected (25 and 12 are parameter values which proved to be optimal in numerous tests). Applying the local matching to those local patterns we get a list of matching proposals for each pattern. Now we apply a simple consistency test to those proposal lists. Two local matchings (of different patterns) are consistent if the underlying transformations are approximately the same. Finally a first global matching is established by searching for the largest family of pairwise consistent local matchings. This first global matching is then used as a skeleton M of landmarks for further extensions. Note that by the chosen parameters M has at most 300 spot pairs.

Another advantage of our approach is the possibility to compute a ”global” matching for images which only partially overlap, see Figure 8. In contrast to commercial programs we do not assume a global alignment of the images. This feature is useful for reconstructing big gel images from several partially overlapping subimages.

3.4 How to Extend a Global Matching

The purpose of the matching extension procedure is twofold. Firstly, we want to match unambiguous spots which, according to their lower intensity, were not in the initial local patterns. Secondly, we have to deal with spots from ambiguous regions. The first problem is solved by following a standard approach: The pairs in the skeleton M form a set of landmarks which defines a piecewise affine

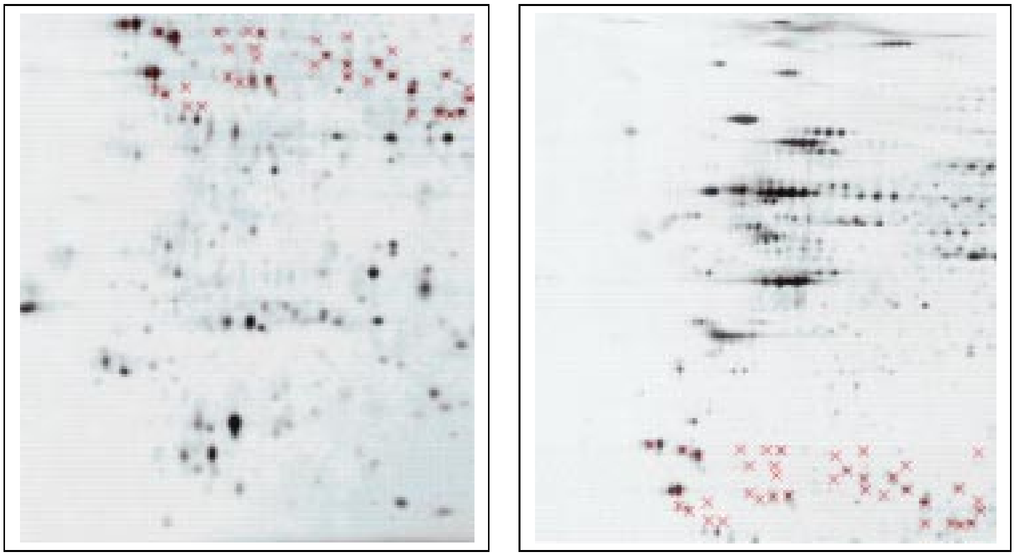

Figure 8: A global matching of two images which have only a small horizontal strip in common

transformation t_M. Again, the Delaunay triangulation proves to be a useful tool for computing t_M. We construct both the Delaunay triangulation DT_{source} of the previously matched source points S_M and DT_{target} of the corresponding target points T_M. In order to map a new source point p we find its Delaunay triangle $\Delta = (s_1, s_2, s_3)$ in DT_{source} by point location. If p is outside of the convex hull of S_M one can choose the Delaunay triangle Δ closest to p. There is a unique affine transformation t_Δ mapping the triangle vertices onto their partners (t_1, t_2, t_3). Thus, the expected position of p's partner in the target image is $t_M(p) := t_\Delta(p)$. Note that the expected time for a point location query is only $O(\log |S_M|)$. The matching candidates and the choice of a matching partner are then computed as explained in Subsection 3.2.

The extension procedure is iterated several times. After the first extension we can assume that the resulting skeleton is fine enough to attack ambiguous regions. To this end we introduce the following terminology. As before we call a spot *unambiguous* if it was determined uniquely by the spot detection. Each ambiguous region is represented by a so called *superspot* which stands for the possibility to consider the whole region as one complicated unit. The coordinates of the superspot are defined by the barycenter of the region. Spots contained in an ellipse covering proposal are called *subspots* of the superspot. We make one more simplification focussing our attention for each complex region only on the superspot and on the subspots of the best proposal computed by the spot detection. In fact, this simplification is appropriate for twin and triple spots rather than for very complex regions. There are two reasons for proceeding this way: The comparability of the edges of the incremental Delaunay triangulation is the basis of the local matching approach. However, inserting spots from different proposals (describing the same region) would yield lists which do not adequately describe the local geometric situation. Instead, inserting the subspots of the best proposal only maximizes the chances to get similar histories of the source and the target image. Therefore these subspots are inserted into the incremental Delaunay triangulation, although they are neither taken as local pattern spots in the source nor as matching candidates in the target. Moreover, considering more than two proposals could cause a combinatorial explosion in the search for candidates and the decision for a partner. The neighborhood of a superspot (which itself is not inserted into the Delaunay triangulation) is formed by taking the set-theoretic union of all subspot neiborhoods without the subspots themselves.

