SERIE B — INFORMATIK

Matching 2D Patterns of Protein Spots*

Frank Hoffmann**
Klaus Kriegel**
Carola Wenk**

B 97-13
November 1997

Abstract

A new algorithmic approach to comparing 2D patterns of protein spots obtained by the 2D gel electrophoresis technique is presented. Both the matching of a local pattern vs. a full 2D gel image and the global matching between full images are discussed. The local matching algorithm relies on a data structure derived from the incremental Delaunay triangulation of a point set and a 2–step hashing technique. The approach for the global matching uses local matching for landmark settings, which in previous algorithmic solutions has been done interactively by the user.

*Part of a joint research project with Deutsches Herzzentrum Berlin, supported by Deutsche Forschungsgemeinschaft, grant FL 165/4–1.
**Institut für Informatik, Freie Universität Berlin, Takustr. 9. D-14195 Berlin
E-mail: *name*@inf.fu-berlin.de

1 Introduction

1.1 Point Pattern Matching

The matching problem for geometric point patterns has been subject of intensive research in the last decade. Given a point pattern P and another target point set T one wants to compute all occurrences of P in T. Usually, a space $\mathcal{A}$ of transformations (e.g. translations, rigid motions and/or scalings) is given which can be used to map the pattern into the point set T. Additionally we have a distance measure d between patterns. In general, we want to find such an $f \in \mathcal{A}$ and a pattern Q in T for which $d(f(P), Q) \leq \epsilon$, where ϵ is a prescribed error tolerance. We distinguish between exact matchings (ϵ is zero) and approximate matching solutions, otherwise. The latter are important in most practical applications. Like in our concrete application it is sometimes even sufficient to find partial matchings, i.e., we will be looking for as large as possible subpatterns of P which have an approximate matching pattern in T.

A survey on several variants of the general geometric matching problem, different geometric approaches, and various algorithmic techniques can be found in [1]. In particular, for most settings there are algorithms which solve the problem well from a theoretical point of view but which are hardly applicable in practice because of their time complexity.

Here we want to mention two approaches which have proved to be useful for our application, too: the alignment method and geometric hashing.
The alignment method is based on the observation that any similarity transformation is determined up to reflection by the mapping of a single line segment. Thus, we will choose two points a, b in the pattern P and map the edge $\overline{ab}$ to all edges $\overline{uv}$ in the target set T. For each mapping we check whether it induces a (large partial approximate) matching of P. Note, in the special situation of partial matchings it is not sufficient to consider only one edge $\overline{ab}$. So in a worst case scenario one has to map all pattern edges to all edges in T. The situation is much easier to handle if scalings are not allowed (or strongly restricted). Then for a given edge $\overline{ab}$ it is sufficient to search for all target edges of approximately the same length. Analogously, if rotations are forbidden it suffices to search for all target edges with approximately the same slope as $\overline{ab}$.
An essential speed up of the alignment method can be obtained if the points in both the pattern and the target are labelled with positive values (intensities) such that for any valid matching the intensity ordering of the pattern is consistent with the intensity ordering in the target. Then the quadratic size search space of all target edges can under certain circumstances be reduced to the set of all edges in the history of the incremental Delaunay triangulation of T which has linear size. The details of this idea will be discussed in the next section.
The main drawback of the alignment method consists in the waste of time caused by the fact that one tries to construct a matching for each legal pair of edges $\overline{ab}$ in P and $\overline{uv}$ in T, but finally only few of these attempts will be successful. One can avoid this by making use of geometric hashing. This method requires some preprocessing for a hash table. After that the number of pattern points which are matched by a transformation (induced by a legal edge pair) can be counted very efficiently. Finally, one has to compute the matchings only for the best transformations.

