IJCV manuscript No.
(will be inserted by the editor)

Using the Particle Filter Approach to Building Partial Correspondences between Shapes

Rolf Lakaemper
Temple University
Philadelphia, USA
lakamper@temple.edu
· **Marc Sobel**
Temple University
Philadelphia
marc.sobel@temple.edu

the date of receipt and acceptance should be inserted later

Abstract Constructing correspondences between points characterizing one shape with those characterizing another is crucial to understanding what the two shapes have in common. These correspondences are the basis for most alignment processes and shape similarity measures. In this paper we use particle filters to establish perceptually correct correspondences between point sets characterizing shapes. Local shape feature descriptors are used to establish the probability that a point on one shape corresponds to a point on the other shape. Global correspondence structures are calculated using additional constraints on domain knowledge. Domain knowledge is characterized by prior distributions which serve to characterize hypotheses about the global relationships between shapes. These hypotheses are formulated online. This means global constraints are learnt during the particle filtering process, which makes the approach especially interesting for applications where global constraints are hard to define a priori. As an example for such a case, experiments demonstrate the performance of our approach on partial shape matching.

1 Introduction

Finding correspondences between visually similar features of pairs of shapes is a classical problem in Computer Vision. For example, Belongie et al (2002) defines a threefold process for distinguishing shape similarity; his steps include (i) constructing correspondences, (ii) aligning shapes with one another, and (iii) employing measures of similarity for comparing shapes with one another. Clearly, robustly establishing correspondences is a crucial step in such a process. Our research aims at a general description of how to construct correspondences between sets of data points based on their shape properties. This includes the construction of correspondences between shapes and parts thereof. In the sequel, we equate shapes with sets of points sampled (appropriately) from them; each such point has one or more (local) features assigned to it. Examples of such local features include curvature (if the shape is defined by a surface, Mokhtarian et al (2002)), point density, shape context Belongie et al (2002), and features based on the Poisson equation given in Gorelick et al (2006). Selecting which local features to use depends on the particular setting and goals involved. The challenge is to determine correspondences between shapes using local descriptors in a globally consistent manner. This challenge includes resolving the competing claims between local and global descriptors. We do this by embedding global constraints in an iterative algorithm, learned by the system in a feedback process.

We approach the correspondence problem using Particle Filters (PF). Particle filters employ probabilities and multiple hypotheses to build correspondences between shapes.

In our setting, we assign a probability to each possible configuration of single point correspondences. It is based on the fitness of participating single point-point correspondences. We model the visual correctness of correspondences; visually better correspondences are assigned higher probability values. Under certain assumptions, this assignment leads to the calculation of probabilities for sequences of correspondences, called particles below. Probabilities attached to particles are, in conformity with statistical terminology, called 'likelihoods', below. The global consistency of particles is enforced using prior distributions. The search space consists of the set of all possible particles.

The goal of particle filters (PF) is to estimate the posterior distribution over the entire search space using discrete

Address(es) of author(s) should be given distributions (constructed dynamically at each of a number of different iterations) based on a limited number of particles. In this sense, particles represent hypotheses about what the true relationship between shapes really is. The best correspondence is the most likely particle (Maximum Likelihood Particle, MLP) surviving at the end of the PF process Liu (2002).

All particles compete in an iterative process, each iteration consisting of two steps: prediction and evaluation. In the prediction step, particles are augmented by adding single correspondences; the resulting set of particles is called the ’preliminary sample’. Correspondences are selected based on a correspondence-weight distribution, which represents single correspondence probabilities as defined by the local feature descriptors. Using Bayes rule, each particle is assigned a weight representing its strength, proportional to its posterior probability. The evaluation step selects particles from the preliminary sample using these weights. We evaluate particles using residual sub-sampling, see e.g. Liu et al (2000). This selection causes stronger hypotheses (particles) to dominate weaker ones, yet randomly permits some weak hypotheses (outliers) to survive and possibly prosper in later iterations. In this way we protect against choosing particles which are only local maxima (Liu et al (2000)).

We augment the PF algorithm by incorporating a *recede* step which destroys correspondences in particles. The introduction of a recede step insures that the entire space of (possible) particles can be visited from any given particle configuration. In the setting of sequential Markov Chain Monte Carlo, recede steps could be referred to as ’evolution’ steps. Evolution steps typically serve as transformations on already constructed particles.

Apart from the recede step, in this paper we do not focus on the question of how best to improve the PF itself. There are many different ways of designing particle filters; each such design is associated with techniques which optimize their performance, see Moral et al (2006).

The **main contribution of our research** is to introduce the use of PF to solve the correspondence problem, and, more specifically, the problem of correctly designing the prediction step. Prediction is based on an iteratively updated feature probability distribution. This distribution is composed of two parts: (i) the local part, representing the correspondence probability based on the local feature descriptors, and (ii) the global part, representing non local constraints. The global constraints can describe topological or geometrical features of the shape and are used to achieve global consistency. They are built into each iteration of the PF process, enforced using the (conditional) information available from the already constructed correspondence sequence. This has the advantage of allowing us the easy task of *learning* specific parameters of global constraints at each iteration, rather than the hard task of predicting them a priori. In general, it is easier to characterize, yet not to specify global constraints in detail. For example, for the partial shape similarity in Section 7.4.1, it is natural to expect the result to be in a certain window of the target shape. *Learning* in our system means, that we only characterize the task formulating ’there is a window’, yet leave the computation of specific parameters (in the example: the window’s location) to the feedback system feeding the ’GC update rule’ (Figure 1). We use prior distributions, built conditionally at each iteration, to enforce the learning constraints.

This leads to a system in which the likelihood of correspondences is defined via: (i) a matrix representing local constraints, and (ii) an update rule which iteratively generates a second matrix specifying parameters of global constraints. The update rule implements the feedback cycle, which enables the system to learn global constraints. Figure 1 illustrates this idea.

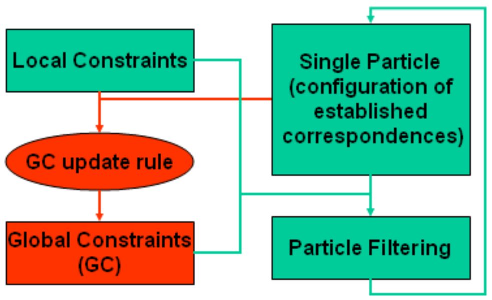


Fig. 1 Global and local constraints influence the PF process. The current particle and the local constraints are fed back to update the global constraints (’GC update rule’): global constraints are learned during the PF process.

The sections are organized as follows. After relating our work to existing approaches, we will introduce our PF approach along the example of finding partial correspondences in 2D boundary curves. We first describe the specification for local constraints (Section 3.3). Section 4 introduces the optimization problem to find consistent feature correspondences. The Particle Filter approach to solve this problem is topic of Section 5. Section 6 covers an example of the PF process. Section 7 explains how to build matrices to describe global constraints for different correspondence tasks. Specifically, Section 7.4 describes how to use the Particle Filter framework to find partial correspondences. Experiments and results can be found in Section 8.

2 Related Work

The process of finding correct correspondences can be seen as a labeling process. The features of one shape correspond to the labels; the features of the other have to be labeled.

In Rosenfeld et al (1976), Rosenfeld introduced the technique of relaxation labeling (RL) to approach this class of problems. It has since been a topic of successful ongoing research (Haralick and Shapiro (1979), Kittler and Illingworth (1986), Yefeng Zheng and David Doermann (2006)). In the soft version, a weight matrix defines the correspondence probabilities between data point and label. The property of consistency is defined in terms of certain given neighborhoods. RL is used to solve an optimization problem subject to the given consistency requirements.

RL is a gradient descent method which guarantees convergence towards some local optimum. It is an iterative, deterministic approach, highly dependent on the initial correspondence probability matrix. In this same connection, our approach can be interpreted as piecewise sequential, non-deterministic, multiple hypotheses relaxation labeling. Sequential, because of the aforementioned prediction step, which assigns a *single* label *unchangeably* to a data point. 'Piecewise', because sequences of correspondence labeling are broken by single re-labeling 'recede' steps'. By contrast, in each iteration, relaxation labeling *re-labels* all the data *simultaneously*. PF enhances strong local feature properties, while RL subsumes them, enhancing the global labeling structure. For these reasons particle filters have the drawback of being easily lured into local optima. This drawback is counterbalanced by the assumption, at each iteration, of multiple hypotheses which compete with one another.

Our research is designed as a general framework, yet the examples and results explore the particular case in which correspondences are built between 2-dimensional boundaries. Sebastian et al (2003) showed the importance of ordering constraints in matching curves. We use this constraint as an example. It shows how to integrate the constraint of contiguity conservation for the simpler case of non-partial shape matching. This special case can also be solved using Dynamic Programming (DP); successfully applied e.g. in Milios and Petrakis (2000), Scott and Nowak (2006), Schmidt et al (2007). Particle filters PF guarantee only a nearly optimal solution, as compared with DP methodology which guarantees an optimal solution in this case. Nonetheless, particle filters generalize in cases where DP methodology does not. For example, particle filters generalize easily to the task of building shape and part correspondences. Our final result shows how we learn constraints in the example of partial shape matching, used to build similarity measures for purposes of querying shape databases with shape parts.

Since local descriptors are imprecise, the optimum, calculated using these descriptors, is not necessarily a more accurate solution; an example is given in Section 8.1. We note that the prediction step for particle filters, described above, samples a new correspondence from the possible correspondences; this selection does not depend on any implicit or explicit ordering of features. In contrast to the DP approach in Scott and Nowak (2006), we do not need a designated start correspondence and handle shape-reflection (clockwise/anti clockwise order of boundary points) automatically. We also don't need additional parameters characterizing the minimum number of correspondences.

RL is a common way of computation for Markov random fields (Qian and Titterington (1992)). Typically, this involves specifying a prior over the search space of particles at any given iteration conditional on all others. This differs from the priors advocated here which are defined conditional on previous iterations. We leave comparisons between this approach and ours to future work. Another single hypothesis probabilistic approach to find correspondences is given in Alt et al (2006). It evaluates shape matchings based on properties of the correspondence set itself, therefore focusing on the global shape properties.

An interesting example of specific 2D-boundary matching is given in Chen et al (2008), applying a local-sequence alignment algorithm originating from bio-informatics to computer vision. Chen et al (2008) successfully strives for enhanced performance on the particular problem of boundary matching, demanding for the ordered point sequence as a necessary condition. Specializing on this problem, such an algorithm can achieve faster performance (in Chen et al (2008): $O(mn)$), yet is not easily extendable. In contrast, our PF system framework is based on feature matching with a high versatility: the presented experiments on boundary part matching (Section 8) should be interpreted as examples. The input to the core PF system is a feature correspondence probability matrix, describing local feature correspondences, as well as a function to evaluate the global consistency of hypotheses (GC update rule in Figure 1). The design of the input depends on the application, the core PF system (right side in Figure 1 works independently.

A PF based point cloud alignment system is presented in Sandhu et al (2008). Based on an Iterative-Closest-Point (ICP) update step for each particle, it finds the optimal transformation of rigid bodies to minimize a least-square fitting target function. In a nutshell this system can be described as multiple hypotheses ICP. Finding the optimal embedding of a query part in the target shape's 3D coordinate system, it leads to impressing results yet is limited to rigid body transformation, since the underlying feature used for point correspondences is the points' location. In contrast, our PF system finds feature correspondences based on a feature-distance function, which is more versatile than the direct location comparison. See Section 8.3 for an example where a spatial alignment system like Sandhu et al (2008) would be likely to fail.

Particle filters have been successfully utilized in both Computer Vision (mostly for object tracking, Rathi et al (2007) and image segmentation, de Bruijne and Nielsen (2004)) and robot mapping, Thrun (2002). An interesting example, in a broader sense related to our system due to application of Particle Filters to solve an optimization problem, is a contour extraction algorithm (JetStream) presented in Perez et al (2001). In there, the target function is a functional over the search space of plane curves, taking into account certain optimality features to achieve smooth, edge fitting curves. The impressing results demonstrate the applicability of the probabilistic PF approach to Computer Vision problems.

PF, if used to solve optimization problems as in our case, are similar to general Evolutionary Algorithms (EA) Holland (1975), Goldberg (1989), Back et al (1997), in the sense that determining peaks of a distribution can be interpreted as an evolution strategy, or 'survival of the fittest'. The estimation of a distribution, which PF aim at, is utilized to make a delayed decision about the selection of an optimum. There are many analogies between PF and EA. Relating our specific PF case to EA, each individual represents a hypothesis of a configuration of correspondences between points of the query and target shape. The weight function provides a fitness, or aptness for each individual, which reflects the local feature correspondence fitness, the global configuration consistency and the completeness of the configuration. The individuals develop in an augmenting manner to find a single optimal individual: starting with no correspondences, the individuals add single, promising correspondences. This enables the approach to focus on likewise promising, relatively small subsets of the huge search space. To compensate effects of development aberration, we allow the algorithm to step back in the development of individuals: a 'recede' step allows individuals to delete inconsistent correspondences of their configuration, which might have been chosen prematurely in earlier stages of development.

Finding correspondences is closely related to shape matching: having correspondences at hand, shape matching is achieved by adding alignment and shape-similarity steps Belongie et al (2002). In shape matching, these three steps do not necessarily have to be separated, and are often combined taking advantage of special features of specific applications. Examples of shape matching algorithms can be found in Ling and Jacobs (2007), using the 'inner distance' between boundary points as a basis for shape feature description and shape similarity. Siddiqi et al (1999) derives shape features and similarities from 'shock graphs', based on a curve evolution process, acting on bounding contours. Using evolutionary properties of boundary curvature turning points on multiple scales, Mokhtarian et al (1996) introduced the 'curvature scale space' for shape boundary comparison, which is the introductory work for the previously mentioned paper Mokhtarian et al (2002). A related approach, yet more tailored towards the discrete nature of shapes represented by polygonal boundary curves, Latecki et al (2000) derives a boundary partitioning scheme and matches parts of complete shapes using a dynamic programming approach. A survey of shape-matching approaches is e.g. given in Veltkamp and Hagedoorn (2001).