Starting with the second extension step all superspots and their (best proposal) subspots in both the source and the target image are taken into account. Thus, on the source side there are new spots p to be matched. On the target side there are new candidates.

We illustrate the advantage of our approach considering once more the elementary twin spot situation. In the source (resp. the target) image there are three possible results the spot detection can return:

1. The overlapping is so strong that the common region is detected as one unambiguous spot s (resp. t).

2. The overlap has been detected, and thus the common region is ambiguous. The best proposal consists of two spots s_1, s_2 (resp. t_1, t_2) which are subspots of a superspot $\bar{s}$ (resp. $\bar{t}$).

3. The algorithm has detected two separated unambiguous spots s_1, s_2 (resp. t_1, t_2).

Depending on the combination of these events and provided that the outside neighborhoods are similar we would get the following matchings:

source	1	1	1	2	2	2	3	3	3
target	1	2	3	1	2	3	1	2	3
matching	$s \leftrightarrow t$	$s \leftrightarrow \bar{t}$	none	$\bar{s} \leftrightarrow t$	$\bar{s} \leftrightarrow \bar{t}$ $s_1 \leftrightarrow t_1$ $s_2 \leftrightarrow t_2$	$s_1 \leftrightarrow t_1$ $s_2 \leftrightarrow t_2$	none	$s_1 \leftrightarrow t_1$ $s_2 \leftrightarrow t_2$	$s_1 \leftrightarrow t_1$ $s_2 \leftrightarrow t_2$

In the traditional approach without ambiguous regions only cases 1 and 3 can occur. Consequently we would obtain a matching only, if the spot detection makes the same decision on both sides (column $\binom{1}{1}$ and $\binom{3}{3}$). Now in our approach a matching will additionally be discovered if at least on one side the ambiguity (case 2) was recognized. This way the overall matching result can be used to resolve ambiguities in the results of the initial spot detection. In our view this is a promising attempt to overcome the separation between preprocessing and matching, and thus to simulate how an expert would compare images by visual inspection, namely, by intertwining the spot detection with the matching process.

4 Implementation Issues and Experimental Results

The 2–DE analysis software system CAROL ([9]) contains the described spot detection and matching algorithms. In order to facilitate its usage over the Internet it consists of two main parts: The first part, the combinatorial and geometrical kernel of the spot detection and matching algorithms, has been implemented in C++. It makes essential use of the Standard Template Library (STL) and of the Computational Geometry Algorithms Library (CGAL) [10]. The latter library provides several geometric data structures and functions and especially an implementation of the incremental Delaunay triangulation. For the LP approach of the spot detection, which we implemented for internal use only, we use the commercial LP-solver CPLEX.

The second part of the CAROL system is the graphical user interface which has been implemented in Java. It can be run as an applet started out of an internet browser or as an application. The communication with the algorithmic procedures is established via internet sockets, whereby the C++ program works as a server which waits for matching or spot detection requests from the Java-client, performs the computation, and eventually sends the results back to the client.

Using CAROL it is possible to analyze gel images from databases all over the Internet. This feature is strongly supported by the client-server architecture and the possibility to run the user interface out of an internet browser. However, a direct Internet access is required that is not restricted by a firewall. CAROL offers the possibility to open GIF images from any 2–DE database, to carry out the spot detection, to perform local or global matchings between two gel images, and to set parameters like tolerance bounds, pattern size, etc. The current version of CAROL can be found at *http://gelmatching.inf.fu-berlin.de*.

The local matching algorithm executed on a Sun Ultra Sparc 300MHz computes the best ten matchings for a pattern of twelve spots in about 0.2s. The first stage of the global matching between two gel images with 900 and 1000 spots takes about 7s and yields about 110 matching spot pairs. After this step the user can optionally correct the proposed landmarks. A subsequent extension step takes 2s. The computation of the spot detection for a full gel image (1200 x 1500 pixels, 4500 spots) takes about half a minute without complex regions; in such large images we observed between 0 and 20 complex regions. The brute force greedy approach for the complex region in Figure 5 (subset of a 76 x 80 array, with 59 inner sample points drawn in black) needs 24s to compute the 10 ellipse covering indicated. The LP-based solution (with initially 76 points defining outer constraints) takes 13s to compute the indicated solution. However, we remark that usually a spot detection is only computed once before storing the image in a database.

