Local Similarity Based Point-Pattern Matching*

Veli Mäkinen and Esko Ukkonen

Department of Computer Science
P.O Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland
{vmakinen,ukkonen}@cs.Helsinki.FI

Abstract. We study local similarity based distance measures for point-patterns. Such measures can be used for matching point-patterns under non-uniform transformations — a problem that naturally arises in image comparison problems. A general framework for the matching problem is introduced. We show that some of the most obvious instances of this framework lead to NP–hard optimization problems and are not approximable within any constant factor. We also give a relaxation of the framework that is solvable in polynomial time and works well in practice in our experiments with two–dimensional protein electrophoresis gel images.

1 Introduction

The point-pattern matching problem is, given two finite sets A and B in d-dimensional Euclidean space, to find the best matching of A and B under some quality measure. Point-pattern matching problems arise in many applications. For example, in two–dimensional gel electrophoresis proteins are separated to form spots in a planar gel, and one wants to match the same proteins from different experiments; Figure 4 in Section 5.3 gives an example of a pair of gels to be matched. Such gels may contain partly different proteins, and protein spots may be distorted non-uniformly. Also the spot intensities and sizes vary from one gel to another (the amount of variance is actually what is interesting to the biologists using the gels). Therefore the problem is normally reduced to matching the midpoints of the detected spots.

In computational geometry, point-pattern matching problems under uniform transformations (like translation, rotation, and scaling) are widely studied (see [2] for a survey). Allowing non-uniform transformations complicates the problem substantially, and a version of the problem has been shown NP-complete [1, 8].

In practice, greedy methods are used to cope with non-uniform transformations: Some pairs of points ("landmarks"), that should match, are selected first and one point-pattern is interpolated with respect to the other so that the selected pairs become aligned. This procedure is repeated until all possible matching pairs are determined. Usually, the selected pairs are determined interactively with the user. This kind of procedure was first proposed by Appel et. al. [3].

* Supported by the Academy of Finland under grant 22584.

A key issue when coping with non-uniform transformations is that although global similarity has not been preserved in the patterns to be compared one can still assume that some local similarity is present. Otherwise the problem would be infeasible. There are many proposals to formalize this criterion. Akutsu et. al. [1] and Jokisch and Müller [8] used an absolute criterion for this: Find the largest matching whose all pairs are consistent with a local similarity criterion.

In more detail, the criterion of [1] is the following. Given positive parameter ϵ, and two point-patterns $A = \{a_1, \ldots, a_m\}$ and $B = \{b_1, \ldots, b_n\}$ in d-dimensional Euclidean space, find a maximum matching $M = \{(a_{i_1}, b_{j_1}), \ldots, (a_{i_l}, b_{j_l})\} \subseteq A \times B$ satisfying for all h, k $(h \neq k)$

$$\frac{1}{1+\epsilon} < \frac{|b_{j_h} - b_{j_k}|}{|a_{i_h} - a_{i_k}|} < 1 + \epsilon.$$

The problem is NP-hard when $d \geq 2$. For $d = 1$, a dynamic programming algorithm was given that solves this problem with time complexity $O(m^2 n^2)$. As noted in [10], this algorithm assumes that the points in the matching M are in increasing order (i.e. $a_{i_1} < \ldots < a_{i_l}$ and $b_{j_1} < \ldots < b_{j_l}$). Without this assumption, the problem is NP-hard also when $d = 1$ as shown by Jokisch and Müller [8].

In this paper we introduce a framework for defining local similarity preserving point-pattern matching in Euclidean spaces. A matching of point-patterns A and B preserves local similarity, if the points of A that are close to each other are translated in about the same way in the matching to the points of B. This also means that the neighborhoods of the points of A are preserved in the matching.

Our framework is based on the following technical idea: If the local similarity is preserved then the translation which takes a point a of A to the corresponding point of B should be the same as what one would expect based on the actual translations of the neighbors of a. This suggests the concept of an expected translation. We then define the cost of a matching as the difference between the actual and the expected translations. One has to find matching for which this difference is smallest possible.

We give different concrete instances of defining the expected translations in the framework. Unfortunately it turns out that finding the corresponding shortest distances and optimal matchings is typically NP–hard and not efficiently approximable. We also discuss different possibilities of allowing outlier points (i.e., mismatches) in the matchings, including a technique based on many-to-one matchings. By relaxing the local similarity constraint we obtain a polynomial–time solvable variant whose evaluation reduces to minimum weight perfect matching in bipartite graphs. This method works well in practice: We report some performance results from our experiments with two–dimensional protein electrophoresis gels. The method finds fully automatically matchings that are almost identical to our reference matchings obtained using an interactive system.

2 Point-patterns and Matchings

A *point-pattern* is any finite set in the Euclidean d–dimensional space. Our main application is in the two–dimensional space in which case the patterns are called 2D point-patterns. However, most of the results are valid in the general case.

Let A and B be two point-patterns. A *matching* of A and B is any set $M \subseteq A \times B$. With the obvious correspondence with the bipartite graph matching in mind, we call the members of M the *edges* of the matching. An edge (a, b) can be interpreted as a vector from a to b. Then it gives the translation that makes a to match b.

A matching M is *one–to–one* if whenever (a, b) and (a', b') are two different edges in M then $a \neq a'$ and $b \neq b'$, and *many–to–one* if (a, b) is an edge in M then at most one of a and b can belong to some other edge in M. A many–to–one matching is *perfect* if each point of A and B belong at least in one edge in M; in the standard graph theory terminology such an M is an *edge cover* of $A \cup B$. If $|A| = |B|$ then a one–to–one matching M of A and B such that $|M| = |A|$ is *perfect*.

3 Local Similarity Preserving Distance Functions

Our goal is to find the best perfect matching between point-patterns A and B such that the local neighborhoods of the points of A are preserved in the matching. That is, we want that points of A that are close to each other are translated to the points of B in about the same way. Hence, as the edges of the matching give these translations, we want that the edge vectors are locally similar.

Depending on the application, the neighborhood of a point that is expected to behave similarly in matching may vary significantly. Thus, it is of interest to study different ways of defining local similarity.

The actual distance between A and B will be defined as the smallest possible sum of differences between the actual translations (edges) applied on each element of A and the 'expected' translations that would be consistent with the neighboring translations.