General information about particle filtering is given e.g. in Doucet et al (2001), Liu (2002).

2.1 Notation

If we deal with closed shape boundaries, all index-math is understood module n_i, the number of points on the boundary. Throughout the paper, $\phi_\sigma(x)$ will denote the mean 0 Gauss distribution with standard deviation σ of a random variable x. We use ϕ 0-normalized, i.e. $\phi_\sigma(0) = 1$. $|g|$ denotes the cardinality of a set g. For matrices $M_1, M_2,\ M_1 \odot M_2$ denotes the element-wise multiplication. For all colored figures the color scale is equal to Figure 7, right.

3 Local Constraint: Features and Correspondence Matrix

In our setting, a shape $S_i = v_1, .., v_k$ is represented by a uniformly sampled $2d$ boundary polygon with k vertices (see Figure 2). We note that different shapes can have different numbers of vertices; to each of these vertices we assign a number of different feature descriptors.

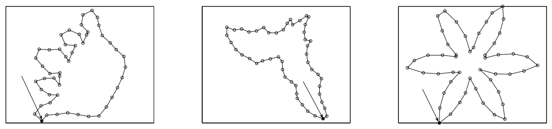


Fig. 2 Objects crown-01, bat-01 and device1-01 from the MPEG-7 data set. The starting point v_1 is marked, the objects are uniformly sampled counter clockwise using 50 points.

The combination of local feature descriptors will be used to generate a matrix of local correspondence likelihoods between pairs of points $(v_i, u_j) \in S_1 \times S_2$. The current implementation applies two feature descriptors: *curvature* and *local distance-signature* (these are examples for descriptors, it is not focus of this research to find or explore optimal local descriptors). Their descriptive power is limited, yet they reveal a certain amount of local shape information. The limitation of these descriptors is reflected in the ambiguity implicit in the correspondence between vertices: a single vertex of shape S_1 can have a large number of possible partners (vertices with similar features) in the second shape S_2 and vice versa. In this setting, insufficient regional or global descriptive information provided by local feature descriptors is not necessarily a drawback. A certain amount of ambiguity allows for a wider variety of possible local hypotheses for each particle.

Sections 3.1 and 3.2 describe the local shape features used in this setting to find a (partial) correspondence between shape boundaries. Section 3.3 introduces and describes the local correspondence matrix. The local correspondence matrix is constructed from shape features.

3.1 First Local Feature: Local Distance Signature

Let the shape S consist of k vertices, $S = \{v_i, .., v_k\}$. The *local distance signature* $LDS(v_i)$ of vertex v_i is a scalar value that contains the normalized weighted average Euclidean distance between v_i and all other vertices $v_j \in S^- = S \setminus \{v_i\}$. The weight w_{ij} of the vertex $v_j \in S^-$ depends on the boundary distance $d_b(v_i, v_j)$ (the shortest distance between v_i and v_j on the boundary polygon); $(j = 1, .., k; j \neq i$. The aforementioned weights define a neighborhood of influence for v_i; they are defined by:

$$w_{ij} = \phi_{\sigma_w}(d_b(v_i, v_j)). \tag{1}$$

We will describe below how to determine σ_w. Below, we use the notation, $d_e(v_i, v_j)$ to describe the Euclidean distance between v_i,v_j. The weighted (not normalized) average distance $lds(v_i)$ (between v_i and S^-) is defined by

$$lds(v_i) = \frac{\sum_{j=1}^{k} w_{ij} d_e(v_i, v_j)}{\sum_{j=1}^{k} w_{ij}}. \tag{2}$$

LDS is the normalized and therefore scale-independent version of lds. We require that $LDS(v_i)$ is 1 for a point v_i with average LDS *in its neighborhood*. Hence, we define LDS to be the weighted average of lds.

$$LDS(v_i) = \frac{lds(v_i) \sum_{j=1}^{k} w_{ij}}{\sum_{j=1}^{k} w_{ij} lds(v_j)} \tag{3}$$

See Figure 3 for an example of a shape and its LDS values. LDS is designed for the purpose of partial matching; it takes into account only a certain neighborhood around each given vertex. The size of this neighborhood is characterized by the parameter, σ_w; this parameter reflects regional properties of the given vertices. The value of σ_w depends on the performed task: if entire shapes are matched (i.e., non-partial matching), large values of σ_w should be used. In this case, LDS reflects global shape properties. Smaller values of σ_w reduce the information contained in LDS. The pathological case in which σ_w is taken to be very small, leads (via equation the aforementioned equation) to the assignment of LDS values which are 1 for all vertices. In contrast, for partial matching (i.e. matching a full shape with a part (of itself)), a smaller neighborhood characterized by smaller values of σ_w have to be assumed. In case of partial matching, we take σ_w to be 20 percent of the boundary length of the full shape. Experiments show that such a value includes sufficient information, (see the example in Figure 3). In such a case $\sigma_w = 0.2$ already contains significant information, since it accurately characterizes global properties, yet it is small enough to react to regional properties.

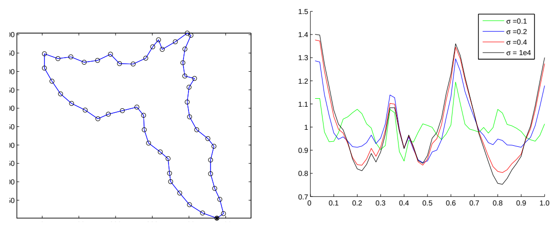


Fig. 3 LDS (right) of a shape (left) using different values for σ_w, boundary length of shape is 1.0. A neighborhood defined by $\sigma = 0.2$ already contains significant information, which can be seen by its structural similarity to the global case, $\sigma \to \infty$. The shape (left) consists of 50 points, starting with the bottom-most point, ordered counter-clockwise.

3.2 Second Local Feature: Curvature

The *curvature* $CV(v_i)$ is the signed turn-angle as measured from vertex v_i, assuming an anticlockwise orientation. Convex vertices have negative curvature values (i.e., they are the vertices joining an oblique angle). $CV(v_i) \in (-\pi..\pi)$. To obtain a more stable curvature description, we smooth the curvature values using a gaussian blur filter (selection of curvature algorithm is not critical, different techniques lead to similar results).

3.3 Local Constraint: Feature Correspondence Matrix

While LDS and CV reflect properties of a single shape, the feature correspondence matrix reflects the correspondence probability of point pairs $(v_i, u_j) \in S_1 \times S_2$, based on fitness of the local features LDS and CV. This matrix defines the local constraints of the point correspondence problem.

We assume two shapes S_1,S_2 with k_1,k_2 vertices respectively and define the *set of correspondences* $\mathcal{C}$ as set of all pairs of vertices of S_1 and S_2.

$$\mathcal{C} = \{(v_i, u_j) | v_i \in S_1, u_j \in S_2\} = S_1 \times S_2$$

We assume a correspondence probability function $\mathcal{P}_\mathcal{C}$ over $\mathcal{C}$. $\mathcal{P}_\mathcal{C}$ is defined by the *local correspondence matrix* $L = [l_{ij}]$. L is a $k_1 \times k_2$ matrix containing the correspondence probabilities, i.e. $\mathcal{P}_\mathcal{C}(v_i, u_j) = l_{ij}$.

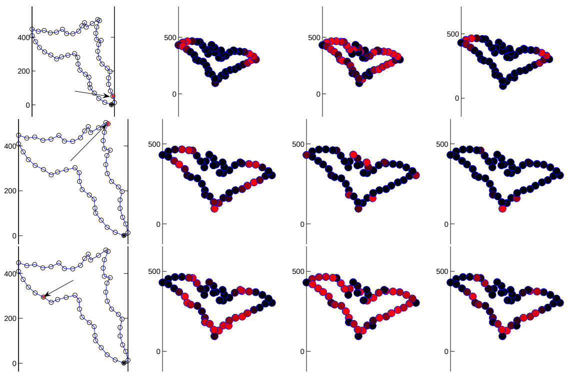


Fig. 4 Properties of LDS and CV. Left column: shape S_1, a single vertex $v_{i'}$ highlighted (arrow). Second column: $L_{LDS}(i'j)$, the LDS based correspondences between $v_{i'} \in S_1$ and all vertices $u_j \in S_2$ (see Equation 5). Stronger red denotes higher correspondence probability. Third column: $L_{CV}(i'j)$ Column 4: joint probability L. All colorings are normalized, i.e. use the entire range of black to red, therefore they do not reflect absolute values. Top row: while LDS gives a good estimation, CV is ambiguous. L shows good fits due to L_{LDS}. Second row: LDS and CV both lead to distinct correspondences, L shows a high probability for the correct fit. Bottom row: Both, LDS and CV are ambiguous, leading to a variety of correspondences in S_2. Shapes used are bat-01 and bat-05 of the MPEG-7 test set, uniformly sampled with 50 points.

The probabilities are based on the local feature descriptors LDS and CV described in sections 3.1 and 3.2. In general, each feature descriptor generates its own probability matrix, the final local correspondence matrix L is the joint probability, i.e. the element-wise product of the respective matrices. In our case, we first obtain two correspondence matrices L_{LDS} and L_{CV}, based on LDS and CV respectively. L is then defined by $L = L_{LDS} \odot L_{CV}$.

The following will describe how to compute L_{LDS}. L_{CV} is obtained accordingly.

3.3.1 Correspondence Matrix based on LDS

We first compute the correspondence likelihood based on the distance between LDS features. Let $L^1 = [l^1_{ij}]$ be a $k_1 \times k_2$ matrix with entries

$$l^1_{ij} = \phi_{\sigma_l}\left(\frac{|LDS(v_i) - LDS(u_j)|}{LDS(v_i)}\right). \tag{4}$$

(The index '1' in 'L^1' points out that LDS is the first of two local descriptors, without regard for order). σ_l is a scale independent parameter that we determine experimentally (we use $\sigma_l = 0.1$). The values l^1_{ij} are normalized by $LDS(v_i)$. l^1_{ij} describes the likelihood of the correspondence pair $v_i \in S_1 \rightarrow u_j \in S_2$. We define L^1_r as the row-normalized version of L^1 for purposes of constructing the directed correspondence probability of $v_i \rightarrow u_j$.

For reasons of symmetry, we compute the directed correspondence probability of $u_j \rightarrow v_i$ analogously, i.e. L^1_c is computed by replacing $LDS(v_i)$ with $LDS(u_j)$ in the denominator of eq.4. This is followed by column normalization.

Finally, L_{LDS} is the normalized joint probability

$$L_{LDS} = \frac{L^1_r \odot L^1_c}{\sum_{i=1}^{k_1}\sum_{j=1}^{k_1}(l^r_{ij} l^c_{ij})}. \tag{5}$$

Definition of L_{LDS} by L^1_r and L^1_c is motivated by the view that L^1_r reflects the probability of correspondence between points v_i and u_j *in the context of* S_2, while L^1_c reflects the probability of correspondence between v_i and u_j *in the context of* S_1. Figure 5 illustrates the necessity of this step: two similar features in two different shapes can have different correspondence likelihoods, depending on the number of similar features in the other shape. L_{LDS} is defined in this way for purposes of ensuring cognitive symmetry; we do not draw a distinction between query and target shape.

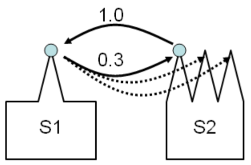


Fig. 5 Different correspondence probabilities in different directions (hypothetical example, values not based on actual local features): $S_1 \rightarrow S_2$: the correspondence probability between the highlighted points is $1/3$, since the point in S_1 can find 3 equal partners in S_2. In contrast, in direction $S_2 \rightarrow S_1$ there's only a single partner for the point in S_2, hence the correspondence probability is 1.

The feature correspondence matrix L_{CV}, based on the curvature feature, is derived accordingly. Figure 4 shows different correspondence cases to illustrate the behavior of LDS and CV.

3.3.2 Local Correspondence Matrix L

L is the element-wise product of L_{LDS} and L_{CV}:

$$L = L_{LDS} \odot L_{CV} \tag{6}$$

In general, L is computed by the joint probability of an arbitrary number n of feature descriptors, i.e.

$$L = L^1 \odot L^2 \odot .. \odot L^n.$$

L has the following properties:

- L reflects the probability of correspondence between $v_i \in S_1$ and $u_j \in S_2$.

- L reflects the correspondence probability based on all local feature descriptors.
- With $S_1' = S_2$ and $S_2' = S_1$, $L(S1, S2) = L^T(S_1', S_2')$; L is order invariant (except transposition) with respect to S_1, S_2. This of course does *not* mean that L is a symmetric matrix. For matrix-symmetry the following holds: L is symmetric $\Leftrightarrow S_1 = S_2$.
- The correspondence matrix L of two identical shapes $S_1 = S_2$ is not necessarily diagonal dominant. However, more similar shapes result in stronger diagonals.
- In the case of closed boundaries, the matrix must be interpreted horizontally and vertically periodic, i.e. it is a torus (for a nice visualization see Schmidt et al (2007)). Although this is not important to the interpretation of L as a correspondence probability distribution, it has an effect on all operations involving the matrix-topology, e.g. contiguity conservation, see Section 7.

Figure 6 shows examples of correspondence matrices.