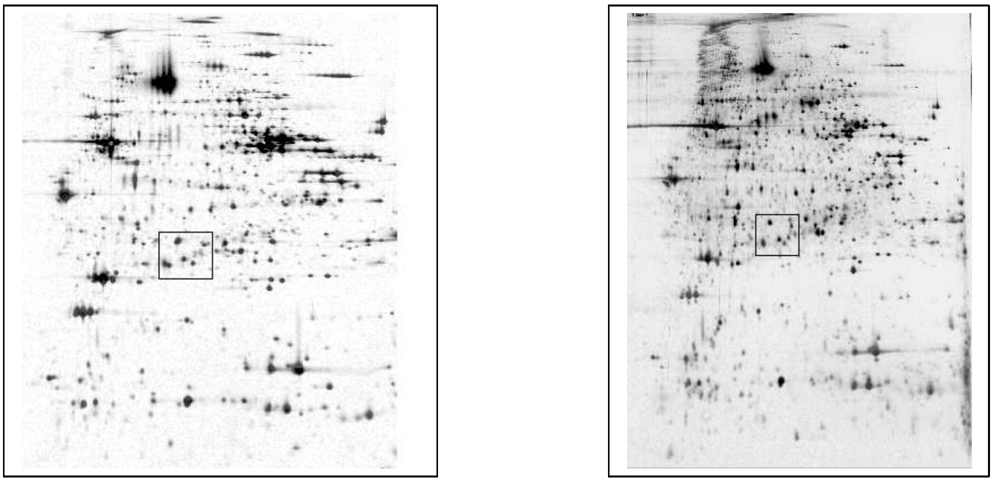

Figure 1: Two gel images of comparable samples

1.2 2D Gel Electrophoresis

Surveying the protein components of cells is a fundamental task in molecular biology. For this purpose the 2–dimensional gel electrophoresis technique for protein separation was introduced by O'Farrell in 1975. With a resolution seperation of several thousand in real samples it is almost two orders of magnitude better than competing techniques available for proteins. A 2D gel is the product of two seperations performed sequentially in acrylamide gel media: isoelectric focusing as the first dimension and a seperation by molecular size as the second dimension. A 2D pattern of spots each representing a protein is the result of that process. Eventually, spots are detected by staining or radiographic methods.

In Figure 1 two typical examples of such a 2–dimensional image of spots – each spot representing a specific protein is shown. The left image shows a gel image of a tissue sample from human heart ventricle. It contains about 1500 spots, while the right image in Figure 1 is a comparable image produced in another laboratory with the same technology but with a resolution of about 3000 spots. Their original size is about 23cm by 29cm.

Comparing visually such images (which are for instance available in different databases on the Internet) is one way for putative protein spot identification without using expensive sequencing techniques.

In medicine, one uses for example the comparison of whole series of 2D gel electrophoresis images to recognize disease associated protein expressions, compare for example [9].

However, for the visual comparison substantial difficulties arise from the fact that images are – due to inaccuracies in the complicated electrophoresis process itself– distorted, have different resolution, different spot expression etc. To a much greater extent this applies to the computer assisted comparison. For the purpose of illustration in Figure 2 more details are shown of the small rectangular window regions marked in Figure 1.

Previous solutions for a computer assisted gel image comparison – like the Melanie software system or the Java based Flicker programme [8] - rely mainly on the use of so–called *landmarks* or a general alignment of the images. Landmarks are spot pairs interactively marked in both images by the user and selected as a putative matching pairs. Using various

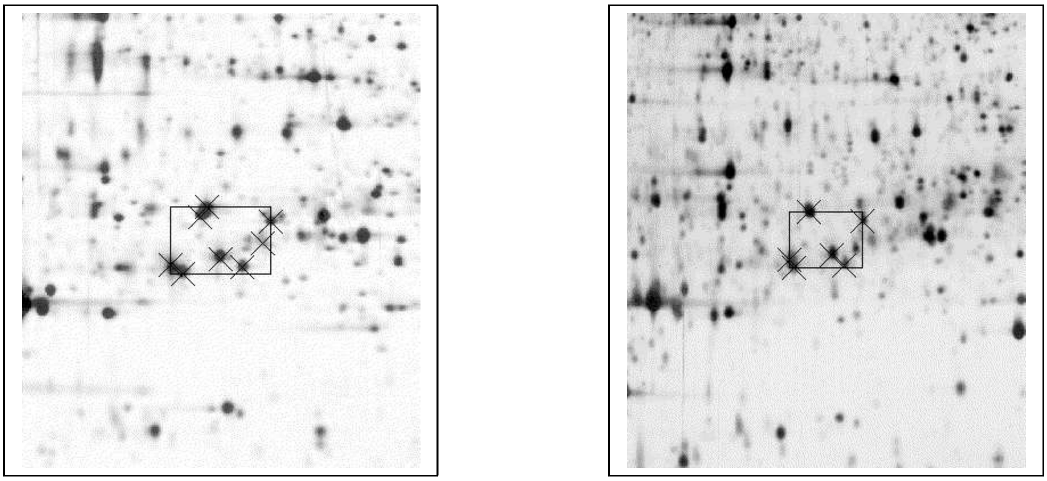

Figure 2: Detailed local images of Fig.1, a selected pattern on the left side and a partial matching