A general framework for distances of this type will be defined as follows. Let $\mathcal{M}$ denote the set of all perfect matchings between A and B (which in this case have to be of equal size) and $\mathcal{M}^{\text{many}}$ the set of all perfect many–to–one matchings between A and B. Let $f : \mathcal{M} \times \mathbf{R}^d \to \mathbf{R}^d$ be a function that given a perfect one-to-one matching $\mathcal{M}$ and a point a gives another point $f(M, a)$. Here the translation from a to $f(M, a)$ is the *expected translation* on a given by f while the *actual translation* on a is given by the edge(s) adjacent to a in the matching M. The purpose is that the expected translation is (almost) the same as the actual translations that M gives for the neighbors of a; later on we will give concrete examples of functions f. A corresponding function $f^{\text{many}} : \mathcal{M}^{\text{many}} \times \mathbf{R}^d \to \mathbf{R}^d$ for perfect many–to–one matching is defined similarly.

Now our local similarity distance will measure the difference between the expected and the actual translations as follows.

Definition 1 *Let* $M = \{(a_1, b_1), \ldots, (a_m, b_m)\}$ *be a perfect (one-to-one or many–to–one) matching between* A *and* B*, and let the* cost *of* M *with respect to* f *be*

$$w_f(A, B, M) = \sum_{k=1}^{|M|} |b_k - f(M, a_k)|.$$

The local similarity distance d_f *over perfect one-to-one matchings between* A *and* B *is defined as*

$$d_f(A, B) = \min\{w_f(A, B, M) \mid M \in \mathcal{M}\},$$

and the distance d_f^{many} *over perfect many–to–one matchings between* A *and* B *as*

$$d_f^{\text{many}}(A, B) = \min\{w_f(A, B, M) \mid M \in \mathcal{M}^{\text{many}}\}.$$

A matching M *that gives the value* $d_f(A, B)$ *or* d_f^{many} *is called the* best perfect matching *of* A *and* B *with respect to* f.

As we use only the perfect matchings in this definition, we assume in the sequel that the matchings considered are perfect if not explicitly stated otherwise.

There are many ways of choosing function f in Def. 1. It is desirable to have a function that captures local similarity and leads to a polynomial time computable distance. We will give below four different definitions for f, which all capture local similarity in some intuitive sense. The definitions range from very local to a full global condition in which all other points have an effect on the expected translation of a point. We are interested in the complexity of the distance evaluation problem in all these cases.

First think of the very local condition for local similarity in which the expected translation of a equals the actual translation of the nearest neighbor of a. This gives $f = f_{\text{NN}}$ as

$$f_{\text{NN}}(M, a) = a + (b' - a'), \tag{1}$$

where a' is the nearest neighbor of a in A, and $(a', b') \in M$. We call the corresponding distance function $d_{f_{\text{NN}}}$ the *nearest neighbor distance*.

An extension of the nearest neighbor distance would be the K*-nearest neighbors distance* $d_{f_{\text{KNN}}}$, where K is an integer, and the expected translations are given as

$$f_{\text{KNN}}(M, a) = a + \frac{\sum_{(a', b') \in M_K} b' - a'}{|M_K|}, \tag{2}$$

where $M_K \subset M$ consists of all $(a', b') \in M_K$ such that a' belongs to the K nearest neighbors of a in A.

Finally a global condition can be defined, where all actual translations have an influence on the expected translation. We make the strength of the influence to depend on the distance: We use a weighted average over the different actual translations, where the weights are inversely proportional to the distances from a:

$$f_{\mathrm{I}}(M,a) = a + \frac{\sum_{(a',b')\in M, a'\neq a} \frac{b'-a'}{(|a-a'|)^r}}{\sum_{(a',b')\in M, a'\neq a} \frac{1}{(|a-a'|)^r}}, \tag{3}$$

where $r \geq 1$ is a constant. We call the corresponding distance $d_{f_{\mathrm{I}}}$ the *interpolation distance.*[1]

To illustrate the three distance functions defined, one should easily see that if B is a translation of A then the distance between them equals 0 in all three cases.

4 Complexity of Local Similarity Preserving Distance

We will now prove the negative result that the three distances $d_{f_{\mathrm{NN}}}, d_{f_{\mathrm{KNN}}}, and d_{f_{\mathrm{I}}}$ are NP-hard to compute. We will start with $d_{f_{\mathrm{NN}}}$.

Theorem 2 *It is NP-hard to compute the nearest neighbor distances $d_{f_{\mathrm{NN}}}$ and $d^{\mathrm{many}}_{f_{\mathrm{NN}}}$.*

Proof. We use reduction from the 3-PARTITION problem [5]. The reduction is similar to what Jokisch and Müller used for *distance-congruence* [8]. The 3-PARTITION is defined as follows:

Definition 3 *Let P be a set of $3p$ elements, $S > 0$ a constant number, and $s : P \to \mathbb{N}$ a cost function with $S/4 < s(a) < S/2$ for all $a \in P$ such that $\sum_{a\in P} s(a) = pS$. The 3-PARTITION is the problem to find a partition $P_1, \ldots, P_p$ of P such that $\sum_{a\in P_i} s(a) = S$ for all $i \in \{1, \ldots, p\}$.*

The 3-PARTITION is strongly NP-complete [5], and thus NP-complete even if the costs $s(a)$ are bounded by a polynomial in the length of the input. We use this property in the reduction.

We construct two point-patterns from an instance of the 3-PARTITION as shown in Figure 1. The two point-patterns, A and B, have both pS points. The

[1] One might notice the connection between the interpolation distance and the image warping problem. In image warping, we have an image that should be warped onto another surface. For some of the pixels we know where they should be mapped. We would like to interpolate these mapping vectors so that the whole image is continuously transformed onto the other surface. This kind of interpolation can be done using $f_{\mathrm{I}}(M,p)$, where M represents the set of mapping vectors and p is a pixel whose mapping is to be calculated. The new location of p becomes $p' = p + f_{\mathrm{I}}(M,p)$. Hence finding the interpolation distance can be seen as an inverse problem for the image warping. Such warping methods have been proposed in the literature, cf. [12] for a survey. However, a common problem in these methods is overfitting; although smooth and continuous, the warp looks too variable for eye. Usually low-degree polynomials are incorporated to make the warp look more natural.

points in A are grouped according to the costs of the elements of P: the group i corresponds to the ith element a_i of P and has $s(a_i)$ points. The points in B are grouped in equal size slots of size S corresponding to the partition of P.