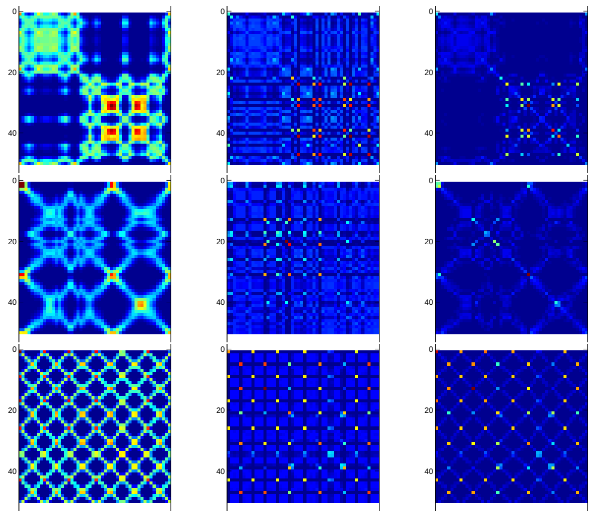


Fig. 6 Correspondence matrices L_{LDS}, L_{CV}, L (columns 1,2,3) for objects crown-01 (top row) and bat-01 (middle row) and device1-01 (bottom row). For simplicity, we here compared the object to themselves (see Figure 2 for objects); i.e. top row=crown∼crown, middle row=bat∼bat, bottom row=device∼device. Top row: The south-east area of the crown matrices contain more information, since the shape features (spikes) are more distinct. Since the first 25 vertices contain no distinct information, the diagonal in the North West part is weak. Middle row: the 3 distinct spots on the diagonal of L_{LDS} are the wing-tips and the tail. L has a relatively strong 2nd diagonal, caused by global symmetry in the shape with respect to the axes through the wing-tips. Bottom row: the effects of symmetry and spikes are clearly visible in the matrices, showing high ambiguity in L. The local ambiguity has to be solved by the Particle Filter system using global constraints.

4 Correspondences and Groupings

As mentioned in Section 3.3, the matrix L defines a probability $P_{\mathcal{C}}$ over the set $\mathcal{C}$ of correspondences: $P_{\mathcal{C}}(v_i, u_j) = l_{ij}$.

A *Grouping* $g \in \mathcal{G}$ is a member of the power set $\mathbb{P}\mathcal{C}$ of $\mathcal{C}$. A grouping defines a configuration of correspondences. $\mathcal{G}$ defines the search space for our PF process. Each element $g \in \mathcal{G}$ takes the form $g = \{\{v_{i_1}, u_{j_1}\}, .., \{v_{i_k}, u_{j_k}\}\}$. Further constraints (e.g. contiguity conservation, see Section 7) on groupings can limit the search space to a subset $\mathcal{G}^- \subset \mathcal{G}$.

A grouping g is *complete*, if it is maximal with respect to the containment ordering in the set $\mathcal{G}^-$. In quantitative terms, $\forall g' \in \mathcal{G}^- : \; g \subset g' \rightarrow g = g'$

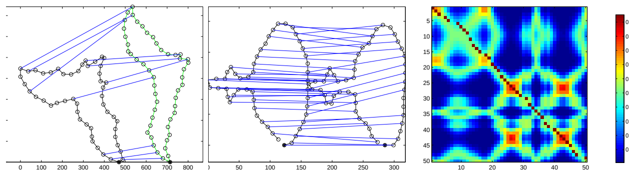


Fig. 7 Left: Incomplete grouping, each blue line shows a single correspondence. Center: A complete, strongly contiguity conserving grouping. Right: Connectivity matrix of the grouping (center), superimposed over correspondence weight matrix L. Each red dot is a single correspondence (red dot means connectivity matrix=1). Note that the grouping is complete although not all points are participating.

Figure 7 shows examples of both non-complete and complete groupings. These are computed under the constraint of strong contiguity conservation (see Section 7). We use the notation, $\mathcal{G}^-$ for the set of contiguity conserving groupings. We interpret groupings using the edges of a graph with vertices $S_1 \cup S_2$. Figure 7 displays this graph in two ways: (i) it is superimposed over the correspondence probability matrix L; and (ii) edges are drawn using the connectivity matrix L.

4.1 Optimal Sets of Correspondences

We define the weight of a grouping as,

$$\mathcal{W}_{\mathcal{G}}(g \in \mathcal{G}) = \prod_{l=1}^{|g|} \exp\left(P_{\mathcal{C}}(v_{i_l}, u_{j_l})\right) = \prod_{l=1}^{|g|} \exp\left(l_{i_l, j_l}\right). \quad (7)$$

We formulate the correspondence problem as one of choosing the complete grouping, $\hat{g} \in \mathcal{G}^-$ from the set of constrained groupings, $\mathcal{G}^-$ with maximal weight or, more specifically,

$$\hat{g} = \arg\max_{g \in \mathcal{G}^-} (\mathcal{W}_{\mathcal{G}}(g)). \quad (8)$$

Lemma 1 *The optimal grouping $\hat{g}$ is complete.*

Proof This is a direct consequence of the fact that correspondence weights are defined to be larger than 1:
$P_{\mathcal{C}}(v_{i_l}, u_{j_l}) \in [0..1] \Rightarrow exp(P_{\mathcal{C}}(v_{i_l}, u_{j_l})) > 1$.

The lemma shows that groupings with more correspondences dominate those with fewer correspondences. This is in contrast to the joint probability, $\prod(l_{i_l j_l})$, which decreases as the number of correspondences increases.

The optimal grouping can therefore be understood as a search for a grouping with as many connections as possible, subject to optimizing the overall weight. Dynamic Programming (DP) methodology in Scott and Nowak (2006) constructs optimal shape matches under the constraint of contiguity conservation; this is a special case of the aforementioned optimization problem. In contrast to the DP approach, which gives smaller weight to groupings with larger numbers of correspondences, we need no additional parameters to ensure that the optimal grouping has some minimal number of correspondences. This is a consequence of our use of complete groupings.

Specific tasks of shape matching, like the matching of parts, require far stronger constraints. Typically, in this case, additional domain knowledge is required to achieve success. Particle filters are well designed to properly formulate and use this knowledge; we employ them below for this purpose. It will be seen that this approach provides more flexibility in solving the optimization problem when these additional constraints are present. PF enable us to learn constraints during the iteration process. This enhances our ability to find better particles.

Intuitively, the advantage of the PF approach follows from the fact that PF is able to track preliminary optimal solutions of a *changing* weight distribution which, in turn, define a changing target function during the optimization process. Using this property we are able to interpret the optimization task eq.8 in a better way. We do not optimize the target function $\mathcal{W}(g)$, which is based on a static, non-changing weight matrix L and a restricted search space $\mathcal{G}^-$. Instead, we dynamically adjust the weight function based on the current state of the particle and an inferred global constraint. The search space is restricted to correspondences with probability greater than 0. Expressing the global constraints as an iterative adjustment to the local constraints allows us to infer the global constraints themselves in an iterative manner. We therefore do not have to precisely define the global constraints a priori. Short: the presented framework allows for adjustability of the cost function as you move along in the optimization, while standard optimization techniques have a fixed energy function. Section 7 will give an example of different global constraints and their usage in the PF system, compared to classic approaches.

This is helpful in cases where such definitions are difficult to use. Partial matching provides an example of this. Section 7.4 explains our solution to this problem. We use hypothetical, iteratively inferred global constraints.

5 Near Optimal Labeling using Particle Filters

Below, we refer to a grouping $g \in \mathcal{G}^-$ as a single particle. We employ the notation, $g_{1:t}$ for a particle at time (or iteration) t. Particles are built by adding single correspondences at each iteration. Correspondences are selected based on a correspondence probability W^t. W^t is an updated version of the correspondence-weight distribution L. The update of $L \rightarrow W^t$ is defined by additional global constraints, which will be explained in Section 7. The following sections will explain the prediction and evaluation step of PF in our setting, as well as the new step of *recede*. In what follows we use the definition: all correspondences are *admissible* at iteration $t = 1$. At iteration $t > 1$, a correspondence $c \in \mathcal{C}$ is admissible if, for a given particle, $g_{1:t-1} \in \mathcal{G}^-$ (at iteration $t-1$), the particle $g_{1:t} = g_{1:t-1} \cup c$ is in $\mathcal{G}^-$. Admissibility therefore means being in accord with global constraints.

While this definition of admissibility by subset relation is a hard definition, Section 7 will introduce a softer definition, using global constraints defined by probabilities. The case of hard admissibility is modeled using probability values of 0 and 1.

5.1 Prediction Step

The prediction step consists of 2 parts: a) select a correspondence based on the updated probability distribution W^t over all admissible correspondences $c \in \mathcal{C}$ at iteration t. b) compute the posterior probability of the resulting augmented particle.

5.1.1 Distribution for Correspondence Selection

W^t is the updated version of L (at iteration t) incorporating global constraints given by a matrix, $GC(g_{1:t})$. $GC(g_{1:t})$ itself depends on the particle $g_{1:t}$ at each iteration t. In this sense, we have,

$$W^t = L \odot GC(g_{1:t}). \tag{9}$$

Note that $GC(g_{1:t})$ changes from one particle g to the next and from one iteration t to the next. This enables us to adjust or learn constraints during the PF process. Section 7 will give examples of such constraints. For single correspondences we use either the notation c_t, if we want to stress that a correspondence is selected at time t, or c_{ij}, to denote the event that a correspondence between i and j is selected. We refer to the update of a grouping g using the notation: $g_{1:t} \rightarrow g_{1:t+1} = g_{1:t} \cup c_{t+1}$. We employ the notation, w^t_{ij}, or, short w_c^t, for the log-likelihood associated with the selection of the correspondence c_{ij} (or c_t) at time t. Correspondences are selected according to the marginal probabilities

$$P(c_{ij}) = \frac{\exp(w_{ij}^t)}{\sum_l \exp(w_{il}^t)}$$

5.1.2 Why Particle Filters are Useful for Constructing Shape Correspondences

As mentioned before, each particle $g \in \mathcal{G}^-$ is a set of correspondences, where $\mathcal{G}^-$ is the state space that contains all admissible particles (=groupings). We use the terminology $\mathcal{G}_t^-$ for the set of all particles $g_{1:t} = (c_1, c_2, .., c_t)$ having t correspondences. The goal is to find a good set of correspondences that is complete. At each stage, as a consequence of admissibility (global consistency), the weights w_c^t attached to correspondences are altered, i.e. for correspondence c_{ij}, in general $w_{ij}^t \neq w_{ij}^{t+1}$. We use the notation π_t for the probability measure, defined on $\mathcal{G}_t^-$ having the property that it is proportional to the product of the weights associated with the correspondences.

$$\pi_t(g_{1:t}) \propto w_{c_1}^t ... w_{c_t}^t \text{ such that } \sum \pi_t(g_{1:t}) = 1 \; ; \; t = 1..T \tag{10}$$

The dynamic model is specified by the probability of $g_{1:t}$ divided by the probability of $g_{1:t-1}$.

$$P(c_t \mid g_{1:t-1}) = \frac{\pi_t(g_{1:t})}{\pi_{t-1}(g_{1:t-1})} \tag{11}$$

Assume that the index of the maximum grouping is T. The goal is to find $g_{1:T}$ which maximizes

$$\begin{aligned} P(g_{1:T}) &= \prod_{t=1}^T P(c_t \mid g_{1:t-1}) \\ &= \prod_{t=1}^T \tfrac{\pi_t(g_{1:t})}{\pi_{t-1}(g_{1:t-1})} \\ &= \pi_T(g_{1:T}) \\ &\propto \prod_{t=1}^T w_{c_t}^T. \end{aligned} \tag{12}$$

This problem is a standard one in the Particle Filter literature, see e.g. DelMoral et al (2007). Methods for constructing particle filters of this sort are also discussed in Doucet et al (2001) and Doucet et al (2006).

We use the notation $A[g_{1:t-1}]$ for the set of correspondences c_t which make the grouping $g_{1:t}$ possible. Our Particle Filter, at time t, selects correspondences according to the probability

$$\hat{P}(c_t \mid g_{1:t-1}) = \frac{w_{c_t}^t}{\sum_{y \in A[g_{1:t-1}]} w_y^t}. \tag{13}$$

But this probability $\hat{P}$ has the property that it approximates $\frac{\pi_t(g_{1:t})}{\pi_{t-1}(g_{1:t-1})}$. This is a consequence of the fact that, approximately, for $t = 1..T$

$$\pi_t(g_{1:t}) \approx \pi_{t-1}(g_{1:t-1}) \frac{w_{c_t}^t}{\sum_{y \in A[g_{1:t-1}]} w_y^t}. \tag{14}$$

So, choosing correspondences according to the probabilities $\hat{P}(c_t \mid g_{1:t-1})$, $(t = 1..T)$ serves to generate groupings which make the product of weights, given in equation 12, large (see Crisan and Doucet (2002) for a more complete discussion on the convergence properties of particle filters in this setting).

Please observe that equation 14 only holds if the weights $w_{c_t}^t$ don't change significantly over the iterations t *for the selected correspondences* c_t. This poses a constraint on the weight update, defined by the global constraint: the most globally consistent correspondences must not be changed significantly. Examples for global constraints obeying this constraint are given in sections 7 and 8.8: while Section 7 introduces 'hard' constraints, where weights belonging to admissible correspondences are multiplied by 1, others by 0 (which surely does not change the weights of admissible correspondences), Section 8.8 introduces a 'soft' update, multiplying the weights with a consistency in $0..1$ (1 = consistent): the weights of most likely correspondences to be selected are only minimally changed, while inconsistent correspondences have a significant weight change, but become increasingly unlikely to be selected.