Tests of the CAROL system on several series of gel images as well as the positive feedback from external test users have confirmed the advantages of our geometric approach. Further work will concentrate on a more accurate mathematical modeling of the biochemical and physical phenomena within the electrophoresis process.

Acknowledgements

We would like to thank the anonymous referees for valuable comments.

References

[1] P.K. Agarwal and C.M. Procopiuc, Covering Points by Strips in the Plane, *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2000, 538–547.

[2] H. Alt and L. Guibas, Discrete Geometric Shapes: Matching, Interpolation, and Approximation, A Survey, In J. Urrutia, J.–R. Sack, eds., *Handbook of Computational Geometry* (North Holland), 1999, 121–153

[3] H. Alt, L. Knipping, and G. Weber, Point Pattern Matching in Astronautics, *J.Symbolic Computation* 17 (1994) 321–340

[4] V. Anil Kumar, S. Arya and H. Ramesh, Hardness of Set Cover with Intersection 1, *Proc. ICALP 2000*, Lect. Notes in Comp. Sc. 1853, 624–635

[5] R. Appel, J. Vargas, P. Palagi, D. Walther and D. Hochstrasser, Melanie II, a third–generation software package for analysis of two–dimensional electrophoresis images: II. Algorithms, *Electrophoresis* 18 (1997), 2735–2748

[6] M. Baker. J. Busse, M. Vogt, An Automatic Registration and Segmentation Algorithm for Multiple Electrophoresis Images, *Proc. of Med. Imaging 2000*, Proc. of SPIE 3979, 426–436.

[7] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, *Computational Geometry, Algorithms and Applications* (Springer), 1997.

[8] H. Brönnimann and M. T. Goodrich, Almost Optimal Set Covers in Finite VC-Dimension, *Discrete Comput. Geom.* 14 (1995), 463–479.

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

[10] CGAL, The Computational Geometry Algorithms Library, *http://www.cs.ruu.nl/CGAL*

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

[12] Joseph C. Culberson, Robert A. Reckhow, Covering Polygons Is Hard, *J. Algorithms* 17(1) (1994), : 2–44.

[13] A. Efrat, F. Hoffmann, K. Kriegel, and C. Schultz, Covering shapes by ellipses for the computer analysis of protein patterns, Technical Report B 00–11, July 2000, Institut für Informatik, Freie Universität Berlin

[14] J. Garrels, The QUEST System for quantitative analysis of 2D gels, *J. Biological Chemistry*, 264 (1989), 5269–5282

[15] L. Guibas, D. Knuth, and M. Sharir, Randomized Incremental Construction of Delaunay and Voronoi Diagrams, *Algorithmica* 7(4) (1992) 381–413

[16] S. Har-Peled, Taking a Walk in a Planar Arrangement, *Proceedings 40th Annual IEEE Symposium on Foundations of Computer Science*, 1999, 100–110.

[17] 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

[18] S. Irani and P. Raghavan, Combinatorial and experimental results for randomized point matching algorithms, *Computational Geometry: Theory and Applications* 12 (1999), 17–31

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

[20] K. Kriegel, I. Seefeldt, F. Hoffmann, C. Schultz, C. Wenk, V. Regitz-Zagrosek, H. Oswald, and E. Fleck, An alternative approach to deal with geometric uncertainties in computer analysis of two–dimensional electrophoresis gels, *Electrophoresis* 21 (2000), 2637–2640

[21] H. Ogawa, Labeled Point Pattern Matching by Delaunay Triangulation and Maximal Cliques, *Pattern Recognition* 19(1) (1986), 35–40

[22] P. F. Lembkin, Comparing two–dimensional electrophoretic gel images across the Internet, *Electrophoresis* 18(3–4) (1997), 461–470

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

[24] R. Seidel, Backwards Analysis of Randomized Geometric Constructions, in J. Pach, ed., *New Trends in Discrete and Computational Geometry* (Springer), 1993

[25] M. Sharir and P.K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications* (Cambridge University Press), 1995.

[26] M. R. Wilkins, K. L. Williams, R. D. Appel, and D. F. Hochstrasser (Eds.) *Proteome Research: New Frontiers in Functional Genomics* (Springer) , 1997