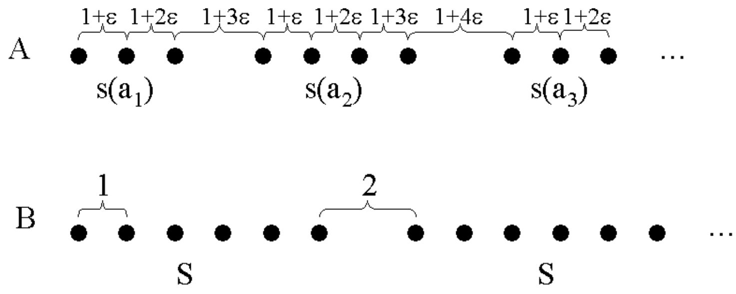

Fig. 1. Reduction from the 3-PARTITION to the nearest neighbor distance.

With this reduction one can solve by computing the nearest neighbor distance between point-patterns A and B also the 3-PARTITION decision problem. To see this consider the following cases.

1. Assume that there is a partition for the 3-PARTITION instance. Then there is a matching M_{yes} between A and B such that in each group of A all the points are mapped in the same order onto points in the same group of B; This matching is perfect and its cost is

$$w_{f_{\text{NN}}}(A, B, M_{\text{yes}}) = \sum_{a_i \in P} (2\epsilon + \sum_{j=2}^{s(a_i)-1} j\epsilon) \tag{4}$$

   which has an upper bound $\sum_{a_i \in P}(2\epsilon + (\frac{S}{2})^2\epsilon) = 3p(2\epsilon + (\frac{S}{2})^2\epsilon)$.
2. Assume that there is no partition. Now we know that there is no perfect matching between A and B that would map all the points in each group of A into the same group of B; there must be a point $a \in A$ whose nearest neighbor a' is mapped in a perfect matching into a different group of B. This gives us the lower bound $1 - \frac{S}{2}\epsilon$ for the distance $d_{f_{\text{NN}}}(A, B)$, since a may be mapped onto the end point of one group of B and a' onto the beginning point of the next group. On the other hand, there are many-to-one matchings M that do not break the groups but these cannot define a partition for the 3-PARTITION problem. A proper many-to-one matching M must have at least one more edge than a perfect matching. Thus the distance $d^{\text{many}}_{f_{\text{NN}}}(A, B)$ is at least $w_{f_{\text{NN}}}(A, B, M_{\text{yes}}) + \epsilon$. Noting that a perfect matching is a special case of many-to-one matching, we altogether obtain the following lower bound: $d^{\text{many}}_{f_{\text{NN}}}(A, B) \geq \min(1 - \frac{S}{2}\epsilon, w_{f_{NN}}(A, B, M_{\text{yes}}) + \epsilon)$.

By choosing small enough rational ϵ, the above lower bounds become larger than the distance (4). Hence we can solve the decision problem 3-PARTITION by computing the nearest neighbor distance between A and B (either $d_{f_{\mathrm{NN}}}$ or $d_{f_{\mathrm{NN}}}^{\mathrm{many}}$) and testing the value against (4). The NP-hardness follows as A and B can be constructed in polynomial time from the instance of 3-PARTITION.

The above reduction can be done in one–dimensional space as patterns A and B can be put on the same line. Hence the theorem is valid in Euclidean spaces of any dimension. □

Theorem 4 *It is NP-hard to compute the K-nearest neighbors distance $d_{f_{\mathrm{KNN}}}$.*

Proof. We construct point-patterns A and B from an instance of 3-PARTITION almost like in the previous proof, but we double S and all the values $s(a_i)$ until $s(a_i) > K$ for all $a_i \in P$. This guarantees that the K nearest neighbors will be found inside the same group in A. Note that the status of the 3-PARTITION decision problem remains the same in this transformation. Thus, we can safely assume that we have been given an instance in which $s(a_i) > K$ for all $a_i \in P$. The reduction is shown in Figure 2. The distances $x_1, x_2, \ldots$ between the groups in B will be defined below.

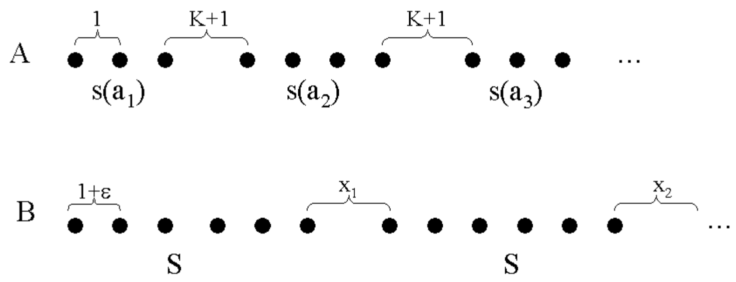

Fig. 2. Reduction from the 3-PARTITION to the K-nearest neighbors distance.

Now, if there is a partition for the 3-PARTITION instance, then there is a perfect one-to-one matching M_{yes} between A and B with cost $w_{f_{\mathrm{KNN}}}(A, B, M_{\mathrm{yes}}) \leq pS\epsilon$. This is because the average vector over the K nearest neighbors for each point in A maps the point to at most distance ϵ from a point in B.

If there is no partition, then there must be at least one broken group in A whose points are mapped into different groups in B in any perfect one-to-one matching M_{no}. We will obtain a lower bound for $w_{f_{\mathrm{KNN}}}(A, B, M_{\mathrm{no}})$ by calculating the cost of mapping only one point. Here one must take the following fact into account: Since we are taking the average over the vectors of the K nearest neighbors, a point could be mapped in the "right" place even inside a broken group due to symmetry. To prevent the possibility for symmetry, we define the distances between groups in B as follows: $x_i = 2(K-\frac{1}{2})((\sum_{j=1}^{i-1} x_j) + i(S-1)(1+\epsilon))$ for $1 \leq i < p$.