5.2 Evaluation Step

We use the standard evaluation technique of residual re-sampling (see, Liu et al (2000)). We do not focus below on optimizing the evaluation step for our correspondence task.

5.3 Extension of Classical PF: Recede Step

We augment the algorithm by incorporating a *recede* step which 'destroys' correspondences in particles. The introduction of a recede step is crucial to insuring the property of 'irreducibility' for the underlying algorithm. The property of irreducibility insures that any particle can be reached from any other with positive probability. Irreducibility is a crucial feature of all Markov Chain Monte Carlo (see Liu et al (2000)) Particle Filter algorithms; it insures that surviving particles converge (in distribution) to the maximum value of equation 7 The introduction of new proposal steps is typically accompanied by the use of Metropolis Hastings based move probabilities. Such probabilities are designed to (randomly) determine whether to accept a recede proposal. Their use in this context was deemed unnecessary for two reasons: (i) The particle filter construction itself automatically gets rid of unlikely particles resulting from such a step, (ii) The way in which particle weights are calculated makes it difficult to properly compare particle fitness before and after recession: the number of correspondences per particle plays a crucial role in determining particle survival. As a consequence, it is important to insure that the particles have similar numbers of correspondences.

Recede steps can be implemented in many different ways. For the aforementioned reasons, we have designed the recede step in such a way that every r'th iteration, a certain number $d < r$ of randomly selected correspondences are destroyed for each and every one of the particles. This guarantees that, in early iterations, all particles have the same number of correspondences. In later iterations, the number of correspondences per particle depends on the property of completeness; we allow this to vary so that particles with larger numbers of correspondences dominate. In the current implementation, r is set to 10, and 5 correspondences are destroyed. Figure 8 shows an example of a particle undergoing a recede step. The improved performance due to the recede step is demonstrated in Figure 17, Section 8.1.

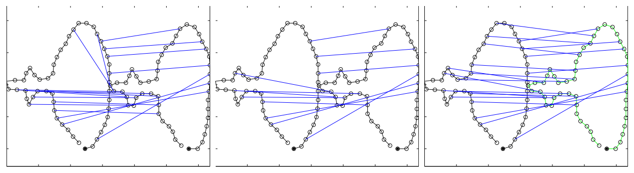


Fig. 8 Result of *recede* step: Left: before recede. Middle: directly after recede, 8 randomly selected correspondences removed from particle. Right: 8 iterations later. The particle is re-built in a more consistent way, containing the same number as the grouping left. The final result can be seen in Figure 7, center

Since we do want to destroy strong correspondences (to escape local optima) as well as weak ones (to refine a configuration), the selection of correspondences to be destroyed is independent of the strength of particular correspondences.

5.4 Role of Local and Gobal Constraint Matrices in the PF Process

This Section concludes the usage of L, W^t and $GC(g_{1:t})$ during the PF process.

- L contains correspondence probabilities based on local features and defines the local constraints. Together with $GC(g_{1:t})$ it is used to generate the distribution for correspondence selection W^t (eq.9)in the PF update. Each particle uses the same matrix L, hence L can be computed offline prior to the PF process.
- W^t is the distribution for correspondence selection, a joint probability of L and $GC(g_{1:t})$, see eq.9.
- $GC(g_{1:t})$ contains correspondence probabilities based on global features. Since $GC(g_{1:t})$ usually depends on $g_{1:t}$, it is in general not known a priori but generated during the PF process. The rule to build $GC(g_{1:t})$ defines the feedback learning process between the particle and the global constraint at time t ('GC update rule' in Figure 1. Each particle uses its own global constraint matrix. In special simple cases, GC can be independent of $g_{1:t}$ and known a priori. The simplest example is the case of no global constraints, where GC consists entirely of 1s in every iteration t.

Note that the PF approach, given in this paper, is a general framework for different correspondence tasks. *Tasks only differ in the definition of the matrix L and the update rule for GC used to generate the correspondences*. Section 7 provides examples.

6 Extended PF Algorithm and Example

6.1 Algorithm

We will now define the general PF process to estimate the Maximum Likelihood Particle. $g^i_{1:t} \in \mathcal{G}^-$ denotes the ith particle in iteration t, $G_{1:t}$ the set of all particles in iteration t. The algorithm follows the classic steps of prediction and evaluation and is extended by the additional recede step.

1)**INIT**: t=1, $g^i_{1:t} = \emptyset \;\; \forall i = 1..m$= number of particles. $W^t = L$. Init r for the recede-step (see Section 5.3).

2)Prepare the constraint matrices $GC(g^i_{1:t})$ for $i = 1..m$ and compute $W^t_i = L \odot GC(g^i_{1:t})$

3)Select a correspondence $c^i \in \mathcal{GC}$ based on the distribution W^t_i.

4)**PREDICTION**: compute posterior distribution (weight of particle) $P(g^i_{1:t+1}|c^i)$, (see Section 5.1.2).

5)normalize weights: $P(g^i_{1:t+1}) \leftarrow \frac{P(g^i_{1:t+1})}{\sum_{j=1}^m P(g^j_{1:t+1})}$

6)**EVALUATION**: compute new set of particles $G_{t+1} \leftarrow RRS(G_t)$ using residual re-sampling (RRS) preserving most probably those particles with dominant weight.

7)**RECEDE**: if $\mod(t,r) = 0$ delete $d < r$ correspondences in each particle in $G_{1:t}$ (see Section 5.3).

8) LOOP: **if** not all particles are complete: $t \leftarrow t+1$, return to step 2 **else** return particle $\hat{g}_{1:t} = argmax_{g_{1:t} \in G_t}(P(g_{1:t}))$ with maximum weight to represent a near optimal solution.

6.2 Example

We want to illustrate the PF algorithm using a simple example. We will use the case of correspondences between two shapes, represented by closed boundaries under the global constraint of strong contiguity conservation (for construction of global constraint matrix see also Section 7.3). The shapes $S_1 = (v_1, .., v_{10})$, $S_2 = (u_1, .., u_7)$ are uniformly sub-sampled using 10 and 7 points respectively, see Figure 9, left. The vertices in each shape are numbered from the bottom-most point counter clockwise. The local correspondence matrix L was built using the curvature feature descriptor only. Figure 9, right, shows L, clearly highlighting the distinct feature points v_1, v_4, v_7 in S_1 and their counterparts u_1, u_3, u_5 in S_2: L suggests mainly the (ambiguous) correspondences
$(v_1, u_1), (v_1, u_5), (v_4, u_3), (v_5, u_1), (v_7, u_5)$.

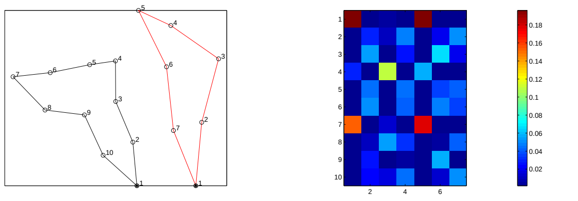


Fig. 9 Shapes S_1 (black), S_2 (red) and local correspondence matrix L.

To keep the example simple, the PF process is performed without recede step, using 4 particles. Each particle is initialized to be ’empty’. The global constraint of strong contiguity conservation has no influence up to and including the choice of the third correspondence, since we also allow reflection (flipping) of the shape. Hence the first 3 correspondences are drawn from the distribution solely defined by L. Figure 10 shows the PF process, iterations 3 to 7. Each row shows the surviving 4 particles after residual sub-sampling, the columns are sorted by particle weight from strong to weak. Each sub-Figure shows a particle consisting of the grouping (=correspondences, blue lines), along with its weight. The arrows between the sub-figures mark the evolution of each particle in the next iteration. The double framed sub-figures mark the line of inheritance of the winning particle $\hat{g}$, which is shown at the bottom left in Figure 10 and in Figure 11, left. Observe that the correspondences in $\hat{g}$ lead to alignment with a reflected version of S_2.

Table 1 shows the correspondences along with their weight in the order they were established, Figure 12 illustrates the corresponding connectivity matrix of $\hat{g}$ in iterations 3 to 5. The first three rows of the table are the correspondences of the double framed sub-Figure in Figure 10, top row. Each succeeding row in the table defines an additional correspondence of the double-framed sub-figures in the successive rows in Figure 10, e.g. correspondence $(4, 3)$ is the correspondence added by the update step to the framed sub Figure in row 1 to gain its successor, the framed sub Figure in row 2. Although the particle starts with a weak (yet correct) correspondence, it survives and evolves into the winning particle establishing stronger correspondences later.

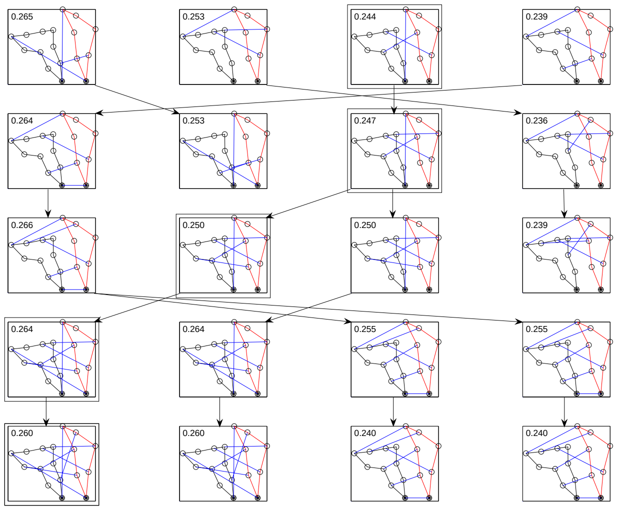


Fig. 10 PF process. The rows show the 4 particles in iterations 3 to 7 after residual sub-sampling. The particles in each row are sorted by weight (strongest first). Arrows show evolution of particles. The evolution of the winning particle can be followed by looking at the double framed sub-figures. See text for further details.

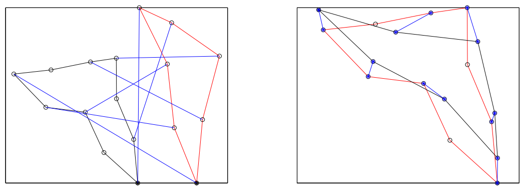


Fig. 11 The winning particle $\hat{g}$ (left). Procrustes alignment using the correspondences of $\hat{g}$ (right).

Iteration i	Corr. (v_{k_i}, u_{j_i})	Weight $L(v_{k_i}, u_{j_i})$
1	(5,2)	0.045
2	(9,6)	0.056
3	(1,5)	0.195
4	(4,3)	0.110
5	(8,7)	0.043
6	(7,1)	0.152
7	(2,4)	0.048

Table 1 Correspondences of the winning particle $\hat{g}$ during the PF process.

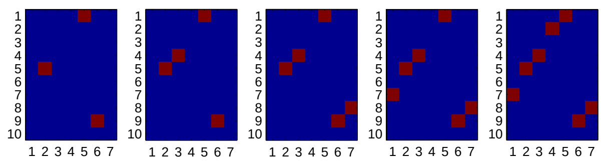


Fig. 12 Connectivity matrix showing correspondences of the winning particle $\hat{g}$ during the PF process.

7 Global Constraints to Solve Specific Correspondence Tasks

This Section shows how to define the global constraint matrices GC to model different tasks. In general GC depends on a single particle $g_{1:t}$ at a certain time t. GC is therefore in general defined as a function of $g_{1:t}$. This function is the 'GC update rule', modeling the feedback process , as mentioned in Figure 1.

The global constraint matrix defines the specific task. We will give examples and mention existing solutions to these tasks to illustrate how the PF system unifies these problems.

7.1 No Global Constraints

The most simple case. C is independent of $g_{1:t}$ and is defined by $c_{ij} = 1$. Since the problem is globally unconstrained, the particle $\hat{g}$ consisting of all possible correspondences solves the optimization problem eq.8.

7.2 One to One Correspondences

To guarantee one to one correspondences (in contrast to one to many), the probability of selecting a correspondence containing a point that is already part of an existing correspondence in particle $g_{1:t}$ must be set to 0. A matrix $GC(g_{1:t}) = [c_{ij}]$ defined by:

$$c_{ij} = 0 \iff (v_i \cup u_j) \cap \left(\bigcup_{i=1}^{|g_{1:t}|} g^i_{1:t} \right) \neq \emptyset, \ else \ c_{ij} = 1$$

guarantees the one to one constraint. Example (particle $g_{1:t}$ is represented by its connectivity matrix):

$$g_{1:t} = \begin{pmatrix} 0&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \end{pmatrix} GC(g_{1:t}) = \begin{pmatrix} 1&0&1&0&1 \\ 0&0&0&0&0 \\ 0&0&0&0&0 \\ 1&0&1&0&1 \end{pmatrix}$$

The problem of one-to-one correspondences without order preservation is classically solved using the Hungarian algorithm. An example for this problem is 'Shape Context Matching', as described in Belongie et al (2002). Since here no topological constraints are implied, the local feature descriptor must contain regional or neighborhood information. The 'Shape Context' descriptor fulfills this requirement. However, the implicit definition of regions in the shape-context descriptor makes the simple Hungarian approach unfeasible for partial matching, as explained in Lakaemper et al (2008).