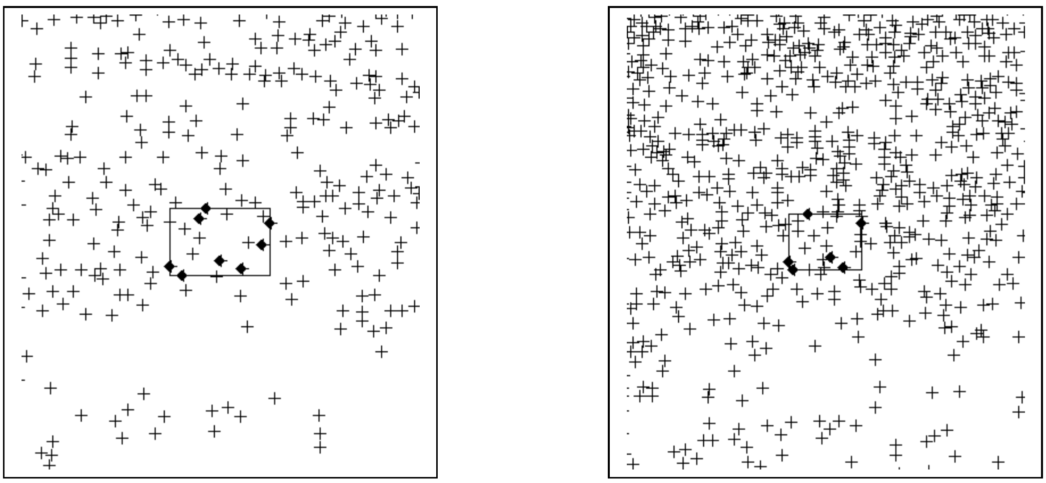

Figure 3: Spot point sets detected with indicated partial matching

heuristics one then algorithmically extends this partial relation to a maximal matching. In our algorithmic solution described below the setting of landmarks is optional.

2 Our Approach to the Matching Problem; Basic Algorithmic Ideas

We want to address first the following Local Matching Problem:

Given a local spot pattern P in a 2D gel source image S find all local spot patterns in a target image T that resemble at least partially both the geometric shape and the spot intensities of P.

To illustrate an instance of this problem consider the example of Figure 2: Given the indicated pattern of eight spots drawn from the small rectangle find all its matching counterparts in the right image. In fact, we even do not want to use necessarily the information about the relative position of the pattern within the source image.

The problem is formalized as follows. During a first feature extracting preprocessing step a so–called *spot detection algorithm* is applied to both source and target image. As a result we obtain for each gel image lists of spots. Now a spot is simply a vector $s = (x(s), y(s), i(s))$ consisting of its nonnegative point coordinates $(x(s), y(s))$ in the euclidean plane and a positive number $i(s)$ describing its intensity. Observe that during this step we necessarily lose a lot of visual information the original gel image carries like geometric spot shapes etc. The spot detection itself is a complicated algorithmic task and a severe source of error for our subsequent matching problem. In Figure 3 the spot point sets (without intensities) and, again, the partial matching corresponding to Figure 2 are shown. Remark that the algorithm computed a partial matching on six spots only although there are candidate spots for the remaining two pattern spots. The reason is that their intensities differ too much from those of the pattern spots.
Remark: The intensities given by the spot detection algorithm range from 2,7 to about 1600 of so–called oversaturated spots in the right example of Fig.1. So, comparing intensities of spots in different images firstly uses an heuristical discretization. The 50 most intensive spots have discrete intensity 10, independent of the spot number in the image , the next 50 get rank 9, the next 100 rank 8, and so on.

Now given gel images as mathematical objects we have to fix both the admissible transformation space and a distance measure to evaluate matchings. Of course, these criteria should be chosen first of all depending on the concrete application but also on the computational feasibility.
What are the restrictions imposed by the gel electrophoresis process?