Let a be a point of a broken group in A such that at least one of the K neighbors of a is mapped into a different group. We can now calculate the lower bound for $w_{f_{\mathrm{KNN}}}(A, B, M_{\mathrm{no}})$ by a case analysis. There are three cases: (1) All the neighbors of a that are mapped into a different group than a, are mapped to the left from where a is mapped. (2) All the neighbors of a that are mapped into a different group than a, are mapped to the right from where a is mapped. (3) The point a has neighbors such that some of them are mapped to the left and some to the right from the group which a is mapped into. A lower bound for the cost in case (1) is easy to achieve; clearly the smallest cost is achieved when a is mapped to the first element of the second group in B, one of the K neighbors is mapped to last element of the first group in B, and all the other $K-1$ neighbors are mapped to last element of the second group in B. A straightforward calculation gives that the cost of matching a is at least $(S-1)(1+\epsilon)$. The lower bound for the case (2) can be derived similarly, but evidently it is greater than $(S-1)(1+\epsilon)$. The case (3) goes also similarly; the smallest cost is achieved when a is mapped to the last element of some group in B, say B_j, one of the K neighbors is mapped to first element of the next group, B_{j+1}, and all the other $K-1$ neighbors are mapped to first element of the first group in B, B_1. The reduction is constructed such that not depending which group a is mapped into, the lower bound cost will be the same as before, $(S-1)(1+\epsilon)$. As M_{no} is arbitrary, it follows that $d_{f_{\mathrm{KNN}}}(A, B) \geq (S-1)(1+\epsilon)$.

Now it is clear that choosing small enough rational value ϵ such as $\epsilon = \frac{1}{pS}$, we can solve the decision problem 3-PARTITION by computing the K-nearest neighbors distance $d_{f_{\mathrm{KNN}}}$. Again, A and B can be put on the same line, hence the result is true in all Euclidean spaces. □

Corollary 5 *It is NP-hard to compute the K-nearest neighbors distance $d^{\mathrm{many}}_{f_{\mathrm{KNN}}}$.*

Proof. The upper bound for the positive instances of the 3-PARTITION as well as the cases (1), (2), and (3) for the negative instances given in the proof of Theorem 4 hold also for $d^{\mathrm{many}}_{f_{\mathrm{KNN}}}$. We still show that in the negative case there can not be many-to-one matchings that do not break the groups and whose cost would be less than $pS\epsilon$.

As $|A| = |B|$, any perfect matching that is properly many-to-one must have at least one point of A, say a, that is mapped into two points in B, say b' and b''. The cost of these edges can be expressed as $|b' - (a + X')| + |b'' - (a + X'')|$, where X' and X'' are the average vectors that are calculated using the K neighbors of a. The average vectors depend only on a, not on its pairs in B, and thus $X' = X''$. Hence the cost of the two edges is $|b' - (a+X)| + |b'' - (a+X)| \geq |b - b'| \geq 1 + \epsilon > 1$. This is also a lower bound for the total cost of any many-to-one matching. Thus, by choosing $\epsilon = \frac{1}{pS}$ we have $d^{\mathrm{many}}_{f_{\mathrm{KNN}}}(A, B) < 1$ for positive instances of the 3-PARTITION and $d^{\mathrm{many}}_{f_{\mathrm{KNN}}}(A, B) > 1$ for negative ones. □

Theorem 6 *It is NP-hard to compute the interpolation distance $d_{f_{\mathrm{I}}}$, when the constant r of the function f_{I} is > 1.*

Proof. Again, we use a reduction from the 3-PARTITION, illustrated in Figure 3. The distances between the groups of A (values Y in the figure) and of B (values $x_1, x_2, \ldots$) are defined as follows: $x_i = 3S^2((\sum_{j=1}^{i-1} x_j) + i(S-1)(1+\epsilon))$ for $1 \leq i < p$, and $Y = x_{p+1}$. This guarantees that distance Y is longer than the diameter of B.

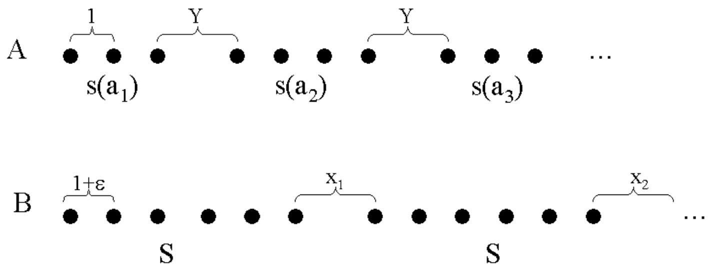

Fig. 3. Reduction from the 3-PARTITION to the interpolation distance.

As before, we want to show that a partition for a positive 3-PARTITION instance corresponds to a one-to-one matching M_{yes} between A and B whose cost is minimal over all one-to-one matchings. We can estimate the cost $w_{f_{\mathrm{I}}}(A, B, M_{\text{yes}})$ as follows. In M_{yes} all the points in the same group of A are mapped into the same group of B. The contribution $w((a,b))$ of one particular edge (a,b) of M_{yes} to the cost can be estimated upwards by assuming that the edge has at least one supporting edge in the same group (an edge (a',b') is called a supporting edge for (a,b) if $|(b-a)-(b'-a')| \leq \epsilon$ where ϵ is a parameter to be fixed below), and the points in the other groups try to take a as far away from b as possible. The goal is to select the parameter ϵ such that the effect of one supporting edge dominates over the total effect of the edges starting from the points in the other groups. Surely an upper bound for the worst effect is achieved when we assume that all the other points lie as near a as possible, i.e. at distance Y from a. Since we can place B after A on a same line, the points in the other groups can support a matching vector that would take a at most to distance $diam(A) + diam(B)$ from b, where $diam(S)$ means the diameter of point-pattern S. Now, we can estimate the cost contribution of the edge (a,b) in M_{yes} using (3) and the fact that the points lie on a line:

$$\begin{aligned} w((a,b)) &= |b - (a + \frac{\sum_{(a',b') \in M, a' \neq a} \frac{b'-a'}{(|a-a'|)^r}}{\sum_{(a',b') \in M, a' \neq a} \frac{1}{(|a-a'|)^r}})| \\ &< |b - (a + \frac{\frac{b-a-\epsilon}{1^r} + \frac{pS(b-a+diam(A)+diam(B))}{Y^r}}{\frac{1}{1^r} + \frac{pS}{Y^r}})| \end{aligned}$$

$$
\begin{aligned}
&< |b-(a+\frac{\frac{b-a-\epsilon}{1}+\frac{pS(b-a+pS+pY)}{Y^r}}{1+\frac{pS}{Y^r}})| \\
&< |b-(a+\frac{\frac{b-a-\epsilon}{1}+\frac{2pS(pS+pY)}{Y^r}}{1+\frac{pS}{Y^r}})| \\
&\leq |b-a-\frac{b-a}{1+\frac{pS}{Y^r}}|+|\frac{\epsilon}{1+\frac{pS}{Y^r}}|+|\frac{2pS(pS+pY)}{Y^r(1+\frac{pS}{Y^r})}| \\
&< |b-a-\frac{b-a}{1+\frac{pS}{Y^r}}|+\epsilon+|\frac{2pS(pS+pY)}{Y^r}| \\
&< \epsilon+\epsilon+\epsilon,
\end{aligned}
$$

for all $\epsilon > 0$, if we redefine Y such that $\frac{b-a}{1+\frac{pS}{Y^r}} > b-a-\epsilon$ and $\frac{2pS(pS+pY)}{Y^r} < \epsilon$. It follows that the latter condition is stronger, so it is enough to choose Y such that $Y^r > 2pS(pS+pY)/\epsilon$. This is possible if and only if $r > 1$. The resulting upper bound is $d_{f_{\mathrm{I}}}(A,B) < 3pS\epsilon$.