7.3 Strong Contiguity Preservation

g is said to be **contiguity conserving** if, for a clockwise ordering of $(v_{i_1}, .., v_{i_k})$ in S_1, $(u_{j_1}, .., u_{j_k})$ is ordered either clockwise or anti-clockwise in S_2 (allowing both clockwise and anti-clockwise orderings in S_2 makes it possible to find correspondences in the reflected version of S_2). g is **strongly contiguity conserving**, if it is contiguity conserving and contains one to one correspondences only. The implementation of such a constraint using matrices is straightforward; we give an example below:

$$g_{1:t} = \begin{pmatrix} 1&0&0&0&0 \\ 0&0&0&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&0 \end{pmatrix} \Rightarrow GC(g_{1:t}) = \begin{pmatrix} 0&0&0&0&0 \\ 0&1&1&0&0 \\ 0&0&0&0&0 \\ 0&0&0&0&1 \end{pmatrix}$$

An example illustrating strong contiguity conserving grouping is in Figure 7. Note that in this case the topology of the matrix is important: if we deal with closed boundaries, GC must be interpreted periodic in rows and columns (torus).

This problem is classically solved using Dynamic Programming (DP), as in Scott and Nowak (2006). DP will find the optimal solution to eq.8, while PF is near optimal only. The experiment Section 8 will show that the loss of precision of PF is visually unimportant.

The PF approach unifies the former problems in a single framework, yet the specific solutions (Hungarian/DP) are superior as they lead to the optimal solution. PF shows its advantages in problems where more sophisticated global constraints are needed. One example is partial shape matching.

7.4 Adding Domain Knowledge for Partial Shape Matching

In all previous examples the definition of $GC(g_{1:t})$ was straightforward: the global constraint was a hard defined admissibility, resulting in an a priori known rule to set the elements of $GC(1 : g_t)$ to either 0 or 1. The following example differs in two aspects: first, the admissibility is defined in a soft way by probabilities in $[0, .., 1]$. Second, the update rule is not known a priori, but is stated as a hypothesis based on the grouping $g_{1:t}$ and weight matrix L. As the number of iterations increases, the hypothesis becomes stronger. This can be interpreted as *learning the constraint* during the PF process.

The global constraint derived here extends the global constraint of strong contiguity conservation. The constraint of strong contiguity conservation is sufficient to find good groupings for closed shape boundaries. However, it fails in finding part correspondences, see example Figure 13, left. Additional domain knowledge has to be implemented.

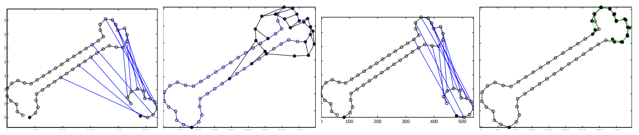


Fig. 13 Part correspondences using different constraints. From left to right: 1) strong contiguity only. 2) Procrustes alignment (see Section 8 based on (1). 3) windowed strong contiguity as introduced in Section 7.4. The window was learned during the PF process. 4) Procrustes alignment of (3)

In the following we assume that shape S_1 *entirely* contains part S_2. This is not a strong assumption, since if S_1 only contains parts of S_2, we can exchange both shapes and use an appropriate subset of S_2 as the part. This assumption is above and beyond the constraints used to compare entire (not parts of) shapes. However, we do not handle the case of correspondences where S_1 and S_2 have only partial overlap in *both* shapes. For purposes of simplification, we assume S_1, S_2 have the same scale and are sub-sampled equally. Ignoring these constraints is done only at the cost of a less intuitive constraint matrix, see Section 7.4.1. These assumptions jointly imply that the corresponding vertices in S_1 have to be 'close together', and not spread out all over S_1. In other words, we focus exclusively on a certain connected region (or window) of S_1.

Figure 13 shows the effect of this constraint. If it is not applied (left), only the center vertices of part S_2 are held together by the contiguity constraint, hence they lead to correct correspondences. The outer vertices of S_2 have the opportunity to correspond to wide regions of shape S_1, since contiguity is a local neighborhood constraint which is weaker for the (intuitively) weaker neighborhood-bound outer vertices. Focusing on a connected region in S_1 leads to the correct correspondences (right).

If the region of focus is known a priori (i.e. the position of the part in the shape is roughly given but the exact single correspondences are not explicitly known), the optimization process is simply a windowed version of entire shape matching. Unfortunately the window is not known. The strength of our approach is that it can estimate the window during the PF process by analysis of the particle given in each iteration. Below, we use the notation, 'diagonal matrices' to refer to matrices with elements on either the main or side-diagonals. To highlight a certain region $r = (v_r, v_{r+1}, .., v_{r+k})$ in S_1, correspondences with $v_i \in r$ have to be masked in L. Due to the equidistant sub sample and equal scale assumptions, the mask is a diagonal matrix. The position of the diagonal defines the allowed region. We use a soft mask, i.e. a gaussian blurred diagonal, see Figure 14. The kernel size, or standard deviation σ_t of the blurring defines the distinctiveness of the region. To learn the window during the PF process, we analyze the connectivity matrix of a particle $g_{1:t}$ at time t for the presence of diagonals. That is to say, we determine the most likely diagonal of its connectivity matrix, taking into account both the connectivity matrix of $g_{1:t}$ as well as the underlying correspondence weights of the matrix L. We gauss-filter the estimated diagonal to obtain a window $GC_{\sigma_t}(g_{1:t})$. The diagonal is a hypothesis for our focus of attention, namely, the region r. The strength of the hypothesis is modeled by the filter kernel size σ_t. A large σ_t defines a weak hypothesis (or indistinct region). Decreasing σ_t during the iterative process, increases the trust in the diagonal hypotheses at later iterations, since it narrows the diagonal constraint window $GC_{\sigma_t}(g_{1:t})$. The final global constraint matrix $GC(g_{1:t})$ is then given by:

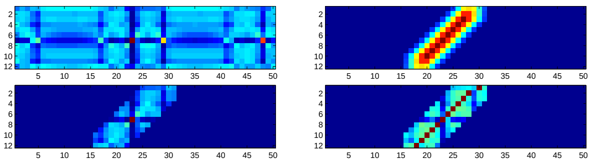


Fig. 14 Windowing for partial shape matching. In reading order: 1) correspondence matrix W of shape and part shown in Figure 13. 2) The final window, learned during the PF process. 3) Windowed version of (1). 4) The final connectivity matrix (superimposed over (3)). (4) corresponds to the grouping shown in Figure 13, right. All matrices are shown in transposed form.

$$GC(g_{1:t}) = GC_{\sigma_t}(g_{1:t}) \odot GC(g_{1:t}). \tag{15}$$

with $GC(g_{1:t})$ modeling the strong contiguity preservation as defined in Section 7.3. To ensure that the final particle $\hat{g}_{1:t}$ is entirely placed in the window $GC_{\sigma_{\hat{t}}}(\hat{g}_{1:t})$, the particle process is repeated: from the first PF process, we only use the final constraint matrix $GC_{\sigma_t}(\hat{g}_{1:t})$ of the winning particle. We then run a second entire particle filtering pass using $GC_{\sigma_t}(\hat{g}_{1:t})$ in each iteration (i.e. we finally run a windowed version of the optimization process). The second pass needs only a few particles since the window limits the search space drastically (in our tests we decreased from 50 particles in the window-learning pass to 8 particles in the windowed pass, which suggests that in the second pass we are close to utilizing a greedy process).

σ_t is determined experimentally. It depends on the size n (number of points) of the target shape and size m of the query part: starting with a value of $n + \sigma_{min}$, we decrease σ_t linearly to $\sigma_{t+1} \leftarrow \sigma_t - n/m$ in each iteration to a minimum of $\sigma_{min} = 3$ (motivation: starting with a wide standard deviation which nearly does not reflect any neighborhood relation, σ_t decreases down to $\sigma_{min} = 3$ in m steps; m is the number of iterations to find the m correspondences not taking into account 'recede'). Since this simple strategy led to satisfying results, no further research (e.g. with non-linear cooling strategies) was performed at this time.

Note that especially the interplay between $GC_{sigma_{\hat{t}}}(g_{1:t})$ and the recede-step already builds a near diagonal solution in the first run, since the domain knowledge gained in iteration t helps to rebuild destroyed connections appropriately inside of the window. With this approach, parts could be matched successfully. For results see Section 8.

7.4.1 Multi Scale Partial Matching

In the work presented in Lakaemper and Sobel (2008) we assumed the same scale for both shapes. We now explain how the system can be extended to operate on multiple scales: as an example, let us assume an entire shape S_1 with boundary length 1.0, uniformly sampled by 100 points, and a part S_2 of length 0.25, uniformly sampled by 25 points. If we assume identical scales, i.e. a scaling factor $sc = sc_2/sc_1$ of 1.0 (with $sc_{i=1,2}$ = scale of $S_{i=1,2}$), the connectivity matrix of the optimal grouping would be diagonal, i.e. describe a line of $\pi/4$ degrees in the matrix, a 'slope' (see Figure 15). Assuming a different scale changes the angle of the slope: the larger the scale of S_2, the more horizontal the line becomes, since now the 25 points of S_2 cover a smaller part of S_1; e.g. with a scaling factor of 2.0 between the 2 shapes the 25 points of S_2 would correspond to a region in S_1 covered by 12 points only. Simple geometry defines the angle α of the slope by $\alpha = atan\left(\frac{sc_2}{sc_1}\right)$. To gain scale independence, we analyze the current particle not only with respect to the probability of diagonals, but determine the probabilities of slopes in a certain range of angles. To cover a range of scale-ratio $\frac{sc_2}{sc_1} \in [0.5..2.0]$, we cover slope-angles in the range $\alpha \in [0.46..1.11]$.

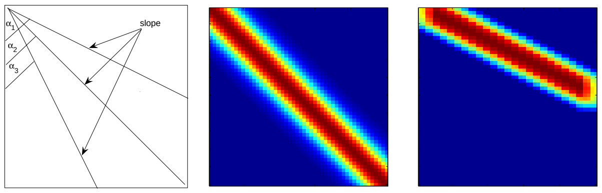


Fig. 15 'Slopes' as generalization of diagonals for scale independent matching. left: 3 slopes with different angles. Center: blurred slope for scaling factor $sc = 1.0$, Right: same as (center) for $sc = 2.0$

Figure 16 gives an illustrative example of the effect of multi scale matching. A part (constant size, 21 points), is matched to versions of a complete shape, sub sampled with a different number of points. The corresponding parts in the complete shapes therefore appear scaled relative to the 21-sample point query part (e.g., the corresponding part in the complete 100-point shape consists of 42 samples). Accordingly, the correspondence matrices W, top row of Figure 16, are of size 30×21, 50×21, 75×21, 100×21. Since they always contain the same matching pattern of complete shape vs. part (in different resolutions), the pattern becomes 'vertically stretched' using higher sample rates, which creates the slopes of different angles. The center row shows the final window of the winning particle, the bottom row the correspondences.

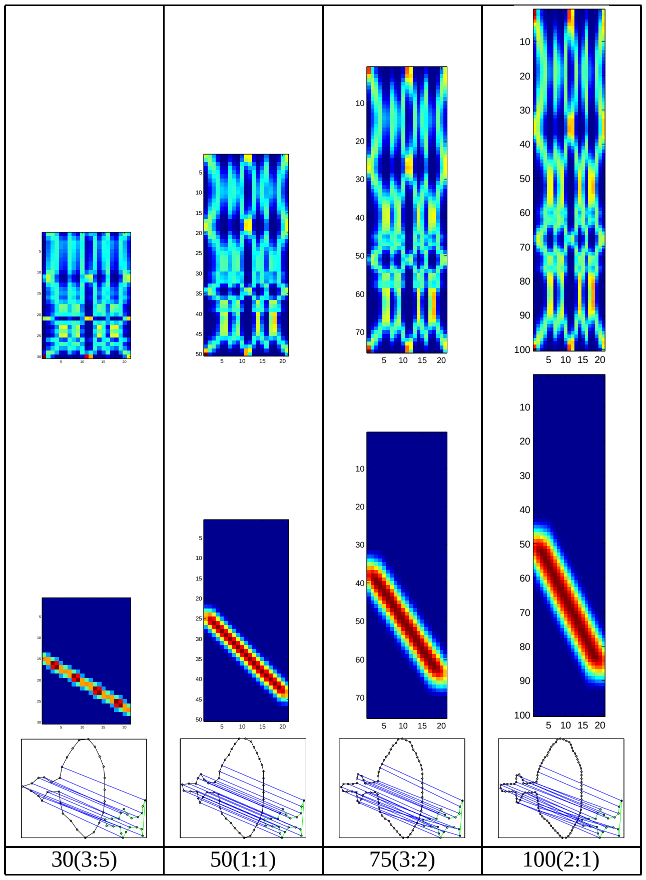


Fig. 16 Multi scale partial matching. The part consists of 21 points of a 50-point sub sampled 'fountain' shape. Columns left to right show comparison to complete 'fountain' shape, sub sampled with 30, 50, 75, 100 points respectively, resulting in scaling of 3:5, 1:1, 3:2, 2:1 (number of points in corresponding part : 21 samples). Top row: correspondence matrix W. The vertical 'stretching' effect, leading to different diagonal slopes, can clearly be seen. Center row: Final window. Bottom row: correspondences.

8 Experiments and Results

8.1 Performance of Particle Filtering

We evaluate the performance of the PF process in the optimization process with respect to different numbers of particles, with and without recede-step. Given a shape S defined by a $2d$ polygonal boundary with 50 equidistant vertices, we compute the correspondence matrix L of S with *itself* (i.e. $S_1 = S_2$) under strong contiguity conservation. This setting provides a ground truth optimum grouping $\hat{g} = \{\{v_1, v_1\}, .., \{v_{50}, v_{50}\}\}$. We perform different runs on the optimization process using 1 to 160 particles, the result of each run is the weight of the best particle. To see the influence of the recede step, we repeat the experiment skipping this step. To see the influence in the alignment resulting from finding a near optimal solution only, we compute the shape similarity based on the best particle correspondence using Procrustes Analysis (PA), Gower and Dijksterhuis (December 2005). PA aligns two shapes S_1, S_2 based on one to one correspondences. It finds the best scaling, reflection, rotation and translation of S_2 to minimize the sum of squared distances between corresponding points. This sum, the PA distance, is used in our experiment as the shape similarity measure. PA is extremely sensitive to outliers and can only be used with a robust correspondence computation. The usability of PA as similarity measure shows the stability of the PF correspondences. Figure 17 illustrates the result. The PF system with recede step computes weights near the optimum (1.45) robustly already with > 60 particles. Interesting is the fact that the Procrustes distance drops to 0 (perfect alignment) much faster, proving that the additional precision gained is visually unimportant, hence the precise optimum, as found by DP approaches, would not enhance the alignment here. The graph also shows the importance of the recede step: skipping it, the performance of the system is significantly less than optimal.