1. Assume we want to choose a pattern P from a small rectangular window in the source image S. Source and target image can have significantly different resolutions, but intensive spots tend to appear first. Therefore it makes only sense to choose and restrict oneself to such patterns P that consist of the locally most intensive spots. On the other side, the target pattern P' should also consist of locally intensive spots.

We use as a heuristic rule that a spot s from the pattern can only be matched to a spot in P' if their discrete intensities differ by at most 2. Moreover, we should also accept solutions in which P' resembles only a large portion of P. This way we can also try to correct errors made by the spot detection algorithm.

2. To model the pure geometric resemblence between P and P' we make use of the following empirical fact. One can expect that if $\overline{s\,t}$ is a line segment connecting spots in P and if there are corresponding spots s', t' in P' then both the absolute slope difference and the relative length of $\overline{s\,t}$ and $\overline{s'\,t'}$ can be bounded by rather small constants α and λ, respectively. We call a pair of edges (λ, α)–*similar* if they satisfy these conditions. Let a (λ, α)–matching between P and P' be a 1–1–matching such that each corresponding pair of edges is (λ, α)–similar.

3. On the other hand, from the mathematical point of view since the edge similarity constants λ and α are small we know each partial (λ, α)–matching between P and P' is close to a translation. More precisely, the percentile one–way Hausdorff distance, see [7], between the translated P and P' is small.

In sum, from the application view we would like to find as large as possible (λ, α)–matchings between subpatterns $P'' \subset P$ and target patterns P' such that additionally P' resembles the intensities of P''. Besides the size of P'' another criterion for evaluating the matching could be the euclidean distance of P' from the expected position of P in the target image provided its positon in the source is known.

Given this general setting our local matching algorithm is based on the following key idea that was first time used in [12].
Let's call a triple of spots in a gel image *intensive* if its enclosing disk does not contain a spot that is more intensive. An edge connecting spots s, t is *intensive* if there exists a third spot forming together with s and t an intensive triple.
This concept of intensive edges is very strongly related to the Delaunay triangulation construction of a point set. A triangulation of a point set S 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 adding one point after the other, compare [4]. Now the main observation in [12] reads in our terminology:

If the Delaunay triangulation of a gel image is computed incrementally by inserting spots in order of decreasing intensity the set of all Delaunay triangles and edges occuring during the history of that process is exactly the set of intensive triangles and intensive edges.

Let $\mathrm{Hist}(T)$ be a data structure representing all intensive edges. How can one use this structure for the matching problem?
If a pattern P of locally intensive spots occurs in T, then we can expect that at least a few of the edges connecting spots in P will be (λ, α)–similar to edges in $\mathrm{Hist}(T)$.
This is the point where our approach and that one from [12] branch. While in [12] according to the alignment technique one tries to extend each occurence of a pair of similar edges to a matching of the complete pattern we have to opt for a different strategy. The main reason for this is the small but nevertheless considerable length and slope tolerance

(in the implementation the default values are $\lambda = 1.2$ and $\alpha = 0.2$) that imply a search range that is too large.

The alternative to the alignment technique in [12] that we choose is a 2–step variant of geometric hashing , see [1].
In a first step we want to compute all locations within the target image where a good matching with the pattern is likely to occur. The actual local matchings are computed subsequentely in a second step.
As pointed out above, if there is a good partial matching between pattern P and a pattern P' in T then a portion of edges connecting spots in P will have (λ, α)–similar matching edges in Hist(T).
Even more, if we associate with each occurence of a similar edge the translation vector of the edge midpoints then a matching pattern will correspond to a cluster of vectors in the translation space. It turns out that the clusters can be computed and evaluated efficiently. For the best candidate clusters we then recompute locally an actual matching.

Another problem is how to proceed in the case that there is a more severe distortion between the pattern P and its counterpart P' in the target image. Obviously, one solution would be to increase the values for slope and length tolerances yielding increased time bounds for the geometric hashing process. What we have implemented instead is a function to distort the pattern P iteratively in a prescribed way and then to search for the distorted pattern while keeping similarity tolerances small. These distortions are combinations of scalings and simple "straining" operations. Let Distort(P) denote the list of all patterns derived that way.