We have still to estimate the cost when there is no partition for the 3-PARTITION instance. We have constructed the point-patterns such that if there is a matching that breaks the groups in A, its cost is lower bounded by $(S-1)(1+\epsilon)$. This can be seen similarly as in the proof of Theorem 4.

Now it is clear that by choosing a small enough rational ϵ (e.g $\epsilon = \frac{1}{3pS}$), we can solve the 3-PARTITION by computing the interpolation distance $d_{f_{\mathrm{I}}}$. Again, the result holds true in all Euclidean spaces. □

Using similar arguments as in the proof of Corollary 5 we also obtain the following.

Corollary 7 *It is NP-hard to compute the interpolation distance* $d_{f_{\mathrm{I}}}^{\mathrm{many}}$ *when* $r > 1$. □

The inter-point distances in the reductions remain polynomially bounded with respect to the input values. Thus all the problems discussed above are strongly NP-hard.

As the upper bound for the distance in the positive instances of 3-PARTITION approaches 0 as $\epsilon \to 0$ while the lower bound in the negative case stays away from 0, it follows that also approximating the distances within a constant factor becomes NP–hard. However, this argument does not hold for the nearest neighbor distance $d_{f_{\mathrm{NN}}}^{\mathrm{many}}$. In this case our bound for the negative instances vanishes with ϵ.

Corollary 8 *It is NP-hard to approximate* $d_{f_{\mathrm{NN}}}$, $d_{f_{\mathrm{KNN}}}$, $d_{f_{\mathrm{KNN}}}^{\mathrm{many}}$, $d_{f_{\mathrm{I}}}$, *and* $d_{f_{\mathrm{I}}}^{\mathrm{many}}$ *(the two last with* $r > 1$*) within any constant factor* $c > 1$, *i.e to give an answer that is* $\leq c \times OPT$, *where* OPT *is the optimal solution for the corresponding minimization problem, and* c *is polynomial time computable on the input.* □

4.1 Allowing Mismatches

We have considered so far only distances based on perfect matchings that cover A and B in full. In typical applications, however, one wants to leave some points out of the match if this makes it possible to find a very good one-to-one matching between some subsets of A and B. We call the outlier points the *mismatches* of the matching.

We will briefly consider two different models of allowing mismatches:

(1) Fix a maximum allowed number κ for the mismatches in advance;
(2) Assign some cost c for the mismatches and find a matching that minimizes the total cost consisting of the cost of the matched elements as well as of the mismatches.

Let us denote the set of all (not necessarily perfect) one-to-one matchings between A and B as $\mathcal{M}'$. The *mismatches induced by a matching* $M \in \mathcal{M}'$ consist of the elements of A and B that are not adjacent to any edge of M. We denote by $m(M)$ the total number of such elements.

There is no obvious way of defining the cost function c in the mismatch model (2). Therefore we just assume that there is available a generic cost function that gives the cost $c(A, B, M)$ of the mismatches induced by M.

Definition 9 *Let* $M = \{(a_1, b_1), \ldots, (a_m, b_m)\} \in \mathcal{M}'$ *be a (not necessarily perfect) one-to-one matching between* A *and* B*, and let the* cost *of* M *with respect to* f *be*

$$w_f(A, B, M) = \sum_{k=1}^{|M|} |b_k - f(M, a_k)|.$$

Let $\kappa \geq 0$ *be an integer. The* local similarity distance with at most κ mismatches d_f^κ *between* A *and* B *is defined as*

$$d_f^\kappa(A, B) = \min\{w_f(A, B, M) \mid M \in \mathcal{M}', m(M) \leq \kappa\},$$

and the local similarity distance with mismatch cost function c *between* A *and* B *as*

$$d_f^c(A, B) = \min\{w_f(A, B, M) + c(A, B, M) \mid M \in \mathcal{M}'\}.$$

The complexity results of Section 4 hold also for the above versions of the distances allowing mismatches. For an example, evaluating $d_{f_{\mathrm{NN}}}^c$ is NP-hard since we can assign for each point a mismatch cost that is larger than the cost of M_{yes} in Theorem 2. Thus, if there is a partition for the 3-PARTITION instance, the optimal solution for $d_{f_{\mathrm{NN}}}^c$ will be the same as for $d_{f_{\mathrm{NN}}}$. If the function c depends on the problem instance, this result naturally does not hold, and a separate proof seems possible, depending on the details of c.

For $d_{f_{\mathrm{NN}}}^\kappa$ we have a stronger result that holds for each fixed κ. We can add κ extra points in the reduction such that if any of them are matched, the cost will be larger than the cost of M_{yes}. Thus, if there is a partition for the 3-PARTITION instance, the optimal solution for $d_{f_{\mathrm{NN}}}^\kappa$ will be the same as for $d_{f_{\mathrm{NN}}}$.

5 Relaxed problem and a useful heuristic

Our local similarity distances turned out NP–hard to evaluate. Therefore we will next look at relaxed formulations. In practice it might be possible to approximately evaluate the distance using an iterative approach. Such a method needs a good matching to start with. It turns out that sometimes we can ignore the local condition at first.

We define still another function $f = f_{\rm id}$ as $f_{\rm id}(M, a) = a$. This gives distance function $d_{f_{\rm id}}$ which as compared to our earlier distance functions does not get anything for free but has to measure the translations of the elements of A in full.

Distance $d_{f_{\rm id}}$ and its different mismatch variants can be evaluated fast using a reduction to a well–known graph matching problem, to be described next.