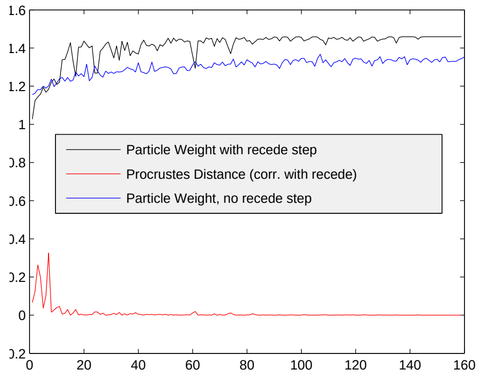


Fig. 17 Performance of PF system with different number (1-160) of particles (x-axis) with and without recede step. Black: with recede. Blue: no recede step. For black/blue graphs the y-axes shows the weight of the best particle $\hat{g}$. Red: Procrustes Distance of aligned shapes.

8.2 Correspondence of Complete Boundaries

This Section will show the behavior of the PF system on complete boundaries and will show examples of correspondences under the influence of noise, occlusion and sparse presence of features. We used shapes of the MPEG-7 data set, all boundaries are uniformly sub-sampled using 50 points and treated as closed boundaries. The global constraint is 'strong contiguity preservation', as explained in Section 7.3. For display reasons, the shapes are also shown after alignment, using a simple, basic form of Procrustes Alignment (PA) (Gower and Dijksterhuis (December 2005)). For display reasons (the optimal solution shows a connectivity matrix close to the diagonal) the shapes were index-aligned, i.e. the starting points in both shapes correspond visually to each other. This does not influence the experiments, since the PF system chooses the initial correspondences independent of the point indices.

Example 1 Matching of identical shapes (Figure 18).

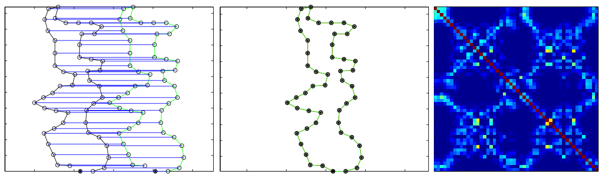


Fig. 18 Matching of a shape to itself. In reading order: a) correspondences b) PA alignment, based on (a), c) connectivity matrix (red dots) of correspondences superimposed over local constraint matrix L.

We matched shape 'carriage-01' to itself. The PF system delivered the perfect result, the connectivity matrix is diagonal, each point corresponds to its identical counterpart of the other shape. The reason is twofold: first, the matrix has a strong diagonal, suggesting these connections as the most probable ones to the PF system. Second, dominance of correspondence-rich particles lead here to the solution with maximal number of correspondences.

Example 2 Occlusion of Parts(Figure 19).

In this example we match similar shapes ('camel-01 vs. 'camel-02'), with a missing part in one shape (see Figure 19) (three legs vs. four legs). The PF systems does not find correspondences to that part, which is correct. The missing part can be detected by a 'jump' in the upper-left part of the connectivity matrix Figure 19,(c), following the higher correspondence probabilities, which are off the main diagonal.

Example 3 High Local Ambiguity (Figure 20).

The two shapes ('apple-01' vs. 'apple-06') in this example lack distinct features, i.e. most of the points are described

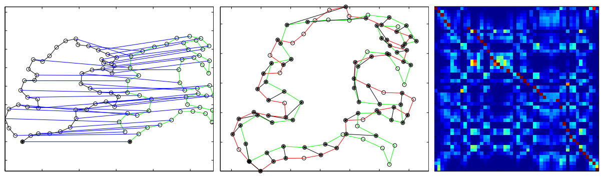


Fig. 19 Matching under occlusion. In reading order: a) shapes and correspondences. The green shape has one more leg. b) PA alignment c) connectivity matrix and local constraint matrix L

by similar values of LDS and CV. The local constraint matrix L therefore shows a huge area of similar correspondence probabilities for the majority of points, see Figure 20(c). This leads to high ambiguity, which can only be solved using the global constraints. The preference of correspondence-rich particles in cooperation with strong contiguity are the driving force to establish correct correspondences for ambiguous cases.

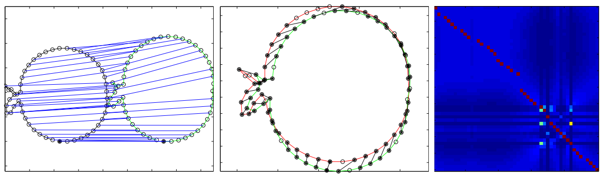


Fig. 20 High local ambiguity. In reading order: shapes (a) with few distinct features. b) PA alignment c) connectivity matrix and local constraint matrix L: the lack of features leads to a huge number of similar probability values (ambiguity)

Example 4 Noise (Figure 21).

Matching of a shape to a noisy version ('device0-01' vs. 'device0-13'). The challenge here is similar to the former experiment, yet based in the opposite reason: feature *richness*, mostly inferred by distinct values in LDS, leads to ambiguity: the 5 highly probable matches in each row (column) show the similarity to convex/concave extrema in the other shape. The PF system reacts robustly to this challenge. Observe that it is somewhat arbitrary which of the strong diagonals is chosen by the PF system, since the shape is highly symmetric and the matrix is interpreted periodically.

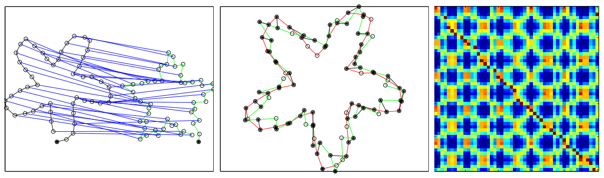


Fig. 21 Matching under noise. In reading order: a) shapes and correspondences, b) PA alignment, c) connectivity matrix and local constraint matrix L: here feature *richness* leads to high ambiguity.

8.3 Complete Matching of Articulated Shapes

This experiment demonstrates the applicability of our approach on the data set of articulated shapes from Ling and Jacobs (2007). The boundary was sub sampled using 50 points. As local feature descriptor, we used curvature only; 50 particles were utilized. See Figure 22 for results.

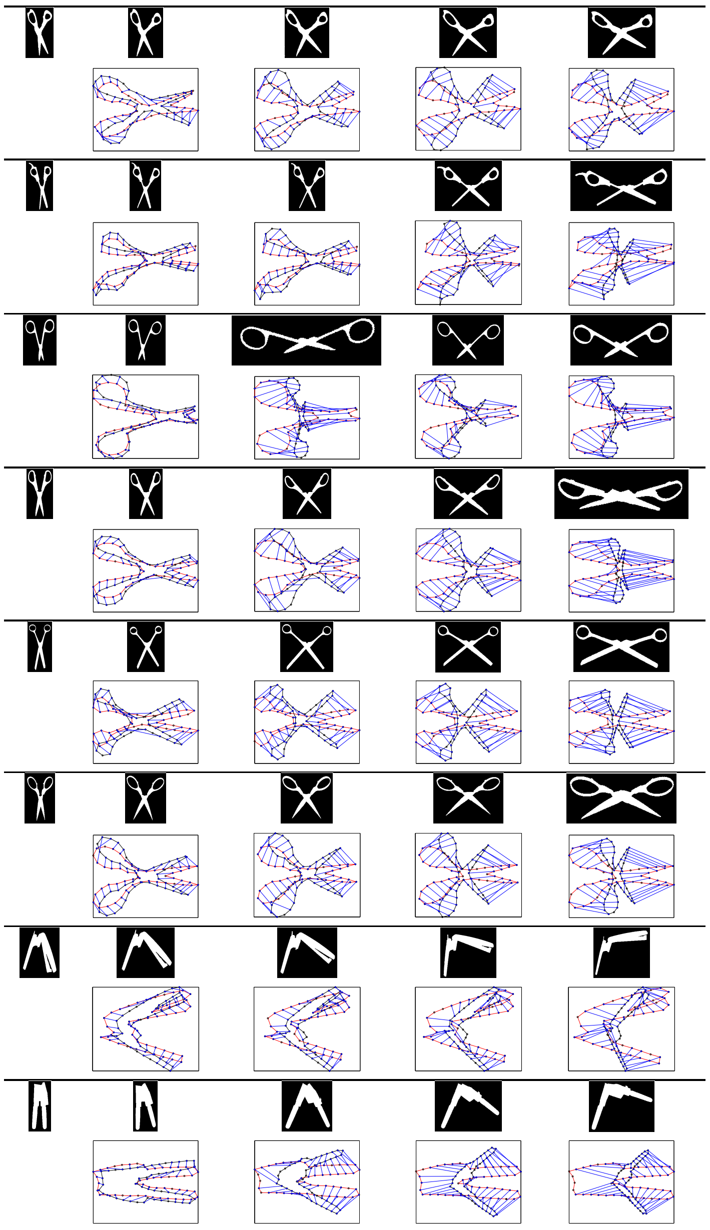


Fig. 22 Matching of Articulated Shapes. The image in the leftmost column is the reference shape for each row. It appears as the red shape in the correspondences.

8.4 Partial Correspondences

In this experiment we provide visual proof of the performance of partial matching. The data are shapes from the standard MPEG-7 similarity data set, see Latecki et al (2007). The data set provides 70 classes of 20 similar shapes each. We use the data set in the form of boundary polygons; each such polygon contains 50 vertices. We randomly select shape S_1 from the data and S_2 as a random *part* (size: 15 vertices) from a different shape in the same class. The PF process (50 particles, constraint: windowed strong contiguity as presented in Section 7.4) leads to correspondences which are used for Procrustes alignment. Figure 23 shows some typical examples of partial matchings.

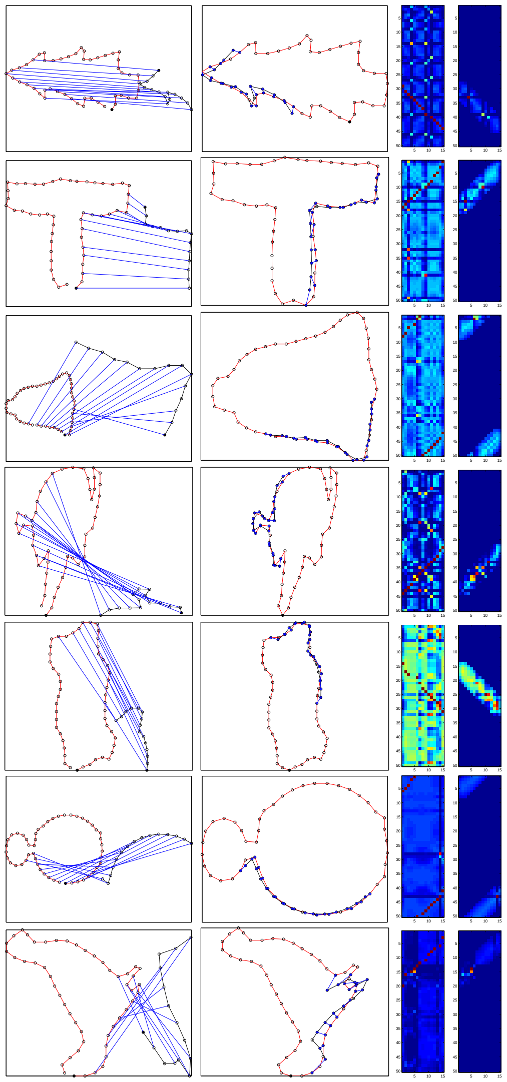


Fig. 23 Each row shows an example of partial matching, using different shapes from the MPEG-7 dataset. The parts are randomly chosen from different shapes of the same class as the full shape. Column 1: correspondence. Column 2: Procrustes alignment. Column 3/4: connectivity matrix and final, learned global constraint matrix, showing the window the correspondences were chosen from.

8.5 Database Retrieval using Parts

We compare our PF correspondence to part matching techniques based on time series algorithms described in Latecki et al (2007). As in Latecki et al (2007), we took 10 parts of shapes of different classes from the MPEG-7 data set, the parts being identical to the ones described in Latecki et al (2007). We then matched each part to the entire database (all 1400 shapes), based on the PF correspondence (PFPA) and Procrustes Alignment. To achieve a fair comparison, we used curvature as shape descriptor, as in Latecki et al (2007). Figure 24 shows the parts (left column) and the top 5 matches for each part. In nearly all cases (96%) the query resulted in shapes of the same class as the query part.

	PFPA	OSB	DTWCW	LCSS
Top 1	100	100	90	90
Top 5	96	92	72	42
Top 10	91	84	67	34
Top 20	74	67	59	26

Table 2 Retrieval Results of Partial Query, PFPA is the described approach.

Table 2 shows the intra class hit percentage for the top $1, 5, 10, 20$ results, i.e. how many of the top-n results are from the same class as the query part. PFPA is the described algorithm, OSB the algorithm described in Latecki et al (2007). DTWCW is a windowed version of Dynamic Time Warping, LCSS stands for Longest Common Sub Sequence. The percentages of OSB, DTWCW and LCSS are taken from Latecki et al (2007), details about these algorithms are described there, too. Due to our robust partial PF correspondence, even a simple similarity measure like PA outperforms the competing approaches.