The overall algorithm to find local matchings between a spot pattern P and a target image T can be roughly summarized as follows:

LocalMatch(P, T)

1. Preprocess the target image via an incremental Delaunay triangulation.
2. <u>FOR</u> each pattern in Distort(P) <u>DO</u>
   Find best putative matching locations;
   Compute and evaluate corresponding local matchings.
3. Return best overall local matches.

3 Details of the Local Matching Algorithm

3.1 Preprocessing the Target Image

We first describe the preprocessing of the target image T. As sketched above we triangulate the underlying point set of T using the incremental Delaunay triangulation algorithm of [4]. Inserting spots according to decreasing absolute (not discrete) spot intensities. When a new spot is inserted a few edges are deleted from the current triangulation, a few new ones added. We call these edges Delaunay edges. Additionally we consider all edges connecting the new spot with opposite spots in neighboring triangles. Let us call

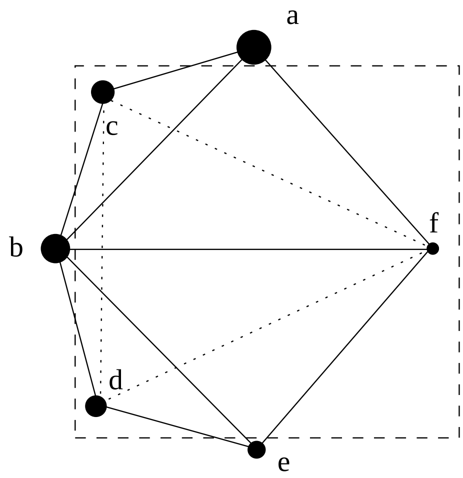

Figure 4: Pattern edges that are not Delaunay edges

these edges *flipped diagonals*. We store all Delaunay edges and flipped diagonals occuring during that process in a data structure $\mathrm{Hist}^*(T)$, that describes the *extended history* of the incremental Delaunay triangulation of T. With each object in $\mathrm{Hist}^*(T)$ we store also its length and its slope. For this purpose range trees are the appropriate data structure, see for example [3].

From [11] it easily follows that the expected number of objects in $\mathrm{Hist}^*(T)$ is $12n$, where n is the number of spots in T.

Example: The spot configuration in Figure 4 explains one reason for including flipped diagonals into the extended history. Let c, d and f be spots in the target pattern P' corresponding to pattern spots in P. The pattern was chosen from a window corresponding to the box drawn with dashed lines. Now consider the incremental Delaunay triangulation which inserts the spots in the order of decreasing intensity, i. e. alphabetically. We observe that none of the edges forming the pattern triangle occurs in the history of the triangulation, but all of them are flipped diagonals in $\mathrm{Hist}^*(T)$. A second reason for using the extended history $\mathrm{Hist}^*(T)$ is that we are able to tolerate small changes in the intensity order.

3.2 Approximating the Matching Locations

For each edge e in the pattern P we search for all (λ, α)–similar edges e' in $\mathrm{Hist}^*(T)$. With each occurence of such an edge we associate the vector $t(e, e')$ that translates the midpoint of e to the midpoint of e'.
How can we compute clusters of such vectors efficiently? Instead of storing all these vectors in a data structure we have used the following *scoring* procedure. The idea is not to store e and e' but to give a certain score to the corresponding region in the translation space.
We overlay a regularly spaced grid on the corresponding translation space which is bounded. With each occurence of a similar edge pair the associated translation vector $t(e, e')$ dis-