5.1 Reductions from $d_{f_{\rm id}}$ and its Variants to Minimum Weight Perfect Matching

Notice that the distance $d_{f_{\rm id}}$ corresponds to a known graph problem; it is an Euclidean instance of *minimum weight perfect matching* in a bipartite graph, when identifying points with nodes and pairs with edges. Moreover, $d^{\rm many}_{f_{\rm id}}$ corresponds to *minimum weight edge cover* in a bipartite graph. Also, Eiter and Mannila [4] have proposed a distance measure for point sets called *link distance*, which equals to $d^{\rm many}_{f_{\rm id}}$. They give in [4] an efficient algorithm for computing $d^{\rm many}_{f_{\rm id}}$ since they show a linear reduction from minimum weight edge cover to minimum weight perfect matching.

The reduction used for $d^{\rm many}_{f_{\rm id}}$ can be modified to cover also $d^{c}_{f_{\rm id}}$ in a special case. In this case the cost function c has to be additive in the sense that $c(A, B, M)$ has a representation as a sum of individual mismatch costs. That is, each element $a \in A \cup B$ has a mismatch cost $c(a)$, which can depend on A and B but not on M, and $c(A, B, M) = \sum_{a \in \bar{M}} c(a)$ where $\bar{M}$ is the set of the mismatches induced by M.

We only sketch the modifications of the reduction that are needed in this case. On the top of page 122 in [4], redefine the edge costs for edges of type $e = \{a_i, a'_i\}$ (resp. $e = \{b_j, b'_j\}$) as $w''(e) = c(a_i)$ ($w''(e) = c(b_j)$), where $c(a_i)$ ($c(b_j)$) is the mismatch cost assigned to point $a_i \in A$ ($b_j \in B$). The optimal perfect matching $M^{G''}$ in this new reduction graph G'' corresponds to an optimal solution M^c of $d^{c}_{f_{\rm id}}$ as follows: If $(a_i, b_j) \in M^{G''}$ then $(a_i, b_j) \in M^c$, if $(a_i, a'_i) \in M^{G''}$ then a_i is a mismatch, and if $(b_j, b'_j) \in M^{G''}$ then b_j is a mismatch.

For the distance d^{κ}_{f} a slightly different reduction is needed. Note first that we can assume that $\kappa - |m - n|$ is divisible by 2 (after removing κ mismatches we must be able to produce a perfect matching). The reduction is the following: Add $|m-n|$ points into the largest point-pattern, and then $(\kappa - |m-n|)/2$ points into both point-patterns. Denote this set of "mismatch sinks" by $S = S_A \cup S_B$, where S_A is the set of points added to B and S_B is the set of points added to A. Assign 0 cost for each pair of points of type (a, s_a) or (s_b, b), $a \in A, s_a \in S_A, b \in B, s_b \in S_b$. For all other pairs the cost is the Euclidean distance between the points. It is easy to see that by computing the perfect matching M^S between $A \cup S_B$ and $B \cup S_A$ one gets an optimal solution M^κ for $d^\kappa_{f_{\mathrm{id}}}$ as follows:

If $(a, b) \in M^S, a \in A, b \in B$ then include (a, b) into M^κ. The other κ pairs contain the mismatches.

Proposition 10 *The distances $d_{f_{\mathrm{id}}}$,$d^\kappa_{f_{\mathrm{id}}}$, $d^c_{f_{\mathrm{id}}}$ (with an additive c), and $d^{\mathrm{many}}_{f_{\mathrm{id}}}$ on point-patterns A and B of sizes m and n, respectively, can be computed asymptotically as fast as minimum weight perfect matching in a bipartite graph of $m + n$ nodes and mn edges.*

5.2 Heuristic Solution

As such, $d_{f_{\mathrm{id}}}$ does not measure local similarity, but if we replace the Euclidean distances between points with a local similarity measure between the neighborhoods of the points, we obtain a polynomial time solvable problem that takes local similarity into account.

Consider first that there are no mismatches. Assume that the data are such that when restricted to a small region in the data (e.g. a region with K nearest neighbors for a point), the mapping that is produced by e.g. optimal solution for the K-Nearest neighbor distance $d_{f_{\mathrm{KNN}}}$, is close to translation. With this kind of data we can estimate the cost of matching an edge (a, b) by comparing the neighborhoods of a and b. Thus, we can assign a cost for each edge (a, b), $a \in A$, $b \in B$. With these costs, we can calculate the minimum weight perfect matching between A and B.

The cost for an edge (a, b) can be calculated as follows. Take K nearest neighbors for a in A, translate them with vector $b - a$, and match each translated point with its nearest neighbor in B. The result is a matching M^{ab} of size $K + 1$ (pair (a, b) is included). Assign cost $w((a, b)) = |b - f_{\mathrm{KNN}}(M^{ab}, a)|$.

We can express this matching problem by slightly abusing our framework by defining

$$f_{\mathrm{H}}(M, a) = -b + w((a, b)).$$

Now the corresponding distance $d_{f_{\mathrm{H}}}(A, B)$ equals to the minimum weight perfect matching between A and B.

Note that when the minimum weight perfect matching M is calculated with these costs, the matching is not guaranteed to include any of M^{ab}; it may happen that a point is matched far away from where its neighbors are matched.

A heuristic solution for preventing the deviation between the matches of neighboring points in M would be the following. When calculating the cost for an edge (a, b), we combine the similarity of the neighborhoods with the distance between a and b. This should work in situations where a and b can be assumed to be relatively close to each others.

Adding the sensibility for mismatches in this model is non-trivial. First, the mismatches affect the calculation of the costs for the possible edges (a, b). To fix this, one could take the "best" portion of the K nearest neighbors that support the edge (a, b). The second problem is how to separate the mismatches from the matches. This can be done either by using $d_{f_{\mathrm{H}}}^{\kappa}$ or $d_{f_{\mathrm{H}}}^{\mathrm{many}}$. They both can be solved in polynomial time using the reductions given in Sect. refsec:idreductions for $d_{f_{\mathrm{id}}}$ (the reductions extend to the cases where non-negative costs are used as distances between points). In $d_{f_{\mathrm{H}}}^{\mathrm{many}}$, one can choose the minimum cost edge from each many-to-one subset of the matching to be a match, and the other points can be regarded as mismatches.

5.3 Experimental Results

We have implemented a heuristic using the ideas presented above and done some experiments on two-dimensional protein electrophoresis gels (see Figure 4 for an example).