8.6 Shape Classification Using Random Parts

Goal of this experiment is to classify (label) a query shape by partial similarity to shapes of a pre-labeled data base, using multiple random parts of the query shape. In short: to which class does the query shape belong? The target data base consists of different classes with multiple representatives. We use the entire MPEG-7 database, consisting of 70 classes, 20 members each.

The basic idea is to randomly select parts from the query shape, and to compare them to different randomly selected members of each of the 70 MPEG-classes. The underlying assumption for this experiment is, that randomly selected parts will in average emphasize common features of the query shape with a class of similar shapes. In contrast, shapes of non-matching classes will offer less fitting features. The advantage of partial matching over entire shape matching is,

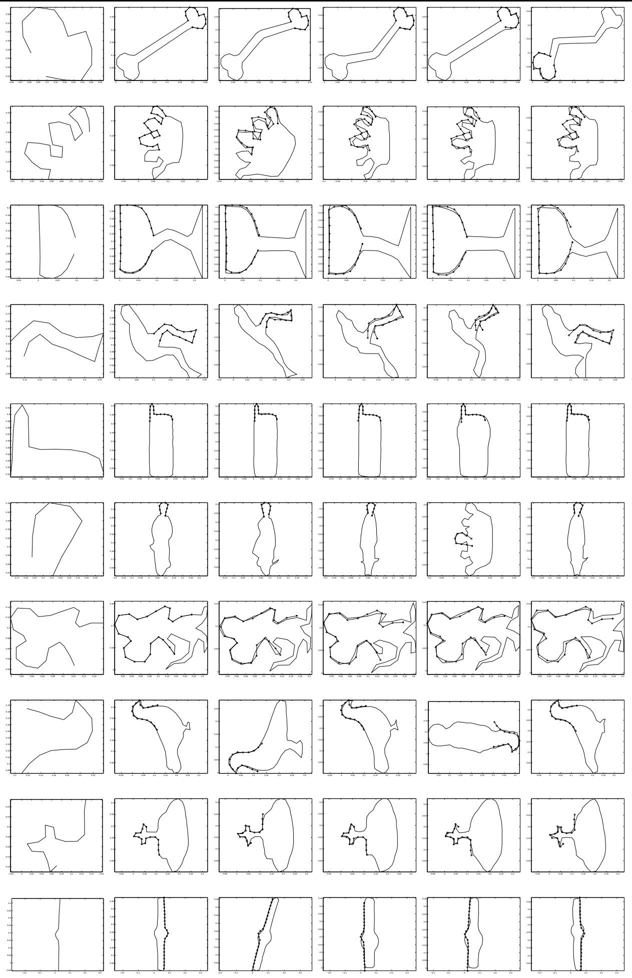


Fig. 24 Results of database query with parts. Left column is the query part, which is compared against the entire MPEG-7 data set (1400 shapes). Each row shows the respective top 5 hits.

that non rigid transformations are less influential as long as distinct shape features are still represented. Since the parts are randomly chosen, a higher number of part tests must be performed to ensure that characteristic shape features are captured. The best average fitness determines the classification.

Similar to the previous experiments, we determine the similarity using simple procrustes fitting based on the partial correspondences returned from the PF system. A correct partial correspondence is crucial for the success of such a system. However, since the parts are randomly selected they are frequently not significantly descriptive. To adjust for that, we refine the similarity measure in the following two ways: first, we do not use the two outmost (the two ends) part correspondences for procrustes matching. In certain cases, the PF system allows these correspondences to fit outside the 'optimal window'. Since procrustes matching is very sensitive to outliers, discarding the endpoints leads to a more robust fit. Second, we take into account the scaling factor of procrustes fitting. Procrustes analysis automatically scales the part for an optimal fit. With non descriptive parts, this can lead to a good geometric fit even with non matching parts. For example, with a simple straight line (query) and a target spiral (of same boundary length) the fit can be drastically improved when scaling the query part's size down to the spiral's outer radius; this improved fit is not intuitive and misleading. Denoting the value of procrustes least squares fit by P , and the procrustes scaling factor by s, we measure the fitness between a query part Q and a target shape T by $d_P(Q,T) = P\,max(s, 1/s)$ ($d_P(Q,T) = 0 \Rightarrow$ best possible fit). Hence, we introduce a penalty for the procrustes scaling. Please note that this factor introduces the assumption that the part and object data is of similar scale. In this experiment we need this assumption, since we do not rely on significant parts, but use random parts. Scale independent shape of such parts would be too general and would introduce many false positive matching results.

We sub-sample every shape of the MPEG-7 data base using 50 uniformly spaced boundary points. The query part size is fixed to 15 points. We use the first shape in each class as query shape, i.e. the experiment classifies 70 shapes. Each of these 70 shapes is compared against randomly chosen shapes of the *entire* MPEG-7 data base. A simple greedy algorithm to weed out non-promising candidate classes is used to shorten the computation time: starting with the 5th iteration, we do not match the query part against members of classes with an average fitness higher than 4 times the currently best fitting class (see 'if' statement in the following algorithm). For each query shape S, we iterate through the following classification process, using a maximum of $partMax = 20$ parts:

```
input = shape $S$
$count = 1$; $partMax = 20$; $numberOfClasses = 70$
average class fitness $acf(1 : numberOfClasses) = 0$
**while** $count < partMax$ **do**
    select random query part $Q$ from input shape $S$
    **for** c=1:numberOfClasses **do**
        **if** $(count < 5) \mid ((count > 5) \,\&\, (acf(c) < 4min(acf)))$
        **then**
            randomly select target shape $T$ from 20 members of class $c$
            compute correspondence $Q$,$T$
            compute procrustes fitness $d_P(Q,T)$
            update $acf(c)$ using $d_P$
        **end if**
    **end for**
    $count \leftarrow count + 1$
**end while**
class of shape $S$ is $argmin(acf(1 : 70))$
```

Figure 25 shows a typical example. It shows the first 4 iterations of the algorithm using the query shape 'comma'. In the first iteration, the single part was not distinct enough to place the 'comma' class in the top 5 results. Iterations $2-4$ all show good matches to (different) members of the 'comma'-class, the average class fitness improves relative to the average fitness of other classes: although single 'comma'-parts might fit better to non-'comma' shapes, in average the matches to the appropriate class prevails to eventually place the 'comma'-class at rank 1 (staying at rank one for the remaining iterations).

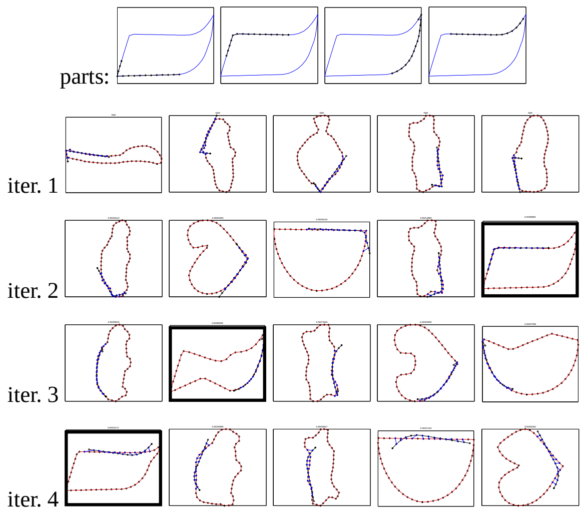


Fig. 25 Four iterations of classification of shape 'comma', Top row: the shape and four randomly selected parts. Rows labeled '1-4': top five results of classification comparing the respective part to randomly selected representatives of all 70 MPEG classes. Using only the first part, the comma does not appear in the top 5 classes. After 4 iterations, the consistent good match with (different) members of class 'comma' averages out the matches in other classes. Class 'comma' stays on rank one for the following iterations (not shown).

Figure 26 shows the result of the experiment. For each of the 70 query shapes, the rank of its correct classification is shown. In general, the classification works well, except for certain problem cases which we will discuss below. The problem cases are the shapes of the 'device' classes in the MPEG-set (classes 23-32). *Excluding all 10 device classes*, table 3 shows the results: 91.67 shapes are correctly classified, no correct classification ranks worse than rank 4 (with only a single class ranked 3, and 4 classes ranked 2). Note that this experiment is different to the popular MPEG-7 *bull's eye* test.

Correct classification rank	number query shapes	percent
1	55	91.67
2	4	6.67
3	1	1.67
total:	60	100

Table 3 Table of correct classification ranking of MPEG-7 set *excluding the 'device' classes*

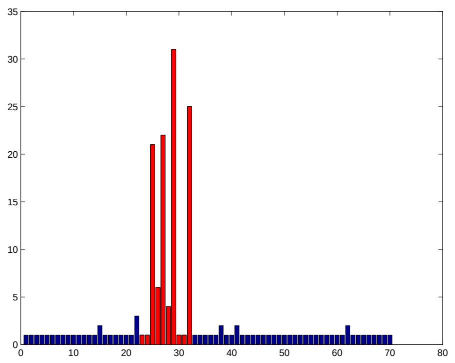


Fig. 26 Ranking (y-axis) of correct classification for the 70 query shapes (index of class: x-axis). Except for the 'device' classes (red, classes 23-32), the classification based on part-correspondence yields impressive results. Please compare table 3. See text for discussion about the 'device'-classes.

8.6.1 Excluding the 'Device' Classes: Explanation

If we look at the boundary representation, the 10 'device' classes in the MPEG-data set are very critical: for certain classes, the intra-class shape variance (intuitive) is very high, such that especially for boundary *parts*, matching to non-device classes is more likely. Additionally, these shapes are highly symmetric and do not add any further information, such that the randomly selected parts from the query shape are all very similar (device0/1/2 are stars with different numbers of rays. Parts of such shapes are not distinguishable, these shapes only differ *globally*). Having a number of similar parts for the query, the algorithm design resembles comparison with a single part. Together with high intra shape variance, this explains why certain device classes are wrongly classified. This explains the insufficient performance of the classification of classes 25, 26 (device2, device3).

In certain cases, the boundary information of such device classes can not lead to any partial similarity at all, see e.g. Figure 27. The shapes in a single class have no common partial boundary features; the boundary of members of the same class is extremely distorted: a partial boundary based approach must fail in these cases. This explains the incorrect classification of classes 27, 28, 29, 32 (device4, device6, device9). The remaining device classes have perfect classification results (classes 23, 24, 30, 31), however, the aforementioned arguments partially hold for these classes, too. We therefore decided to eliminate all device classes from the result statistics of table 3.

8.7 Experiments on Scalability

The time complexity of the algorithm is determined by the number of iterations and the complexity of the prediction

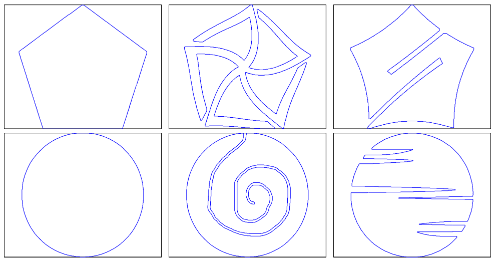


Fig. 27 Certain classes in the MPEG-7 set contain extreme boundary distortions. These classes only have a common *global* appearance, while, looking at boundary parts, the intra class similarity is very low. Such classes are hard to use for boundary part based classification. This holds especially for most of the 'device' classes, with the examples of 'device6' (top row) and 'device9' (bottom row) shown here. *We excluded the results on all device classes (shown in red in Figure 26) in table 3*

step for each particle [1]. The latter one depends on the complexity to build the constraint matrix. In the case of partial matching the constraint matrix can be determined in $O(n^2)$. The number of iterations is n, since we aim for completeness. Hence, in this case, the time complexity is $O(kn^3)$.

In our current MATLAB implementation, matching of two full boundary shapes with 50 points each takes about 2 seconds on a typical PC desktop (runtime for the following examples: computation of 500 points, 50 particles: 15 seconds; 800 points 200 particles: 90 seconds).

Apart from runtime complexity, we examined the performance of the algorithm using different sampling rates. In digital objects, 'local' descriptors like curvature are actually 'regional' descriptors. The underlying neighborhood size is determined by the sample rate (if sub sampled uniformly). A higher sample rate therefore decreases the regional information in each sample point. This generates local correspondence matrices containing a high number of points with similar correspondence probability, see Figure 28. The peaks in the higher resolution version are more distinct. However, the number of ambiguous points (dark blue) increases in each row about linearly (a row gives, up to a constant factor of proportionality, the correspondence probability of a single point in shape 1 to all points in shape 2). This reflects the significant growth of the set of possible matching candidates in each augmenting particle update especially in later iterations, when the peak-correspondences are already established. Assuming a constant number of particles, the system becomes more likely to be caught in a local optimum, the more points the data set contains. While the results in a certain range (20-500) of data points with a relatively low number of particles (50) are satisfying in this experiment (see Figure 29, a labeling task using 800 points could be handled increasing the number of particles to 200, see Figure 30. We leave the strategy to determine the number of particles to later research.

[1] the constant factor depends on the number k of particles and number of correspondences destroyed in the recede step

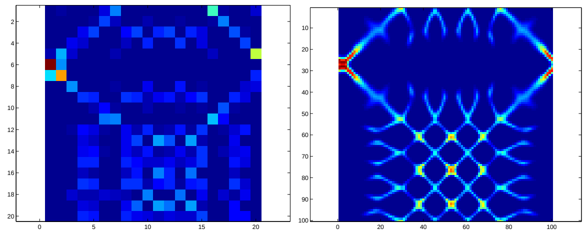


Fig. 28 Correspondence matrix of forks (see Figure 29) Left: both forks containing 20 points. Right: both forks containing 100 points. See text for further explanations.