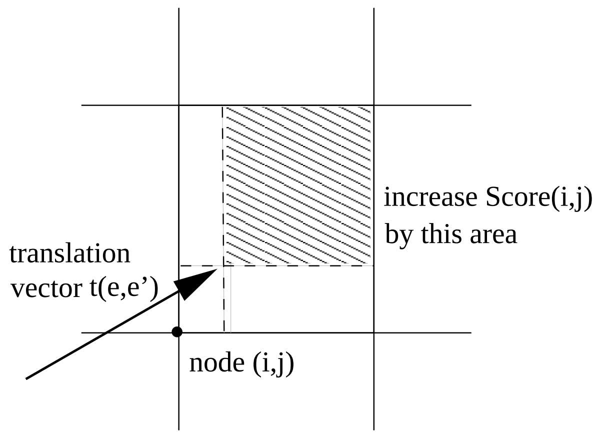

Figure 5: Updating the score for a pair e, e' of similar edges

tributes a fixed maximal score (scaled according to discrete intensity differences between e and e') among the four grid nodes defining the grid cell the vector falls into. This is done as follows: t subdivides the cell into four rectangles as depicted in Figure 5. Each of the four grid nodes adds to its current total score an amount proportional to the area of the opposite rectangle. Let $\text{Score}(i, j)$ be the total score accumulated in grid node (i, j) after probing all $\binom{|P|}{2}$ pattern edges. All local maxima that are greater than a threshold value depending on $|P|$ are considered to correspond to potential matching locations.
Eventually, we approximate the actual center 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. Figure 6 illustrates the result of the scoring procedure in the neighborhood of the actual matching position in the target in Figure 2.

3.3 Verifying and Evaluating a Local Matching

After the scoring procedure we are given a list of putative locations of matching pattern centers c. Next we recompute the actual patterns that define the matchings. To this end we consider the bounding box of the pattern P, scale it by length tolerance factor λ and for each center c we compute the rectangular subimage T_c centered at c of the target image. Next we have a *voting* procedure to compute a partial (λ, α)–matching between P and some pattern P' in T_c. This voting is very similar to the scoring procedure above. We compute the extended history $\text{Hist}^*(T_c)$. For each pattern edges $\overline{st}$ in P we search for all (λ, α)–similar edges $\overline{s't'} \in \text{Hist}^*(T_c)$. But this time we insert each found spot s' in a *candidate list* for s and t' in a corresponding list for t. We say that s gets a vote from s' and t from t', respectively. Analogously to the scoring procedure we again weight each vote according to the discrete intensity difference between s and s'.
Eventually, we form a tentative partial matching between all pattern spots s and their matching candidates that accumulated maximum number of votes and exceed a certain threshold depending on the pattern size.
This tentative matching is neither necessarily a 1-1–matching nor a (λ, α)–matching. Such a situation especially occurs if two pattern spots s, t are very close to each other but in the target there is simply only one sufficiently intensive spot in that corresponding place,

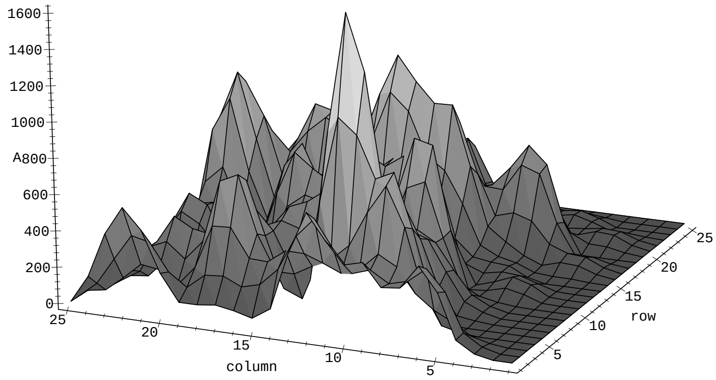

Figure 6: Local result of the scoring procedure

or there is a spot pair s', t' that violates the slope tolerance. Therefore there is a final *clearing* step that chooses a maximum size submatching of the tentative matching that is both 1–1 and meets the tolerance bounds.

How should we evaluate a found local matching? This is clearly application depending. If source and target gel image represent non–comparable samples then the matching cardinality is the only criterion. Otherwise, a ranking that combines both the cardinality and the distance from the expected location is possible. In the latter case we have also the possiblity to test the consistency of a local matching result by the following simple iterative method. We accept a local matching for P only if it is confirmed by local matchings for neighboring patterns Q that have nontrivial intersections with P. This idea serves as the basis for a *global matching* algorithm, see below.

3.4 Strongly Distorted Patterns

The local matching algorithm described before is robust to the respect that it indeed computes for each pattern a list of partial matchings within the prescribed tolerance bounds. Next we describe how to extend it to deal also with strongly distorted patterns.
Experiments with the implementation of the local matching algorithm have shown that one cannot simply increase the values for λ and α, because the scoring function will have almost everywhere large values and no distinguished local maxima that indicate possible matching locations.
There are two alternatives. The first one is to divide the matching process sequentially. We first apply a "typical" distortion transformation f to a pattern P and search then for $f(P)$. Such transformations are independent $x - y$–scalings or shifts that transform rectangular regions into parallelograms.
The other more elegant way is the following. We consider the 4–dimensional transformation space of translations and x– resp. y–scalings. Given upper bounds on the single

coordinates we can implement a scoring procedure that accumulates scores in nodes of a sufficiently refined 4–dimensional grid for each pair of similar edges.