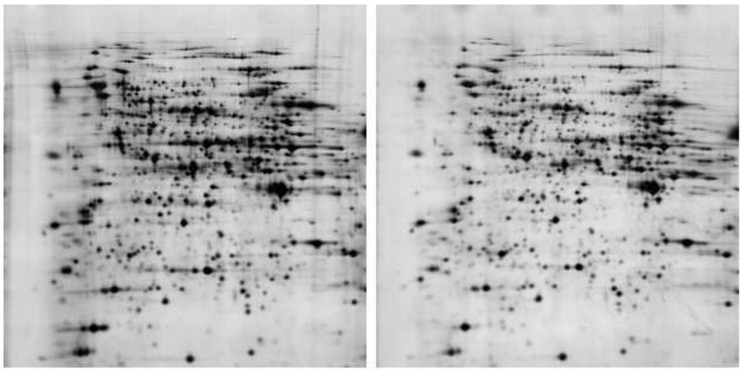

Fig. 4. Two-dimensional protein electrophoresis gel images.

The goal is to match two gel images so that the corresponding protein spots will be matched. Our abstraction of this problem is to identify the spots with their midpoints and match these sets of midpoints. So the input for our method is two point-patterns, that both contain a lot of missing points and the transformation between the point-patterns is non-uniform.

To test our method, we had 15 pairs of point-patterns extracted from different gel images (about 1000 points in each point-pattern). We also had the "original" matchings for these point-patterns to compare, that were constructed with PDQuest$^{\mathrm{TM}}$ software, where the user can give landmarks (trusted matches) and the program tries to manage from the rest.

For computing the matchings, we used the implementation of Goldberg and Kennedy [6] for minimum weight perfect matching in a bipartite graph. Due to the linear time reductions introduced before, also the versions where mismatches were allowed could be computed efficiently (typically in couple of seconds on a 700MHz Pentium III computer).

Because there were two kind of errors between these point-patterns, missing points and non-uniform motion, we decided to test first the sensitivity for pure non-uniformity. For this we removed all the points that were not matched in the original matching. We compared the original matching with the matching produced by our method. With most of the point-pattern pairs the result was fully consistent with the original matching and the largest error was 0.4% (meaning that 0.4% of the original matches were not detected).

Next we tested the real life situation, where both missing points (10-25%) and non-uniform motion were present. We used two mismatch models, $d^{\kappa}_{f_{\mathrm{H}}}$ and $d^{\mathrm{many}}_{f_{\mathrm{H}}}$. In the first one, we needed some estimate for the number of mismatches κ. To test the upper limits of the method, we used the correct value for κ that was calculated from the original matching. The results are shown in Figure 5. Surprisingly, $d^{\mathrm{many}}_{f_{\mathrm{H}}}$ managed better than $d^{\kappa}_{f_{\mathrm{H}}}$ although we knew the number of mismatches beforehand. However, $d^{\mathrm{many}}_{f_{\mathrm{H}}}$ naturally contained more false negatives than $d^{K}_{f_{\mathrm{H}}}$ since also the mismatching points were matched. Altogether the results were quite good, on the average 93.7% of the original matches were found by $d^{\mathrm{many}}_{f_{\mathrm{H}}}$.

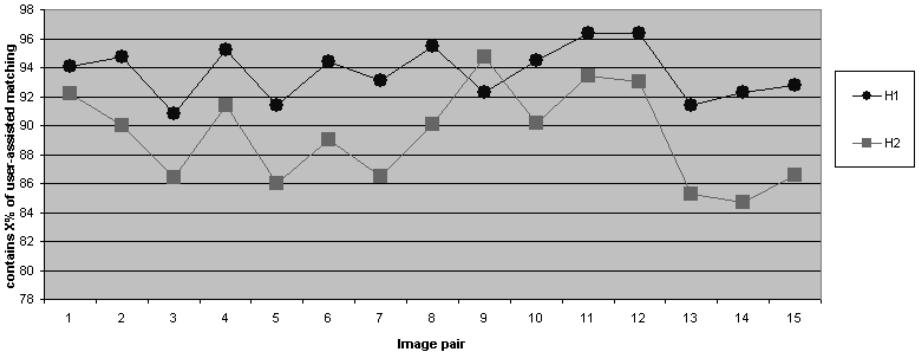

Fig. 5. Comparing the results of H1=$d^{\mathrm{many}}_{f_{\mathrm{H}}}$ and H2=$d^{\kappa}_{f_{\mathrm{H}}}$ against the original user-assisted matching.

We tried to improve the heuristic with basically the following iterative step: After a matching M is achieved (with $d^{\kappa}_{f_{\mathrm{H}}}$ or $d^{\mathrm{many}}_{f_{\mathrm{H}}}$), assign a cost for each pair based on the Interpolation distance ($w((a,b)) = |b - (a + f_{\mathrm{I}}(M,a))|$). Remove $X\%$ of the pairs with the highest cost from M. Move the points in A based on the matching M using the Interpolation distance ($A' = \{a + f_{\mathrm{I}}(M,a) | a \in A\}, A = A'$). Either recalculate matching and start the iteration again, or match each unmatched point in A to the nearest unmatched point in B if it is not more than Y times farther than the nearest matched point in B.

It turned out that the results did not change much after the first iteration (we used $X = 10\%$ and $Y = 2$). The results were not significantly better than with the basic approach, now the average was 94.9% when one iteration was used. However, the one iteration step was necessary for getting a useful matching, since the solution of $d_{f_{\mathrm{H}}}^{\mathrm{many}}$ contained a lot of false positives, which were now removed. An example of a matching produced by this iterative method is shown in Figure 6.

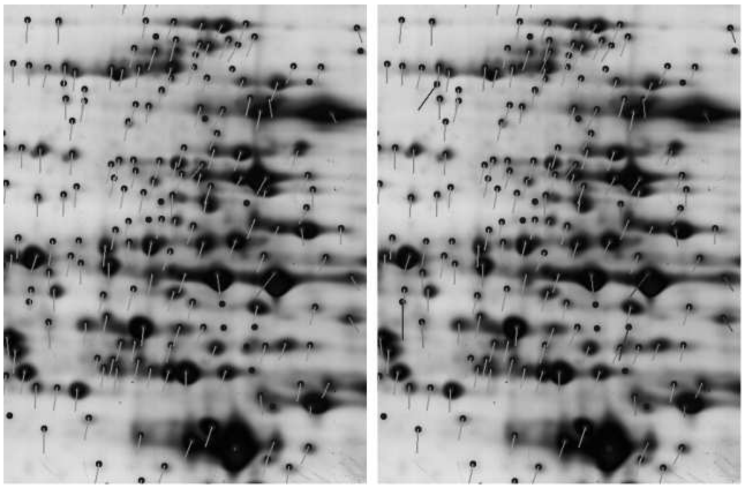

Fig. 6. Visualization of matchings. The original user-assisted matching is in the left, and the result of automatic matching is in the right. Both are portions of the matchings for the images in Fig. 4. As can be seen, the matching "flow" is relatively constant when restricted to a small neighborhood. This is essential for the heuristic to work. Experiments with gels with sparser selection of detected spots and greater non-uniformity have confirmed this fact.