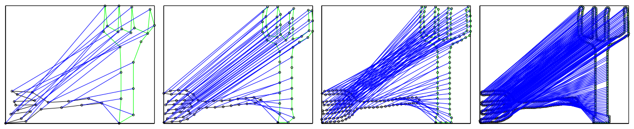


Fig. 29 Correspondences between shapes 'fork19' and 'fork20' of the MPEG-7 data set, uniformly sub sampled with (from left to right) 20, 50, 100, 500 points. The correspondences were computed using a constant number of 50 particles.

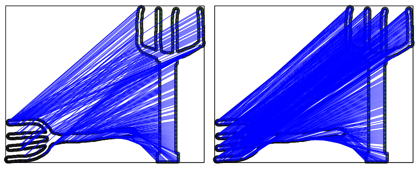


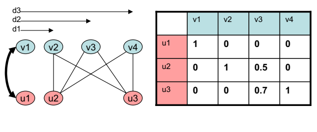
Fig. 30 Correspondences using 50 and 200 particles using a sample rate of 800 points. While the low number of 50 particles was insufficient (left), an increased number of particles (right) led to correct correspondences.

Please note that this example demonstrated the influence of lowering the significance of local features. In this case, the global constraint of order conservation becomes increasingly important. However, this simple global constraint is too weak to be carried by a low number of particles. The weakness of the global constraint had to be counter balanced using a higher number of particles. The next example will show how the same PF framework, applied to partial matching of 2D point clouds, requires significantly less particles if the global constraint is defined in a different way.

8.8 Partial Matching of Point Clouds

In this example, we leave the area of boundary comparison and head towards a more versatile data representation, point clouds. The versatility allows to represent shapes with inner structures, or, more general, of arbitrary topology, see Figure 32. The drawback of this representation is that no canonical point order information is given. Hence, the previously derived global constraint of order conservation can not be applied here. We define the global constraints on point clouds by means of neighborhood consistency. Please observe, that the PF approach for this significantly different task only differs in the definition of the GC-update rule (and, naturally, the pre-processing of local features). We intend to illustrate the scalability and extandability of our PF approach with this example, which was previously presented in Lakaemper et al (2008). The underlying definitions of local and global constraints can be found in more detail in Lakaemper et al (2008).

Consistent neighborhoods are defined in terms of distance consistency of shape points, to points participating in already established connections. Figure 31 illustrates the motivation: v_i and u_j are points of two shapes S_1, S_2 in $\mathbb{R}^2$. $d_{1,2,3}$ denote the (Euclidean) distances between v_1 and $v_{2,3,4}$ as well as the respective distances between u_1 and $u_{2,3}$. We assume an established correspondence (v_1, u_1), and call v_1 and u_1 the 'seed points' for the update. The update values $GC(v_i, u_j)$ are computed using the difference between their distances to the seed points. Only those point pairs (v_i', u_j') should be assigned a low value (close to 0.0), where a) at least one point of $\{v_i, u_j\}$ is close to its seed, and b) the distances of v_i and u_j to their respective seeds is different. All other pairs should be assigned a value closer to 1.0. The table fig.31, right shows example update values $GC(v_i, u_j)$ in accord with the motivation: $GC(v_2, u_2) = 1$ since $d1 = d(v_2, v_1) = d(u_2, u_1)$ and both distances $d(.,.)$ are relatively small. $GC(v_4, u_3) = 1$ for a different reason: both distances are large ($= d_3$), hence no statement can be inferred. $GC(v_3, u_2) < GC(v_3, u_3)$ although $|d_2 - d_1| = |d_2 - d_3|$ since $min(d_1, d_2) < min(d_2, d_3)$: a correspondence weight between two points which are both relatively far away from the established points should be less (value closer to 1.0) influenced. The 0.0 values for $(v_1, \cdot)$ and $(\cdot, u_1)$ guarantee one to one correspondences. For further details on the exact definition of the GC update rule, please see Lakaemper et al (2008).



	v1	v2	v3	v4
u1	1	0	0	0
u2	0	1	0.5	0
u3	0	0	0.7	1

Fig. 31 Updating weights based on a single established correspondence and different distances, see text for explanation.

We used a radius-limited version of shape context Belongie et al (2002) as local descriptor. In Belongie et al (2002), shape context is used for shape matching of complete shapes. For this purpose, the radius is automatically determined, capturing global shape information. The quality of this descriptor is demonstrated by impressive results using simple matching based on the hungarian algorithm. However, for partial matching this approach is not applicable: to capture regional shape properties only, the radius for the shape context computation has to be decreased, leading to failure in the hungarian matching, see Figure 33, left column (max. radius: 0.2). Our experiment uses a fixed radius shape context (max. radius: 0.2, compared to a shape diameter of about 1.0), but replaces the hungarian matching with our PF approach. Therefore, in contrast to the hungarian algorithm, which does not respect any global consistency constraints, our re-defined GC-update rule adds global consistency to the local/regional shape context feature. The improved results can be seen in Figure 33.

In contrast to the previous (fork) experiment, where the local descriptor became weak due to a high sampling rate, in this case the local descriptor is strong enough to support the interplay with the global descriptor sufficiently. The experiments were conducted with a very low number of particles: only 10 particles were required. The data sets contain 200 points each.

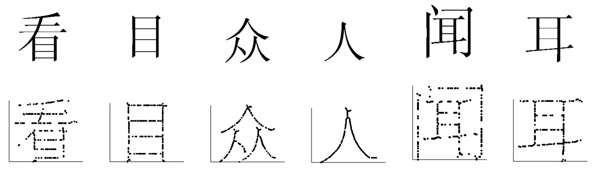


Fig. 32 Top row: chinese words. bottom row: Our data consists of 200 random samples from a skeletonized version of the top row. We will find the partial match between (top row in reading order) a,b; c,d; e,f.

9 Conclusion and Outlook

We presented a Particle Filter framework to solve the correspondence problem. The performance was demonstrated on partial shape matching of $2d$ polygonal boundaries, which was solved through a windowing approach. The window was learned during the iterative Particle Filter process. Further examples demonstrated the extendability to a more versatile data representation, point clouds. Though the examples of this paper were restricted to $2d$data, the PF framework can handle arbitrary dimensionality. Future work will therefore focus on $3D$ shapes. $3D$ shapes pose a special problem, since the number of data points is in general much higher. In order to tackle this problem, an appropriate GC-update rule will be specified. Additionally, the recede step, which currently chooses deletion candidates from a uniform distribution, will be extended.

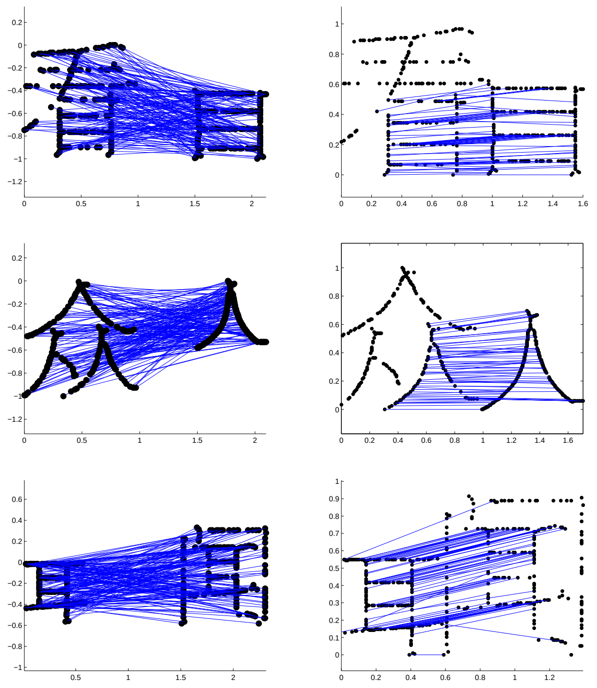


Fig. 33 Partial matching of data in fig.32 based on local descriptor 'shape context'. Left column: approach of Belongie et al (2002), using the Hungarian method. Right column: our PF approach.

10 Acknowledgements

Rolf Lakaemper was partly funded by grants NSF IIS-0534929 and Dept. of Energy DE-FG52-27508.

References

Alt H, Scharf L, Scholz S (2006) Probabilistic matching of sets of polygonal curves. In: Proceedings of the 22nd European Workshop on Computational Geometry (EWCG), Delphi, Greece, pp 107–110

Back T, Hammel U, Schwefel HP (1997) Evolutionary computation: Comments on the history and current state. IEEE Transactions on Evolutionary Computation 1:3–17

Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape contexts. IEEE Trans Pattern Analysis and Machine Intelligence 24:705–522

de Bruijne M, Nielsen M (2004) Image segmentation by shape particle filtering. In: ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 3, IEEE Computer Society, Washington, DC, USA, pp 722–725

Chen L, Feris R, Turk M (2008) Efficient partial shape matching using smith-waterman algorithm. Computer Vision and Pattern Recognition Workshops, 2008 CVPRW '08 IEEE

Crisan D, Doucet A (2002) A survey of convergence results on particle filtering methods for practitioners. IEEE Transactions on Signal Processing, Vol 50, No 3, March 2002

DelMoral P, Doucet A, Jasra A (2007) Sequential monte carlo for bayesian computation. Bayesian Statistics 8, pp1-34, Oxford University Press

Doucet A, De Freitas N, Gordon N (eds) (2001) Sequential Monte Carlo methods in practice. Springer, New York

Doucet A, DelMoral P, Jasra A (2006) Sequential monte carlo samplers. Journal of the Royal Statistics Society B, Vol 68, No 3, pp411-436

Goldberg DE (1989) Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional

Gorelick L, Galun M, Brandt A (2006) Shape representation and classification using the poisson equation. IEEE Trans Pattern Anal Mach Intell 28(12):1991–2005

Gower J, Dijksterhuis G (December 2005) Procrustes problems, vol 70. Springer New York

Haralick RM, Shapiro LG (1979) The consistent labeling problem: Part i. IEEE Trans Pattern Anal Machine Intell PAMI-1(2):173–184

Holland JH (1975) Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. The MIT Press

Kittler J, Illingworth J (1986) Relaxation labelling algorithms-a review. Image Vision Comput 3(4):206–216

Lakaemper R, Sobel M (2008) Correspondences between parts of shapes with particle filters. IEEE Int Conf on Computer Vision and Pattern Recognition (CVPR)

Lakaemper R, Li S, Sobel M (2008) Partial correspondences of point sets using particle filters (accepted, to appear). IAPR Int Conf on Pattern Recognition (ICPR)

Latecki LJ, Lakaemper R, Eckhardt U (2000) Shape descriptors for non-rigid shapes with a single closed contour. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp 424–429

Latecki LJ, Wang Q, Koknar-Tezel S, Megalooikonomou V (2007) Optimal subsequence bijection. IEEE Int Conf on Data Mining (ICDM)

Ling H, Jacobs DW (2007) Shape classification using the inner-distance. IEEE Trans Pattern Anal Mach Intell 29(2):286–299

Liu J, Chen R, Logvinenko T (2000) A theoretical framework for sequential importance sampling and resampling. technical report, stanford university, department of statistics, january 2000.

Liu JS (2002) Monte Carlo Strategies in Scientific Computing. Springer

Milios E, Petrakis E (2000) Shape retrieval based on dynamic programming. Trans Image Proc 9(1):141–146

Mokhtarian F, Abbasi S, Kittler J (1996) Robust and efficient shape indexing through curvature scale space. In: In Proceedings of British Machine Vision Conference, pp 53–62

Mokhtarian F, Khalili N, Yuen P (2002) Estimation of error in curvature computation on multi-scale free-form surfaces. Int J Comput Vision 48(2):131–149

Moral PD, Doucet A, Jasra A (2006) Sequential monte carlo samplers. Journal Of The Royal Statistical Society Series B 68(3):411–436

Perez P, Blake A, Gangnet M (2001) Jetstream: Probabilistic contour extraction with particles. In: Proc. of ICCV, pp 424–531

Qian W, Titterington DM (1992) Stochastic relaxations and em algorithms for markov random fields. Journal of Statistical Computing and Simulation 40

Rathi Y, Vaswani N, Tannenbaum A (2007) A generic framework for tracking using particle filter with dynamic shape prior. IP 16(5):1370–1382

Rosenfeld A, Hummel RA, Zucker SW (1976) Scene labeling by relaxation operations. IEEE Trans Syst, Man, Cybern SMC-6(6):420–433

Sandhu R, Dambreville S, Tannenbaum A (2008) Particle filtering for registration of 2d and 3d point sets with stochastic dynamics

Schmidt F, Farin D, Cremers D (2007) Fast matching of planar shapes in sub-cubic runtime. In: IEEE International Conference on Computer Vision (ICCV)

Scott C, Nowak R (2006) Robust contour matching via the order-preserving assignment problem. IEEE Transactions on Image Processing 15(7):1831–1838

Sebastian T, Klein P, Kimia B (2003) On aligning curves. PAMI 25(1):116–125

Siddiqi K, Shokoufandeh A, Dickinson SJ, Zucker SW (1999) Shock graphs and shape matching. Int J Comput Vision 35(1):13–32

Thrun S (2002) Particle filters in robotics. In: Proceedings of the 17th Annual Conference on Uncertainty in AI (UAI)

Veltkamp RC, Hagedoorn M (2001) State-of-the-art in shape matching. Tech. rep., Principles of Visual Information Retrieval

Yefeng Zheng, David Doermann (2006) Robust Point Matching for Nonrigid Shapes By Preserving Local Neighborhood Structures. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(4):643–649