3.5 Estimating the Time Complexity

First of all, we have to remark that the results of the following worst case anlysis do not completely reflect the performance of the algorithms for real world input data. However, this analysis is useful in order to compare our algorithm with previous approaches and to make clear where the progress comes from.
Let k and n denote the spot number in the pattern P and the target T, respectively.
The running time of the alignment method is the product of the number $sep(P,T)$ of similar edge pairs and the costs $m_t(P,T)$ to compute a matching under a translation t induced by such an edge pair. In the general setting we have $sep(P,T) = \mathcal{O}(k^2n^2)$ and $m_t(P,T) = \mathcal{O}(k \log n)$. Switching to the extended Delaunay history Hist^* we get $sep(P,T) = \mathcal{O}(k^2n)$. This implies a total upper bound of $\mathcal{O}(k^3n \log n)$ which was obtained in [12]. Our geometric hashing variant (i. e.,the voting procedure) allows to remove the $k \log n$ factor and to replace it by some additive terms. The general upper bound for geometric hashing of $\mathcal{O}(n^3)$ is also very bad, even without taking into accout the preprocessing and final computation of the matching.
In contrast, our voting procedure requires $\mathcal{O}(n \log n) + \mathcal{O}(k^2n) + \mathcal{O}(|G|)$ time, where the first term represents the costs of the incremental Delaunay triangulation, while the second one bounds the number of similar edge pairs. In the third term, $|G|$ denotes the size of the grid used to store the votes. It is of smaller order and can be ignored. Finally, computing the C best matchings costs $C \cdot m_t(P,T)$.
Thus, altogether we achieve an $\mathcal{O}(n(\log n + k^2))$ upper bound.

3.6 Global Matching via Local Matching

Previous algorithms for the global matchings of gel images are based mainly on landmarks set by the user. In this context a landmark is a pair of points, one in the source image S, the other one in the target T. Thus, the user fixes a partial matching on a sufficiently large set of so called support points $S_{supp} \subset S$. Triangulating S_{supp} and constructing the corresponding triangulation in the target image one gets a piecewise affine transformation f defined on the triangles. Finally, for any source point $p \in S$ one has to search for the nearest neighbor of $f(p)$ in T where the distance is a combination of the euclidian distance and the intensity difference.
Our aim is to avoid interactive landmark setting by making use of the local matching solution. The problem is that this algorithm computes several matchings of a chosen pattern and in general it is not clear which is the right one. There are two approaches to improve the confidence in a found matching. The first idea is to insert a spot $p \in S$ into different patterns $P_1, \ldots, P_m$ and to compute their best local matchings independently. We considered four patterns extending a bounding box from p into the four quadrants. Then a point $q \in T$ will be accepted as an image of p if for all patterns there is at least one (of the best) matching mapping p to q. It turns out that answers found this way are sufficiently satisfying. however, due to the strong restrictions within this approach it may happen that no answer will be replied.
The second approach consists in covering the source image by patterns in a grid like fashion.

Computing the best local matchings for all patterns one has to look for a consistent choice of matchings. Let P_1 and P_2 be neighboring patterns in S. A matching of P_1 to P_1' in T will be called consistent with a matching of P_2 to P_2' if for their centers $c(P)$ holds that the edges $\overline{c(P_1), c(P_2)}$ and $\overline{c(P_1'), c(P_2')}$ are (λ', α')-similar. Here, we have to enlarge the tolerance bounds to reflect the fact that for each single pattern P_i the matching can stem from the list $\mathrm{Dist}(P_i)$.