As we can calculate the Interpolation distance for a given matching, we can now compare the quality of the matching proposed by our method with respect to the original matching. This comparison is shown in Figure 7. As can be seen, the user-assisted matching has smaller cost in each image pair. However the difference is quite small, so the heuristic solution is quite close to the optimum (assuming that the optimum of $d_{f_{\mathrm{I}}}$ is close to the user-assisted matching).

6 Conclusions and Future Work

There are several possibilities for future studies, still. We have only considered a few example definitions for function f in our framework; other definitions could lead to more practical distance measures. Extremely interesting is the tradeoff

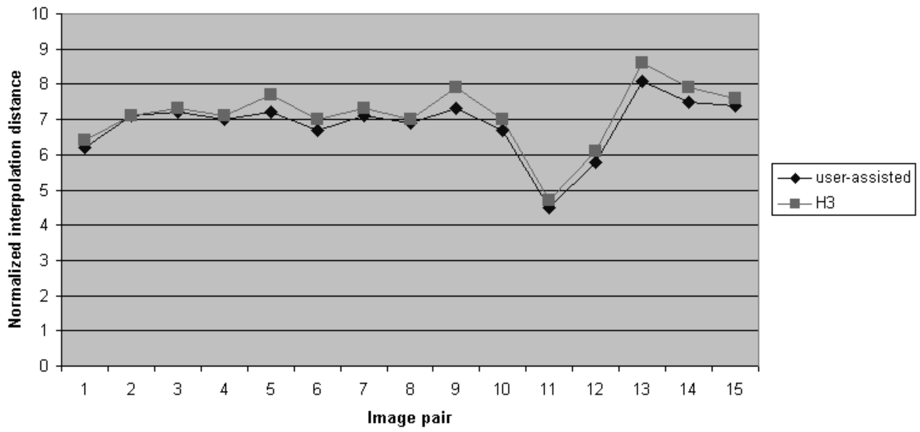

Fig. 7. Comparing the solution of the iterative heuristic (H3) to the solution of the user-assisted matching. The Normalized interpolation distance is d_{f_I}/m. Intuitively the value means how many units (e.g. pixels) on average a point must be moved once the effect of the overall matching is taken into account.

problem: How good local similarity function f one can use such that the problem can still be solved in polynomial time.

Another direction is to represent the problem as a *Metric Labeling Problem* [9], where one seeks for a labeling of objects such that the labeling cost plus the difference between labelings of neighboring objects is simultaneously minimized. Here one could identify one point-pattern with objects and the other as labels. The labeling cost could be the same as what we used in the heuristic in Sect. 5.2; the prior estimate of local similarity between an object and a label. The difference between labelings of neighboring objects could simply be measured as the distance between labels (points in the other point-pattern). In addition, one would need to make the definitions symmetric (i.e. all labels must be used as well as all objects must be assigned a label). There are approximation algorithms for some metric labeling problems (e.g. [9]) and even polynomial time algorithms for some special cases (e.g. [7]). On the other hand, the requirement of symmetric definitions seems to make the problem harder; it is a special case of *Quadratic Assignment Problem*, which in general is NP-hard to approximate even in the metric case [11].

Acknowledgements

We would like to thank Laura Salusjärvi from VTT Biotechnology for providing the data for the experiments.

References

1. T. Akutsu, K. Kanaya, A. Ohyama, and A. Fujiyama. Matching of Spots in 2D Electrophoresis Images. Point Matching Under Non-uniform Distortions. In *Proc. 10th Annual Symposium on Combinatorial Pattern Matching (CPM'99)*, LNCS 1645, pp. 212–222, 1999.
2. H. Alt and L. Guibas. Discrete Geometric Shapes: Matching, Interpolation, and Approximation, In J.-R. Sack, J. Urrutia, editors, *Handbook of Computational Geometry*, pp. 121–153. Elsevier Science Publishers B.V. North-Holland, Amsterdam, 1999.
3. R. D. Appel, J. R. Vargas, P. M. Palagi, D. Walther, and D. F. Hochstrasser. Melanie II - A Third Generation Software Package for Analysis of Two-dimensional Electrophoresis Images: II. Algorithms. *J. Electrophoresis* 18, pp. 2735–2748, 1997.
4. T. Eiter and H. Mannila. Distance Measures for Point Sets and Their Computation. *Acta Informatica*, 34(2):109–133, 1997.
5. M. R. Garey, D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness.* W. H. Freeman and Company, San Francisco, 1979.
6. A. V. Goldberg and R. Kennedy. An Efficient Cost Scaling Algorithm for the Assignment Problem. *Mathematical Programming*, Vol. 71, pp. 153–178, 1995.
7. D. S. Hochbaum. An Efficient Algorithm for Image Segmentation, Markov Random Fields and Related Problems. *Journal of the ACM*, Vol. 48, No. 4, pp. 686–701, 2001.
8. S. Jokisch and H. Müller. Inter-Point-Distance-Dependent Approximate Point Set Matching. *Research Report* No. 653, University of Dortmund, Department of Computer Science, July 1997.
9. J. Kleinberg and E. Tardos. Approximation Algorithms for Classification Problems with Pairwise Relationships: Metric Labeling and Markov Random Fields. In *Proc. 40th Annual IEEE Symposium on the Foundations of Computer Science (FOCS'99)*, pp. 14–23, 1999.
10. V. Mäkinen. Using Edit Distance in Point-Pattern Matching. In *Proc. 8th Workshop on String Processing and Information Retrieval (SPIRE 2001)*, IEEE CS Press, pp. 153–161, 2001.
11. M. Queyranne. Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. *Oper. Res. Lett.*, 4, pp. 232–234, 1986.
12. D. Ruprecht and H. Müller. Image Warping with Scattered Data Interpolation. *IEEE Computer Graphics and Applications*, Vol. 3, pp. 37–43, 1995.