3.7 Implementation and User Interface

The implementation of the Carol system has been divided into two parts: The first part, the combinatorial and geometrical kernel of the matching algorithm, has been implemented in C++. It makes essential use of the Standard Template Library (STL) and of the Computational Geometry Algorithms Library (CGAL), [5]. The latter library provides several geometric data structures and functions and especially an implementation of the incremental Delaunay triangulation. 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 algorithmical program part is established via internet sockets, whereby the C++-program works as a server which waits for matching requests from the Java-client, performs the computation and sends eventually back the results to the client. The program will be eligible to match gel images from databases all over the internet. This feature is strongly supported by the possibility to run the user interface as an applet and furthermore by the client-server architecture of the program.

The user has the possibility to set parameters like tolerance bounds, pattern size etc., see http:www.inf.fu–berlin.de/gelmatching for more details of the user interface.
An unavoidable and critical preprocessing step is the spot detection stage. It is planned to include into the Carol system a spot detection algorithm that has been recently developed at Deutsches Herzzentrum Berlin, see [10].

The local matching algorithm run on a Sparc Station 10 computes the best 9 matchings for a pattern of 8 spots in about 6 seconds including the preprocessing of a 3000 spot image. Each further pattern in the list $\mathrm{Distort}(P)$ increases this time by about 0.5 seconds.

4 Conclusions and Directions for Further Work

We have presented new ideas for an algorithmic solution of the matching problem of 2D patterns of protein spots. Its main features are:

1. Local matchings for a source pattern are found in the target image without knowledge of its context. Therefore neither landmarks nor image alignment assumptions are required.

2. The local matching algorithm works for locally intensive patterns. But there are standard techniques that extend the solution to other spots. For example, point location can be used to determine the intensive triangle a spot belongs to. Applying the affine mapping that transforms the triangle to the matching triangle in the target

yields the tentative location of the matching spot and a subsequent nearest neighbor search gives the result.

3. The local matching algorithm can be used as a basic step for the global matching problem for comparable gel images. In fact, local matching is used then like landmark setting.

4. The central idea for the algorithm stems from the use of the extended history of the incremental Delaunay triangulation, which not only reduces the search range from quadratic (all target edges) to linear size but also reflects the intensity distribution in the target and tolerates small changes in the spot intensity order.

5. Although we started designing the matching algorithms with the application to protein spot patterns in mind, it is striking how little, in the end, application specific knowledge has been used. This will necessarily change in the fine tuning phase. Nevertheless it is our strong believe that the solution is applicable to distorted patterns of points with intensities that come up in quite different fields.

Acknowledgement: We would like to thank Klaus–Peter Pleißner from Deutsches Herzzentrum Berlin, Helmut Alt, Christian Knauer and Sven Schönherr from Freie Universität Berlin for their generous help.

References

[1] H. Alt, L. Guibas, Discrete Geometric Shapes: Matching, Interpolation, and Approximation, A Survey, TR B 96-11, Inst. f. Informatik, Freie Universität Berlin, to appear in J. Urrutia, J. –R. Sack, eds., Handbook for Computational Geometry (North Holland)

[2] H. Alt, F. Hoffmann, K. Kriegel, C. Wenk, K.-P. Pleißner, CAROL - New Algorithmic Tools for Comparing Two–Dimensional Electrophoretic Gel Images, Poster presented at Electrophorese Forum '97, Strasbourg

[3] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computational Geometry: Algorithms and Applications, Springer Verlag, 1997

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

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

[6] F. Hoffmann, K. Kriegel, A Partial Point Pattern Matching: Comparing 2D Patterns of Protein Spots, manuscript, 1997

[7] D. P. Huttenlocher, G. Klanderman and W Rucklidge, Comparing images using the Hausdorff distance, IEEE Trans. Pattern Analysis and Machine Intelligence 15 (1993) 850–863

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

[9] K. –P. Pleissner, V. Regitz–Zagrosek, C. Weise, M. Neuss, B. Kruedewagen, P. Soeding, K. Buchner, F. Hucho, A. Hildebrandt, E Fleck, Chamber–specific expression of human myocardial proteins detected by two–dimensional gel electrophoresis, Electrophoresis 16 (1995) 841–850

[10] K. –P. Pleissner, A. Sahlström, S. Wegner, H. Oswald, E. Fleck, Protein spot detection in WWW–accessible 2–D gel images using the Watershed Transformation, Poster presented at Electrophorese Forum '97, Strasbourg

[11] R. Seidel, Backwards Analysis of Randomized Geometric Constructions, Technical Report ICSI, Berkeley, TR–92–014 (1992)

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